Packet Architects AB

Ethernet Switch/Router
IPSec/MACSec 9x10G + 2x40G
User Guide

Core Revision unknown

Datasheet Revision unknown

Contents

List of Figures

List of Tables

Chapter 1

Overview

This L2/L3 Ethernet Switching/Routing Core offers full wire-speed on all 11 ports. Each port has 8 egress queues which are controlled by a multi-level scheduler.

The core is built around a shared buffer memory architecture capable of simultaneous wire-speed switching on all ports without head of line blocking. Packets are stored in the shared buffer memory as fixed size cells of 192 bytes. In total the buffer memory has a capacity of 1024 cells.

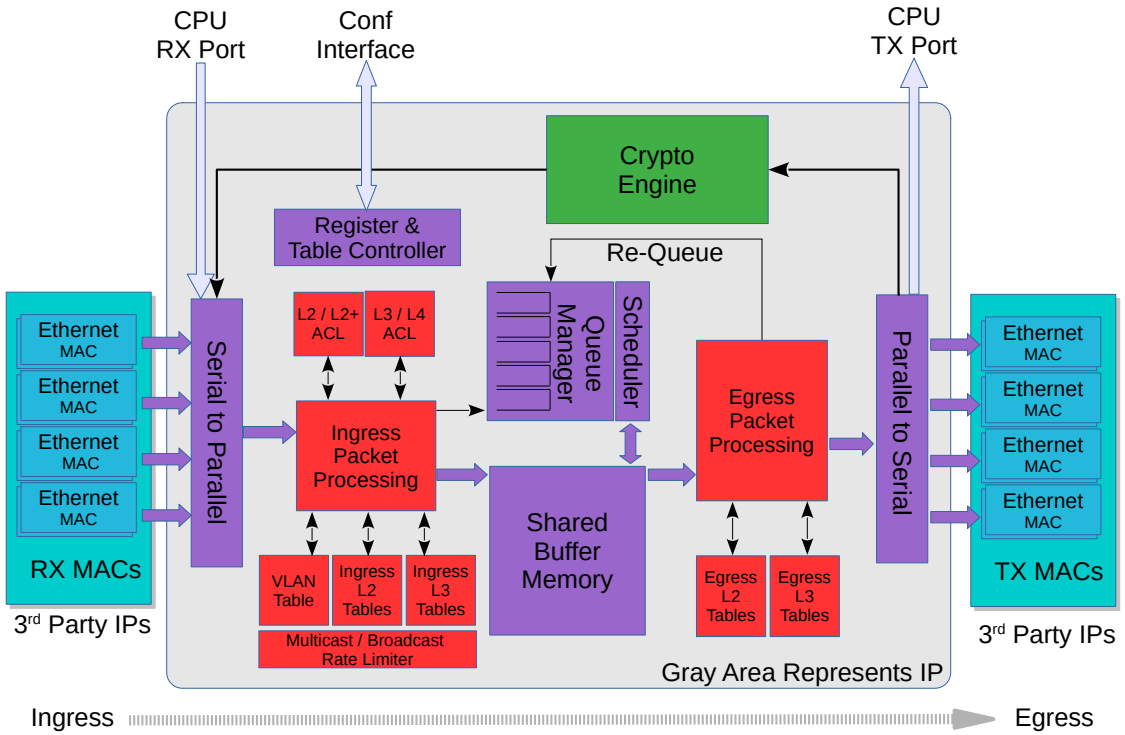


Figure 1.1: Switch Core Overview

Configuring tables and registers are done through a Configuration interface. However it is not required to perform any configuration. The core is ready to receive and forward Ethernet frames once the reset sequence has been completed.

1.1 Feature Overview

- 9 ports of 10 Gigabit Ethernet.
- 2 ports of 40 Gigabit Ethernet.
- Full wire-speed on all ports and all Ethernet frame sizes.
- Store and forward shared memory architecture.
- Support for jumbo packets up to 32733 bytes.
- Passes maximum overlap mesh test (RFC2899) excluding the CPU port, for all packet sizes up to 1601 bytes.
- Queue management operations:
  - Disable scheduling of packets on a port.
  - Disable queuing new packets to a port.
  - Allow a port to be drained without sending out packets.
  - Allow checking if a port is empty or not.
- Input and output mirroring.
- 4 source MAC address ranges with a number of different actions.
- 4 destination MAC address ranges with a number of different actions.
- 16,384 entry L2 MAC table, hash based 8-way.
- 4,096 entry VLAN table.
- 32 entry synthesized CAM to solve hash collisions.
- 4 entries of the synthesized CAM are fully maskable.
- 512 entry L2 multicast table.
- Automatic aging and wire-speed learning of L2 addresses. Does not require any CPU/software intervention.
- Spanning tree support, ingress and egress checks.
- 16 multiple spanning trees, ingress and egress checks.
- Allows software to inject special packets which are used to write into MAC tables while hardware learning engine is running.
- Allows software to track which L2 MAC entries are being learned and port moved.
- Allows software to track which L2 MAC entries are being aged out.
- Egress VLAN translation table allowing unique VID-to-VID translation per egress port.
- VLAN priority tag can bypass VLAN processing and be popped on egress.
- MPLS forwarding with support for swap,push,pop and penultimate pop operations.
- 4 entry VRF table.
- 16,384 * 4 hash based L3 routing table.
- 32 entry L3 routing TCAM.
- 2,048 entry next hop table. Pointed to from the routing entries.
- 2,048 entry packet modification table used by the next hop table to determine how build l2 fields in a packet to find the next hop.
- Configurable ECMP support based on L3 protocol field,L3 Tos, and L4 SP/DP.

- ECMP supports with up to 64 paths.
- 8,192 number of Ingress Network Address Translation (NAT) entries.
- 8,192 number of Egress Network Address Translation (NAT) entries.
- 10184 entries of ingress classification / ACL Lookups. The classification / ACL keys are configurable for each source port and the fields are selected from a incoming packets L2, L3 or L4 fields. The selection is described in 15.2 The classificaiton / ACL key can be up to 560 bits long. The classification / ACL lookup is based on a combination of hash and TCAM. The actions which can be done is listed below:
  - Multiple actions can be assigned to each result. All results can be done in parallel if the user so wishes.
  - Result action can be to drop a packet.
  - Result action can be to send a packet to the CPU port.
  - Result action can be to send a packet to a specific port.
  - Result action can be to update a counter. There are 64 counters which can be used by the classification / ACL engine.
  - Result action can be to force packet to a specific queue on a egress port.
  - Result action can be to assign a meter/market/policer to measure the packet bandwidth.
  - Result action can be to assign a color to the packet which is used by the meter/marker/policer.
  - Result action can be to force the packet to use a specific VID when doing the VLAN table lookup.
  - Result action can be to do a input mirror on a packet.
  - Result action can be to not allow the packet to be learned in L2 MAC table.
- The ingress configurable classification / ACL engine can use the type and code fields from ICMP frames.
- The ingress configurable classification / ACL engine can use the fields, including the group address, from IGMP frames.
- 9232 entries of egress classification / ACL rules. The classification / ACL keys are configurable based on what forwarding actions has been done and the fields are selected from the incoming packets L2, L3 or L4 fields and from forwarding results. The selection is described in 15.4 The ACL key can be up to 135 bits long. For each field there are options to only select part of the bits in a field. The ACL lookup is based on a combination of hash and TCAM. The actions are listed below:
  - Multiple actions can be assigned to each result. All results can be done in parallel if the user so wishes.
  - Result action can be to drop a packet.
  - Result action can be to send a packet to the CPU port.
  - Result action can be to update a counter. There are 64 counters which can be used by the classification / ACL engine.
  - Result action can be to force packet to a specific queue on a egress port.
- The egress configurable classification / ACL engine can use the type and code fields from ICMP frames.
- The egress configurable classification / ACL engine can use the fields, including the group address, from IGMP frames.
- 1572864 bits shared packet buffer memory for all ports divided into 1024 cells each of 192 bytes size
- 8 priority queues per egress port.

- Configurable mapping of egress queue from IP TOS, MPLS exp/tc or VLAN PCP bits.
- 64 ingress admission control entries.
- Deficit Weighted Round Robin Scheduler.
- Bandwidth shapers per port.
- Individual bandwidth shapers for each priority on each port.
- Individual bandwidth shapers for each queue on each port.
- Egress queue resource limiter/guarantee with four sets of configurations.
- Configuration interface for accessing configuration and status registers/tables.
- Multicast/Broadcast storm control with separate token buckets for flooding, broadcast and multicast packets.
- Multicast/Broadcast storm control is either packet or byte-based, configurable per egress port.
- LLDP frames can optionally be sent to the CPU.
- IEEE 1588 / PTP support for 1-step and 2-step Ordinary Clock mode. The switch supports transfer of 8 byte timestamp from receive MAC to software and form software to transmit MAC.
- The packets which are sent to the CPU can contain extra sw-defined "meta-data" which software sets up. Meta-data is 2 bytes and can come from a number of different tables.
- Wirespeed tunnel exit and tunnel entry. No looping of packets is needed.
- Tunnel unit for both tunnel entry and tunnel exit. Tunnel exit can be done in the beginning of the packet processing or after normal L2, L3, ACL lookups. The tunnel exit can be done on known fields or by looking up bytes anywhere in the first cell of the packet. Tunnel entry can be done as a result from the normal L2,L3, ACL processing.
- The tunnel exit allows packet headers/bytes to removed and certain information to be copied from the original packet to new tunnel exited packet. Once a tunnel exit has been done the new tunnel exited packet will be processed as normal packet at wirespeed.
- The tunnel entry allows packet headers/bytes to be added and certain information from the previous packet to be copied to the new tunnel headers. The tunnel entry is reached from normal L2,L3 and ACL processing and happens just before the packet is sent out allowing the inner packet to do full switching and routing.
- A crypt unit enables support for IPsec with both AH , ESP and ESP tunneling support. Includes many crypto modes and authentication modes for both encryption, decryption and authentication.
- The core also supports MACsec encryption and decryption. The MACsec support allows VLANs to be located both before and after the MACsec header.
- Table syncronization mechanism using versioning of functionality divided into multiple tables. See chapter 26.

A Packets Way Through The Core

This section describes the path of a packet through the core from reception to transmission, i.e from the RX MAC bus to the TX MAC bus. See Figure 1.1.

1. A packet is received on the RX MAC bus with a *start of packet* signal.
2. Ingress port counters are updated.
3. The asynchronous ingress FIFO synchronizes the incoming data from the data rate of the MAC clock to the data rate of the core clock.
4. The serial to parallel converter accumulates 192 bytes to build a cell, and the cell is sent to ingress processing, if a packet consists of more than 192 bytes then a new cell is built. This is repeated until the *end of packet* signal is asserted.
5. Ingress processing (see chapter 3.1) determines the destination port (or ports) and egress queue of the packet. It then decides whether the packet shall be queued or dropped. Many different tables and registers are used in the process to determine the final portmask and final egress queue for the packet.
6. If the packet matches a certain traffic type whose bandwidth is monitored by the core, it will be pointed to one of the 64 meter-marker-droppers to do the rate measurement. The result may drop the packet or change the packet color.
7. Packets are never modified before they are written into the buffer memory. Rather an ingress to egress header (I2E header) is appended to the packet. Any modifications are done in the egress packet processing pipeline, based on the I2E header.
8. Unless the packet is dropped, the packet is written cell-by-cell into the buffer memory with the I2E header appended.
9. The buffer memory has enough read and write performance for any traffic scenario and will never cause head of line blocking due to read / write conflicts.
10. Once the entire packet is written to buffer memory, it is placed in one or more egress queues and made available to the egress scheduler.
11. Each queue is a linked list of pointers to the first cell in each packet linked to the queue. Each egress queue can link all the packets in the buffer memory even if the buffer memory is filled with only minimum size packets.
12. Counters of the number of cells per ingress port, per ingress port priority, per egress port and egress port queue are updated according to where the packet is sent.
13. A port with packets available for transmission, will only transmit a new packet if the port shaper allows it to.
14. When an instance of the packet is selected for output by the egress scheduler, the queue manager will read the packet from the buffer memory and send it, cell-by-cell to the egress packet processing.
15. Egress processing (see chapter 3.2) determines how and if the packet shall be sent out and does the final modifications of the packet. A packet can be re-queued again if it shall be sent out multiple times, which could be the case if input/output mirroring is used. L3 multicast may also re-queue a packet multiple times to the same port.
16. Once the packet is no longer part of any egress queue, the cells it occupied in the buffer memory are deallocated so they can be used by other packets.
17. The parallel to serial converter divides the cell into MAC-bus sized chunks.
18. One asynchronous FIFO per egress port synchronizes the outgoing data from the core clock to the MAC clock.
19. Data is transmitted on the output port.

20. Egress port counters are updated.

1.2 Port Numbering Table

Table 1.1 shows the port numbering. Port 10 can serve as a CPU port.

Interface Number	BW	Clock	Clock Frequency	Sync With Core Clock	Port Number & Multicast Table Bit	CPU Port
0	40.0Gbit/s	clk_mac_rx/tx_0	312.50MHz	No	0	No
1	40.0Gbit/s	clk_mac_rx/tx_1	312.50MHz	No	1	No
2	10.0Gbit/s	clk_mac_rx/tx_2	312.50MHz	No	2	No
3	10.0Gbit/s	clk_mac_rx/tx_3	312.50MHz	No	3	No
4	10.0Gbit/s	clk_mac_rx/tx_4	312.50MHz	No	4	No
5	10.0Gbit/s	clk_mac_rx/tx_5	312.50MHz	No	5	No
6	10.0Gbit/s	clk_mac_rx/tx_6	312.50MHz	No	6	No
7	10.0Gbit/s	clk_mac_rx/tx_7	312.50MHz	No	7	No
8	10.0Gbit/s	clk_mac_rx/tx_8	312.50MHz	No	8	No
9	10.0Gbit/s	clk_mac_rx/tx_9	312.50MHz	No	9	No
10	10.0Gbit/s	clk_mac_rx/tx_10	312.50MHz	No	10	Yes

Table 1.1: Port Numbering Table

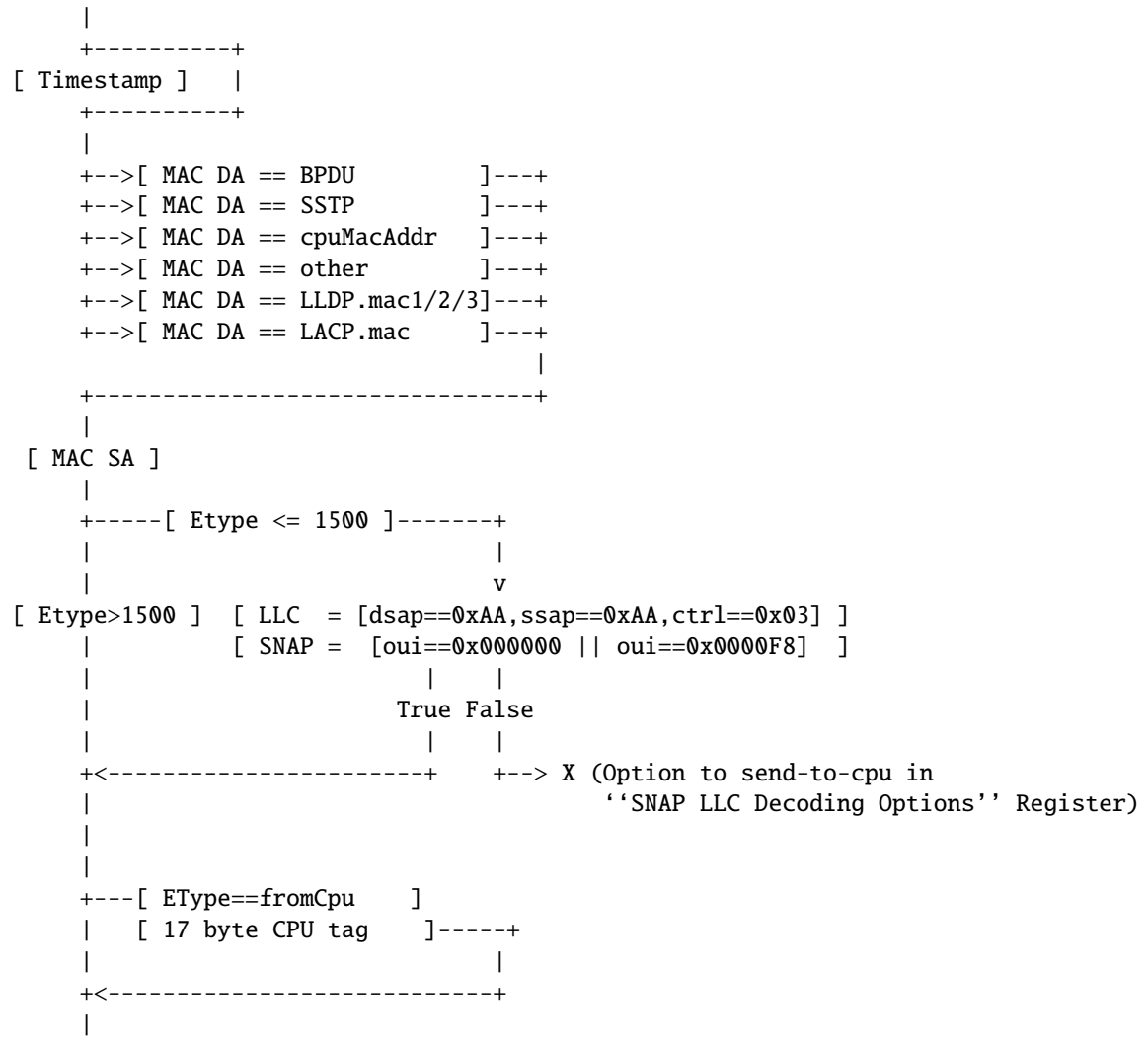
Chapter 2

Packet Decoder

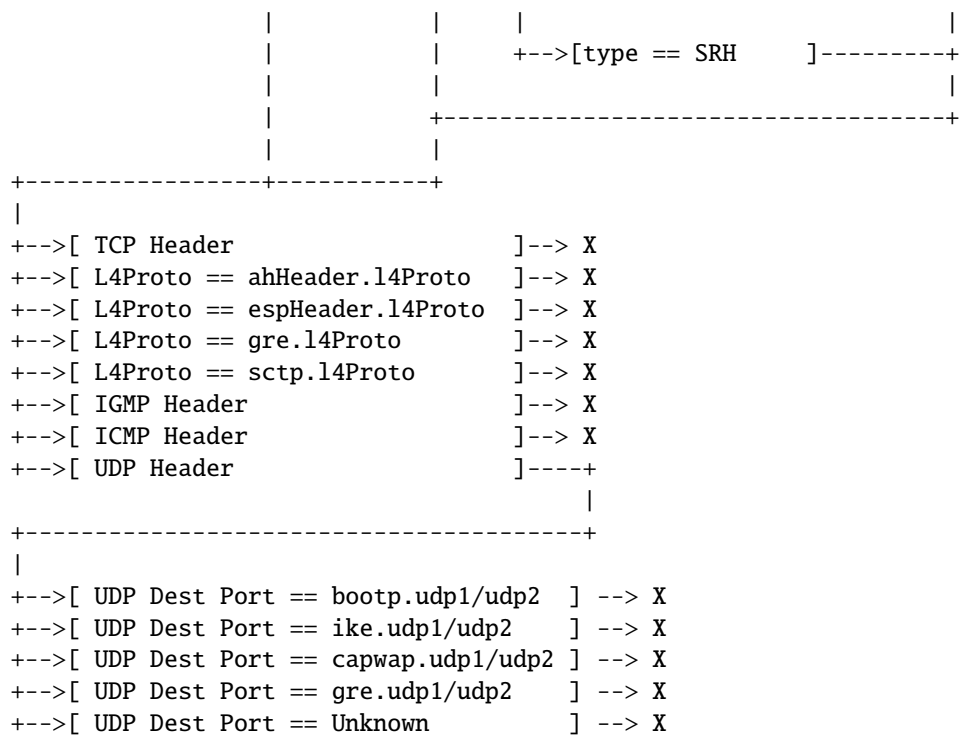
The packet decoder identifies protocols and extracts information to be used in the packet processing.

2.1 Decoding Sequence

In the following diagram the decoding of the incoming packet header is described. The comparison used to determine protocol types are described as well as the order they are decoded. The end of decoding process is denote by an *X*.




0,1,2 VLAN tags
EType==C-/S-VLAN TPID
2 byte VLAN TCI
Etype <= 1500
Etype>1500
LLC = [dsap==0xAA,ssap==0xAA,ctrl==0x03]
SNAP = [oui==0x000000 || oui==0x0000F8]
True False
X (Option to send-to-cpu in
''SNAP LLC Decoding Options'' Register)
EType==LLDP.eth
EType==IEEE_1722_AVTP.eth
EType==ARP.eth
EType==RARP.eth
EType==ieee1588EthType.eth
EType==ieee8021xEthType.eth
EType==PTP
EType==MPLS
MPLS tag 1
MPLS tag 2
MPLS tag 3
MPLS tag 4
nibble==IPv4
nibble==IPv6
nibble==unknown
EType==unknown
EType==PPPoE
PPPoE header
EType!=IPv6 or EType !=IPv4
EType==IPv6
EType==IPv4
IPv4 Header
IPv6 Header
Routing Header
type == unknown
segments left > 0
segments left ==0



The packet decoding is done according to the figure above. The packet decoding steps are described below.

1. A packet arrives at the ingress packet processing pipeline.
2. The destination MAC address is extracted and compared.
   (a) If the address matches the BPDU multicast address (01:80:C2:00:00:00) the packet can be sent to the CPU if enabled in **Send to CPU**. There is no decoding done apart from the MAC address comparison. BPDU frames are usually 802.3 encapsulated with a 802.2 LLC header. This decoding is not done by the switch. Note that packets that match the LLDP criteria described below will not be considered BPDU packets.
   (b) If the address matches the SSTP (Shared Spanning Tree Protocol) multicast address (01:00:0C:CC:CC:CD) the packet can be sent to the CPU if enabled in **Send to CPU**. There is no decoding done apart from the MAC address comparison.
   (c) If the address matches the configurable **cpuMacAddr** and this feature is enabled then the packet will be sent to the CPU port.
   (d) If the address matches one of the mac1/mac2/mac3 addresses in the **LLDP Configuration** the packet will subject to further LLDP decoding.
   (e) If the DA MAC is equal to the register **LACP Packet Decoder Options** field **mac** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.
3. The source MAC address is extracted from the packet.
4. The Ethernet type is extracted from the packet and is then compared to known types.
   (a) LLC SNAP
   If the Ethernet Type is smaller than 1500 then a packet is considered a LLC/SNAP packet. These can be located both before and after the VLAN headers. If the LLC/SNAP is not equal LLC != (dsap==0xAA,ssap==0xAA,ctrl==0x03) or SNAP != (oui==0x000000 ——

oui==0x0000F8) then there exists a option to send the packet to the CPU in register **SNAP LLC Decoding Options**. If not sent to the CPU the decoding will stop here.

(b) LLDP
If the MAC DA address is equal to any of the **LLDP Configuration** mac1/mac2/mac3 addresses and the Ethernet Type is equal to the register **LLDP Configuration** field **eth** then the field **portmask** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. Default is to forward LLDP frames to the CPU port. A packet that matches the LLDP critera will not be considered a BPDU packet even if it matches the BPDU multicast address.

(c) ARP
If the Ethernet Type field is equal to the **ARP Packet Decoder Options** field **eth** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.

(d) RARP
If the Ethernet Type field is equal to the register **RARP Packet Decoder Options** field **eth** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.

(e) 802.1X and EAPOL Packets
If the Ethernet Type field is equal to register **IEEE 802.1X and EAPOL Packet Decoder Options** field **eth** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped. The drop counter is located in **IEEE 802.1X and EAPOL Decoder Drop**.

(f) IEEE 1588 L2 Ethernet Type
If the Ethernet Type field is equal to register **IEEE 1588 L2 Packet Decoder Options** field **eth** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.

(g) PTP
When identified as a PTP/1588 packet by the EtherType and if the packet is sent to the CPU with a To CPU Tag then the *ptp* bit will be set.

(h) VLAN Tags
There are a number of fixed VLAN types that are identified as well as configurable types. The VLAN processing will use the VLAN tags that decoding has identified and ignore intermediate tags of other types.

i. Customer VLAN Type - 0x8100

ii. Service VLAN Tag - 0x88A8

iii. Configurable VLAN Type setup **Ingress Ethernet Type for VLAN tag**.

When using the Configurable Customer/Service VLAN Type the egress pipeline needs to be setup with the same values if there are actions configured that pushes new VLAN tags to the packet. This is setup in register **Egress Ethernet Type for VLAN tag**.

(i) MPLS.
One MPLS tag is decoded. No other L3 decoding will be done after this.

(j) From CPU Tags
Packets from CPU will use a Ethernet type value of 0x9988.The From CPU Tag is further described in Chapter 33.

(k) IPv4 or IPv6.
If the type identifies these protocols (potentially also after a PPPoE header) the following IPv4

or IPv6 headers are decoded. IPv4 packet with wrong header checksum can be accepted or dropped according to the **Check IPv4 Header Checksum** register. If the L4 protocol is TCP or UDP these headers are also decoded.

(l) Routing Header.
If a routing header is identified in IPv6, the L4 protocol is decoded from the routing header. The core supports further process for the segment routing header, for other routing types the core will skip the routing header if the segments left field is 0, otherwise the packet will be treated as unrecognized and sent to the CPU.

(m) L4 Protocol.
If the packet is either a IPv4 or IPv6 and if the L4 protocol is either UDP or TCP then the source port and destination port fields will be extracted.

i. ICMP header
The ICMP type along with the code extracted.

ii. IGMP header
The IGMP type along with the code and IPv4 group address is extracted.

iii. AH Header
If the next protocol field in IPv4 or IPv6 is equal to the register **AH Header Packet Decoder Options** field **l4Proto** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.

iv. ESP Header
If the next protocol field in IPv4 or IPv6 is equal to the register **ESP Header Packet Decoder Options** field **l4Proto** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.

v. GRE
If the next protocol field in IPv4 or IPv6 is equal to the register **GRE Packet Decoder Options** field **l4Proto** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.

vi. SCTP
If the next protocol field in IPv4 or IPv6 is equal to the register **SCTP Packet Decoder Options** field **l4Proto** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.

(n) UDP or TCP Source or Destination Port Checks

i. GRE
If the Destination Port in UDP is equal to the **GRE Packet Decoder Options** field **udp1** or field **udp2** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.

ii. DNS
If the Destination Port in UDP or TCP is equal to the **DNS Packet Decoder Options** field **l4Port** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.

iii. BOOTP or DHCP
If the Destination Port in UDP is equal to the register **BOOTP and DHCP Packet Decoder Options** field **udp1** or field **udp2** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.

iv. CAPWAP
If the Destination Port in UDP is equal to the register **CAPWAP Packet Decoder Options** field **udp1** or field **udp2** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.

v. IKE
If the Destination Port in UDP is equal to the register **IKE Packet Decoder Options** field **udp1** or field **udp2** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.

vi. IEEE 1588 L4
If the Destination Port, and IPv4 or IPv6 and the UDP is equal to the register **IEEE 1588 L4 Packet Decoder Options** then the field source port bit in the **toCpu** determines if the packet shall be sent directly to the CPU, bypassing normal forwarding process. The source port bit in the field **drop** determines if the packet shall be dropped.

(o) Unknown.
After an unknown Ethernet type no further decoding is done.

If the packet is too short to be decoded then normal processing will be interrupted and the packet will be dropped. This occurs if the packet ends such that a field required by the core is not complete. Only packet fields actually used by the cores processing will be subject to this check. Input mirroring can still be used to observe the incoming packet.

Chapter 3

Packet Processing

3.1 Ingress Packet Processing

The ingress packet processing is done as soon as the packet enters the switch. The packet is not sent to the buffer memory until the ingress packet processing is done.

1. Ingress Functional Control
   Depending on where the packet came from and what headers the packet has a functional control register is selected. This register allows for fine grained control of what functions the ingress packet processing shall do. For details see the **Functional Control** chapter.

2. Source Port to Link Aggregate
   Source port is mapped to a link aggregate through the **Link Aggregation Membership** table. From this point all references to source ports are actually link aggregate numbers. For details see the **Link Aggregation** chapter.

3. Packet Decoding for Tunnel Exit Lookup
   The packet headers are decoded and data extracted. For details see the **Packet Decoder For Tunnel Exit** section in the tunneling chapter.

4. Tunnel Exit Lookup
   The packet is subjected to a tunnel exit lookup which if found true can remove a part of the packets headers and/or payload of the packet. Certain fields from the original packet can also be copied to the inner packet. Once this has been done the packet processing will be only done on the inner packet. For details see the **Tunnel Exit** section.

5. Packet Decoding
   The packet headers are decoded and data extracted. For details see the **Packet Decoding** chapter.

6. Destination MAC Address Range Classification
   The destination MAC address is compared with **Reserved Destination MAC Address Range** table to determine if it should be dropped, sent to CPU or if priority should be forced.

7. Source MAC Address Range Classification
   The destination MAC address is compared with **Reserved Source MAC Address Range** table to determine if it should be dropped, sent to CPU or if priority should be forced.

8. SMON
   If the packets source port and the VID for the outermost VLAN matches an SMON counter then that counter will be updated (see the **Statistics** chapter).

9. Ingress Port Packet Type Filter
   The ingress packet type filter, setup through **Ingress Port Packet Type Filter** per source port, determines if the packet will be dropped or be processed further. This is based on protocol type and type of VLAN. See the **VLAN and Packet Type Filtering** chapter.

10. Configurable ACL
    The incoming packet is classified on a configurable selection of L2, L3 and L4 fields. The ACL lookup is a d-left hash search, described in Dleft Lookup. There are numerous actions that can be applied when a packet matches an ACL entry. For details see the **Configurable ACL Engine** section.

11. Ingress Spanning Tree
    The ingress spanning tree state of the source port (from the **Source Port Table**) is checked to determine if packet processing should continue. STP is further described in the **Spanning Tree** chapter.

12. Ingress VLAN Processing
    VLAN processing consists of two parts. Determining the VLAN membership and performing VLAN header modifications.

    The VLAN membership is determined from the assigned ingress VID. See the **Assignment of Ingress VID** section. This will then be used to index into the **VLAN Table** to determine, among other things, VLAN port membership , MSTP and Global ID used in L2 lookups.

13. Ingress MSTP
    The VLAN membership determines which MSTP the packet belongs to by pointing into the **Ingress Multiple Spanning Tree State** table. The state of the source port within this MSTP is checked to determine if packet processing should continue. MSTP is further described in the **Spanning Tree** chapter.

14. IP Routing
    The routing function figures out where to forward the packet by determining the Next Hop. For details on the routing function see the **Routing** chapter.

    (a) Determine Next Hop
    The routing function is entered if an IP packet matches the router ports MAC address (**Router Port MAC Address**) and routing is allowed on the packets VLAN. L2 lookup, learning and aging will not be performed on routed packets. The router will search for the IP destination address in the routing tables to determine the packets Next Hop, i.e. which port to send the packet to.

    (b) VLAN Operations
    The Next Hop will also determine up to two VLAN operations to perform on the routed packet.

15. IPv4 checksum check and drop.
    For IPv4 packets calculate the checksum value and optionally drop the packet with wrong checksum value. For a routed IPv4 packet the check and drop is always performed.

16. L2 Switching
    If the packet is not routed the destination MAC address is searched for in the **L2 DA Hash Lookup Table**. If the address is found the corresponding entry in the **L2 Destination Table** will return a single destination port or multiple egress ports (if the destination address points to a multicast entry). The status in the **L2 Aging Table** is also updated. If the destination address is not found then the packet will be flooded to all ports that are members of the packets VLAN. See chapter **L2 Switching** for details.

17. L2 Action Table Lookup
    The L2 Action Table Lookups provides a extra level of controll over what shall be done with the L2 packets. It can be used to archive 802.1X compliance and be used to secure the switch. The functionality has a enable bit in the **Source Port Table** field **enableL2ActionTable**. Depending on the result from both the L2 SA Lookup , L2 DA Lookup and status on source port (**I2ActionTablePortState**) and destination port(s) **L2 Action Table Egress Port State** a address is formed to read out L2 Action Tables. The **L2 Action Table** is based on the packets destiantion ports, while **L2 Action Table Source Port** is based on the packets incoming source port. If the packet is going to no egress port (portmask==0) then none of the **L2 Action Table** actions will be done while the **L2 Action Table Source Port** is always carried out (When function is enabled).

18. Egress Spanning Tree

When the destination port(s) are known, the spanning tree state for the destination ports are checked in **Egress Spanning Tree State** register.

19. Egress MSTP
    The MSPT state for the destination ports are checked in the **Egress Multiple Spanning Tree State** register. The MSTP id, determined above, is used to index the table.

20. Learning Lookup
    If the packet is not routed the source MAC address is searched in the **L2 SA Hash Lookup Table**. If the address is not found or it has moved to a different port then the Learning Engine will update the tables unless the packet was marked to be dropped. See the **Learning and Aging** chapter for details.

21. IP Statistics
    Statistics of IP unicast, multicast and routed packets are updated.

22. Configurable Egress ACL
    The Egress ACL can classify incoming packet based on a configurable selection of L2, L3 and L4 fields but also based on the result from switching and routing. The ACL lookup is a D-left hash search, described in Dleft Lookup. There are numerous actions that can be applied when a packet matches an ACL entry. For details see the **Configurable Egress ACL Engine** section.

23. Ingress/Egress Port Packet Type Filter
    As the packet is ready to be queued, the **Ingress Egress Port Packet Type Filter** is applied for each egress port where the the packet is to be queued. See chapter **VLAN and Packet Type Filtering**.

24. Link Aggregation
    The destination ports are now mapped to physical ports using a hash function on the packet headers. The hash index selects which of the physical member ports of this link aggregate that the packet should be sent to. See the **Link Aggregation** chapter.

25. Multicast Broadcast Storm Control
    Multicast packets that are destined for physical ports that have exceeded the MBSC limits will be dropped at this point. See chapter **Multicast Broadcast Storm Control**.

26. Input Mirroring
    If the source port is setup to be input mirrored the mirror port is now added to the list of destination ports. A copy of the input packet, without modifications, will be transmitted on the selected mirror port.

27. Determine Egress Queue Priority
    Egress queues are assigned to packets based on their L2/L3 protocols or classification results. See the **Determine Egress Queue Priority** section.

28. Packet Initial Coloring
    Initial colors are assigned to packets based on their L2/L3 protocols or classification results to represent the drop precedence. See the **Ingress Packet Initial Coloring** section.

29. NAT Action Table Check
    Certain processing bits, if the packet was routed, if the packet was switch, if the packet was flooded along with bits from ingress and egress ACL plus status bits from ports are looked up in the table **Egress Port NAT State**. This table can redirect packets to the CPU, Drop the packet or do nothing.

30. Queue Management
    If queue management has turned off queuing to a port the packet will be dropped at this point. See section **Queue Management** for details.

31. Drop Statistics
    If the preceding processing has not set any destination ports then the packet is dropped and the **Empty Mask Drop** counter is incremented.

32. Ingress Admission Control
    Packets are grouped into traffic groups based on source port numbers and packet headers, and the

bandwidth of each traffic group is measured. If a traffic group exceeds the configured bandwidth or burst size, the initial packet color can be remarked or the packet can be dropped. See the **Ingress Admission Control** section. While the groupping process is through sequence of ingress packet processing steps, the metering process is after all other ingress packet processing are done and before the enqueuing of the packet.

3.2 Egress Packet Processing

After ingress packet processing the packet is stored in the packet buffer memory. The egress packet processing is done when the packet is scheduled for transmission. A single packet can be sent out in multiple copies, for example due to broadcast or mirroring. If the copies are not identical, or multiple copies should be transmitted on the same port, then the packet will be re-queued. This means that it will be re-inserted into the queue engine, where it will again be selected for output and passed once more through the egress packet processing.

1. Egress Functional Control
   Depending on where the packet came from and what headers the packet has a functional control register is selected. This register allows for fine grained control of what functions the egress packet processing shall do. For details see the **Functional Control** chapter.
2. Output Mirroring
   If output mirroring is enabled for the egress port then the packet is re-queued, so that a copy of the outgoing packet will be transmitted on the output mirror destination port. See the **Mirroring** chapter.
3. IP Header Update
   For routed packets the IP checksum is updated after TTL update, as setup in **Egress Router Table**.
4. Routed DA/SA MAC Update
   For routed packets update the MAC addresses based on the Next Hop.
5. Egress Port VLAN
   A VLAN header operation can be performed based on the physical output port. See the **VLAN Processing** chapter.
6. Egress Port Packet Type Filter
   The egress packet type filter, setup through **Egress Port Configuration** per egress port, determines if the packet will be dropped or be allowed to be transmitted. See the **VLAN and Packet Type Filtering** chapter.
7. VRF Statistics
   If the packet is routed it will be counted in **Transmitted Packets on Egress VRF** counter for the VRF it belongs to.
8. Egress VLAN Translation
   Potentially replace the outgoing VID and Ethernet Type on a specific port with a specific VID. Uses a Dleft lookup in **Egress VLAN Translation Small Table**, **Egress VLAN Translation Large Table** and **Egress VLAN Translation TCAM**.
9. Reassemble Packet Headers
   Depending on if the packet shall entern a tunnel or not this can be the final step in the egress processing which is to reassembly the outgoing (potentially inner) packet header.
10. Tunnel Entry
    Result from packet processing, both ingress and egress, can result in that a packet shall enter a tunnel. This tunnel is described as a number of bytes to be added to the packet at certain points. There also exists options which allows the outer packet to copy certain data from the inner packet (such as TOS byte, next header).

Chapter 4

Latency and Jitter

This chapter is meant as an introduction to the causes of latency and jitter in the core. It gives some numbers, but mostly points out the general principles.

The switch has a fixed minimal latency, the bulk of which comes from the ingress and egress packet processing, the store-and-forward operation, and the dataflow registers between design units.

4.1 Latency

The major contributors to latency:

1. The Serial to Parallel converter (SP) gathers the data chunks from the MAC into wider cells.
2. The IPP has a fixed latency of 102 core clock cycles.
3. The queue engine stores the entire packet in buffer memory before adding it to the queues.
4. The EPP has a fixed latency of 56 core clock cycles.
5. Packet modifications that decrease the packet size (for example removing a VLAN) will cause a packet to be delayed one scheduling slot for certain packet sizes.

4.2 Jitter

There are tree places (t1-t3) in the core where latency jitter can be introduced. See Figure 4.1 on page 36.

t1 In the SP the ports are visited in a fixed order, thus introducing a jitter the size of the port visitation period. There is also an asyncronous FIFO between the port and the core clock regions, adding one clock period (of the slowest clock) of jitter.

t2 The egress scheduler visits the ports in a fixed order, introducing a jitter the size of the port visitation period.

t3 The asyncronous FIFO between the core and port clock regions adds one core clock period (of the slowest clock) of jitter.

Note, though, that the core is dimensioned to handle even the worst case jitter without causing packet drops or increased IFG.

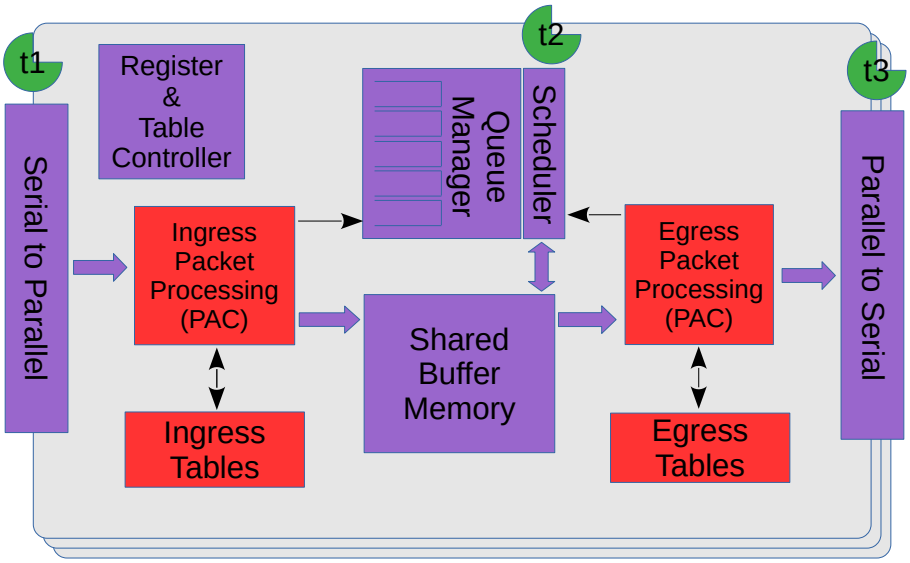


Figure 4.1: Jitter Overview

Chapter 5

VLAN Processing

5.1 Assignment of Ingress VID

All packets entering the switch will be assigned an ingress VID even if the incoming packet doesn't have a VLAN header. This is the VID used to lookup in the **VLAN Table**.

The ingress VID assignment is processed in several steps. The initial assignment is controlled per source port by the **vlanAssignment** in the **Source Port Table** and then it can be updated in a number of ways ranging from L2 to L4 protocols.

5.1.1 VID Assignment from Packet Fields

Ingress VID can be assigned from certain packet fields, other than the packets incoming VID.

There exists a number of these field tables listed below:

- On the L2 MAC layer in **Ingress VID MAC Range Search Data** and its result table **Ingress VID MAC Range Assignment Answer**, the search data can be either on source MAC or destination MAC ranges.
- On the Outer VID in **Ingress VID Outer VID Range Search Data** and its result table **Ingress VID Outer VID Range Assignment Answer**. If the packet has no outer VID then this is skipped. There exists options if the packets VID shall be matched depending on if this is a S-tag or C-tag.
- On the Inner VID in **Ingress VID Inner VID Range Search Data** and its result table **Ingress VID Inner VID Range Assignment Answer**. If the packet has no inner VID then this is skipped. There exists options if the packets VID shall be matched depending on if this is a S-tag or C-tag.
- On the Ethernet Type which is following the innermost VLAN tag. The setup is in **Ingress VID Ethernet Type Range Search Data** and its result table **Ingress VID Ethernet Type Range Assignment Answer**.

VID Assignment Search Order

If there are matches in multiple tables then the "order" field determines which result to use. The result with the highest order value will be used. The search order within a table is not affected by the order field.

The search is carried out as follows:

1. The MAC ranges, defined in **Ingress VID MAC Range Search Data**
2. The Outer VID ranges, defined in **Ingress VID Outer VID Range Search Data**
3. The Inner VID ranges, defined in **Ingress VID Inner VID Range Search Data**

4. The Ethernet Type ranges, defined in **Ingress VID Ethernet Type Range Search Data**

5.1.2 Force Ingress VID from Ingress Configurable ACL

The ACL engine has an option to override the ingress VID assigned above. If the forceVidValid field in the **Ingress Configurable ACL N Small Table** is set to 1, the corresponding forceVid field will be used as the new ingress VID value. The same applies to the **Ingress Configurable ACL N Large Table** and **Ingress Configurable ACL N TCAM Answer** tables. The detailed L2 ACL match and action are described in the **Configurable ACL Engine** section.

5.2 VLAN membership

All packets entering the switch will be member of a VLAN, either assigned from the incoming VLAN headers or through a default configuration described below.

The VLAN membership defines which ports that are part of a VLAN. Packets belonging to a VLAN can only enter on the ports that are member of the VLAN.

The L2 switching can only send out packet on the ports that are members of the VLAN, including broadcast, multicast and flooding. This limitation does not apply to routed packets.

The VLAN membership also assigns a global identifier (GID) to a packet which is used during L2 lookup to allow multiple VLANs to share the same L2 tables.

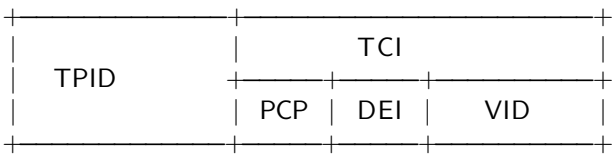
The VLAN membership also determines which multiple spanning tree (MSTP) a packet is part.

The egress queue priority can also be assigned from the VLAN membership (see chapter 23.1).

5.3 VLAN operations

There are a number of operations that can be performed on the packet's VLAN headers such as push/pop etc. Multiple operations can be performed in sequence such that the resulting VLAN header stack from one operation becomes the input to the following operation. However the content of the VLAN headers do not come from previous VLAN operations, they are always created from the original incoming packet or from tables.

For reference here is the 802.1Q VLAN header:



When referring to outermost and innermost VLAN header, outermost means the first VLAN header that the packet decoding has identified as a VLAN header. Innermost means the second VLAN header as identified by the packet decoder.

The VLAN operations that can be performed are:

- Pop - The outermost VLAN header in the packet is removed.
- Push - A new VLAN header is added to the packet before any previous VLANs. It will become the new outer VLAN. The selection of each of the VLAN fields such as TPID, VID, PCP and DEI/CFI are configurable. These fields can either come from existing VLAN headers in the original incoming packet or from tables.
- Swap/Replace - The outermost VLAN header in the packet is replaced. The selection of each of the VLAN fields such as TPID, VID, PCP and DEI/CFI are configurable. These fields can either come from existing VLAN headers in the original incoming packet or from tables.

- Penultimate Pop - All VLAN headers (up to as many as supported by the packet decoder) are removed from the packet.

Figure 5.1 shows the effect of one of these operations on a packet.

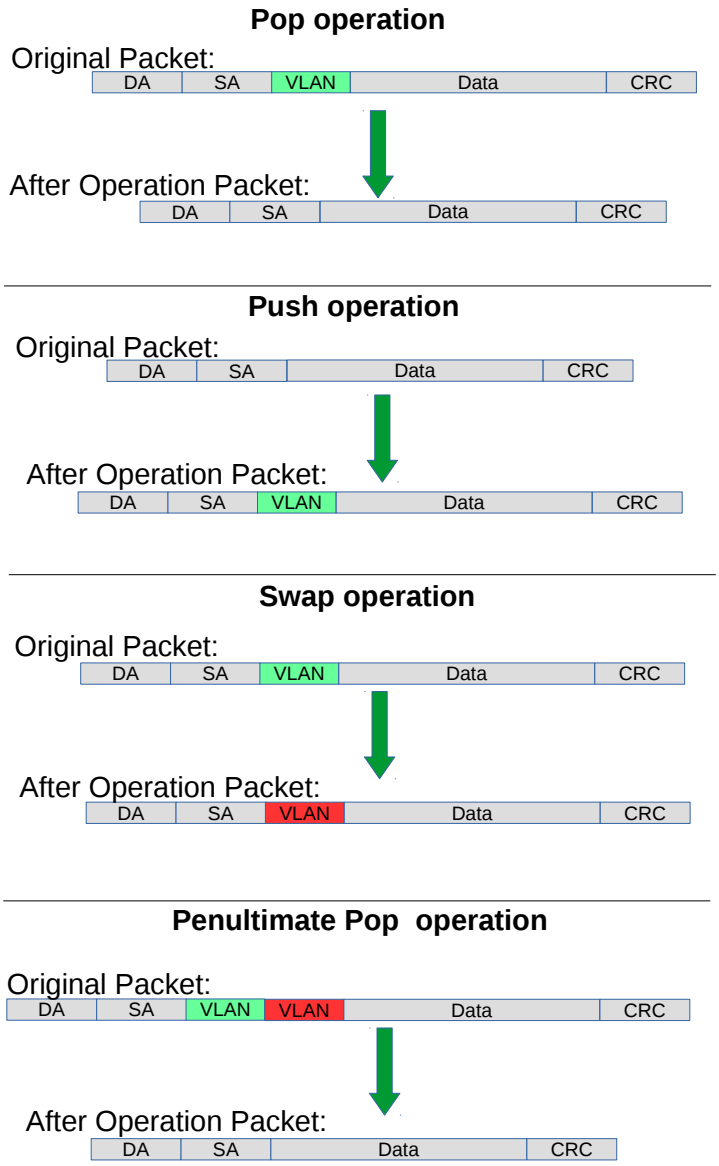


Figure 5.1: VLAN Packet Operations

5.3.1 Default VLAN Header

When a packet enters without a VLAN header an internal default VLAN header will be created. The internal header will have VID, CFI and PCP from **Source Port Table** fields **defaultVid**, **defaultCfiDei**, **defaultPcp**.

The default VLAN header is only used in VLAN operations that selects data from the VLAN packet header.

5.3.2 Source Port VLAN Operation

A VLAN operation to be performed (e.g. push, pop, swap) can be selected by the **vlanSingleOp** field in **Source Port Table**.

If the packet is routed this VLAN operation will not be performed.

5.3.3 Operation Based On Incoming Packets Number of VLANs

There exists a option which overrides the default **vlanSingleOp** field depending on the number of VLANs the packet has. This operation allows a user to set a specific operation depeding on the number of VLANs the incoming packet has. This VID operation then overrides the default VID operation. This operation is setup in field **nrVlansVidOperationIf**.

5.3.4 Configurable ACL VLAN Swap Operation

The **Ingress Configurable ACL N Small Table** , **Ingress Configurable ACL N Large Table** and **Ingress Configurable ACL N TCAM Answer** tables provides three fields updateVid, updatePcp and updateCfiDei to perform a VLAN swap operation. The VLAN type can also be changed using the updateEType. VLAN push and pop operations are not supported in this ACL.

If the packet is routed then the VLAN swap operation in the ACL will not be performed.

5.3.5 VLAN Table Operation

The **VLAN Table** defines the VLAN port membership, which GID (Global Identifier) to use in L2 lookups, the MSPT to use , if routing is allowed and a VLAN operation to be performed (e.g. push, pop or swap).

If the packet is routed then the VLAN operation from **VLAN Table** will not be performed.

5.3.6 VLAN Table VID Operation Based On the Packets Number of VLANs

There exists a option which overrides the default **vlanSingleOp** field depending on the number of VLANs the packet has. This operation allows a user to set a specific operation depeding on the number of VLANs the incoming packet has after the source port operation push/pop/swap/penultmate pop has been done. The VID operation then overrides the default VID operation specified in field **vlanSingleOp** and all its data fields. This operation is setup in field **nrVlansVidOperationIf**. This setting is done on a per port basis allowing each source port to have its own setting. Source port 0 is represented in bits [1:0] , Source port 1 is represented in bits [3:2] and so on.

5.3.7 Egress Port VLAN Operation

A VLAN operation to be performed (e.g. push, pop, swap) can be selected by the **vlanSingleOp** field in **Egress Port Configuration**.

A pop operation is done on packets that match a specific VID if **enablePriorityTag** is set in **Source Port Table**.

5.3.8 Egress Port VID Operation

Egress Port VID Operation provides an option to override the default **vlanSingleOp** depending on the number of VLANs the packet has and the ingress VID of the packet. Each entry of the **Egress Port VID Operation** register compares the egress port, ingress VID and VLAN tagging conditions and activate the corresponding VLAN operation from the first hit.

5.3.9 Egress Vlan Translation

This operation which is located in the egress path allows a replacement of the outermost VLAN Identifier in the packet. The egress port, the outermost VID of the packet after all VLAN operations and the outermost VID type (C or S tag) creates a lookup key to be used in a Dleft lookup using the **Egress VLAN Translation Small Table**, **Egress VLAN Translation Large Table** and **Egress VLAN Translation TCAM** Tables. If multiple hits the **Egress VLAN Translation Selection** can be used to determine which result to select. It is possible to mask the search data using **Egress VLAN Translation Search Mask**..

5.3.10 Priority Tagged Packets

Priority tagged packets are packets that have a VLAN tag with VLAN ID equal to 0. The purpose of these are to extract the PCP bits and use as priority.

The priority extraction can be done as described in 23.1 **Determine Egress Queue** section.

The priority tag can be ignored in all VLAN processing and finally removed on the egress if **enablePriorityTag** is set in **Source Port Table**. Which VLAN ID that triggers this is configured in **priorityVid**

The priority extraction is not dependent on the **enablePriorityTag** setting.

5.3.11 Router VLAN Operations

- If a packet is routed then any VLAN headers in the incoming packet detected by the packet decoder will be removed on the egress.
- All other VLAN operations during ingress packet processing will not be done on routed packets.
- The routers next hop will point to the **Next Hop Packet Modifications** table which can specify up to two push VLAN operations to perform.
- The **Egress Port Configuration** VLAN operation is performed on routed packets after the VLAN operations specified in **Next Hop Packet Modifications**.

5.3.12 VLAN Operation Order

All VLAN operations are performed in sequence on a packet. They follow the order as:

1. One of the four VLAN operations from:
   - **Source Port Table** VLAN operation.
   - Inner VLAN push operation from routers **Next Hop Packet Modifications**.
2. One VLAN swap operation from:
   - **updateVid**, **updatePcp**, **updateCfiDei** or **updateEType** in the **Configurable ACL Engine**.
3. One of the four VLAN operations from:
   - **VLAN Table** VLAN operation.
   - Outer VLAN push operation from routers **Next Hop Packet Modifications**.
4. One of the four VLAN operations from:
   - **Egress Port Configuration** VLAN operation.

The input to the first VLAN operation is the incoming packet. The packet decoder identifies the position of the VLAN headers in the packet and this information is used for the subsequent VLAN operations.

The output from one VLAN operation is input to the next VLAN operation. For example if the first VLAN operation is a push and the second is a swap then the effect will be that the pushed header is replaced by the swap.

If a VLAN operation needs a VLAN header in the packet, i.e. a swap or a pop, and there is no VLAN header in the packet then the operation will not be performed.

5.3.13 VLAN Operation Examples

This process is first described informally with a few examples but to fully specify the behavior it is also described as pseudo code.

Here are examples of sequences of VLAN operations performed on packets with mixed VLANs and custom tags. The incoming packet headers, sequence of VLAN operations and outgoing packet header are briefly described.

```
'V1'..'V2' are VLAN tags in original packet
'new V1'..'new V2' are VLAN tags that have been created by the VLAN operations


Example 1)
incoming packet:
[DA][SA][V1]

VLAN operations: 1. swap new V1
outgoing packet:
[DA/SA][new V1]

Example 2)
incoming packet:
[DA][SA][V1]

VLAN operations: 1. push new V1

outgoing packet:
[DA/SA][new V1][V1]

Example 3)
incoming packet:
[DA][SA][V1][V2]

VLAN operations: 1. push new V1

outgoing packet:
[DA/SA][new V1][V1][V2]

Example 4)
incoming packet:
[DA][SA][V1][V2]

VLAN operations: 1. pop

outgoing packet:
[DA/SA][V2]

Example 5)
incoming packet:
[DA][SA][V1][V2]

VLAN operations: 1. pop
VLAN operations: 2. swap new V1
VLAN operations: 3. push new V2

outgoing packet:
[DA/SA][new V2][new V1]
```

5.3.14 VLAN Reassembly

The reassembly of the VLAN headers uses data from the packet decoding together with data from the VLAN operations to create the new packet headers.

The following is Python code that exactly models the reassembly operation. The process starts when the L3 and payload in the outgoing packet has been reassembled but before any VLAN or other L2 tags have been added.

The code uses the same incoming packet and VLAN operations as **Example 5)** in the previous section to illustrate the data structure.

```
# The design supports this number of VLAN tags in the ingress packet.
nr_of_ingress_vlans = 2

# Packet decoding results in a list of all VLAN tags from the ingress packet.
pkt_vlan_tags = [ 'V2', 'V1' ]

# Number of VLAN tags that will be used from the original packet.  Before any
# VLAN operations this equals number of incoming VLANs, it could be decreased by
# swap or pop but can't be increased. When nr_of_new_vlans==0, pop or swap will
# decrement it. At any time popAll will set it to 0.
nr_of_pkt_vlans = 2

# Number of new VLAN tags to be used in the reassembly. Push and swap operations
# will increment this and at the same time the new VLAN to the end of new_vlans.
# popAll will set it to 0.
nr_of_new_vlans = 0

# New VLAN tags to be used in the reassembly.
new_vlans = []

# After all VLAN operation sequences: pop, swap new V1, push new V2, VLAN
reassembly collects needed information to get started.
nr_of_pkt_vlans = 0
nr_of_new_vlans = 2
pkt_vlan_tags = [ 'V2', 'V1' ]
new_vlan_tags = [ 'new V1', 'new V2' ]

# At the starting point of re-assembling the VLAN tags the egress packet contains the
# updated packet after the original tags, i.e. L3/L4/payload.
egress_pkt = ['payload']

# Reassemble the tags with updated VLANs.
while nr_of_pkt_vlans > 0: # Egress packet has VLAN tags from ingress
    # Pop inner most tag from pkt_vlan_tags and insert it first in the egress_pkt
    egress_pkt.insert(0,pkt_vlan_tags[0])
    pkt_vlan_tags = pkt_vlan_tags[1:]
    nr_of_pkt_vlans -= 1

while nr_of_new_vlans > 0: # Egress packet has new VLAN tags
    # Insert a new VLAN first in the egress_pkt from internal VLAN stack.
    egress_pkt.insert(0,new_vlan_tags[0])
    new_vlan_tags = new_vlan_tags[1:]
    nr_of_new_vlans -= 1


# Now egress_pkt contains all updated VLAN headers and tags.  After this new DA/SA
# and other new tags like to_cpu_tag is added to get the final egress packet.
```

Chapter 6

Switching

Most packets will be subjected to a L2 MAC destination address lookup to determine the destination egress port (or ports). These are the exceptions:

- Packet decoder determines that this protocol should be send to the CPU. See Packet Decoder chapter.
- A classification unit action dropped the packet, sent the packet to the CPU, or sent the packet to a specific egress port. See Classification chapter.
- The packet has a From CPU tag which allows the normal packet forwarding process to be bypassed. See Packet From CPU Port section.
- The packet is routed. See the Routing chapter.
- The packet is dropped earlier in the packet processing chain. See chapter Ingress Packet Processing for details.

6.1 L2 Destination Lookup

If none of the above applies a L2 MAC address destination lookup will be performed in the following manner:

- The GID is given by the **gid** field from the **VLAN Table** lookup. See the VLAN Processing chapter.
- The hash is calculated with {GID,DA MAC} as key (see MAC Table Hashing).
- The hash is used as index into the **L2 DA Hash Lookup Table**. 8 entries are read out in parallel, each corresponding to a hash bucket.
- The bucket entries are all compared with the {GID,DA MAC} key and if one entry is equal to the key that entry is considered a match.
- The {GID, DA MAC} key is also compared with all the entries in the **L2 Lookup Collision Table** CAM. The CAM is searched starting from entry 0 and the first matching entry is treated as a match. Any following matching entries are ignored.
- Some entries in **L2 Lookup Collision Table** has per-bit masks. These are set up in the **L2 Lookup Collision Table Masks** registers. Using the mask an entry can define with single-bit granularity what shall be included in the comparison. A zero in the mask means that the corresponding bit shall be ignored, while a one means that the bit shall be compared.
- An entry in the **L2 DA Hash Lookup Table** is only compared if the corresponding valid bits are set. The valid bits are located in the **L2 Aging Table** , the **L2 Aging Status Shadow Table** and the **L2 Aging Status Shadow Table - Replica** . If all the valid bits are not set then this will result in a non-match even if the {destination MAC , GID} in the **L2 DA Hash Lookup Table** entry matches. For the collision CAM the valid bits are located in the **L2 Aging Collision Table** and **L2 Aging Collision Shadow Table**. See figure 6.1.

- If both CAM and L2 hash tables return a match, the result from the CAM table will take precedence.
- Once the final entry has been determined, the result is read out from the **L2 Destination Table**. It has enough entries to fit the destinations for both the L2 hash table and the L2 CAM table. The L2 CAM table entries are located after the L2 hash table entries.
- If the **pktDrop** field in the **L2 Destination Table** is set the packet will be dropped.
- If the destination shall be a single port (i.e. it is not to be multicasted) then the **uc** field shall be set to one and the **destPort or mcAddr** field shall contain the egress port number.
- If a packet shall be sent to multiple output ports then the **uc** field shall be set to zero and the **destPort or mcAddr** field shall contain a pointer to a entry in the **L2 Multicast Table**. The entry in the **L2 Multicast Table** contains a portmask where bit 0 represents port 0, bit 1 port 1, and so on. A bit set to one results in the corresponding port receiving a packet.
- The DA MAC address ff:ff:ff:ff:ff:ff is the broadcast address, meaning that all the member ports in the VLAN (configured in the **VLAN Table vlanPortMask** field) will receive a packet.
- Normally the source port is excluded from the destination portmask. If that results in an empty destination port mask then the packet is dropped and counted in the **L2 Lookup Drop** register.

  This behaviour can be changed using the **Hairpin Enable** register, allowing a packet to be switched to the same port it came in.
- Ports that are not members of the VLAN will be removed from the portmask. If there are no ports left in the port mask then the packet is dropped and counted in the **L2 Lookup Drop** register.
- If there is no hit in either the **L2 DA Hash Lookup Table** or the **L2 Lookup Collision Table**, then the packet will be flooded, i.e. sent out to all ports in the VLAN. This means that the port mask for the outgoing packet will be taken from the **vlanPortMask** field in the **VLAN Table**.
- If the **Flooding Action Send to Port** is enabled on this source port (using **enable** set to one) and the packet is flooded then the packet is sent to the destination port pointed to by the field **destPort** instead of being flooded to all ports part of the packets VLAN. The destination port does not need to be part of the packets VLAN group membership.
- If there is a hit then the hit bit in the **L2 Aging Table** is set to one.
- The final physical port is determined by the link aggregation. See chapter Link Aggregation for more information.
- Learning new unknown SA MAC addresses is described in chapter Learning and Aging.

6.2 Software Interaction

Observe that L2 tables can not be directly written by software if learning engine is turned on. Doing so can cause packets to be dropped and/or flooded and the learning engine may stop working. See chapter Learning and Aging for information how to safely update the L2 tables.

6.3 L2 Action Table

There is two tables which allows detailed control for each packet depending on the source L2 MAC table result, the destination L2 MAC table result and the ingress and egress port which each has a configurable state. This the L2 Action Table used for each egress port which the packet shall be sent to is defined in **L2 Action Table** and secondly the **L2 Action Table Source Port**. Both tables used a number of bits from the source port table, egress port state, SA and DA MAC lookups to form a address into the tables which is then read out and acted on. Each source port enables if the L2 Action tables shall be used or not using the field **enableL2ActionTable**. The L2 Action Tables can be used to permit specific frames from certain source ports to other destination ports using a filter defined in **Allow Special Frame Check For L2 Action Table**. There are 4 rules which are shared among all ports and pointed from the L2 Action Tables as a result by setting **useSpecialAllow** to one and then pointing to the rule using field **allowPtr**.

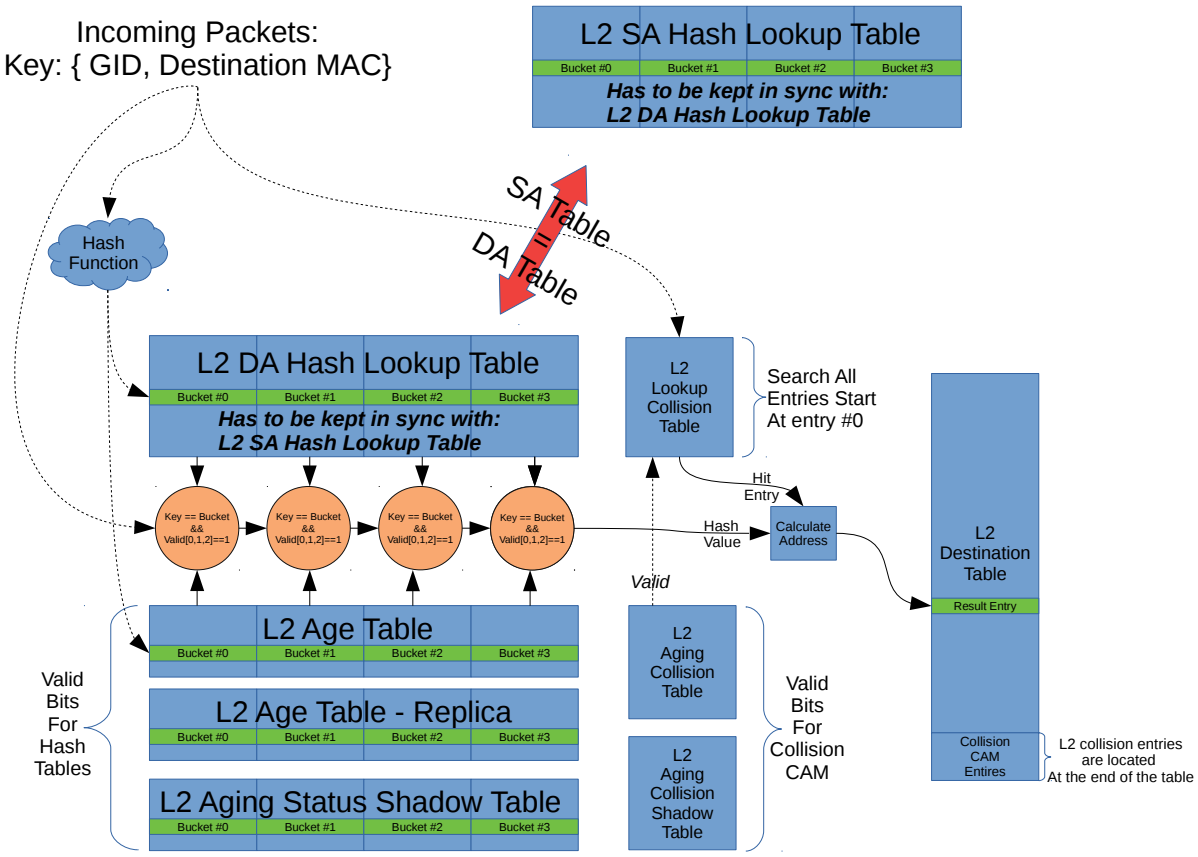


Figure 6.1: L2 Lookup Overview

If a packet is going to no egress ports (portmask==0) then none of the actions in the **L2 Action Table** will be carried out, while the **L2 Action Table Source Port** will always be carrie out since a packet always comes in on a source port. Because of this the addressing is slightly different for these two table lookups.

The use cases for the tables is described below. Both tables have the same result actions.

6.3.1 Learning Unicast and Learning Multicast

As stated before the L2 Action Table can be used to stop learning on certain frames. There is a additional setting allowing the user to define if the learning is not to be allowed for unicast or multicast packets. Since a learning lookup is based on the Source MAC address this is also what is compared against. If the SA MAC is a multicast address then the **noLearningMc** field will be used to determine if the packet shall be learned or if SA MAC address is a unicast then the **noLearningUc** will determine if the packet shall be learned or not.

6.3.2 Drop and Learning

If a packet is dropped by the L2 Action Table the packet will be still be learned. If you want the packets not to be learned then both **dropAll** and **noLearningUc** and **noLearningMc** should be turned on (set to one).

6.3.3 Priorities Between Actions

There are multiple actions from the L2 action table this section explains the order between them.

1. The drop special packet is first carried out and drops all instances of the packet

2. The drop port move then takes priority and drops all instance of the packet
3. The drop-all drops all instances of a packet however special type packets can still be accepted if they are setup to do so.
4. After the drops the send-to-CPU is carried out. Only a single copy will be sent to the CPU.

6.3.4 Using L2 Action Table for 802.1X

Simple Port Authentication

By using the source port bit **l2ActionTablePortState** and the egress port state bit in register **L2 Action Table Egress Port State** to indicate if a port is authenticated or not packets can be limited to communicate with other ports. This is done by setting up the different addresses in the L2 Action Table to do drop operations when a packet comes in from a non-authenticated port going to a authenticated port.

Port Authentication with MAC addresses

In order to allow already existing computers (MAC address) allow to pass through the switch without any problems the SA lookup result bit **l2ActionTableSaStatus** can be used indicate if this source MAC address (i.e. computer/end-station) has been authenticated or not on this port. A non-authenticated computer shall still be able to communicate with other ports which are not authenticated. Since the three bits partly forms the address into the L2 Action Table it is possible to form rules which when a packet is allowed to access other ports depending on what the state of these ports are and if the computer it wants to communicate with is known to the switch or not. The field **l2ActionTableDaStatus** can be used to further enhance the security wheather or not two computers shall be able to communicate.

Port Authentication Enhancements with Learning and Port-Move

As the network security needs to be enhanced further the L2 Action Table allows setting up rules if a packet coming in and going to different ports shall be able be able to be learned or if a already existing MAC address shall be able to be port moved.

Port Authentication Enhancements only allow certain traffic types

As the last enhancement there can be special rules formed which allows only certain packet types to pass on a port combination using the result options **useSpecialAllow** and **allowPtr**. This allowPtr points to general rules of which packet types to drop or to allow. This rules are setup in **Allow Special Frame Check For L2 Action Table**.

Chapter 7

Routing

This core supports IPv4 and IPv6 routing as well as MPLS switching.

The routing is disabled by default and needs to be setup from the configuration interface before it can be used. This core supports virtual router ports/functions (VRFs). The VRFs allow the core to handle multiple virtual routers sharing the same set of tables and register. A VRF identifier is used to determine which virtual router each table entry belongs to.

The routing is done separately from the L2 switching. There is no switching done before or after the router. The router is entered when a packets destination MAC address equals the routers MAC address. The packet exits the router directly to an egress port.

MPLS follows the same order of operations as IP routing and uses the same tables. The MPLS processing is therefore described here.

7.1 Order of Operation

Routing function is done after the L2 ACLs. The routing engine performs the following steps:

1. Check if the VLAN allows packets to be routed. If this is not the case normal L2 lookups will be done. This is specified by the **allowRouting** field in **VLAN Table**.
2. Compare the incoming packets MAC destination address with all the entries in the **Router Port MAC Address**.There are per source port option in field **selectMacEntryPortMask** which allows the compared MAC address to be different based on which source port the packet comes in. The alternative MAC address to compare is located in field **altMacAddress**. If no match then the routing function is skipped. If the router port search found a match then the packet enters the router with an assigned VRF from the table.
3. The carried packet type (IPv4, IPv6 or MPLS) is checked against the allowed type that are setup in **Ingress Router Table**. If the type is not allowed the packet will be dropped. There is a alternative to dropping the packets and instead send them to the CPU. This can be archived by setting the **sendToCpuOrDrop** bit to one.
4. If the incoming packets TTL is below the allowed TTL, as specified in **Ingress Router Table** then the packet is dropped.
5. To determine the packets destination/next hop the destination address combined with the assigned VRF is searched for in the **Hash Based L3 Routing Table** and in the **L3 Routing TCAM**. If there is a match in both the TCAM and the hash table then the hash entry is selected since the hash table always contains the longest prefix. For the hash based search the next hop result is setup in the **Hash Based L3 Routing Table** and for the LPM search it is setup in **L3 LPM Result**.

   The difference between MPLS and IP search is that in MPLS the 20-bit MPLS label from the outermost MPLS header is used as destination address.

6. If there is a match in the routing tables and the ECMP is enabled in the matched entry (either the **useECMP** in the **Hash Based L3 Routing Table** or **useECMP** in the **L3 LPM Result** table) then ECMP next hop calculation is performed.

   ECMP calculates a hash based on the IP source and destination addresses, the IP proto field, IP TOS and the TCP/UDP source port and destination port.

   For MPLS the ECMP hash key consists of the outermost header and does not include embedded IP headers. The hash value is added as an offset to the **nextHopPointer** after masking (**ecmpMask**) and shifting (**ecmpShift**).

7. If there is no hit in the destination address search then the default next hop is used. The default is defined in **L3 Routing Default** per VRF. There are also options to drop the packet or send to CPU port.

8. IP statistics is updated in the **IP Unicast Received Counter**, **IP Unicast Routed Counter** and **Received Packets on Ingress VRF** registers. MPLS forwarded packets are only counted in **Received Packets on Ingress VRF**

9. The next hop from the previous steps is used as index into the **Next Hop Table**. The entries determine where to route the packet, which is either a single destination port or a pointer to a L2 multicast entry. There are also options to drop the packet or send to CPU port.

   Each entry also contains a packet modification pointer which points to several tables that determines what header modification that should be done when the packet exits the router.

   - The **Next Hop Packet Modifications** table determines what VLAN operations to perform when exiting the router. If the entry's valid bit is not set the packet will be send to the CPU.
   - The **Next Hop DA MAC** which determines the destination MAC address to use in the outgoing packet.
   - For MPLS the **Next Hop MPLS Table** determines what MPLS header modifications that should be done on the outgoing packets. These are described in detail in the register description and in the **MPLS** chapter.

   The **srv6Sid** flag is the local instantiated SRv6 segment identifier that enables the packet modification on egress to update the IPv6 destination address to the next segment. When hitting the SRv6 segment identifier, a legal segment routing header nneds to be provided, otherwise the packet will be send to the CPU instead.

10. An MTU check, as specified in the **Router MTU Table**, is performed on incoming routed packets. This check is executed by comparing the IPv4 Total Length field with the limit configured in field **maxIPv4MTU**, separately for each destination port and VRF. Similarly, the IPv6 Payload Length field is compared with field **maxIPv6MTU**. If either length field exceeds its respective limit, the packet will be forwarded to the CPU for further processing. Notably, the MTU check is not applied to MPLS packets.

11. When next hop hit status updates are enabled in the **Ingress Router Table** then each time a packet is routed using a **Next Hop Table** entry the corresponding status bit is set in the **Next Hop Hit Status**.

12. The ingress part of routing is now completed. This is followed by other ingress functions such as L3 ACL etc. Finally the packet is queued to one or multiple egress ports.

13. The egress processing of the routed packet performs the packet header modifications. First step is update of the TTL field which is controlled by the **Egress Router Table**.

14. There exists an option called **Next Hop Packet Insert MPLS Header** which enables a outgoing routed packet to add MPLS labels after the L2 / VLAN headers. This allows the router to enter a MPLS tunnel in order to reach the next hop though a MPLS network. If a packet is already a MPLS packet this option offers a way to insert extra MPLS headers on top of the MPLS label stack. NOTE: It is not possible to insert MPLS headers if the packet has a PPPoE header, If the packet is a PPPoE then no MPLS insertion is then carried out.

15. A new L2 header is constructed with a DA MAC from the **Next Hop DA MAC** table. The SA MAC will be the incoming DA MAC. The **Router Port Egress SA MAC Address** allows the user to insert a alternative SA MAC address instead of the normal which should have been the packets DA MAC address. This setting is done per egress port.
16. For the a SA MAC address there exists a option to select a SA MAC address from a list of addresses based on the resulting next hop entry. This can be done by setting the field **useSaTable** to one. Once this is done then the field **saNextHopPtr** is used to read out the table **Router MAC SA Table** which holds the SA MAC address to be used in the routed packet.
17. The routers VLAN operations are performed. See the **VLAN Processing** chapter.
18. The segment routing operations are performed if needed.
    - Decrement Segments Left by 1.
    - Copy Segment List[Segments Left] from the SRH to the destination address of the IPv6 header.
19. The IPv4 header checksum is recalculated.
20. Egress router statistics is updated in **Transmitted Packets on Egress VRF**.
21. Egress VLAN Translation is done using the Dleft lookup 19, on the newly assigned outermost egress VID of the packet.
22. If the result from the **Next Hop Table** points to a tunnel entry using fields **tunnelEntry** then the tunnel entry is carried out after all the packet modifications has been done according to the router exit.
23. If the result from the next **Next Hop Table** points to a tunnel exit using fields **tunnelExit** then the tunnel exit is carried out before the packet is modified by the router. Please note that if the tunnel exit packet modifications are modifying the same fields as the router (SA/DA MAC and VLANs and TTL fields) then these fields will be overwritten by the router.
24. The egress ports VLAN operations are performed. See the **VLAN Processing** chapter.

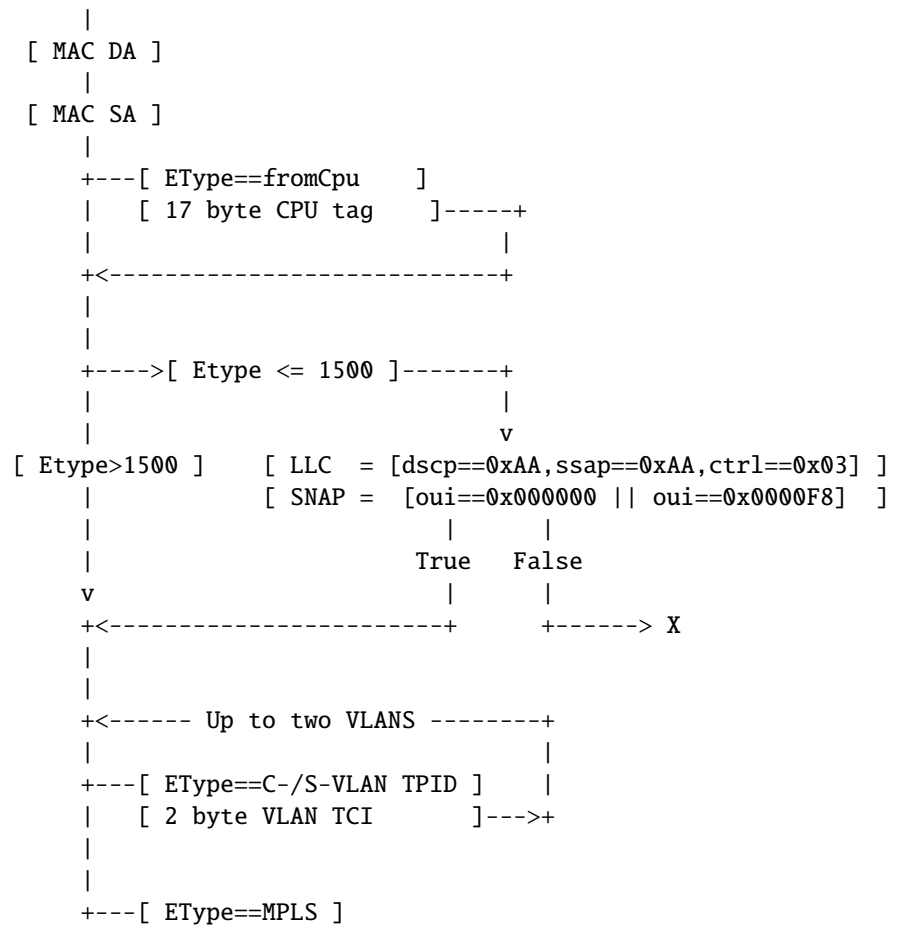
Chapter 8

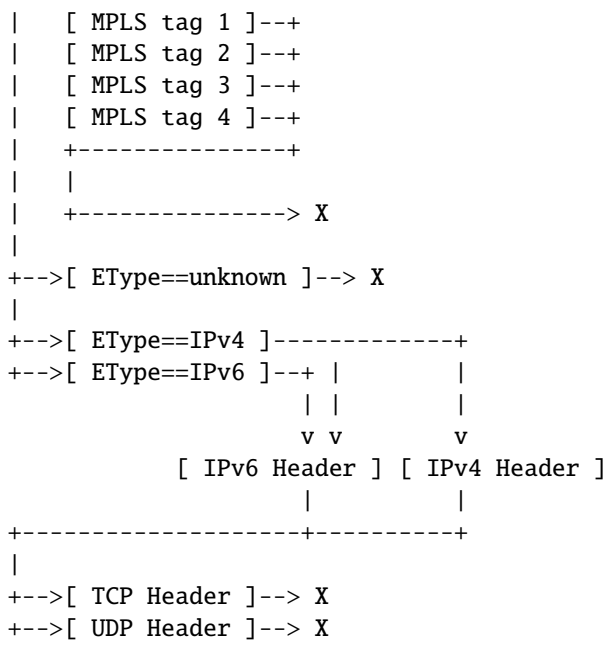
Tunneling

The tunneling has two functions, first the tunnel exit, which enables the user to remove a number of bytes from a incoming packet, then process the packet as the inner layer packet. Secondly enter a tunnel in which a outgoing packet is encapulated with a number of bytes somewhere in the packet.

8.1 Packet Decoder For Tunnel Exit

In the following diagram the decoding of the incoming packet header is described. The comparison used to determine protocol types are described as well as the order they are decoded. The end of decoding process is denote by an *X*.





There are options for the tunnel exit when it comes to reconizing the Ethernet Types for VLANs. The settings are located in register **L2 Tunnel Decoder Setup** which allows the user to setup custom types for C-tagged and S-tagged VLAN packets. If a packet originates from the CPU port and bears the from-CPU-Tag, with the Force Original Bit enabled within this header, there will be no execution of tunnel exit or tunnel entry.

8.2 Tunnel Exit

The tunnel exit can be done in multiple ways. In order for a packet to be enabled do a tunnel exit the field in register **Source Port Table** field **disableTunnelExit** in the incoming source port needs to be set to zero. The packet decoder decribed in 8.1 extracts the relevant fields from the incoming packet:

1. Source MAC address
2. Destination MAC address
3. Packet is a SNAP/LLC Packet
4. Outer VLAN
5. Inner VLAN
6. Ethernet Type Field after possible VLAN headers
7. Ethernet Type for L3
8. If MPLS: Up to 4 MPLS headers.
9. If IPv4 then IPv4 Destination Address
10. If IPv4 then IPv4 Source Address
11. If IPv6 then IPv6 Destination Address
12. If IPv6 then IPv6 Source Address
13. L4 source port, if TCP or UDP packet
14. L4 destination port, if TCP or UDP packet
15. One bit to indicate that the incoming packet had a from CPU tag.

All of these fields are then looked up in the Tunnel Exit Table using the Dleft function described in Dleft Tunnel Exit If the first tunnel exit lookup has a hit then the packet will do a second tunnel lookup which

can result in a tunnel exit. The second lookup is needed because some protocols require a second field to be looked up before a tunnel exit can be determined, example of these types of protocols are VxLAN and GRE-over-UDP. There also exists options which enbles the user to not use the packet data for the second lookup, instead use data from the first lookup answer fields, thereby allowing the first lookup to be the only lookup which matters (second lookup will still be performed but data is controlled from first lookup).

Packets with From CPU Tag

When a packet matches the criteria for tunnel exit and is tagged with a 'from CPU' label, and if the 'force-original-packet' bit is set within this tag, the packet will not undergo tunnel exit. Consequently, any rules configured to forward such packets to the CPU upon a hit in the initial lookup but not in the second tunnel lookup will also not be executed.

Tunnel Exit Places in the packet

The tunnel exit operations can remove configurable number of bytes, a maximum of 192 bytes can be removed, at the following places:

1. At the beginning of the packet.
2. After the DA and SA MAC and the VLAN headers.
3. After the DA and SA MAC, the VLAN headers and IPv4,IPv6 headers .

8.2.1 To Not To Use Second Lookup

To only use the first lookup and not select any new data for the second lookup this is done by setting the direct bit to one (**direct**) and setting up the field tabIndex, This tabIndex field is then used to do the search in the **Second Tunnel Exit Lookup TCAM**. The result in the **Second Tunnel Exit Lookup TCAM Answer** tells the tunnel unit how to remove the data from the packet.

8.2.2 Use Second Lookup With Packet Data

If the user does not want to use the direct method but rather extract new data from the packet then the second lookup field is extracted on a byte boundry which is pointed out by the field **secondShift**, this second shift value can also take into account if the incoming packet has zero,one or two vlans by setting the field **secondIncludeVlan** to one (which will increase the value of **secondShift** with 4 for each VLAN.). For the value used in the second lookup (either from packet data or from result from first table lookup) there exists options if each bit should be used or not in the lookup (using a mask located in field **lookupMask**). The second tunnel exit lookup has a type field which comes from the first Dleft tunnel exit lookup result thereby allowing different tunnel exit types not to get a false positive.

8.2.3 How To Remove Data From Packet In A Tunnel Exit

Once the first and second tunnel exit lookups are done a tunnel exit is performed. How the tunnel exit is done is described by the result from the second tunnel exit lookup. The results are located in tables **Second Tunnel Exit Lookup TCAM** and **Second Tunnel Exit Lookup TCAM Answer**

There are important fields are **howManyBytesToRemove** and **removeVlan** which specifices which bytes to remove in the incoming packet starting from the position after the L2 DA/SA + VLANs. The removeVlan removes 1 or 2 VLANs in the coming packet.

There exists options if the second lookup should fail which allows the user to drop the packet (while the first was tunnel exit lookup was a hit). This is located in **Second Tunnel Exit Miss Action**. This action has a setting for each of the different packet keys which comes from the first tunnel exit lookup result.

The same operations done at ingress during a tunnel exit must be mirrored in the egress register **Egress Tunnel Exit Table** otherwise the packet will look different once its sent out. Which entry to use in the **Egress Tunnel Exit Table** is pointed to by the field **tunnelExitEgressPtr**.

```
Example 1) Remove IPv4 Header

Incoming packet:
+---------+-------+-------------+-------------------------+
| DA | SA | VLAN  | IPv4 Header |  Rest of packet         |
+---------+-------+-------------+-------------------------+
                    |
                    |
                    v
Outgoing packet:
+---------+-----------------------------+
| DA | SA |  Rest of packet              |
+---------+-----------------------------+

In tunnel exit table: SA+DA+VLAN ID+IPv4 SA+IPv4 DA.
Remove from start byte:12 to end byte:72.
The removal is done setting the register howManyBytesToRemove = 20
   and then setting the removeVlan = 1


Example 2) Remove Dual VLANs.

Incoming packet:
+---------+-------------+-------------+-------------------------+
| DA | SA | first VLAN  | Second VLAN |  Rest of packet         |
+---------+-------------+-------------+-------------------------+
                    |
                    |
                    v
Outgoing packet:
+---------+-----------------------------+
| DA | SA |  Rest of packet              |
+---------+-----------------------------+

In tunnel exit table: SA+DA+2 * VLAN Headers
Remove from start byte:12 to end byte:20.
The removal is done setting the register howManyBytesToRemove = 0
   and then setting the removeVlan = 1
```

8.2.4 Packet Insertion and Removal Limits

For the core to operate correctly it needs enough bytes in the first part of the packet. The packet processing gets the first 192 bytes of the whole packet. Once a packet is passed to the egress processing pipeline 34 bytes of the total 192 bytes is consumed by a internal header. For tunnel exit this means that if the inner packet headers (L2+L3+L4) after a tunnel exit goes beyond 192 - 34 bytes minus the tunnel exit removed bytes then this inner packet will be dropped due to insufficient bytes to decode the packet.

8.2.5 Tunnel Exit Options

Besides the above tunnel exit operations there are also a number of other operations which can be done.

- Drop the packet. Using field **dontExit**.
- Set the VLAN table VID which shall be used. Using field **replaceVid**set to one and then setting the VID to be used in field **newVid**
- Do not do the tunnel exit. Using field**dontExit**.

8.2.6 Tunnel Exit from Tables

Tables such as VLAN, L2, L2 Multicastouting,L3 tables and ACLs have option to do a tunnel exit. If this packet already did a tunnel exit then the packet will be sent to CPU since the hardware can not process two tunnel exits after each other, the packet will be sent to the CPU with the reason code 19,473.

8.3 Tunnel Entry

Entering a tunnel allows adding protocol headers at the beginning of the packet, after the L2 headers (After Ethernet MAC DA/SA and VLANs) and finall after L3 (After IPv4/IPv6/MPLS headers, before L4 headers). The After L2 tunnel headers starts with an IP header (IPv4 or IPv6) followed by an optional UDP header. After IP/UDP headers additional headers can be inserted but the content of those headers are not modified by the switch.

The same table address is read out in all of the tunnel entry instructions **Tunnel Entry Instruction Table**, **Beginning of Packet Tunnel Entry Instruction Table**, **L2 Tunnel Entry Instruction Table** and **L3 Tunnel Entry Instruction Table**) pointed to by the tunnel entry pointers from L2,L3,ACL tables.

```
Original egress packet before tunnel header insertion:
 +--------+--------+-------+------------+-------------+-------------+
 | MAC DA | MAC SA | VLAN* | Orig EType | Original L3 | Original L4 |
 +--------+--------+-------+------------+-------------+-------------+

After tunnel insertion at beginning of packet:
 +-------------------------------------------+--------+--------+-------+-----------+--------+--------+
 | (NEW) At Beginnig Tunnel Header           | orig.  | orig.  |       | Original  |Original|Original|
 | MAC DA/SA | VLAN* | IP/MPLS Hdr | UDP | X  | MAC DA | MAC SA | VLAN* |  EType    |  L3    |  L4    |
 +-----------+-------+-------------+-----+----+--------+--------+-------+-----------+--------+--------+

After tunnel insertion After L2 Headers:
 +--------+--------+-------+-----------------------------+-------------+--------+
 +        |        |       |  (NEW)After L2 Tunnel Header  |             |Original|
 | MAC DA | MAC SA | VLAN* | New EType  | IP  | UDP | X    | Original L3 |  L4    |
 +--------+--------+-------+------------+-----+-----------+-------------+--------+

After tunnel insertion After L3 Headers:
 +--------+--------+-------+----------------------+------------------------------+---------+
 |        |        |       | Updates in L3 Header: | (NEW) After L3 Tunnel Header | Original|
 | MAC DA | MAC SA | VLAN* | -L4 Type and IP Length|              X               |   L4    |
 |        |        |       |     (original L3)     |                              |         |
 +--------+--------+-------+----------------------+------------------------------+---------+
```

The content of the inserted protocol headers is configured in the **Tunnel Entry Header Data** table.

The length of the IPv4 header is fixed at 20 bytes. The IPv6 header is 40 bytes. These can be followed by 8 bytes of UDP header.

The tunnel entry is done in the egress processing after all other packet modifications. For example any VLAN operations are done before tunnel headers are inserted. If the packet was routed the Next Hop packet modifications (such as IP header TTL update and MAC DA/SA update) will be done before the tunnel header insertion.

For tunnel entry after L2 the insertion point after Ethernet and VLAN headers is automatically identified and for tunnel entry after L3 headers the insertion point after the L2, L3 headers is automatically identifed.

The tunnel inserted header can be updated with correct Payload Length/ Total Length fields if the fields **Beginning of Packet Tunnel Entry Instruction Table l3Type** is IPv6 or IPv4 or in **L2 Tunnel Entry Instruction Table** field **l3Type** is set to IPv4 or IPv6. For IPv4 header the Header Checksum is calculated based on the configured header but after updating the Total Length field. For these length fields and the checksum field the value stored in the **Tunnel Entry Header Data** is not used.

All other fields in the IP header are unchanged and taken directly from the **Tunnel Entry Header Data**. It is up to the software configuration to create a valid IP header. This includes setting the Protocol/Next Header field if an UDP header follows the IP header.

After the inserted IP/UDP headers can follow additional headers up to the maximum width of the **Tunnel Entry Header Data** (64 bytes).

The tunnel insertion process will always perform the tunnel header insertion if instructed to by table actions in the ingress processing. There is no check at all of the content of the original protocol headers at this point.

For tunnel entry after L3 there exists options in which the preceeding IPv4 or IPv6 headers protocol type / next header byte will be updated. This is controlled by the **L3 Tunnel Entry Instruction Table** field **updateL4Type**. Besides this the IPv4 or IPv6 length field is updated with the header added.

8.3.1 Tunnel Length Insertion

There exists a option to insert a length into the header data. This length field is first inserted ,by overwriting 2 bytes in the insertion data, defined in the **Tunnel Entry Header Data** which is then inserted into the packet. Software needs to make room for the insertion data. There is no extra length added to the insertion data.

Example as follows: (Inserting at beginning of packet, same applies to all other insertions.)

```
Original egress packet before tunnel header insertion:
+--------+--------+-------+------------+-------------+-------------+
| MAC DA | MAC SA | VLAN* | Orig EType | Original L3 | Original L4 |
+--------+--------+-------+------------+-------------+-------------+

Tunnel Header to be inserted.
+-----------------------------------------------+
| Data                                          |
+-----------------------------------------------+

First insertion of length field in Tunnel Header is carried out:
(Data is written over in the Tunnel Header)
+-----------------------------------+------------+
| Data                              |   Length   |
+-----------------------------------+------------+
                                    <-- 2 bytes-->

After tunnel insertion at beginning of packet:
+-----------------------------------+------------+--------+--------+-------+-----------+--------+--------+
| <-------------- Data Inserted ----------------->| orig.  | orig.  |       | Original  |Original|Original|
|                                   |   Length   | MAC DA | MAC SA | VLAN* |  EType    |  L3    |  L4    |
+-----------------------------------+------------+--------+--------+-------+-----------+--------+--------+
```

8.3.2 Tunnel Entry Tables

A packet can enter the tunnel from a number of tables. Each table has a tunnel entry action bit and a pointer into the **Tunnel Entry Instruction Table**, this table is the master table which then determines which of the tunnel entry tables **Beginning of Packet Tunnel Entry Instruction Table** , **L2 Tunnel Entry Instruction Table** or **L3 Tunnel Entry Instruction Table** to use. This is determined by the field **tunnelEntryType**.

In the **Tunnel Entry Instruction Table** is a pointer to the tunnel header to be inserted, the length of the tunnel header. This table also contains a instruction which enables a 2-byte length field to be inserted into the tunnelHeader at any byte position. If this is used the bytes in this position will be overwritten.

1. VLAN Table
   If a packet shall be switched but misses the L2 Destination table and is flooded or broadcasted then the VLAN tables field **tunnelEntry** and field **tunnelEntryPtr** is used to define how packets shall look when going out on each port. Since this is essentially a multicast entry the **tunnelEntryPtr** is the base address and depending on which ports the packet is sent out this port ID is added as the address to read out the **Tunnel Entry Instruction Table** from.
2. L2 Destination Table
   By setting field **tunnelEntry** and field **tunnelEntryPtr** the outgoing packet will enter into a tunnel.
3. L2 Multicast Table
   The field **tunnelEntry** and field **tunnelEntryPtr**. The tunnel entry for the packets are done using the **tunnelEntryPtr** as a base offset and each egress packets instruction pointer will be used as a offset from the base address in the **tunnelEntryPtr**.
4. Next Hop Table
   The field **tunnelEntry** and field **tunnelEntryPtr** points to a tunnel entry instruction.
5. Egress Port Configuration
   The field **tunnelEntry** and field **tunnelEntryPtr** points to a tunnel entry instruction.
6. Result from a Ingress Configutable ACL
   Result can point to a tunnel entry or a tunnel exit. The user can define if this should be done as a unicast or multicast tunnel entry. In a unicast entry all the packets use the same **Tunnel Entry Instruction Table** entry independent of the outgoing port while the multicast entries means that each destination port is used as a offset to the base pointer.

Table	Unicast or Multicast	Comment
L2 Destination Table	Unicast	A L2 Table entry is unicast.
L2 Multicast Table	Multicast	A L2 Multicast Table entry is Multicast.
VLAN Table	Multicast	A VLAN Table entry is Multicast.
Next Hop Table	Unicast or Multicast	A Next Hop Entry is unicast, however it can point to a L2 Multicast Entry.
Ingress ACL Result Tables	Unicast or Multicast	Enables the user to freely select if unicast or multicast.
Egress ACL Result Tables	Unicast or Multicast	Enables the user to freely select if unicast or multicast.

Table 8.1: Tunnel Entry Unicast or Multicast

7. Result from a Egress Configutable ACL
   Result can point to a tunnel entry or a tunnel exit. The user can define if this should be done as a unicast or multicast tunnel entry. In a unicast entry all the packets use the same **Tunnel Entry Instruction Table** entry independent of the outgoing port while the multicast entries means that each destination port is used as a offset to the base pointer.

The tunnel pointers from these tables can be used as unicast or multicast pointers. Multicast means that the tunnel entry can be configured differently for each egress port. When a pointer is of unicast type the pointer value is used to directly index the **Tunnel Entry Instruction Table**.

If a pointer is of multicast type then the destination port number will be added to the pointer before index the **Tunnel Entry Instruction Table**. This allows for using different tunnel headers for different ports.

8.3.3 Priority between Tunnel Exit and Tunnel Entry in Tables

Since the tunnel entry and tunnel exit can be pointed to by several tables what is the priority between them.

- Egress Port Configuration Priority
  If a port has set the **Egress Port Configuration** with tunnel entry or tunnel exit this will take priority over previous set tunnel exit or tunnel entry.
- Egress Port Configuration and Tunnel Exit Lookup
  Between tunnel exit unit and tunnel exit from egress port configuration table then the egress port configuration table takes precedence. This means that what the processing done on ingresss can alter from how the packet will actually look when it is sent out.
- Tunnel Exit Lookup and Tables Tunnel Exit
  If both the tunnel exit lookup and tables tunnel exit says to do a tunnel exit then the packet will be sent to the CPU port for further checks by software.

8.3.4 Tunnel Entry and Routing with MTU check

Since a ACL or IP entry might call upon a packet to enter a tunnel this might mean that the outgoing IPv4 or IPv6 packet might be too long for the next hops MTU. This check can be turned on for each tunnel entry and it will only be checked if a packet is routed. If the packet is over the MTU then it will be removed from the output ports destination masks and a copy will be sent to the CPU. In order for this to work the table **Tunnel Entry MTU Length Check** must be setup to reflect the additional bytes being added to the IP packet headers.

Chapter 9

MPLS

This core is equipped with MPLS forwarding. The processing of MPLS packets follows the same pattern as IP routing, with the major difference that an MPLS header operation (such as push, pop, swap and penultimate pop) can be carried out. Since the order of operation for MPLS is almost identical to IP routing it is described in the **Routing** chapter.

9.1 MPLS Header Operations

In addition to the processing that is done for IP routed packets the MPLS router can perform operations on the MPLS header stack.

The **Next Hop MPLS Table** determines which operation to perform.

- Pop - The outermost MPLS header in the packet is removed.
- Push - A new MPLS header is added to the packet before any previous MPLS headers. The label for the new header and the source for the EXP bits are specified in the table entry.
- Swap/Replace - The outermost MPLS header in the packet is replaced. The label for the new header and the source for the EXP bits are specified in the table entry.
- Penultimate Pop - All MPLS headers (up to as many as supported by the packet decoder, see **Packet Decoding** chapter) are removed from the packet. In addition the Ethernet Type is set to IPv4 or IPv6, see the following section.
- Remapping of EXP bits in the outermost MPLS header. Either use the existing value, use form the table or use a remapping table **Egress Queue To MPLS EXP Mapping Table**.

The **Egress MPLS TTL Table** determines which operation on the TTL field to perform when exiting the VRP, either decrement the TTL or set a new TTL. Each VRP can have their own setting.

9.2 MPLS Penultimate Pop

A normal Pop operation removes one MPLS header but leaves the Ethernet Type unmodified (identifying the packet as still being a MPLS packet).

The Penultimate Pop operation removes all MPLS headers and also updates the packets Ethernet Type. This assumes that the payload in the MPLS packet is an IP packet. The first nibble in the payload is then decoded (see **Packet Decoding** chapter) to determine if the packet is IPv4 or IPv6 and then the Ethernet Type is updated accordingly.

9.3 MPLS Header Insertion To Reach Next Hop

There exists an option called **Next Hop Packet Insert MPLS Header** which enables a outgoing routed packet to add up to MPLS labels after the L2 / VLAN headers. The operation is pointed out by the field **nextHopPacketMod** in table **Next Hop Table**. If a packet is already a MPLS packet this option offers a way to insert extra MPLS headers on top of the MPLS label stack.

NOTE: It is not possible to insert MPLS headers if the packet has a PPPoE header. If the packet is a PPPoE then no MPLS insertion is then carried out.

Chapter 10

NAT - Network Address Translation

There are two functions that can determine if a NAT operation shall be performed, the Configurable Ingress ACL and the Configurable Egress ACL. Each of these point to a NAT operation table that will be performed in egress, **Ingress NAT Operation** and **Egress NAT Operation**.

The ACL pointers points to a base index and the egress port number can be added to this base index. The register **NAT Add Egress Port for NAT Calculation** determines if this shall be done or not, there is one setting for ingress NAT and one setting for egress NAT. This is a global setting.

The Ingress ACL and Egress ACL has independent NAT operation tables and corresponding NAT actions.

An action is one of the following.

- Replace source IP address.
- Replace destination IP address.
- Replace TCP/UDP source port number.
- Replace TCP/UDP destination port number.

The two NAT operations are performed in the order ingress operation first followed by egress operation (in the case where both operations would modify the same packet field).

If the layer 4 type is TCP and IP address or TCP port number is changed then the TCP checksum is recalculated.

If the layer 4 type is UDP and IP address or UDP port number is changed then the UDP checksum is recalculated.

If an IP address is changed then the IP header checksum is recalculated.

When a NAT operation is perform the status registers **Egress NAT Hit Status** and **Ingress NAT Hit Status** are updated.

10.1 Ingress Packet Processing Option

Since the packet operations for NAT is carried out just before the packet is sent out there are cases where the user want the ingress routing and other processes to use the private or public IP address (and/or L4 address). This can be done by setting the **enableUpdateIp** or **enableUpdateL4** fields in one of **Ingress Configurable ACL N Small Table** , **Ingress Configurable ACL N Large Table** and **Ingress Configurable ACL N TCAM Answer** .

10.2 NAT Action Table Check

At the end of the ingress packet processing a NAT port operation check is done. This involves checking all the egress ports NAT state (in register **Egress Port NAT State**) and comparing them to the ingress port NAT state (in field **natPortState**) together with the NAT operations from ingress and egress ACL and if the packet was routed or switched. These five bits are used as a address into the table **NAT Action Table**. For all the egress ports the

packet is going out on the table is checked and if any of the actions are send to cpu or drop this takes precedence and is carried out instead of sending out the packet on the already looked up ports.When to a packet is sent to the CPU from the NAT Action table there are options if the packet should be the original packet or the modified packet, this is setup in **NAT Action Table Force Original Packet**, there is separate setting for each reason code enabling options when using the two different packets to CPU.

The priority of the **NAT Action Table** is as follows: (Only a single action is carried out.)

1. If all actions are No actions the packet is sent to egress.
2. If any action has the Sent to CPU then the packet will be sent to the CPU
3. If any action has drop then all instances are dropped and a counter is updated

Chapter 11

Crypto

This IP supports a number of encryption and decryption functions.

The crypto functions are connected to an internal port in the switch, port number 11, and packets are redirected to this port to be encrypted or decrypted. The crypto port has the same number of queues as a normal port and the same scheduling capabilities as any other port.

An encrypted or decrypted packet is passed through both ingress and egress packet processing twice, once before encryption or decryption and once after.

The crypto modes and authentication modes available are specified in the **Security Association Table** along with the **IPSec Table**. Observe that each **Security Association Table** entry is used either for encrypt or decrypt, which means that for a single bidirectional flow two different entries are needed. Together these two tables defines which crypto and authentication modes to use. The table **Crypto Sequence Numbers** keeps track of the highest seen or last generated sequence number for each flow. More general crypto configuration is available in **Crypto Configuration**.

Packets can be sent to the crypto engine through a number of tables:

- The **Next Hop Table** together with the **IPSec Table**.
- The ingress ACL.
- The egress ACL.

In the tables mentioned above, packets are redirected to the crypto engine by setting the sendToCrypto field to one. The **Security Association Table** entry to use is selected using the associated saPtr field, and the cryptographic operation is selected using the associated cryptoOp field. The encapsulating protocol (AH, ESP or MACsec) to use is selected using the cryptoProto field when using ACLs or the protocol field in the **IPSec Table** when using the **Next Hop Table**.

MACSec encryption and decryption can also be controlled using the **MACsec Port** register. See section 11.3 for more details.

11.1 Encrypt

The following steps are carried out before encryption:

- During the initial packet processing using the tables described above it is determined that the packet shall be encrypted. The encryption method is selected by using a pointer to the **Security Association Table** along with selecting which encapsulating protocol to use.
- A packet which shall be encrypted by the crypto block will first be processed with all the normal packet lookups and modifications, for example VLAN operations, NAT operations, tunnel entry and/or tunnel exit, before it is queued towards the crypto engine.
- The packets destination queue and port(s) are determined using the normal packet flow. This information and the original source port number are propagated to, and through, the crypto engine.

- Before a packet is enqueued towards the crypto engine the buffer memory status is checked by the Resource Limiter (described in chapter 29). If there is no room for these packets in the buffer memory, then the packet is dropped.
- If traffic to the crypto engine exceeds the crypto processing performance a queue will start to build and eventually packets will be dropped.
- After all packet modifications before the crypto engine, the new packets length is calculated. If the length of the packet exceeds 16,387 bytes or if the packet is smaller than 60 bytes, then the packet will be dropped and the drop counter **Minimum and Maximum Packet Size Drops** will be updated.

The crypto engine will then encrypt the packet according to the **Security Association Table** and the crypto operation. The packet is modified and additional headers are added to the packet.

- For AH authentication, an AH header is added and the IP header is updated along with the checksum in IPv4 packets.
- For ESP encryption, an ESP header is added and the IP header is updated along with the checksum in IPv4 packets.
- For MACsec encryption, a MACsec header is added after the number of VLANs specified in the **MACsec Vlan** register.

The initialization vector needed for ESP and AH is generated by a Linear Feedback Shift Register (LFSR) which is updated by hardware for each encrypted packet. The LFSR can be accessed from software through the **Linear Feedback Shift Register**.

The following steps are carried out after encryption:

- The now encrypted packet from the crypto engine is injected into the ingress packet processing block using the original source port number and the destination port(s) and queue determined by the pre crypto processing.
- Information that the packet has been encrypted, and the **Security Association Table** entry used is provided to be used in the ingress/egress ACLs to do additional packet modifications on the encrypted packet. For instance a new header can be added by using the tunnel entry action in the ACL.
- The packet is queued towards the original destination port(s), modified again in the egress processing if this has been requested, and then sent out.

11.2 Decrypt

The following steps are carried out before decryption:

- During the initial packet processing using the tables described above it is determined that the packet shall be decrypted. The decryption method is selected by using a pointer to the **Security Association Table** along with selecting which encapsulating processing to use.
- The packets destination queue and port(s) are determined using the normal packet flow. This information and the original source port number are propagated to, and through, the crypto engine. If the fields that determine the egress queue are in the encrypted part of the packet, the egress queue and port selection must be done after the decryption.
- Before a packet is enqueued towards the crypto engine the buffer memory status is checked by the Resource Limiter (described in chapter 29). If there is no room for these packets in the buffer memory, then the packet is dropped.
- If a traffic to the crypto engine exceeds the crypto processing performance a queue will start to build and eventually packets will be dropped.
- After all packet modifications before the crypto engine, the new packets length is calculated. If the length of the packet exceeds 16,387 bytes or if the packet is smaller than 60 bytes, then the packet will be dropped and the drop counter **Minimum and Maximum Packet Size Drops** will be updated.

The crypto engine will then decrypt the packet and do packet modifications. These modifications includes removing ESP, AH and MACsec headers.

- For AH packets the AH header is removed and inside the IP header the original L4 packet type is put into the IP header. For IPv4 the checksum is recalculated.

- For ESP packets the ESP header is removed and inside the IP header the original L4 packet type is put into the IP header. For IPv4 the checksum is recalculated.
- For MACsec packets the MACsec header is removed and the original Ethernet Type header becomes visible.

During decryption an anti replay check is performed, and if it fails the packet is dropped or sent to the CPU with an unique reason code.

For IPSec, an anti replay window of 128 packets per security association is used. It keeps track of which of the last 128 sequence numbers, counted from the highest sequence number received, have already been received. If a packet with a sequence number already received, or a sequence number outside the replay window is received, it will be dropped or sent to the CPU.

For MACsec, the anti replay window is a programmable range counted back from the highest sequence number received. Any packet with a sequence number outside this window will be dropped or sent to the CPU. I.e. with a sequence number smaller than the largest seen minus the replay window size.

The following steps are carried out after decryption:

- The now decrypted packet form the crypto engine is injected into the ingress packet processing block using the original source port number and the destination port(s) and queue determined by the pre crypto processing.
- Normal packet processing is done on the packet as if it came in on the original source port.

11.3 MACsec

The MACsec functionality can be enabled using ACLs and the **MACsec Port** register.

The former is used for identifying unique encrypted flows, based on e.g. source port or MAC address.

The latter is used to enforce MACsec encryption of specific Ethernet packet types on one single statically selected port.

ACLs must always be used when decrypting MACsec packets, while encryption can be controlled by both ACLs and the **MACsec Port** register.

When the **MACsec Port** functionality is **enabled**, packets going to the egress port **portId** are by default encrypted using the security association **encryptSaPtr**. This can be overridden by using the cancelCrypto ACL action or by overriding the crypto operation using an ACL crypto action.

The field **dropNonEncryptedPackets** in **MACsec Port** can be used to make sure that only packets which has a MACsec tag are allowed on a port (with the exceptions specified in **MACsec Port**). The drop counter used is **MACsec Drops**).

For VLANs and MACsec there exists options to either put the MACsec header first or after one, two or three VLANs. This is specified in **MACsec Vlan**. SNAP headers will always be before the MACsec header.

11.4 Special Events

There are a number of different special events / errors which can occur during the encryption/decryption process. If one of these events occur then the packet is either sent to the CPU along with an unique reason code or dropped (as specified by the **drop** field in the **Crypto Configuration** register). The special events are listed below.

- No Security Association
  The IPSec SPI or MACsec SCI in the packet does not match the entry in the **Security Association Table**, or the MACsec SECTAG in the packet has an illegal value.
- Fragment
  The packet has the fragmentation bits set in its IPv4 header.
- Sequence Number Overflow
  The sequence number counter has overflowed and wrapped.
- Anti Replay
  The Anti Replay check has failed.
- Integrity Check
  The packet failed the integrity check.

11.5 Packet Operations Before and After Crypto Operations

The packet processing which shall be done before and after a crypto operation is determined by the functional control setting registers. For details see the **Functional Control** chapter.

11.6 Crypto Performance

The following table shows the crypto performance for different packet sizes. Each table entry is of the form S:P where S is the packet size in bytes prior to encryption or decryption and P is the performance in bits per cycle. Packet processing speed is the same from packet sizes over 800 bytes as 800 byte packets.

Algorithm	Encrypt S	Decrypt S	Encrypt M	Decrypt M	Encrypt L	Decrypt L
ah-md5	60 : 1.38	80 : 1.80	300: 3.99	300: 3.97	800: 5.13	800: 5.12
ah-sha1	60 : 1.12	80 : 1.47	300: 3.29	300: 3.27	800: 4.26	800: 4.25
ah-sha256	60 : 1.38	84 : 1.89	300: 3.59	304: 4.02	800: 5.12	804: 5.15
ah-sha384	60 : 1.38	92 : 2.08	300: 4.20	302: 4.25	800: 6.27	802: 6.30
ah-sha512	60 : 1.36	100: 2.26	300: 4.19	300: 4.24	800: 6.26	800: 6.31
ah-sm3	60 : 1.38	84 : 1.89	300: 3.59	304: 4.02	800: 5.12	804: 5.15
esp-3des-cbc-md5	60 : 1.71	86 : 2.31	300: 3.51	302: 3.40	800: 4.22	806: 3.86
esp-3des-cbc	60 : 5.71	74 : 7.22	300: 5.06	306: 4.74	800: 5.00	810: 4.44
esp-3des-cbc-sha1	60 : 1.40	86 : 1.90	300: 3.21	302: 3.12	800: 4.05	806: 3.72
esp-aes128-cbc-hmac-sha384	60 : 1.39	114: 2.50	300: 4.54	306: 4.25	800: 7.26	802: 6.74
esp-aes128-cbc-hmac-sha512	60 : 1.38	122: 2.67	300: 4.52	314: 4.36	800: 7.25	810: 6.81
esp-aes128-cbc	60 : 8.57	90 : 10.00	300: 10.86	314: 8.91	800: 11.39	810: 8.70
esp-aes128-ccm	60 : 10.67	82 : 10.41	300: 17.78	302: 11.96	800: 19.94	802: 12.51
esp-aes128-gcm-esn	60 : 10.21	90 : 11.43	300: 20.34	302: 13.35	800: 23.36	802: 13.92
esp-aes128-gcm	60 : 10.21	90 : 11.43	300: 20.34	302: 13.35	800: 23.36	802: 13.92
esp-aes128-gmac-esn	60 : 10.00	90 : 11.08	300: 19.51	302: 13.20	800: 22.61	802: 13.71
esp-aes128-gmac	60 : 10.00	90 : 11.08	300: 19.51	302: 13.20	800: 23.02	802: 13.86
esp-aes256-cbc-hmac-md5	60 : 1.40	102: 2.23	300: 4.44	310: 4.05	800: 6.04	806: 5.13
esp-aes256-cbc-hmac-sha1	60 : 1.13	102: 1.83	300: 3.60	310: 3.35	800: 4.86	806: 4.26
esp-aes256-cbc-hmac-sha256	60 : 1.39	106: 2.32	300: 4.44	314: 4.10	800: 6.03	810: 5.15
esp-aes256-cbc	60 : 7.50	90 : 9.00	300: 8.30	314: 7.26	800: 8.51	810: 6.95
esp-aes256-ccm	60 : 9.23	82 : 9.24	300: 13.95	302: 10.15	800: 15.24	802: 10.50
esp-aes256-gcm-esn	60 : 9.41	90 : 10.14	300: 15.38	302: 11.24	800: 17.11	802: 11.52
esp-aes256-gcm	60 : 9.41	90 : 10.14	300: 15.38	302: 11.24	800: 17.11	802: 11.52
esp-aes256-gmac-esn	60 : 8.28	90 : 9.60	300: 14.72	302: 11.03	800: 16.54	802: 11.30
esp-aes256-gmac	60 : 8.28	90 : 9.60	300: 14.72	302: 11.03	800: 16.84	802: 11.44
esp-des-cbc-md5	60 : 1.71	86 : 2.31	300: 3.51	302: 3.40	800: 4.22	806: 3.86
esp-des-cbc	60 : 5.71	74 : 7.22	300: 5.06	306: 4.74	800: 5.00	810: 4.44
esp-des-cbc-sha1	60 : 1.40	86 : 1.90	300: 3.21	302: 3.12	800: 4.05	806: 3.72
esp-sm4-cbc	60 : 7.06	90 : 8.57	300: 7.43	314: 6.65	800: 7.53	810: 6.31
esp-sm4-cbc-sm3	60 : 1.39	106: 2.32	300: 4.34	314: 4.10	800: 5.83	810: 5.15
esp-sm4-ccm-esn	60 : 8.57	82 : 8.75	300: 12.57	302: 9.44	800: 13.62	802: 9.72
esp-sm4-ccm	60 : 8.57	82 : 8.75	300: 12.57	302: 9.44	800: 13.62	802: 9.72
esp-sm4-gcm-esn	60 : 8.73	90 : 9.60	300: 13.87	302: 10.41	800: 15.09	802: 10.60
esp-sm4-gcm	60 : 8.73	90 : 9.60	300: 13.71	302: 10.41	800: 15.09	802: 10.60
macsec-aes128	60 : 10.67	80 : 10.85	300: 20.00	300: 13.04	800: 22.94	800: 13.79
macsec-aes128-xpn	60 : 10.67	80 : 10.85	300: 20.00	300: 13.04	800: 22.94	800: 13.79
macsec-aes256	60 : 8.73	80 : 9.28	300: 15.00	300: 10.81	800: 16.71	800: 11.35
macsec-aes256-xpn	60 : 8.73	80 : 9.28	300: 15.00	300: 10.81	800: 16.71	800: 11.35

Chapter 12

Mirroring

This core supports both input and output mirroring.

12.1 Input Mirroring

Input mirroring allows all packets received by an ingress port to be copied to an egress port without packet modifications.

- For each port, one input mirroring port can be configured through the **Source Port Table**. The **inputMirrorEnabled** field enables a input mirror copy and send it to the port configured in the **destInputMirror** field.
- Packets hit in the **Configurable ACL Engine** can send an input mirror copy to the port configured in ACL's **`destInputMirror`** field if there is an enabled **`inputMirror`** action.

By default the input mirror copy will bypass any packet modification or drop decisions during the ingress or egress packet processing. Extra options are given in the **Source Port Table** to limit the range of the mirroring destination. **imUnderVlanMembership** only allows the input mirror copy to be sent to the members of the VLAN. **imUnderPortIsolation** only allows the input mirror copy to be sent to the destination that does not block the current source port from the **Ingress Egress Port Packet Type Filter**. If a packet has an input mirror action from the ACL and its source port also enables input mirroring, the destination port of that copy is determined by the ACL result.

12.2 Output Mirroring

Output mirroring allows the user to select an egress port to be mirrored so that packet that is transmitted to that egress port can have a copy sent to an egress port. For each port, one output mirroring port can be configured through the **Output Mirroring Table**:

1. The output mirroring functionality can be enabled per port using the **outputMirrorEnabled** field from the **Output Mirroring Table**.
2. The port to which the mirror copy is sent is setup by the **outputMirrorPort** field in the **Output Mirroring Table**. Multiple input ports can use the same output mirroring destination port.

With input mirroring, a port can be used to observe the traffic received by any port. With output mirroring, a port can be used to observe the traffic transmitted from any port. When there are multiple mirror copies requested or the CPU port is involved, the switch works as follows:

- An input mirrored packet can be output mirrored again.
- An output mirrored packet will not be mirrored again even if the destination port has output mirroring turned on.
- When a packet is mirrored to the CPU port, it will not carry an extra to-CPU tag since it is the copy of another packet.

It is possible that a packet is sent out in multiple copies on the same port when mirroring is turned on. In this case at most four instances of the same received packet can appear on an egress port. The order of the packet instances will be:

1. Normal switched/routed packet
2. Input mirror copy
3. Output mirror copy of the switched/routed packet
4. Output mirror copy of the input mirror copy

12.2.1 Requeueing FIFO

Output mirroring (and input mirroring to oneself) is accomplished by requeuing the packets in separate requeueing FIFOs after External Packet Processing. There is one requeue FIFO per egress port.

The egress scheduling will only see the packet at the head of each FIFO, but this packet will be selected before the packets belonging to the same queue in the normal egress queues.

This method of output mirroring means that:

1. The requeuing FIFOs are truly FIFOs per port, so there will be head-of-line blocking between packets of different egress queues mirrored to the same port.
2. The (up to three) mirroring copies for a single input packet are created in series. The first one is not created until the original packet has been scheduled and gone through Egress Packet Processing, the second one not until the first copy has been scheduled and gone through Egress Packet Processing and so on...
3. When several ports output mirror to the same port, or a higher speed port mirrors to a lower speed port (physical or shaped port speed) the requeueing FIFO for the mirroring destination port may fill up and cause packet drops.

The depth of the requeueing FIFOs is ten packets per egress port.

Drops due to the requeueing FIFOs overflowing are counted in the **Re-queue Overflow Drop** register.

Chapter 13

Link Aggregation

Link aggregation is a solution to bundle multiple ports into a higher bandwidth link. Each link aggregate is setup using the **Link Aggregation Membership** and **Link Aggregation To Physical Ports Members**.

The **Link Aggregation Membership** register maps the incoming packets source port number to a link aggregate number. The link aggregate number is then used during ingress packet processing instead of source port/destination port numbers.

When a destination port (destination link aggregate number) has been determined by ingress packet processing the **Link Aggregation To Physical Ports Members** table maps the link aggregate number to which physical ports that are part of the link aggregate, i.e. the physical ports the packet shall be transmitted to.

Note that once link aggregation is enabled all ports needs to be setup as link aggregates, even if a port only has a single port part of its link aggregate. These ports are usually setup as having a one-to-one mapping, i.e. source port number, link aggregate number and physical port number are all the same.

The **Link Aggregation Membership** register and the **Link Aggregation To Physical Ports Members** register must be kept in sync by software.

To distribute the packets over the ports that are part of a link aggregate, a hash is calculated over some of the packets fields which is configured by register **Link Aggregation Ctrl**. The hash value calculated is used to index the **Link Aggregate Weight** table which results in a port mask of the ports that will be used for this specific hash.

The ratio that each port in a link aggregate is used is determined by the number of times the port is set in the **Link Aggregate Weight** table divided by the number of entries in the table.

It is important to setup all entries in the **Link Aggregate Weight** table with one port set for each link aggregate, otherwise a certain hash value will have no port set thereby causing the packet to be dropped.

13.0.1 One-to-one Port Mapping

To setup a one-to-one mapping, then the bit which corresponds to the port number shall be set in the **members**. This maps each link aggregate number to a physical port with the same number.

The **la** should then be set so that each source port number maps to the link aggregate with the same number, i.e. table entry 0 should hold a value of 0, table address 1 should hold a value 1, etc.

13.1 Example

Lets say that a link aggregate shall use physical ports 0,1,2 and each port shall have equal amount of traffic. Another link aggregate will use ports 6,7 also with equal load between the ports. The remaining ports are setup to be one-to-one. In this example these are ports 3,4 and 5, on a switch with 8 ports.

To setup the **Link Aggregation Membership** register we associate the source port with the link aggregate number that it belongs to. Ports 0,1,2 are part of link aggregate 0 and port 6,7 are part or link aggregate 1. The remaining ports are setup to use the same link aggregate number as the port number.

```
for port in [0,1,2]:
  rg_sp2la[port] = 0

for port in [6,7]:
  rg_sp2la[port] = 1

for port in [3,4,5]:
  rg_sp2la[port] = port
```

In **Link Aggregation To Physical Ports Members** we need to setup the relation from link aggregate number to physical port members.

```
  rg_la2Phy[0] = 0b00000111  # la #0 = ports 0,1,2
  rg_la2Phy[1] = 0b11000000  # la #1 = ports 6,7
  rg_la2Phy[3] = 0b00001000  # la #3 = port 3
  rg_la2Phy[4] = 0b00010000  # la #4 = port 4
  rg_la2Phy[5] = 0b00100000  # la #5 = port 5
```

To setup how the traffic is distributed between the link aggregate member ports we first select which packet headers that will be used in the hash calculation. In this example we chose to select source MAC, destination MAC, IP addres, L4, TOS value and vlan header as calculation base for the hash value.

```
  rg_linkAggCtrl.useSaMacInHash = 1
  rg_linkAggCtrl.useDaMacInHash = 1
  rg_linkAggCtrl.useIpInHash = 1
  rg_linkAggCtrl.useL4InHash = 1
  rg_linkAggCtrl.useTosInHash = 1
  rg_linkAggCtrl.useVlanInHash = 1
```

The table **Link Aggregate Weight** shall then be setup so that ports 0,1,2 have equal weight. This is accomplished by configuring so that the number of bits set for port 0 in all hash entries are equal to number of bits for port 1 and port 2. Which bits are set are not important as long as only one bit per entry are set and the total number of bits per port are equal.

If the hash of the packets fields are distributed evenly then $1/3$ of the packets will be distributed to each of the three ports part of the link aggregate.

Similarly to setup a link aggregate on ports 6,7 with equal load between the ports then each entry in the **Link Aggregate Weight** table must have bit 6 or 7 set and with equal number of bits for the two ports.

The ratio for link aggregation 0, is 34% on port 0, 33% on port 1 and 33% on port 2. For link aggregation 1, it is 50% on each port.

```
  for hash_index in range(0,85):        # 34%
    r_hash2LA[hash_index] = 0b00000001  # port 0
  for hash_index in range(86,170):      # 33%
    r_hash2LA[hash_index] = 0b00000010  # port 1
  for hash_index in range(171,256):     # 33%
    f_hash2LA[hash_index] = 0b00000100  # port 2

  for hash_index in range(128):         # 50%
    r_hash2LA[hash_index] |= 0b01000000 # port 6
  for hash_index in range(128,256):     # 50%
    r_hash2LA[hash_index] |= 0b10000000 # port 7

  for hash_index in range(256):         # 100%
    r_hash2LA[hash_index] |= 0b00001000 # port 3
    r_hash2LA[hash_index] |= 0b00010000 # port 4
    r_hash2LA[hash_index] |= 0b00100000 # port 5
```

Finally when all the registers have been configured the link aggregation function is enabled in the **Link Aggregation Ctrl** register.

```
rg_linkAggCtrl.enable = 1
```

13.2 Hash Calculation

The hash key consists of the following fields in the order listed starting with the msb.

- MAC DA, 48 bits
- MAC SA, 48 bits
- VLAN ID, 12 bits
- IP TOS, 8 bits
- TCP/UDP Source Port, 16 bits
- TCP/UDP Destination Port, 16 bits
- IP Proto, 8 bits
- IPv4/IPv6 Source Address, 128 bits
- IPv4/IPv6 Destination Address, 128 bits
- Source Port, 4 bits

If a field is disabled in the **Link Aggregation Ctrl** register then the field in the hash key will be 0.

If a packet is routed then the MAC DA field will contain the next hop pointer instead of the MAC address and the VLAN ID will be 0.

The hashing is done in two steps, first the key is build, and the fields used in the key depends on the **Link Aggregation Ctrl** register, once the key is build then hash function is used to determine the address used ot lookup the **Link Aggregation To Physical Ports Members**.

```
def build_key(daMac, useDaMacInHash,
              saMac, useSaMacInHash,
              vlanId, useVlanIdInHash,
              tos, useTosInHash,
              sp, useL4InHash,
              dp,
              proto,
              salp, useIpInHash,
              dalp,
              nextHop, useNextHopInHash,
              srcPort, routed):
    # This function builds the key to be
    #   used for calculating the hash.
    final_data = 0
    if useDaMacInHash==0:
        daMac = 0
    if useNextHopInHash==0:
        nextHop = 0
    if routed==1:
        daMac = nextHop
        vlanId = 0

    final_data = final_data <<48
    final_data = final_data | daMac
    final_data = final_data <<48
    if useSaMacInHash==1:
        final_data = final_data | saMac
    final_data = final_data <<12
    if useVlanIdInHash==1:
```

```
        final_data = final_data | vlanId
    final_data = final_data <<8
    if useTosInHash==1:
        final_data = final_data | tos
    final_data = final_data <<16
    if useL4InHash==1:
        final_data = final_data | sp
    final_data = final_data <<16
    if useL4InHash==1:
        final_data = final_data | dp
    final_data = final_data <<8
    if useL4InHash==1:
        final_data = final_data | proto
    final_data = final_data <<128
    if useIpInHash==1:
        final_data = final_data | salp
    final_data = final_data <<128
    if useIpInHash==1:
        final_data = final_data | dalp
    final_data = final_data <<4
    final_data = final_data | srcPort
    return final_data


def calcLaHash( key ):
    mask = (1 << 8) - 1
    _hash = 0
    for j in range(52):
        _hash     = _hash ^ (key & mask)
        key = key >> 8
    return _hash & mask
```

Chapter 14

IEEE 1588/PTP Support

The core has support for IEEE 1588 / PTP with a number of features.

- Transfer of timestamp from RX MAC to CPU in the **To CPU Tag**.
- Identify PTP packets and send to CPU.
- Control of TX MAC action from settings in the **From CPU Tag**.
- Transfer of timestamp in the **From CPU Tag** to the TX MAC.
- Provide position of packet fields to the TX MAC needed for timestamp operation.

14.1 Timestamp from RX MAC

Each ingress port can receive a timestamp at the end of the packet. When the ingress port receives the end of the packet from the MAC, a timestamp valid flag indicates whether the packet is timestamped.

The timestamp size is 8 bytes.

14.1.1 Timestamp to the CPU

The RX MAC timestamp will be transferred to the CPU in the **Timestamp** field of the **To CPU Tag**. This will only be done when the packet is identified as a PTP packet by setting the ptp bit and the packet is sent to the CPU port with a **To CPU Tag**. For all other packets the timestamp will be discarded.

If redirecting to the CPU with ptp bit set without having a timestamp header on the source port will result in an invalid timestamp field in the **To CPU Tag** header.

14.2 PTP Frame Decoding

The switch supports PTP packets embedded in an 802.3 Ethernet frame, in an UDP/IPv4 frame or in an UDP/IPv6 frame.

PTP Header Field		byte position
transportSpecific	messageType	byte 0
reserved	versionPTP	byte 1
...	...	byte 2-6
correctionField		byte 8-15
...	...	byte 16-33
originTimestamp		byte 34-43

Table 14.1: PTP Header Format

MAC DA	MAC SA	EtherType=0x88F7	PTP

Table 14.2: PTP over 802.3 Ethernet

14.2.1 PTP over 802.3 Ethernet

The packet decoder identifies PTP packets embedded in 802.3 Ethernet frames by the Ethernet Type. There is no comparison of the Ethernet destination address.

In order to be sent to the CPU any function (except input mirroring) that sends to the CPU port can be used. For example the 1588 standard multicast group addresses (01-1B-19-00-00-00, 01-80-C2-00-00-0E) can be set in the **L2 Destination Table** and point to entries in the **L2 Multicast Table**. For the link local multicast (01-80-C2-00-00-0E) that should be dropped by bridges, only the CPU port should be set in the **mcPortMask**. For the general multicast group address (01-1B-19-00-00-00) that should be broadcasted, then set all ports including the CPU port in the mask.

The **ptp** bit in the **To CPU Tag** will be set when the Ethernet Type matches the PTP type.

14.2.2 PTP over UDP

MAC DA	MAC SA	EtherType	IPv4	UDP	PTP

Table 14.3: PTP over UDP/IPv4

MAC DA	MAC SA	EtherType	IPv6	UDP	PTP	Checksum Correction

Table 14.4: PTP over UDP/IPv6

PTP embedded in IPv4/IPv6 UDP can be identified with an L3 ACL rule and sent to the CPU using the sendToCpu action. The ptp action must also be set in order for the **ptp** bit in the **To CPU Tag** to be set together with a valid Timestamp field.

14.3 Software Control of TX MAC PTP Actions

The TX MAC needs to perform the following PTP actions.

- TX MAC updates timestamp in outgoing packet.
- TX MAC produces timestamp to be read by software.
- TX MAC updates correction field in outgoing packet with current time minus software time from the timestamp in the **From CPU Tag**.

These actions are controlled by software by sending PTP packets from the CPU port with a **From CPU Tag**. In the **From CPU Tag** header there are fields that will be transferred directly to the transmit MAC on dedicated signals (see **Packet Interface**).

- *oupd_ts_ps_***N** - this signals will be set when the From CPU Tag field **upd_ts** is set. This is used to tell the transmit MAC that it should update the packets originTimestamp field.
- *oupd_cf_ps_***N** - this signals will be set when the From CPU Tag field **upd_cf** is set. This is used to tell the transmit MAC that it should update the correctionField.
- *ots_ps_***N** - this signal will have the value of the **From CPU Tag ptp_ts** field and should be used by the transmit MAC when updating the correctionField.
- *ots_to_sw_ps_***N** - this signal will have the value of the **From CPU Tag ts_to_sw** field. This is used to tell the transmit MAC that it should create a timestamp of the current packet and transfer the timestamp to software. The switch is not involved in the transfer of the timestamp to software.

14.3.1 Packet Updates by the Transmit MAC

When the transmit MAC updates a PTP packet it needs to know position of the fields in the packet. This information is decoded by the switch and passed to the transmit MAC on dedicated ports.

- IPv4/UDP checksum field.
- IPv6/UDP checksum correction field (last 2 bytes in IPv6/UDP packet).
- PTP originTimestamp field.
- PTP correctionField.

When the transmit MAC updates a PTP packets and PTP is embedded in UDP/IP then the UDP checksum needs to be updated.

- For IPv4/UDP packets the UDP checksum field is zeroed by the MAC and therefore needs the position of the UDP checksum field.
- For IPv6/UDP it is forbidden to use zero checksum. Instead the last two bytes of the PTP packet is used to correct the checksum. The MAC therefore needs position of the UDP checksum field and the position of the second-to-last byte of the packet. (see IETF RFC 7821 - UDP Checksum Complement)

The transmit MAC also needs the position of the originTimestamp and correctionField. The position of the originTimestamp is provided to the MAC and from that position the MAC can calculate the position of the correctionField since that is always in the same relative position.

All this information is transferred to the MAC on dedicated signals (see **Packet Interface**).

- *oudp4_ps_***N** - when this is set the packet is a UDP packet encapsulated in IPv4.
- *oudp6_ps_***N** - when this is set the packet is a UDP packet encapsulated in IPv6.
- *oudp_csum_ps_***N** - this is the first byte of the UDP Checksum field relative to the first byte of the packet.
- *ots_pos_ps_***N** - this is the first byte of the originTimestamp field in a PTP packet relative to the first byte of the packet. This position is correct for all three encapsulation types.
- *oudp_corr_ps_***N** - this is the first byte of the UDP checksum correction field. This field is always the last two bytes of the packet.

14.4 Support for Ordinary Clock

In this section is described how to implement the PTP packet handling for Ordinary Clock mode.

14.4.1 Master sending Sync

Software sends a PTP Sync packet to the CPU port with a **From CPU Tag**. In the **From CPU Tag** the destination port (or ports) are set and the control needed for the TX MAC connected to the egress port are included.

In 1-step mode the outgoing frames timestamp field is updated by the MAC with the timestamp. The timestamp is not used by software.

The TX MAC will get the position of the timestamp field from the switch.

If the packet is an IP/UDP packet then the checksum needs to be update by the MAC since the PTP header is changed. The MAC will get the position of the checksum field from the switch.

If PTP is embedded in IPv4/UDP then the UDP checksum field is cleared by the MAC. If it's IPv6/UDP then UDP checksum is not allowed to be cleared and instead the last two bytes in the frame is padding used for checksum adjustment. The MAC will get the position of the checksum adjustment field from the switch.

In 2-step mode the timestamp from the TX MAC is read out by software and the outgoing frame is not modified by the MAC. The **From CPU Tag** must control the MAC to produce a timestamp for software.

14.4.2 Slave receiving Sync

The RX MAC timestamps all packets. The timestamp must be prepended to the frames before they enter the switch. The switch port must be configured to receive the prepended timestamp.

Software needs to configure the switch to direct the Sync frame to the CPU port with a **To CPU Tag**. The ptp bit must be set so that the timestamp that was prepended to the frame is sent to the CPU in the **To CPU Tag**.

14.4.3 Slave sending DelayReq

Software sends a PTP DelayReq packet to the CPU port with a **From CPU Tag**. In the **From CPU Tag** the destination port (or ports) are set and the control needed for the TX MAC connected to the egress port.

The TX MAC must produce a timestamp of this packet. The timestamp from the TX MAC is read out by software and the outgoing frame is not modified by the MAC.

14.4.4 Master receiving DelayReq

The hardware mechanisms used are exactly as in Slave receiving Sync.

14.4.5 Master sending DelayReply

Software sends a PTP DelayReply packet to the CPU port with a **From CPU Tag**. In the **From CPU Tag** the destination port (or ports) are set.

There is no timestamp needed for this frame so the TX MAC is not directed to produce any timestamp.

14.4.6 Slave receiving DelayReply

Software needs to configure the switch to direct the DelayReply frame to the CPU port. The timestamp produced by the RX MAC is not used and the **To CPU Tag** therefore does not need to include the timestamp.

14.5 Support for 1-step Peer to Peer

14.5.1 Initiator sending PDelayReq

Software sends a PTP PDelayReq packet to the CPU port with a **From CPU Tag**. In the **From CPU Tag** the destination port (or ports) are set and the control needed for the TX MAC connected to the egress port.

The TX MAC must produce a timestamp of this packet. The timestamp from the MAC is read out by software and the outgoing frame is not modified by the MAC.

14.5.2 Peer receiving PDelayReq

The hardware mechanisms used are exactly as in Slave receiving Sync.

14.5.3 Peer sending PDelayResp

Software sends a PTP PDelayReq packet to the CPU port with a **From CPU Tag**. In the **From CPU Tag** the destination port (or ports) are set and the control needed for the TX MAC connected to the egress port.

The TX MAC must produce a timestamp of this packet.

In 1-step mode the outgoing frames correction field is updated by the MAC with the difference between the produced timestamp and software supplied timestamp (from a received PDelayReq). The produced timestamp is not used by software. The TX MAC will get the position of the correction field from the switch.

14.5.4 Initiator receiving PDelayResp

Software needs to configure the switch to direct the PDelayResp frame to the CPU port. The ptp bit must be set so that the timestamp that was prepended to the frame is sent to the CPU in the **To CPU Tag**.

Chapter 15

Classification

15.1 L2 Classification

- L2 Destination MAC range classification is setup in table **Reserved Destination MAC Address Range**.
  - The table is searched starting from entry 0.
  - When a range is matched the corresponding actions (drop, send to cpu, force egress queue) will be activated.
  - If multiple ranges are matched, any matching range that sets drop will cause a drop.
  - Any match that sets sendToCpu will cause send to CPU (this has priority over drop).
  - When multiple ranges that match has set the forceQueue then the highest numbered entry will determine the value.
- L2 Source MAC range classification is setup in table **Reserved Source MAC Address Range**.
  - The table is searched starting from entry 0.
  - When a range is matched the corresponding actions (drop, send to cpu, force egress queue) will be activated.
  - If multiple ranges are matched, any matching range that sets drop will cause a drop.
  - Any match that sets sendToCpu will cause send to CPU (this has priority over drop).
  - When multiple ranges that match has set the forceQueue then the highest numbered entry will determine the value.
- If the destination MAC address bits [47:8] matches the **L2 Reserved Multicast Address Base** then bits [7:0] of the destination MAC address is used as a index in the table **L2 Reserved Multicast Address Action** which determines what action to take on the packet. Actions are set per source port and can either be to drop the packet or to send it to the CPU.

15.2 Configurable Ingress ACL Engine

The ingress ACL engine uses a configurable selection of fields from the incoming packet headers, from L2 fields to L4 fields. From the selected fields a hash table lookup is then done using D-left hashing. The hashing is combined with a TCAM to resolve hash collisions and to enable per entry masking of data. Each of the hash tables can also be masked, but only a single mask can be applied for all data in a hash table.

There are 3 parallell ACL engines that each can perform one lookup per packet. All lookups are done in parallel and then there is a post processing of all the matching results to determine what actions to perform. There can be multiple actions taken for a single packet. How the actions are determined when there are multiple matches are described below.

15.2.1 Field Selection

For each source port the **useAcl*N*** field in the **Source Port Table** configures if the incoming packets shall be subjected to an ACL lookup. By default the ACL is turned off.

If the ACL is turned on then the field **aclRule*N*** is used as a pointer into **Ingress Configurable ACL N Rules Setup** . This determines which fields that are used in the ACL lookup for this source port.

Each ACL engine has its own unique fields which can be selected. These are listed below. A field is selected by setting the corresponding bit in the fieldSelectBitMask.

ACL Engine	Width of Search Data	Fields to select from	Nr of Rules (Fields) to maximum use	Number of Parallel Hash Tables	Small Table Entries	Large Table Entries	TCAM Entries
0	430	16	7	4	1024	8192	16
1	235	39	7	2	512	256	8
2	560	34	22	2	32	128	16

Table 15.1: Ingress ACL Engine Settings

Pre Lookup for Configurable Ingress ACL Table 0

This ACL engine has a pre-lookup. This is done to enable a different rule on how to build the ACL fields to be selected. If this lookup does not result in a valid rule poitner then the rule pointer from the source port table will be selected. The prelookup is setup in **Ingress Configurable ACL 0 Pre Lookup**

Packet Field	Size in Bits	Description
Source Port Bits	2 bits	The source port bits from source port table **preLookupAclBits**.
Type of L3 Packet	2 bits	The packets L3 Type 0 = IPv4 1 = IPv6 2 = MPLS 3 = Others.

Fields for Configurable Ingress ACL Table 0

The following fields can be selected for Configurable Ingress ACL Table 0, the column Bit in Select Bitmask is the number which is set in the bitmask to select the field.

Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
0	MAC DA	48	Always valid	The packets destination MAC address.
1	MAC SA	48	Always valid	The packets source MAC address
2	IPv4 SA	32	When L2 frame holds a IPv4 packet.	IPv4 Source Address.
3	IPv4 DA	32	When L2 frame holds a IPv4 packet.	IPv4 Destination Address.
4	IPv6 SA	128	When L2 frame holds a IPv6 packet.	IPv6 Source Address.
5	IPv6 DA	128	When L2 frame holds a IPv6 packet.	IPv6 Destination Address.
6	L4 Source Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets source port.
7	L4 Destination Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets destination port.
8	L4 Protocol	8	When packet is a IPv4 or IPv6	IPv4, IPv6 L4 protocol type byte.
9	Ethernet Type	16	Always valid	The packets Ethernet Type after VLANs.
10	L4 Type	3	Always valid	The type of an L4 packet. 0 = Not any type in this list. 1 = IPv6 or IPv4 packet but L4 protocol is not UDP, TCP, IGMP, ICMP, ICMPv6 or MLD 2 = UDP in IPv4/6 3 = TCP in IPv4/6 4 = IGMP in IPv4/6 5 = ICMP in IPv4/6 6 = ICMPv6 in IPv6, excluding MLD 7 = MLD - sub protocol of ICMPv6

Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
11	L3 Type	2	Always valid	The type of an L3 packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4,IPv6 or MPLS.
12	Source Port	4	Always valid	The source port of the packet.
13	From Crypto	1	Always valid	This packet has been subjected to encryption/decryption.
14	From Crypto Security Association Ptr	6	When a packet came from the crypto engine and the crypto operation was successful.	The Security Association Ptr used by the crypto engine which was carried out on the packet.
15	Rule Pointer	3	Always valid	The rule pointer (index in the **Ingress Configurable ACL N Rules Setup**).

15.2.2 Example Of Selecting Fields For Configurable Ingress ACL Table 0

Since this ACL engine can select up to 7 fields. This is done by setting bits in the rule pointers fieldSelectBitmask. Lets look at a few examples of the layout of the 430 bits in search key looks like when different fields are selected.

Example ACL with MAC DA

In this example we only want to create a rule with one field which is the MAC destiantion address. This means that the fieldSelectBitmask, which is 16 bits , will be set as follows 1 in binary format (Hex value of 0x1) and the lookup data will be located as follows:

0	MAC DA	Valid
-	Width : 48	1
49	48 1	0 0

Table 15.4: Hash Key Example for MAC DA

Example of Simple L2 ACL

In this example we want to create a rule which with three L2 fields which are Destiantion MAC address, source MAC address and Ethernet Type. Typically this is a L2 ACL Engine. This means that the fieldSelectBitmask, which is 16 bits , will be set as follows 1000000011 in binary format (Hex value of 0x203) and the lookup data will be located as follows:

0	Ethernet Type	MAC DA	MAC SA	Valid
-	Width : 16	Width : 48	Width : 48	3
115	114 99	98 51	50 3	2 0

Table 15.5: Hash Key Example for Simple L2 ACL

Example of L3 IPv4 ACL

In this example we want to create a rule which with four L3 fields which are Destiantion IPv4 address, source IPv4 address, L3 Packet Type and L4 Protocol. Typically this is a L3 ACL Engine. This means that the fieldSelectBitmask, which is 16 bits , will be set as follows 100100001100 in binary format (Hex value of 0x90c) and the lookup data will be located as follows:

0	L3 Type	IPv4 DA	IPv4 SA	L4 Protocol	Valid
-	Width : 2	Width : 32	Width : 32	Width : 8	4
78	77 76	75 44	43 12	11 4	3 0

Table 15.6: Hash Key Example for L3 IPv4 ACL

Example of L4 ACL

In this example we want to create a rule which with five fields which are source port, L4 destiantion Port, L4 source port, L3 Packet Type and L4 Protocol. Typically this is a L4 ACL Engine. This means that the fieldSelectBitmask, which is 16 bits , will be set as follows 1100111000000 in binary format (Hex value of 0x19c0) and the lookup data will be located as follows:

0	Source Port	L3 Type	L4 Protocol	L4 Destination Port	L4 Source Port	Valid
-	Width : 4	Width : 2	Width : 8	Width : 16	Width : 16	5
51	50 47	46 45	44 37	36 21	20 5	4 0

Table 15.7: Hash Key Example for L4 ACL

Example of Ingress NAT Entry

In this example we want to create an rule where the result would be used to change source IP address and/or source L4 Address. This means that the fieldSelectBitmask, which is 16 bits , will be set as follows 1110001000100 in binary format (Hex value of 0x1c44) and the lookup data will be located as follows:

0	Source Port	L3 Type	IPv4 SA	L4 Type	L4 Source Port	Valid
-	Width : 4	Width : 2	Width : 32	Width : 3	Width : 16	5
62	61 58	57 56	55 24	23 21	20 5	4 0

Table 15.8: Hash Key Example for Ingress NAT Entry

Pre Lookup for Configurable Ingress ACL Table 1

This ACL engine has a pre-lookup. This is done to enable a different rule on how to build the ACL fields to be selected. If this lookup does not result in a valid rule poitner then the rule pointer from the source port table will be selected. The prelookup is setup in **Ingress Configurable ACL 1 Pre Lookup**

Packet Field	Size in Bits	Description
Source Port Bits	2 bits	The source port bits from source port table **preLookupAclBits**.
Number of VLANS	2 bits	The packets number of incoming VLANs.
L2 Protocol	1 bits	The packets L2 Type 0 = Other than this list. 1 = MACsec
Type of L3 Packet	2 bits	The packets L3 Type 0 = IPv4 1 = IPv6 2 = MPLS 3 = Others.
Type of L4 Packet	3 bits	The packets L4 Type 0 = Not known. 1 = Is IPv4 or IPv6 but type is not any L4 type in this list. 2 = UDP 3 = TCP 4 = IGMP 5 = ICMP 6 = ICMPv6 7 = MLD

Fields for Configurable Ingress ACL Table 1

The following fields can be selected for Configurable Ingress ACL Table 1, the column Bit in Select Bitmask is the number which is set in the bitmask to select the field.

Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
0	TCP Flags	9	When packet has a L4 TCP protocol and is not a fragment.	The tcp flags for the packet. Bit 0 : ns, Bit 1: cwr, Bit 2: ece, Bit 3: urg, Bit 4: ack, Bit 5: psh, Bit 6: rst, Bit 7:syn, Bit 8: fin
1	MAC DA	48	Always valid	The packets destination MAC address.
2	MAC SA	48	Always valid	The packets source MAC address
3	Outer VID	12	When packet has a VLAN.	The packets outermost VLAN Identifier (VID)
4	Has VLANs	1	Always valid	Does the packet have any VLAN tags 0 = No VLAN in packet 1 = One or more VLANs in packet
5	Outer VLAN Tag Type	1	When packet has an outer VLANs.	When the packet has an outer VLAN what Ethernet Type is this VLAN? 0 = Customer VLAN Tag 1 = Service VLAN Tag
6	Inner VLAN Tag Type	1	When packet has an inner VLAN.	When the packet has an inner VLAN what Ethernet Type is this VLAN? 0 = Customer VLAN Tag 1 = Service VLAN Tag
7	Outer PCP	3	When packet has a VLAN.	The packets outermost VLAN PCP field.
8	Outer DEI	1	When packet has a VLAN.	The packets outermost VLAN DEI field.
9	Inner VID	12	When packet has a two VLANs.	The packes innermost VLAN Identifier (VID).
10	Inner PCP	3	When packet has a two VLANs.	The packets innermost VLAN PCP field.
11	Inner DEI	1	When packet has a two VLANs.	The packets innermost VLAN DEI field.
12	IPv4 SA	32	When L2 frame holds a IPv4 packet.	IPv4 Source Address.
13	IPv4 DA	32	When L2 frame holds a IPv4 packet.	IPv4 Destination Address.
14	IPv6 SA	128	When L2 frame holds a IPv6 packet.	IPv6 Source Address.
15	IPv6 DA	128	When L2 frame holds a IPv6 packet.	IPv6 Destination Address.
16	Outer MPLS	20	When L2 frame holds a MPLS packet.	Outermost MPLS label.
17	TOS	8	When packet is a IPv4 or IPv6	IPv4 or IPv6 Type-Of-Service (TOS) byte.
18	TTL	8	When packet is a IPv4,IPv6 or MPLS	IPv4, IPv6 or MPLS Time-To-Live (TTL) byte.
19	IPSEC SPI	32	When packet is a IPv4 or IPv6 and has AH or ESP L4 Header.	ESP or AH SPI.

Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
20	IPSEC SEQ	32	When packet is a IPv4 or IPv6 and has AH or ESP L4 Header.	ESP or AH sequence number.

21	MACsec Key	64	When packet has a MACsec header.	When a frame with an explicit SCI is expected, program the value according to the following pseudo code:
				`if (pkt.SecTag.SC)`
				`  MACsec_key = pkt.SecTag.SCI`
				`else if (pkt.SecTag.ES and pkt.SecTag.SCB)`
				`  MACsec_key = pkt.MAC_SA, 0x0000`
				`else if (pkt.SecTag.ES and not pkt.SecTag.SCB)`
				`  MACsec_key = pkt.MAC_SA, 0x0001`
				`else [if (not pkt.SecTag.ES)]`
				`  MACsec_key = 0xFFFF_FFFF_FFFF_FFFF.`

22	MACsec Control Byte	8	When packet has a MACsec header.	The first byte after the MACSec Ethernet Type Header. This contains the version, end-station, sci avaliable, single copy broadcast, encryption, changed Text and association number fields.
23	L4 Source Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets source port.
24	L4 Destination Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets destination port.
25	MLD Address	128	When packet is a IPv6 and the ICMPv6 type is equal to 130,131,132	The MLD headers Multicast Address field.
26	ICMP Type	8	When L4 packet is a ICMP packet	ICMP Type.
27	ICMP Code	8	When L4 packet is a ICMP packet	ICMP Code.
28	IGMP Type	8	When L4 packet is a IGMP	IGMP Type.
29	IGMP Group Address	32	When L4 packet is a IGMP	IGMP Group Address.
30	IPv6 Flow Label	20	When a packet is a IPv6.	IPv6 Flow Label.
31	L4 Protocol	8	When packet is a IPv4 or IPv6	IPv4, IPv6 L4 protocol type byte.
32	Ethernet Type	16	Always valid	The packets Ethernet Type after VLANs.
33	L4 Type	3	Always valid	The type of an L4 packet. 0 = Not any type in this list. 1 = IPv6 or IPv4 packet but L4 protocol is not UDP, TCP, IGMP, ICMP, ICMPv6 or MLD 2 = UDP in IPv4/6 3 = TCP in IPv4/6 4 = IGMP in IPv4/6 5 = ICMP in IPv4/6 6 = ICMPv6 in IPv6, excluding MLD 7 = MLD - sub protocol of ICMPv6

Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
34	L3 Type	2	Always valid	The type of an L3 packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4,IPv6 or MPLS.
35	Source Port	4	Always valid	The source port of the packet.
36	From Crypto	1	Always valid	This packet has been subjected to encryption/decryption.
37	From Crypto Security Association Ptr	6	When a packet came from the crypto engine and the crypto operation was successful.	The Security Association Ptr used by the crypto engine which was carried out on the packet.
38	Rule Pointer	3	Always valid	The rule pointer (index in the **Ingress Configurable ACL N Rules Setup**).

15.2.3 Example Of Selecting Fields For Configurable Ingress ACL Table 1

Since this ACL engine can select up to 7 fields. This is done by setting bits in the rule pointers fieldSelectBitmask. Lets look at a few examples of the layout of the 235 bits in search key looks like when different fields are selected.

Example ACL with TOS Byte

In this example we only want to create a rule with one field which is the TOS. This means that the fieldSelectBitmask, which is 39 bits , will be set as follows 100000000000000000 in binary format (Hex value of 0x20000) and the lookup data will be located as follows:

0	TOS	Valid
-	Width : 8	1
9	8 1	0 0

Table 15.11: Hash Key Example for TOS Byte

Example with Destiantion MAC Address and Outer VLAN VID

In this example we want to create a rule which with two fields which are destiantion MAC address and outermost VLAN Identifier. This means that the fieldSelectBitmask, which is 39 bits , will be set as follows 1010 in binary format (Hex value of 0xa) and the lookup data will be located as follows:

0	MAC DA	Outer VID	Valid
-	Width : 48	Width : 12	2
62	61 14	13 2	1 0

Table 15.12: Hash Key Example for Destiantion MAC Address and Outer LAN VID

Example of Complex L2 ACL

In this example we want to create a rule which with six L2 fields which are Destiantion MAC address, source MAC address and Ethernet Type, inner and outer VLANs. The rule pointer would be used to enable different number of VLANs. Typically this is a L2 ACL Engine. This means that the fieldSelectBitmask, which is 39 bits , will be set as follows 1000001000000000000000000001000001110 in binary format (Hex value of 0x410000020e) and the lookup data will be located as follows:

0	MAC DA	MAC SA	Ethernet Type	Outer VID	Inner VID	Rule Pointer	Valid
-	Width : 48	Width : 48	Width : 16	Width : 12	Width : 12	Width : 3	6
145	144 97	96 49	48 33	32 21	20 9	8 6	5 0

Table 15.13: Hash Key Example for Complex L2 ACL

Example of L3 IPv4 ACL

In this example we want to create a rule which with four L3 fields which are Destiantion IPv4 address, source IPv4 address, L3 Packet Type and L4 Protocol. Typically this is a L3 ACL Engine. This means that the fieldSelectBitmask, which is 39 bits , will be set as follows 1001000000000000000011000000000000 in binary format (Hex value of 0x480003000) and the lookup data will be located as follows:

0	L3 Type	IPv4 DA	IPv4 SA	L4 Protocol	Valid
-	Width : 2	Width : 32	Width : 32	Width : 8	4
78	77 76	75 44	43 12	11 4	3 0

Table 15.14: Hash Key Example for L3 IPv4 ACL

Example of L4 ACL

In this example we want to create a rule which with five fields which are source port, L4 destiantion Port, L4 source port, L3 Packet Type and L4 Protocol. Typically this is a L4 ACL Engine. This means that the fieldSelectBitmask, which is 39 bits , will be set as follows 1100100000011000000000000000000000 in binary format (Hex value of 0xc81800000) and the lookup data will be located as follows:

0	Source Port	L3 Type	L4 Protocol	L4 Destination Port	L4 Source Port	Valid
-	Width : 4	Width : 2	Width : 8	Width : 16	Width : 16	5
51	50 47	46 45	44 37	36 21	20 5	4 0

Table 15.15: Hash Key Example for L4 ACL

Example of Openflow Entry

In this example we want to create a rule which looks like an Openflow entry. This can be done by selecing source port, destiantion MAC, source MAC, Ethernet Type, inner VLAN, outer VLAN, L3 Type, IPv4 SA, IPv4 DA, L4 protocol, L4 Source port and L4 Destiantion port and finally the rule pointer. All in all 13 fields are selected. This means that the fieldSelectBitmask, which is 39 bits , will be set as follows 1001101100000011000000000011001000001110 in binary format (Hex value of 0x4d8180320e) and the lookup data will be located as follows:

0	Source Port	MAC DA	MAC SA	Outer VID	Inner VID	Ethernet Type	L3 Type
-	Width : 4	Width : 48	Width : 48	Width : 12	Width : 12	Width : 16	Width : 2
262	261 258	257 210	209 162	161 150	149 138	137 122	121 120

IPv4 SA	IPv4 DA	L4 Protocol	L4 Destination Port	L4 Source Port	Rule Pointer	Valid
Width : 32	Width : 32	Width : 8	Width : 16	Width : 16	Width : 3	13
119 88	87 56	55 48	47 32	31 16	15 13	12 0

Table 15.16: Hash Key Example for Openflow Entry

Example of Ingress NAT Entry

In this example we want to create an rule where the result would be used to change source IP address and/or source L4 Address. This means that the fieldSelectBitmask, which is 39 bits , will be set as follows 1110000000001000000000001000000000000 in binary format (Hex value of 0xe00801000) and the lookup data will be located as follows:

0	Source Port	L3 Type	IPv4 SA	L4 Type	L4 Source Port	Valid
-	Width : 4	Width : 2	Width : 32	Width : 3	Width : 16	5
62	61 58	57 56	55 24	23 21	20 5	4 0

Table 15.17: Hash Key Example for Ingress NAT Entry

Example of IPsec Decryption Entry

In this example we want to create an rule where the result would be used to send the packet to the crypto engine to be decrypted. This means that the fieldSelectBitmask, which is 39 bits , will be set as follows 110000000000000010000011000000000000 in binary format (Hex value of 0xc00083000) and the lookup data will be located as follows:

0	Source Port	IPv4 SA	IPv4 DA	L3 Type	IPSEC SPI	Valid
-	Width : 4	Width : 32	Width : 32	Width : 2	Width : 32	5
107	106 103	102 71	70 39	38 37	36 5	4 0

Table 15.18: Hash Key Example for IPsec Decryption Entry

Pre Lookup for Configurable Ingress ACL Table 2

This ACL engine has a pre-lookup. This is done to enable a different rule on how to build the ACL fields to be selected. If this lookup does not result in a valid rule poitner then the rule pointer from the source port table will be selected. The prelookup is setup in **Ingress Configurable ACL 2 Pre Lookup**

Packet Field	Size in Bits	Description
Source Port Bits	2 bits	The source port bits from source port table **preLookupAclBits**.
Number of VLANS	2 bits	The packets number of incoming VLANs.
L2 Protocol	1 bits	The packets L2 Type 0 = Other than this list. 1 = MACsec
Type of L3 Packet	2 bits	The packets L3 Type 0 = IPv4 1 = IPv6 2 = MPLS 3 = Others.

Fields for Configurable Ingress ACL Table 2

The following fields can be selected for Configurable Ingress ACL Table 2, the column Bit in Select Bitmask is the number which is set in the bitmask to select the field.

Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
0	TCP Flags	9	When packet has a L4 TCP protocol and is not a fragment.	The tcp flags for the packet. Bit 0 : ns, Bit 1: cwr, Bit 2: ece, Bit 3: urg, Bit 4: ack, Bit 5: psh, Bit 6: rst, Bit 7:syn, Bit 8: fin
1	MAC DA	48	Always valid	The packets destination MAC address.
2	MAC SA	48	Always valid	The packets source MAC address
3	Outer VID	12	When packet has a VLAN.	The packets outermost VLAN Identifier (VID)
4	Has VLANs	1	Always valid	Does the packet have any VLAN tags 0 = No VLAN in packet 1 = One or more VLANs in packet

Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
5	Outer VLAN Tag Type	1	When packet has an outer VLANs.	When the packet has an outer VLAN what Ethernet Type is this VLAN? 0 = Customer VLAN Tag 1 = Service VLAN Tag
6	Inner VLAN Tag Type	1	When packet has an inner VLAN.	When the packet has an inner VLAN what Ethernet Type is this VLAN? 0 = Customer VLAN Tag 1 = Service VLAN Tag
7	Outer PCP	3	When packet has a VLAN.	The packets outermost VLAN PCP field.
8	Outer DEI	1	When packet has a VLAN.	The packets outermost VLAN DEI field.
9	Inner VID	12	When packet has a two VLANs.	The packes innermost VLAN Identifier (VID).
10	Inner PCP	3	When packet has a two VLANs.	The packets innermost VLAN PCP field.
11	Inner DEI	1	When packet has a two VLANs.	The packets innermost VLAN DEI field.
12	IPv4 SA	32	When L2 frame holds a IPv4 packet.	IPv4 Source Address.
13	IPv4 DA	32	When L2 frame holds a IPv4 packet.	IPv4 Destination Address.
14	IPv6 SA	128	When L2 frame holds a IPv6 packet.	IPv6 Source Address.
15	IPv6 DA	128	When L2 frame holds a IPv6 packet.	IPv6 Destination Address.
16	Outer MPLS	20	When L2 frame holds a MPLS packet.	Outermost MPLS label.
17	TOS	8	When packet is a IPv4 or IPv6	IPv4 or IPv6 Type-Of-Service (TOS) byte.
18	TTL	8	When packet is a IPv4,IPv6 or MPLS	IPv4, IPv6 or MPLS Time-To-Live (TTL) byte.
19	IPSEC SPI	32	When packet is a IPv4 or IPv6 and has AH or ESP L4 Header.	ESP or AH SPI.
20	IPSEC SEQ	32	When packet is a IPv4 or IPv6 and has AH or ESP L4 Header.	ESP or AH sequence number.

Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description

21	MACsec Key	64	When packet has a MACsec header.	When a frame with an explicit SCI is expected, program the value according to the following pseudo code:
				`if (pkt.SecTag.SC)`
				`  MACsec_key = pkt.SecTag.SCI`
				`else if (pkt.SecTag.ES and pkt.SecTag.SCB)`
				`  MACsec_key = pkt.MAC_SA, 0x0000`
				`else if (pkt.SecTag.ES and not pkt.SecTag.SCB)`
				`  MACsec_key = pkt.MAC_SA, 0x0001`
				`else [if (not pkt.SecTag.ES)]`
				`  MACsec_key = 0xFFFF_FFFF_FFFF_FFFF.`

22	MACsec Control Byte	8	When packet has a MACsec header.	The first byte after the MACSec Ethernet Type Header. This contains the version, end-station, sci avaliable, single copy broadcast, encryption, changed Text and association number fields.
23	L4 Source Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets source port.
24	L4 Destination Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets destination port.
25	IPv6 Flow Label	20	When a packet is a IPv6.	IPv6 Flow Label.
26	L4 Protocol	8	When packet is a IPv4 or IPv6	IPv4, IPv6 L4 protocol type byte.
27	Ethernet Type	16	Always valid	The packets Ethernet Type after VLANs.
28	L4 Type	3	Always valid	The type of an L4 packet. 0 = Not any type in this list. 1 = IPv6 or IPv4 packet but L4 protocol is not UDP, TCP, IGMP, ICMP, ICMPv6 or MLD 2 = UDP in IPv4/6 3 = TCP in IPv4/6 4 = IGMP in IPv4/6 5 = ICMP in IPv4/6 6 = ICMPv6 in IPv6, excluding MLD 7 = MLD - sub protocol of ICMPv6
29	L3 Type	2	Always valid	The type of an L3 packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4,IPv6 or MPLS.
30	Source Port	4	Always valid	The source port of the packet.
31	From Crypto	1	Always valid	This packet has been subjected to encryption/decryption.
32	From Crypto Security Association Ptr	6	When a packet came from the crypto engine and the crypto operation was successful.	The Security Association Ptr used by the crypto engine which was carried out on the packet.
33	Rule Pointer	2	Always valid	The rule pointer (index in the **Ingress Configurable ACL N Rules Setup**).

15.2.4 Example Of Selecting Fields For Configurable Ingress ACL Table 2

Since this ACL engine can select up to 22 fields. This is done by setting bits in the rule pointers fieldSelectBitmask. Lets look at a few examples of the layout of the 560 bits in search key looks like when different fields are selected.

Example ACL with Ethernet Type

In this example we only want to create a rule with one field which is the Ethernet Type. This means that the fieldSelectBitmask, which is 34 bits , will be set as follows 1000000000000000000000000000000 in binary format (Hex value of 0x8000000) and the lookup data will be located as follows:

0	Ethernet Type	Valid
-	Width : 16	1
17	16 1	0 0

Table 15.21: Hash Key Example for Ethernet Type

Example with Destiantion MAC Address and Outer VLAN VID

In this example we want to create a rule which with two fields which are destiantion MAC address and outermost VLAN Identifier. This means that the fieldSelectBitmask, which is 34 bits , will be set as follows 1010 in binary format (Hex value of 0xa) and the lookup data will be located as follows:

0	MAC DA	Outer VID	Valid
-	Width : 48	Width : 12	2
62	61 14	13 2	1 0

Table 15.22: Hash Key Example for Destiantion MAC Address and Outer LAN VID

Example of Simple L2 ACL

In this example we want to create a rule which with three L2 fields which are Destiantion MAC address, source MAC address and Ethernet Type. Typically this is a L2 ACL Engine. This means that the fieldSelectBitmask, which is 34 bits , will be set as follows 1000000000000000000000000110 in binary format (Hex value of 0x8000006) and the lookup data will be located as follows:

0	Ethernet Type	MAC DA	MAC SA	Valid
-	Width : 16	Width : 48	Width : 48	3
115	114 99	98 51	50 3	2 0

Table 15.23: Hash Key Example for Simple L2 ACL

Example of L3 IPv6 ACL

In this example we want to create a rule which with four L3 fields which are Destiantion IPv4 address, source IPv4 address, L3 Packet Type and L4 Protocol. Typically this is a L3 ACL Engine. This means that the fieldSelectBitmask, which is 34 bits , will be set as follows 1001000000000011000000000000000 in binary format (Hex value of 0x2400c000) and the lookup data will be located as follows:

0	L3 Type	IPv6 DA	IPv6 SA	L4 Protocol	Valid
-	Width : 2	Width : 128	Width : 128	Width : 8	4
270	269 268	267 140	139 12	11 4	3 0

Table 15.24: Hash Key Example for L3 IPv6 ACL

Example of L4 ACL

In this example we want to create a rule which with five fields which are source port, L4 destiantion Port, L4 source port, L3 Packet Type and L4 Protocol. Typically this is a L4 ACL Engine. This means that the fieldSelectBitmask, which is 34 bits , will be set as follows 11001011000000000000000000000000 in binary format (Hex value of 0x65800000) and the lookup data will be located as follows:

0	Source Port	L3 Type	L4 Protocol	L4 Destination Port	L4 Source Port	Valid
-	Width : 4	Width : 2	Width : 8	Width : 16	Width : 16	5
51	50 47	46 45	44 37	36 21	20 5	4 0

Table 15.25: Hash Key Example for L4 ACL

Example of Openflow Entry

In this example we want to create a rule which looks like an Openflow entry. This can be done by selecing source port, destiantion MAC, source MAC, Ethernet Type, inner VLAN, outer VLAN, L3 Type, IPv4 SA, IPv4 DA, L4 protocol, L4 Source port and L4 Destiantion port and finally the rule pointer. All in all 13 fields are selected. This means that the fieldSelectBitmask, which is 34 bits , will be set as follows 10011011011000000000011001000001110 in binary format (Hex value of 0x26d80320e) and the lookup data will be located as follows:

0	Source Port	MAC DA	MAC SA	Outer VID	Inner VID	Ethernet Type	L3 Type
-	Width : 4	Width : 48	Width : 48	Width : 12	Width : 12	Width : 16	Width : 2
261	260 257	256 209	208 161	160 149	148 137	136 121	120 119

IPv4 SA	IPv4 DA	L4 Protocol	L4 Destination Port	L4 Source Port	Rule Pointer	Valid
Width : 32	Width : 32	Width : 8	Width : 16	Width : 16	Width : 2	13
118 87	86 55	54 47	46 31	30 15	14 13	12 0

Table 15.26: Hash Key Example for Openflow Entry

Example of Ingress NAT Entry

In this example we want to create an rule where the result would be used to change source IP address and/or source L4 Address. This means that the fieldSelectBitmask, which is 34 bits , will be set as follows 11100001000000000010000000000000 in binary format (Hex value of 0x70801000) and the lookup data will be located as follows:

0	Source Port	L3 Type	IPv4 SA	L4 Type	L4 Source Port	Valid
-	Width : 4	Width : 2	Width : 32	Width : 3	Width : 16	5
62	61 58	57 56	55 24	23 21	20 5	4 0

Table 15.27: Hash Key Example for Ingress NAT Entry

Example of IPsec Decryption Entry

In this example we want to create an rule where the result would be used to send the packet to the crypto engine to be decrypted. This means that the fieldSelectBitmask, which is 34 bits , will be set as follows 11000000000100000011000000000000 in binary format (Hex value of 0x60083000) and the lookup data will be located as follows:

0	Source Port	IPv4 SA	IPv4 DA	L3 Type	IPSEC SPI	Valid
-	Width : 4	Width : 32	Width : 32	Width : 2	Width : 32	5
107	106 103	102 71	70 39	38 37	36 5	4 0

Table 15.28: Hash Key Example for IPsec Decryption Entry

15.2.5 ACL Search

The hash key is used to perform a lookup using the D-left hashing function described in detail in chapter D-left Lookup.

Before the hash key is used the mask in **Ingress Configurable ACL N Search Mask** is applied.

D-left calculates two hash values from the hash key. These hash values are then used to index the **Ingress Configurable ACL N Small Table** and **Ingress Configurable ACL N Large Table** . The hash calculations are described in section Hash function for Configurable ACL.

In addition to the D-left search the hash key is also used to search in the **Ingress Configurable ACL N TCAM** .

15.2.6 ACL Actions

Once a hit has been determined by any of the searches above, the answer is read out from the corresponding answer entry. If it was a D-left hash hit then the answer actions is part of the hash memories (**Ingress Configurable ACL N Small Table** , **Ingress Configurable ACL N Large Table**). If it was a hit in the TCAM then the **Ingress Configurable ACL N TCAM Answer** is used.

The behavior for multiple hits is configured in **Ingress Configurable ACL N Selection** .

The statistics counter which can be updated are located in the **Ingress Configurable ACL Match Counter**

15.3 Multiple ACL Lookups

The section above describes a single ACL Lookup. There are however 3parallel ACL lookups. The functionality in the different lookup engines is the same with the exception that ACL engine 0 has seperate keys for IGMP, ICMP or MLD packets which are not available in the other engines.

Each of the ACL engines has its own rule configuration as well as its own hash and TCAM tables. The hash and TCAM table sizes and search data width for the different engines are as follows.

By using the same rules for multiple engines the table space for a rule can be extended.

15.3.1 Multiple Actions

If the parallel ACL engines have multiple matches the result actions from each search engine can take effect. How multiple actions are handled depends on the type of action.

Any Match

If one or more ACL engines matches and has this action set then the action will take effect.

Action Field	Ingress Acl 0 Has Action	Ingress Acl 1 Has Action	Ingress Acl 2 Has Action
ptp	No	Yes	Yes
noLearning	No	Yes	Yes
dropEnable	Yes	Yes	Yes
sendToCpu	Yes	Yes	Yes

Table 15.29: Actions that will take effect if one or more is set.

First Match or Priority

If multiple ACL engines matches and has this action set then the value from the lowest numbered engine will be used.If an entry has the priority field set this value will be used and the values which do not have priority set will be ignored.If multiple matches have the priority field set then value from the highest numbered engine will be used.

Enable Field	Priority Field	Value Field	Ingress Acl 0 Has Action	Ingress Acl 1 Has Action	Ingress Acl 2 Has Action
forceVidValid	forceVidPrio	forceVid	No	Yes	Yes
forceQueue	forceQueuePrio	eQueue	Yes	Yes	Yes
forceRoute	N/A	nextHopPtr vrf	Yes	No	No
forceColor	forceColorPrio	color	Yes	Yes	Yes
mmpValid	mmpOrder	mmpPtr	Yes	Yes	Yes
updateCfiDei	cfiDeiPrio	newCfiDeiValue	No	Yes	Yes
updatePcp	pcpPrio	newPcpValue	No	Yes	Yes
updateVid	vidPrio	newVidValue	No	Yes	Yes
updateEType	ethPrio	newEthType	No	Yes	Yes
imPrio	inputMirror	destInputMirror	Yes	Yes	Yes
natOpValid	natOpPrio	natOpPtr	Yes	Yes	Yes
tunnelEntry	tunnelEntryPrio	tunnelEntryPtr tunnelEntryUcMc	No	Yes	Yes
tunnelExit	tunnelExitPrio	tunnelExitPtr	No	Yes	No
sendToCrypto	cryptoPrio	secPtr cryptoOp cryptoProto cryptoPort	No	Yes	Yes
sendToPort	N/A	destPort	Yes	Yes	Yes
metaDataValid	metaDataPrio	metaData	Yes	Yes	Yes
updateCounter	N/A	counter	Yes	Yes	Yes
enableUpdateIp	N/A	updateSaOrDa newIpValue	Yes	Yes	Yes
enableUpdateL4	N/A	updateL4SpOrDp newL4Value	Yes	Yes	Yes
updateTosExp	N/A	newTosExp	Yes	Yes	Yes

Table 15.30: The lowest numbered takes effect if no priority else the highest numbered with priority set.

Counter Update

All matches that have counter update action, **updateCounter** set will take effect. Each counter pointed to will be updated. If multiple actions point to the same counter then the counter value will only be incremented by one.

Send To Port

All matches that have an action **sendToPort** will take effect by setting the port number in the packet destination port mask, possibly resulting in a multicast.

Send To CPU

If any match has the **sendToCpu** action set it will take effect. When the To CPU Tag is used the reason code will indicate table index in the lowest numbered engine.

Ingress Admission Control Pointer

If there are multiple matches with actions to set the MMP pointer, mmpPointer then the selection will be done based on the mmpOrder field. This selection is described in Ingress Admission Control.

Update IP Action

In some engines there can also be actions to update the IP fields. Since these actions are only available in one ACL engine there is no need to resolve multiple hits. If an action is enabled and the entry is hit it will take effect.

15.3.2 Default Port ACL action

When a port has the field **enableDefaultPortAcl** set then once a packet misses the ingress ACL lookup, on this source port, this action will be carried out. The action to be carried out is specified in the register **Source Port Default ACL Action**. The actions are the same which can be done for the ACL Lookup. If the bit is set in field **forcePortAclAction** then all packets coming in on this source port are subjected to the actions specified in **Source Port Default ACL Action**. This force ACL default action overrides all other ingress ACL actions/decisions.

15.4 Configurable Egress ACL Engine

The egress ACL engine uses a configurable selection of fields from the incoming packet headers, from L2 fields to L4 fields. From the selected fields a hash table lookup is then done using D-left hashing. The hashing is combined with a TCAM to resolve hash collisions and to enable per entry masking of data. Each of the hash tables can also be masked, but only a single mask can be applied for all data in a hash table.

ACL Engine	Width of Search Data	Fields to select from	Nr of Rules (Fields) to maximum use	Number of Parallel Hash Tables	Small Table Entries	Large Table Entries	TCAM Entries
0	135	21	7	4	1024	8192	16

Table 15.31: Egress ACL Engine Settings

15.4.1 Field Selection

Which fields that will be used in the ACL search is configured in the **Egress Configurable ACL Rules Setup** table. To determine which rule in the table to use the forwarding result from routing and switching is input to a search in **Egress ACL Rule Pointer TCAM**.

The rule pointer determined through this search is then index into **Egress Configurable ACL Rules Setup** table. This table determines which fields that will be part of the hash key in the ACL search.

The possible fields to select are shown below for each ACL engine.

Determining Rule

The forwarding result fields that are used to search in the **Egress ACL Rule Pointer TCAM** are listed below. There is also a mask field for each of the search data fields allowing a selection of which bits in a field that should be compared.

Field	Description
destPortMask	The packets egress ports, one bit per port.
routed	The packet was routed.
vrf	The VRF used when routed.
flooded	The packet was flooded due to L2 table miss.
ucSwitched	The packet was L2 switched to a unicast destination port.
mcSwitched	The packet was L2 switched to a multicast group.
vid	The index used in the VLAN table lookup.
L3 Type	The incoming packets L3 type. IPv4, IPv6 , MPLS or other.
L4 Type	The incoming packets L4 type. TCP,UDP,IGMP,ICMP,ICMPv6,MLD etc.
srcPort	The packets source port.

Table 15.32: Fields used in the rule search.

The TCAM is searched starting at entry 0 and the first matching entry is used. The result is then taken from the **Egress ACL Rule Pointer TCAM Answer** table at the corresponding position. The result is a rule pointer into the **Egress Configurable ACL Rules Setup** tables.

If there is no match in the TCAM the rule pointer 0 will be used. The rule setup can thus not be used to disable the ACL search.

Creating the hash key

All the bits from the fields selected in a rule are concatenated into a hash key. The hash key is used in several places.

- From the hash key two hash indexes are calculated which points into the **Egress Configurable ACL Small Table** and **Egress Configurable ACL Large Table**.
- Secondly the hash key is stored in the compareData field of the hash table entries.
- If a **Egress Configurable ACL TCAM** entry is used the packet data keys are stored in the compareData field of that entry.
- When searching the tables a hash key is constructed from the incoming packets decoded packet fields. The appropriate valid bits are set.

Following the valid bits are the field data in the order that the fields are selected in **Egress Configurable ACL Rules Setup**.

Selectable Packet Fields

The table below lists which fields that are possible to select along with a description on when the fields are valid.

A selected field will only result in a match if the incoming packet has the correct protocol type for the selected field, as determined by the Packet Decoder. For example to match an IPv4 source address does therefore not require that the rule contains a field that checks that the protocol type is a IPv4 packet.

Fields for Configurable Egress ACL Table

The following fields can be selected for Configurable Egress ACL Table, the column Bit in Select Bitmask is the number which is set in the bitmask to select the field.

Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
0	MAC DA	48	Always valid	The packets destination MAC address.
1	MAC SA	48	Always valid	The packets source MAC address
2	IPv4 SA	32	When L2 frame holds a IPv4 packet.	IPv4 Source Address.
3	IPv4 DA	32	When L2 frame holds a IPv4 packet.	IPv4 Destination Address.
4	IPv6 SA	128	When L2 frame holds a IPv6 packet.	IPv6 Source Address.
5	IPv6 DA	128	When L2 frame holds a IPv6 packet.	IPv6 Destination Address.
6	IPSEC SPI	32	When packet is a IPv4 or IPv6 and has AH or ESP L4 Header.	ESP or AH SPI.
7	IPSEC SEQ	32	When packet is a IPv4 or IPv6 and has AH or ESP L4 Header.	ESP or AH sequence number.

Bit in Select Bitmask	Field Name	Size in Bits	When is field valid?	Description
8	L4 Source Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets source port.
9	L4 Destination Port	16	When packet is a IPv4 or IPv6 and UDP or TCP L4 protocol is present	L4 TCP or UDP packets destination port.
10	GID	12	Always valid	GID Pointer from VLAN Table.
11	VID	12	Always valid	The internal VID assigned to the packet.
12	L2 Multicast Pointer	9	When a L2 or L3 router points to a L2 Multicast entry.	If a packet uses a multicast pointer (To the L2 Multicast table) then this is the pointer value.
13	Destination Portmask	11	Always valid	The destination portmask for the packet.
14	L4 Protocol	8	When packet is a IPv4 or IPv6	IPv4, IPv6 L4 protocol type byte.
15	Ethernet Type	16	Always valid	The packets Ethernet Type after VLANs.
16	L4 Type	3	Always valid	The type of an L4 packet. 0 = Not any type in this list. 1 = IPv6 or IPv4 packet but L4 protocol is not UDP, TCP, IGMP, ICMP, ICMPv6 or MLD 2 = UDP in IPv4/6 3 = TCP in IPv4/6 4 = IGMP in IPv4/6 5 = ICMP in IPv4/6 6 = ICMPv6 in IPv6, excluding MLD 7 = MLD - sub protocol of ICMPv6
17	L3 Type	2	Always valid	The type of an L3 packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4,IPv6 or MPLS.
18	Source Port	4	Always valid	The source port of the packet.
19	From Crypto	1	Always valid	This packet has been subjected to encryption/decryption.
20	Rule Pointer	3	Always valid	The rule pointer (index in the **Ingress Configurable ACL N Rules Setup**).

15.4.2 Example Of Selecting Fields For Configurable Egress ACL Table

Since this ACL engine can select up to 7 fields. This is done by setting bits in the rule pointers fieldSelectBitmask. Lets look at a few examples of the layout of the 135 bits in search key looks like when different fields are selected.

Example ACL with MAC DA

In this example we only want to create a rule with one field which is the MAC destiantion address. This means that the fieldSelectBitmask, which is 21 bits , will be set as follows 1 in binary format (Hex value of 0x1) and the lookup data will be located as follows:

0	MAC DA	Valid
-	Width : 48	1
49	48 1	0 0

Table 15.34: Hash Key Example for MAC DA

Example of Simple L2 ACL

In this example we want to create a rule which with three L2 fields which are Destiantion MAC address, source MAC address and Ethernet Type. Typically this is a L2 ACL Engine. This means that the fieldSelectBitmask, which is 21 bits , will be set as follows 1000000000000011 in binary format (Hex value of 0x8003) and the lookup data will be located as follows:

0	Ethernet Type	MAC DA	MAC SA	Valid
-	Width : 16	Width : 48	Width : 48	3
115	114 99	98 51	50 3	2 0

Table 15.35: Hash Key Example for Simple L2 ACL

Example of L3 IPv6 ACL

In this example we want to create a rule which with four L3 fields which are Destiantion IPv4 address, source IPv4 address, L3 Packet Type and L4 Protocol. Typically this is a L3 ACL Engine. This means that the fieldSelectBitmask, which is 21 bits , will be set as follows 100100000000110000 in binary format (Hex value of 0x24030) and the lookup data will be located as follows:

0	L3 Type	IPv6 DA	IPv6 SA	L4 Protocol	Valid
-	Width : 2	Width : 128	Width : 128	Width : 8	4
270	269 268	267 140	139 12	11 4	3 0

Table 15.36: Hash Key Example for L3 IPv6 ACL

Example of L4 ACL

In this example we want to create a rule which with five fields which are source port, L4 destiantion Port, L4 source port, L3 Packet Type and L4 Protocol. Typically this is a L4 ACL Engine. This means that the fieldSelectBitmask, which is 21 bits , will be set as follows 1100100001100000000 in binary format (Hex value of 0x64300) and the lookup data will be located as follows:

0	Source Port	L3 Type	L4 Protocol	L4 Destination Port	L4 Source Port	Valid
-	Width : 4	Width : 2	Width : 8	Width : 16	Width : 16	5
51	50 47	46 45	44 37	36 21	20 5	4 0

Table 15.37: Hash Key Example for L4 ACL

Example of Egress NAT Entry

In this example we want to create an rule where the result would be used to change destination IP address and/or destination L4 Address. This means that the fieldSelectBitmask, which is 21 bits , will be set as follows 110000001000001000 in binary format (Hex value of 0x30208) and the lookup data will be located as follows:

0	L3 Type	IPv4 DA	L4 Type	L4 Destination Port	Valid
-	Width : 2	Width : 32	Width : 3	Width : 16	4
57	56 55	54 23	22 20	19 4	3 0

Table 15.38: Hash Key Example for Egress NAT Entry

Example of IPsec Encryption Entry

In this example we want to create an rule where the result would be used to send the packet to the crypto engine to be encrypted before it should be sent out. This means that the fieldSelectBitmask, which is 21 bits , will be set as follows 100000000000001000 in binary format (Hex value of 0x20008) and the lookup data will be located as follows:

0	L3 Type	IPv4 DA	Valid
-	Width : 2	Width : 32	2
36	35 34	33 2	1 0

Table 15.39: Hash Key Example for IPsec Encryption Entry

Example of MACsec Encryption Entry

In this example we want to create an rule where the result would be used to send the packet to the crypto engine to be encrypted before it should be sent out. This means that the fieldSelectBitmask, which is 21 bits , will be set as follows 10000000000000 in binary format (Hex value of 0x2000) and the lookup data will be located as follows:

0	Destination Portmask	Valid
-	Width : 11	1
12	11 1	0 0

Table 15.40: Hash Key Example for MACsec Encryption Entry

15.4.3 ACL Search

The hash key is used to perform a lookup using the D-left hashing function described in detail in chapter D-left Lookup.

Before the hash key is used the mask in **Egress Configurable ACL Search Mask** is applied.

D-left calculates two hash values from the hash key. These hash values are then used to index the **Egress Configurable ACL Small Table** and **Egress Configurable ACL Large Table**. The hash calculations are described in section Hash function for Configurable ACL.

In addition to the D-left search the hash key is also used to search in the **Egress Configurable ACL TCAM**.

15.4.4 ACL Actions

Once a hit has been determined by any of the searches above, the answer is read out from the corresponding answer entry. If it was a D-left hash hit then the answer actions is part of the hash memories (**Egress Configurable ACL Small Table**, **Egress Configurable ACL Large Table**). If it was a hit in the TCAM then the **Egress Configurable ACL TCAM Answer** is used.

The behavior for multiple hits is configured in **Egress Configurable ACL Selection**.

The statistics counter which can be updated are located in the **Egress Configurable ACL Match Counter**

Chapter 16

VLAN and Packet Type Filtering

This chapter gives an overview of the filtering options available on ingress and egress. Filtering allows different types of packets to be accepted or dropped.

A filter is applied at the source port as packets enter the switch core. This is set up in the **Ingress Port Packet Type Filter** register.

When the packet is ready to be queued, the **Ingress Egress Port Packet Type Filter** is applied for each egress port the packet is to be queued onto. If the packet is dropped then a drop counter is updated for each packet which is dropped.

Before a packet is to be sent out, the egress port it is checked in the **Egress Port Configuration** to see if the packet is allowed to be sent out.

The settings are unique for each port.

A packet of a certain type may be allowed to enter on a certain ingress port. But this does not mean the frame is ultimately allowed to be transmit, since ingress and egress port filters are setup independently.

In addition to the egress port packet type filter, there is also a source port filter on the egress port. This is found in **srcPortFilter**. The source port filter on the egress port allows a user to decide whether packets from a certain source port are allowed to be sent out on an egress port. The outcome of the filtering options are either to drop a packet, or to allow it.

Since the source port table, vlan table and egress port configuration can all have VLAN operations which changes the packet, it is important to understand on which packet the filtering is actually done.

- The source port filtering is done on the packet as it enters the switch without any packet modifications.
- The ingress egress port filtering is done on the packet after the source port and VLAN table VLAN operations. The L2 Multicast is calculated in the same way as MBSC register **L2 Multicast Handling**.
- The egress port filtering is done after all the VLAN operations has been carried out including the egress ports own VLAN operations.

Note that if a user defined VLAN tag is pushed, it will always be regarded as a C-VLAN tag by the filtering.

Chapter 17

Function Control

The functional control settings which are located in register **Ingress Function Control** and register **Egress Function Control** allows the user to turn off specific functions within the packet processing flow.These settings are not intended to be changed for normal use of the switch. The default settings will give the correct behavior. It is recommended to consult Packet Architects when changing these since it often requires understanding of the internals of the switch.

17.1 Ingress Function Control

For each ingress port the physical source port number is used to read out the **Ingress Function Pointer Source Port** which hold a pointer to the **Ingress Function Control** table to be used for a packet.

- If a packet came from the CPU port and did not have a From CPU Tag then register **Ingress Function Control Packet From CPU Port** is used.
- If a packet came from the CPU port and has a From CPU Tag then register **Ingress Function Control Packet From CPU Tag** is used.
- If a packet came from the CPU port and has a From CPU Tag and has the modified bit set to one (1) then register **Ingress Function Control Packet From CPU Tag Do Not Modify** is used.
- If a packet is going to be encrypted then the **Ingress Function Control Packet To Crypto Engine** is going to be used. However since this operation is determined late in the pipeline only a few options can be controlled. See below.
- If a packet came from the crypto engine and was encrypted, the register **Ingress Function Control Packet From Crypto Engine Encrypted** is used.
- If a packet came from the crypto engine and was decrypted, the register **Ingress Function Control Packet From Crypto Engine Decrypted** is used.

17.2 Egress Function Control

For each egress port the physical egress port number is used to read out the **Egress Function Pointer Egress Port** which hold a pointer to the **Egress Function Control**.

- If a packet came from the CPU port and did not have a From CPU Tag then register **Egress Function Control Packet From CPU Port** is used.
- If a packet came from the CPU port and has a From CPU Tag then register **Egress Function Control Packet From CPU Tag** is used.
- If a packet is going to the CPU port, from a L2/L3 table entry, then register **Egress Function Control Packet To CPU Port with Reason Zero** is used.
- If a packet is going to the CPU port but was sent there from a exception (a sent-to-cpu action), thereby resulting in that the reason code is not zero, then register **Egress Function Control Packet To CPU Port** is used.
- If a packet is both going to the CPU and had a From CPU Tag then this register **Egress Function Control Packet To CPU Port** or **Egress Function Control Packet To CPU Port with Reason Zero** is still used.

- If a packet came from the CPU port and has a From CPU Tag and has the modified bit set to one (1) then register **Egress Function Control Packet From CPU Tag Do Not Modify** is used.
- If a packet is going to be encrypted then the **Egress Function Control Packet To Crypto Engine** will be used. However since this operation is determined late in the pipeline only a few options will be controlled. See below.
- If a packet came from the crypto engine and was encrypted, the register **Egress Function Control Packet From Crypto Engine Encrypted** is used.
- If a packet came from the crypto engine and was decrypted, the register **Egress Function Control Packet From Crypto Engine Decrypted** is used.

17.3 Functional Control in Ingress Packet Processing

The following control exists for a packet in the ingress packet processing pipeline:

- **drop** Forces the packet to be dropped.
- **doL2L3Lookup** Should the packet perform a L2 switching and L3 routing lookup?
- **allowRouting** Should the packet be allowed to do routing?
- **noLearning** Should the packet be learned?
- **checkEgressQueueOn** If the egress queue is turned off should this be ignored?
- **enableReservedDmac** Shall the Reserved Destination MAC lookup be done?
- **enableReservedSmac** Shall the Reserved Source MAC lookup be done?
- **enableSrcPortVlanOps** Shall the source port be able to perform VLAN operations?
- **enableTunnelExit** Shall the tunnel exit lookup be done?
- **allowSmon** Shall the SMON counters be updated?
- **allowMbsc** Shall the Mbsc block be consulted for multicast/broadcast/flooding packets?
- **enableIngressPortFilter** Shall the ingress packet port type filters be done?
- **ingressAclEnabled** Shall the ingress ACL lookup be done?
- **checkIngressSpt** Shall the ingress spanning tree be checked?
- **checkEgressSpt** Shall the egress spanning tree be checked?
- **allowVlanPortMembershipDrop** Shall the VLAN table drop packets due to packets not being part of VLAN PortMembership, this affects both the source port and egress port(s) being checked on the vlan-port-membership mask?
- **enableVidVlanOps** Shall the VLAN operations in the VLAN table be carried out?
- **checkIngressMspt** Shall the ingress multiple spanning tree be checked?
- **checkEgressMspt** Shall the egress multiple spanning tree be checked?
- **allowPortMove** Shall the packet be allowed to do a L2 Table port move?
- **checkIngressMmp** Shall the ingress meter-marker-policer be done?
- **doVrfStat** Shall the ingress VRF statistics be updated?
- **doNhHitUpdate** Shall the Next Hop Hit status be updated?
- **routerVops** Shall the router VLAN operations be carried out?
- **egressAclEnabled** Shall the egress ACL lookups be done?
- **allowIngressNat** Shall the packet be allowed to do a ingress NAT operation?
- **allowEgressNat** Shall the packet be allowed to do a egress NAT operation?
- **natActionTable** Shall the NAT Action table lookup be carried out?
- **checkInputMirror** Shall the input mirror function be carried out?
- **updateStatPortMib** Shall the Statistics: IPP Ingress Port Receive statistics be carried out?
- **usePmFromCryAfterEncrypt** After a packet comes back from the crypto unit and the packet was encrypted shall the port mask be selected from the lookups done in IPP before the packet was sent to Crypto Engine?
- **useQueueFromCryAfterEncrypt** After a packet comes back from the crypto unit and the packet was encrypted shall the queue be selected from the lookups done in IPP before the packet was sent to Crypto Engine?
- **cryptoInputMirroring** For a packet going to the crypto engine shall input mirroring be carried out?
- **enableIngressEgressPortFilter** Shall the ingress-egress packet type filtering be carried out?

17.4 Functional Control in Egress Packet Processing

The following control exists for a packet in the egress packet processing pipeline:

- **drop** Forces packet to be dropped at egress.
- **enableEgressPortVlanOperation** Shall the egress port VLAN operation be carried out?
- **updateVrfOutStat** Shall the egress VRF statistics be updated?
- **doEgressVlanTranslation** Shall the egress VLAN translation be carried out?
- **cancelIngressNatOp** Shall the ingress NAT operation be canceled?
- **cancelEgressNatOp** Shall the egress NAT operation be canceled?
- **doEgressQueueRemapping** Shall the egress queue mapping be carried out?
- **removeSNAP** Shall the SNAP remove be done?
- **cancelTunnelEntry** Shall the tunnel entry be cancelled?
- **cancelTunnelExit** Shall the tunnel exit be cancelled?
- **cancelRouting** Shall the routing packet header updates be done?
- **doEgressMplsOp** Shall the MPLS operation (swap,push,pop,penultimate pop) be done?
- **allowTosUpdatesFromColoring** Shall the TOS update from the color be done?
- **allowTosUpdatesFromAcl** Shall the TOS update from the ACL be done?
- **enableEgressPortFilter** Shall the egress port filter be carried out?

Chapter 18

Hashing

Hashing is used to enable the use of SRAM memories instead of using CAMs for lookups.

18.1 Hashing Functions

This section describes the hash functions used in this core.

18.1.1 MAC Table Hashing

The hash function receives the destination MAC address and GID as an input and it returns a hash with the same bit width as the address for the **L2 DA Hash Lookup Table** divided by number of buckets (8). The table is divided into equal sized parts/buckets which are readout in parallel.

Hash Function for MAC Table

The XOR hash function splits the key into 6 parts, each with the width of the hash value. To obtain the hash value a bitwise XOR is performed on all the parts.

When learning random MAC addresses the hash function results in an average utilization of the L2 table of 44% (including/excluding multicast addresses does not change this). When learning sequential MAC addresses (such as in the RFC2889) the utilization is 100%.

Python code for the hashing function is shown below as well as a test case to clarify how the key is calculated.

```
def calc_l2_hash( key ):
  """ key: 60 bits hash key
          key[59:48] = GID
          key[47:0] = MAC
      fold count = 6
      returns: 11 bits hash value
  """
  hashval = key & 0b11111111111
  hashval = hashval ^ (key>>11)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>22)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>33)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>44)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>55)
  hashval = hashval & 0b11111111111
  return hashval

def mac_str2int( mac_adr ):
    """ Convert Ethernet MAC address from string format, e.g. '46:61:62:bc:84:dd'
```

```
    to integer. """
    hx = ''.join(mac_adr.split(':'))
    return int(hx,16)

def l2_hash( gid, mac ):
  """ Calculate index into L2 hash table from GID and MAC address.
      Both parameters must be integers """
  key = (gid & 0xfff) << 48
  key |= mac & 0xffffffffffff
  return calc_l2_hash( key )



def l2_hash_test():
    # Simple test of the hash function to clarify how the key is calculated.
    # MAC: 46:61:62:bc:84:dd (leftmost byte is first byte received)
    # GID:2094
    key = (2094)<< 48 | 0x466162bc84dd
    hashval = calc_l2_hash(key) # the hash value is used as index into the L2 DA Hash Table
    assert hashval == 1795
```

18.1.2 IP Table Hashing

The hash function receives the destination IP address and VRF as key and returns a hash with the same number of bits as the address for the **Hash Based L3 Routing Table** .

Hash Function for IPv4

The XOR hash function splits the key into parts, each with the width of the hash value. To obtain the hash value a bitwise XOR is performed on all the parts.

When learning random IPv4 addresses the hash function results in an average utilization of the hash table of 9% .

Python code for the IPv4 hashing function is shown below as well as a test case to clarify how the key is calculated.

```
def calc_l3_ipv4_hash( key ):
  """ key: 34 bits hash key
          key[33:32] = VRF
          key[33:0] = IP address
      fold count = 3
      returns: 14 bits hash value
  """
  hashval = key & 0b11111111111111
  hashval = hashval ^ (key>>14)
  hashval = hashval & 0b11111111111111
  hashval = hashval ^ (key>>28)
  hashval = hashval & 0b11111111111111
  return hashval

def ipv4_str2int( ip_addr ):
    """ Convert IPv4 address from string format, e.g. 192.168.0.123,
        to integer """
    parts = ip_addr.split('.')
    res = 0
    for p in parts:
        res <<= 8
        res |= int(p)
    return res

def l3_ipv4_hash( vrf, ip_addr ):
  """ Calculate index into L3 hash table from VRF and IP address.
```

```
        Both parameters must be integers. """
    key = (vrf & 0x3) << 32
    key |= ip_addr
    return calc_l3_ipv4_hash( key )


def ipv4_hash_test():
    # Simple test of the hash function to clarify how the key is calculated.
    # IP: 70.119.98.188 (leftmost byte is first byte received)
    # VRF:3
    vrf = 3
    ip = 0x467762bc
    key = ( vrf << 32 ) | ip
    # the hash value is used as index into the Hash Based L3 Routing Table
    hashval = calc_l3_ipv4_hash(key)
    assert hashval == 15189
```

Hash Function for IPv6

The XOR hash function splits the key into parts, each with the width of the hash value. To obtain the hash value a bitwise XOR is performed on all the parts.

When learning random IPv6 addresses the hash function results in an average utilization of the hash table of 9% .

Python code for the IPv6 hashing function is shown below as well as a test case to clarify how the key is calculated.

```
def calc_l3_ipv6_hash( key ):
    """ key: 130 bits hash key
            key[129:128] = VRF
            key[129:0] = IP address
        fold count = 10
        returns: 14 bits hash value
    """
    hashval = key & 0b11111111111111
    hashval = hashval ^ (key>>14)
    hashval = hashval & 0b11111111111111
    hashval = hashval ^ (key>>28)
    hashval = hashval & 0b11111111111111
    hashval = hashval ^ (key>>42)
    hashval = hashval & 0b11111111111111
    hashval = hashval ^ (key>>56)
    hashval = hashval & 0b11111111111111
    hashval = hashval ^ (key>>70)
    hashval = hashval & 0b11111111111111
    hashval = hashval ^ (key>>84)
    hashval = hashval & 0b11111111111111
    hashval = hashval ^ (key>>98)
    hashval = hashval & 0b11111111111111
    hashval = hashval ^ (key>>112)
    hashval = hashval & 0b11111111111111
    hashval = hashval ^ (key>>126)
    hashval = hashval & 0b11111111111111
    return hashval

def l3_ipv6_hash( vrf, ip_addr ):
    """ Calculate index into L3 hash table from VRF and IP address.
        Both parameters must be integers. """
    key = (vrf & 0x3) << 128
    key |= ip_addr
    return calc_l3_ipv6_hash( key )
```

```
def ipv6_hash_test():
    # Simple test of the hash function to clarify how the key is calculated.
    # IP: d8a7:da8b:: (leftmost byte is first byte received)
    # VRF:3
    vrf = 3
    ip = 0xd8a7da8b00000000000000000000000
    key = ( vrf << 128 ) | ip
    hashval = calc_l3_ipv6_hash(key)
    # the hash value is used as index into the Hash Based L3 Routing Table
    assert hashval == 7690
```

18.1.3 MPLS Table Hashing

The hash function receives the outermost MPLS label, source port number and VRF as key and returns a hash with the same number of bits as the address for the **Hash Based L3 Routing Table**

Hash Function for MPLS

The XOR hash function splits the key into parts , each with the width of the hash value. To obtain the hash value a bitwise XOR is performed on all the parts.

When storing random MPLS labels the hash function results in an average utilization of the hash table of 9% .

Python code for the MPLS hashing function is shown below as well as a test case to clarify how the key is calculated.

```
def calc_l3_mpls_hash( key ):
  """ key: 26 bits hash key
          key[25:24] = VRF
          key[23:4] = MPLS label
          key[3:0] = source port
      fold count = 2
      returns: 14 bits hash value
  """
  hashval = key & 0b11111111111111
  hashval = hashval ^ (key>>14)
  hashval = hashval & 0b11111111111111
  return hashval

def l3_mpls_hash( vrf, source_port, label ):
  key = (vrf & 0xfff) << 24
  key |= label & 0xfffff << 4
  key |= ( source_port & 0xf )
  return calc_l3_mpls_hash( key )
```

```
def mpls_hash_test():
    # Simple test of the hash function to clarify how the key is calculated.
    # MPLS label: 588770 (leftmost byte is first byte received)
    # VRF:2
    # source port:8
    mpls_label = 588770
    vrf = 2
    srcport = 8
    key = (vrf << (4 + 20) |
           srcport << 20 |
           mpls_label)
```

```
    hashval = calc_l3_mpls_hash(key)
    # the hash value is used as index into the Hash Based L3 Routing Table
    assert hashval == 12737
```

18.1.4 Hash function for Ingress Configurable ACL 0

The hash function recevies the lookup key created by selecting the fields from the packet determined by the **Ingress Configurable ACL 0 Rules Setup** The lookup key is up to 430 bits wide. The XOR hash function splits the key into parts each with the width of the hash value. To obtain the hash value a bitwise XOR is performed on all the parts.

Python code for the hashing function is shown below as well as a test case to clarify how the key is calculated.

```
def calc_confAcl_small0_hash( key ):
  """ key: 430 bits hash key
      fold count = 54
      returns: 8 bits hash value
  """
  hashval = key & 0b11111111
  hashval = hashval ^ (key>>8)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>16)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>24)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>32)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>40)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>48)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>56)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>64)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>72)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>80)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>88)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>96)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>104)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>112)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>120)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>128)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>136)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>144)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>152)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>160)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>168)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>176)
```

```
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>184)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>192)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>200)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>208)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>216)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>224)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>232)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>240)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>248)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>256)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>264)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>272)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>280)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>288)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>296)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>304)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>312)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>320)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>328)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>336)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>344)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>352)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>360)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>368)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>376)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>384)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>392)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>400)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>408)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>416)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>424)
```

```
    hashval = hashval & 0b11111111
    return hashval

def confAcl_small0_hash( destination_address ):
    """ Calculate index into confAcl_small0 hash table from
        the Destination Address. The parameter must be an integer. """
    key = destination_address & 0x3fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
    return calc_confAcl_small0_hash( key )


def calc_confAcl_large0_hash( key ):
    """ key: 430 bits hash key
        fold count = 40
        returns: 11 bits hash value
    """
    hashval = key & 0b11111111111
    hashval = hashval ^ (key>>11)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>22)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>33)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>44)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>55)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>66)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>77)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>88)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>99)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>110)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>121)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>132)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>143)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>154)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>165)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>176)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>187)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>198)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>209)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>220)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>231)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>242)
    hashval = hashval & 0b11111111111
    hashval = hashval ^ (key>>253)
    hashval = hashval & 0b11111111111
```

```
  hashval = hashval ^ (key>>264)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>275)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>286)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>297)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>308)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>319)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>330)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>341)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>352)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>363)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>374)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>385)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>396)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>407)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>418)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>429)
  hashval = hashval & 0b11111111111
  return hashval

def confAcl_large0_hash( destination_address ):
  """ Calculate index into confAcl_large0 hash table from
      the Destination Address. The parameter must be an integer. """
  key = destination_address & 0x3fffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
  return calc_confAcl_large0_hash( key )



def confAcl0_hash_test():
    key = 184090694140251530287590506897237047983451110792224642826094521040844478950237126082
    hashval = confAcl_small0_hash(key)
    assert hashval == 81

    hashval = confAcl_large0_hash(key)
    assert hashval == 331
```

18.1.5 Hash function for Ingress Configurable ACL 1

The hash function recevies the lookup key created by selecting the fields from the packet determined by the **Ingress Configurable ACL 1 Rules Setup** The lookup key is up to 235 bits wide. The XOR hash function splits the key into parts each with the width of the hash value. To obtain the hash value a bitwise XOR is performed on all the parts.

Python code for the hashing function is shown below as well as a test case to clarify how the key is calculated.

```
def calc_confAcl_small1_hash( key ):
  """ key: 235 bits hash key
      fold count = 30
```

```
    returns: 8 bits hash value
"""
hashval = key & 0b11111111
hashval = hashval ^ (key>>8)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>16)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>24)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>32)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>40)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>48)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>56)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>64)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>72)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>80)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>88)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>96)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>104)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>112)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>120)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>128)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>136)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>144)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>152)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>160)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>168)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>176)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>184)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>192)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>200)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>208)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>216)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>224)
hashval = hashval & 0b11111111
hashval = hashval ^ (key>>232)
hashval = hashval & 0b11111111
return hashval
```

```
def confAcl_small1_hash( destination_address ):
  """ Calculate index into confAcl_small1 hash table from
      the Destination Address. The parameter must be an integer. """
  key = destination_address & 0x7ffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
  return calc_confAcl_small1_hash( key )


def calc_confAcl_large1_hash( key ):
  """ key: 235 bits hash key
      fold count = 34
      returns: 7 bits hash value
  """
  hashval = key & 0b1111111
  hashval = hashval ^ (key>>7)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>14)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>21)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>28)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>35)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>42)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>49)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>56)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>63)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>70)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>77)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>84)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>91)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>98)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>105)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>112)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>119)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>126)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>133)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>140)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>147)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>154)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>161)
  hashval = hashval & 0b1111111
  hashval = hashval ^ (key>>168)
  hashval = hashval & 0b1111111
```

```
    hashval = hashval ^ (key>>175)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>182)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>189)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>196)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>203)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>210)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>217)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>224)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>231)
    hashval = hashval & 0b1111111
    return hashval

def confAcl_large1_hash( destination_address ):
    """ Calculate index into confAcl_large1 hash table from
        the Destination Address. The parameter must be an integer. """
    key = destination_address & 0x7ffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
    return calc_confAcl_large1_hash( key )



def confAcl1_hash_test():
    key = 45945798899178661181533594162047397785742902704752452414489590021841044
    hashval = confAcl_small1_hash(key)
    assert hashval == 30

    hashval = confAcl_large1_hash(key)
    assert hashval == 40
```

18.1.6 Hash function for Ingress Configurable ACL 2

The hash function recevies the lookup key created by selecting the fields from the packet determined by the **Ingress Configurable ACL 2 Rules Setup** The lookup key is up to 560 bits wide. The XOR hash function splits the key into parts each with the width of the hash value. To obtain the hash value a bitwise XOR is performed on all the parts.

Python code for the hashing function is shown below as well as a test case to clarify how the key is calculated.

```
def calc_confAcl_small2_hash( key ):
    """ key: 560 bits hash key
        fold count = 140
        returns: 4 bits hash value
    """
    hashval = key & 0b1111
    hashval = hashval ^ (key>>4)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>8)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>12)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>16)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>20)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>24)
```

```
hashval = hashval & 0b1111
hashval = hashval ^ (key>>28)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>32)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>36)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>40)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>44)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>48)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>52)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>56)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>60)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>64)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>68)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>72)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>76)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>80)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>84)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>88)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>92)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>96)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>100)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>104)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>108)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>112)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>116)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>120)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>124)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>128)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>132)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>136)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>140)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>144)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>148)
```

```
hashval = hashval & 0b1111
hashval = hashval ^ (key>>152)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>156)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>160)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>164)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>168)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>172)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>176)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>180)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>184)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>188)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>192)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>196)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>200)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>204)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>208)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>212)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>216)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>220)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>224)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>228)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>232)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>236)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>240)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>244)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>248)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>252)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>256)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>260)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>264)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>268)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>272)
```

```
hashval = hashval & 0b1111
hashval = hashval ^ (key>>276)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>280)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>284)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>288)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>292)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>296)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>300)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>304)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>308)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>312)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>316)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>320)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>324)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>328)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>332)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>336)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>340)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>344)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>348)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>352)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>356)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>360)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>364)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>368)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>372)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>376)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>380)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>384)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>388)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>392)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>396)
```

```
hashval = hashval & 0b1111
hashval = hashval ^ (key>>400)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>404)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>408)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>412)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>416)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>420)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>424)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>428)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>432)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>436)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>440)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>444)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>448)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>452)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>456)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>460)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>464)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>468)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>472)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>476)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>480)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>484)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>488)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>492)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>496)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>500)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>504)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>508)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>512)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>516)
hashval = hashval & 0b1111
hashval = hashval ^ (key>>520)
```

```
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>524)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>528)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>532)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>536)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>540)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>544)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>548)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>552)
    hashval = hashval & 0b1111
    hashval = hashval ^ (key>>556)
    hashval = hashval & 0b1111
    return hashval

def confAcl_small2_hash( destination_address ):
    """ Calculate index into confAcl_small2 hash table from
        the Destination Address. The parameter must be an integer. """
    key = destination_address & 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
    return calc_confAcl_small2_hash( key )


def calc_confAcl_large2_hash( key ):
    """ key: 560 bits hash key
        fold count = 94
        returns: 6 bits hash value
    """
    hashval = key & 0b111111
    hashval = hashval ^ (key>>6)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>12)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>18)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>24)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>30)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>36)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>42)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>48)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>54)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>60)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>66)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>72)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>78)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>84)
    hashval = hashval & 0b111111
```

```
hashval = hashval ^ (key>>90)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>96)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>102)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>108)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>114)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>120)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>126)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>132)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>138)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>144)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>150)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>156)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>162)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>168)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>174)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>180)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>186)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>192)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>198)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>204)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>210)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>216)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>222)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>228)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>234)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>240)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>246)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>252)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>258)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>264)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>270)
hashval = hashval & 0b111111
```

```
hashval = hashval ^ (key>>276)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>282)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>288)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>294)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>300)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>306)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>312)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>318)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>324)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>330)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>336)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>342)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>348)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>354)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>360)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>366)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>372)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>378)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>384)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>390)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>396)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>402)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>408)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>414)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>420)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>426)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>432)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>438)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>444)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>450)
hashval = hashval & 0b111111
hashval = hashval ^ (key>>456)
hashval = hashval & 0b111111
```

```
    hashval = hashval ^ (key>>462)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>468)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>474)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>480)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>486)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>492)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>498)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>504)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>510)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>516)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>522)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>528)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>534)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>540)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>546)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>552)
    hashval = hashval & 0b111111
    hashval = hashval ^ (key>>558)
    hashval = hashval & 0b111111
    return hashval

def confAcl_large2_hash( destination_address ):
    """ Calculate index into confAcl_large2 hash table from
        the Destination Address. The parameter must be an integer. """
    key = destination_address & 0xffffffffffffffffffffffffffffffffffffffffffffffffffffffffffffff
    return calc_confAcl_large2_hash( key )



def confAcl2_hash_test():
    key = 2923777827478276180289699106532030774361946949086266888194904512645914750916152426205
    hashval = confAcl_small2_hash(key)
    assert hashval == 0

    hashval = confAcl_large2_hash(key)
    assert hashval == 57
```

18.1.7 Hash function for Egress Configurable ACL

The hash function recevies the lookup key created by selecting the fields from the packet determined by the **Egress Configurable ACL Rules Setup** The lookup key is up to 135 bits wide. The XOR hash function splits the key into parts each with the width of the hash value. To obtain the hash value a bitwise XOR is performed on all the parts.

Python code for the hashing function is shown below as well as a test case to clarify how the key is calculated.

```
def calc_confAcl_small0_hash( key ):
```

```
  """ key: 135 bits hash key
      fold count = 17
      returns: 8 bits hash value
  """
  hashval = key & 0b11111111
  hashval = hashval ^ (key>>8)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>16)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>24)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>32)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>40)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>48)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>56)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>64)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>72)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>80)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>88)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>96)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>104)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>112)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>120)
  hashval = hashval & 0b11111111
  hashval = hashval ^ (key>>128)
  hashval = hashval & 0b11111111
  return hashval

def confAcl_small0_hash( destination_address ):
  """ Calculate index into confAcl_small0 hash table from
      the Destination Address. The parameter must be an integer. """
  key = destination_address & 0x7fffffffffffffffffffffffffffffffff
  return calc_confAcl_small0_hash( key )


def calc_confAcl_large0_hash( key ):
  """ key: 135 bits hash key
      fold count = 13
      returns: 11 bits hash value
  """
  hashval = key & 0b11111111111
  hashval = hashval ^ (key>>11)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>22)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>33)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>44)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>55)
  hashval = hashval & 0b11111111111
```

```
  hashval = hashval ^ (key>>66)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>77)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>88)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>99)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>110)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>121)
  hashval = hashval & 0b11111111111
  hashval = hashval ^ (key>>132)
  hashval = hashval & 0b11111111111
  return hashval

def confAcl_large0_hash( destination_address ):
  """ Calculate index into confAcl_large0 hash table from
      the Destination Address. The parameter must be an integer. """
  key = destination_address & 0x7fffffffffffffffffffffffffffffff
  return calc_confAcl_large0_hash( key )



def confEgressAcl0_hash_test():
    key = 263262093704657170311533916689756200733 48
    hashval = confEgressAcl_small0_hash(key)
    assert hashval == 170

    hashval = confEgressAcl_large0_hash(key)
    assert hashval == 1059
```

18.1.8 Hash function for Egress Vlan Translation

The hash function receives the outermost VID of the modified packet at egress, the egress port number, along with the VLAN Ethernet type (C or S tag). The XOR hash function splits the key into parts each with the width of the hash value. To obtain the hash value a bitwise XOR is performed on all the parts.

Python code for the hashing function is shown below as well as a test case to clarify how the key is calculated.

```
def calc_egressVlanTranslation_small_hash( outermostVidType,
                                           outermostVid,
                                           dstPort ):
  """ key: 17 bits hash key
      fold count = 3
      returns: 6 bits hash value
  """
  key = 0
  key = key << 1 | (outermostVidType & 0x1)
  key = key << 12 | (outermostVid & 0xfff)
  key = key << 4 | (dstPort & 0xf)
  hashval = key & 0b111111
  hashval = hashval ^ (key>>6)
  hashval = hashval & 0b111111
  hashval = hashval ^ (key>>12)
  hashval = hashval & 0b111111
  return hashval
def egressVlanTranslation_small_hash( outermostVidType,
                                      outermostVid,
                                      dstPort ):
  """ Calculate index into egressVlanTranslation_small hash table from
```

```
        the different fields. The parameter must be an integer. """

    return calc_egressVlanTranslation_small_hash( outermostVidType=outermostVidType,
                                                  outermostVid=outermostVid,
                                                  dstPort=dstPort )


def calc_egressVlanTranslation_large_hash( outermostVidType,
                                           outermostVid,
                                           dstPort ):
    """ key: 17 bits hash key
        fold count = 3
        returns: 7 bits hash value
    """
    key = 0
    key = key << 1 | (outermostVidType & 0x1)
    key = key << 12 | (outermostVid & 0xfff)
    key = key << 4 | (dstPort & 0xf)
    hashval = key & 0b1111111
    hashval = hashval ^ (key>>7)
    hashval = hashval & 0b1111111
    hashval = hashval ^ (key>>14)
    hashval = hashval & 0b1111111
    return hashval
def egressVlanTranslation_large_hash( outermostVidType,
                                      outermostVid,
                                      dstPort ):
    """ Calculate index into egressVlanTranslation_large hash table from
        the different fields. The parameter must be an integer. """

    return calc_egressVlanTranslation_large_hash( outermostVidType=outermostVidType,
                                                  outermostVid=outermostVid,
                                                  dstPort=dstPort )


def egressVlanTranslation_hash_test():
    dstPort = 6
    outermostVid = 2139
    outermostVidType = 1

    hashval = egressVlanTranslation_small_hash( outermostVidType=outermostVidType,
                                                outermostVid=outermostVid,
                                                dstPort=dstPort)
    assert hashval == 56

    hashval = egressVlanTranslation_large_hash( outermostVidType=outermostVidType,
                                                outermostVid=outermostVid,
                                                dstPort=dstPort)
    assert hashval == 59
```

18.1.9 Hash function for Tunneling

The tunneling exit lookups consits of two lookups. First the tunnel exit lookup and secondly the second tunnel exit lookup.

First Tunnel Exit Hash

Uses only TCAM in this design. Located in table **Tunnel Exit Lookup TCAM**.

Second Tunnel Exit Hash

Uses only TCAM in this design. Located in table **Second Tunnel Exit Lookup TCAM**.

Chapter 19

D-left Lookup

D-left is a hash table search algorithm that reduces the risk of hash collisions by using two hash tables each indexed by a separate hash key.

This implementation uses two hash tables, one smaller and one larger, combined with a synthesized TCAM to resolve hash collisions. This is shown in figure 19.1.

The hash search is done by taking a hash key and calculating two hashes from that. The two hash values are used as index into the small and large hash tables.

Each table has a number of buckets for each hash index. All buckets for the selected index are read out in parallel. The hash key is then compared with the compareData from each bucket. There is a hit if one of the buckets compareData matches the hash key. If multiple buckets matches then the highest numbered bucket is used.

This is done in parallel for both the small and the large table.

In addition the hash key is also searched in the TCAM. In the TCAM search all entries are compared with the hash and if there are multiple matches then the lowest numbered entry is used.

Since a single search can result in multiple hits in all three tables there is configuration that selects which table shall be used in this case.

The two hash tables have separate masks which allows some bits to be masked away. For the TCAM there is a mask per entry.

19.1 Functions using D-left

The following functions use D-left Lookup.

19.1.1 Egress VLAN Translation

The Egress VLAN Translation table:

- The hash tables are **Egress VLAN Translation Small Table** and **Egress VLAN Translation Large Table**. Each of the the hash tables has 4 buckets for each hash index.
- The search data/hash key is the egress port, the outermost VID and the outermost VID Type, a C-tag (0) or S-tag (1).
- The TCAM is **Egress VLAN Translation TCAM**.
- The hash functions used to index the **Egress VLAN Translation Small Table** and **Egress VLAN Translation Large Table** are described in section Hash function for Egress VLAN Translation.
- The masks for the hash tables are **Egress VLAN Translation Search Mask**.
- The configuration for resolving multiple hits is in **Egress VLAN Translation Selection**.
- While the hash tables stores the answer in the same memories as the lookup key, the TCAM has a seperate table holding the answer: **Egress VLAN Translation TCAM Answer**.

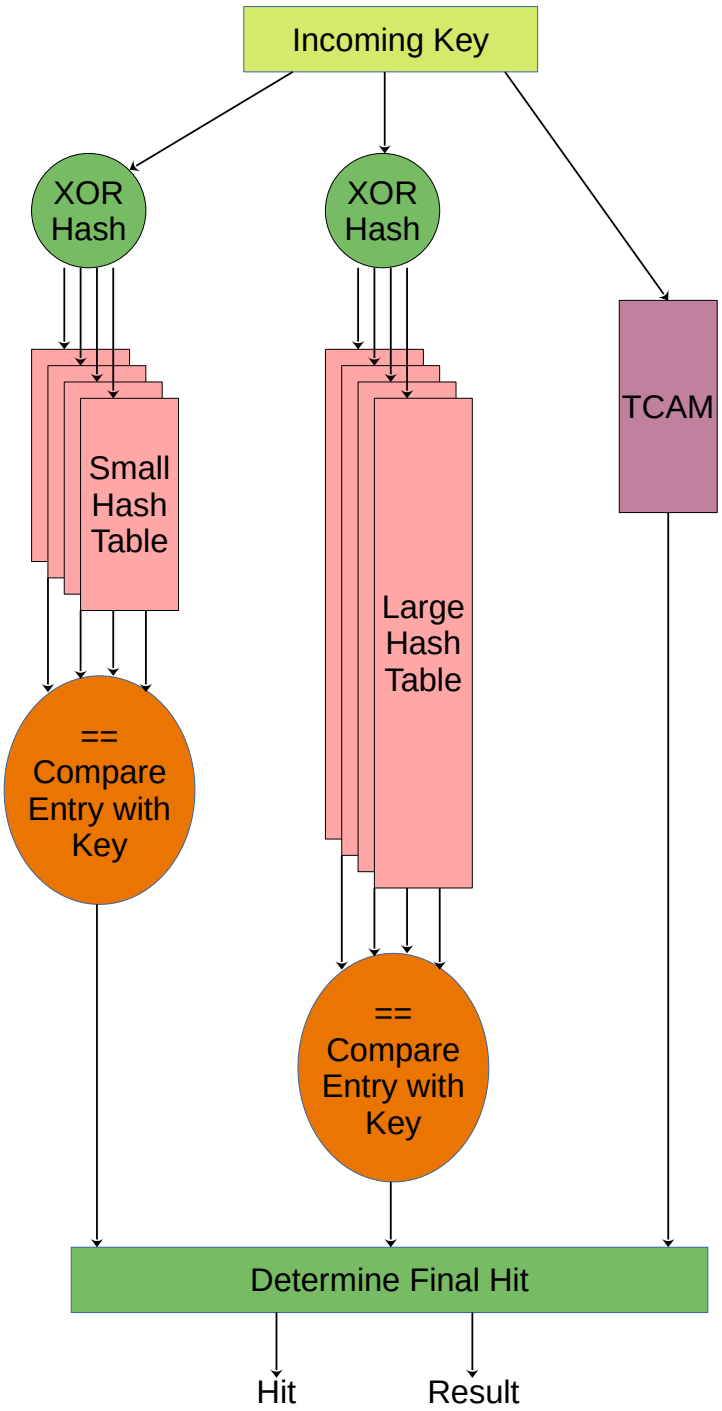


Figure 19.1: D-left Function

19.1.2 Ingress Configurable ACL

The ingress configurable ACL is setup by using the following registers and tables.

- The search data/hash key is the selected packet header fields (see Selectable Packet Fields).
- Hash tables
  - The hash functions used to index the hash tables are described in section Hash function for Configurable ACL.
  - **Ingress Configurable ACL 0 Small Table**

  - **Ingress Configurable ACL 0 Large Table**
  - **Ingress Configurable ACL 1 Small Table**
  - **Ingress Configurable ACL 1 Large Table**
  - **Ingress Configurable ACL 2 Small Table**
  - **Ingress Configurable ACL 2 Large Table**
- TCAM
  - **Ingress Configurable ACL 0 TCAM**
  - **Ingress Configurable ACL 1 TCAM**
  - **Ingress Configurable ACL 2 TCAM**
- Masks for the hash tables
  - **Ingress Configurable ACL 0 Search Mask**
  - **Ingress Configurable ACL 1 Search Mask**
  - **Ingress Configurable ACL 2 Search Mask**
- Configuration for resolving multiple hits
  - **Ingress Configurable ACL 0 Selection**
  - **Ingress Configurable ACL 1 Selection**
  - **Ingress Configurable ACL 2 Selection**
- The ACL actions are stored in the hash tables but the actions for TCAM hits are stored in a separate tables
  - **Ingress Configurable ACL 0 TCAM Answer**
  - **Ingress Configurable ACL 1 TCAM Answer**
  - **Ingress Configurable ACL 2 TCAM Answer**

19.1.3 Egress Configurable ACL

The ingress configurable ACL is setup by using the following registers and tables.

- The search data/hash key is the selected packet header fields (see Selectable Packet Fields).
- Hash tables
  - The hash functions used to index the hash tables are described in section Hash function for Configurable ACL.
  - **Egress Configurable ACL Small Table**
  - **Egress Configurable ACL Large Table**
- TCAM
  - **Egress Configurable ACL TCAM**
- Masks for the hash tables
  - **Egress Configurable ACL Search Mask**
- Configuration for resolving multiple hits
  - **Egress Configurable ACL Selection**
- The ACL actions are stored in the hash tables but the actions for TCAM hits are stored in a separate tables
  - **Egress Configurable ACL TCAM Answer**

19.1.4 Tunnel Exit

The first tunnel exit lookup uses only TCAM. Located in table **Tunnel Exit Lookup TCAM**.

The second tunnel exit lookup uses only TCAM. Located in table **Second Tunnel Exit Lookup TCAM**.

Chapter 20

Learning and Aging

The switch supports automatic hardware learning and aging as well as software controlled learning and aging.

- With hardware learning the switch can be functional after reset without any software setup. The hardware learning engine saves the source port number, the source MAC address with a Global Identifier (GID) from the **VLAN Table** in the forwarding information base.
- If the destination MAC address and the GID of a packet is in the L2 forwarding information base, the L2 forwarding process will know the destination port of this packet.
- If a learned {GID, MAC} has not been hit by a source or destination MAC address for a while, the hardware aging engine will remove this entry from the table.
- When a learned MAC address is received as MAC SA on a different port than it was setup in the **L2 Destination Table**, it is considered a port move.
- When the hardware aging is enabled, all non-static entries will be aged out after a certain silent period. **Hardware Learning Configuration** configures the initial status of the newly learned entries.
- The software learning and aging feature allows users to fully control the L2 forwarding information base.
- The hardware learning and aging functions are by default turned on and can be turned off through the **Learning And Aging Enable** register.
- When the hardware learning is enabled, all source ports are allowed to get their unknown source MAC address learned. By setting **learningEn** field in the **Source Port Table** to 0 the learning process can be disabled on the corresponding source port.
- For an unknown MAC DA, **dropUnknownDa** field in the **Source Port Table** determines either to drop the packet or allow it to be flooded.

20.1 L2 Forwarding Information Base (FIB)

Multiple tables in groups are involved in the learning and aging functions when making L2 forwarding decisions:

20.1.1 Tables for MAC DA lookup

1. L2 Hash tables.
   (a) **L2 DA Hash Lookup Table**
   (b) **L2 Aging Status Shadow Table**
2. L2 Collision tables.
   (a) **L2 Lookup Collision Table**
   (b) **L2 Aging Collision Shadow Table**
3. **L2 Destination Table**.
4. **L2 Multicast Table**.

MAC DA lookups are used to find L2 forwarding destinations and the related tables are written as results from learning or aging functions. The forwarding function relies on a hash algorithm described in Section MAC Table Hashing and a search algorithm described in Section L2 Destination Lookup. In this core, destination MAC addresses and GIDs are combined together to create a 60-bit hash key and the hash function returns a 11-bit hash value.

20.1.2 Tables for MAC SA lookup

1. **L2 SA Hash Lookup Table**. Holding the same contents as **L2 DA Hash Lookup Table**.
2. **L2 Aging Status Shadow Table - Replica**. Holding the same contents as **L2 Aging Status Shadow Table**.
3. **L2 Destination Table - Replica**. Holding the same contents as **L2 Destination Table**.

The MAC SA lookups are used to create new learning requests and requiring the same tables as MAC DA lookups. Due to the fact that the core mostly uses tables with single read port towards the ingress processing pipeline, there are three MAC DA tables duplicated to MAC SA tables listed above to support one read per cycle from the ingress processing pipeline (one MAC DA lookup and one MAC SA lookup at every clock cycle). No matter when the MAC DA/MAC SA lookup tables are updated, the corresponding SA/DA lookup tables need to be filled with the same updates. The L2 collision tables are built to support parallel read by both DA and MAC SA lookups and therefore are not duplicated.

The MAC SA lookups form a key-hash pair by {GID,MAC SA} and do a two step check:

1. Hit or not. Hit is given in two cases:
   (a) The key-hash pair is found in the **L2 SA Hash Lookup Table** and the related entry in **L2 Aging Status Shadow Table - Replica** is valid.
   (b) The key is found in the **L2 Lookup Collision Table** and the related entry in **L2 Aging Collision Table** is valid.
2. The source port number matches the port number in the L2 destination table.

Based on the lookup result there are three possible learning decisions:

1. Learn a new entry: Not hit.
2. Port move request: Hit with port number mismatching.
3. SA hit update operation: Hit with port number matching.

Figure 6.1 demonstrates how the FIB addressing looks like.

20.1.3 Status Tables

1. **L2 Aging Table**
2. **L2 Aging Collision Table**

The status tables are located inside the learning and aging engine to monitor and maintain the status of all entries in the FIB. An FIB entry has three status bits:

1. **`valid`**: Indicate if a hit in the FIB is valid.
2. **`stat`**: Indicate if an entry is static. Static entries cannot be modified by hardware.
3. **`hit`**: Indicate either MAC SA or DA has successfully hit this entry since the last aging scan.

When the hardware learning or aging updates the status table, the **`valid`** bit will be copied to the shadow tables in the ingress processing pipeline.

As in Figure 20.1 the FIB can be accessed from three units:

1. From software through the configuration interface: read and write.
2. Learning and aging unit: read and write.
3. Ingress processing pipeline: read only.

Notice that shadow tables in the FIB have to be updated simultaneously with status tables. MAC SA lookup tables have to be updated simultaneously with MAC DA lookup tables. Unexpected behavior will occur if the tables do not have the same content.

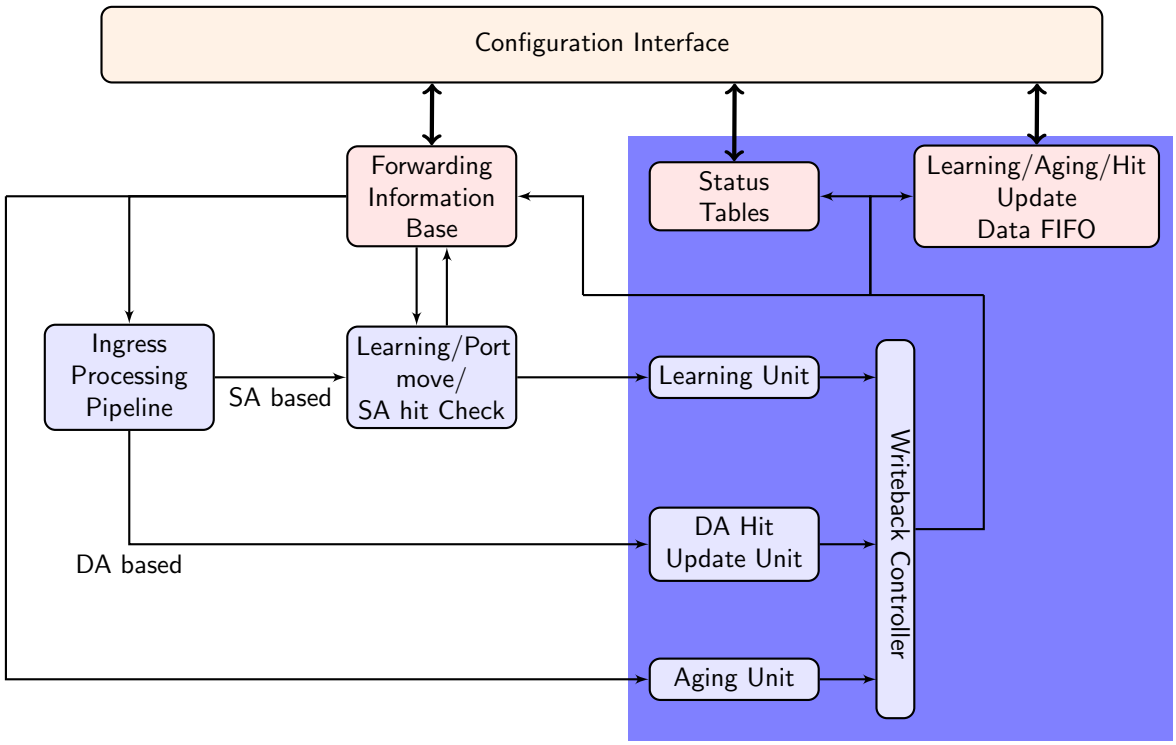


Figure 20.1: Learning and Aging Engine

20.1.4 Hash Collision Accommodation

In order to solve hash collisions, the **L2 DA Hash Lookup Table** has 8 buckets each with 2,048 entries. A given key-hash pair can search in the 8 buckets in parallel by reading from the address that equals the hash value. The 8 buckets entries are all compared with the {GID,MAC DA} key and if one entry is equal to the key that entry is considered a match.

Besides the **L2 DA Hash Lookup Table**, there is an extra **L2 Lookup Collision Table** in case the number of hash collisions is more than the **L2 DA Hash Lookup Table** can handle. For instance, if the hash function calculated the same hash value for more than 8 keys, the first 8 keys can be accommondated in the 8 buckets of **L2 DA Hash Lookup Table** while the rest are stored in the **L2 Lookup Collision Table**. Searching in the **L2 Lookup Collision Table** will return the first entry index that holds the corresponding key.

Addressing into the **L2 Destination Table** is based on the hit index from either the **L2 DA Hash Lookup Table** or the **L2 Lookup Collision Table**.

- Hit in the **L2 DA Hash Lookup Table**: get a 14-bit hit index with the hash value in the lower 11 bits and the bucket number in the higher 3 bits. The corresponding **L2 Destination Table** address equals the hit index.
- Hit in the **L2 Lookup Collision Table**: get a 5-bit hit index from the hit entry address. The corresponding **L2 Destination Table** address is (hit index + 16,384).

20.2 Hardware Learning and Aging

20.2.1 Learning Unit

The core has a dedicated learning unit in hardware, which is tasked with learning L2 MAC addresses combined with GIDs as entries to do L2 destination port lookups. A new learning request is created and processed in several steps:

1. For every packet a learning check is performed based on its MAC SA and GID and issues learning requests to the learning unit.

2. If it is a known entry but the **hit** bit in the status table is 0, the **hit** bit will be refreshed to 1.
3. If the learning request is to learn a new entry, **Hardware Learning Counter** will be checked against the **learnLimit** in **Hardware Learning Configuration**. **learnLimit** limits the maximum number of entries can be learned on a port.
4. If the maximum learning limit is not reached on a port, the status table lookup will try to provide an available entry in a certain order:
   (a) Find a free entry.
      i. Select a free bucket for this hash value.
      ii. If all hash buckets are used, select a free collision table entry.
   (b) If there is no free entry and **lru** in the **Learning And Aging Enable** register is 0, the learning unit will search in the collision table and overwrite the non-static entries in a round robin order.
   (c) If there is no free entry and **lru** in the **Learning And Aging Enable** register is 1, the learning unit will overwrite a least recently used non-static entry as follows:
      i. Search in hash buckets for a bucket with **hit**=0 and **stat**=0. Return the last match.
      ii. If all buckets have **hit**=1 or **stat**=1, search in the collision table for an entry with **hit**=0 and **stat**=0. Return the first match.
   (d) If all entries are static or have been hit since the last aging scan, overwrite a non-static entry.
      i. Search in hash buckets for a bucket with **stat**=0. Return the last match.
      ii. If all buckets are static, search in the collision table for an entry with **stat**=0 in a round robin order.
5. If the learning unit failed to accomondate the unknown MAC SA and GID combination, or the learning limit on a port is reached, the learning request will be ignored and the corresponding MAC SA, GID and port number will be updated to the **Learning Overflow** register.
6. If a valid entry is found, the learning unit will link it to the port number from the learning request as a L2 unicast entry.
7. If the learning request is for a port move, the process will operate on existing non-static entries directly. For static entries, the **Port Move Options** register gives optional operations for each previously learned port.
8. If the learning unit failed to execute port move due to immutable static entry or the learning limit is reached, the learning request will be ignored and the corresponding MAC SA, GID and port number will be updated to the **Learning Conflict** register.
9. A valid learning decision is sent to a writeback bus which manages all decisions from different learning and aging units. The learning decisions have the highest priority to use the writeback bus.
10. The writeback bus pushes the learning decision to the **Learning Data FIFO**. By default the writeback bus is allowed to send decisions to the FIB, but there is also an option to block the table updates from the configuration interface.
11. By setting the **hwLearningWriteBack** field in the **Learning And Aging Writeback Control** to 0, table updates from the hardware learning unit is blocked. In this case the software shall maintain the hardware learning decision from the **Learning Data FIFO**, and updates the FIB as described in Section Software Learning and Aging.

20.2.2 Hardware Learning Exceptions

The switch support fine granular control to allow certain packets with unknown MAC SA address to not be learned. These settings described below enables a varity of different ways to turn it off on a per packet basis.

- Source port exceptions.
  - If **uniqueCpuMac** is set to 1, the CPU port cannot be learned.
  - If the packet from the CPU port has a from CPU tag, it will bypass L2 lookup hence bypass the learning process.
  - For any source port if its **learningEn** is set to 0 the learning process is disabled.
- To CPU packet. If the packet is sent to the CPU port with a non-zero reason code. [1]
- Classification.
  - If the packet hit in a classification rule that override L2 lookup (i.e. force the destination port), it will not be learned.

[1] Check all reason codes in Table 33.2

  - If the packet hit in the **Configurable ACL Engine** with **noLearning** enabled.
- Routed. A routed packet will not be learned.
- Dropped. If the ingress processing drops the packet (post-ingress processing is not counted), the packet will not be learned unless it is due to the ingress spanning tree drop and the state says **Learning**. [2]
- Multicast MAC SA. In the switch core a MAC address with the least-significant bit of the first octet equals 1 (e.g. 01:80:c2:00:00:00) but not equals to ff:ff:ff:ff:ff:ff is marked as Ethernet multicast address. By default a MAC SA that matches an Ethernet multicast address will not be learned. This can be configured per port through the **learnMulticastSaMac** field in the **Source Port Table**.

20.2.3 Aging Unit

When a new L2 entry is learned by the hardware learning unit, the initial entry status is from the **Hardware Learning Configuration** register. A valid non-static entry will be aged out if no L2 MAC SA/DA lookup hit it within a certain time and static entries must have software interactions to get aged/changed. By default a non-static entry will be learned with both **hit** and **valid** set to 1 to prevent it from being aged out immediately. Static entries can be established on a per source port basis by setting the **stat** field in **Hardware Learning Configuration** to 1.

The hardware aging function does a periodic check of the L2 entry status in the **L2 Aging Table** and the **L2 Aging Collision Table**. The waiting period between two checks is tick based [3] and configurable via the **Time to Age** register. During an aging check period, the aging unit loops through all entries in the **L2 Aging Table** and **L2 Aging Collision Table** to get the current status. The possible updates are listed in Table 20.1. If the **valid** bit (bit 0) is turned to 0 the entry is aged out. An aged out entry can be learned again.

If the **Time to Age** register is reconfigured during runtime, the updated **tickCnt** will not be available to aging unit until the current aging period is complete. In order to load new values immediately, the aging unit needs to be restarted via the **agingEnable** field in the **Learning And Aging Enable** register. However, changes to the **tick** selection are always applied immediately.

Current Status	Update Status
0b101	0b001
0b001	0b000(entry cleared)
Other values	No update

Table 20.1: Hardware Aging Operations

20.2.4 MAC DA Hit Update Unit

The learning unit has a built-in MAC SA hit update unit to refresh the **hit** bit while another MAC DA hit update unit can operate in parallel. The MAC DA hit update unit can be turned on or off by the **daHitEnable** field in the **Learning And Aging Enable** register and works as such:

1. A packet with L2 MAC DA lookup returns a valid and non-static entry issues a hit update request for the corresponding MAC DA.
2. A hit update FIFO is prepared to buffer the update requests.
3. A hit update request is popped from the FIFO when the writeback bus is free.
4. If the writeback bus keeps busy with learning decisions and causes a buildup in the hit update FIFO, new hit update requests will be ignored when the FIFO is full.
5. The writeback bus forwards the hit update request to both the **Hit Update Data FIFO** and the FIB, optionally the FIB updates could be turned off by the **hwHitWriteBack** field in the **Learning And Aging Writeback Control** register.

According to Table 20.1, the automatic **hit** bit update for an non-static L2 entry will keep the hardware aging unit away from setting the **valid** bit to 0, hence avoid aging out the entry.

20.3 Software Learning and Aging

Instead of automatic learning and aging, the switch provides two options for software to manipulate learning and aging behaviors.

[2] See more in Chapter Spanning Tree.
[3] The system ticks are described in Chapter Tick.

20.3.1 Injection of Learning Packets

The switch features a learning protocol to let all ports accept special learning packets to fully control the FIB. The learning packet format is shown in Figure 20.2. The MAC DA of a learning packet must match the address configured in the **Learning DA MAC** register. With a compliant MAC DA the packet is dropped inside the switch but the carried learning tag will be decoded and sent to the learning unit.

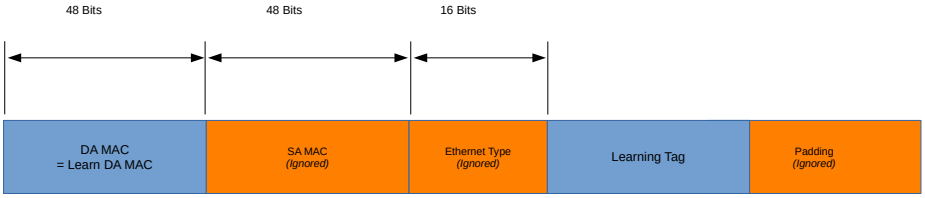


Figure 20.2: Learning Frame

The following table describes the fields which will be in the learning tag.

Name	Short Name	Field Size	Bits	Description
MAC	mac	48	[47:0]	MAC to learn.
GID	gid	16	[63:48]	Bit [15:12] Reserved. Bit [11:0] Global identifier to learn.
Unicast Port or Multicast Pointer	portOrPtr	32	[95:64]	Bit [31:11] Reserved. Bit [10:0] **destPort or mcAddr** field in the **L2 Destination Table**.
Unicast	uc	8	[103:96]	Bit [7:1] Reserved. Bit [0] **uc** field in the **L2 Destination Table**.
Drop	drop	8	[111:104]	Bit [7:1] Reserved. Bit [0] **pktDrop** field in the **L2 Destination Table**.
L2 Destination Table Address	daDestAddr	32	[143:112]	When the value is within the range 0 to 16415, the learning request will be updated to the corresponding entry in the **L2 Destination Table**. An out-of-range value will ask the learning unit to provide an available entry, and fill it with the data from the learning tag.
Valid	valid	8	[151:144]	Bit [7:1] Reserved. Bit [0] **valid** field in the **L2 Aging Table**.
Static	stat	8	[159:152]	Bit [7:1] Reserved. Bit [0] **stat** field in the **L2 Aging Table**.
Hit	hit	8	[167:160]	Bit [7:1] Reserved. Bit [0] **hit** field in the **L2 Aging Table**.
L2 Action Table DA Status	l2ActDa	8	[175:168]	Bit [7:1] Reserved. Bit [0] **l2ActionTableDaStatus** field in the **L2 Destination Table**.
L2 Action Table SA Status	l2ActSa	8	[183:176]	Bit [7:1] Reserved. Bit [0] **l2ActionTableSaStatus** field in the **L2 Destination Table**.
Tunnel Entry	tunnelEntry	8	[191:184]	Bit [7:1] Reserved. Bit [0] **tunnelEntry** field in the **L2 Destination Table**.
Tunnel Entry Pointer	tunnelEntryPtr	8	[199:192]	Bit [7:5] Reserved. Bit [4:0] **tunnelEntryPtr** field in the **L2 Destination Table**.

Name	Short Name	Field Size	Bits	Description
Tunnel Exit	tunnelExit	8	[207:200]	Bit [7:1] Reserved. Bit [0] **tunnelExit** field in the **L2 Destination Table**.
Tunnel Exit Pointer	tunnelExitPtr	8	[215:208]	Bit [7:5] Reserved. Bit [4:0] **tunnelExitPtr** field in the **L2 Destination Table**.
Meta Data	meta	16	[231:216]	**metaData** field in the **L2 Destination Table**.

Table 20.2: Learning Header

The fields in the learning tag consist of FIB fields and one 32-bit field for the L2 destination table address. Based on the different values, the learning tag can either directly update an entry in the FIB or ask the learning unit for an available entry. When the value is less than 16,416, the corresponding table entry will be updated directly, regardless of its current state. According to Section Hash Collision Accommodantion, the first 16,384 entries in the **L2 Destination Table** are reserved for **L2 DA Hash Lookup Table** hits and the rest are for **L2 Lookup Collision Table** hits, hence the L2 destination table address implies the address to the two search tables.

When the field value exceeds the range of the **L2 Destination Table**, the learning unit will excute the task to find an avaiable FIB entry, and update it with the corresponding information from the learning tag fields.

20.3.2 Direct Access to FIB

All tables in the FIB allow direct software writes through a configuration interface. However, the learning and aging engine may constantly update the FIB. Before updating the FIB from the configuration interface the learning and aging engine needs to be turned off through the **Learning And Aging Enable** register to avoid hazards. An alternative approach is to use reserved static entries as described in Section Software Reserved Entry.

If the hardware learning unit needs to be turned on again after software setups, it is important to write to both L2 aging tables and the corresponding shadow tables while setting valid entries. Partial validation will cause inconsistencies between the L2 forwarding process and the learning and aging engine. Since the FIB consists of multiple tables it is recommended that the shadow tables are updated in the last step, to ensure the data consistency.

20.3.3 Software Reserved Entry

If the **stat** field in the **L2 Aging Table** is set to 1 and the **valid** field is set to 0, the corresponding entry in the FIB is considered as a reserved static entry and can be used for future software configuration. A reserved static entry is not used for L2 forwarding and is not available as a hardware learning entry.

A typical use case is to pre-allocate entries for L2 multicast. The hardware learning unit can automatically learn L2 unicast but not L2 multicast. One way to reserve entries for L2 multicast is to create a reserved static bucket, i.e. choose one bucket from the L2 hash table and make all entries reserved static. This approach allows the software to update entries in the reserved bucket during traffic without checking hash collisions, and without turning off the hardware learning and aging engine.

20.3.4 Software Aging

The aging unit has a software aging mode which can take over the automatic aging turned on in the **Software Aging Enable** register. Under software aging mode the aging steps will then be:

1. Software determines the time to age and responsible to periodically trigger the aging process.
2. Software writes 1 to the **Software Aging Start Latch** register to trigger an aging check period.
3. The same procedure as the automatic aging is done, **hash_aging** and **cam_aging** interrupts listed in Table 34.7 are raised.

20.4 Software And Hardware Interaction

The three units in the learning and aging engine (learning unit, aging unit, hit update unit) share the same writeback bus to the FIB as in Figure 20.1, the learning unit has the highest priority, followed by the hit update unit and then the aging unit. In order to let software keep track of FIB updates from the learning and aging engine, the writeback bus is snooped and transactions are made available in three FIFOs. The FIFOs are accessible from the configuration interface.

- **Learning Data FIFO** (LDF)
- **Aging Data FIFO** (ADF)
- **Hit Update Data FIFO** (HDF)

20.4.1 Data FIFO Interrupts

For each of the three FIFOs there are two interrupts:

- High watermark interrupt: ldf_level/adf_level/hdf_level interrupt in Table 34.7. The threshold is configurable through:
  - **Learning Data FIFO High Watermark Level**
  - **Aging Data FIFO High Watermark Level**
  - **Hit Update Data FIFO High Watermark Level**
- Overflow interrupt: ldf_full/adf_full/hdf_full interrupt in Table 34.7

The LDF/ADF/HDF are all tail drop FIFOs, if new entries are to be pushed to a full LDF/ADF/HDF they will not be written but ignored and cause ldf_full/adf_full/hdf_full interrupt. The HDF holds the hit update result which does not change L2 forwarding behaviors, but if software is unable to keep up reading out the LDF/ADF and cause ldf_full/adf_full interrupt, then software is no longer in sync with hardware tables. A way to recover from this would be:

1. Turn off the learning and aging engine.
2. Read out all the entries in the LDF/ADF/HDF to make sure they are empty.
3. Read out all tables in the FIB to compare between software tables. Update whatever the difference is to make tables become synchronized again.
4. Turn on the learning and aging engine.

20.4.2 Writeback Bus Control

As mentioned in Section Hardware Learning and Aging, the writeback bus can be configured through the **Learning And Aging Writeback Control** register to block the hardware learning/aging/hit-update decisions to the FIB. By doing so the automatic hardware learning/aging/hit-update units cannot do any changes to the FIB. If needed, the software is able to inspect the hardware decisions from LDF/ADF/HDF and update the FIB either through learning packets or direct table accesses.

Chapter 21

Spanning Tree

Spanning-Tree Protocol (STP) and Multiple Spanning-Tree Protocol (MSTP) support is provided in order to create loop-free logical topology when several ethernet switches are connected. Through registers the STP state of the ports can be controlled by the host SW. The default behavior at power up is that spanning tree is not enabled and spanning tree functionality must therefore be configured by SW before it can be used. A switch running the spanning-tree protocols utilizes BPDU (Bridge Protocol Data Unit) frames to exchange information with other switches in order to decide how to configure it's ports to get a loop-free (tree) logical network topology.

BPDUs are forwarded to the CPU based on the used destination address. By default the MAC multicast addresses 01:80:C2:00:00:00 and 01:00:0C:CC:CC:CD are forwarded to the CPU. Modifications of this is possible through the register **Send to CPU**.

In order to be able to forward BPDU frames from the CPU to other switches on egress ports where general forwarding is currently not allowed, the bit **enable** in register **Forward From CPU** shall be set.

More information on the forwarding features to and from the CPU port is available in Chapter 33

21.1 Spanning Tree

The Spanning-Tree Protocol (STP) state for a port can be independently configured for source and egress behaviors to allow precise management. For ingress in the **spt** field of **Source Port Table**. Similarly for egress, the STP state can be configured in the **sptState** in the **Egress Spanning Tree State**. When STP is used on a port, all the port's associated MSTP instance states (ingress and egress) shall be **Forwarding**, i.e. MSTP is not enabled for this port.The behavior of the different STP states. The difference between Ingress and Egress STP state is only that learning is not affected by the Egress state.

- Blocking and Listening
  Learning is disabled and no frames are forwarded except BPDU which will be forwarded to the CPU. Frames that are not forwarded is counted in a drop counter.
- Learning
  Learning is enabled but no frames are forwarded except BPDU which will be forwarded to the CPU. Frames that are not forwarded is counted in a drop counter.
- Forwarding and Disabled
  Normal operation, learning is enabled and normal switching. BPDU frames will be forwarded to the CPU.

21.2 Multiple Spanning Tree

When VLANs are used in a network there is a need for the Multiple Spanning Tree Protocol (MSTP) to manage the individual spanning-tree instances for the different VLANs. If an incoming frame doesn't have an assigned VLAN membership it will get a default VLAN membership automatically as described in Chapter 5. VLAN membership decides which MSTP instance (MSTI) the frame belongs to. Hence, all frames will belong to an MSTI. The **msptPtr** in the register **VLAN Table** is an index to the MSTI tables which the packet shall be assigned to. The port's states of this MSTI are available in the tables **Ingress Multiple Spanning Tree State** and **Egress Multiple Spanning Tree State** for ingress and egress respectively. When a port uses MSTP it's STP states (source and egress) shall be set to **Disabled**, i.e. STP is not enabled for this port.

21.3 Spanning Tree Drop Counters

When a port's ingress or egress spanning tree states causes a frame to be dropped, the frames direction and spanning-tree state are used to select which drop counter to increase with one. The available drop counter registers are:

- **Ingress Spanning Tree Drop: Listen**
- **Ingress Spanning Tree Drop: Learning**
- **Ingress Spanning Tree Drop: Blocking**
- **Egress Spanning Tree Drop**

The ingress registers are common for all ports. There is one egress register per port.

The registers above are also used to count MSTI-state caused frame drops. A port's ingress-MSTI drop-causing state is mapped as follows: The state **Learning** is mapped to the register **Ingress Spanning Tree Drop: Learning** and **Discarding** to **Ingress Spanning Tree Drop: Blocking**. For a port's egress MSTI, both the states **Learning** and **Discarding** are mapped to the port's generic egress drop counter **Egress Spanning Tree Drop**.

Chapter 22

Token Bucket

This core provides a rich set of QoS functions, and when a function needs to compare the internal packet or byte rate to a configurable rate, we use token bucket as the basic measurement component. A token bucket is usually combined with packet classifications, packet colorings or the shared buffer memory to achieve metering, marking, policing or shaping with different granularities.

A token bucket has four key parameters:

- bucket capacity
- bucket threshold
- initial tokens in the bucket
- token fill in rate

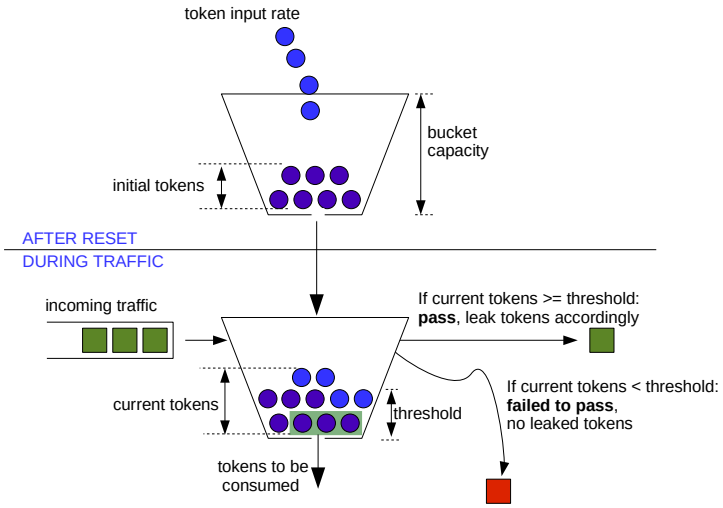


Figure 22.1: General Token Bucket Illustration

Figure 22.1 shows a token bucket with adjustable bucket threshold, the remaining tokens below the threshold can be used to handle the burst. This type of token bucket is used by:

- multicast broadcast storm control
- queue shaper
- prio shaper
- egress port shaper

In different QoS functions, tokens are represented as packets or bytes. The token fill in rate is achieved by periodically adding a certain number of tokens to the bucket and the fill in frequency is determined by one of the five core ticks.

Chapter 23

Egress Queues and Scheduling

The order of packet output on each egress port is decided by a complex interaction of back-pressure and different QoS functions, but at the heart of the matter is the the egress queue. The egress queues are the lists of packet pointers created by the queue manager when packets have been written to the packet buffer. Each egress port has eight such queues.

When a packet has been written in full to the packet buffer, the queue manager will add pointers to the packet to the end of at least one egress queue[1].

More than one egress port may get the packet linked (due to multicast), but on any single port the same packet may only be linked once. You cannot have the same packet in more than one egress queue on any single egress port.

The order in each egress queue is fixed. Once the packets are linked, the order cannot be changed. What QoS functions and back-pressure can affect is the order in which the packets in different queues are output.

Each egress queue has a *priority* (or prio) attribute, ranging from zero to seven. There are no limitations to how the priorities are assiged. All egress queues may have the same priority, or they may all have different priorities (if there are enough priorities to go around). If at all possible, an egress queue with a higher[2] priority will always get to output a packet before a queue with a lower priority. Egress queues with the same priority will be selected in a round robin manner by the DWRR scheduler.

The egress queue is determined by the ingress packet processing. If a packet is forwarded to multiple egress ports, each packet instance will have the same egress queue assigned.

23.1 Determine Egress Queue

Figure 23.1 describes how the egress queue is determined. If a configuration in the diagram includes a reference number in the end, the related field or register to setup can be found in the list below:

1. **Configurable ACL Engine** has a forceQueue action enabled.
2. **forceQueue** in **Reserved Source MAC Address Range**
3. **forceQueue** in **Reserved Destination MAC Address Range**
4. **prioFromL3** in **Source Port Table**
5. **IPv4 TOS Field To Egress Queue Mapping Table**
6. **IPv6 Class of Service Field To Egress Queue Mapping Table**
7. **MPLS EXP Field To Egress Queue Mapping Table**
8. **eQueue** in **Force Unknown L3 Packet To Specific Egress Queue**
9. **forceQueue** in **Force Non VLAN Packet To Specific Queue**

This process is completed only once per packet, and the result is applied to all destination ports for the packet. The input to the process can come from:

[1] That is unless the packet is to be dropped, because then the pointer is instead added to the end of the throw queue.
[2] Priorities are numbered backward, so zero is the highest priority

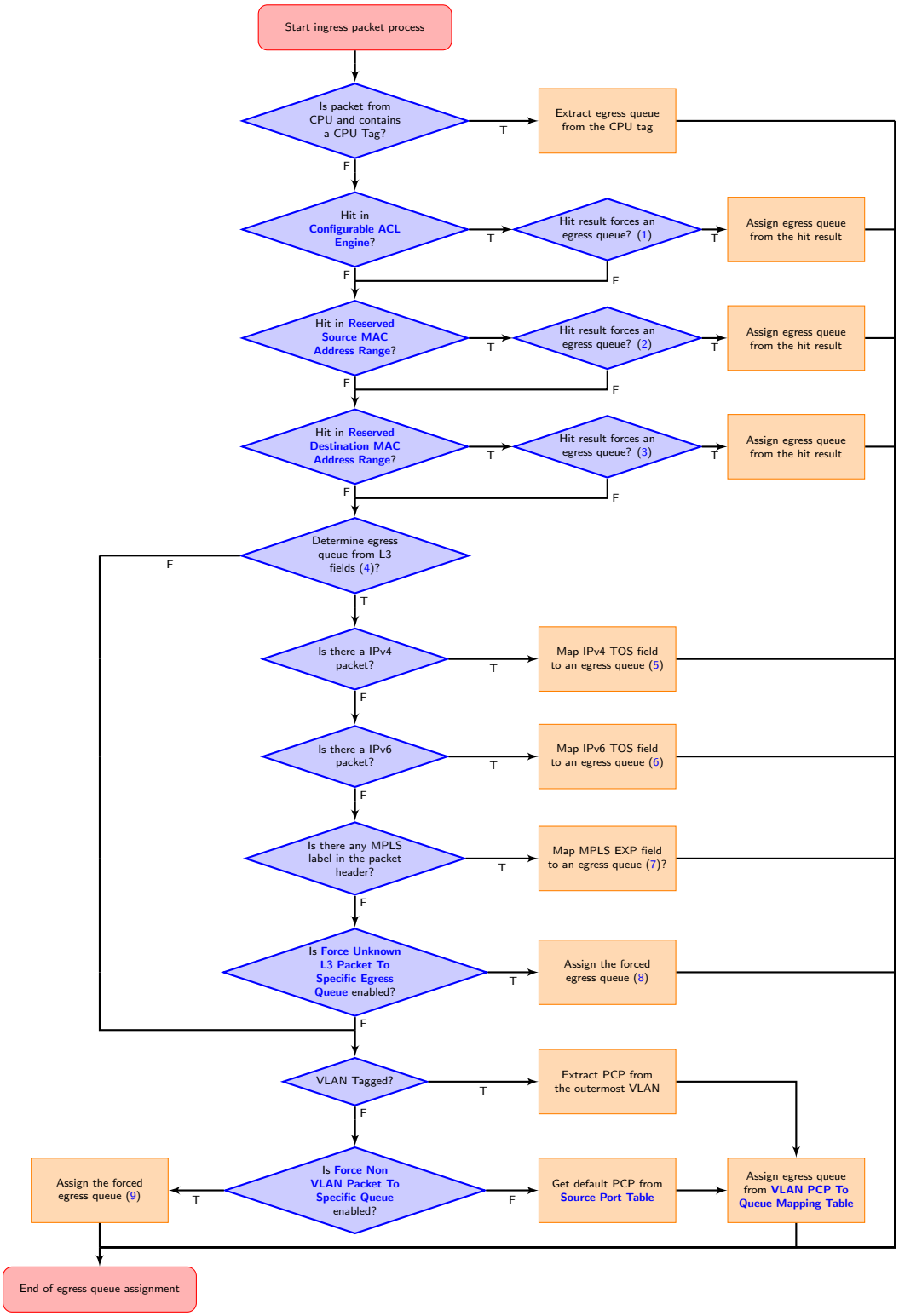


Figure 23.1: Egress Queue Selection Diagram

- Packet L2 headers
- Packet L3 headers
- Packet L4 ports
- Classification results

The available classification engines are described in the Classification chapter.

Egress queue from packet headers is operated under either trust L2 mode, to map egress queues from L2 headers, or trust L3 mode, to map egress queues from both L2 and L3 headers. In trust L2 mode, the egress queue can be mapped from:

- Priority code point(PCP) field from the outermost VLAN tag
- Source port default PCP when packet is non-VLAN tagged
- Optionally force non-VLAN tagged packets to the same egress queue, ignores source port based default mapping.

In trust L3 mode, a packet first tries to get its egress queue by mapping from:

- Type of Service (TOS)/DiffServ field from IPv4
- Traffic Class(TC) field from IPv6
- Traffic Class(TC)/EXP field from MPLS
- When none of the above are executed, the egress queue mapping under trust L3 mode will fall back on the trust L2 mode and get the egress queue from L2 headers of the packet.

23.2 Determine a packets outgoing QoS headers PCP, DEI and TOS fields

23.2.1 Remap Egress Queue to Packet Headers

This core supports remapping determined egress queues to outgoing packets' headers. These remappings are done first then if field **useEgressQueueRemapping** is set to one the remapping described in 23.2.2.

- Egress queue to next hop router VLAN PCP remapping:
  For routed packets, packets' original VLAN tags are removed and at most two next hop router VLANs are added. Egress queue can be mapped to the PCP field in these VLAN tags through the **Router Egress Queue To VLAN Data** table when:
  1. **innerVlanAppend** is set and its PCP field selection(**innerPcpSel**) chooses mapping from egress queue.
  2. **outerVlanAppend** is set and its PCP field selection(**outerPcpSel**) chooses mapping from egress queue.
- Egress queue to next hop router VLAN CFI/DEI remapping:
  Similar with next hop router VLAN PCP mapping, egress queue can be mapped to the CFI/DEI field in next hop router VLANs through the **Router Egress Queue To VLAN Data** table when:
  1. **innerVlanAppend** is set and its CFI/DEI field selection(**innerCfiDeiSel**) chooses mapping from egress queue.
  2. **outerVlanAppend** is set and its CFI/DEI field selection(**outerCfiDeiSel**) chooses mapping from egress queue.
- Egress queue to outgoing outermost VLAN PCP remapping:
  Egress port VLAN push or swap operation provides an option to map egress queue to the outgoing outermost VLAN PCP field. The mapping table is **Egress Queue To PCP And CFI/DEI Mapping Table** and the required configurations are:
  1. **vlanSingleOp** in **Egress Port Configuration** is *push* or *swap*.
  2. **pcpSel** in **Egress Port Configuration** selects mapping from egress queue.
- Egress queue to outgoing outermost VLAN CFI/DEI remapping:
  Similar with outgoing outermost VLAN PCP mapping, egress port VLAN push or swap operation provides an option to map egress queue to the outgoing outermost VLAN CEI/DEI field. The mapping table is **Egress Queue To PCP And CFI/DEI Mapping Table** and the required configurations are:
  1. **vlanSingleOp** in **Egress Port Configuration** is *push* or *swap*.
  2. **cfiDeiSel** in **Egress Port Configuration** selects mapping from egress queue.

- Egress queue to MPLS TC/EXP remapping:
  Packets with MPLS labels have an option to map their egress queues to MPLS TC/EXP field when egressing the core. The mapping table is **Egress Queue To MPLS EXP Mapping Table** and the required configurations are:
  1. **mplsOperation** is *push* or *swap*.
  2. **expSel** in **Next Hop MPLS Table** selects mapping from egress queue.

23.2.2 Using Packet Type, Destination Port and Switching/Routing to do QoS Mappings

This core supports remapping determined by egress queues to outgoing packets' headers using the information if the packet was switched, routed, forwarded by classification rules, if the packet type was IP or MPLS and packets outgoing PCP, DEI, TOS and EXP fields. The steps to remap the packet are described below. The input values for PCP, DEI comes from the remapping tables described in 23.2.1. The TOS values comes from the **Color Remap From Ingress Admission Control** or **Color Remap From Egress Port**.

1. Determine Which Mapping Table To Use
   The mapping table to use to map the internal state to a the outgoing packet is determined by the table **Select Which Egress QoS Mapping Table To Use**. The packets destination port, packet type and packet forwarding type is used to calculate which entry to read out from the table. This table then points to the one of the QoS remapping tables which remapps the internal state to the outgoing packets PCP,DEI and potentially L3 fields such as TOS field . Since the address takes egress port, forwarding type and packet type into consideration there can be seperate rules setup for how to remap the fields in the outgoing packet.
2. Mapping Tables
   Use the Mapping tables to map into outgoing packets PCP,DEI , TOS and EXP values.
   (a) **L2 QoS Mapping Table**
   This table can be used for all packets being sent out. There exists 2which the field **whichTablePtr** points to which to use.
   (b) **IP QoS Mapping Table**
   This table can be used for IPv6 and IPv4 packets. There exists 2L3 mapping tables. This remaps part of the TOS byte which has to do with ECN and uses the higher TOS bits [7:2] from the coloring tables (**Color Remap From Ingress Admission Control** or **Color Remap From Egress Port**).
   (c) **TOS QoS Mapping Table**
   This table can be used for IPv6 and IPv4 packets. There exists 2TOS mapping tables. This remaps the whole of the TOS byte from **Color Remap From Ingress Admission Control** or **Color Remap From Egress Port** to a new TOS bytes along with PCP and DEI information. There is a support to remap to EXP values which can be used if the packet enters a MPLS tunnel in the Next Hop Tables
   (d) **MPLS QoS Mapping Table**
   This table can be used for MPLS packets. This remaps the outgoing packets PCP, DEI and EXP values. There exists 2TOS mapping tables.

23.3 Priority Mapping

Each queue is mapped to one of eight egress priorities in the **Map Queue to Priority** register. Thus each priority will have between none and all queues as members. The priority mapping affects the scheduling of the packets. See Section 23.6, below for the details.

The priorities are ranked in descending order, from the highest priority (zero), to the lowest (seven).

Note that the priority mapping must not be changed for any queue that has packets queued. Doing so would make the ERM counters irrevocably corrupted, necessitating a reset for the core to continue normal operation.

23.4 Shapers

For a queue to be eligable for sending a packet there has to be a packet available in the queue and the average bandwidth for the queue, as measured by the token buckets in the queue shaper, has to be below the threshold set up in the **Queue Shaper Rate Configuration** registers.

Additionaly the average bandwidth of the priority to which the queue is mapped has to be below the threshold set up in the **Prio Shaper Rate Configuration** registers.

23.4.1 Queue Shaper

The egress queue rates are shaped by token buckets configured in the **Queue Shaper Rate Configuration** registers. While the token bucket level is below the threshold configured in the **Queue Shaper Bucket Threshold Configuration** register, no new packets are scheduled for the corresponding egress queue. Ongoing packets are not affected by the shaping bucket status.

The queue shapers are enabled using the **Queue Shaper Enable** register, and the saturation level of the queue shaper buckets is controlled by the **Queue Shaper Bucket Capacity Configuration** register.

23.4.2 Prio Shaper

The egress prio rates are shaped by token buckets configured in the **Prio Shaper Rate Configuration** registers. While the token bucket level is below the threshold configured in the **Prio Shaper Bucket Threshold Configuration** register, no new packets are scheduled for the corresponding egress prio. Ongoing packets are not affected by the shaping bucket status.

The prio shapers are enabled using the **Prio Shaper Enable** register, and the saturation level of the prio shaper buckets is controlled by the **Prio Shaper Bucket Capacity Configuration** register.

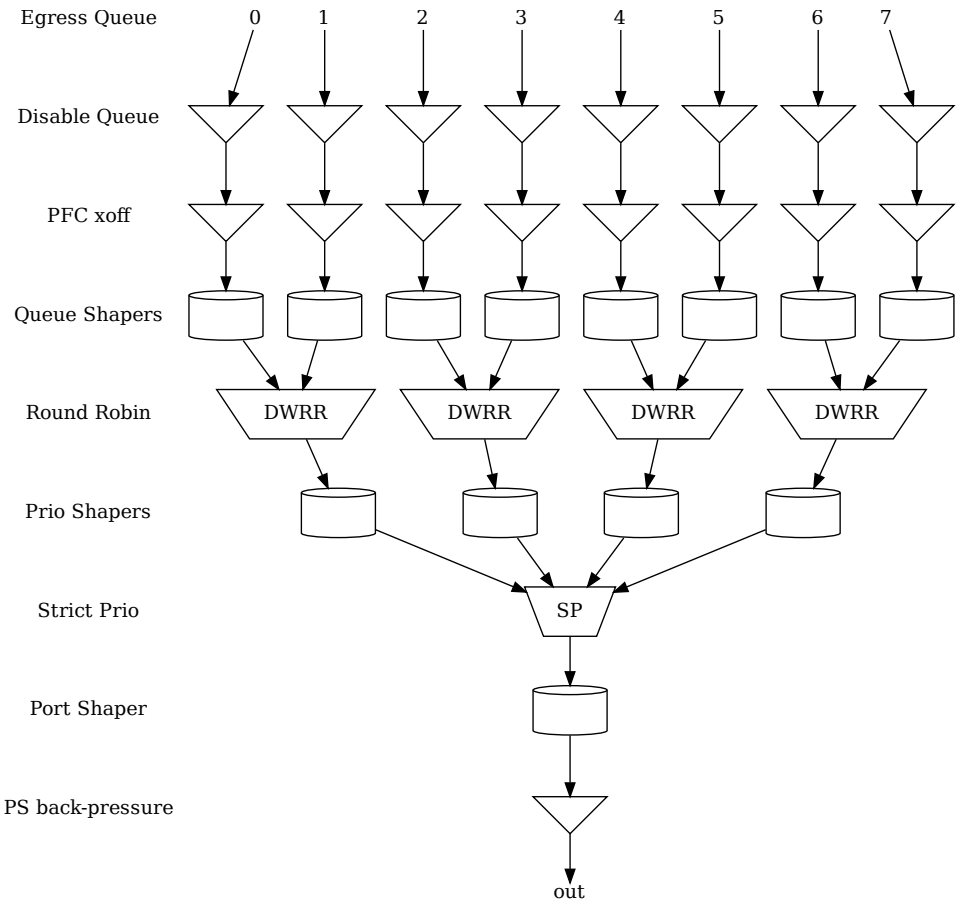


Figure 23.2: Egress Queue Scheduling example. Here using half the priorities, with two queues mapped to each.

23.5 Scheduling

The egress queue scheduling is accomplished by a combination of strict priority schedulers for the priorities and round robin queue schedulers for the queues mapped to the same priority. A visual representation of this is can be found in Figure 23.2. This figure is an example where half the priorities are used and two queues map to each priority[3].

For a priority to be allowed to output a packet it must have mapped queues with available packets. It must also:

- be allowed to send by the prio shaper
- not be paused
- not be halted

From the priorities getting through the above needle's eye the highest priority is selected, and then the available queues mapped to that priority are selected by a byte-based deficit weighted round robin scheduler (described below).

23.6 DWRR Scheduler

The DWRR scheduler only acts on queues mapped to the same priority. Within each group of such queues it selects the most appropriate queue to output by comparing the number of bytes output for each queue with the weights set up for the queues.

This is accomplished using one byte counting bucket per queue and port. The non-empty queue with the highest bucket count in the group is selected. Bytes are subtracted from the corresponding bucket when a packet is sent out. Whenever the value in a bucket goes below the value configured in the **threshold** field of the **DWRR Bucket Misc Configuration** register, the buckets for all the queues belonging to the same priority will be replenished. The number of bytes added to each bucket is **weight** $<< X$, where weight is taken from the **DWRR Weight Configuration** register, and X is a multiplier (for all queues) that is calculated to make sure that at least one cell worth of bytes is added to the queue that went below the threshold.

$X = max(0, highestSetBit(cellBytes) - highestSetBit(weigth))$

If a queue has no data to send, its bucket will eventually saturate at the cap set in the **DWRR Bucket Capacity Configuration** register.

The value in the **ifg** field of the **DWRR Bucket Misc Configuration** is additionally subtracted from the buckets for each packet.

23.7 Queue Management

This core features a set of queue management operations which can be used by the CPU to monitor, redirect and disable queues and ports. The current size of the queues can be readout by using the **Egress Port Depth** and **Egress Queue Depth** registers, while the current total number of cells left available can be seen in the **Buffer Free** register. The minimum level reached since core was initialized is available in **Minimum Buffer Free**. From this status the CPU can take active actions to determine what the core shall do with the packets on the ports. The optional operations are listed below.

- Disable scheduling to port: Disable the core from scheduling a new packet for transmission on a specific port and queue. This is setup in the **Output Disable** register. This allows per-queue granularity of what packets gets scheduled on a specific port. The packets are still kept in the queues until the port or queue is enabled again.
- Disable queueing to port: Disable the enqueueing of packets to a specific port and queue. Once the corresponding bit in the **Enable Enqueue To Ports And Queues** register is cleared, no new packets will be queued to that egress queue. New packets destined to that specific queue will be dropped and the **Queue Off Drop** counter for the egress port will be incremented.
- Drain port: Drop all packets in all queues on one specific port. This allows the user to clear all packets which have been queued on a port. The register **Drain Port** is used to control this functionality. Statistics for this operation is collected in the **Drain Port Drop** counter.

[3]So other similar diagrams would result with different settings in the **Map Queue to Priority** register.

23.8 How To Make Sure A Port Is Empty

First, so that no new packets are queued to the port, use the **Enable Enqueue To Ports And Queues** to disable all the queues on the port. If the already queued packets should not be sent out, then use the **Drain Port** functionality. Once this is done start to read out the **Packet Buffer Status** and check the bit which corresponds to the port. When the port bit is high, this means that all the queues on this port are empty.

Now, there may still be a few cells of data being processed in the egress packet processing pipeline, or stored in the parallel-to-serial memories. This data will be drained at the speed of the port, so the time from the port-bit going high in the **Packet Buffer Status** register to the port being truly empty will depend on the port speed.

Chapter 24

Packet Coloring

24.1 Ingress Packet Initial Coloring

This core marks packets with 3 colors internally to represent packet drop precedences. The three colors are coded as in Table 24.1.

Color	Code
Green	0
Yellow	1
Red	2

Table 24.1: Code for Colors

A packet's initial color is assigned according to L2/L3 protocols or classification results. It follows similar process steps as the egress queue assignment described in Section 23.1.

1. **Configurable ACL Engine** has a forceColor action enabled.
2. **forceColor** in **Reserved Source MAC Address Range**
3. **forceColor** in **Reserved Destination MAC Address Range**
4. **colorFromL3** in **Source Port Table**
5. **IPv4 TOS Field To Packet Color Mapping Table**
6. **IPv6 Class of Service Field To Packet Color Mapping Table**
7. **MPLS EXP Field To Packet Color Mapping Table**
8. **forceColor** in **Force Unknown L3 Packet To Specific Color**
9. **forceColor** in **Force Non VLAN Packet To Specific Color**

A diagram in Figure 24.1 describes how initial colors are determined. All classification engines which can force egress queues also have an option to force packet initial colors. If none of the engines force the color and the initial color marking is operating under trust L2 mode, the color is mapped from:

- Priority Code Point(PCP) field with Drop Eligible Indicator(DEI) field from the ingress outermost VLAN tag.
- Source port default PCP with default DEI when packet is non-VLAN tagged.
- Optionally force non-VLAN tagged packets to the same specific initial color, ignores source port based default marking.

Otherwise, the initial color marking will be working under trust L3 mode and the color is mapped from:

- Type of Service(TOS)/DiffServ field from IPv4
- Traffic Class(TC) field from IPv6
- Optionally force non-IP packets to the same initial color.

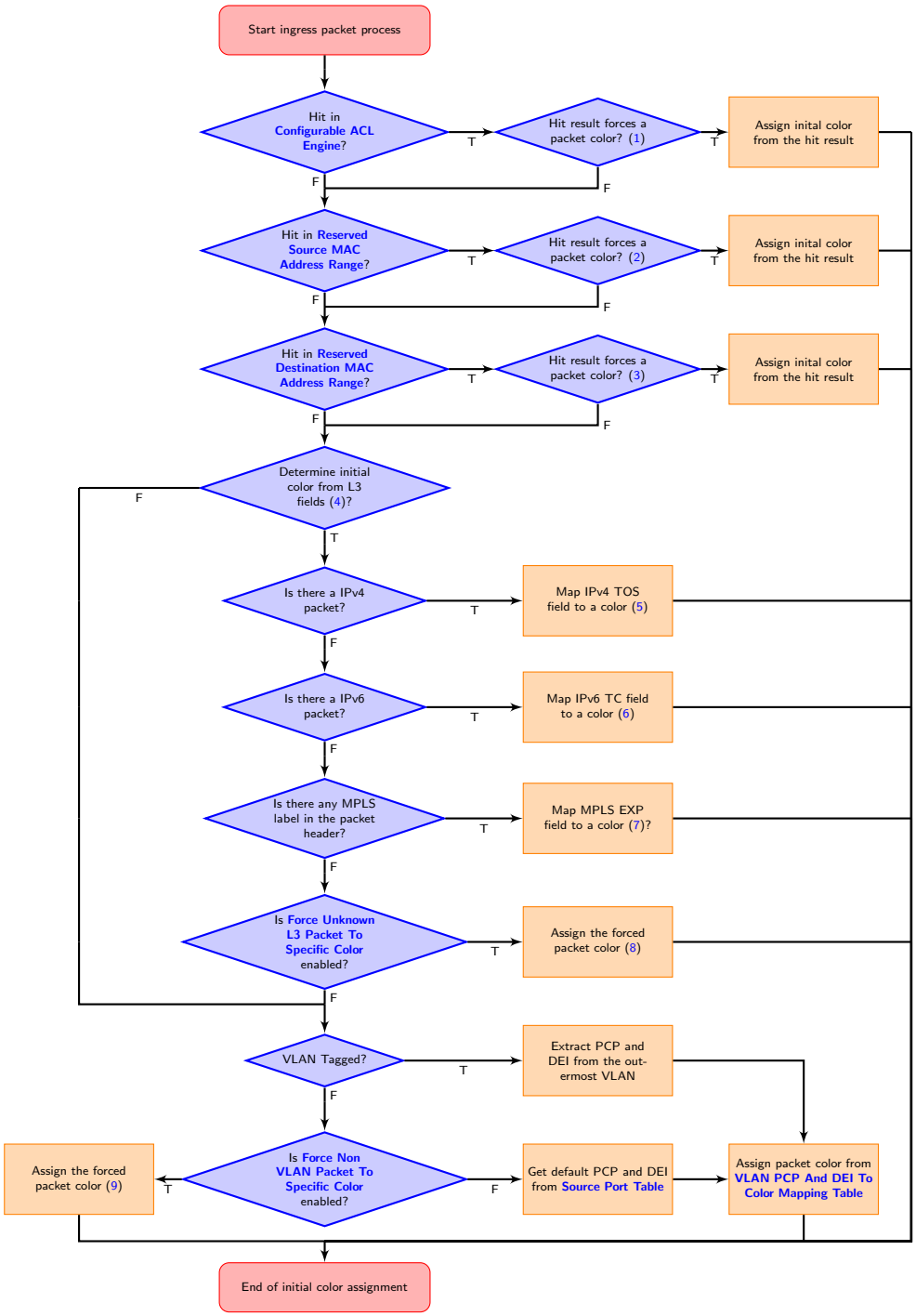


Figure 24.1: Packet Initial Color Selection Diagram

- When none of the above markings are executed, the initial color marking under trust L3 mode falls into processes in trust L2 mode.

By default, green marked packets have low drop probability, yellow marked packets have medium drop probability and red marked packets have high drop probability. But the remarking process has its own configurable settings to decide if packets with a certain remarked color shall be dropped.

24.2 Remap Packet Color to Packet Headers

During egress packet processing, each egress port can be set as color aware or color blind through the **colorRemap** field in the **Egress Port Configuration** table. If an egress port is color blind, packets to that port will not have its color represented in packet headers. If an egress port is color aware, a color remap process is executed to optionally remap the egress packet color to outgoing packet headers.

When an egress port is color aware, the default remap options for that port are configured in the **Color Remap From Egress Port** table. If a packet to a color aware egress port has ingress admission control applied, its meter-marker-policer pointer can also provide color remap options from the **Color Remap From Ingress Admission Control** table. The **enable** field in the table determines whether to perform a color remap operation for each pointer.

The color remap has four modes:

- Skip/Disable:
  Color is not remapped to packet headers. This includes overriding previous color remap decisions.
- Remap to L3 only:
  Color is remapped to IPv4 TOS field or IPv6 TC field with an AND mask (tosMask). For each bit in the TOS/TC field, the update requires the corresponding bit in the mask set to one. i.e.

```
tos[i] = ( color2Tos[i] & tosMask[i] ) | ( tos[i] & (~tosMask[i]) )
```

- Remap to L2 only:
  A valid color remap updates the DEI bit in the VLAN tag of the outgoing packet. The updated DEI bit will not be changed during further egress packet processes. If there are more than one VLAN tag in the transmitted packet, the color to DEI mapping will be operated on the outermost VLAN.
- Remap to L2 and L3:
  Color is remapped to both L2 and L3 fields as listed above.

Chapter 25

Admission Control

25.1 Ingress Admission Control

This core features an ingress admission control unit to control the bandwidth of certain traffic types. If the traffic flow in a group exceeds the configured bandwidth it may get the packet color changed or get denied to be enqueued in the buffer memory.

Ingress admission control includes two main functions. The first function creates admission control groups to classify packets based on source information in packet headers or ACL matches. The second function measures the classified traffic rate against a certain policy to make permit/deny decisions. The decision may take the given packet color into account.

25.1.1 Traffic Groups

The traffic group is classified based on source port number and L2 or L3 packet headers. Initially packets are grouped by their source port numbers and L2 priorities, but during the subsequent admission control processes they may fall into other traffic groups. For each potential traffic group, three configurations are given to validate a policy:

1. mmpValid: Determine if there is a valid Meter-Marker-Policer(MMP) pointer. If there is no valid pointer through the entire process, the packet will not be classified to any traffic group.
2. mmpOrder: Order of the pointer. If a valid pointer exists, its order needs to be higher than the order of previously assigned pointers to override them.
3. mmpPtr: MMP pointer for this traffic group.

The process to set the MMP pointer is illustrated in Figure 25.1. A packet can only belong to one traffic group so hierarchical traffic groups are not possible.

The order of the classification sequence is:

1. Source port number and L2 priority:
   First assignment for traffic groups and MMP pointers. For VLAN tagged packet, L2 priority is from its outermost VLAN PCP field. For non-VLAN tagged packet, L2 priority is the default PCP based on the source port number (**defaultPcp** in the **Source Port Table**). Lookup in the **Ingress Admission Control Initial Pointer** table gives a base pointer and its order, also indicates if it is a valid pointer.
2. Source MAC:
   Source MAC hit an entry in the **Reserved Source MAC Address Range**.
3. Destination MAC:
   Destination MAC hit an entry in the **Reserved Destination MAC Address Range**.
4. ACL rules:
   Hit in the **Configurable ACL Engine**.
5. Ingress VID:
   Lookup in **VLAN Table** based on the **ingress VID**.
6. VRF:
   For a routed packet, lookup in **Ingress Router Table** based on its VRF.

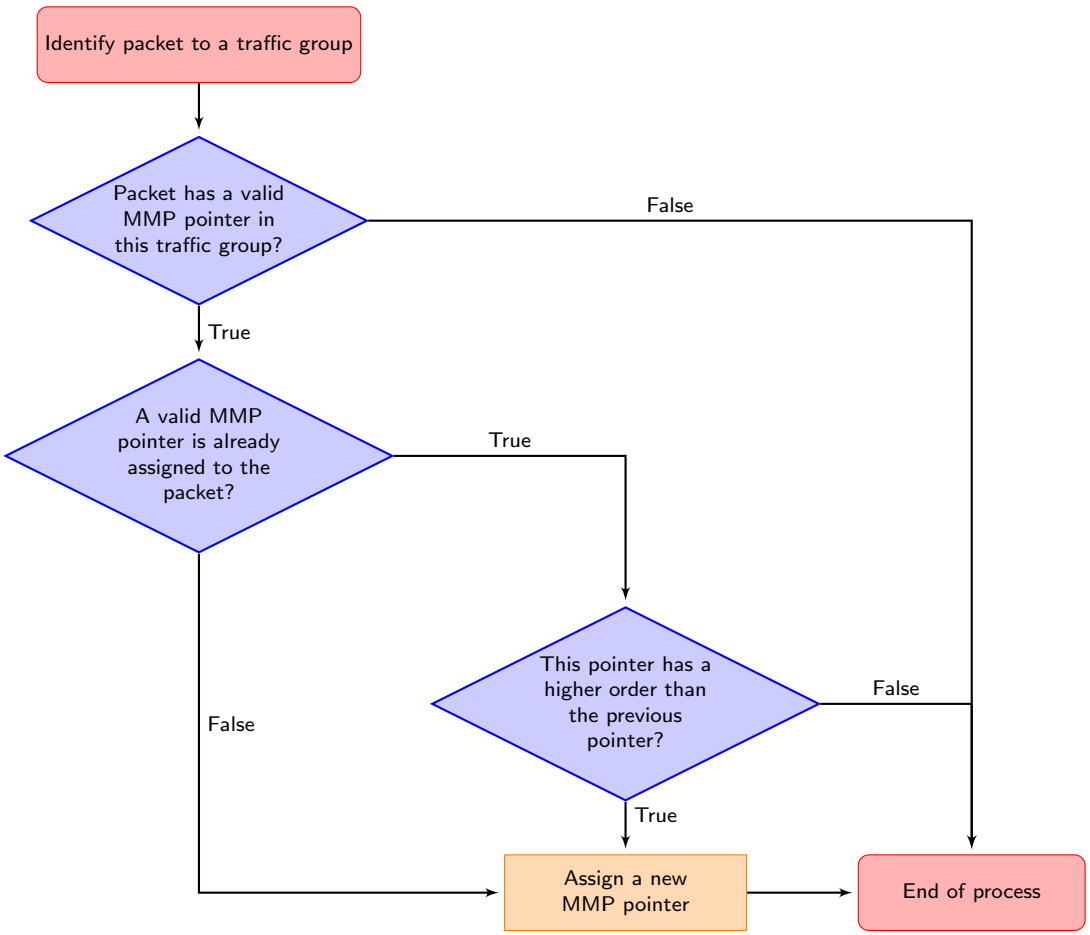


Figure 25.1: MMP pointer Selection Diagram

When a packet arrives to ingress packet processing, it walks through ingress admission control classifications in the order above. A hit in one of the above groups will result in a pointer and a matching order. The pointer is linked to a policy/entry in a meter-marker-policer engine, which will measure the byte rate belonging to this entry. Although a packet can have multiple hits in traffic groups, it will finally fall into one pointer according to the order of the pointers. Later matches only win when they have a higher order than the previous ones.

25.2 Meter-Marker-Policer

An admission control unit contains a meter-marker-policer (MMP) bank where each MMP refers to one admission control policy. An MMP is based on token buckets, and each entry includes two configurable buckets.

The MMP bank used by ingress admission control consists of 64 policies/entries with three related tables.

1. **Ingress Admission Control Token Bucket Configuration**
2. **Ingress Admission Control Reset**
3. **Ingress Admission Control Current Status**

While only one ingress admission control policy is applied to any single packet, the same policy/entry can be pointed to from several different traffic types.

In the Ingress Admission Control , an MMP entry is configured through the **Ingress Admission Control Token Bucket Configuration** register to perform either a single rate three color marker (RFC2697: srTCM) or a two rate three color marker (RFC2698: trTCM). The selected marker is operated in either color-aware or color-blind mode, and the packet is marked with a new color when the rate exceeds a certain bandwidth. Based on the updated

packet color, **dropMask** from register **Ingress Admission Control Token Bucket Configuration** decides whether the packet is allowed to be enqueued in the buffer memory.

An MMP entry has a **Ingress Admission Control Mark All Red Enable** option to permanently block the metering process and drop all packets with the corresponding MMP pointer. When **Ingress Admission Control Mark All Red Enable** is set to one, a packet drop on this entry will raise the **Ingress Admission Control Mark All Red** to one, then further packets to that entry will be dropped before metering. The blocking status can be cleared by writing zero to one of the two registers.

When an MMP is selected to be either srTCM or trTCM, it still requires configurations of the two token buckets to make it work properly.

- srTCM: Only the length, not the peak rate of the burst determines service eligibility.
  - Committed Information Rate (CIR): Combining **tokens 0** and **tick 0** to achieve the target rate. Details for tick is described in the **Tick** chapter. Configuration examples are shown in Table 25.1. Under srTCM mode, rate settings for the second token bucket (**tokens 1** and **tick 1**) will not take effect.
  - Committed Burst Size (CBS): **bucketCapacity 0**.
  - Excess Burst Size (EBS): **bucketCapacity 1**.
- trTCM: Enforce peak rate separately from the committed rate.
  - Committed Information Rate (CIR): **tokens 0** and **tick 0**.
  - Committed Burst Size (CBS): **bucketCapacity 0**.
  - Peak Information Rate (PIR): **tokens 1** and **tick 1**.
  - Peak Burst Size (PBS): **bucketCapacity 1**.
- Runtime configuration update:
  Any update to register **Ingress Admission Control Token Bucket Configuration** requires writing 1 to register **Ingress Admission Control Reset**. This will reset the buckets to the initial state.
- Status update from hardware:
  Besides **Ingress Admission Control Reset**, MMP has a another status register: **Ingress Admission Control Current Status**. It shows the number of tokens in each bucket. Hardware updates these two registers only when a metering process is done, hence **Ingress Admission Control Current Status** shows the number of tokens left in the bucket since the last token consumption in this bucket. **Ingress Admission Control Reset** is always changed back to 0 again after token consumptions.

Bandwidth	Token Bucket Update Frequency	Tick Index	Added Tokens Per Tick (bytes)
8000 bit/s	1KHz	3	1
16000 bit/s	1KHz	3	2
N*64000 bit/s	1KHz	3	N*8
N*1544000 bit/s	1KHz	3	N*193
N*56000 bit/s	1KHz	3	N*7
10M bit/s	10KHz	2	125
250M bit/s	10KHz	2	3125
N*1G bit/s	1Mhz	0	N*125

Table 25.1: Rate Configuration Example (Assume tickFreqList = [1MHz, 100KHz, 10KHz, 1KHz, 100Hz])

Chapter 26

Table Synchronization

This chapter describes the synchronization of multiple physical tables provided in the core. A challenge arises when updating these tables: incoming packets may access an entry simultaneously being modified by the system software, leading to erroneous data in outgoing packets. The acceptability of this scenario varies depending on the final system implementation.

The complexity increases when a table is involved in both ingress and egress packet processing. In such cases, buffer memory might contain a mix of packets—some requiring the old value, while others need the new updated value.

To mitigate these issues, each entry in specific tables includes a version field. Packet processing and transmission only occur if all linked entries share the same version number. If there is a discrepancy in versioning, the packet is discarded, and a corresponding counter is incremented.

Two distinct counters are maintained: one for Ingress Packet Processing (IPP), denoted as **Ingress Table Not In Sync Drop**, and another for Egress Packet Processing (EPP), denoted as **Egress Table Not In Sync Drop**.

26.1 NAT

The NAT functionality utilizes a pointer from both ingress and egress ACLs, each with a version number that is compared to the egress pipeline NAT entries version number.

- The ingress NAT tables.
  Specifically using the ingress classification/ACL along with the egress table located in **Ingress NAT Operation**
- The egress NAT tables.
  Specifically using the ingress classification/ACL along with the egress table located in **Egress NAT Operation**

26.2 Routing

Routing tables involve several steps, encompassing both ingress and egress tables. All tables in both ingress and egress must share the same version number; otherwise, packets are discarded.

- The routing commences with the **Hash Based L3 Routing Table** or **L3 LPM Result**.
- The default route, **L3 Routing Default**, includes a version field which is used if the Hash or LPM table misses.
- Then the version value from this initial table is checked in ingress with result from **Next Hop Table** and **Next Hop Packet Modifications**.
- The routing entry version is stored with the packet in buffer memory.
- In the egress packet processing the entry version from ingress is compared with the entry in **Next Hop DA MAC** , and table **Next Hop MPLS Table**.
- In the egress if the packet shall insert a MPLS header to reach the next hop then **Next Hop Packet Insert MPLS Header** must also have a correct version field.

Chapter 27

Tick

All token buckets - and all other functions dependent on measuring time - in the core are basing their time measurements on the system ticks.

Tick number zero is the master tick. It is created by dividing the core clock by the number configured in the clkDivider field of the **Core Tick Configuration** register. The following tick signals (five in total) are created by dividing the previuous tick by a factor set up in the stepDivider field of the **Core Tick Configuration** register, so `tick1` is clkDivider slower than `tick0`, `tick2` is clkDivider slower than `tick1`, and so on.

If the **Core Tick Configuration** is updated during runtime, all features relying on the core tick need to be updated accordingly. Meanwhile, inaccurate time measurement will be performed until the first tick after the reconfiguration is generated.

By default the input to the Core Tick divider is the core clock, but using the **Core Tick Select** register the input to the tick divider can be disabled, or chosen to be driven from *debug_write_data* pin 0.

Chapter 28

Multicast Broadcast Storm Control

The multicast/broadcast storm control (MBSC) unit is used to make sure that a switch does not flood the network with too much multicast/broadcast traffic. The MBSC unit prevents several traffic types from transmitting to an egress port if the corresponding traffic rate on that egress port has exceeded a certain limit.

The basic component of the MBSC unit is a token bucket (illustrated in Figure 22.1). For each egress port there is one token bucket per inspected traffic type. In principle a token bucket controls the traffic rate (packet rate or byte rate) on an egress port. A token bucket operates as follows:

1. A configurable number of tokens are periodically added to the token bucket. The bucket level will saturate at the configured capacity.
2. When a packet of the traffic type is received a configurable number of tokens are consumed, i.e. the bucket level is decreased. The number of tokens consumed per packet is either packet length plus IFG adjustment or one per packet.
3. As long as the bucket level is at or above the threshold the bucket will accept all given traffic.
4. When the bucket level drops below the threshold all packets of the inspected traffic type, destined for the corresponding egress port, are dropped. Note that instances of the same packet destined for other egress ports are not affected and have their own token buckets to check the traffic rate.
5. The **MBSC Drop** counter will be incremented once for each egress port where the packet is dropped.

In this core three kinds of traffic are checked by the MBSC unit:

- L2 Broadcast
- L2 Flooding
- L2 Multicast

For each type of traffic there is an individual control unit, consisting of one token bucket per egress port. Every token bucket can be turned on or off separately through a control register (listed in the next section).

28.1 Inspected Traffic

- L2 Broadcast: A Packet with DA = ff:ff:ff:ff:ff:ff.
  - Token bucket configurations:
    - **L2 Broadcast Storm Control Enable**
    - **L2 Broadcast Storm Control Bucket Capacity Configuration**
    - **L2 Broadcast Storm Control Bucket Threshold Configuration**
    - **L2 Broadcast Storm Control Rate Configuration**
- L2 Flooding: A packet that will be L2 switched but the DA is unknown. In this case the packet is flooded to all VLAN member ports.
  - Token bucket configurations:
    - **L2 Flooding Storm Control Enable**
    - **L2 Flooding Storm Control Bucket Capacity Configuration**

    - * **L2 Flooding Storm Control Bucket Threshold Configuration**
    - * **L2 Flooding Storm Control Rate Configuration**
- L2 Multicast: A packet that will be L2 switched and has a known multicast DA MAC in the L2 tables. (The DA MAC has Ethernet multicast bit set to 1). The core can optionally include or exclude certain packets as L2 multicast traffic. The configuration is through the **L2 Multicast Handling** register.
  - Token bucket configurations:
    - * **L2 Multicast Storm Control Enable**
    - * **L2 Multicast Storm Control Bucket Capacity Configuration**
    - * **L2 Multicast Storm Control Bucket Threshold Configuration**
    - * **L2 Multicast Storm Control Rate Configuration**

28.2 Rate Configuration

From the configuration registers a token bucket can be shaped with its capacity, threshold and token settings. The L2 broadcast storm control is here used as an example to demonstrate the operations.

From the **L2 Broadcast Storm Control Rate Configuration** register a user can configure how tokens are consumed by a packet, and how new tokens are supplemented to the bucket.

- Token consumption
  1. The token bucket can be set to count either packets or bytes by the **packetsNotBytes** field. This setting puts a token bucket in either packet or byte mode to control the maximum packet rate or byte rate on an egress port respectively.
  2. – In packet mode, every L2 broadcast packet instance to an egress port will consume one token and the bucket value will be decreased by one.
     – In byte mode, every L2 broadcast packet instance to an egress port will consume as many tokens as there are bytes in the packet plus the specified IFG correction in the **ifgCorrection** field.
- Token Injection
  1. The token injection frequency is tick [1] based. The tick timer determines the time period between token injections. The **tick** field from the **L2 Broadcast Storm Control Rate Configuration** register selects which tick timer to use.
  2. When it is time to inject new tokens, the number of tokens that will be added is configured in the **tokens** field.
- Token bucket capacity and threshold. The two configuration registers **L2 Broadcast Storm Control Bucket Capacity Configuration** and **L2 Broadcast Storm Control Bucket Threshold Configuration** are used to setup how the token bucket handles traffic bursts.

By default the MBSC unit is operating in packet mode, and all token buckets are set to allow the inspected traffic to have at most 5% of the full packet rate for 64-byte packets. Python example code to configure the maximum packet rate to 5% follows:

```
#!/usr/bin/python

rate     = 0.05

minLen   = 64 # bytes
slice    = 1 # switch slices
ifg      = 20 # bytes
pnb      = 1 # = packet mode
portBW   = 40000 # Mbits/s
tickFreqList = [1.728,
                0.1728,
                0.01728,
                0.001728,
                0.0001728] # Mhz

fullByteRate            = portBW/8.0
fullPktRate             = fullByteRate/(minLen+ifg)
```

[1] The system ticks are described in Chapter 27.

```
pktRate  = fullPktRate*rate
pktTokenIn          = 10*slice

tick = len(tickFreqList)-1
for i in range(len(tickFreqList)):
    if tickFreqList[i] * pktTokenIn <= pktRate:
       tick = i
       break

pktTokenIn = int(1.0*pktRate   / tickFreqList[tick])

pktCap  = pktTokenIn * 20
pktThr  = pktTokenIn * 10

# Field settings for the rate configuration register
settings = {
    'packetsNotBytes' : pnb,
    'tokens'          : pktTokenIn,
    'tick'            : tick,
    'ifgCorrection'   : ifg,
    'capacity'        : pktCap,
    'threshold'       : pktThr}
```

Chapter 29

Egress Resource Manager

The core includes an Egress Resource Manager (ERM) unit for controlling the shared buffer memory occupancy of egress ports and queues. The primary objective of the egress resource manager is to avoid persistent buildup of queue length in the buffer memory and prevent the blockage of enqueuing at other ports and queues. Additionally, during buffer memory congestion, ERM facilitates prioritized enqueuing of egress queues with higher priorities.

The resource management granularity is cells and there are 1024 cells, each 192 byte wide, available in the buffer memory. A packet is written to the buffer memory with the original packet data plus a 34 byte ingress to egress header, thus a 1600 byte packet will have 1634 bytes and occupy nine cells. A packet plus the ingress to egress header longer than *n* cells but shorter than *(n+1)* cells will require *(n+1)* cells for storage. For example, a 159 byte packet will use two cells. ERM traces the buffer memory occupancy and decides if a cell is allowed to be written to the buffer memory.

The ERM determines the congestion of the buffer memory based on the amount of free space (number of free cells) available. The ERM classifies the congestion levels into Green (no congestion), Yellow (slightly congested) or Red (heavily congested). When the buffer memory is in the yellow or red zone, **Resource Limiter Set** gives four sets of limits to check the queue length for different egress ports and queues. An egress port chooses limit sets for each of its queues from the **Egress Resource Manager Pointer** lookup.

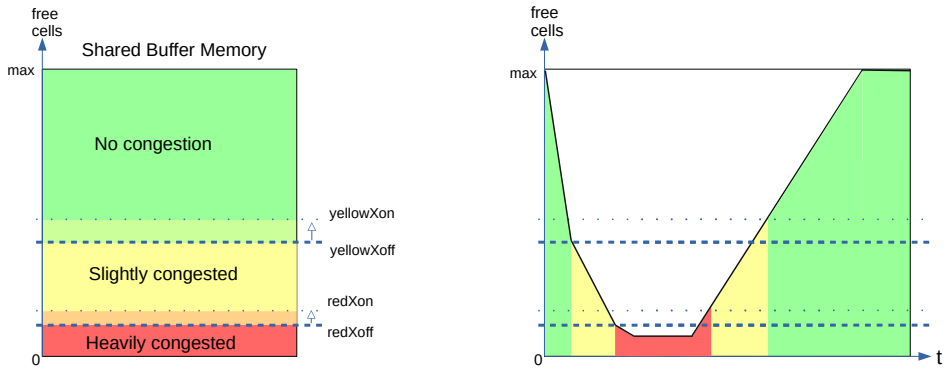


Figure 29.1: Buffer memory congestion zones

29.1 Yellow Zone

ERM Yellow Configuration defines how to enter and exit the yellow zone. The yellow zone is entered when the number of free cells goes below **yellowXoff**. To leave the yellow zone, the number of free cells need to go above **yellowXon**.

ERM checks

The buffer memory is considered partially congested when it is in the yellow zone. The ERM allows moderate buildups in all queues to a certain limit. An incoming cell of a packet is not allowed to be enqueued under two conditions:

1. The number of enqueued cells in the assigned egress queue is more than **yellowLimit**, while the total number of enqueued cells in the same queue and higher priority queues is more than **yellowAccumulated**.
2. **ERM Yellow Configuration** offers an optional check on a per egress port basis. A port can be considered as a red port in the yellow zone if the enqueued cells on that port are above **redPortXoff**. An incoming cell to a red port is not allowed if the length of the assigned queue is larger than **redLimit**.

29.2 Red Zone

ERM Red Configuration defines how to enter and exit the red zone. The red zone is entered when the number of free cells goes below **redXoff**. To leave the red zone, the number of free cells need to go above **redXon**.

ERM checks

The buffer memory is considered severely congested when it is in the red zone and the ERM shall only accept enqueuing to nearly empty queues. An incoming cell of a packet is not allowed to be enqueued in two cases:

1. The number of enqueued cells in the assigned egress queue is more than **redLimit**.
2. The ongoing packet length in cells has exceeded **redMaxCells**.

29.3 Green Zone

When the buffer memory is neither in the yellow zone nor in the red zone, the ERM considers the buffer memory to be uncongested and all incoming cells are accepted and stored in their assigned queues.

29.4 Configuration Example

A commonly used non-default ERM configuration involves allowing a queue to grow up to length **G** without packet drops (guarantees), and preventing new packets from being enqueued when the queue length is beyond **L** (limits). Between queue length **G** and **L** the enqueuing decision is made based on the overall free space in the buffer memory. This configuration imposes the following requirements:

1. **`redXon`** $\geq$ **`redXoff`** $\geq sum($**`redLimit`**$)$
   The red zone is used as guarantees, its configuration needs to ensure that **`redXon`** is large enough so that the buffer memory does not get full before all queues reach their **`redLimit`**. Set **`redLimit`** a few cells more than the desired guarantee size to have a margin for the latency.
2. Set **`yellowAccumulated`** to 0, ensuring that **`yellowLimit`** is always checked in the yellow zone.
3. **`yellowXon`** $\geq$ **`yellowXoff`** $\geq$ **`maxBufferFree`**
   Put the ERM in the yellow zone even when the buffer memory is empty hence keep **`yellowLimit`** check under an always on state.

29.5 Restrictions

Be aware that the **Map Queue to Priority** settings need to be done when there is no traffic on any port. Update with ongoing traffic may provide a wrong enqueuing snapshot to the ERM and cause inconsistencies that can not be recovered without a reset.

Chapter 30

Flow Control

The purpose of flow control is to give access to storage in the packet buffer in an fair manner between the ports sending packets to this switch. No single source port or, if configured for it, traffic class, shall be able to behave in a way that punishes other source ports (or traffic classes). For this purpose flow control has two tools at its disposition: Pausing and tail-drop.

30.1 Pausing

Pausing, or Ethernet flow control, is a method of remote controlling the far-end interface's transmissions to this switch using dedicated pause frames. Hence, for successful pause operation the far-end interface also needs to be set up properly. The remote control is done by regularly sending pause frames (by this switch's MACs) to the far-end interfaces.

The switch core will only provide the MACs with a vector of the current pause state. It is up to the MAC to detect state changes and send the appropriate pause frames. The interface for the pause state vector is described in Section 34.5.

The pause frames are entirely handled by the MAC. It both creates frames and consumes incoming frames. The switch does not expect any pause frames on the packet interface from MAC, and the switch will not create any pause frames.

The beauty of pausing is that it can be used to set up flow control without packet drops. If the size of the packet buffer is large enough to cope with the data in flight from all the far end interfaces, and they all support pausing, it is possible to configure a completely drop-less system.

If, however, some far end interfaces do not support pausing, or the amount of data in flight is too large, it is necessary to make use of tail dropping.

30.2 Tail-Drop

Tail-drop is an implicit flow-control scheme. By deliberately dropping incoming packets (tail refers to the tail of the queue) there is an induced limitation of flows by Layer 3 transport protocols with flow control (e.g. TCP). So in contrast to Pausing, Tail-drop is not reliant on features of neighboring interfaces, but on features of higher level protocols. Transport protocols without flow control (e.g. UDP) will not limit their flows due to drops, but tail-drop will still prevent those flows, when misbehaving, from interfering with traffic from other source ports (or traffic classes).

Note that for flow control to function correctly all source ports have to be set up for either pausing or tail-drop (or both). If a single source port is not configured properly, it can starve all the others of buffering resources.

30.2.1 Tail-drop as police for Pausing

Even on Pause-enabled ports it may be useful to set up tail dropping as back-up for Pausing. By setting the tail-drop threshold at a level where we would have stopped receiving data from a Pausing-enabled source port, had it observed our pause frame, we can protect our packet buffering resources even in the case that a remote interface fails to act on the pause frame.

30.3 Buffer partitioning

The packet buffer space is partitioned into reserved and free-for-all (FFA) areas. Properly configured tail-drop will never drop a packet so long as only the reserved areas are used. Below I will use "resource" to mean "source port" on a non-PFC port and "source port/traffic class" on a PFC-enabled port.

The number of FFA cells that are are allowed to be consumed by each resource before it will be hit by flow control is configured individually per resource. When the number of used free-for-all cells reaches the configured Xoff threshold, the pause state will be set to Xoff. And when the tail-drop threshold is exceeded a packet may be dropped (depending on whether there are reserves left).

The flow control decision will only be made once the last cell of a packet is about to be written to the packet buffer. Thus the thresholds need to be set so that there is space for one maximum packet per source port set aside.

30.3.1 Reserves

The tail-drop and the pausing share the reserved settings and the counters but the meaning of reserve is different between them. For tail-drop a reserve is really a reserve. Meaning that if, for instance, a source port still has reserves left it will not drop even if the global threshold is exceeded. For pausing, when an Xoff threshold is reached it will cause pausing whether or not there are reserves left. So when the global Xoff threshold is reached all ports with pausing enabled will be paused. Even those that have reserves left.

The reason that tail drop and pausing work differently is that pausing needs hysteresis between Xoff and Xon, and tail drop does not. It would be difficult to maintain the hysteresis if the reserves were observed for pausing.

Each port can be set up to work in either PFC-mode, and non-PFC-mode. In PFC-mode the accounting is done per port and traffic class, while in non-PFC-mode the accounting is only per port.

30.4 Non-PFC mode

In non-PFC mode the traffic class is disregarded, and accounting is only done per source port. The mode is controlled individually per source port by the **Port Pause Settings:mode** fields for pausing and by the **Port Tail-Drop Settings:mode** fields for tail-drop. The **Port Reserved** registers define the number of cells reserved per source port.

These counters are used in non-PFC mode:

- **FFA Used PFC**: The total number of free-for-all cells occupied by ports in PFC-mode
- **FFA Used non-PFC**: Total number of free-for-all cells occupied by ports in non-PFC-mode
- **Port Used**: Number of cells occupied by each source port

Note that the global threshold is for the sum of FFA cells, that is the sum of **FFA Used PFC** and **FFA Used non-PFC**

30.5 PFC-mode

In PFC mode accounting is additionally done per traffic class. The **Port/TC Reserved** registers define the number of cells reserved for each specific source port and traffic class combination.

Figure 30.1 illustrates the partitioning of reserved and FFA areas.

These counters are used in PFC mode:

- **FFA Used PFC**: The total number of free-for-all cells occupied by ports in PFC-mode
- **FFA Used non-PFC**: Total number of free-for-all cells occupied by ports in non-PFC-mode
- **Port FFA Used**: The number of free-for-all cells occupied for each source port
- **TC FFA Used**: The number of free-for-all cells occupied for each traffic class
- PFC Inc/Dec Counters: The cell counters per Port/TC are comprised of separate increment and decrement counters per Port/TC. The current counter value is calculated by taking the increment minus the decrement modulo the counter size.

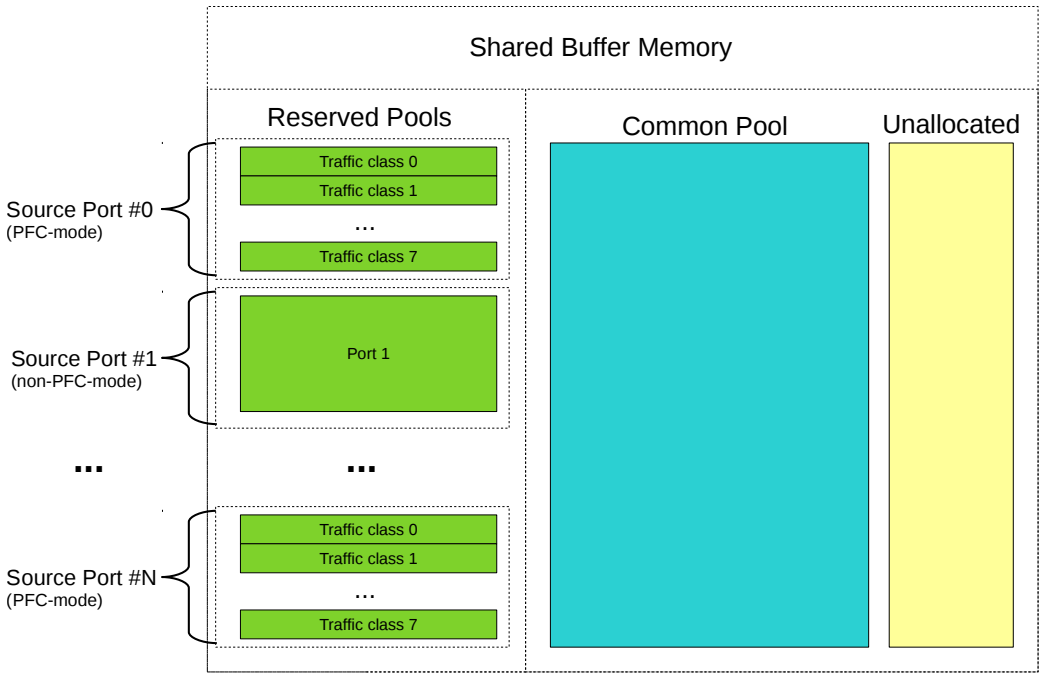


Figure 30.1: The buffer memory is partitioned into Reserved and FFA areas. The unallocated area is the space set aside for the currently incoming packets.

30.5.1 Pausing Thresholds

For tail-drop there is a single set of thresholds above which packets are dropped. For pausing there are two sets of thresholds, Xon thresholds and Xoff thresholds, thus forming a hysteresis area to avoid bursts of pause frames at the threshold. Going above the Xoff threshold will produce a pause frame turning off the packet flow at the remote interface, but to produce a pause frame turning it back on requires going all the way down below the Xon threshold.

These are the pausing thresholds:

- **Xoff FFA Threshold**: When the total number of used FFA cells is at or above this threshold the global pause state is set to paused.
- **Xon FFA Threshold**: When the total number of used FFA cells goes below this threshold the global pause state is set to un-paused.
- **TC Xoff FFA Threshold**: When the total number of used FFA cells for a traffic class is at or above this threshold the traffic class state is set to paused
- **TC Xon FFA Threshold**: When the total number of used FFA cells for a traffic class goes below below this threshold the traffic class state is set to un-paused.
- **Port Xoff FFA Threshold**: When the total number of used FFA cells for a source port is at or above this threshold the source port state will be set to paused.
- **Port Xon FFA Threshold**: When the total number of used FFA cells for a source port goes below this threshold the source port state is set to un-paused.
- **Port/TC Xoff Total Threshold**: When the sum of the FFA and Reserved cells used for a specific source port and traffic class combination is at or above this threshold, the state of this specific source port and traffic class combination will be set to paused.
- **Port/TC Xon Total Threshold**: When the sum of the FFA and Reserved cells used for a specific source port and traffic class combination goes below this threshold the state for this specific source port and traffic class combination is set to un-paused

Note that all thresholds are for the number of FFA cells used, except for the Port/TC threshold which is for the total number of cells used.

In non-PFC-mode each resource is affected by two thresholds: The source port threshold and the global threshold. Both need to be in the un-paused state for the source port to the set to un-paused.

In PFC-mode each resource (source port and traffic class) is affected by four thresholds:

- Source Port/Traffic Class
- Source Port
- Traffic Class
- Global

All four need to be in the un-paused state for the source port and traffic class combination to be set to un-paused.

30.5.2 Tail-drop Thresholds

For tail-drop there is no hysteresis so there is only a single set of thresholds:

- **Tail-Drop FFA Threshold**: When the total number of used FFA cells is above this threshold all packets will be dropped from the tail-drop-enabled ports that have no reserved cells left to spend
- **Port Tail-Drop FFA Threshold**: When the total number of used FFA cells for a source port is above this threshold incoming packets from this source port will be dropped unless the port is in PFC-mode and there are reserved cells left to spend
- **TC Tail-Drop FFA Threshold**: When the total number of used FFA cells for a traffic class is above this threshold any incoming packet belonging to the traffic class will be dropped unless the port/TC has reserved cells left to spend. Only valid in PFC-mode
- **Port/TC Tail-Drop Total Threshold**: When the sum of the FFA and Reserved cells used for a specific source port and traffic class combination is above this threshold any incoming packet from this source port assigned to this traffic class will be dropped. Only valid in PFC-mode

The **Tail-Drop FFA Threshold**, **TC Tail-Drop FFA Threshold** and **Port Tail-Drop FFA Threshold** are not obeyed strictly. The first packet exceeding the threshold may be accepted, causing a one-packet over-shoot.

30.6 Enabling Tail-Drop

Tail-drop is enabled per source port using the **Port Tail-Drop Settings:enable** fields. The individual thresholds are enabled using the enable fields in each threshold register. See Section 30.5.1 above.

30.7 Enabling Pausing

Pausing is enabled per source port using **Port Pause Settings:enable** fields. The individual thresholds are enabled using the enable fields in each threshold register. See Section 30.5.1 above.

30.8 Dropped packets

Packets that are dropped will still consume resources while they are waiting for deallocation. This applies even to broken packets, for instance packets with CRC errors.

The packets dropped due to exceeding the Tail-Drop thresholds are counted in the **Ingress Resource Manager Drop** register.

30.9 Reconfiguration

The Xon, Xoff and tail-drop thresholds can be reconfigured at any time. The reserved settings, however, cannot be changed on any source port on which there is traffic. The reserved settings also cannot be changed for any source port that has packets queued. If the reserved settings are changed in these cases the flow control counters will be irrevocably corrupted, necessitating a reset for the core to continue normal operation.

30.10 Debug Features

Each threshold can be forced to trigger using the trip fields of the threshold registers. For tail-drop only drop can be forced this way, but accept can of course be assured by disabling the threshold using the enable field.

For pausing a specific pause state can be forced using the force and pattern fields of the **Port Pause Settings** register.

Chapter 31

Egress Port Shaper

The egress port rates are shaped by token buckets configured in the **Port Shaper Rate Configuration** registers. While the token bucket level is below the threshold configured in the **Port Shaper Bucket Threshold Configuration** register, no new packets are scheduled for the corresponding egress port. Ongoing packets are not affected by the shaping bucket status.

The port shapers are enabled using the **Port Shaper Enable** register, and the saturation level of the port shaper buckets is controlled by the **Port Shaper Bucket Capacity Configuration** register.

An illustration of a token bucket can be seen in Figure 22.1 (despite what the illustration says the shaper will of course never drop any packets).

Chapter 32

Statistics

Short Name	Register Name
1. rxIf	MAC Interface Counters For RX
3. macBrokenPkt	MAC RX Broken Packets
4. macRxMin	MAC RX Short Packet Drop
4. macRxMax	MAC RX Long Packet Drop
5. spOverflow	SP Overflow Drop
11. ipppDrop	Unknown Ingress Drop Empty Mask Drop Ingress Spanning Tree Drop: Listen Ingress Spanning Tree Drop: Learning Ingress Spanning Tree Drop: Blocking L2 Lookup Drop Ingress Table Not In Sync Drop Ingress Packet Filtering Drop Reserved MAC DA Drop Reserved MAC SA Drop VLAN Member Drop Minimum Allowed VLAN Drop Maximum Allowed VLAN Drop Invalid Routing Protocol Drop Expired TTL Drop L3 Lookup Drop IP Checksum Drop Second Tunnel Exit Drop Tunnel Exit Miss Action Drop Tunnel Exit Too Small Packet Modification Drop Learning Packet Drop L2 Decoder Packet Drop L3 Decoder Packet Drop L2 Reserved Multicast Address Drop Ingress Configurable ACL Drop Egress Configurable ACL Drop ARP Decoder Drop RARP Decoder Drop L2 IEEE 1588 Decoder Drop L4 IEEE 1588 Decoder Drop IEEE 802.1X and EAPOL Decoder Drop SCTP Decoder Drop LACP Decoder Drop AH Decoder Drop ESP Decoder Drop DNS Decoder Drop BOOTP and DHCP Decoder Drop CAPWAP Decoder Drop

Short Name	Register Name
	IKE Decoder Drop GRE Decoder Drop NAT Action Table Drop Crypto Drops MACsec Drops L2 Action Table Special Packet Type Drop L2 Action Table Drop L2 Action Table Port Move Drop Source Port Default ACL Action Drop Ingress Functional Control Drops
11. smon	SMON Set 0 Packet Counter SMON Set 1 Packet Counter SMON Set 2 Packet Counter SMON Set 3 Packet Counter SMON Set 4 Packet Counter SMON Set 5 Packet Counter SMON Set 6 Packet Counter SMON Set 7 Packet Counter SMON Set 0 Byte Counter SMON Set 1 Byte Counter SMON Set 2 Byte Counter SMON Set 3 Byte Counter SMON Set 4 Byte Counter SMON Set 5 Byte Counter SMON Set 6 Byte Counter SMON Set 7 Byte Counter
11. ippAcl	Ingress Configurable ACL Match Counter
11. vrfIn	Received Packets on Ingress VRF
11. nextHop	Next Hop Hit Status
11. eppAcl	Egress Configurable ACL Match Counter
11. preEppDrop	Queue Off Drop Egress Spanning Tree Drop MBSC Drop Ingress-Egress Packet Filtering Drop L2 Action Table Per Port Drop
11. ip	IP Unicast Received Counter IP Multicast Received Counter IP Unicast Routed Counter IP Multicast Routed Counter IP Multicast ACL Drop Counter
11. ippDebug	Debug IPP Counter Debug EPP Counter
12. ipmOverflow	IPP PM Drop
13. ippTxPkt	IPP Packet Head Counter IPP Packet Tail Counter
14. eopDrop	IPP Empty Destination Drop
14. mmp	Flow Classification And Metering Drop
15. erm	Egress Resource Manager Drop
16. bmOverflow	Buffer Overflow Drop
16. irm	Ingress Resource Manager Drop
18. pbTxPkt	PB Packet Head Counter PB Packet Tail Counter
19. epppDrop	Unknown Egress Drop Egress Port Disabled Drop Egress Port Filtering Drop Egress Table Not In Sync Drop Tunnel Exit Too Small Packet Modification To Small Drop Minimum and Maximum Packet Size Drops Egress Functional Control Drops Egress Cell Size Drop

Short Name	Register Name
19. vrfOut	**Transmitted Packets on Egress VRF**
19. nat	**Ingress NAT Hit Status** **Egress NAT Hit Status**
21. drain	**Drain Port Drop**
22. epmOverflow	**EPP PM Drop**
24. rqOverflow	**Re-queue Overflow Drop**
24. eppTxPkt	**EPP Packet Head Counter** **EPP Packet Tail Counter**
25. psTxPkt	**PS Packet Head Counter** **PS Packet Tail Counter**
25. psError	**PS Error Counter**
28. txIf	**MAC Interface Counters For TX**

Table 32.1: Sequence of Statistics Counters

This core supports full statistics with 32-bit wrap around counters. The statistics is divided into groups depending on the type of statistics and location in the switch. Figure 32.1 gives the location of the counters from ingress to egress, with a sequence number to show their process orders. The counters which are green are for packet drops based on forwarding decisions while the red counters are related to system errors. The details of the counters in Figure 32.1 can be found through Table 32.1.

32.1 Packet Processing Pipeline Drops

During the ingress/egress packet processing, the forwarding algorithm can drop a packet for various reasons. For each type of drop reason at least one drop counter is attached. The counter update is either based on received packets or to-be-transmitted packets.

- Statistics: IPP Ingress Port Drop.
  Each drop reason has a unique drop identifier (drop ID). The IPP ingress port drop statistics has a counter for each drop ID. In two cases a corresponding drop ID counter can be updated:
  1. When a received packet is dropped before any destination port is assigned.
  2. When all targeting destination ports are filtered out the **Empty Mask Drop** counter is updated.
- Statistics: IPP Egress Port Drop.
  This is a per drop ID and per egress port counter located in the ingress processing pipeline. When a packet has obtained one or more destination ports but the following ingress packet process filters out one of the obtained destination ports, a counter is updated for the corresponding egress port with the related drop ID. The **Empty Mask Drop** counter might be updated at the same time if no more destination port is set after the filtering.

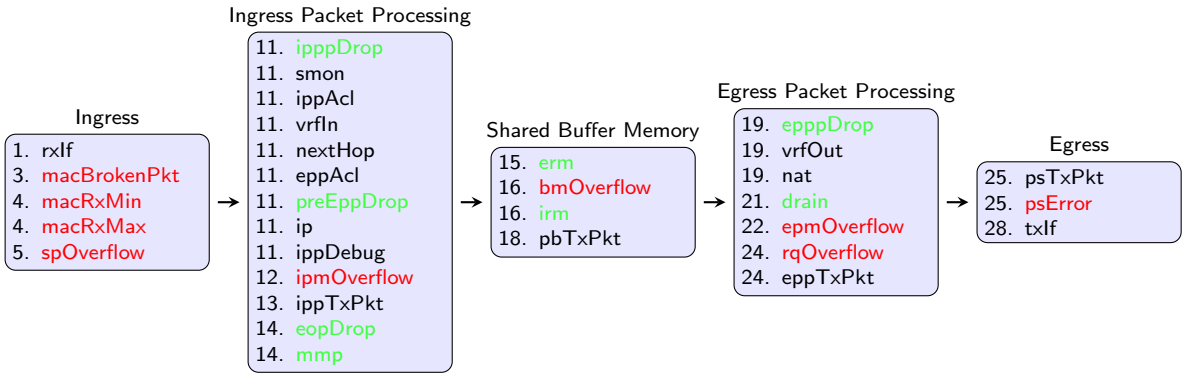


Figure 32.1: Location of Statistics Counters

- Statistics: EPP Egress Port Drop.
  This is similar to IPP egress port drop statistics but located in the egress packet processing pipeline. Drops that occur in EPP will cause bubbles on the transmit interface.

32.2 ACL Statistics

When a packet matches an ACL rule as described in Chapter Classification, the result operation can be configured to update a counter. In this case the result operation has a pointer to which counter to update. All the related counters are in Section Statistics: ACL.

32.3 SMON Statistics

There are 8 sets of SMON counters located in the ingress packet processing pipeline, each equipped with one counter per PCP value. The combination of the ingress port number and packet VLAN ID will provide the target SMON set to update through the **SMON Set Search** register. Each SMON set counts both the number of packets and number of bytes as shown in Section Statistics: SMON.

32.4 Routing Statistics

Section Statistics: Routing has three routing related statistics:

- **Received Packets on Ingress VRF**. Update when a packet enters a VRF in the ingress processing pipeline.
- **Transmitted Packets on Egress VRF**. Update when a packet leaves a VRF in the egress processing pipeline.
- **Next Hop Hit Status**. Update when IPv4/IPv6/MPLS packets hit a next hop entry.

32.5 Ingress Port Receive Statistics

Section Statistics: IPP Ingress Port Receive lists available statistics for good received packets on a per ingress port basis.

- Good received IP packets
  - **IP Unicast Received Counter**
  - **IP Unicast Routed Counter**
  - **IP Multicast Received Counter**
  - **IP Multicast Routed Counter**
  - **IP Multicast ACL Drop Counter**

32.6 Packet Datapath Statistics

Section Statistics: Packet Datapath gives a list of start of packet and end of packet counters in the main blocks of the core. They act as datapath checkpoints and can be helpful in tracing unexpected packet drops or corruptions.

A packet will cross three clock domains on its way through the core:

- RX MAC clock domain.
  Packet datapath statistics in the RX MAC clock domain are on the receive edge of the switch, counting received packets as well as illegal packet patterns. Clock crossing synchronizations are applied to these counters in order to share the same configuration bus in the core clock domain. The included counters are:
  1. **MAC Interface Counters For RX**.
- TX MAC clock domain.
  Packet datapath statistics in the TX MAC clock domain are on the transmit edge of the switch, counting transmitted packets as well as protocol errors on the TX interface of the switch. Clock crossing synchronizations are applied to these counters in order to share the same configuration bus in the core clock domain.
  1. **MAC Interface Counters For TX**.

- Core clock domain.

  Packet datapath statistics in the core clock domain are counting in different internal blocks. Each block has a pair of counters for packet heads and tails to identify the pass through of a complete packet. The datapath counting follows the order in Figure 1.1:

  1. **IPP Packet Head Counter** and **IPP Packet Tail Counter**.
  2. **PB Packet Head Counter** and **PB Packet Tail Counter**.
  3. **EPP Packet Head Counter** and **EPP Packet Tail Counter**.
  4. **PS Packet Head Counter** and **PS Packet Tail Counter**.

  If a stage has unequal packet head and tail counters while the counters in the previous stages are identical, packets are corrupted in this stage.

32.7 Miscellaneous Statistics

The core is designed to have no silent packet drops and all missing packets on the transmit interface can be found in a dedicated drop counter. Besides the drop counters mentioned above, there are more counters located in all other places where a packet drop might occur. Detailed drop counter list is in Section Statistics: Misc.

32.8 Debug Statistics

Section Statistics: Debug lists a group of statistics prepared for debug purposes. These counters indicate possible locations when fatal errors occurred inside the core. Typical error events include inaccurate clock frequencies, unacceptable configurations, etc. The switch will try to remain functional after an error state, but a correct behaviour cannot be guaranteed.

32.8.1 Debug Statistics Accuracy

Some of the statistics counters are located in a different clock domain than the configuration bus. The values are therefore transferred through synchronization registers. In order to reduce the hardware cost of these debug counters the synchronization can result in reading incorrect values if readout is done while the counters are incrementing. The counter itself will always have the correct value. It's only the readout that, with a very low probability, can have incorrect value on bits that are toggling.

Chapter 33

Packets To And From The CPU

The CPU port (number 10) has support for two special CPU tags in the packet header. In packets received by the switch on the CPU port, the tag can determine which port the packet shall be sent to. A tag can also be added to packets transmitted by the switch on the CPU port. This allows the software stack to determine where the packet came from and the reason why it was sent to the CPU port.

33.1 Packets From the CPU

Packets sent from the CPU are normally processed as any other packet that enters the switch, so the destination port is determined by the L2 lookup. When the CPU needs to direct a packet to a specific port, bypassing the normal L2 lookup, it is accomplished by adding a protocol header.

Byte Number	**Contents of Byte**
0-1	[10:0] port bit mask. Bit 0 is port number 0, bit 1 is port number 1 etc. Port 0 is located in bit 0 of byte number 1. The port numbers are physical ports, not link aggregation port numbers. The link aggregation will always be bypassed when sending packets with a From CPU Tag.
2	Bits [2:0] specifies which egress queue the packet shall use. Bit [3] Specifies if the packet shall go out un-modified or modified on the egress ports. If this bit is set to one all ACL actions are bypassed. 0 = Modified. 1 = Unmodified.
3	Bit [0] will set the *upd_ts* signal on the transmit MAC interface when the packet is transmitted. Bit [1] will set the *upd_cf* signal on the transmit MAC interface when the packet is transmitted. Bit [2] will set the *ts_to_sw* signal on the transmit MAC interface when the packet is transmitted.
4-11	PTP Timestamp that will be set on the transmit MAC interface when the packet is transmitted. The lowest numbered byte contains the msb of the timestamp value.
14-16	Reserved. Not used.

Table 33.1: From CPU tag format

The header consists of a specific Ethernet Type (39065) followed by a CPU Tag. The CPU tag has a 2 byte(s)

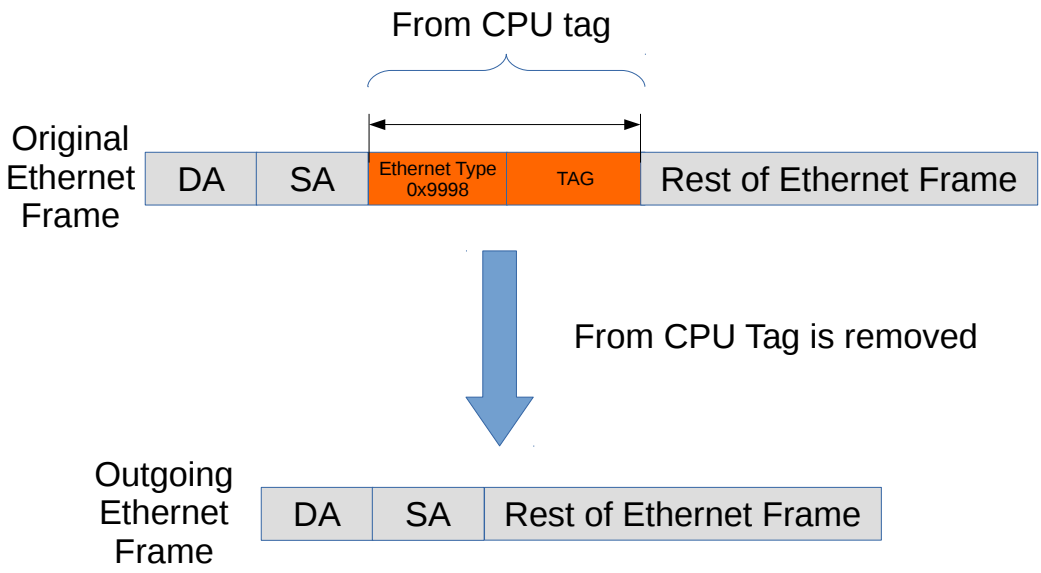


Figure 33.1: Packet from CPU with CPU tag

destination port mask field[1] and 1 byte egress queue field (encoded as specified in table 33.1). The switch core will remove the extra protocol header and send out the packet on the ports requested by the destination port mask in the protocol header. This is shown in the figure 33.1.

The port mask in the CPU Tag field determines which ports the packet shall be sent to. If multiple bits are set in the port mask, the packet is treated as a multicast packet in the resource limiters. The packet will be sent out on all ports with the corresponding bit set.

33.1.1 Identify the From CPU Tag

By default, only packets that are received on the CPU port will be able to support identifying the specific Ethernet type for the from CPU tag. This means that packets with this Ethernet type that are received on other ports of the switch will be treated as unknown and will not enter the packet processing based on the from CPU tag.

If non-CPU ports need to identify the from CPU tag, it can be achieved by the **enableFromCpuTag** from the **Source Port Table**. Notice the CPU port is not affected by this setting and always decode the from CPU tag.

33.1.2 From CPU Header and Packet Modification and Operations

There are a number of operations which are not carried out when a packet is sent in with the From CPU header. The following lists details this in greater detail what is done and what is not done.

- Link Aggregation is done.
- None of the VLAN operations are carried out.
- Mirroring is done. However with regards to ACL mirroring see below.
- Drops are ignored, example VLAN table , spanning tree / multiple spanning tree drops.
- L2 Lookup result is ignored.
- If the packet hits decoding rules for BPDU, Rapid Stanning Tree, Multiple Spanning tree, or other protocols such as 802.1X-EAPOL AH ARP AVTP DHCP CAPWAP DNS ESP GRE IKE L2 1588 L4 1588 LACP RARP SCTP then the packet will still send a extra copy to the CPU port. This can be disabled by setting the cpu port to zero in the send-to-cpu bitmask in each function.
- Routing is not carried out.
- SMON statistics is performed.
- Basic assignment of MMP is done.

[1] The ordering described in 33.1 is the receive/transmit order.

- Meter-Marker-Policer check is done.
- MBSC is bypassed.
- All spanning tree and multiple spanning treeperations are bypassed.
- No learning operation.
- If the From CPU tag has the Modified bit set to one (1) then the following happens:
- Check Reserved DMAC is bypassed.
- Check Reserved SMAC is bypassed.
- ACL operations are not done.
- ACL statistics are not done.
- Tunneling are not done (tunnel entry or tunnel exit).
- SMON statistics is not done.
- NAT operations are not done.
- If the From CPU tag has the Modified bit set to zero (0) then the following happens:
- Check Reserved DMAC is done.
- Check Reserved SMAC is done.
- ACL operations are done.
- ACL statistics are done.
- Tunneling is done (tunnel entry or tunnel exit).
- SMON statistics is done.
- NAT operations are done.

33.2 Packets To the CPU

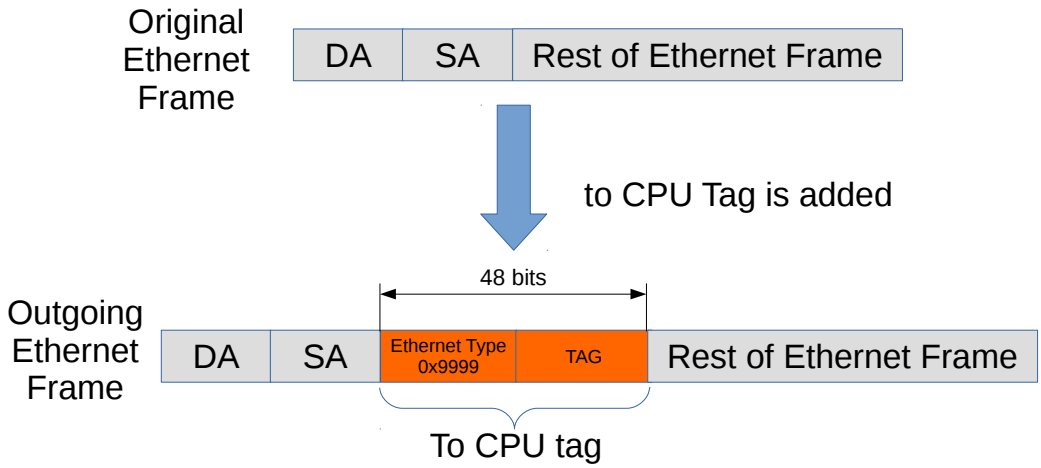


Figure 33.2: Packet to CPU with CPU tag

Packets can also be sent to the CPU port bypassing the normal L2 lookup. By default all packets to the CPU port have an extra protocol header (as shown in Figure 33.2). The header indicates the reason that the packet was sent to the CPU, and the port on which it was received. If the packets shall be the original copy as it came in on a source port or if they shall be the processed version depends first on a register called **Default Packet To CPU Modification**, at some places there also exists extra bits to change this setting.

When packets are sent to the CPU port (number 10 in this core), the packets are tagged with a specific Ethernet Type (type 39321). Figure 33.2 shows the Ethernet type field followed by a tag, and together these constitute the extra protocol header mentioned above. The unmodified incoming packet follows just after this header.

The insertion of the extra protocol header can be disabled by setting the register **Disable CPU tag on CPU Port** to 1.

33.3 To CPU Header format

The following table describes the fields which will be in the toCPU tag. The original bit is set when packets are modified by the egress packet processing, if the modification is the same as the original packet this modification bit will still be set.

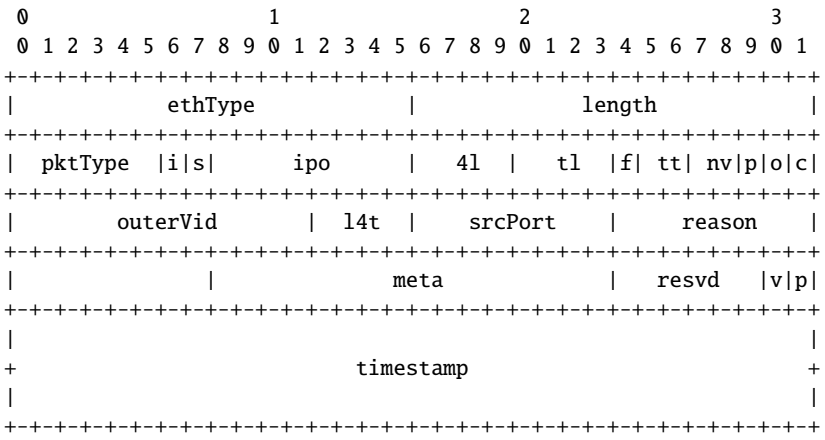
Name	Short Name	Field Size	IETF bit index	Description
Ethernet Type	ethType	16	[15:0]	Ethernet Type, 0x9999
Length	length	16	[31:16]	Length of Packet
Packet Type	pktType	6	[37:32]	Packet Type, see table 33.3
IPv4	i	1	[38]	This is a IPv4 Packet. 0 = No 1 = Yes
IPv6	s	1	[39]	This is a IPv6 Packet. 0 = No 1 = Yes
IP Offset	ipo	8	[47:40]	IP Header Offset.
IPv4 length	4l	4	[51:48]	IPv4 Header Length in 4 Octets.
TCP length	tl	4	[55:52]	TCP Header Length in 4 Octets. NOTE: If the packet is a IPv6 and has a segment routing header then this value will be set to zero.
Fragment	f	1	[56]	Fragment Indicator from IPv4 header.
Transmit Type	tt	2	[58:57]	The transmitt type. If the packet first passes through the crypto engine before it is sent out then this will show a value of 0. 0 = Unicast 1 = Multicast 2 = BroadCast 3 = Flooding
Nr Of Vlans	nv	2	[60:59]	The nr of VLANs. 0 = Zero 1 = One 2 = Two 3 = More than two
is_PPPoE	p	1	[61]	PPPoE Header exists in packet.
original	o	1	[62]	Original or modified packet. 0 = Original 1 = Modified
fromCrypto	c	1	[63]	This packet came from crypto engine. 0 = No 1 = Yes
Outermost VID	outerVid	12	[75:64]	The outermost VLAN ID on the packet
L4 Type	l4t	4	[79:76]	The L4 Type of the packet. 0 = Not known. 1 = Is IPv4 or IPv6 but type is not any L4 type in this list. 2 = UDP 3 = TCP 4 = IGMP 5 = ICMP 6 = ICMPv6 7 = MLD 8..15 - Reserved.
Source Port	srcPort	8	[87:80]	Source Port, bits 3:0 Contains the source port number
Reason	reason	16	[103:88]	Reason Code, Byte 1 is the msb of the reason code. see table 33.4
Meta Data	meta	16	[119:104]	The meta data comes from the forwarding tables. It is setup by software to enable software to determine the reason why a entry was sent to the CPU port.

Name	Short Name	Field Size	IETF bit index	Description
Reserved	resvd	6	[125:120]	Reserved.
Valid Timestamp	v	1	[126]	If set to one then the Timestamp field is valid.
PTP	p	1	[127]	If set to one then the packet is a PTP packet.
Timestamp	timestamp	64	[191:128]	64-bit Timestamp of packet

Table 33.2: To CPU Header

33.3.1 To CPU Header in IETF format

The packets-to-CPU header expressed in a IETF header format below, using the short names from the 33.2 as field names. Bit 0, Byte0 is the first bit and byte which is being transferred out on the CPU port (see appendix B in RFC791):



33.3.2 Packet Type Table

As seen above there is a packet type field and this is determined by the packet decoder information to tell the receiving CPU what type of packet it is.

Packet Type Id	Name	Description
0	ARP	The packet is a ARP packet. Decoding setup and options are avalible in register **ARP Packet Decoder Options**
1	RARP	The packet is a RARP packet. Decoding setup and options are avalible in register **RARP Packet Decoder Options**
2	LLDP	The packet is a LLDP packet. Decoding setup and options are avalible in register **LLDP Configuration**
3	L2_1588	The packet is a IEEE 1588 L2 packet. Decoding setup and options are avalible in register **IEEE 1588 L2 Packet Decoder Options**
4	8021X_EAPOL	The packet is a 802.1X or EAPOL packet. Decoding setup and options are avalible in register **IEEE 802.1X and EAPOL Packet Decoder Options**
5	MPLS	The packet is a MPLS packet.
6	GRE	The packet is a GRE packet. Decoding setup and options are avalible in register **GRE Packet Decoder Options**
7	SCTP	The packet is a SCTP packet. Decoding setup and options are avalible in register **SCTP Packet Decoder Options**
8	IGMP	This is a IGMP packet.
9	MLD	This is a MLD packet.
10	ICMP	This is a ICMP packet.
11	LACP	The packet is a LACP packet. Decoding setup and options are avalible in register **LACP Packet Decoder Options**
12	AH	The packet is a IPsec AH packet. Decoding setup and options are avalible in register **AH Header Packet Decoder Options**
13	ESP	The packet is a IPsec ESP packet. Decoding setup and options are avalible in register **ESP Header Packet Decoder Options**
14	DNS	The packet is a DNS packet. Decoding setup and options are avalible in register **DNS Packet Decoder Options**
15	BOOTP_DHCP	The packet is a BOOTP or DHCP packet. Decoding setup and options are avalible in register **BOOTP and DHCP Packet Decoder Options**
16	L4_1588	The packet is a IEEE 1588 L4 packet. Decoding setup and options are avalible in register **IEEE 1588 L4 Packet Decoder Options**
17	CAPWAP	The packet is a CAPWAP packet. Decoding setup and options are avalible in register **CAPWAP Packet Decoder Options**
18	IKE	The packet is a IPsec IKE packet. Decoding setup and options are avalible in register **IKE Packet Decoder Options**
19	BPDU	The packet is a BPDU packet.
20	UDP_LARGER_THAN_1024	The packet is a UDP packet where destination port ¿ 1024.
21	TCP_LARGER_THAN_1024	The packet is a TCP packet where destination port ¿ 1024.
22	CANCEL_TE	A tunnel exit was performed but then the original packet shall be sent to the CPU. Hence the inner packet type information is lost. CPU needs to determine packet type by itselves.
63	default	When all above identifications fails.

Table 33.3: Packet Type Table

33.3.3 Reason Table

The reason codes why a packet was sent to the CPU. Reason code 0 means that the packet was switches or routed and the CPU port was part of the normal forwardings destination ports.If a packet can be directed to the CPU port with multiple reasons, the first hit in the check list below will give the reason code to the egress packet header.

Reason	Description
0	The MAC table, L2 MC table, ACL send to port action, MPLS table, the from-CPU-TAG contained the CPU port or routing tables sent the packet to the CPU port.
1	The packet decoder requires more than one cell.
2	This is a BPDU / RSTP frame.
3	The Unique MAC address to the CPU was hit.
4 + HitIndex	The Source MAC range sent the packet to the CPU..Index to rule.
8 + HitIndex	The Destination MAC range sent the packet to the CPU..Index to rule.
12 + HitIndex	The source port default ACL action sent the packet to the CPU..Index to source port which sent the packet in.
23 + HitIndex	The TCAM in the configurable ingress ACL engine 0 sent the packet to the CPU..Index to rule.
39 + HitIndex	The small table in the configurable ingress ACL engine 0 sent the packet to the CPU..Index to rule.
1063 + HitIndex	The large table in the configurable ingress ACL engine 0 sent the packet to the CPU..Index to rule.
9255 + HitIndex	The TCAM in the configurable ingress ACL engine 1 sent the packet to the CPU..Index to rule.
9263 + HitIndex	The small table in the configurable ingress ACL engine 1 sent the packet to the CPU..Index to rule.
9775 + HitIndex	The large table in the configurable ingress ACL engine 1 sent the packet to the CPU..Index to rule.
10031 + HitIndex	The TCAM in the configurable ingress ACL engine 2 sent the packet to the CPU..Index to rule.
10047 + HitIndex	The small table in the configurable ingress ACL engine 2 sent the packet to the CPU..Index to rule.
10079 + HitIndex	The large table in the configurable ingress ACL engine 2 sent the packet to the CPU..Index to rule.
10207 + HitIndex	The TCAM in the configurable egress ACL engine sent the packet to the CPU..Index to rule.
10223 + HitIndex	The small table in the configurable egress ACL engine sent the packet to the CPU..Index to rule.
11247 + HitIndex	The large table in the configurable egress ACL engine sent the packet to the CPU..Index to rule.
19439	This is an L2 1588 frame.
19440	This is an L4 1588 frame.
19441	This is an ARP frame.
19442	This is an RARP frame.
19443	This is an LLDP frame.
19444	This is an 802.1X EAPOL frame.
19445	This is an GRE frame.
19446	This is an SCTP frame.
19447	This is an LCAP frame.
19448	This is an AH frame.
19449	This is an ESP frame.
19450	This is an DNS frame.
19451	This is a BOOTP or DHCP frame.
19452	This is an CAPWAP frame.
19453	This is an IKE frame.
19454	The IP TTL field was expired in the packet.
19455	The router ports check about which IPv4/IPv6/MPLS packets was allowed in the router failed.

Reason	Description
19456	The default routes send2cpu bit was set.
19457	The IP length exceeded the MTU setup.
19458	The entry in the **Next Hop Table** is invalid.
19459	The entry in **Next Hop Packet Modifications** pointed to from the **Next Hop Table** is invalid.
19460	The next hop entry had a send2cpu bit set.
19461	The IPv4 header size field was not equal to five.
19462	IPv4/IPv6 multicast was detected and redirected to CPU.
19463	The IPv6 routing header contained an unrecognized routing type
19464	The IPv6 segment routing header contained an unexpected routing header length
19465	The IPv6 segment routing header contained TLV field
19466	The pointer to the Security Association was not valid
19467	A packet offered for processing appears to be an IP fragment, i.e., the OFFSET field is non-zero or the MORE FRAGMENTS flag is set.
19468	Attempt to transmit a packet that would result in Sequence Number overflow.
19469	The received packet fails the anti-replay checks.
19470	The integrity check fails.
19471	The maxium number of MPLS tags was detected in a packet.
19472	Packet matched an L2 Multicast Reserved Address
19473	Packet was suppose to do a two tunnel exits.
19474	The first tunnel exit lookup was a hit but the second tunnel exit lookup was a miss and the source port table said this packet shall then be sent to the CPU.
19475	Tunnel Exit result said send to CPU.
19476	After Tunnel entry the MTU was too small for this packet.
19477	The **NAT Action Table** has sent the packet to the CPU with this code.
19478	The **NAT Action Table** has sent the packet to the CPU wit this code.
19479	The **L2 Action Table** has determined that this packet shall be sent to the CPU.
19480	The **SNAP LLC Decoding Options** has determined that this packet shall be sent to the CPU.

Table 33.4: Reason for packet sent to CPU

The possible reasons are listed in Table 33.4.

1. Hit in the **Reserved Source MAC Address Range** with a **sendToCpu** action.
2. Hit in the **Reserved Destination MAC Address Range** with a **sendToCpu** action.
3. Hit in the **L2 Reserved Multicast Address Base** with **sendToCpuMask** enabled for the corresponding source port.
4. Hit in the **LLDP Configuration**.
5. Hit in the **Send to CPU** register.
   - Notice that when **uniqueCpuMac** is enabled then unicast packet will not be switched to the CPU port. Instead packets from any source port with MAC DA equal to **cpuMacAddr** will be sent to the CPU. Other mechanism for sending to the CPU port are not affected (e.g. ACL's).
6. Hit in the **Configurable ACL Engine** with a sendToCpu action.

33.3.4 Reason Code Operations

If the packet is sent to the CPU port with a non-zero reason code, the **CPU Reason Code Operation** register allows extra actions based on the corresponding reason code. The reason code number is checked in 16 given ranges from the first entry to the last entry. If the reason code has multiple hits, different operations can be done in parallel and the same operation in the latter one will override the previous hit.

- **mutableCpu** allows the packets that are sent to the CPU port use another port number for the CPU port. In this case the to CPU tag is always inserted to the packet and will not be controlled by **Disable CPU tag on CPU Port**.
- **forceQueue** alters the egress queue of the packets that are sent to the CPU port.

Chapter 34

Core Interface Description

This chapter describes the interfaces to the core. An *input* is an input to the core, and an *output* is a signal driven by the core. In analogy *reception* refers to packets to the core and *transmission* means packets from the core.

34.1 Clock, Reset and Initialization interface

There is a core clock, mac clock signals for the packet interfaces, a global reset signal, mac reset signals for the packet interfaces, and a *doing_init* output (indicating when the core is in initialization and thus not ready to receive packets).

When the global reset, *rstn*, is asserted all packets buffered in the switch will be dropped, the learning and aging engines and all statistics counters will be reset to the initial status. Reset can be pulled at any time, but any ongoing transmit packets will be immidiately interrupted and no end of packet signal will be given.

The packet interface resets cannot be used independently. If one reset is asserted, all resets (including the core reset) have to be asserted before any reset can be released.[1]

[1] Thus the packet interface resets cannot be used to empty a specific packet interface. To do that, follow the procedure in Section 23.8, while making sure that the packet interface halt is kept low.

Signal Name	Size	In Out	Description
clk	1	In	Core clock. For 170 Gbit/s wire-speed throughput use a core clock frequency of 270 MHz
rstn	1	In	Global asynchronous reset (active low)
clk_mac_rx_N	1	In	Clock for the RX packet interface for port **N**.
rstn_mac_rx_N	1	In	Asynchronous reset (active low) for the RX packet interface for port **N**
clk_mac_tx_N	1	In	Clock for the TX packet interface for port **N**.
rstn_mac_tx_N	1	In	Asynchronous reset (active low) for the TX packet interface for port **N**
clk_int_0	1	In	Clock for the internal crypto engine. Should be 312.5MHz for full performance.
rstn_int_0	1	In	Asynchronous reset (active low) for the internal crypto engine.
assert_reset	1	Out	Signal indicating that the core has experienced an unrecoverable error, and should be reset.
consistency_check	1	In	When pulled high internal checks will be made. This is a simulation-only port, it shall be tied low in hardware.
idle	1	Out	Indicates when the packet processing pipelines are empty.
doing_init	1	Out	Indicates that the core is in initialization. The operation of the core is undefined if packets are injected on the rx-interfaces when the core is in initialization

Table 34.1: Clock and Reset interfaces

Core Initialization

Before packets are sent to the core it needs to be initialized. The initialization is initiated when reset is released. Reset activation is asynchronous to any clock. The reset should be kept low at least one cycle of the slowest clock. Releasing reset must be done synchronously with respect to all clocks. During initialization *doing_init* is kept high. See Figure 34.1. The length of the initialization is dependent on the depth of the deepest initialized memory.

During initialization no activity is expected on the configuration interface or on the packet RX interfaces, and the operation of the core is undefined if any such activity occurs.

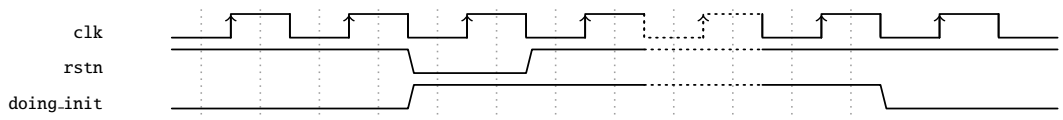


Figure 34.1: Core Initialization

34.1.1 Assert Reset

The *assert_reset* signal will go high, and stay high, if the core experiences an unrecoverable error. The behaviour of the core when *assert_reset* is high is undefined, and the only way to get back to normal operation is to reset the core.

The configuration bus will most likely still work when *assert_reset* is high, but to figure out what went wrong you will probably need to use the debug interface.

34.2 Packet Interface

There are 11 packet interfaces, or ports for short, each divided into a reception part and a transmission part. The ports are numbered from 0 to 10.

Pin	Size	Direction	Description
idata_sp_**N**	128	In	Packet data.
ivalid_bytes_sp_**N**	5	In	Indicates the number of valid data bytes. For all transactions where *last* is not high, this shall be equal to the data width in bytes.
ifirst_sp_**N**	1	In	Start-of-packet flag.
ilast_sp_**N**	1	In	End-of-packet flag. The *last* field is also used to signal broken packets. For a correctly transmitted packet *last* is asserted for the last data transaction of the packet. If *last* is set high when *valid_bytes* is zero, the packet is marked as broken, and will be dropped by the core.
ivalid_ts_sp_**N**	1	In	Validates the presence of the timestamp value. Valid when *last* is set.
its_sp_**N**	64	In	PTP Timestamp value. Only available when *last* is set.

Table 34.2: Packet RX interface for ports 0 and 1. **N** is the ingress interface number.

The port interfaces are not all the same. There are two different port interface variants in this core, each with an RX and a TX direction:

1. Ports 0 and 1: RX-interface see Table 34.2 on page 199, TX-interface see Table 34.3 on page 200
2. Ports 2-10: RX-interface see Table 34.4 on page 201, TX-interface see Table 34.5 on page 202

Each direction of a packet interface consists of *first*, *last*, *valid_bytes*, and *data* fields. The transmit direction has an additional *halt* signal to allow the receiving end to moderate the data rate transmitted from the core, and a *pkt_length* field giving the the packet length in bytes. The *pkt_length* field is only valid for the first cell of a packet. There is also a *ptp* flag on the transmit side that is only valid for the first cell. It is copied from the PTP bit in the CPU header, and is thus always zero for all ports except the CPU port.

Packet data is presented in order, i.e. the most recent byte is the, so far, highest numbered byte in the packet. The first valid byte on the bus is byte 0, and all bytes are valid up to the number indicated in *valid_bytes*. Unless the *last* flag is set all bytes or no bytes must be valid.

Sending and Receiving packets

Data transmission, either to or from the core, begins with a transaction where the *first* field is high and the *valid_bytes* field is non-zero, and ends with a data transmission where the *last* field is high. Idle transactions—where valid_bytes, *first* and *last* are all zero—are allowed at any time, but unless halted there will be no idle transactions on the transmission interfaces other than between packets.

By default, the core has a short packet size limit of 60 bytes. All shorter packets will be dropped. This assumes that the receiving MAC removes the FCS before sending the packet to the core.

Jumbo packets

The maximum packet length that this core can cope with is 32733 bytes. If this length was allowed to be exceeded either on the ingress or the egress it would corrupt the internal counters.

It should be noted that it is not guaranteed that a packet of that length will always be able to pass through the switch, even if the destination queue is not congested. Depending on the Egress Resource Management settings, and/or the congestion status of other ports, there may not be enough free cells in the packet buffer to store such

Pin	Size	Direction	Description
odata_ps_**N**	128	Out	Packet data.
ovalid_bytes_ps_**N**	5	Out	Indicates the number of valid data bytes. For all transactions where *last* is not high, this is equal to the data width in bytes.
ofirst_ps_**N**	1	Out	Start-of-packet flag.
olast_ps_**N**	1	Out	End-of-packet flag. For a correctly transmitted packet *last* is asserted for the last data transaction of the packet. If *last* is set high when *valid_bytes* is zero, the packet shall be dropped or terminated with an error by the MAC.
oupd_ts_ps_**N**	1	Out	The TX MAC should update the PTP Timestamp field in the current packet. Only valid when *first* is set.
oupd_cf_ps_**N**	1	Out	The TX MAC should update the PTP correction field in the current packet. Only valid when *first* is set.
ots_to_sw_ps_**N**	1	Out	The TX MAC should take a timestamp of the current packet and send to software.
ots_ps_**N**	64	Out	PTP Timestamp value. Only valid when *first* is set.
oudp4_ps_**N**	1	Out	The packet is an IPv4/UDP packet. Only valid when *first* is set.
oudp6_ps_**N**	1	Out	The packet is an IPv6/UDP packet.
oudp_csum_ps_**N**	9	Out	Byte position of the start of the UDP checksum field. Only valid when *first* is set.
ots_pos_ps_**N**	9	Out	Byte position of the start of the Timestamp field in a PTP packet. Only valid when *first* is set.
oudp_corr_ps_**N**	15	Out	Byte position of the start of the UDP checksum correction position in a PTP packet. Only valid when *first* is set.
opkt_length_ps_**N**	15	Out	The packet length in bytes. Only valid when *first* is set.
tx_halt_ps_**N**	1	In	Interrupt the data transmission from egress port **N**.

Table 34.3: Packet TX interface for ports 0 and 1. **N** is the egress interface number.

a large packet. But the switch core will, when properly configured and reasonably uncongested, be able to switch 32733-byte packets.

Longest Packet for No-Overlap Mesh

The longest packet that can pass a no-overlap mesh test is highly dependent on the ERM settings. But with the default settings you can expect to pass a no-overlap mesh test with 7462-byte packets.

Inter-frame gap

For small packets it is possible to feed the switch with more packets than it can handle. This will cause the SP to overflow, and packets to be dropped. To avoid packet drops an inter-frame gap (IFG) of at least 192 bits is needed between each packet. There is a small fifo in the SP, so a single smaller IFG is fine, but it needs to average at or above the minimum IFG over a window of a few packets.

On the output from the switch packets will be sent back to back, without IFG, and it is up to the receiver to halt the transmission using the *halt* interface to prevent overflows.

Pin	Size	Direction	Description
idata_sp_**N**	32	In	Packet data.
ivalid_bytes_sp_**N**	3	In	Indicates the number of valid data bytes. For all transactions where *last* is not high, this shall be equal to the data width in bytes.
ifirst_sp_**N**	1	In	Start-of-packet flag.
ilast_sp_**N**	1	In	End-of-packet flag. The *last* field is also used to signal broken packets. For a correctly transmitted packet *last* is asserted for the last data transaction of the packet. If *last* is set high when *valid_bytes* is zero, the packet is marked as broken, and will be dropped by the core.
ivalid_ts_sp_**N**	1	In	Validates the presence of the timestamp value. Valid when *last* is set.
its_sp_**N**	64	In	PTP Timestamp value. Only available when *last* is set.

Table 34.4: Packet RX interface for ports 2-10. **N** is the ingress interface number.

Broken packets

A packet ending with *last* set high and *valid_bytes* set to zero is considered a broken packet. Broken packets received by the core will never be output on the egress ports, but will be dropped at the earliest convenience. So any broken packets output from the switch are packet that were somehow corrupted in the core. There are no benign cases where this happens. Depending on the packet length a broken packet input to the core will be dropped either before or after ingress packet processing. Broken packets larger than a cell will pass through the packet processing pipeline and then been dropped, while packets shorter than a cell will be filtered out before the packet processing pipeline.

All broken packets are counted in the **MAC RX Broken Packets**.

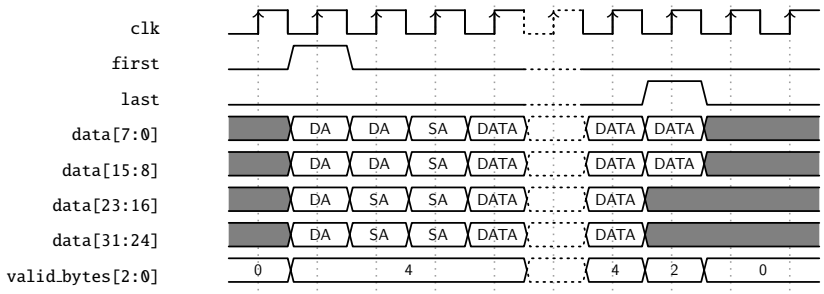


Figure 34.2: Sending and Receiving packets without error (32-bit)

Halts

Data transmission from the transmit interface of the core can be interrupted individually per egress port using the *halt* signals. A high halt signal on the positive edge of mac clock, will cause the transmission to be idle for the corresponding egress port on the same positive edge. Data transmission will resume on the next positive edge of mac clock when halt is again low.

Byte Order

We define the packet byte order by the first transmitted/received byte on the wire labeled byte 0, as in IEEE 802.3. On a packet interface wider than 8 bits the packets byte 0 is placed on the bits data[7:0] followed by byte 1 on bits

Pin	Size	Direction	Description
odata_ps_**N**	32	Out	Packet data.
ovalid_bytes_ps_**N**	3	Out	Indicates the number of valid data bytes. For all transactions where *last* is not high, this is equal to the data width in bytes.
ofirst_ps_**N**	1	Out	Start-of-packet flag.
olast_ps_**N**	1	Out	End-of-packet flag. For a correctly transmitted packet *last* is asserted for the last data transaction of the packet. If *last* is set high when *valid_bytes* is zero, the packet shall be dropped or terminated with an error by the MAC.
oupd_ts_ps_**N**	1	Out	The TX MAC should update the PTP Timestamp field in the current packet. Only valid when *first* is set.
oupd_cf_ps_**N**	1	Out	The TX MAC should update the PTP correction field in the current packet. Only valid when *first* is set.
ots_to_sw_ps_**N**	1	Out	The TX MAC should take a timestamp of the current packet and send to software.
ots_ps_**N**	64	Out	PTP Timestamp value. Only valid when *first* is set.
oudp4_ps_**N**	1	Out	The packet is an IPv4/UDP packet. Only valid when *first* is set.
oudp6_ps_**N**	1	Out	The packet is an IPv6/UDP packet.
oudp_csum_ps_**N**	9	Out	Byte position of the start of the UDP checksum field. Only valid when *first* is set.
ots_pos_ps_**N**	9	Out	Byte position of the start of the Timestamp field in a PTP packet. Only valid when *first* is set.
oudp_corr_ps_**N**	15	Out	Byte position of the start of the UDP checksum correction position in a PTP packet. Only valid when *first* is set.
opkt_length_ps_**N**	15	Out	The packet length in bytes. Only valid when *first* is set.
tx_halt_ps_**N**	1	In	Interrupt the data transmission from egress port **N**.

Table 34.5: Packet TX interface for ports 2-10. **N** is the egress interface number.

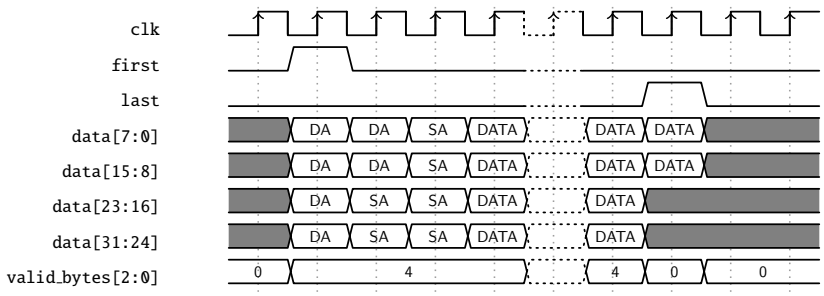


Figure 34.3: Sending and Receiving packets with error (32-bit)

data[15:8] and so on.

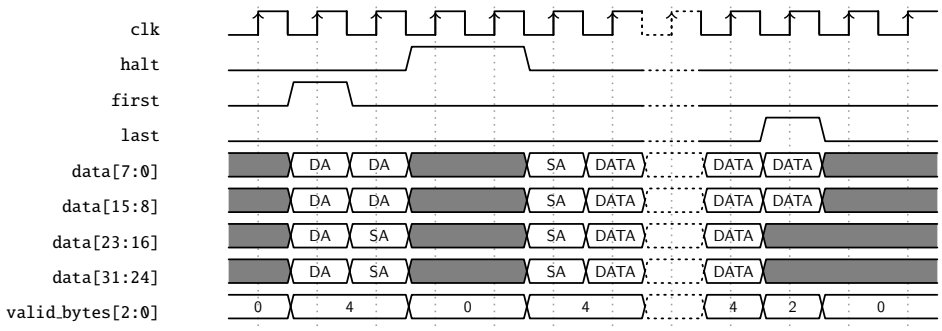


Figure 34.4: Halted transmit packet (32-bit)

The *valid_bytes* indicates how many of the bytes of the data field that holds valid packet data. From the start of a packet this must always be all bytes on the bus up till the last transfer. At the end of the packet on the last bus transfer the *valid_bytes* can indicate less than the full bus width. In this case the byte order is still the same as previous transfers. For example when *valid_bytes* is 1 the last byte of the packet is placed on bits [7:0] and with *valid_bytes* of 2 the last byte of the packet is placed on bits [15:8] and the second to last is on bits [7:0].

34.3 Configuration Interface

The CPU-accessible registers and tables in the core are accessed using the configuration interface.

Each transaction on the configuration interface consists of a request to the core and a resulting reply from the core.

The pins for the configuration interface are listed in Table 34.6 below.

Pin	Size	Direction	Description
apb_paddr	28	In	Address. This is the APB address bus. The highest address bit (27) on the APB bus is not a normal address bit and is referred to as the Accumulator Bit. This is described further in section 35.
apb_psel	1	In	Select.
apb_penable	1	In	Enable.
apb_pwrite	1	In	Direction. This signal indicates an APB write access when HIGH and an APB read access when LOW.
apb_pwdata	32	In	Write data.
apb_pready	1	Out	Ready. The slave uses this signal to extend an APB transfer.
apb_prdata	32	Out	Read Data.
apb_pslverr	1	Out	Error. This signal indicates a transfer failure.

Table 34.6: The APB interface signals

The *paddr* is a byte address, however the core only supports accessing complete 32-bit words. The lowest address bits, which addresses the byte within a bus word, will always be discarded. The register addresses described in this document always refer to word addresses, not byte addresses.

The core has a varying access latency and therefore an APB master should use *pready*.

The *pslverr* signal is set when a transaction is aborted due to an internal timeout. This can occur if the core clock is lower than required and there is a high traffic rate. It will also occur if the address is outside of any defined register.

For a detailed description of the APB interface see the AMBA APB Protocol Specification Version 2.0, available at developer.arm.com

34.4 Interrupt Interface

The interrupt interface is a vector of interrupt flags. When an interrupt occurs it will become a one cycle long pulse on an interrupt flag. I.e. an interrupt has occured whenever an interrupt flag is high on the positive edge of the clock.

There is no interrupt mask nor any interrupt status register to be cleared.

Pin	Function	Size	Direction	Description
interrupts[0]	ldf_level	1	Out	Raised when the level of **Learning Data FIFO** is not below **Learning Data FIFO High Watermark Level** and receives a push request.
interrupts[1]	ldf_full	1	Out	Raised when **Learning Data FIFO** is full but still receives a push request.
interrupts[2]	adf_level	1	Out	Raised when the level of **Aging Data FIFO** is not below **Aging Data FIFO High Watermark Level** and receives a push request.
interrupts[3]	adf_full	1	Out	Raised when **Aging Data FIFO** is full but still receives a push request.
interrupts[4]	hdf_level	1	Out	Raised when the level of **Hit Update Data FIFO** is not below **Hit Update Data FIFO High Watermark Level** and receives a push request.
interrupts[5]	hdf_full	1	Out	Raised when **Hit Update Data FIFO** is full but still receives a push request.
interrupts[6]	hash_aging	1	Out	Indicating an aging process on the L2 hash tables is done by either the hardware aging or the software aging. When Aging Engine is operating with **Software Aging Enable** turned on, it will be silent till **Software Aging Start Latch** is pulled to one and trigger an aging process immediately. **L2 Aging Table** entries are evenly divided to 8 buckets while the aging process loops through them in parallel. Each bucket is checked from the first entry to the last entry and in the end raise a corresponding interrupt.
interrupts[7]	cam_aging	1	Out	Indicating an aging process on the L2 collision table is done by either the hardware aging or the software aging. When **Software Aging Enable** is turned on and **Software Aging Start Latch** is pulled to one, an aging process will loop through all **L2 Aging Collision Table** entries immediately from first to last. After it is done this interrupt will be raised.
interrupts[8:31]	reserved	24	Out	Reserved.

Table 34.7: Interrupt interface

34.5 Pause Interfaces

There are separate pause interfaces for sending status information from the switch to the MAC, *opfc_status*, and from the MAC to the switch, *iext_pause*. Note that these interfaces are in the core clock domain, so they have to be syncronized to the MAC clock if connected to the MAC. However the interfaces can be though of as quasi static. With properly configured pausing thresholds there will never be a short high pulse (due to hysteresis), and losing a short low pulse due to synchronization will create no problems.

34.5.1 PFC Status

The ipfc_status interface is used to transfer pause status from the switch resource manager to the MAC, so the MAC can generate pause frames.

The switch will merely indicate its current pause status, it is up to the MAC to generate the necessary pause frames to keep the far end switch in the desired pausing state.

In port mode the status interface will send 0 in unpaused state, and 0xff in paused state.

34.5.2 External Pause

The *iext_pause* interface is used to transfer PFC pause status received by the MAC to the switch egress scheduler. When the status is XOFF the switch egress scheduler will not send any new packets. Ongoing packets are not affected. There is one iext_pause interface for each packet interface. Even when priority pause is not enabled the external pause interface is still operating per priority.

Pin	Direction	Size	Description
`iext_pause_N`	In	8	Xoff=1, Xon=0 status for each PFC channel (0..7)
`opfc_status_N[7:0]`	Out	8	Xoff=1, Xon=0 status for each PFC channel (0..7)

Table 34.8: ThePFC status and External Pause interfaces, where **N** is the packet interface number

34.6 Debug Read Interface

The debug read interface outputs internal debug signals on the *debug_read_data* port. Which signals to observe is selected with the *debug_read_select* port. The mapping between select value and debug signal is described in Table 34.10. Both these signals are pipelined.

Pin	Direction	Size	Description
`debug_read_select`	In	9	Selects the signal to monitor. See Table 34.10.
`debug_read_data`	In	32	The debug output data.

Table 34.9: The Debug Read interface

id	instance	signal
0	pa_top.switch.mactop	constant-0
1	—"—	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck11 {3'valid_bytes, 1'halt, 1'last, 1'first}
2	—"—	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck10 {3'valid_bytes, 1'halt, 1'last, 1'first}
3	—"—	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck9 {3'valid_bytes, 1'halt, 1'last, 1'first}
4	—"—	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck8 {3'valid_bytes, 1'halt, 1'last, 1'first}
5	—"—	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck7 {3'valid_bytes, 1'halt, 1'last, 1'first}
6	—"—	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck6 {3'valid_bytes, 1'halt, 1'last, 1'first}
7	—"—	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck5 {3'valid_bytes, 1'halt, 1'last, 1'first}
8	—"—	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck4 {3'valid_bytes, 1'halt, 1'last, 1'first}
9	—"—	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck3 {3'valid_bytes, 1'halt, 1'last, 1'first}
10	—"—	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck2 {3'valid_bytes, 1'halt, 1'last, 1'first}
11	—"—	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck1 {5'valid_bytes, 1'halt, 1'last, 1'first}
12	—"—	pa.top.switch.mactop.iTxedgecheck.iProtocolcheck0 {5'valid_bytes, 1'halt, 1'last, 1'first}
13	—"—	pa.top.switch.mactop.iRxedgecheck.iProtocolcheck11 {3'valid_bytes, 1'last, 1'first}
14	—"—	pa.top.switch.mactop.iRxedgecheck.iProtocolcheck10 {3'valid_bytes, 1'last, 1'first}
15	—"—	pa.top.switch.mactop.iRxedgecheck.iProtocolcheck9 {3'valid_bytes, 1'last, 1'first}

id	instance	signal
16	—"—	pa_top.switch.mactop.iRxedgecheck.iProtocolcheck8 {3'valid_bytes, 1'last, 1'first}
17	—"—	pa_top.switch.mactop.iRxedgecheck.iProtocolcheck7 {3'valid_bytes, 1'last, 1'first}
18	—"—	pa_top.switch.mactop.iRxedgecheck.iProtocolcheck6 {3'valid_bytes, 1'last, 1'first}
19	—"—	pa_top.switch.mactop.iRxedgecheck.iProtocolcheck5 {3'valid_bytes, 1'last, 1'first}
20	—"—	pa_top.switch.mactop.iRxedgecheck.iProtocolcheck4 {3'valid_bytes, 1'last, 1'first}
21	—"—	pa_top.switch.mactop.iRxedgecheck.iProtocolcheck3 {3'valid_bytes, 1'last, 1'first}
22	—"—	pa_top.switch.mactop.iRxedgecheck.iProtocolcheck2 {3'valid_bytes, 1'last, 1'first}
23	—"—	pa_top.switch.mactop.iRxedgecheck.iProtocolcheck1 {5'valid_bytes, 1'last, 1'first}
24	—"—	pa_top.switch.mactop.iRxedgecheck.iProtocolcheck0 {5'valid_bytes, 1'last, 1'first}
25	—"—	rx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
26	—"—	tx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
27	—"—	rx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
28	—"—	tx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
29	—"—	rx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
30	—"—	tx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
31	—"—	rx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
32	—"—	tx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
33	—"—	rx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
34	—"—	tx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
35	—"—	rx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
36	—"—	tx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
37	—"—	rx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
38	—"—	tx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
39	—"—	rx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
40	—"—	tx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
41	—"—	rx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
42	—"—	tx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
43	—"—	rx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
44	—"—	tx_pkt_bus {27'data, 3'valid_bytes, 1'last, 1'first}
45	—"—	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
46	—"—	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
47	—"—	rx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
48	—"—	tx_pkt_bus {25'data, 5'valid_bytes, 1'last, 1'first}
49	—"—	constant-49
50	pa_top.switch.ipp0	constant-50
51	—"—	ipp_ipkt_bus {18'data, 8'valid_bytes, 4'id, 1'last, 1'first}
52	—"—	ipp_opkt_bus {18'data, 8'valid_bytes, 4'id, 1'last, 1'first}
53	—"—	pass_da_0
54	—"—	pass_da_1
55	—"—	dut_iIpp_iDropper_dbg_drop
56	—"—	dut_iIpp_iDropper_dbg_ifirst
57	—"—	dut_iIpp_iDropper_dbg_ilast
58	—"—	pass_sa_0
59	—"—	pass_sa_1
60	—"—	constant-60
61	pa_top.switch.ipp0.pm	constant-61
62	—"—	pm_fifo_overflow
63	—"—	dut_dbg_fifo_full
64	—"—	halt_from_pm
65	—"—	dut_iFifoa_11_iF_iFifos_zFcnt_pop_empty
66	—"—	dut_iFifoa_11_iF_iFifos_zFcnt_push_full
67	—"—	dut_iFifoa_10_iF_iFifos_zFcnt_pop_empty
68	—"—	dut_iFifoa_10_iF_iFifos_zFcnt_push_full
69	—"—	dut_iFifoa_9_iF_iFifos_zFcnt_pop_empty
70	—"—	dut_iFifoa_9_iF_iFifos_zFcnt_push_full
71	—"—	dut_iFifoa_8_iF_iFifos_zFcnt_pop_empty
72	—"—	dut_iFifoa_8_iF_iFifos_zFcnt_push_full
73	—"—	dut_iFifoa_7_iF_iFifos_zFcnt_pop_empty
74	—"—	dut_iFifoa_7_iF_iFifos_zFcnt_push_full
75	—"—	dut_iFifoa_6_iF_iFifos_zFcnt_pop_empty
76	—"—	dut_iFifoa_6_iF_iFifos_zFcnt_push_full
77	—"—	dut_iFifoa_5_iF_iFifos_zFcnt_pop_empty
78	—"—	dut_iFifoa_5_iF_iFifos_zFcnt_push_full
79	—"—	dut_iFifoa_4_iF_iFifos_zFcnt_pop_empty
80	—"—	dut_iFifoa_4_iF_iFifos_zFcnt_push_full
81	—"—	dut_iFifoa_3_iF_iFifos_zFcnt_pop_empty
82	—"—	dut_iFifoa_3_iF_iFifos_zFcnt_push_full
83	—"—	dut_iFifoa_2_iF_iFifos_zFcnt_pop_empty
84	—"—	dut_iFifoa_2_iF_iFifos_zFcnt_push_full
85	—"—	dut_iFifoa_1_iF_iFifos_zFcnt_pop_empty
86	—"—	dut_iFifoa_1_iF_iFifos_zFcnt_push_full
87	—"—	dut_iFifoa_0_iF_iFifos_zFcnt_pop_empty
88	—"—	dut_iFifoa_0_iF_iFifos_zFcnt_push_full
89	—"—	constant-89
90	pa_top.switch.sp0	constant-90
91	—"—	dut_iSpbridge_assert_reset_sp_bridge
92	—"—	dut_iSpbridge_assert_reset_sp_bridge
93	—"—	dut_iSpbridge_assert_reset_sp_bridge
94	—"—	dut_iSpbridge_assert_reset_sp_bridge
95	—"—	dut_iSpbridge_assert_reset_sp_bridge

id	instance	signal
96	—"—	dut_iSpbridge_assert_reset_sp_bridge
97	—"—	dut_iSpbridge_assert_reset_sp_bridge
98	—"—	dut_iSpbridge_assert_reset_sp_bridge
99	—"—	dut_iSpbridge_assert_reset_sp_bridge
100	—"—	dut_iSpbridge_assert_reset_sp_bridge
101	—"—	dut_iSpbridge_assert_reset_sp_bridge
102	—"—	dut_iSpbridge_assert_reset_sp_bridge
103	—"—	constant-103
104	pa_top.switch.pb0	constant-104
105	—"—	dut_iPbu_debug_refc_inc
106	—"—	dut_iPbu_debug_port_sch
107	—"—	dut_iPbu_dmux_wrr
108	—"—	dut_iPbu_debug_qenext
109	—"—	dut_iPbu_assert_qediff
110	—"—	dut_iPbu_assert_reque_sp
111	—"—	Mask of currently receiving packets that have been broken due to BM full
112	—"—	dut_iPbu_follow_pfc_accept
113	—"—	dut_iPbu_iAssertpacket_0_assert_out
114	—"—	pa.top.switch.pb0.iAssertpacket0 {8'valid_bytes, 4'port, 1'last, 1'first}
115	—"—	dut_iPbu_iPortshaper_iBuckets_reg_stat
116	—"—	dut_iPbu_zPassdbgqeread_0_o
117	—"—	dut_iPbu_iRequeue_iReFifo_11_iF_iFifos_zFcnt_pop_empty
118	—"—	dut_iPbu_iRequeue_iReFifo_11_iF_iFifos_zFcnt_push_full
119	—"—	dut_iPbu_iRequeue_iReFifo_10_iF_iFifos_zFcnt_pop_empty
120	—"—	dut_iPbu_iRequeue_iReFifo_10_iF_iFifos_zFcnt_push_full
121	—"—	dut_iPbu_iRequeue_iReFifo_9_iF_iFifos_zFcnt_pop_empty
122	—"—	dut_iPbu_iRequeue_iReFifo_9_iF_iFifos_zFcnt_push_full
123	—"—	dut_iPbu_iRequeue_iReFifo_8_iF_iFifos_zFcnt_pop_empty
124	—"—	dut_iPbu_iRequeue_iReFifo_8_iF_iFifos_zFcnt_push_full
125	—"—	dut_iPbu_iRequeue_iReFifo_7_iF_iFifos_zFcnt_pop_empty
126	—"—	dut_iPbu_iRequeue_iReFifo_7_iF_iFifos_zFcnt_push_full
127	—"—	dut_iPbu_iRequeue_iReFifo_6_iF_iFifos_zFcnt_pop_empty
128	—"—	dut_iPbu_iRequeue_iReFifo_6_iF_iFifos_zFcnt_push_full
129	—"—	dut_iPbu_iRequeue_iReFifo_5_iF_iFifos_zFcnt_pop_empty
130	—"—	dut_iPbu_iRequeue_iReFifo_5_iF_iFifos_zFcnt_push_full
131	—"—	dut_iPbu_iRequeue_iReFifo_4_iF_iFifos_zFcnt_pop_empty
132	—"—	dut_iPbu_iRequeue_iReFifo_4_iF_iFifos_zFcnt_push_full
133	—"—	dut_iPbu_iRequeue_iReFifo_3_iF_iFifos_zFcnt_pop_empty
134	—"—	dut_iPbu_iRequeue_iReFifo_3_iF_iFifos_zFcnt_push_full
135	—"—	dut_iPbu_iRequeue_iReFifo_2_iF_iFifos_zFcnt_pop_empty
136	—"—	dut_iPbu_iRequeue_iReFifo_2_iF_iFifos_zFcnt_push_full
137	—"—	dut_iPbu_iRequeue_iReFifo_1_iF_iFifos_zFcnt_pop_empty
138	—"—	dut_iPbu_iRequeue_iReFifo_1_iF_iFifos_zFcnt_push_full
139	—"—	dut_iPbu_iRequeue_iReFifo_0_iF_iFifos_zFcnt_pop_empty
140	—"—	dut_iPbu_iRequeue_iReFifo_0_iF_iFifos_zFcnt_push_full
141	—"—	dut_iPbu_iRefc_refc_mem_debug
142	—"—	dut_iPbu_zPassqesp_zPasslist_0_o
143	—"—	Filter mask for packets dropped by ERM
144	—"—	dut_iPbu_debug_pb_drop
145	—"—	constant-145
146	pa_top.switch.pb0.erm.dut_iEql	constant-146
147	—"—	red_zone
148	—"—	constant-148
149	pa_top.switch.pb0.pfc	constant-149
150	—"—	dut_debug_sp_above_rsv
151	—"—	constant-151
152	pa_top.switch.pb0.qe0	constant-152
153	—"—	dut_assert_dfifo
154	—"—	dut_assert_firstflag
155	—"—	dut_assert_reset_next
156	—"—	dut_drop_cnt
157	—"—	dut_send_cnt
158	—"—	dut_iDfifo_iF_iFifos_zFcnt_pop_empty
159	—"—	dut_iDfifo_iF_iFifos_zFcnt_push_full
160	—"—	dut_ipkt_fifo_11_debug_in
161	—"—	dut_ipkt_fifo_11_debug_out
162	—"—	dut_ipkt_fifo_10_debug_in
163	—"—	dut_ipkt_fifo_10_debug_out
164	—"—	dut_ipkt_fifo_9_debug_in
165	—"—	dut_ipkt_fifo_9_debug_out
166	—"—	dut_ipkt_fifo_8_debug_in
167	—"—	dut_ipkt_fifo_8_debug_out
168	—"—	dut_ipkt_fifo_7_debug_in
169	—"—	dut_ipkt_fifo_7_debug_out
170	—"—	dut_ipkt_fifo_6_debug_in
171	—"—	dut_ipkt_fifo_6_debug_out
172	—"—	dut_ipkt_fifo_5_debug_in
173	—"—	dut_ipkt_fifo_5_debug_out
174	—"—	dut_ipkt_fifo_4_debug_in
175	—"—	dut_ipkt_fifo_4_debug_out

id	instance	signal
176	—"—	dut_ipkt_fifo_3_debug_in
177	—"—	dut_ipkt_fifo_3_debug_out
178	—"—	dut_ipkt_fifo_2_debug_in
179	—"—	dut_ipkt_fifo_2_debug_out
180	—"—	dut_ipkt_fifo_1_debug_in
181	—"—	dut_ipkt_fifo_1_debug_out
182	—"—	dut_ipkt_fifo_0_debug_in
183	—"—	dut_ipkt_fifo_0_debug_out
184	—"—	dut_pfifo_level
185	—"—	dut_pfifo_level
186	—"—	dut_pfifo_level
187	—"—	dut_pfifo_level
188	—"—	dut_pfifo_level
189	—"—	dut_pfifo_level
190	—"—	dut_pfifo_level
191	—"—	dut_pfifo_level
192	—"—	dut_pfifo_level
193	—"—	dut_pfifo_level
194	—"—	dut_pfifo_level
195	—"—	dut_pfifo_level
196	—"—	constant-196
197	pa_top.switch.pb0.wrr	constant-197
198	—"—	dut_debug_below
199	—"—	dut_zPassdebugbvalpipe_zPasslist_7_o
200	—"—	dut_zPassdebugbvalpipe_zPasslist_6_o
201	—"—	dut_zPassdebugbvalpipe_zPasslist_5_o
202	—"—	dut_zPassdebugbvalpipe_zPasslist_4_o
203	—"—	dut_zPassdebugbvalpipe_zPasslist_3_o
204	—"—	dut_zPassdebugbvalpipe_zPasslist_2_o
205	—"—	dut_zPassdebugbvalpipe_zPasslist_1_o
206	—"—	dut_zPassdebugbvalpipe_zPasslist_0_o
207	—"—	dut_reg_bval
208	—"—	dut_reg_bval
209	—"—	dut_reg_bval
210	—"—	dut_reg_bval
211	—"—	dut_reg_bval
212	—"—	dut_reg_bval
213	—"—	dut_reg_bval
214	—"—	dut_reg_bval
215	—"—	dut_reg_bval
216	—"—	dut_reg_bval
217	—"—	dut_reg_bval
218	—"—	dut_reg_bval
219	—"—	dut_reg_bval
220	—"—	dut_reg_bval
221	—"—	dut_reg_bval
222	—"—	dut_reg_bval
223	—"—	dut_reg_bval
224	—"—	dut_reg_bval
225	—"—	dut_reg_bval
226	—"—	dut_reg_bval
227	—"—	dut_reg_bval
228	—"—	dut_reg_bval
229	—"—	dut_reg_bval
230	—"—	dut_reg_bval
231	—"—	dut_reg_bval
232	—"—	dut_reg_bval
233	—"—	dut_reg_bval
234	—"—	dut_reg_bval
235	—"—	dut_reg_bval
236	—"—	dut_reg_bval
237	—"—	dut_reg_bval
238	—"—	dut_reg_bval
239	—"—	dut_reg_bval
240	—"—	dut_reg_bval
241	—"—	dut_reg_bval
242	—"—	dut_reg_bval
243	—"—	dut_reg_bval
244	—"—	dut_reg_bval
245	—"—	dut_reg_bval
246	—"—	dut_reg_bval
247	—"—	dut_reg_bval
248	—"—	dut_reg_bval
249	—"—	dut_reg_bval
250	—"—	dut_reg_bval
251	—"—	dut_reg_bval
252	—"—	dut_reg_bval
253	—"—	dut_reg_bval
254	—"—	dut_reg_bval
255	—"—	dut_reg_bval

id	instance	signal
256	—"—	dut_reg_bval
257	—"—	dut_reg_bval
258	—"—	dut_reg_bval
259	—"—	dut_reg_bval
260	—"—	dut_reg_bval
261	—"—	dut_reg_bval
262	—"—	dut_reg_bval
263	—"—	dut_reg_bval
264	—"—	dut_reg_bval
265	—"—	dut_reg_bval
266	—"—	dut_reg_bval
267	—"—	dut_reg_bval
268	—"—	dut_reg_bval
269	—"—	dut_reg_bval
270	—"—	dut_reg_bval
271	—"—	dut_reg_bval
272	—"—	dut_reg_bval
273	—"—	dut_reg_bval
274	—"—	dut_reg_bval
275	—"—	dut_reg_bval
276	—"—	dut_reg_bval
277	—"—	dut_reg_bval
278	—"—	dut_reg_bval
279	—"—	dut_reg_bval
280	—"—	dut_reg_bval
281	—"—	dut_reg_bval
282	—"—	dut_reg_bval
283	—"—	dut_reg_bval
284	—"—	dut_reg_bval
285	—"—	dut_reg_bval
286	—"—	dut_reg_bval
287	—"—	dut_reg_bval
288	—"—	dut_reg_bval
289	—"—	dut_reg_bval
290	—"—	dut_reg_bval
291	—"—	dut_reg_bval
292	—"—	dut_reg_bval
293	—"—	dut_reg_bval
294	—"—	dut_reg_bval
295	—"—	dut_reg_bval
296	—"—	dut_reg_bval
297	—"—	dut_reg_bval
298	—"—	dut_reg_bval
299	—"—	dut_reg_bval
300	—"—	dut_reg_bval
301	—"—	dut_reg_bval
302	—"—	dut_reg_bval
303	—"—	dut_reg_rank
304	—"—	dut_reg_rank
305	—"—	dut_reg_rank
306	—"—	dut_reg_rank
307	—"—	dut_reg_rank
308	—"—	dut_reg_rank
309	—"—	dut_reg_rank
310	—"—	dut_reg_rank
311	—"—	dut_reg_rank
312	—"—	dut_reg_rank
313	—"—	dut_reg_rank
314	—"—	dut_reg_rank
315	—"—	constant-315
316	pa_top.switch.pb0.qshp	constant-316
317	—"—	dut_iPrioshaper_reg_stat
318	—"—	dut_iQueueshaper_reg_stat
319	—"—	constant-319
320	pa_top.switch.bm0	constant-320
321	—"—	dut_bm_ifree_debug_free
322	—"—	constant-322
323	pa_top.switch.ps0	constant-323
324	—"—	halt_from_ps
325	—"—	dut_iPs2_zPsAssert_item
326	—"—	dut_iPs2_iBridge_10_assert_reset
327	—"—	dut_iPs2_iBridge_9_assert_reset
328	—"—	dut_iPs2_iBridge_8_assert_reset
329	—"—	dut_iPs2_iBridge_7_assert_reset
330	—"—	dut_iPs2_iBridge_6_assert_reset
331	—"—	dut_iPs2_iBridge_5_assert_reset
332	—"—	dut_iPs2_iBridge_4_assert_reset
333	—"—	dut_iPs2_iBridge_3_assert_reset
334	—"—	dut_iPs2_iBridge_2_assert_reset
335	—"—	dut_iPs2_iBridge_1_assert_reset

id	instance	signal
336	—"—	dut_iPs2_iBridge_0_assert_reset
337	—"—	dut_iPs2_iSplitter_0_assert_noend
338	—"—	dut_iPs2_iSplitter_0_assert_ptr
339	—"—	dut_iPs2_iSplitter_0_used_mem
340	—"—	dut_iPs2_iSplitter_0_used_mem
341	—"—	dut_iPs2_iSplitter_0_used_mem
342	—"—	dut_iPs2_iSplitter_0_used_mem
343	—"—	dut_iPs2_iSplitter_0_used_mem
344	—"—	dut_iPs2_iSplitter_0_used_mem
345	—"—	dut_iPs2_iSplitter_0_used_mem
346	—"—	dut_iPs2_iSplitter_0_used_mem
347	—"—	dut_iPs2_iSplitter_0_used_mem
348	—"—	dut_iPs2_iSplitter_0_used_mem
349	—"—	dut_iPs2_iSplitter_0_used_mem
350	—"—	dut_iPs2_iSplitter_0_used_mem
351	—"—	constant-351
352	pa_top.switch.epp0	constant-352
353	—"—	dut_iEpp_assert_ipkt
354	—"—	dut_iEpp_assert_opkt
355	—"—	epp_ipkt_bus {18'data, 8'valid_bytes, 4'id, 1'last, 1'first}
356	—"—	epp_opkt_bus {18'data, 8'valid_bytes, 4'id, 1'last, 1'first}
357	—"—	dut_iEpp_iDropper_da_0
358	—"—	dut_iEpp_iDropper_da_1
359	—"—	dut_iEpp_iDropper_dbg_drop
360	—"—	dut_iEpp_iDropper_dbg_ifirst
361	—"—	dut_iEpp_iDropper_dbg_ilast
362	—"—	dut_iEpp_iDropper_sa_0
363	—"—	dut_iEpp_iDropper_sa_1
364	—"—	pa.top.switch.epp0.iPacketassertpm {8'valid_bytes, 4'port, 1'last, 1'first}
365	—"—	pa.top.switch.epp0.iPacketassertin {8'valid_bytes, 4'port, 1'last, 1'first}
366	—"—	constant-366
367	pa_top.switch.epp0.pm	constant-367
368	—"—	pm_fifo_overflow
369	—"—	dut_dbg_fifo_full
370	—"—	halt_from_pm
371	—"—	dut_iFifoa_11_iF_iFifos_zFcnt_pop_empty
372	—"—	dut_iFifoa_11_iF_iFifos_zFcnt_push_full
373	—"—	dut_iFifoa_10_iF_iFifos_zFcnt_pop_empty
374	—"—	dut_iFifoa_10_iF_iFifos_zFcnt_push_full
375	—"—	dut_iFifoa_9_iF_iFifos_zFcnt_pop_empty
376	—"—	dut_iFifoa_9_iF_iFifos_zFcnt_push_full
377	—"—	dut_iFifoa_8_iF_iFifos_zFcnt_pop_empty
378	—"—	dut_iFifoa_8_iF_iFifos_zFcnt_push_full
379	—"—	dut_iFifoa_7_iF_iFifos_zFcnt_pop_empty
380	—"—	dut_iFifoa_7_iF_iFifos_zFcnt_push_full
381	—"—	dut_iFifoa_6_iF_iFifos_zFcnt_pop_empty
382	—"—	dut_iFifoa_6_iF_iFifos_zFcnt_push_full
383	—"—	dut_iFifoa_5_iF_iFifos_zFcnt_pop_empty
384	—"—	dut_iFifoa_5_iF_iFifos_zFcnt_push_full
385	—"—	dut_iFifoa_4_iF_iFifos_zFcnt_pop_empty
386	—"—	dut_iFifoa_4_iF_iFifos_zFcnt_push_full
387	—"—	dut_iFifoa_3_iF_iFifos_zFcnt_pop_empty
388	—"—	dut_iFifoa_3_iF_iFifos_zFcnt_push_full
389	—"—	dut_iFifoa_2_iF_iFifos_zFcnt_pop_empty
390	—"—	dut_iFifoa_2_iF_iFifos_zFcnt_push_full
391	—"—	dut_iFifoa_1_iF_iFifos_zFcnt_pop_empty
392	—"—	dut_iFifoa_1_iF_iFifos_zFcnt_push_full
393	—"—	dut_iFifoa_0_iF_iFifos_zFcnt_pop_empty
394	—"—	dut_iFifoa_0_iF_iFifos_zFcnt_push_full
395	—"—	constant-395
396	pa_top.switch.ingress_common	constant-396
397	—"—	dut_iLearnage_iHitUpdate_iFifo_0_iF_iFifos_zFcnt_pop_empty
398	—"—	dut_iLearnage_iHitUpdate_iFifo_0_iF_iFifos_zFcnt_push_full
399	—"—	dut_iLearnage_iConf_iFifo_2_iFifo_iF_iFifos_zFcnt_pop_empty
400	—"—	dut_iLearnage_iConf_iFifo_2_iFifo_iF_iFifos_zFcnt_push_full
401	—"—	dut_iLearnage_iConf_iFifo_1_iFifo_iF_iFifos_zFcnt_pop_empty
402	—"—	dut_iLearnage_iConf_iFifo_1_iFifo_iF_iFifos_zFcnt_push_full
403	—"—	dut_iLearnage_iConf_iFifo_0_iFifo_iF_iFifos_zFcnt_pop_empty
404	—"—	dut_iLearnage_iConf_iFifo_0_iFifo_iF_iFifos_zFcnt_push_full
405	—"—	dut_iMbsc_iFlood_reg_stat
406	—"—	dut_iMbsc_iMc_reg_stat
407	—"—	dut_iMbsc_iBc_reg_stat
408	—"—	constant-408
409	pa_top.switch.interface_common	constant-409
410	—"—	dut_zFaii_iMf_zMf_1_item
411	—"—	dut_zFaip_iMf_zMf_1_item
412	—"—	dut_zFaie_iMf_zMf_1_item
413	—"—	dut_zFaiq_iMf_zMf_1_item
414	—"—	dut_zFais_iMf_zMf_1_item
415	—"—	constant-415

id	instance	signal
416	pa_top.cryptotop0	constant-416
417	—"—	dut_iPFifo_iF_iFifos_zFcnt_pop_empty
418	—"—	dut_iPFifo_iF_iFifos_zFcnt_push_full
419	—"—	constant-419

Table 34.10: Debug Selection Map

34.7 Debug Write Interface

The debug write interface is an input port to the Switch Core that can be used for debugging purposes. In normal operation the *debug_write_data* pins must be tied low. The function of the debug write interface is controlled by registers in the individual blocks. In this core only the tick dividers use the debug write interface. See **Core Tick Select**.

Pin	Direction	Size	Description
debug_write_data	In	1	The debug write input data. Must be tied low for normal switch operation.

Table 34.11: The Debug Write interface

Chapter 35

Configuration Interface

The configuration interface is an AMBA APB interface used for monitoring the core and for configuration of internal registers and tables. The pins are described in Table 34.6 on page 203, but for a detailed description of the APB interface see the AMBA APB Protocol Specification Version 2.0, available at developer.arm.com

35.1 Response time

The response time may vary between registers, and even vary for the same register depending on how busy the core is switching packets. The response time is in the order of tens of core clock cycles.

35.2 Out of range accesses

There is no range check on the configuration interface, so an access to an address that is not mapped to any register will result in a internal timeout and raise the *pslverr* on the bus.

35.3 Atomic Wide Access

There are a few recommendations how to access wide registers (registers that are wider than the APB data bus). The interface does allow a more flexible access pattern than what is described here. If that is needed then see the next section.

The highest address bit (27) on the APB bus is not a normal address bit. It is used to control wide register access. It will be referred to as the Accumulator Bit in the following description.

- Wide Reads
  - always read wide register starting with the lowest address and ending with the highest address.
  - when reading the lowest address of the register the Accumulator Bit should be 0.
  - when reading the other addresses of the register the Accumulator Bit should be 1.
- Wide Writes
  - always write wide register starting with the lowest address and ending with the highest address.
  - when writing the highest address of the register the Accumulator Bit should be 0.
  - when writing the other addresses of the register the Accumulator Bit should be 1.
- Narrow reads and writes
  If the register fits within the APB data bus width then the Accumulator Bit should be 0.

Note that if there are bridges between the CPU and the APB bus then they need to be set up to guarantee the order of accesses.

The software API implementation provided with the switch handles the Accumulator Bit thereby hiding it completely for the software that use the API.

35.4 Accumulator Accesses

Each table or register bank where the data is wider than the configuration data bus, will be equipped with a shadow-register called an accumulator. The accumulator allows the full data width to be updated atomically even though the bus width is narrower than the data. The accumulator is accessed by setting bit 27 of the address high during a normal register access. An access with bit 27 of the address low we call a `DEFAULT` access, while an access with bit 27 of the address high is called an `ACCUMULATOR` access. The register section of the datasheet will only list the addresses for `DEFAULT` access to the registers. Address bit 27 is considered an accumulator flag, and not a part of the address.

A `DEFAULT` read will return the requested data in the reply, and at the same time load the full data width into the accumulator. Thus following up the `DEFAULT` read with `ACCUMULATOR` reads will allow reading the state of the register at the time of the original `DEFAULT` read. If data consistency is not important, all the reads can be of the `DEFAULT` type, but there is no point because the read performance is the same. In fact reading a table will potentially be faster using the accumulator, because only the first access will have to wait for access to the physical memory.

Writes work similarly, but the other way around. The accumulator will first be loaded using `ACCUMULATOR` writes and then the contents of the accumulator is written to the register. The final `DEFAULT` write will use the data given as *wdata*, and fill it out with the data in the accumulator. Writing data wider than the bus cannot be done without taking the accumulator into account.

If only a part of a very wide register is to be written, the most efficient approach may be to do a `DEFAULT` read (loading the accumulator) followed by a `DEFAULT` write. But note that there is no way to do a truly atomic read-modify-write. Any write that the core slips in while the accumulator is loaded will be over-written by the following `DEFAULT` write.

When the data is wider than the bus the address is stepped by 2^n between table indexes or registers. For instance a 32-bit bus and a 65 bit wide table will result in index 1 starting at address 4, with address 3 unused and address 2 only containing a single valid bit.

Chapter 36

Debugging the Design

The design contains debug points. They are avalible as registers in the design. For each debug point there is a a counter. The fields which are more than a single bit also have a comparison register. This register is used for updating the counter only for specific matching values.

36.1 Debug Counters in Ingress Packet Processing

The Cnt Id field in the table below points to the counter to be updated in the counter bank of the **Debug IPP Counter** register.

Register	Cnt Id	Bits	Description
IPP Debug finalVid	0	13	The VID used to lookup in the VLAN table The setup of mask and compare is located in register **Debug Counter finalVid Setup**. This register enables the user to see if a specific value has been seen.
IPP Debug vlanVidOp	1	3	The VLAN Table VID Operation The setup of mask and compare is located in register **Debug Counter vlanVidOp Setup**. This register enables the user to see if a specific value has been seen.
IPP Debug l2DaTcamHitsAndCast	2	17	If the L2 TCAM was hit and which type was returned. The setup of mask and compare is located in register **Debug Counter l2DaTcamHitsAndCast Setup**. This register enables the user to see if a specific value has been seen.
IPP Debug l2DaHashKey	3	60	The hash value for the packet. The setup of mask and compare is located in register **Debug Counter l2DaHashKey Setup**. This register enables the user to see if a specific value has been seen.
IPP Debug l2DaHash	4	11	The hash value for the packet. The setup of mask and compare is located in register **Debug Counter l2DaHash Setup**. This register enables the user to see if a specific value has been seen.
IPP Debug l2DaHashHitAndBucket	5	4	The L2 bucket used and hit bit. The setup of mask and compare is located in register **Debug Counter l2DaHashHitAndBucket Setup**. This register enables the user to see if a specific value has been seen.
IPP Debug l2DaHashHitAndBucket	5	4	The L2 bucket used and hit bit. The setup of mask and compare is located in register **Debug Counter l2DaHashHitAndBucket Setup**. This register enables the user to see if a specific value has been seen.
IPP Debug routerHit	6	1	The router was hit
IPP Debug nextHopPtrLpm	7	11	The LPM functions next hop pointer The setup of mask and compare is located in register **Debug Counter nextHopPtrLpm Setup**. This register enables the user to see if a specific value has been seen.
IPP Debug nextHopPtrHash	8	11	The L3 hash functions next hop pointer The setup of mask and compare is located in register **Debug Counter nextHopPtrHash Setup**. This register enables the user to see if a specific value has been seen.
IPP Debug nextHopPtrLpmHit	9	1	The LPM functions had a hit in the LPM table.
IPP Debug nextHopPtrHashHit	10	1	The L3 hash functions had a hit in the L3 hash table.

Register	Cnt Id	Bits	Description
IPP Debug nextHopPtrFinal	11	11	The final next hop pointer after ECMP and default route. The setup of mask and compare is located in register **Debug Counter nextHopPtrFinal Setup**. This register enables the user to see if a specific value has been seen.
IPP Debug srcPort	12	4	The source port which the packet came in on. The setup of mask and compare is located in register **Debug Counter srcPort Setup**. This register enables the user to see if a specific value has been seen.
IPP Debug dropPktAfterL2Decode	13	1	Packet was dropped after L2 packet decoder
IPP Debug nrVlans	14	2	The number of VLANs the incoming packet has The setup of mask and compare is located in register **Debug Counter nrVlans Setup**. This register enables the user to see if a specific value has been seen.
IPP Debug dropPktAfterL3Decode	15	1	Packet was dropped after L3 packet decoder
IPP Debug spVidOp	16	3	The Source port VID Operation The setup of mask and compare is located in register **Debug Counter spVidOp Setup**. This register enables the user to see if a specific value has been seen.
IPP Debug routed	17	1	The packet was routed
IPP Debug isFlooding	18	1	Was the packet flooded
IPP Debug isBroadcast	19	1	Was the packet broadcased
IPP Debug doL2Lookup	20	1	This packet shall do lookup in L2 tables.
IPP Debug dstPortmask	21	11	The packets final portmask The setup of mask and compare is located in register **Debug Counter dstPortmask Setup**. This register enables the user to see if a specific value has been seen.
IPP Debug debugMatchIPP0	22	22	This allows a user to match all the above debug registers to make a counter update. This allows a user to update a counter based on multiple events happening for the same packet.The Cnt bit indicates which bit is is in the bit-field. The setup of mask and compare is located in register **Debug Counter debugMatchIPP0 Setup**. This register enables the user to see if a specific value has been seen.

Table 36.1: IPP Debug List

36.2 Debug Counters in Egress Packet Processing

The Cnt Id field in the table below points to the counter to be updated in the counter bank of the **Debug EPP Counter** register.

Register	Cnt Id	Bits	Description
EPP Debug delSpecificVlan	0	1	This packet has a vid which shall be viewed as a priority VID and it will be deleted from the outgoing packet.
EPP Debug updateTosExp	1	1	This packet shall have a updated TOS/EXP field.
EPP Debug isIPv4	2	1	Packet is a IPv4 packet.
EPP Debug isIPv6	3	1	Packet is a IPv6 packet.
EPP Debug addNewMpls	4	1	Packet shall add a new MPLS header.
EPP Debug isPPPoE	5	1	Packet has a PPPoE header.
EPP Debug imActive	6	1	This packet shall be input mirrored.
EPP Debug imActive	6	1	This packet shall be input mirrored by sending out a second copy to the same destination port.
EPP Debug imExtra	7	1	This packet will send a extra input mirrored packet copy since the packet is already going out on this port.
EPP Debug omEnabled	8	1	This packet shall be output mirrored.
EPP Debug omImActive	9	1	This packet shall be both input mirrored and output mirroed.
EPP Debug reQueue	10	1	This packet shall be requeued.
EPP Debug reQueuePortId	11	4	This packet shall be requeued to this port. The setup of mask and compare is located in **Debug Counter reQueuePortId Setup**. This register enables the user to see if a specific value has been seen.
EPP Debug reQueuePkt	12	1	This packet will be requeued one more time since on the same port there shall be multiple copies.
EPP Debug fromPort	13	11	The port which the packet is going to be sent out on. The setup of mask and compare is located in **Debug Counter fromPort Setup**. This register enables the user to see if a specific value has been seen.
EPP Debug debugMatchEPP0	14	14	This allows a user to match all or part of them. This allows a user to update a counter based on multiple events happening for the same packet. The Cnt bit indicates which bit is is in the bit-field. The setup of mask and compare is located in **Debug Counter debugMatchEPP0 Setup**. This register enables the user to see if a specific value has been seen.

Table 36.2: EPP Debug List

Chapter 37

Implementation

37.1 Floorplanning

The top of the core is the *pa_top* level, it wraps the switch core, *pa_top_switch*, and may also contain interface bridges.

The switch hierarchy is divided into six major blocks that we call floorplan blocks. These are: SP, IPP, BM, PB, EPP, and PS. There is also two smaller blocks: ingress_common, interface_common. In some configurations these are very small, but in some the ingress_common can be quite substantial.

Besides the configuration bus, which spreads it's tentacles to every corner of the core, the dataflow through the floorplan blocks is basically that of the path of a packet. The flow from ingress to egress is SP, IPP, BM/PB, EPP, and PS. The PB/BM are lumped together in the list because the packet data goes through the BM, and the control data through the PB. The ingress_common contains auxillary functions for the ingress packet processing and thus mainly talks to the IPP. The other small block, interface_common, is mostly comprised of shim logic for the external interfaces.

37.1.1 Pipelining

The number of pipeline stages in the data paths between the floorplan blocks can be set freely when the RTL is generated. The same goes for the number of input flops and output flops on each floorplan block. If you need to change the number of pipeline stages it is a trivial task, but the RTL has to be re-generated. It cannot be adjusted in the existing verilog files.

Connection	Pipeline stages
SP $\leftrightarrow$ IPP	1
IPP $\leftrightarrow$ PB/BM	1
PB $\leftrightarrow$ BM	1
BM $\leftrightarrow$ EPP	1
EPP $\leftrightarrow$ PS	1

Table 37.1: The settings for pipeline flops between floorplan blocks

Floorplan block	Input flops	Output flops
SP	0	0
IPP	0	0
PB	0	1
BM	0	0
EPP	0	0
PS	1	1

Table 37.2: The settings for input and output flops for the floorplan blocks

The pipeline settings used when generating this core are shown in Table 37.1, and the input/output flops are listed in Table 37.2[1].

37.1.2 Configuration and debug

The configuration and debug busses are in principle extremely flexible in how they can be pipelined. Flops can be added and removed anywhere so long as each bus is still in sync. This, as the other changes in pipelining, can only be done by generating new RTL.

37.2 Clock crossings

The bulk of the core is in a single clock domain, the core domain, driven by the *clk* clock. Each packet interface has separate clock domains for TX and RX. All paths between these domains are synchronized by either two synchronization flops, or by an asynchronous memory. The synchronization flops are always instantiations of the *verilog_sync_flops* verilog module, and the asynchronous memories are always instantiations of *verilog_memory_2c*.

37.2.1 IPP and EPP Structure

The IPP and EPP modules are both pipelines with a main dataflow from input to output. The floorplan is recommended to follow the pipeline dataflow. The logic input to a memory comes from the preceding pipeline stage and the output goes to the following pipeline stage. Which pipeline stage a specific memory belongs to is documented in the delivered files eppp0_raw_opt.ramstat and ippp0_raw_opt.ramstat.

In addition to the memory instances, the pipeline flipflops belonging to each pipeline stage is documented in ippp0_raw_opt.fflist and eppp0_raw_opt.fflist.

The exact Verilog instance names are not listed in these files but the names in the lists are part of the instance names and uniquely identify them.

In addition to the main dataflow there is also a configuration bus that has access to all memory instances and to the configuration registers. These paths are normally not in the critical path.

The configuration registers as opposed to the configuration memories can be accessed in multiple pipeline stages and therefore does not have a simple placement strategy.

37.3 Memory wrappers

The memories in the core are instantiated using the verilog_memory.v wrapper. It is expected that this wrapper is replaced, or modified, by the customer to instanciate appropriate memory macros. The macros needed are listed in Table 37.3. For memories with the *write_through* attribute set, simultaneous reading and writing the of same address is expected to yield the write data as read result. For memories with *write_through* set to 0 simultaneous reading and writing to the same address shall not occur.

type	width	depth	write through	write mask	input flops	output flops
dp	3	2048	1	None	0	0
dp	123	32	1	None	0	0
dp	577	48	1	None	0	0
dp	286	16	1	None	0	0
dp	592	2048	1	None	0	0
dp	592	256	1	None	0	0
dp	161	16	1	None	0	0
dp	4	1024	1	None	0	0
dp	453	128	1	None	0	0

[1]It should be noted that the input/output flops for the PS is not as clear cut as for the other blocks, due to the slightly more complex interface to the MAC.

dp	453	256	1	None	0	0
dp	771	64	1	None	0	0
dp	771	16	1	None	0	0
dp	210	16	1	None	0	0
dp	221	11	1	None	0	0
dp	125	4096	1	None	0	0
dp	156	16384	1	None	0	0
dp	63	2048	1	None	0	0
dp	54	2048	1	None	0	0
dp	1	2048	1	None	0	0
dp	60	2048	1	None	0	0
dp	40	16416	1	None	0	0
dp	24	128	1	None	0	0
dp	211	2048	1	None	0	0
dp	211	256	1	None	0	0
dp	92	64	1	None	0	0
dp	32	64	1	None	0	0
dp	120	64	1	None	0	0
dp	60	66	0	None	0	0
dp	117	22	0	None	0	0
dp	1440	9	0	None	0	0
dp	333	9	0	None	0	0
dp	37	79	0	None	0	0
dp	180	17	0	None	0	0
dp	258	20	0	None	0	0
dp	358	10	0	None	0	0
dp	266	41	0	None	0	0
dp	319	10	0	None	0	0
dp	128	20	0	None	0	0
dp	130	20	0	None	0	0
dp	96	32	0	None	0	0
dp	72	32	0	None	0	0
dp	48	49	0	None	0	0
dp	333	10	0	None	0	0
dp	268	39	0	None	0	0
dp	257	31	0	None	0	0
dp	256	8	0	None	0	0
dp	67	64	0	None	0	0
dp	33	66	0	None	0	0
dp	333	13	0	None	0	0
dp	132	39	0	None	0	0
dp	131	39	0	None	0	0
dp	177	13	0	None	0	0
dp	64	33	0	None	0	0
dp	70	38	0	None	0	0
dp	272	98	0	None	0	0
dp	1264	98	0	None	0	0
dp	1904	50	0	None	0	0
dp	1536	12	1	None	0	0
dp	8	1024	0	None	0	0
dp	19	1024	0	None	0	0
dp	4	1024	0	None	0	0
dp	82	1024	0	None	0	0
dp	11	1024	0	None	0	0
dp	33	1024	1	None	0	0

dp	1536	1024	0	None	0	0
dp	10	1024	1	None	0	0
dp	512	32	1	None	0	0
dp	38	64	1	None	0	0
dp	57	2048	1	None	0	0
dp	31	2048	1	None	0	0
dp	139	1024	1	None	0	0
dp	46	128	1	None	0	0
dp	46	64	1	None	0	0
dp	54	8192	1	None	0	0
dp	10	256	1	None	0	0
dp	18	256	1	None	0	0
dp	9	256	1	None	0	0
dp	1	8192	1	None	0	0
dp	580	32	0	None	0	0
dp	197	14	0	None	0	0
dp	880	8	0	None	0	0
dp	2464	8	0	None	0	0
dp	163	14	0	None	0	0
dp	512	32	0	None	0	0
dp	272	29	0	None	0	0
dp	288	31	0	None	0	0
dp	352	31	0	None	0	0
dp	128	29	0	None	0	0
dp	192	29	0	None	0	0
dp	2224	19	0	None	0	0
dp	576	20	0	None	0	0
dp	320	19	0	None	0	0
dp	68	35	0	None	0	0
dp	240	19	0	None	0	0
dp	432	19	0	None	0	0
dp	2656	19	0	None	0	0
dp	192	21	0	None	0	0
dp	130	18	0	None	0	0
dp	2342	446	0	None	0	0
dp	9	446	1	None	0	0
dp	629	108	0	None	0	0
dp	940	64	1	None	0	0
dp	192	64	1	None	0	0
dp	36	4123	1	None	0	0
dc	13	8	0	None	0	0
dc	193	8	0	None	0	0
dc	97	8	0	None	0	0
dc	14	16	0	None	0	0
dc	245	16	0	None	0	0
dc	149	16	0	None	0	0
dc	63	8	0	None	0	0
dc	35	8	0	None	0	0

Table 37.3: The memory macros needed for this core.Types: dp=two ports, one read and one write, running on the same clock. dc=two ports, one read and one write, with separate clocks for read and write.

For this design all dual-clock memories are generated as memory instances, but for synchronous memories only those with 2048 bits or more have been generated as a memory instance. Smaller synchronous memories are created as arrays of flops in the verilog source code. To change the criterium for making a memory as an instance or as an array of flops, new RTL has to be generated[2].

37.4 Dual ported memories

All memories are dual ported. Some dual-ported memories have different clocks for the two ports, these are all instanciated using *verilog_memory_2c* wrapper. For these a real dual-port memory macro is the preferred choice. Most dual-port memories, however, are running on a single clock, and for these a better approch is to use a single-port memory macro clocked at twice the frequency. Unless, of course, the frequency would be prohibitively high. Note in the example timing diagram that the write is done in the first clock cycle to satisfy the *write_through* criterium. For memories that are not *write_through* it may be desirable for timing reasons to have the read in the first clock cycle.

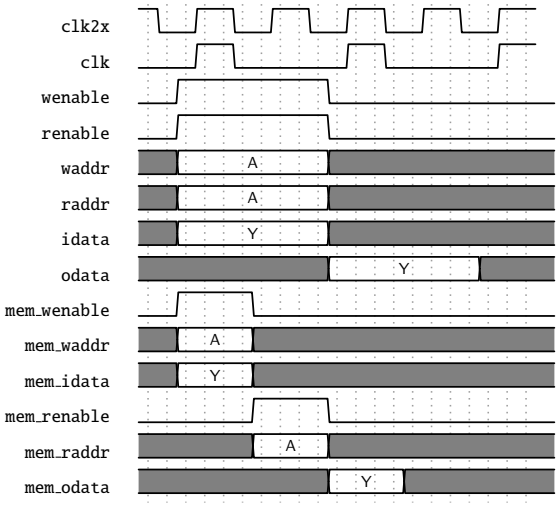


Figure 37.1: Timing diagram for a single ported memory used in the dual ported memory wrapper. In this case a concurrent read and write to the same address of a memory wrapper set for one cycle latency and with the write through attribute set.

There is no dedicated double frequency clock connected to the memories, it has to be provided using the *meminst_in busses to the memory wrappers.

37.5 Memory timing

All memories in the design can be selected to have either:

- One cycle latency
- Two cycles latency, with the flop added on the input to the memory
- Two cycles latency, with the flop added on the output from the memory
- Three cycles latency, with flops added on both the input and the output

Which setting is used for each memory instance can be seen in the *input flops* and *output flops* columns of Table 37.3.

[2]Although, any instantiated memory wrapper can of course be left as is, and thus be implemented as an array of flops in synthesis.

37.6 Lint set up

For spyglass linting the following settings are assumed:

- set_parameter ignore_local_variables yes
- set_parameter handle_zero_padding "W362"

37.6.1 Waivers

Besides the inline waivers in the code these blanket waivers shall be applied:

- waive -rule STARC05-2.11.3.1 -comment "Case statements are used in the sequential blocks of state-machines. This is not an issue"
- waive -rule STARC05-2.2.3.3 -comment "Flip-flops may be written several times in the same sequential block. This is not an issue"
- waive -regexp -du "consistency_check.*" -rule "W240" -comment "consistency_check is guarded by SYNTHESIS, and is not used in hardware."
- waive -rule W415a -comment "Assigning multiple times in the same always block is a code style we use. This is not an issue"
- waive -rule W528 -comment "The way we pipeline will leave a lot of unread signals. This is not an issue"

Chapter 38

Registers and Tables

Contents

All registers and tables that are accessible from a configuration interface are listed in this chapter. A user guide for the configuration interface is found in Chapter 35, and the pins for the configuration interfaces are described in Section 34.3.

38.1 Address Space For Tables and Registers

All tables in the address space are linear. The size of a table entry is always rounded up to nearest power of two of the bus width. For example if the bus is 32 bits and a entry in a table is 33 bits wide, it will then use two addresses per entry. Second example, the bus is still 32 bits, but the entry is 181 bits wide, the entry will then use a address space of 8 addresses per table entry (181 bits fits within 6 bus words but is rounded up to nearest power of two). This is shown in figure 38.1. The total address space used by this core is 1201254 addresses.

38.2 Byte Order

When a register field is wider than a byte and the field represents an integer value or the field is related to a packet header field, the order of the bytes needs to be defined.

Integer fields in the registers have a little endian byte order so that the lowest bits in a field will be at lowest bits on the configuration bus. When a field spans multiple configuration bus addresses the lowest address will hold the lowest bits of the field. If this is memory mapped and accessed by a host CPU it will be in the correct byte order for a little endian CPU.

In network byte order the first transmitted or received byte has byte number 0. One example is the Ethernet MAC address with the printed representation *a1-b2-c3-d4-e5-f6* where *a1* would be sent first and would be byte 0). When used in a register field the highest bits in the register field corresponds to the lowest network byte. Therefore the MAC address above would be the value *0xa1b2c3d4e5f6* and as seen by a little endian host CPU the byte *0xf6* would be at the lowest address.

A special case are IPv6 addresses. In the standard printed representation *0102:0304:0506:...* the leftmost byte *01* is byte 0 in the network order followed by byte *02* as network byte 1. When configuring this in a

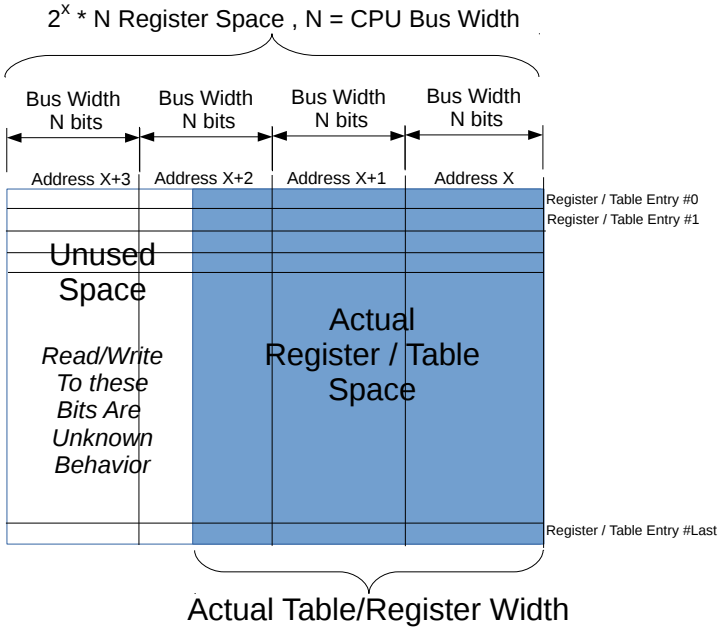


Figure 38.1: Address space usage by tables

register field the lowest bytes are from the lowest network byte numbers. However each pair of bytes are also swapped. The address above would therefore be the value *0x....050603040102*.

38.3 Register Banks

A bank is a hardware unit which holds a number of registers or a single table. In a bank containing data wider than 32 bits, registers (or table entries) must be accessed one at a time, or the accesses will interfere with each other.

Bank Name	Connected Registers or Tables
switch_info_regbank	Core Version
top_regs	Buffer Free Core Tick Configuration Core Tick Select Scratch
pa top switch mactop iRxedgecheck iProtocolcheck0 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck1 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck2 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck3 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck4 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck5 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck6 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck7 table	MAC Interface Counters For RX

Bank Name	Connected Registers or Tables
pa top switch mactop iRxedgecheck iProtocolcheck8 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck9 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck10 table	MAC Interface Counters For RX
pa top switch mactop iRxedgecheck iProtocolcheck11 table	MAC Interface Counters For RX
rx_length_ref	MAC RX Maximum Packet Length[0..11]
rx_length_drop	MAC RX Broken Packets[0..11] MAC RX Short Packet Drop[0..11] MAC RX Long Packet Drop[0..11]
pa top switch mactop iTxedgecheck iProtocolcheck0 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck1 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck2 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck3 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck4 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck5 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck6 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck7 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck8 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck9 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck10 table	MAC Interface Counters For TX
pa top switch mactop iTxedgecheck iProtocolcheck11 table	MAC Interface Counters For TX
l2_broadcast_storm_control_rate_settings	L2 Broadcast Storm Control Rate Configuration
l2_broadcast_storm_control_bucket_settings	L2 Broadcast Storm Control Bucket Capacity Configuration L2 Broadcast Storm Control Bucket Threshold Configuration
l2_broadcast_storm_control_misc	L2 Broadcast Storm Control Enable
l2_multicast_storm_control_rate_settings	L2 Multicast Storm Control Rate Configuration
l2_multicast_storm_control_bucket_settings	L2 Multicast Storm Control Bucket Capacity Configuration L2 Multicast Storm Control Bucket Threshold Configuration
l2_multicast_storm_control_misc	L2 Multicast Storm Control Enable
l2_flooding_storm_control_rate_settings	L2 Flooding Storm Control Rate Configuration
l2_flooding_storm_control_bucket_settings	L2 Flooding Storm Control Bucket Capacity Configuration L2 Flooding Storm Control Bucket Threshold Configuration
l2_flooding_storm_control_misc	L2 Flooding Storm Control Enable
le_ae_status	Learning Conflict Learning Overflow
le_ae_control	Learning And Aging Enable Software Aging Enable Learning And Aging Writeback Control Learning Data FIFO High Watermark Level Aging Data FIFO High Watermark Level

Bank Name	Connected Registers or Tables
	Hit Update Data FIFO High Watermark Level Hardware Learning Configuration[0..10] Time to Age
age_cam_register_bank	L2 Aging Collision Table[0..31]
mac_cnt_register_bank	Hardware Learning Counter[0..10]
L2 Aging Table	L2 Aging Table
le_ae_control_latch	Software Aging Start Latch
ldf	Learning Data FIFO
adf	Aging Data FIFO
hdf	Hit Update Data FIFO
count_sp_ss0	SP Overflow Drop
count_broken_pkt_ss0	IPP PM Drop IPP Empty Destination Drop
count_pa top switch ipp0 conf	Unknown Ingress Drop Empty Mask Drop Ingress Spanning Tree Drop: Listen Ingress Spanning Tree Drop: Learning Ingress Spanning Tree Drop: Blocking L2 Lookup Drop Ingress Table Not In Sync Drop Ingress Packet Filtering Drop Reserved MAC DA Drop Reserved MAC SA Drop VLAN Member Drop Minimum Allowed VLAN Drop Maximum Allowed VLAN Drop Invalid Routing Protocol Drop Expired TTL Drop L3 Lookup Drop IP Checksum Drop Second Tunnel Exit Drop Tunnel Exit Miss Action Drop Tunnel Exit Too Small Packet Modification Drop Learning Packet Drop L2 Decoder Packet Drop L3 Decoder Packet Drop L2 Reserved Multicast Address Drop Ingress Configurable ACL Drop Egress Configurable ACL Drop ARP Decoder Drop RARP Decoder Drop L2 IEEE 1588 Decoder Drop L4 IEEE 1588 Decoder Drop IEEE 802.1X and EAPOL Decoder Drop SCTP Decoder Drop LACP Decoder Drop AH Decoder Drop ESP Decoder Drop DNS Decoder Drop BOOTP and DHCP Decoder Drop CAPWAP Decoder Drop IKE Decoder Drop GRE Decoder Drop NAT Action Table Drop Crypto Drops

Bank Name	Connected Registers or Tables
	MACsec Drops L2 Action Table Special Packet Type Drop L2 Action Table Drop L2 Action Table Port Move Drop Source Port Default ACL Action Drop Ingress Functional Control Drops
count_opkt_pa top switch ipp0 conf	IPP Packet Head Counter IPP Packet Tail Counter
Tunnel Exit Lookup TCAM Answer	Tunnel Exit Lookup TCAM Answer
Ingress Admission Control Initial Pointer	Ingress Admission Control Initial Pointer
Ingress Configurable ACL 0 Pre Lookup	Ingress Configurable ACL 0 Pre Lookup
Ingress Configurable ACL 0 Large Table	Ingress Configurable ACL 0 Large Table
Ingress Configurable ACL 0 Small Table	Ingress Configurable ACL 0 Small Table
Ingress Configurable ACL 0 TCAM Answer	Ingress Configurable ACL 0 TCAM Answer
Ingress Configurable ACL 1 Pre Lookup	Ingress Configurable ACL 1 Pre Lookup
Ingress Configurable ACL 1 Large Table	Ingress Configurable ACL 1 Large Table
Ingress Configurable ACL 1 Small Table	Ingress Configurable ACL 1 Small Table
Ingress Configurable ACL 1 TCAM Answer	Ingress Configurable ACL 1 TCAM Answer
Ingress Configurable ACL 2 Pre Lookup	Ingress Configurable ACL 2 Pre Lookup
Ingress Configurable ACL 2 Large Table	Ingress Configurable ACL 2 Large Table
Ingress Configurable ACL 2 Small Table	Ingress Configurable ACL 2 Small Table
Ingress Configurable ACL 2 TCAM Answer	Ingress Configurable ACL 2 TCAM Answer
Source Port Default ACL Action	Source Port Default ACL Action
VLAN Table	VLAN Table
Ingress Multiple Spanning Tree State	Ingress Multiple Spanning Tree State
Ingress Router Table	Ingress Router Table
L3 LPM Result	L3 LPM Result
Hash Based L3 Routing Table	Hash Based L3 Routing Table
Next Hop Table	Next Hop Table
Next Hop Packet Modifications	Next Hop Packet Modifications
L2 Aging Status Shadow Table	L2 Aging Status Shadow Table
L2 DA Hash Lookup Table	L2 DA Hash Lookup Table
L2 Destination Table	L2 Destination Table
IPSec Table	IPSec Table
L2 SA Hash Lookup Table	L2 SA Hash Lookup Table
L2 Aging Status Shadow Table - Replica	L2 Aging Status Shadow Table - Replica
L2 Destination Table - Replica	L2 Destination Table - Replica
L2 Action Table	L2 Action Table
L2 Action Table Source Port	L2 Action Table Source Port
Egress ACL Rule Pointer Large Table	Egress ACL Rule Pointer Large Table
Egress ACL Rule Pointer Small Table	Egress ACL Rule Pointer Small Table
Egress ACL Rule Pointer TCAM Answer	Egress ACL Rule Pointer TCAM Answer
Egress Configurable ACL Large Table	Egress Configurable ACL Large Table
Egress Configurable ACL Small Table	Egress Configurable ACL Small Table
Egress Configurable ACL TCAM Answer	Egress Configurable ACL TCAM Answer
Tunnel Entry MTU Length Check	Tunnel Entry MTU Length Check
ipp_register_bank_ss0	Ingress Function Pointer Source Port Enable Enqueue To Ports And Queues Flooding Action Send to Port Link Aggregation To Physical Ports Members Link Aggregate Weight

Bank Name	Connected Registers or Tables
	Ingress Egress Port Packet Type Filter NAT Action Table Egress Configurable ACL Rules Setup Allow Special Frame Check For L2 Action Table Egress Multiple Spanning Tree State Router MTU Table Hairpin Enable L2 Aging Collision Shadow Table Router Egress Queue To VLAN Data MPLS EXP Field To Packet Color Mapping Table IPv6 Class of Service Field To Packet Color Mapping Table IPv4 TOS Field To Packet Color Mapping Table VLAN PCP And DEI To Color Mapping Table MPLS EXP Field To Egress Queue Mapping Table IPv6 Class of Service Field To Egress Queue Mapping Table IPv4 TOS Field To Egress Queue Mapping Table VLAN PCP To Queue Mapping Table L3 Routing Default Ingress VID Ethernet Type Range Assignment Answer Ingress VID Inner VID Range Assignment Answer Ingress VID Outer VID Range Assignment Answer Ingress VID MAC Range Assignment Answer Ingress Configurable ACL 0 Rules Setup Ingress Port Packet Type Filter SMON Set Search Default Packet To CPU Modification L2 Reserved Multicast Address Action Second Tunnel Exit Miss Action Link Aggregation Membership Link Aggregation Ctrl Debug Counter srcPort Setup SNAP LLC Decoding Options Ingress Ethernet Type for VLAN tag Debug Counter nrVlans Setup SCTP Packet Decoder Options AH Header Packet Decoder Options ESP Header Packet Decoder Options Debug Counter spVidOp Setup Ingress Configurable ACL 0 Selection Ingress Configurable ACL 1 Selection Ingress Configurable ACL 2 Selection Debug Counter finalVid Setup Debug Counter vlanVidOp Setup Debug Counter nextHopPtrLpm Setup Debug Counter nextHopPtrHash Setup Debug Counter nextHopPtrFinal Setup Check IPv4 Header Checksum Force Non VLAN Packet To Specific Queue Force Unknown L3 Packet To Specific Egress Queue Force Non VLAN Packet To Specific Color Force Unknown L3 Packet To Specific Color Debug Counter l2DaHash Setup Debug Counter l2DaHashHitAndBucket Setup Forward From CPU Port Move Options L2 Action Table Egress Port State

Bank Name	Connected Registers or Tables
	L2 Multicast Handling Egress Configurable ACL Selection Egress Port NAT State NAT Action Table Force Original Packet MACsec Port Ingress MMP Drop Mask Debug Counter dstPortmask Setup IPP Debug srcPort IPP Debug dropPktAfterL2Decode IPP Debug nrVlans IPP Debug dropPktAfterL3Decode IPP Debug spVidOp IPP Debug finalVid IPP Debug vlanVidOp IPP Debug routerHit IPP Debug nextHopPtrLpm IPP Debug nextHopPtrHash IPP Debug nextHopPtrLpmHit IPP Debug nextHopPtrHashHit IPP Debug nextHopPtrFinal IPP Debug l2DaHash IPP Debug l2DaHashHitAndBucket IPP Debug l2DaTcamHitsAndCast IPP Debug routed IPP Debug isFlooding IPP Debug isBroadcast IPP Debug doL2Lookup IPP Debug dstPortmask IPP Debug debugMatchIPP0 Ingress Function Control CPU Reason Code Operation L2 Multicast Table L2 Lookup Collision Table Masks L2 Lookup Collision Table Ingress VID Ethernet Type Range Search Data Ingress VID Inner VID Range Search Data Ingress VID Outer VID Range Search Data Ingress Configurable ACL 2 Rules Setup Ingress Configurable ACL 1 Rules Setup Second Tunnel Exit Lookup TCAM Answer Ingress Function Control Packet From CPU Port Ingress Function Control Packet From Crypto Engine Decrypted Ingress Function Control Packet From Crypto Engine Encrypted L2 Tunnel Decoder Setup Learning DA MAC Ingress Function Control Packet From CPU Tag Ingress Function Control Packet From CPU Tag Do Not Modify L2 Reserved Multicast Address Base ARP Packet Decoder Options RARP Packet Decoder Options IEEE 1588 L2 Packet Decoder Options IEEE 802.1X and EAPOL Packet Decoder Options GRE Packet Decoder Options

Bank Name	Connected Registers or Tables
	DNS Packet Decoder Options BOOTP and DHCP Packet Decoder Options CAPWAP Packet Decoder Options IKE Packet Decoder Options Debug Counter l2DaTcamHitsAndCast Setup Egress Spanning Tree State Ingress Function Control Packet To Crypto Engine Debug Counter debugMatchIPP0 Setup IPP Debug l2DaHashKey Source Port Table Egress ACL Rule Pointer TCAM Ingress VID MAC Range Search Data Reserved Source MAC Address Range Reserved Destination MAC Address Range Send to CPU LACP Packet Decoder Options Debug Counter l2DaHashKey Setup Egress ACL Rule Pointer Search Mask Tunnel Exit Lookup TCAM Ingress Configurable ACL 0 TCAM IEEE 1588 L4 Packet Decoder Options Ingress Configurable ACL 0 Search Mask Second Tunnel Exit Lookup TCAM Egress Configurable ACL TCAM Ingress Configurable ACL 1 TCAM Ingress Configurable ACL 1 Search Mask Egress Configurable ACL Search Mask LLDP Configuration Router Port MAC Address Ingress Configurable ACL 2 Search Mask Ingress Configurable ACL 2 TCAM
ipp_register_bank_misc_ss0	Ingress Drop Options
L3 Routing TCAM	L3 Routing TCAM
count_packets ipp0_smonStatisticsBlock	SMON Set 0 Packet Counter[0..7] SMON Set 1 Packet Counter[0..7] SMON Set 2 Packet Counter[0..7] SMON Set 3 Packet Counter[0..7] SMON Set 4 Packet Counter[0..7] SMON Set 5 Packet Counter[0..7] SMON Set 6 Packet Counter[0..7] SMON Set 7 Packet Counter[0..7]
count_bytes ipp0_smonStatisticsBlock	SMON Set 0 Byte Counter[0..7] SMON Set 1 Byte Counter[0..7] SMON Set 2 Byte Counter[0..7] SMON Set 3 Byte Counter[0..7] SMON Set 4 Byte Counter[0..7] SMON Set 5 Byte Counter[0..7] SMON Set 6 Byte Counter[0..7] SMON Set 7 Byte Counter[0..7]
count_ipp0_aclConfStatisticsBlock	Ingress Configurable ACL Match Counter[0..63]
count_ipp0_vrfInStatisticsBlock	Received Packets on Ingress VRF[0..3]
Next Hop Hit Status	Next Hop Hit Status
count_ipp0_egressAclStatisticsBlock	Egress Configurable ACL Match Counter[0..63]
count_ipp0_egressDropStatisticsBlock	Queue Off Drop[0..10] Egress Spanning Tree Drop[0..10]

Bank Name	Connected Registers or Tables
	MBSC Drop[0..10] Ingress-Egress Packet Filtering Drop[0..10] L2 Action Table Per Port Drop[0..10]
count_ucipp0_igrPortMibBlock	IP Unicast Received Counter[0..10]
count_mcipp0_igrPortMibBlock	IP Multicast Received Counter[0..10]
count_uc_routedipp0_igrPortMibBlock	IP Unicast Routed Counter[0..10]
count_mc_routedipp0_igrPortMibBlock	IP Multicast Routed Counter[0..10]
count_mc_acl_dropipp0_igrPortMibBlock	IP Multicast ACL Drop Counter[0..10]
count_ipp0_debugIppStatisticsBlock	Debug IPP Counter[0..22]
bk_mmp_stat_0	Flow Classification And Metering Drop
bk_ingress_admission_control_all_red_en_0	Ingress Admission Control Mark All Red Enable
bk_ingress_admission_control_all_red_0	Ingress Admission Control Mark All Red
Ingress Admission Control Token Bucket Configuration	Ingress Admission Control Token Bucket Configuration
Ingress Admission Control Reset	Ingress Admission Control Reset
Ingress Admission Control Current Status	Ingress Admission Control Current Status
bk_erm_ss0	ERM Yellow Configuration Resource Limiter Set[0..3] ERM Red Configuration Egress Resource Manager Pointer[0..11]
count_erm_ss0	Egress Resource Manager Drop[0..11]
pb_info_regbank_ss0	Packet Buffer Status
count_drop_pa top switch pb0	Buffer Overflow Drop Ingress Resource Manager Drop
pb_queue_manage_register_bank_ss0	Map Queue to Priority[0..11]
count_drop_pa top switch pb0 iRequeue	Re-queue Overflow Drop
pfc_regbank_rsv_size_ss0	Port/TC Reserved[0..95]
pfc_regbank_port_rsv_size_ss0	Port Reserved[0..11]
PFC Inc Counters for ingress ports 0 to 11	PFC Inc Counters for ingress ports 0 to 11
PFC Dec Counters for ingress ports 0 to 11	PFC Dec Counters for ingress ports 0 to 11
pfc_regbank_cmn_misc_ss0	Port FFA Used[0..11] Port Used[0..11] TC FFA Used[0..7] FFA Used PFC FFA Used non-PFC
pfc_regbank_pause_settings1_ss0	Port Pause Settings[0..11]
pfc_regbank_taildrop_settings0_ss0	Port Tail-Drop Settings[0..11]
pfc_regbank_misc_ss0	Xon FFA Threshold Xoff FFA Threshold Tail-Drop FFA Threshold TC Xon FFA Threshold[0..7] TC Xoff FFA Threshold[0..7] TC Tail-Drop FFA Threshold[0..7] Port Xon FFA Threshold[0..11] Port Xoff FFA Threshold[0..11] Port Tail-Drop FFA Threshold[0..11] Port/TC Xon Total Threshold[0..95] Port/TC Xoff Total Threshold[0..95] Port/TC Tail-Drop Total Threshold[0..95]
qe_register_bank_ss0_sp0	Egress Port Depth[0..11] Egress Queue Depth[0..95]
pb_r_register_bank_ss0	Minimum Buffer Free

Bank Name	Connected Registers or Tables
disable_queue_output_register_bank_ss0	Output Disable[0..11]
dwrr_bucket_capacity_settings_ss0	DWRR Bucket Capacity Configuration[0..11]
dwrr_bucket_misc_settings_ss0	DWRR Bucket Misc Configuration[0..11]
dwrr_weight_settings_ss0	DWRR Weight Configuration[0..95]
queue_shaper_rate_settings	Queue Shaper Rate Configuration
queue_shaper_bucket_settings	Queue Shaper Bucket Capacity Configuration Queue Shaper Bucket Threshold Configuration
queue_shaper_misc	Queue Shaper Enable
prio_shaper_rate_settings	Prio Shaper Rate Configuration
prio_shaper_bucket_settings	Prio Shaper Bucket Capacity Configuration Prio Shaper Bucket Threshold Configuration
prio_shaper_misc	Prio Shaper Enable
port_shaper_rate_settings	Port Shaper Rate Configuration
port_shaper_bucket_settings	Port Shaper Bucket Capacity Configuration Port Shaper Bucket Threshold Configuration
port_shaper_misc	Port Shaper Enable
count_opkt_pa top switch pb0	PB Packet Head Counter PB Packet Tail Counter
drain_port_ss0	Drain Port
drain_drop_ss0	Drain Port Drop[0..11]
count_pa top switch epp0 conf	Unknown Egress Drop[0..11] Egress Port Disabled Drop[0..11] Egress Port Filtering Drop[0..11] Egress Table Not In Sync Drop[0..11] Tunnel Exit Too Small Packet Modification To Small Drop[0..11] Minimum and Maximum Packet Size Drops[0..11] Egress Functional Control Drops[0..11] Egress Cell Size Drop[0..11] EPP PM Drop
count_opkt_pa top switch epp0 conf	EPP Packet Head Counter EPP Packet Tail Counter
Egress Port Configuration	Egress Port Configuration
Egress Tunnel Exit Table	Egress Tunnel Exit Table
Tunnel Entry Instruction Table	Tunnel Entry Instruction Table
Tunnel Entry Header Data	Tunnel Entry Header Data
Beginning of Packet Tunnel Entry Instruction Table	Beginning of Packet Tunnel Entry Instruction Table
L2 Tunnel Entry Instruction Table	L2 Tunnel Entry Instruction Table
L3 Tunnel Entry Instruction Table	L3 Tunnel Entry Instruction Table
Color Remap From Egress Port	Color Remap From Egress Port
Color Remap From Ingress Admission Control	Color Remap From Ingress Admission Control
Egress Router Table	Egress Router Table
Next Hop DA MAC	Next Hop DA MAC
Router Port Egress SA MAC Address	Router Port Egress SA MAC Address
Router MAC SA Table	Router MAC SA Table
Next Hop MPLS Table	Next Hop MPLS Table
Egress MPLS TTL Table	Egress MPLS TTL Table
Next Hop Packet Insert MPLS Header	Next Hop Packet Insert MPLS Header
Egress Queue To PCP And CFI/DEI Mapping Table	Egress Queue To PCP And CFI/DEI Mapping Table
Egress VLAN Translation Large Table	Egress VLAN Translation Large Table
Egress VLAN Translation Small Table	Egress VLAN Translation Small Table

Bank Name	Connected Registers or Tables
Egress VLAN Translation TCAM Answer	Egress VLAN Translation TCAM Answer
Ingress NAT Operation	Ingress NAT Operation
Egress NAT Operation	Egress NAT Operation
Select Which Egress QoS Mapping Table To Use	Select Which Egress QoS Mapping Table To Use
L2 QoS Mapping Table	L2 QoS Mapping Table
IP QoS Mapping Table	IP QoS Mapping Table
TOS QoS Mapping Table	TOS QoS Mapping Table
MPLS QoS Mapping Table	MPLS QoS Mapping Table
epp_register_bank_ss0	Output Mirroring Table Egress Queue To MPLS EXP Mapping Table Egress Function Control Egress Function Pointer Egress Port Debug Counter reQueuePortId Setup Egress Function Control Packet From Crypto Engine Decrypted Egress Function Control Packet From Crypto Engine Encrypted Egress Function Control Packet From CPU Port Egress Function Control Packet To CPU Port with Reason Zero Egress Function Control Packet To CPU Port Egress Function Control Packet From CPU Tag Egress Function Control Packet From CPU Tag Do Not Modify Egress Function Control Packet To Crypto Engine Debug Counter fromPort Setup Egress MPLS Decoding Options Egress Ethernet Type for VLAN tag Egress VLAN Translation Selection NAT Add Egress Port for NAT Calculation Disable CPU tag on CPU Port MACsec Vlan Debug Counter debugMatchEPP0 Setup EPP Debug imActive EPP Debug imExtra EPP Debug omEnabled EPP Debug omImActive EPP Debug reQueue EPP Debug reQueuePortId EPP Debug reQueuePkt EPP Debug fromPort EPP Debug delSpecificVlan EPP Debug updateTosExp EPP Debug isIPv4 EPP Debug isIPv6 EPP Debug addNewMpls EPP Debug isPPPoE EPP Debug debugMatchEPP0 Egress Port VID Operation Egress VLAN Translation Search Mask Egress VLAN Translation TCAM
count_epp0_vrfOutStatisticsBlock	Transmitted Packets on Egress VRF[0..3]
Ingress NAT Hit Status	Ingress NAT Hit Status
Egress NAT Hit Status	Egress NAT Hit Status

Bank Name	Connected Registers or Tables
count_epp0_debugEppStatisticsBlock	Debug EPP Counter[0..14]
count_opkt_pa top switch ps0 ps_wrap_bridge	PS Packet Head Counter PS Packet Tail Counter
count_error_pa top switch ps0 ps_wrap_bridge	PS Error Counter
Security Association Table	Security Association Table
Crypto Sequence Numbers	Crypto Sequence Numbers
Crypto Configuration	Crypto Configuration Linear Feedback Shift Register

38.4 Registers and Tables in Alphabetical Order

Name	Address Range
AH Decoder Drop	16899
AH Header Packet Decoder Options	1121221
ARP Decoder Drop	16892
ARP Packet Decoder Options	1122483
Aging Data FIFO	16769
Aging Data FIFO High Watermark Level	322
Allow Special Frame Check For L2 Action Table	1119455 - 1119458
BOOTP and DHCP Decoder Drop	16902
BOOTP and DHCP Packet Decoder Options	1122495
Beginning of Packet Tunnel Entry Instruction Table	1131448 - 1131479
Buffer Free	1
Buffer Overflow Drop	1129076
CAPWAP Decoder Drop	16903
CAPWAP Packet Decoder Options	1122497
CPU Reason Code Operation	1121275 - 1121306
Check IPv4 Header Checksum	1121232
Color Remap From Egress Port	1131544 - 1131565
Color Remap From Ingress Admission Control	1131566 - 1131693
Core Tick Configuration	2
Core Tick Select	3
Core Version	0
Crypto Configuration	1201248
Crypto Drops	16907
Crypto Sequence Numbers	1200736 - 1201247
DNS Decoder Drop	16901
DNS Packet Decoder Options	1122493
DWRR Bucket Capacity Configuration	1129921 - 1129932
DWRR Bucket Misc Configuration	1129933 - 1129944
DWRR Weight Configuration	1129945 - 1130040
Debug Counter debugMatchEPP0 Setup	1182086
Debug Counter debugMatchIPP0 Setup	1122507
Debug Counter dstPortmask Setup	1121248
Debug Counter finalVid Setup	1121227
Debug Counter fromPort Setup	1182079
Debug Counter l2DaHash Setup	1121237

Name	Address Range
Debug Counter I2DaHashHitAndBucket Setup	1121238
Debug Counter I2DaHashKey Setup	1122675
Debug Counter I2DaTcamHitsAndCast Setup	1122501
Debug Counter nextHopPtrFinal Setup	1121231
Debug Counter nextHopPtrHash Setup	1121230
Debug Counter nextHopPtrLpm Setup	1121229
Debug Counter nrVlans Setup	1121219
Debug Counter reQueuePortId Setup	1182070
Debug Counter spVidOp Setup	1121223
Debug Counter srcPort Setup	1121216
Debug Counter vlanVidOp Setup	1121228
Debug EPP Counter	1198588 - 1198602
Debug IPP Counter	1128470 - 1128492
Default Packet To CPU Modification	1120681 - 1120691
Disable CPU tag on CPU Port	1182084
Drain Port	1130706
Drain Port Drop	1130707 - 1130718
EPP Debug addNewMpls	1182099
EPP Debug debugMatchEPP0	1182101
EPP Debug delSpecificVlan	1182095
EPP Debug fromPort	1182094
EPP Debug imActive	1182087
EPP Debug imExtra	1182088
EPP Debug isIPv4	1182097
EPP Debug isIPv6	1182098
EPP Debug isPPPoE	1182100
EPP Debug omEnabled	1182089
EPP Debug omImActive	1182090
EPP Debug reQueue	1182091
EPP Debug reQueuePkt	1182093
EPP Debug reQueuePortId	1182092
EPP Debug updateTosExp	1182096
EPP PM Drop	1130815
EPP Packet Head Counter	1130816
EPP Packet Tail Counter	1130817
ERM Red Configuration	1129050
ERM Yellow Configuration	1129040
ESP Decoder Drop	16900
ESP Header Packet Decoder Options	1121222
Egress ACL Rule Pointer Large Table	1044400 - 1044655
Egress ACL Rule Pointer Search Mask	1122679
Egress ACL Rule Pointer Small Table	1044656 - 1044783
Egress ACL Rule Pointer TCAM	1122555 - 1122618
Egress ACL Rule Pointer TCAM Answer	1044784 - 1044799
Egress Cell Size Drop	1130803 - 1130814
Egress Configurable ACL Drop	16891
Egress Configurable ACL Large Table	1044800 - 1110335
Egress Configurable ACL Match Counter	1128296 - 1128359
Egress Configurable ACL Rules Setup	1119447 - 1119454
Egress Configurable ACL Search Mask	1124299
Egress Configurable ACL Selection	1121243
Egress Configurable ACL Small Table	1110336 - 1118527
Egress Configurable ACL TCAM	1123899 - 1124154

Name	Address Range
Egress Configurable ACL TCAM Answer	1118528 - 1118591
Egress Ethernet Type for VLAN tag	1182081
Egress Function Control	1182057 - 1182058
Egress Function Control Packet From CPU Port	1182073
Egress Function Control Packet From CPU Tag	1182076
Egress Function Control Packet From CPU Tag Do Not Modify	1182077
Egress Function Control Packet From Crypto Engine Decrypted	1182071
Egress Function Control Packet From Crypto Engine Encrypted	1182072
Egress Function Control Packet To CPU Port	1182075
Egress Function Control Packet To CPU Port with Reason Zero	1182074
Egress Function Control Packet To Crypto Engine	1182078
Egress Function Pointer Egress Port	1182059 - 1182069
Egress Functional Control Drops	1130791 - 1130802
Egress MPLS Decoding Options	1182080
Egress MPLS TTL Table	1137914 - 1137917
Egress Multiple Spanning Tree State	1119459 - 1119474
Egress NAT Hit Status	1190396 - 1198587
Egress NAT Operation	1164054 - 1180437
Egress Port Configuration	1130818 - 1130839
Egress Port Depth	1129800 - 1129811
Egress Port Disabled Drop	1130731 - 1130742
Egress Port Filtering Drop	1130743 - 1130754
Egress Port NAT State	1121244
Egress Port VID Operation	1182102 - 1182165
Egress Queue Depth	1129812 - 1129907
Egress Queue To MPLS EXP Mapping Table	1182049 - 1182056
Egress Queue To PCP And CFI/DEI Mapping Table	1146110 - 1146117
Egress Resource Manager Drop	1129063 - 1129074
Egress Resource Manager Pointer	1129051 - 1129062
Egress Router Table	1131694 - 1131697
Egress Spanning Tree Drop	1128371 - 1128381
Egress Spanning Tree State	1122503
Egress Table Not In Sync Drop	1130755 - 1130766
Egress Tunnel Exit Table	1130840 - 1130871
Egress VLAN Translation Large Table	1146118 - 1147141
Egress VLAN Translation Search Mask	1182166
Egress VLAN Translation Selection	1182082
Egress VLAN Translation Small Table	1147142 - 1147653
Egress VLAN Translation TCAM	1182168 - 1182199
Egress VLAN Translation TCAM Answer	1147654 - 1147669
Empty Mask Drop	16867
Enable Enqueue To Ports And Queues	1118635 - 1118645
Expired TTL Drop	16880
FFA Used PFC	1129423
FFA Used non-PFC	1129424
Flooding Action Send to Port	1118646 - 1118656
Flow Classification And Metering Drop	1128493
Force Non VLAN Packet To Specific Color	1121235
Force Non VLAN Packet To Specific Queue	1121233
Force Unknown L3 Packet To Specific Color	1121236
Force Unknown L3 Packet To Specific Egress Queue	1121234
Forward From CPU	1121239
GRE Decoder Drop	16905

Name	Address Range
GRE Packet Decoder Options	1122491
Hairpin Enable	1119519 - 1119529
Hardware Learning Configuration	324 - 334
Hardware Learning Counter	369 - 379
Hash Based L3 Routing Table	347632 - 871919
Hit Update Data FIFO	16771
Hit Update Data FIFO High Watermark Level	323
IEEE 1588 L2 Packet Decoder Options	1122487
IEEE 1588 L4 Packet Decoder Options	1123707
IEEE 802.1X and EAPOL Decoder Drop	16896
IEEE 802.1X and EAPOL Packet Decoder Options	1122489
IKE Decoder Drop	16904
IKE Packet Decoder Options	1122499
IP Checksum Drop	16882
IP Multicast ACL Drop Counter	1128459 - 1128469
IP Multicast Received Counter	1128426 - 1128436
IP Multicast Routed Counter	1128448 - 1128458
IP QoS Mapping Table	1180758 - 1181013
IP Unicast Received Counter	1128415 - 1128425
IP Unicast Routed Counter	1128437 - 1128447
IPP Debug debugMatchIPP0	1121270
IPP Debug doL2Lookup	1121268
IPP Debug dropPktAfterL2Decode	1121250
IPP Debug dropPktAfterL3Decode	1121252
IPP Debug dstPortmask	1121269
IPP Debug finalVid	1121254
IPP Debug isBroadcast	1121267
IPP Debug isFlooding	1121266
IPP Debug l2DaHash	1121262
IPP Debug l2DaHashHitAndBucket	1121263
IPP Debug l2DaHashKey	1122509
IPP Debug l2DaTcamHitsAndCast	1121264
IPP Debug nextHopPtrFinal	1121261
IPP Debug nextHopPtrHash	1121258
IPP Debug nextHopPtrHashHit	1121260
IPP Debug nextHopPtrLpm	1121257
IPP Debug nextHopPtrLpmHit	1121259
IPP Debug nrVlans	1121251
IPP Debug routed	1121265
IPP Debug routerHit	1121256
IPP Debug spVidOp	1121253
IPP Debug srcPort	1121249
IPP Debug vlanVidOp	1121255
IPP Empty Destination Drop	16865
IPP PM Drop	16864
IPP Packet Head Counter	16914
IPP Packet Tail Counter	16915
IPSec Table	962096 - 962159
IPv4 TOS Field To Egress Queue Mapping Table	1120370 - 1120625
IPv4 TOS Field To Packet Color Mapping Table	1119834 - 1120089
IPv6 Class of Service Field To Egress Queue Mapping Table	1120114 - 1120369
IPv6 Class of Service Field To Packet Color Mapping Table	1119578 - 1119833
Ingress Admission Control Current Status	1128942 - 1129005

Name	Address Range
Ingress Admission Control Initial Pointer	17172 - 17299
Ingress Admission Control Mark All Red	1128558 - 1128621
Ingress Admission Control Mark All Red Enable	1128494 - 1128557
Ingress Admission Control Reset	1128878 - 1128941
Ingress Admission Control Token Bucket Configuration	1128622 - 1128877
Ingress Configurable ACL 0 Large Table	17316 - 279459
Ingress Configurable ACL 0 Pre Lookup	17300 - 17315
Ingress Configurable ACL 0 Rules Setup	1120654 - 1120661
Ingress Configurable ACL 0 Search Mask	1123739
Ingress Configurable ACL 0 Selection	1121224
Ingress Configurable ACL 0 Small Table	279460 - 312227
Ingress Configurable ACL 0 TCAM	1123195 - 1123706
Ingress Configurable ACL 0 TCAM Answer	312228 - 312355
Ingress Configurable ACL 1 Large Table	313380 - 317475
Ingress Configurable ACL 1 Pre Lookup	312356 - 313379
Ingress Configurable ACL 1 Rules Setup	1122435 - 1122450
Ingress Configurable ACL 1 Search Mask	1124283
Ingress Configurable ACL 1 Selection	1121225
Ingress Configurable ACL 1 Small Table	317476 - 325667
Ingress Configurable ACL 1 TCAM	1124155 - 1124282
Ingress Configurable ACL 1 TCAM Answer	325668 - 325731
Ingress Configurable ACL 2 Large Table	325860 - 329955
Ingress Configurable ACL 2 Pre Lookup	325732 - 325859
Ingress Configurable ACL 2 Rules Setup	1122427 - 1122434
Ingress Configurable ACL 2 Search Mask	1124451
Ingress Configurable ACL 2 Selection	1121226
Ingress Configurable ACL 2 Small Table	329956 - 330979
Ingress Configurable ACL 2 TCAM	1124515 - 1125538
Ingress Configurable ACL 2 TCAM Answer	330980 - 331107
Ingress Configurable ACL Drop	16890
Ingress Configurable ACL Match Counter	1126180 - 1126243
Ingress Drop Options	1125539
Ingress Egress Port Packet Type Filter	1118924 - 1118934
Ingress Ethernet Type for VLAN tag	1121218
Ingress Function Control	1121271 - 1121274
Ingress Function Control Packet From CPU Port	1122467
Ingress Function Control Packet From CPU Tag	1122477
Ingress Function Control Packet From CPU Tag Do Not Modify	1122479
Ingress Function Control Packet From Crypto Engine Decrypted	1122469
Ingress Function Control Packet From Crypto Engine Encrypted	1122471
Ingress Function Control Packet To Crypto Engine	1122505
Ingress Function Pointer Source Port	1118624 - 1118634
Ingress Functional Control Drops	16913
Ingress MMP Drop Mask	1121247
Ingress Multiple Spanning Tree State	347580 - 347595
Ingress NAT Hit Status	1182204 - 1190395
Ingress NAT Operation	1147670 - 1164053
Ingress Packet Filtering Drop	16873
Ingress Port Packet Type Filter	1120662 - 1120672
Ingress Resource Manager Drop	1129077
Ingress Router Table	347596 - 347599

Name	Address Range
Ingress Spanning Tree Drop: Blocking	16870
Ingress Spanning Tree Drop: Learning	16869
Ingress Spanning Tree Drop: Listen	16868
Ingress Table Not In Sync Drop	16872
Ingress VID Ethernet Type Range Assignment Answer	1120638 - 1120641
Ingress VID Ethernet Type Range Search Data	1122403 - 1122410
Ingress VID Inner VID Range Assignment Answer	1120642 - 1120645
Ingress VID Inner VID Range Search Data	1122411 - 1122418
Ingress VID MAC Range Assignment Answer	1120650 - 1120653
Ingress VID MAC Range Search Data	1122619 - 1122634
Ingress VID Outer VID Range Assignment Answer	1120646 - 1120649
Ingress VID Outer VID Range Search Data	1122419 - 1122426
Ingress-Egress Packet Filtering Drop	1128393 - 1128403
Invalid Routing Protocol Drop	16879
L2 Action Table	1044144 - 1044271
L2 Action Table Drop	16910
L2 Action Table Egress Port State	1121241
L2 Action Table Per Port Drop	1128404 - 1128414
L2 Action Table Port Move Drop	16911
L2 Action Table Source Port	1044272 - 1044399
L2 Action Table Special Packet Type Drop	16909
L2 Aging Collision Shadow Table	1119530 - 1119561
L2 Aging Collision Table	337 - 368
L2 Aging Status Shadow Table	880112 - 896495
L2 Aging Status Shadow Table - Replica	994928 - 1011311
L2 Aging Table	380 - 16763
L2 Broadcast Storm Control Bucket Capacity Configuration	219 - 229
L2 Broadcast Storm Control Bucket Threshold Configuration	230 - 240
L2 Broadcast Storm Control Enable	241
L2 Broadcast Storm Control Rate Configuration	208 - 218
L2 DA Hash Lookup Table	896496 - 929263
L2 Decoder Packet Drop	16887
L2 Destination Table	929264 - 962095
L2 Destination Table - Replica	1011312 - 1044143
L2 Flooding Storm Control Bucket Capacity Configuration	287 - 297
L2 Flooding Storm Control Bucket Threshold Configuration	298 - 308
L2 Flooding Storm Control Enable	309
L2 Flooding Storm Control Rate Configuration	276 - 286
L2 IEEE 1588 Decoder Drop	16894
L2 Lookup Collision Table	1122339 - 1122402
L2 Lookup Collision Table Masks	1122331 - 1122338
L2 Lookup Drop	16871
L2 Multicast Handling	1121242
L2 Multicast Storm Control Bucket Capacity Configuration	253 - 263
L2 Multicast Storm Control Bucket Threshold Configuration	264 - 274
L2 Multicast Storm Control Enable	275
L2 Multicast Storm Control Rate Configuration	242 - 252
L2 Multicast Table	1121307 - 1122330
L2 QoS Mapping Table	1180694 - 1180757
L2 Reserved Multicast Address Action	1120692 - 1120947
L2 Reserved Multicast Address Base	1122481
L2 Reserved Multicast Address Drop	16889
L2 SA Hash Lookup Table	962160 - 994927

Name	Address Range
L2 Tunnel Decoder Setup	1122473
L2 Tunnel Entry Instruction Table	1131480 - 1131511
L3 Decoder Packet Drop	16888
L3 LPM Result	347600 - 347631
L3 Lookup Drop	16881
L3 Routing Default	1120634 - 1120637
L3 Routing TCAM	1125540 - 1126051
L3 Tunnel Entry Instruction Table	1131512 - 1131543
L4 IEEE 1588 Decoder Drop	16895
LACP Decoder Drop	16898
LACP Packet Decoder Options	1122671
LLDP Configuration	1124315
Learning And Aging Enable	318
Learning And Aging Writeback Control	320
Learning Conflict	310
Learning DA MAC	1122475
Learning Data FIFO	16765
Learning Data FIFO High Watermark Level	321
Learning Overflow	314
Learning Packet Drop	16886
Linear Feedback Shift Register	1201249
Link Aggregate Weight	1118668 - 1118923
Link Aggregation Ctrl	1121215
Link Aggregation Membership	1121204 - 1121214
Link Aggregation To Physical Ports Members	1118657 - 1118667
MAC Interface Counters For RX	48 - 71
MAC Interface Counters For TX	120 - 167
MAC RX Broken Packets	84 - 95
MAC RX Long Packet Drop	108 - 119
MAC RX Maximum Packet Length	72 - 83
MAC RX Short Packet Drop	96 - 107
MACsec Drops	16908
MACsec Port	1121246
MACsec Vlan	1182085
MBSC Drop	1128382 - 1128392
MPLS EXP Field To Egress Queue Mapping Table	1120106 - 1120113
MPLS EXP Field To Packet Color Mapping Table	1119570 - 1119577
MPLS QoS Mapping Table	1181526 - 1182037
Map Queue to Priority	1129078 - 1129089
Maximum Allowed VLAN Drop	16878
Minimum Allowed VLAN Drop	16877
Minimum Buffer Free	1129908
Minimum and Maximum Packet Size Drops	1130779 - 1130790
NAT Action Table	1118935 - 1119446
NAT Action Table Drop	16906
NAT Action Table Force Original Packet	1121245
NAT Add Egress Port for NAT Calculation	1182083
Next Hop DA MAC	1131698 - 1135793
Next Hop Hit Status	1126248 - 1128295
Next Hop MPLS Table	1135866 - 1137913
Next Hop Packet Insert MPLS Header	1137918 - 1146109
Next Hop Packet Modifications	876016 - 880111
Next Hop Table	871920 - 876015

Name	Address Range
Output Disable	1129909 - 1129920
Output Mirroring Table	1182038 - 1182048
PB Packet Head Counter	1130704
PB Packet Tail Counter	1130705
PFC Dec Counters for ingress ports 0 to 11	1129295 - 1129390
PFC Inc Counters for ingress ports 0 to 11	1129199 - 1129294
PS Error Counter	1198642 - 1198653
PS Packet Head Counter	1198640
PS Packet Tail Counter	1198641
Packet Buffer Status	1129075
Port FFA Used	1129391 - 1129402
Port Move Options	1121240
Port Pause Settings	1129425 - 1129436
Port Reserved	1129187 - 1129198
Port Shaper Bucket Capacity Configuration	1130637 - 1130648
Port Shaper Bucket Threshold Configuration	1130649 - 1130660
Port Shaper Enable	1130661
Port Shaper Rate Configuration	1130625 - 1130636
Port Tail-Drop FFA Threshold	1129500 - 1129511
Port Tail-Drop Settings	1129437 - 1129448
Port Used	1129403 - 1129414
Port Xoff FFA Threshold	1129488 - 1129499
Port Xon FFA Threshold	1129476 - 1129487
Port/TC Reserved	1129091 - 1129186
Port/TC Tail-Drop Total Threshold	1129704 - 1129799
Port/TC Xoff Total Threshold	1129608 - 1129703
Port/TC Xon Total Threshold	1129512 - 1129607
Prio Shaper Bucket Capacity Configuration	1130429 - 1130524
Prio Shaper Bucket Threshold Configuration	1130525 - 1130620
Prio Shaper Enable	1130621
Prio Shaper Rate Configuration	1130333 - 1130428
Queue Off Drop	1128360 - 1128370
Queue Shaper Bucket Capacity Configuration	1130137 - 1130232
Queue Shaper Bucket Threshold Configuration	1130233 - 1130328
Queue Shaper Enable	1130329
Queue Shaper Rate Configuration	1130041 - 1130136
RARP Decoder Drop	16893
RARP Packet Decoder Options	1122485
Re-queue Overflow Drop	1129090
Received Packets on Ingress VRF	1126244 - 1126247
Reserved Destination MAC Address Range	1122651 - 1122666
Reserved MAC DA Drop	16874
Reserved MAC SA Drop	16875
Reserved Source MAC Address Range	1122635 - 1122650
Resource Limiter Set	1129042 - 1129049
Router Egress Queue To VLAN Data	1119562 - 1119569
Router MAC SA Table	1135802 - 1135865
Router MTU Table	1119475 - 1119518
Router Port Egress SA MAC Address	1135794 - 1135801
Router Port MAC Address	1124323 - 1124450
SCTP Decoder Drop	16897
SCTP Packet Decoder Options	1121220
SMON Set 0 Byte Counter	1126116 - 1126123

Name	Address Range
SMON Set 0 Packet Counter	1126052 - 1126059
SMON Set 1 Byte Counter	1126124 - 1126131
SMON Set 1 Packet Counter	1126060 - 1126067
SMON Set 2 Byte Counter	1126132 - 1126139
SMON Set 2 Packet Counter	1126068 - 1126075
SMON Set 3 Byte Counter	1126140 - 1126147
SMON Set 3 Packet Counter	1126076 - 1126083
SMON Set 4 Byte Counter	1126148 - 1126155
SMON Set 4 Packet Counter	1126084 - 1126091
SMON Set 5 Byte Counter	1126156 - 1126163
SMON Set 5 Packet Counter	1126092 - 1126099
SMON Set 6 Byte Counter	1126164 - 1126171
SMON Set 6 Packet Counter	1126100 - 1126107
SMON Set 7 Byte Counter	1126172 - 1126179
SMON Set 7 Packet Counter	1126108 - 1126115
SMON Set Search	1120673 - 1120680
SNAP LLC Decoding Options	1121217
SP Overflow Drop	16816 - 16827
Scratch	4
Second Tunnel Exit Drop	16883
Second Tunnel Exit Lookup TCAM	1123771 - 1123898
Second Tunnel Exit Lookup TCAM Answer	1122451 - 1122466
Second Tunnel Exit Miss Action	1120948 - 1121203
Security Association Table	1198688 - 1200735
Select Which Egress QoS Mapping Table To Use	1180438 - 1180693
Send to CPU	1122667
Software Aging Enable	319
Software Aging Start Latch	16764
Source Port Default ACL Action	331108 - 331195
Source Port Default ACL Action Drop	16912
Source Port Table	1122511 - 1122554
TC FFA Used	1129415 - 1129422
TC Tail-Drop FFA Threshold	1129468 - 1129475
TC Xoff FFA Threshold	1129460 - 1129467
TC Xon FFA Threshold	1129452 - 1129459
TOS QoS Mapping Table	1181014 - 1181525
Tail-Drop FFA Threshold	1129451
Time to Age	335
Transmitted Packets on Egress VRF	1182200 - 1182203
Tunnel Entry Header Data	1130936 - 1131447
Tunnel Entry Instruction Table	1130872 - 1130935
Tunnel Entry MTU Length Check	1118592 - 1118623
Tunnel Exit Lookup TCAM	1122683 - 1123194
Tunnel Exit Lookup TCAM Answer	16916 - 17171
Tunnel Exit Miss Action Drop	16884
Tunnel Exit Too Small Packet Modification Drop	16885
Tunnel Exit Too Small Packet Modification To Small Drop	1130767 - 1130778
Unknown Egress Drop	1130719 - 1130730
Unknown Ingress Drop	16866
VLAN Member Drop	16876
VLAN PCP And DEI To Color Mapping Table	1120090 - 1120105
VLAN PCP To Queue Mapping Table	1120626 - 1120633
VLAN Table	331196 - 347579

Name	Address Range
Xoff FFA Threshold	1129450
Xon FFA Threshold	1129449

38.5 Active Queue Manager

38.5.1 ERM Red Configuration

Configurations to mark the buffer memory congestion status as Red (heavily congested).

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1129050

Field Description

Bits	Field Name	Description	Default Value
10:0	redXoff	Number of free cells below this value will invoke the red congestion check for the incoming cells. The checks include the number of enqueued cells in the current queue and the packet length. The incoming packet might be terminated and dropped based on the check result.	0x6c
21:11	redXon	Once the red congestion check is applied, number of free cells need to go above this value to disable the check again. The value needs to be larger than **redXoff** to provide an effective hysteresis.	0x100
29:22	redMaxCells	Maximum allowed packet length in cells when the buffer memory congestion status is red.	0x9

38.5.2 ERM Yellow Configuration

Configurations to mark the buffer memory congestion status as Yellow (slightly congested).

Number of Entries : 1
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 1129040

Field Description

Bits	Field Name	Description	Default Value
10:0	yellowXoff	Number of free cells below this value will invoke yellow congestion checks for the incoming cells. The checks include the number of enqueued cells in the current queue, higher priority queues and optionally the total number of enqueued cells for the current egress port. Incoming packets might be terminated and dropped based on the check result.	0x180
21:11	yellowXon	Once the yellow congestion check is applied, number of free cells need to go above this value to disable the check again. The value needs to be larger than **yellowXoff** to provide an effective hysteresis.	0x210
33:22	redPortEn	When the buffer memory congestion status is yellow and a single port consumes more than **redPortXoff** cells, this field can apply the **redLimit** check on a per port basis.	0xfff
44:34	redPortXoff	When the buffer memory congestion status is yellow and the total number of cells enqueued on an egress port is larger than this value, **redLimit** check for that port will be invoked. Only valid when **redPortEn** is turned on.	0xab

38.5.3 Egress Resource Manager Pointer

This table provides each egress port a set of limiters. Different egress queues can have different pointers to the **Resource Limiter Set**.

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 1129051 to 1129062

Field Description

Bits	Field Name	Description	Default Value
1:0	q0	Pointer to the **Resource Limiter Set** for egress queue 0.	0x0
3:2	q1	Pointer to the **Resource Limiter Set** for egress queue 1.	0x0
5:4	q2	Pointer to the **Resource Limiter Set** for egress queue 2.	0x0
7:6	q3	Pointer to the **Resource Limiter Set** for egress queue 3.	0x0
9:8	q4	Pointer to the **Resource Limiter Set** for egress queue 4.	0x0
11:10	q5	Pointer to the **Resource Limiter Set** for egress queue 5.	0x0
13:12	q6	Pointer to the **Resource Limiter Set** for egress queue 6.	0x0
15:14	q7	Pointer to the **Resource Limiter Set** for egress queue 7.	0x0

38.5.4 Resource Limiter Set

This resource limiter is for comparing how many cells are ahead of the incoming cell for scheduling, that includes cells are enqueued in the same egress queue and all cells with a higher scheduling priority.

Number of Entries :	4
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	Pointer from the **Egress Resource Manager Pointer**
Address Space :	1129042 to 1129049

Field Description

Bits	Field Name	Description	Default Value
10:0	yellowAccumulated	When the buffer memory is slightly congested (yellow), the ERM allows accumulation of cells with the same queue or higher scheduling priorities to the limit in this field before appling the **yellowLimit**.	0x1d
21:11	yellowLimit	When the buffer memory is slightly congested (yellow)and **yellowAccumulated** is reached, the packet will be terminated and dropped if the enqueued cells in the corresponding queue is more than this value.	0x28
32:22	redLimit	When the buffer memory is heavily congested (red), the incoming packet will be terminated and dropped if the enqueued cells in the corresponding egress queue is more than this value.	0x15
40:33	maxCells	Maximum allowed packet length in cells for this limiter. Packet with cells more than this value will be dropped.	0xff

38.6 Core Information

38.6.1 Core Version

Adress 0 is reserved for the core version. Make sure the register value is the same as the revision number in the front page of the datasheet.

Number of Entries :	1
Type of Operation :	Read Only
Address Space :	0

Field Description

Bits	Field Name	Description	Default Value
31:0	version	Version of the core.	0xcda53817

38.7 Crypto

38.7.1 Crypto Configuration

General crypto configuraion

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1201248

Field Description

Bits	Field Name	Description	Default Value
0	drop	Packet drop control. If set, drop packets that do not pass cryptographic integrety check or anti replay checks. If not set, send these packets to the CPU.	0x0
1	fix_iv	Use fixed IPSec encrypt initialization vector. I.e. the hardware does not update the **Linear Feedback Shift Register**. Shall only be set for debugging purposes.	0x0

38.7.2 Crypto Sequence Numbers

Sequence numbers for encrypted data streams

Number of Entries : 64
Number of Addresses per Entry : 8
Type of Operation : Read/Write
Addressing : saPtr
Address Space : 1200736 to 1201247

Field Description

Bits	Field Name	Description	Default Value
63:0	snr	Sequence/ Number. Initialize to 1 when setting up a new stream.	0x1
191:64	bitmask	Anti replay bit mask. When software updates the sequence number, this field should be cleared to zero. For MACsec no bitmask is needed, but the replay window is configurable. The 32 least significant bits of this field defines the allowed replay window for MACsec.	0x0

38.7.3 Linear Feedback Shift Register

LFSR value access

Number of Entries : 1
Number of Addresses per Entry : 4
Type of Operation : Read/Write
Address Space : 1201249

Field Description

Bits	Field Name	Description	Default Value
127:0	lfsr	Current value of LFSR used for IPSec encryption initialization vectors. Updated by hardware for each encrypted packet. This field should normally never be written by software. Must not be set to zero, and should not be modified during operation as this might break the initialization vector uniqueness criteria. The LFSR is of maximum length, and uses the following taps: 128, 126, 101 and 99.	0x6ac10ce8eb1f74328d4bbd6d396d5a

38.7.4 Security Association Table

The configuration table for IPSec and MACsec.

The supported combinations of Crypto Mode, Auth Mode and Extended Sequence Numbering are listed in the table below.

Protocol	Crypto Mode	Auth Mode	Extended Sequence Numbers
AH	NULL	NULL	ESN=0
AH	NULL	NULL	ESN=1
AH	NULL	HMAC_MD5_96	ESN=0
AH	NULL	HMAC_SAH1_96	ESN=0
AH	NULL	HMAC_SHA2_256	ESN=0
AH	NULL	HMAC_SHA2_384	ESN=0
AH	NULL	HMAC_SHA2_512	ESN=0
AH	NULL	HMAC_SM3_128	ESN=0
ESP	NULL	NULL	ESN=0
ESP	NULL	NULL	ESN=1
ESP	AES128_CBC	NULL	ESN=0
ESP	AES128_CBC	NULL	ESN=1
ESP	AES128_CBC	HMAC_MD5_96	ESN=0
ESP	AES128_CBC	HMAC_SHA1_96	ESN=0
ESP	AES128_CBC	HMAC_SHA2_256	ESN=0
ESP	AES128_CBC	HMAC_SHA2_384	ESN=0
ESP	AES128_CBC	HMAC_SHA2_512	ESN=0
ESP	AES256_CBC	NULL	ESN=0
ESP	AES256_CBC	NULL	ESN=1
ESP	AES256_CBC	HMAC_MD5_96	ESN=0
ESP	AES256_CBC	HMAC_SHA1_96	ESN=0
ESP	AES256_CBC	HMAC_SHA2_256	ESN=0
ESP	AES256_CBC	HMAC_SHA2_384	ESN=0
ESP	AES256_CBC	HMAC_SHA2_512	ESN=0
ESP	AES128_CCM_8	NULL	ESN=0
ESP	AES128_CCM_8	NULL	ESN=1
ESP	AES256_CCM_8	NULL	ESN=0
ESP	AES256_CCM_8	NULL	ESN=1
ESP	AES128_GCM_16	NULL	ESN=0
ESP	AES128_GCM_16	NULL	ESN=1
ESP	AES256_GCM_16	NULL	ESN=0
ESP	AES256_GCM_16	NULL	ESN=1
ESP	DES_CBC	NULL	ESN=0
ESP	DES_CBC	HMAC_MD5_96	ESN=0
ESP	DES_CBC	HMAC_SHA1_96	ESN=0
ESP	DES3_CBC	NULL	ESN=0
ESP	DES3_CBC	HMAC_MD5_96	ESN=0
ESP	DES3_CBC	HMAC_SHA1_96	ESN=0
ESP	SM4_CBC	NULL	ESN=0
ESP	SM4_CBC	HMAC_SM3_128	ESN=0
ESP	SM4_CCM_8	NULL	ESN=0
ESP	SM4_CCM_8	NULL	ESN=1
ESP	SM4_GCM_8	NULL	ESN=0
ESP	SM4_GCM_8	NULL	ESN=1
ESP	NULL	AES_128_GMAC	ESN=0
ESP	NULL	AES_128_GMAC	ESN=1
ESP	NULL	AES_256_GMAC	ESN=0
ESP	NULL	AES_256_GMAC	ESN=1
MACSEC	GCM_AES_128	NULL	ESN=0
MACSEC	GCM_AES_128	NULL	ESN=1
MACSEC	GCM_AES_256	NULL	ESN=0
MACSEC	GCM_AES_256	NULL	ESN=1

Number of Entries :	64
Number of Addresses per Entry :	32
Type of Operation :	Read/Write
Addressing :	Security Association Pointer
Address Space :	1198688 to 1200735

Field Description

Bits	Field Name	Description	Default Value
3:0	cryptoMode	0 = NULL 1 = AES128_CBC 2 = AES256_CBC 3 = AES128_CCM_8 4 = AES256_CCM_8 5 = AES128_GCM_16 6 = AES256_GCM_16 7 = DES_CBC 8 = DES3_CBC 9 = SM4_CBC 10 = SM4_CCM_8 11 = SM4_GCM_16 12 = GCM_AES_128 13 = GCM_AES_256	0x0
7:4	authMode	0 = NULL 1 = HMAC_MD5_96 2 = HMAC_SHA1_96 3 = HMAC_SHA2_224 4 = HMAC_SHA2_256 5 = HMAC_SHA2_384 6 = HMAC_SHA2_512 7 = HMAC_SM3_128 8 = AES_128_GMAC 9 = AES_256_GMAC	0x0
103:8	salt	IPSec (bits 0 through 31) or MACsec salt value. The byte order is the same as for the **encryptionKey**.	0x0

359:104	encryptionKey	Encryption Key. For shorter keys, the key shall be aligned towards bit 0, and the upper, unused bits, shall be set to zero.
		
		A keys is stored so that the first byte of the key ends up on the most significant bits, while the last byte ends up on the least significant bits (i.e. bits 0 through 7 of this field).
		
		E.g. The 128 bit AES key (as printed in the AES standard):
		
		`2b 7e 15 16 28 ae d2 a6`
		`ab f7 15 88 09 cf 4f 3c`
		
		is stored so that `2b` ends up on bits 120-127 while `3c` ends up on bits 0-7 of this field.
		
		Note: For AES CBC and SM4 CBC **decryption**, the key from the last round of the key schedule shall be stored here, rather than the original encryption key.
			0x0

Bits	Field Name	Description	Default Value
871:360	integrityKey	Integrity Key. For shorter keys, the key shall be aligned towards bit 0, and the upper, unused bits, shall be set to zero. The byte order is the same as for the **encryptionKey**.	0x0
935:872	spi	IPSec Security Parameter Index (SPI) (bits 0 through 31) or MACsec Secure Channel Identifier (SCI). Bits 0 through 31 are also used as MACsec Short Secure Channel Identifier (SSCI). The value is stored as a little endian 64-bit unsigned value.	0x0
936	esn	Enable IPsec ESN or MACsec XPN 64-bit extended sequence numbers. For MACsec this implies using the GCM-AES-XPN Cipher Suites.	0x0
937	sc	Set the MACsec SC bit when encrypting. I.e. include the SCI in the MACsec SecTAG sent. For decryption, the SC bit from the SecTAG must be consistent with this field.	0x0
939:938	an	MACsec SecTAG AN value. Encoded in the SecTAG when encrypting, For decryption, the AN field from the SecTAG must be consitent with this field.	0x0

38.8 Egress Packet Processing

38.8.1 Beginning of Packet Tunnel Entry Instruction Table

The is the L2 tunnel entry instruction which described how a tunnel entry should be done after the L3 header. If the L3Type is either IPv4, IPv6 then the length fields are updated in the IP headers, for IPv4 the checksum is re-calculated. If the hasUDP is turned on then the UDP length-field is updated.

Number of Entries :	32
Type of Operation :	Read/Write
Addressing :	Tunnel entry pointer
Address Space :	1131448 to 1131479

Field Description

Bits	Field Name	Description	Default Value
1:0	l3Type	Inserted header type, when selecting MPLS/Other no updates will be done to the data. 0 = IPv4 1 = IPv6 2 = MPLS/Other. 3 = Reserved.	0x0
7:2	ipHeaderOffset	Where does the IPv4/IPv6 header start in this header. Only valid if the L3-Header type is IPv4 or IPv6.	0x0
8	hasUdp	If the header is a IPv4 or IPv6 then a an UDP header is after the IP header. 0 = No. 1 = Yes.	0x0

38.8.2 Color Remap From Egress Port

Options for remapping internal packet color to outgoing packet headers. Each egress port has a separate color to field mapping.

Number of Entries :	11
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	Egress Port
Address Space :	1131544 to 1131565

Field Description

Bits	Field Name	Description	Default Value
1:0	colorMode	0 = Skip remap 1 = Remap to L3 only 2 = Remap to L2 only 3 = Remap to L2 and L3	0x1
25:2	color2Tos	New TOS/TC value based on packet color. bits [0:7] : TOS/TC value for green bits [8:15] : TOS/TC value for yellow bits [16:23] : TOS/TC value for red	0x0
33:26	tosMask	Mask for updating the TOS/TC field. For each bit in the mask, 0 means keep original value, 1 means update new value to that bit.	0x0
36:34	color2Dei	New DEI value based on packet color. This is located in the outermost VLAN of the transmitted packet. bit 0 : DEI value for green bit 1 : DEI value for yellow bit 2 : DEI value for red	0x0

38.8.3 Color Remap From Ingress Admission Control

Options from ingress admission control to remap internal packet color to outgoing packet headers.

Number of Entries :	64
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	Meter Pointer
Address Space :	1131566 to 1131693

Field Description

Bits	Field Name	Description	Default Value
0	enable	If set, the **colorMode** field determines the remap process. Otherwise color remapping based on the ingress admission control is skipped.	0x0
2:1	colorMode	0 = Remap disabled 1 = Remap to L3 only 2 = Remap to L2 only 3 = Remap to L2 and L3	0x0

Bits	Field Name	Description	Default Value
26:3	color2Tos	New TOS/TC value based on packet color. bits [0:7] : TOS/TC value for green bits [8:15] : TOS/TC value for yellow bits [16:23] : TOS/TC value for red	0x0
34:27	tosMask	Mask for updating the TOS/TC field. For each bit in the mask, 0 means keep original value, 1 means update new value to that bit.	0x0
37:35	color2Dei	New DEI value based on packet color. This is located in the outermost VLAN of the transmitted packet. bit 0 : DEI value for green bit 1 : DEI value for yellow bit 2 : DEI value for red	0x0

38.8.4 Debug Counter debugMatchEPP0 Setup

Packet processing debug setup for
registerDebug debugMatchEPP0.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182086

Field Description

Bits	Field Name	Description	Default Value
13:0	mask	Mask for comparison to update debug counter.	0x0
27:14	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.8.5 Debug Counter fromPort Setup

Packet processing debug setup for
registerDebug fromPort.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182079

Field Description

Bits	Field Name	Description	Default Value
10:0	mask	Mask for comparison to update debug counter.	0x0
21:11	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.8.6 Debug Counter reQueuePortId Setup

Packet processing debug setup for
registerDebug reQueuePortId.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182070

Field Description

Bits	Field Name	Description	Default Value
3:0	mask	Mask for comparison to update debug counter.	0x0
7:4	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.8.7 Disable CPU tag on CPU Port

When a packet is sent to the CPU port normally a To CPU Tag will be added to the packet. This register provides a option to disable the CPU tag

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182084

Field Description

Bits	Field Name	Description	Default Value
0	disable	When set, the CPU port will no longer add a CPU Tag to packets going to the CPU port. 0 = To CPU Tag enabled 1 = To CPU Tag disabled	0x0
1	disableReason0	When set, the CPU port will no longer add a CPU Tag to packets going to the CPU port with reason code 0(default reason). 0 = To CPU Tag enabled 1 = To CPU Tag disabled	0x0

38.8.8 Drain Port

Drop all packets on all queues to egress ports. The dropped packets are counted in the **Drain Port Drop** counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1130706

Field Description

Bits	Field Name	Description	Default Value
11:0	drainMask	Egress ports to be drained. One bit for each port in the current switch slice where bit 0 corresponds to local port 0.	0x0

38.8.9 EPP Debug addNewMpls

Packet processing pipeline status for addNewMpls.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182099

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.8.10 EPP Debug debugMatchEPP0

Packet processing pipeline status for debugMatchEPP0.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182101

Field Description

Bits	Field Name	Description	Default Value
13:0	value	Status from last processed packet.	0x0

38.8.11 EPP Debug delSpecificVlan

Packet processing pipeline status for delSpecificVlan.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182095

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.8.12 EPP Debug fromPort

Packet processing pipeline status for fromPort.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182094

Field Description

Bits	Field Name	Description	Default Value
10:0	value	Status from last processed packet.	0x0

38.8.13 EPP Debug imActive

Packet processing pipeline status for imActive.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182087

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.8.14 EPP Debug imExtra

Packet processing pipeline status for imExtra.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182088

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.8.15 EPP Debug isIPv4

Packet processing pipeline status for isIPv4.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182097

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.8.16 EPP Debug isIPv6

Packet processing pipeline status for isIPv6.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182098

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.8.17 EPP Debug isPPPoE

Packet processing pipeline status for isPPPoE.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182100

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.8.18 EPP Debug omEnabled

Packet processing pipeline status for omEnabled.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182089

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.8.19 EPP Debug omImActive

Packet processing pipeline status for omImActive.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182090

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.8.20 EPP Debug reQueue

Packet processing pipeline status for reQueue.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182091

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.8.21 EPP Debug reQueuePkt

Packet processing pipeline status for reQueuePkt.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182093

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.8.22 EPP Debug reQueuePortId

Packet processing pipeline status for reQueuePortId.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182092

Field Description

Bits	Field Name	Description	Default Value
3:0	value	Status from last processed packet.	0x0

38.8.23 EPP Debug updateTosExp

Packet processing pipeline status for updateTosExp.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182096

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.8.24 Egress Ethernet Type for VLAN tag

Ethernet type used in VLAN operations when typeSel selects User Defined VLAN type. This Ethernet type is only used in VLAN push operations. In VLAN filtering a pushed user defined VLAN will be considered to be a C-VLAN.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182081

Field Description

Bits	Field Name	Description	Default Value
15:0	typeValue	Ethernet Type value.	0xffff

38.8.25 Egress Function Control

This register controls which functions a packet shall execute in the egress packet processing pipeline.

Number of Entries : 2
Type of Operation : Read/Write
Addressing : See **Egress Function Pointer Egress Port** and see function control chapter.
Address Space : 1182057 to 1182058

Field Description

Bits	Field Name	Description	Default Value
0	enableEgressPortVlanOperation	Shall the egress port VLAN operation be performed? 0 = No. 1 = Yes.	0x1
1	updateVrfOutStat	Shall the VRF out statistics counters be updated (they are only updated when a packet is routed)? 0 = No. 1 = Yes.	0x1
2	doEgressVlanTranslation	Shall the outgoing packet do the egress VLAN translation? 0 = No. 1 = Yes.	0x1
3	cancelIngressNatOp	Shall the outgoing packets ingress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
4	cancelEgressNatOp	Shall the outgoing packets egress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
5	doEgressQueueRemapping	Shall the egress QoS Remapping be carried out? 0 = No. 1 = Yes.	0x1
6	removeSNAP	Shall the outgoing packet remove the SNAP header? 0 = No. 1 = Yes.	0x1
7	cancelTunnelEntry	Cancel tunnel entry operation? 0 = No. 1 = Yes.	0x0
8	cancelTunnelExit	Cancel tunnel exit operation? 0 = No. 1 = Yes.	0x0
9	cancelRouting	Can the routing packet header updates? 0 = No. 1 = Yes.	0x0
10	doEgressMplsOp	Shall the egress MPLS operation be carried out? 0 = No. 1 = Yes.	0x1
11	allowTosUpdatesFromColoring	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the coloring? 0 = No. 1 = Yes.	0x1
12	allowTosUpdatesFromAcl	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the ACL action? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
13	enableEgressPortFilter	Shall the egress port vlan filtering operation (these are the fields **dropCtagged-Vlans**,**dropStaggedVlans**,**moreThanOneVlans**,**dropUntaggedVlans**,**dropS**) be carried out? 0 = No. It will not be carried out 1 = Yes.It will be carried out. 0 = No. 1 = Yes.	0x1
14	drop	Force the packet to be dropped, updates the **Ingress Functional Control Drops** counter. 0 = No. 1 = Yes.	0x0

38.8.26 Egress Function Control Packet From CPU Port

This register controls which functions a packet shall execute in the egress packet processing pipeline when a packet shall be sent to the CPU port.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182073

Field Description

Bits	Field Name	Description	Default Value
0	enableEgressPortVlanOperation	Shall the egress port VLAN operation be performed? 0 = No. 1 = Yes.	0x1
1	updateVrfOutStat	Shall the VRF out statistics counters be updated (they are only updated when a packet is routed)? 0 = No. 1 = Yes.	0x1
2	doEgressVlanTranslation	Shall the outgoing packet do the egress VLAN translation? 0 = No. 1 = Yes.	0x1
3	cancelIngressNatOp	Shall the outgoing packets ingress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
4	cancelEgressNatOp	Shall the outgoing packets egress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
5	doEgressQueueRemapping	Shall the egress QoS Remapping be carried out? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
6	removeSNAP	Shall the outgoing packet remove the SNAP header? 0 = No. 1 = Yes.	0x1
7	cancelTunnelEntry	Cancel tunnel entry operation? 0 = No. 1 = Yes.	0x0
8	cancelTunnelExit	Cancel tunnel exit operation? 0 = No. 1 = Yes.	0x0
9	cancelRouting	Can the routing packet header updates? 0 = No. 1 = Yes.	0x0
10	doEgressMplsOp	Shall the egress MPLS operation be carried out? 0 = No. 1 = Yes.	0x1
11	allowTosUpdatesFromColoring	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the coloring? 0 = No. 1 = Yes.	0x1
12	allowTosUpdatesFromAcl	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the ACL action? 0 = No. 1 = Yes.	0x1
13	enableEgressPortFilter	Shall the egress port vlan filtering operation (these are the fields **dropCtaggedVlans**,**dropStaggedVlans**,**moreThanOneVlans**,**dropUntaggedVlans**,**dropS**) be carried out? 0 = No. It will not be carried out 1 = Yes.It will be carried out. 0 = No. 1 = Yes.	0x1
14	drop	Force the packet to be dropped, updates the **Ingress Functional Control Drops** counter. 0 = No. 1 = Yes.	0x0

38.8.27 Egress Function Control Packet From CPU Tag

This register controls which functions a packet shall execute in the egress packet processing pipeline when a packet shall be sent from the CPU port and had a from-CPU tag.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182076

Field Description

Bits	Field Name	Description	Default Value
0	enableEgressPortVlanOperation	Shall the egress port VLAN operation be performed? 0 = No. 1 = Yes.	0x1
1	updateVrfOutStat	Shall the VRF out statistics counters be updated (they are only updated when a packet is routed)? 0 = No. 1 = Yes.	0x1
2	doEgressVlanTranslation	Shall the outgoing packet do the egress VLAN translation? 0 = No. 1 = Yes.	0x1
3	cancelIngressNatOp	Shall the outgoing packets ingress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
4	cancelEgressNatOp	Shall the outgoing packets egress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
5	doEgressQueueRemapping	Shall the egress QoS Remapping be carried out? 0 = No. 1 = Yes.	0x1
6	removeSNAP	Shall the outgoing packet remove the SNAP header? 0 = No. 1 = Yes.	0x1
7	cancelTunnelEntry	Cancel tunnel entry operation? 0 = No. 1 = Yes.	0x0
8	cancelTunnelExit	Cancel tunnel exit operation? 0 = No. 1 = Yes.	0x0
9	cancelRouting	Can the routing packet header updates? 0 = No. 1 = Yes.	0x0
10	doEgressMplsOp	Shall the egress MPLS operation be carried out? 0 = No. 1 = Yes.	0x1
11	allowTosUpdatesFromColoring	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the coloring? 0 = No. 1 = Yes.	0x1
12	allowTosUpdatesFromAcl	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the ACL action? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
13	enableEgressPortFilter	Shall the egress port vlan filtering operation (these are the fields **dropCtagged-Vlans**,**dropStaggedVlans**,**moreThanOneVlans**,**dropUntaggedVlans**,**dropS**) be carried out? 0 = No. It will not be carried out 1 = Yes.It will be carried out. 0 = No. 1 = Yes.	0x1
14	drop	Force the packet to be dropped, updates the **Ingress Functional Control Drops** counter. 0 = No. 1 = Yes.	0x0

38.8.28 Egress Function Control Packet From CPU Tag Do Not Modify

This register controls which functions a packet shall execute in the egress packet processing pipeline when a packet shall be sent to the CPU port and had a from-CPU tag where the do not change bit set to one.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182077

Field Description

Bits	Field Name	Description	Default Value
0	enableEgressPortVlanOperation	Shall the egress port VLAN operation be performed? 0 = No. 1 = Yes.	0x1
1	updateVrfOutStat	Shall the VRF out statistics counters be updated (they are only updated when a packet is routed)? 0 = No. 1 = Yes.	0x1
2	doEgressVlanTranslation	Shall the outgoing packet do the egress VLAN translation? 0 = No. 1 = Yes.	0x1
3	cancelIngressNatOp	Shall the outgoing packets ingress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
4	cancelEgressNatOp	Shall the outgoing packets egress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
5	doEgressQueueRemapping	Shall the egress QoS Remapping be carried out? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
6	removeSNAP	Shall the outgoing packet remove the SNAP header? 0 = No. 1 = Yes.	0x1
7	cancelTunnelEntry	Cancel tunnel entry operation? 0 = No. 1 = Yes.	0x0
8	cancelTunnelExit	Cancel tunnel exit operation? 0 = No. 1 = Yes.	0x0
9	cancelRouting	Can the routing packet header updates? 0 = No. 1 = Yes.	0x0
10	doEgressMplsOp	Shall the egress MPLS operation be carried out? 0 = No. 1 = Yes.	0x1
11	allowTosUpdatesFromColoring	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the coloring? 0 = No. 1 = Yes.	0x1
12	allowTosUpdatesFromAcl	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the ACL action? 0 = No. 1 = Yes.	0x1
13	enableEgressPortFilter	Shall the egress port vlan filtering operation (these are the fields **dropCtaggedVlans**,**dropStaggedVlans**,**moreThanOneVlans**,**dropUntaggedVlans**,**dropS**) be carried out? 0 = No. It will not be carried out 1 = Yes.It will be carried out. 0 = No. 1 = Yes.	0x1
14	drop	Force the packet to be dropped, updates the **Ingress Functional Control Drops** counter. 0 = No. 1 = Yes.	0x0

38.8.29 Egress Function Control Packet From Crypto Engine Decrypted

This register controls which functions a packet shall execute in the egress packet processing pipeline when a packet shall be sent to the crypto engine.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182071

Field Description

Bits	Field Name	Description	Default Value
0	enableEgressPortVlanOperation	Shall the egress port VLAN operation be performed? 0 = No. 1 = Yes.	0x1
1	updateVrfOutStat	Shall the VRF out statistics counters be updated (they are only updated when a packet is routed)? 0 = No. 1 = Yes.	0x1
2	doEgressVlanTranslation	Shall the outgoing packet do the egress VLAN translation? 0 = No. 1 = Yes.	0x1
3	cancelIngressNatOp	Shall the outgoing packets ingress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
4	cancelEgressNatOp	Shall the outgoing packets egress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
5	doEgressQueueRemapping	Shall the egress QoS Remapping be carried out? 0 = No. 1 = Yes.	0x1
6	removeSNAP	Shall the outgoing packet remove the SNAP header? 0 = No. 1 = Yes.	0x1
7	cancelTunnelEntry	Cancel tunnel entry operation? 0 = No. 1 = Yes.	0x0
8	cancelTunnelExit	Cancel tunnel exit operation? 0 = No. 1 = Yes.	0x0
9	cancelRouting	Can the routing packet header updates? 0 = No. 1 = Yes.	0x0
10	doEgressMplsOp	Shall the egress MPLS operation be carried out? 0 = No. 1 = Yes.	0x1
11	allowTosUpdatesFromColoring	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the coloring? 0 = No. 1 = Yes.	0x1
12	allowTosUpdatesFromAcl	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the ACL action? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
13	enableEgressPortFilter	Shall the egress port vlan filtering operation (these are the fields **dropCtagged-Vlans**,**dropStaggedVlans**,**moreThanOneVlans**,**dropUntaggedVlans**,**dropS**) be carried out? 0 = No. It will not be carried out 1 = Yes.It will be carried out. 0 = No. 1 = Yes.	0x1
14	drop	Force the packet to be dropped, updates the **Ingress Functional Control Drops** counter. 0 = No. 1 = Yes.	0x0

38.8.30 Egress Function Control Packet From Crypto Engine Encrypted

This register controls which functions a packet shall execute in the egress packet processing pipeline when a packet shall be sent to the crypto engine.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182072

Field Description

Bits	Field Name	Description	Default Value
0	enableEgressPortVlanOperation	Shall the egress port VLAN operation be performed? 0 = No. 1 = Yes.	0x1
1	updateVrfOutStat	Shall the VRF out statistics counters be updated (they are only updated when a packet is routed)? 0 = No. 1 = Yes.	0x1
2	doEgressVlanTranslation	Shall the outgoing packet do the egress VLAN translation? 0 = No. 1 = Yes.	0x1
3	cancelIngressNatOp	Shall the outgoing packets ingress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
4	cancelEgressNatOp	Shall the outgoing packets egress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
5	doEgressQueueRemapping	Shall the egress QoS Remapping be carried out? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
6	removeSNAP	Shall the outgoing packet remove the SNAP header? 0 = No. 1 = Yes.	0x1
7	cancelTunnelEntry	Cancel tunnel entry operation? 0 = No. 1 = Yes.	0x0
8	cancelTunnelExit	Cancel tunnel exit operation? 0 = No. 1 = Yes.	0x0
9	cancelRouting	Can the routing packet header updates? 0 = No. 1 = Yes.	0x0
10	doEgressMplsOp	Shall the egress MPLS operation be carried out? 0 = No. 1 = Yes.	0x1
11	allowTosUpdatesFromColoring	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the coloring? 0 = No. 1 = Yes.	0x1
12	allowTosUpdatesFromAcl	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the ACL action? 0 = No. 1 = Yes.	0x1
13	enableEgressPortFilter	Shall the egress port vlan filtering operation (these are the fields **dropCtaggedVlans**,**dropStaggedVlans**,**moreThanOneVlans**,**dropUntaggedVlans**,**dropS**) be carried out? 0 = No. It will not be carried out 1 = Yes.It will be carried out. 0 = No. 1 = Yes.	0x1
14	drop	Force the packet to be dropped, updates the **Ingress Functional Control Drops** counter. 0 = No. 1 = Yes.	0x0

38.8.31 Egress Function Control Packet To CPU Port

This register controls which functions a packet shall execute in the egress packet processing pipeline when a packet shall be sent to the CPU port.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182075

Field Description

Bits	Field Name	Description	Default Value
0	enableEgressPortVlanOperation	Shall the egress port VLAN operation be performed? 0 = No. 1 = Yes.	0x1
1	updateVrfOutStat	Shall the VRF out statistics counters be updated (they are only updated when a packet is routed)? 0 = No. 1 = Yes.	0x1
2	doEgressVlanTranslation	Shall the outgoing packet do the egress VLAN translation? 0 = No. 1 = Yes.	0x1
3	cancelIngressNatOp	Shall the outgoing packets ingress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
4	cancelEgressNatOp	Shall the outgoing packets egress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
5	doEgressQueueRemapping	Shall the egress QoS Remapping be carried out? 0 = No. 1 = Yes.	0x1
6	removeSNAP	Shall the outgoing packet remove the SNAP header? 0 = No. 1 = Yes.	0x1
7	cancelTunnelEntry	Cancel tunnel entry operation? 0 = No. 1 = Yes.	0x0
8	cancelTunnelExit	Cancel tunnel exit operation? 0 = No. 1 = Yes.	0x0
9	cancelRouting	Can the routing packet header updates? 0 = No. 1 = Yes.	0x0
10	doEgressMplsOp	Shall the egress MPLS operation be carried out? 0 = No. 1 = Yes.	0x1
11	allowTosUpdatesFromColoring	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the coloring? 0 = No. 1 = Yes.	0x1
12	allowTosUpdatesFromAcl	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the ACL action? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
13	enableEgressPortFilter	Shall the egress port vlan filtering operation (these are the fields **dropCtagged-Vlans**,**dropStaggedVlans**,**moreThanOneVlans**,**dropUntaggedVlans**,**dropS**) be carried out? 0 = No. It will not be carried out 1 = Yes.It will be carried out. 0 = No. 1 = Yes.	0x1
14	drop	Force the packet to be dropped, updates the **Ingress Functional Control Drops** counter. 0 = No. 1 = Yes.	0x0

38.8.32 Egress Function Control Packet To CPU Port with Reason Zero

This register controls which functions a packet shall execute in the egress packet processing pipeline when a packet shall be sent to the CPU port, the packet is sent to the CPU port from a L2/L3 table entry, i.e. the reason code is zero.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182074

Field Description

Bits	Field Name	Description	Default Value
0	enableEgressPortVlanOperation	Shall the egress port VLAN operation be performed? 0 = No. 1 = Yes.	0x1
1	updateVrfOutStat	Shall the VRF out statistics counters be updated (they are only updated when a packet is routed)? 0 = No. 1 = Yes.	0x1
2	doEgressVlanTranslation	Shall the outgoing packet do the egress VLAN translation? 0 = No. 1 = Yes.	0x1
3	cancelIngressNatOp	Shall the outgoing packets ingress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
4	cancelEgressNatOp	Shall the outgoing packets egress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
5	doEgressQueueRemapping	Shall the egress QoS Remapping be carried out? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
6	removeSNAP	Shall the outgoing packet remove the SNAP header? 0 = No. 1 = Yes.	0x1
7	cancelTunnelEntry	Cancel tunnel entry operation? 0 = No. 1 = Yes.	0x0
8	cancelTunnelExit	Cancel tunnel exit operation? 0 = No. 1 = Yes.	0x0
9	cancelRouting	Can the routing packet header updates? 0 = No. 1 = Yes.	0x0
10	doEgressMplsOp	Shall the egress MPLS operation be carried out? 0 = No. 1 = Yes.	0x1
11	allowTosUpdatesFromColoring	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the coloring? 0 = No. 1 = Yes.	0x1
12	allowTosUpdatesFromAcl	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the ACL action? 0 = No. 1 = Yes.	0x1
13	enableEgressPortFilter	Shall the egress port vlan filtering operation (these are the fields **dropCtaggedVlans**,**dropStaggedVlans**,**moreThanOneVlans**,**dropUntaggedVlans**,**dropS**) be carried out? 0 = No. It will not be carried out 1 = Yes.It will be carried out. 0 = No. 1 = Yes.	0x1
14	drop	Force the packet to be dropped, updates the **Ingress Functional Control Drops** counter. 0 = No. 1 = Yes.	0x0

38.8.33 Egress Function Control Packet To Crypto Engine

This register controls which functions a packet shall execute in the egress packet processing pipeline when a packet shall be sent to the crypto engine.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182078

Field Description

Bits	Field Name	Description	Default Value
0	enableEgressPortVlanOperation	Shall the egress port VLAN operation be performed? 0 = No. 1 = Yes.	0x1
1	updateVrfOutStat	Shall the VRF out statistics counters be updated (they are only updated when a packet is routed)? 0 = No. 1 = Yes.	0x1
2	doEgressVlanTranslation	Shall the outgoing packet do the egress VLAN translation? 0 = No. 1 = Yes.	0x1
3	cancelIngressNatOp	Shall the outgoing packets ingress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
4	cancelEgressNatOp	Shall the outgoing packets egress NAT operation be canceled? 0 = No. 1 = Yes.	0x0
5	doEgressQueueRemapping	Shall the egress QoS Remapping be carried out? 0 = No. 1 = Yes.	0x1
6	removeSNAP	Shall the outgoing packet remove the SNAP header? 0 = No. 1 = Yes.	0x1
7	cancelTunnelEntry	Cancel tunnel entry operation? 0 = No. 1 = Yes.	0x0
8	cancelTunnelExit	Cancel tunnel exit operation? 0 = No. 1 = Yes.	0x0
9	cancelRouting	Can the routing packet header updates? 0 = No. 1 = Yes.	0x0
10	doEgressMplsOp	Shall the egress MPLS operation be carried out? 0 = No. 1 = Yes.	0x1
11	allowTosUpdatesFromColoring	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the coloring? 0 = No. 1 = Yes.	0x1
12	allowTosUpdatesFromAcl	Shall the TOS byte in IPv4 or Traffic Class in IPv6 be allowed to be updated from the ACL action? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
13	enableEgressPortFilter	Shall the egress port vlan filtering operation (these are the fields **dropCtagged-Vlans**,**dropStaggedVlans**,**moreThanOneVlans**,**dropUntaggedVlans**,**dropS**) be carried out? 0 = No. It will not be carried out 1 = Yes.It will be carried out. 0 = No. 1 = Yes.	0x1
14	drop	Force the packet to be dropped, updates the **Ingress Functional Control Drops** counter. 0 = No. 1 = Yes.	0x0

38.8.34 Egress Function Pointer Egress Port

This register controls which basic function a port shall be using by pointing to the entry in the **Egress Function Control** which a egress port shall use.

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Egress Port
Address Space : 1182059 to 1182069

Field Description

Bits	Field Name	Description	Default Value
0	ptr	Which functinal setting shall be used for this egress port.	0x0

38.8.35 Egress MPLS Decoding Options

When doing a Penultimate Pop then compare the first nibble after the innermost MPLS tag with this registers field nibbleForIpv4 to determine if the outgoing packet should have an IPv4 or IPv6 Ethernet Type.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182080

Field Description

Bits	Field Name	Description	Default Value
3:0	nibbleForIpv4	The nibble value which is used to identify a IPv4 packet after a MPLS header. If the nibble does not match this value it is assumed to be an IPv6 packet.	0x4

38.8.36 Egress MPLS TTL Table

Configuration of what modification shall be done on the TTL field in MPLS routed packets.

Number of Entries : 4
Type of Operation : Read/Write
Addressing : Packets VRF
Address Space : 1137914 to 1137917

Field Description

Bits	Field Name	Description	Default Value
0	addNewTTL	Select if the router should decremented TTL in the outgoing packet or if it should be set to a fixed value. 0 = Decrement TTL 1 = Set the TTL to **newTTL**	0x0
8:1	newTTL	New TTL for the packet. Only used when addNewTTL is set to 1	0x0

38.8.37 Egress Multiple Spanning Tree State

Table of egress Multiple Spanning Tree Protocol Instances. Depends on routed or not, the pointer used to address the instance/entry in this table can from **msptPtr** in the **Next Hop Packet Modifications** table or **msptPtr** in the **VLAN Table**. Each entry contains the ingress spanning tree states for all ports in this MSTI.

Number of Entries : 16
Type of Operation : Read/Write
Addressing : msptPtr from VLAN Table or Next Hop Packet Modifications Table
Address Space : 1119459 to 1119474

Field Description

Bits	Field Name	Description	Default Value
21:0	portSptState	The egress spanning tree state for this MSTI. Bit[1:0] is the state for port #0, bit[3:2] is the state for port #1, etc. 0 = Forwarding 1 = Discarding 2 = Learning	0x0

38.8.38 Egress NAT Operation

Egress NAT Operation Table.

Number of Entries : 8192
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Addressing : Egress ACL NAT Pointer plus egress port number.
Address Space : 1164054 to 1180437

Field Description

Bits	Field Name	Description	Default Value
0	replaceSrc	Replace Source or Destination. 0 = Destination 1 = Source	0x0
1	replaceIP	Replace IP address. 0 = No. 1 = Yes.	0x0
2	replaceL4Port	Replace TCP/UDP port. 0 = No. 1 = Yes.	0x0
34:3	ipAddress	The new IP Address.	0x0
50:35	port	The new L4 Port.	0x0
53:51	entryVersion	The version of this entry.Must be same version as the entry pointing to it.	0x0

38.8.39 Egress Port Configuration

This table configures various functions that are dependent on which port the packet leaves the switch. A VLAN operation (e.g. push, pop, swap) to be performed can be selected by the **vlanSingleOp** field. For the push and swap operations the information used to create the new VLAN header is controlled by the fields **vidSel**, **cfiDeiSel**, **pcpSel** and **typeSel**. Other configurations are VLAN LUT index, port disable and different filtering rules based on packet VLAN fields when the egress processing is done.

Number of Entries :	11
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	Egress port
Address Space :	1130818 to 1130839

Field Description

Bits	Field Name	Description	Default Value
0	colorRemap	If set, color remapping to outgoing packet headers is allowed. The default color remapping options are based on the egress port number from the **Color Remap From Egress Port** table. If a packet is subjected to ingress admission control, its ingress admission control pointer can provide remap options from the **Color Remap From Ingress Admission Control** table to override default options.	0x0
3:1	vlanSingleOp	The egress port VLAN operation to perform on the packet. 0 = No operation. 1 = Swap. 2 = Push. 3 = Pop. 4 = Penultimate pop(remove all VLAN headers).	0x0

Bits	Field Name	Description	Default Value
4	removeSNAP	If a packet which has SNAP/LLC encoding then remove it before sending out the packet on this egress port. 0 = No. Keep it. 1 = Yes. Remove it.	0x0
6:5	typeSel	Selects which TPID to use when building a new VLAN header in a push or swap operation. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register **Egress Ethernet Type for VLAN tag** field **typeValue**.	0x0
8:7	vidSel	Selects which VID to use when building a new VLAN header in a egress port push or swap operation. If the selected outermost VLAN header doesn't exist in the packet then this table entry's **vid** will be used. 0 = From outermost VLAN in the packet (if any). 1 = From this table entry's **vid**. 2 = From the Ingress VID as selected in the **Source Port Table**.	0x0
10:9	cfiDeiSel	Selects which CFI/DEI to use when building a new VLAN header in a egress port push or swap operation. If the selected outermost VLAN header doesn't exist in the packet then this table entry's **cfiDei** will be used. 0 = From outermost VLAN in the packet (if any). 1 = From this table entry's **cfiDei**. 2 = From **Egress Queue To PCP And CFI/DEI Mapping Table**.	0x0
12:11	pcpSel	Selects which PCP to use when building a new VLAN header in a egress port push or swap operation. If the selected outermost VLAN header doesn't exist in the packet then this table entry's **cfiDei** will be used. 0 = From outermost VLAN in the packet (if any). 1 = From this table entry's **pcp**. 2 = From **Egress Queue To PCP And CFI/DEI Mapping Table**.	0x0
24:13	vid	The VID used in egress port VLAN push or swap operation if selected by **vidSel**.	0x0
25	cfiDei	The CFI/DEI used in egress port VLAN push or swap operation if selected by **cfiDeiSel**.	0x0
28:26	pcp	The PCP used in egress port VLAN push or swap operation if selected by **pcpSel**.	0x0
29	disabled	Disabling this port. All packets to this port is dropped and **Egress Port Disabled Drop** is incremented. 0 = All packets will be sent out. 1 = All packets will be dropped.	0x0

Bits	Field Name	Description	Default Value
30	dropCtaggedVlans	Drop or allow customer VLANs tagged packets on this egress port. Will only drop packets that has exactly one VLAN tag. Must set moreThanOneVlans when this is used. 0 = Allow C-VLANs. 1 = Drop C-VLANs.	0x0
31	dropStaggedVlans	Drop or allow service VLANs tagged packets on this egress port. Will only drop packets that has exactly one VLAN tag. Must set moreThanOneVlans when this is used. 0 = Allow S-VLANs. 1 = Drop S-VLANs.	0x0
32	moreThanOneVlans	When filtering with dropCtaggedVlans or dropStaggedVlans then this field must be set to 1.	0x0
33	dropUntaggedVlans	Drop or Allow packets that are VLAN untagged on this egress port. 0 = Allow untagged packets. 1 = Drop untagged packets.	0x0
34	dropSingleTaggedVlans	Drop or Allow packets that has one VLAN tag on this egress port. 0 = Allow untagged packets. 1 = Drop untagged packets.	0x0
35	dropDualTaggedVlans	Drop or allow packets which has more than one VLAN tag on this egress port. 0 = Allow packets which has more than one VLAN tag. 1 = Drop packets which has more than one VLAN tag.	0x0
36	dropCStaggedVlans	Drop or allow packets which has a C-VLAN followed by a S-VLAN tagged on this egress port. 0 = Allow packets which has a C-VLAN tag followed by a S-VLAN tag. 1 = Drop packets which has a C-VLAN tag followed by a S-VLAN tag.	0x0
37	dropSCtaggedVlans	Drop or allow packets which has a S-VLAN followed by a C-VLAN tagged on this egress port. 0 = Allow packets which has a S-VLAN followed by a C-VLAN tag. 1 = Drop packets which has a S-VLAN tag followed by a C-VLAN tag.	0x0
38	dropCCtaggedVlans	Drop or allow packets which has a C-VLAN followed by a C-VLAN tagged on this egress port. 0 = Allow packets which has a C-VLAN tag followed by a C-VLAN tag. 1 = Drop packets which has a C-VLAN tag followed by a C-VLAN tag.	0x0
39	dropSStaggedVlans	Drop or allow packets which has a S-VLAN followed by a S-VLAN tagged on this egress port. 0 = Allow packets which has a S-VLAN tag followed by a S-VLAN tag. 1 = Drop packets which has a S-VLAN tag followed by a S-VLAN tag.	0x0
40	tunnelEntry	Shall all of these packets enter into a tunnel on this egress port. Other tunnel entry on this port will be overriden.	0x0

Bits	Field Name	Description	Default Value
41	tunnelEntryUcMc	Shall this entry point to the **Tunnel Entry Instruction Table** using the destiantion port as a offset or directly without any offset. 0 = Unicast **Tunnel Entry Instruction Table** without port offset. 1 = Multicast **Tunnel Entry Instruction Table** with offset.	0x0
46:42	tunnelEntryPtr	The tunnel entry which this packet shall enter upon exiting the switch.	0x0
47	tunnelExit	Shall this packet do a tunnel exit. 0 = No 1 = Yes	0x0
51:48	tunnelExitPtr	Pointer to tunnel exit described in **Egress Tunnel Exit Table**.	0x0
52	useEgressQueueRemapping	Which remapping to final PCP, DEI, EXP and TOS fields shall be used for this port. 0 = Only use Egress Queue Remapping Tables 1 = First use the Egress Queue Remapping Tables then use the **Select Which Egress QoS Mapping Table To Use** to determine the final DEI,CFI,TOS and EXP fields.	0x0

38.8.40 Egress Port VID Operation

This search table checks the ingress VID and the number of VLANs before the egress port VLAN operation. If both ingress VID and number of VLANs are in the defined range then the VLAN operation in this table will override egress port VLAN operations. In case of multiple hit, VLAN operation from the first hit takes effect.

Number of Entries :	16
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	1182102 to 1182165

Field Description

Bits	Field Name	Description	Default Value
2:0	vlanSingleOpIf	If this entry is hit, then this VLAN operation will override egress port VLAN operation. 0 = No operation. 1 = Swap. 2 = Push. 3 = Pop. 4 = Penultimate pop(remove all VLAN headers).	0x0
4:3	typeSelIf	If this entry is hit, selects which TPID to use when building a new VLAN header in a push or swap operation. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register **Egress Ethernet Type for VLAN tag** field **typeValue**.	0x0

Bits	Field Name	Description	Default Value
6:5	vidSelIf	Selects which VID to use when building a new VLAN header in a egress port push or swap operation. If the selected outermost VLAN header doesn't exist in the packet then this table entry's vidIf will be used. 0 = From outermost VLAN in the packet (if any). 1 = From this table entry's **vidIf**. 2 = From the Ingress VID as selected in the **Source Port Table**.	0x0
8:7	cfiDeiSelIf	Selects which CFI/DEI to use when building a new VLAN header in a egress port push or swap operation. If the selected outermost VLAN header doesn't exist in the packet then this table entry's **cfiDei** will be used. 0 = From outermost VLAN in the packet (if any). 1 = From this table entry's **cfiDeiIf**. 2 = From **Egress Queue To PCP And CFI/DEI Mapping Table**.	0x0
10:9	pcpSelIf	Selects which PCP to use when building a new VLAN header in a egress port push or swap operation. If the selected outermost VLAN header doesn't exist in the packet then this table entry's **cfiDeiIf** will be used. 0 = From outermost VLAN in the packet (if any). 1 = From this table entry's **pcp**. 2 = From **Egress Queue To PCP And CFI/DEI Mapping Table**.	0x0
22:11	vidIf	VID used in VLAN push or swap operation if vidSelIf chooses VID from this table.	0x0
23	cfiDeiIf	CFI/DEI used in VLAN push or swap operation if cfiDeiSelIf chooses CFI/DEI from this table.	0x0
26:24	pcpIf	PCP used in VLAN push or swap operation if pcpSelIf chooses PCP from this table.	0x0
38:27	startVid	Start of ingress VID to hit.	0x0
50:39	endVid	End of ingress VID to hit.	0x0
53:51	minNrVlans	Minimum number of VLANs to hit	0x0
56:54	maxNrVlans	Maximum number of VLANs to hit	0x0
67:57	validPorts	Determine the valid egress port list.	0x0

38.8.41 Egress Queue To MPLS EXP Mapping Table

Map from egress queue number to MPLS EXP value to be used in MPLS operations selected by **Next Hop MPLS Table** and by **Next Hop Packet Insert MPLS Header** .

Number of Entries : 8
Type of Operation : Read/Write
Addressing : Egress Queue
Address Space : 1182049 to 1182056

Field Description

Bits	Field Name	Description	Default Value
2:0	exp	The outgoing Exp value for this queue.	0x0

38.8.42 Egress Queue To PCP And CFI/DEI Mapping Table

Get PCP and CFI/DEI from egress queues if selected by egress port VLAN operations push or swap.

Number of Entries : 8
Type of Operation : Read/Write
Addressing : Egress Queue
Address Space : 1146110 to 1146117

Field Description

Bits	Field Name	Description	Default Value
0	cfiDei	Map from egress queue to CFI/DEI.	0x0
3:1	pcp	Map from egress queue to PCP.	0x0

38.8.43 Egress Router Table

Configuration of what modification shall be done on the TTL field in routed packets.

Number of Entries : 4
Type of Operation : Read/Write
Addressing : Packets VRF
Address Space : 1131694 to 1131697

Field Description

Bits	Field Name	Description	Default Value
0	addNewTTL	Select if the router should decremented TTL in the outgoing packet or if it should be set to a fixed value. 0 = Decrement TTL 1 = Set the TTL to **newTTL**	0x0
8:1	newTTL	New TTL for the packet. Only used when addNewTTL is set to 1	0x0

38.8.44 Egress Tunnel Exit Table

The same packet exit which is is done at ingress described in the second tunnel exit lookup. Setting must be the same. This tunnel exit can also be used by the L2, L3 and ACL actions.

Number of Entries : 16
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Addressing : From Various tables during ingress packet processing
Address Space : 1130840 to 1130871

Field Description

Bits	Field Name	Description	Default Value
7:0	howManyBytesToRemove	How many bytes to remove.	0x0

Bits	Field Name	Description	Default Value
8	updateEthType	If packet is removed after L2+VLAN headers then update the Ethernet Header Type Field	0x0
24:9	ethType	If packet is removed after L2+VLAN headers then the New Ethernet Type which will overwrite the existing lowest 16 bits after the removal operation.	0x0
25	removeVlan	If packet is removed after L2+VLAN headers then remove the VLAN headers on the incoming packet.	0x0
26	updateL4Protocol	If packet is removed after L3 headers then update the L4 Protocol in IP header.	0x0
34:27	l4Protocol	If packet is removed after L3 headers then this new L4 Protocol will be written.	0x0
36:35	whereToRemove	Where to do the tunnel exit from 0 = At Byte Zero 1 = After L2 and up to two VLAN headers. 2 = After L3 IPv4/IPv6 headers. 3 = Reserved.	0x0

38.8.45 Egress VLAN Translation Large Table

The outermost VID and VID Ethernet Type (Service tag or Customer tag types) of the outgoing packet is compared.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries : 512
Number of Addresses per Entry : 2
Type of Operation : Read/Write

Addressing :

address[6:0]	:	hash of { dstPort outermostVid outermostVidType }
address[8:7]	:	bucket number

Address Space : 1146118 to 1147141

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
4:1	dstPort	This is a field which is used as search data. The destination port which the packet is going out on	0x0
16:5	outermostVid	This is a field which is used as search data. The outermost VID of the modified packet.	0x0
17	outermostVidType	This is a field which is used as search data. The outermost VID is a S-tag or C-Tag. 0 = Customer tag 1 = Service tag	0x0
29:18	newVid	This is a result field used when this entry is hit. The new VID for the outgoing packet.	0x0
45:30	ethType	This is a result field used when this entry is hit. The new Ethernet Type for the outgoing packet	0x0

38.8.46 Egress VLAN Translation Search Mask

Before the hashing and searching is done in the **Egress VLAN Translation Large Table** and **Egress VLAN Translation Small Table** The search data is AND:ed with this mask. If a bit in the mask is set to

zero then this bit in the lookup will be viewed as do not care. Seperate masks exists for both small and large tables.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	1182166

Field Description

Bits	Field Name	Description	Default Value
3:0	dstPort_mask_small	Which bits to compare in the field dstPort in **Egress VLAN Translation Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0xf
7:4	dstPort_mask_large	Which bits to compare in the field dstPort **Egress VLAN Translation Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0xf
19:8	outermostVid_mask_small	Which bits to compare in the field outermostVid in **Egress VLAN Translation Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0xfff
31:20	outermostVid_mask_large	Which bits to compare in the field outermostVid **Egress VLAN Translation Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0xfff
32	outermostVidType_mask_small	Which bits to compare in the field outermostVidType in **Egress VLAN Translation Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x1
33	outermostVidType_mask_large	Which bits to compare in the field outermostVidType **Egress VLAN Translation Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x1

38.8.47 Egress VLAN Translation Selection

This register selects which result to use when there are multiple hits.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	1182082

Field Description

Bits	Field Name	Description	Default Value
0	selectTcamOrTable	If set to zero then TCAM answer is selected. If set to one then hash table answer is selected.	0x0
1	selectSmallOrLarge	If set to zero then small hash table is selected. If set to one then large hash table is selected.	0x0

38.8.48 Egress VLAN Translation Small Table

The outermost VID and VID Ethernet Type (Service tag or Customer tag types) of the outgoing packet is compared.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries :	256
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	address[5:0] : hash of { dstPort outermostVid outermostVidType }
	address[7:6] : bucket number
Address Space :	1147142 to 1147653

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
4:1	dstPort	This is a field which is used as search data. The destination port which the packet is going out on	0x0
16:5	outermostVid	This is a field which is used as search data. The outermost VID of the modified packet.	0x0
17	outermostVidType	This is a field which is used as search data. The outermost VID is a S-tag or C-Tag. 0 = Customer tag 1 = Service tag	0x0
29:18	newVid	This is a result field used when this entry is hit. The new VID for the outgoing packet.	0x0
45:30	ethType	This is a result field used when this entry is hit. The new Ethernet Type for the outgoing packet	0x0

38.8.49 Egress VLAN Translation TCAM

The outermost VID and VID Ethernet Type (Service tag or Customer tag types) of the outgoing packet is compared.

Number of Entries :	16
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	1182168 to 1182199

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
4:1	dstPort_mask	Mask for dstPort.	0xf
8:5	dstPort	The destination port which the packet is going out on	0x0
20:9	outermostVid_mask	Mask for outermostVid.	0xfff
32:21	outermostVid	The outermost VID of the modified packet.	0x0
33	outermostVidType_mask	Mask for outermostVidType.	0x1
34	outermostVidType	The outermost VID is a S-tag or C-Tag. 0 = Customer tag 1 = Service tag	0x0

38.8.50 Egress VLAN Translation TCAM Answer

This is the table holding the answer for the **Egress VLAN Translation TCAM**.

Number of Entries : 16
Type of Operation : Read/Write
Addressing : **Egress VLAN Translation TCAM** hit index
Address Space : 1147654 to 1147669

Field Description

Bits	Field Name	Description	Default Value
11:0	newVid	The new VID for the outgoing packet.	0x0
27:12	ethType	The new Ethernet Type for the outgoing packet	0x0

38.8.51 IP QoS Mapping Table

Set the outgoing packets PCP and CFI values for the outermost VLAN ID and ECN bits in the TOS Byte if selected from **Select Which Egress QoS Mapping Table To Use**. The rest of the TOS bits comes from the coloring mapping or MMP mapping tables.

Number of Entries : 256
Type of Operation : Read/Write

Addressing :

Address [2:0] :	The egress queue which the packet was queued on.
Address [4:3]:	The color of the packet.
Address [6:5] :	The ECN ToS bits TOS[1:0] after coloring operation.
Address [7] :	The Pointer from the **Select Which Egress QoS Mapping Table To Use whichTablePtr**.

Address Space : 1180758 to 1181013

Field Description

Bits	Field Name	Description	Default Value
0	updateCfiDei	Update CfiDei field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
1	cfiDei	Packets new CFI/DEI	0x0
2	updatePcp	Update Pcp field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
5:3	pcp	Packets new PCP	0x0
7:6	ecnTos	The outgoing TOS [1:0] ECN bits	0x0
8	updateExp	If the packet enterns a new MPLS tunnel using the **Next Hop Packet Insert MPLS Header** then use this Exp for the outermost MPLS label. 0 = No. Dont Remap. 1 = Yes. Remap to this new value	0x0
11:9	newExp	New Exp value to be used.	0x0

38.8.52 Ingress NAT Operation

Ingress NAT Operation Table.

Number of Entries :	8192
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	Ingress ACL NAT Pointer plus egress port number.
Address Space :	1147670 to 1164053

Field Description

Bits	Field Name	Description	Default Value
0	replaceSrc	Replace Source or Destination. 0 = Destiantion 1 = Source	0x0
1	replaceIP	Replace IP address. 0 = No. 1 = Yes.	0x0
2	replaceL4Port	Replace TCP/UDP port. 0 = No. 1 = Yes.	0x0
34:3	ipAddress	The new IP Address.	0x0
50:35	port	The new L4 Port.	0x0
53:51	entryVersion	The version of this entry.Must be same version as the entry pointing to it.	0x0

38.8.53 L2 QoS Mapping Table

Set the outgoing packets PCP and CFI values for the outermost VLAN ID if selected from **Select Which Egress QoS Mapping Table To Use**.

Number of Entries : 64
Type of Operation : Read/Write

Addressing :

Address [2:0] :	The egress queue which the packet was queued on.
Address [4:3]:	The color of the packet.
Address [5] :	The Pointer from the **Select Which Egress QoS Mapping Table To Use whichTablePtr**.

Address Space : 1180694 to 1180757

Field Description

Bits	Field Name	Description	Default Value
0	updateCfiDei	Update CfiDei field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
1	cfiDei	Packets new CFI/DEI.	0x0
2	updatePcp	Update Pcp field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
5:3	pcp	Packets new PCP.	0x0

38.8.54 L2 Tunnel Entry Instruction Table

The is the L2 tunnel entry instruction which described how a tunnel entry should be done after the L2 MAC and VLAN headers in the packet. If the L3Type is either IPv4, IPv6 then the length fields are updated in the IP headers, for IPv4 the checksum is re-calculated. If the hasUDP is turned on then the UDP length-field is updated.

Number of Entries : 32
Type of Operation : Read/Write
Addressing : Tunnel entry pointer
Address Space : 1131480 to 1131511

Field Description

Bits	Field Name	Description	Default Value
1:0	l3Type	Insert header type. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Other.	0x0
2	hasUdp	If the header is a IPv4 or IPv6 then a insert an UDP header after IP header.	0x0
3	updateEtherType	Shall the Ethernet Type be updated. 0 = No 1 = Yes	0x0
19:4	outerEtherType	EtherType preceding the tunnel entry point.	0x0

38.8.55 L3 Tunnel Entry Instruction Table

The is the L3 tunnel entry instruction which described how a tunnel entry should be done after the L3 IPv4/IPv6/MPLS headers in the packet.

Number of Entries : 32
Type of Operation : Read/Write
Addressing : Tunnel entry pointer
Address Space : 1131512 to 1131543

Field Description

Bits	Field Name	Description	Default Value
1:0	updateL4Type	If the packet is a IPv4 or IPv6 then the Next Header/Protocol field shall be updated. IPv4 Packet will see a updated header checksum.	0x0
9:2	l4Protocol	The new Next Header/Protocol byte	0x0

38.8.56 MACsec Vlan

MACsec can encrypt everything in the header including the VLANs. This setting enables up to 2 VLANs to be visible on the port before the MACsec header. Default is that all VLAN headers will be after the MACsec header, and thus encrypted.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182085

Field Description

Bits	Field Name	Description	Default Value
1:0	nrVlans	After how many VLANs shall the MACsec header start.	0x0

38.8.57 MPLS QoS Mapping Table

Set the outgoing packets PCP and CFI values for the outermost VLAN ID and outermost EXP MPLS label if selected from **Select Which Egress QoS Mapping Table To Use**.

Number of Entries : 512
Type of Operation : Read/Write
Addressing :

Address [2:0] :	The egress queue which the packet was queued on.
Address [4:3]:	The color of the packet.
Address [7:5] :	The outermost label EXP bits.
Address [8] :	The Pointer from the **Select Which Egress QoS Mapping Table To Use whichTablePtr**.

Address Space : 1181526 to 1182037

Field Description

Bits	Field Name	Description	Default Value
0	updateCfiDei	Update CfiDei field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
1	cfiDei	Packets new CFI/DEI.	0x0
2	updatePcp	Update Pcp field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
5:3	pcp	Packets new PCP.	0x0
8:6	exp	The outgoing Exp value for this queue in the outermost MPLS label.	0x0

38.8.58 NAT Add Egress Port for NAT Calculation

Should the ingress and egress NAT pointers from the ingress and egress ACL be added with the egress port number.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1182083

Field Description

Bits	Field Name	Description	Default Value
0	dontAddIngress	Do not add egress port when calculating the ingress NAT offset pointer. 0 = Add Egress Port. 1 = Do not add Egress Port.	0x0
1	dontAddEgress	Do not add egress port when calculating the egress NAT offset pointer. 0 = Add Egress Port. 1 = Do not add Egress Port.	0x0

38.8.59 Next Hop DA MAC

Determines the destination MAC address to use in the packet exiting the router.

Number of Entries : 2048
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Addressing : **nextHopPacketMod**
Address Space : 1131698 to 1135793

Field Description

Bits	Field Name	Description	Default Value
47:0	daMac	The destination MAC address for the next hop.	0x0

Bits	Field Name	Description	Default Value
48	useSaTable	Instead of using the DA for the outgoing SA when a packet is being routed the result from the result from pointed from this table to **Router MAC SA Table** 0 = No. 1 = Yes.	0x0
53:49	saNextHopPtr	Pointer to the **Router MAC SA Table**	0x0
56:54	entryVersion	The version of this entry. Must be same version as the entry pointing to it.	0x0

38.8.60 Next Hop MPLS Table

Determines the MPLS tag operation to perform.

Number of Entries :	2048
Type of Operation :	Read/Write
Addressing :	**nextHopPacketMod**
Address Space :	1135866 to 1137913

Field Description

Bits	Field Name	Description	Default Value
2:0	mplsOperation	The egress MPLS tag operation to perform on the packet. 0 = No operation. 1 = Swap. 2 = Push. 3 = Pop. 4 = Penultimate Pop(remove all MPLS tags).	0x0
4:3	expSel	Select which EXP bits to use when building a new MPLS tag in Push or Swap operation. 0 = From this entries EXP field. 1 = From egress queue remapping in **Egress Queue To MPLS EXP Mapping Table** 2 = From the MPLS label (outermost MPLS tag if a swap and innermost if a push.	0x0
7:5	exp	Value to use for the EXP field when building a new MPLS tag in a swap or push operation.	0x0
27:8	label	MPLS label to use when building a new MPLS tag in a swap or push operation.	0x0
30:28	entryVersion	The version of this entry. Must be same version as the entry pointing to it.	0x0

38.8.61 Next Hop Packet Insert MPLS Header

Shall MPLS lables (up tp 4) be inserted on the packet before it is sent out. This enables a IP packet to go into a MPLS tunnel. Header is placed after L2 and VLANs before the IP packet header. MPLS EXP field comes from destination queue to EXP mapping table defined in **Egress Queue To MPLS EXP Mapping Table**. Only the lowest entries from 0 to 1024-1 in the

ieldNext Hop TablenextHopPacketMod can be used to insert a MPLS header.

Number of Entries : 1024
Number of Addresses per Entry : 8
Type of Operation : Read/Write
Addressing : **nextHopPacketMod** bits [9 : 0]
Address Space : 1137918 to 1146109

Field Description

Bits	Field Name	Description	Default Value
2:0	howManyLabelsToInsert	How many labels shall be inserted. Setting a zero here means no labels will be added.	0x0
3	whichEthernetType	Which Ethernet Type shall be used for these MPLS labels. 0 = 0x8847 1 = 0x8848	0x0
23:4	mplsLabel0	First/Outermost MPLS label to be inserter.	0x0
24	copyTtl0	Where shall the TTL come from in the MPLS label 0. 0 = From this table, field ttl0. 1 = From the inner packet.	0x0
32:25	ttl0	TTL table value for MPLS label 0.	0x0
33	expFromQueue0	Where shall the EXP come from in the MPLS label 0. 0 = From this table, field exp0. 1 = From the **Egress Queue To MPLS EXP Mapping Table**.	0x0
36:34	exp0	EXP table value for MPLS label 0.	0x0
56:37	mplsLabel1	MPLS label 1 to be inserter.	0x0
57	copyTtl1	Where shall the TTL come from in the MPLS label 1. 0 = From this table, field ttl1. 1 = From the inner packet.	0x0
65:58	ttl1	TTL table value for MPLS label 1.	0x0
66	expFromQueue1	Where shall the EXP come from in the MPLS label 1. 0 = From this table, field exp1. 1 = From the **Egress Queue To MPLS EXP Mapping Table**.	0x0
69:67	exp1	EXP table value for MPLS label 1.	0x0
89:70	mplsLabel2	MPLS label 2 to be inserter.	0x0
90	copyTtl2	Where shall the TTL come from in the MPLS label 2. 0 = From this table, field ttl2. 1 = From the inner packet.	0x0
98:91	ttl2	TTL table value for MPLS label 2.	0x0
99	expFromQueue2	Where shall the EXP come from in the MPLS label 2. 0 = From this table, field exp2. 1 = From the **Egress Queue To MPLS EXP Mapping Table**.	0x0
102:100	exp2	EXP table value for MPLS label 2.	0x0
122:103	mplsLabel3	MPLS label 3 to be inserter.	0x0

Bits	Field Name	Description	Default Value
123	copyTtl3	Where shall the TTL come from in the MPLS label 3. 0 = From this table, field ttl3. 1 = From the inner packet.	0x0
131:124	ttl3	TTL table value for MPLS label 3.	0x0
132	expFromQueue3	Where shall the EXP come from in the MPLS label 3. 0 = From this table, field exp3. 1 = From the **Egress Queue To MPLS EXP Mapping Table**.	0x0
135:133	exp3	EXP table value for MPLS label 3.	0x0
138:136	entryVersion	The version of this entry. Must be same version as the entry pointing to it.	0x0

38.8.62 Output Mirroring Table

Output mirroring configuration. An egress port can be set to have a mirrored port, but output mirroring cannot link more than one port. i.e. If Port A has an output mirroring Port B, Port B has an output mirroring Port C, packets sent to port A will not be mirrored to Port C.

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 1182038 to 1182048

Field Description

Bits	Field Name	Description	Default Value
0	outputMirrorEnabled	If set to one, output mirroring is enabled for this port.	0x0
4:1	outputMirrorPort	Destination of output mirroring. Only valid if outputMirrorEnabled is set. Notice if the design contains more than one switch slice, packets egressed on one slice cannot be mirrored to another slice.	0x0

38.8.63 Router MAC SA Table

This table holds a SA MAC address to be used if a packet is routed. The addresses are selected by setting the **useSaTable** to one and pointing to the entry to be used in this table using the **saNextHopPtr** field.

Number of Entries : 32
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Addressing : **saNextHopPtr**
Address Space : 1135802 to 1135865

Field Description

Bits	Field Name	Description	Default Value
47:0	newSa	The MAC SA to be used for a routed packet.	0x0

38.8.64 Router Port Egress SA MAC Address

The routers SA MAC address to use when a packet exits the router. In normal cases this would be the incoming Destination MAC address. However when using NAT there are cases which this does not work and hence this table allows the usage of a alternative MAC address.

Number of Entries : 4
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Addressing : VRF
Address Space : 1135794 to 1135801

Field Description

Bits	Field Name	Description	Default Value
10:0	selectMacEntryPortMask	Portmask to select which SA MAC address to use as router MAC address. One bit per destination port. 0 = use incoming packets DA MAC address. 1 = use altMacAddress.	0x0
58:11	altMacAddress	The alternative base destination MAC address that is used to identify packets to the router.	0x0

38.8.65 Select Which Egress QoS Mapping Table To Use

This is the initial table which is looked up by all packets in order to determine how the mapping from internal QoS to packets final PCP, DEI, TOS/EXP field shall look like. In order for this table to be executed the field **useEgressQueueRemapping** must be set to one.

Number of Entries : 256
Type of Operation : Read/Write
Addressing :

Address Bit [1:0]:	Forwarding type to this port. 0 = Switched Packet 1 = Routed Packet 2 = Classification Rule Forwarded Packet 3 = Others - Send-to-CPU and packet from CPU
Address Bit [3:2]:	Packet type 0 = L2 - Not IPv4/IPv6/MPLS 1 = IPv4 2 = IPv6 3 = MPLS
Address Bit [8:4]:	Egress Port

Address Space : 1180438 to 1180693

Field Description

Bits	Field Name	Description	Default Value
2:0	whichTableToUse	Select which table type to use. 0 = None. No remapping 1 = **L2 QoS Mapping Table** 2 = **IP QoS Mapping Table** 3 = **TOS QoS Mapping Table** 4 = **MPLS QoS Mapping Table** 5 = Use this tables remapping of DEI and PCP bits.	0x0
3	whichTablePtr	Which index of the tables to use. For most QoS tables there exists multiple tables to choose from.	0x0
4	updateCfiDei	Update CfiDei field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
5	cfiDei	Packets new CFI/DEI.	0x0
6	updatePcp	Update Pcp field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
9:7	pcp	Packets new PCP.	0x0

38.8.66 TOS QoS Mapping Table

Set the outgoing packets PCP and CFI values for the outermost VLAN ID and TOS Byte if selected from **Select Which Egress QoS Mapping Table To Use**. The input TOS byte to this mapping table comes from the coloring or MMP mapping tables.

Number of Entries : 512

Type of Operation : Read/Write

Addressing :

Address [7:0] :	The TOS byte.
Address [8] :	The Pointer from the **Select Which Egress QoS Mapping Table To Use whichTablePtr**.

Address Space : 1181014 to 1181525

Field Description

Bits	Field Name	Description	Default Value
0	updateCfiDei	Update CfiDei field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
1	cfiDei	Packets new CFI/DEI	0x0
2	updatePcp	Update Pcp field in outgoing packet. 0 = Do not update. 1 = Update.	0x0
5:3	pcp	Packets new PCP	0x0
13:6	newTos	The outgoing new TOS bits	0x0
14	updateExp	If the packet enterns a new MPLS tunnel using the **Next Hop Packet Insert MPLS Header** then use this Exp for the outermost MPLS label. 0 = No. Dont Remap. 1 = Yes. Remap to this new value	0x0
17:15	newExp	New Exp value to be used.	0x0

38.8.67 Tunnel Entry Header Data

The this is the byte data which is used to do tunnel insertions. The data to be used is pointed to from the **Tunnel Entry Instruction Table**

Number of Entries :	32
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Addressing :	**tunnelHeaderPtr**
Address Space :	1130936 to 1131447

Field Description

Bits	Field Name	Description	Default Value
511:0	data	Tunnel header data (bytes) to be inserted at tunnel entry point in packet.Byte 0 is the start of the tunnel header.	0x0

38.8.68 Tunnel Entry Instruction Table

The tunnel entry instruction describes how a tunnel shall be entered. The same pointer address is used to read out the **Beginning of Packet Tunnel Entry Instruction Table** , **L2 Tunnel Entry Instruction Table** and **L3 Tunnel Entry Instruction Table**. The field tunnelEntryType determine which tunnel entry table to use. The insertion of the length field is independent from the other tunnel header length updates which is done.

Number of Entries :	32
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	Tunnel Entry Pointer from various tables
Address Space :	1130872 to 1130935

Field Description

Bits	Field Name	Description	Default Value
1:0	tunnelEntryType	A tunnel entry shall be done. Where shall the tunnel entry be done 0 = At Byte Zero described in **Beginning of Packet Tunnel Entry Instruction Table** 1 = After L2 and up to two VLAN headers. described in **L2 Tunnel Entry Instruction Table** 2 = After L3 IPv4/IPv6/MPLS headers. 3 = Reserved.	0x0
2	insertLength	Insert the a packet length fields. The 2 byte length of the frame will overwrite current 2 bytes in the header data to be inserted at **lengthPos**. 0 = Yes. Insert a length field. 1 = No. Don't insert a length field.	0x0
8:3	lengthPos	If length shall be inserted , where shall it be inserted. A value of 0 means beginning of tunnel entry data.	0x0
22:9	lengthNegOffset	How much shall be decremented from the total packet (frame) length.	0x0

Bits	Field Name	Description	Default Value
36:23	lengthPosOffset	How much shall be incremented from the total packet (frame) length.	0x0
37	incVlansInLength	Should the outgoing packets number of VLANs be included in the length calculation? 0 = No. 1 = Yes.	0x0
42:38	tunnelHeaderPtr	Points to which header to insert from register **Tunnel Entry Header Data**.	0x0
48:43	tunnelHeaderLen	The length of the tunnel header, in bytes, to insert from register **Tunnel Entry Header Data**.	0x0

38.9 Flow Control

38.9.1 FFA Used PFC

Total number of cells from the common pool used by ports in PFC-mode.

Number of Entries : 1
Type of Operation : Read Only
Address Space : 1129423

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Number of cells	0x0

38.9.2 FFA Used non-PFC

Total number of cells used from the common pool by ports in non-PFC mode.

Number of Entries : 1
Type of Operation : Read Only
Address Space : 1129424

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Number of cells	0x0

38.9.3 PFC Dec Counters for ingress ports 0 to 11

Wrapping counters of deallocated cells. The number of currently used cells is the allocated minus the deallocated modulo the counter size.

Number of Entries : 96
Type of Operation : Read Only
Addressing : 8*(Source port) + Traffic class
Address Space : 1129295 to 1129390

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Number of cells	0x0

38.9.4 PFC Inc Counters for ingress ports 0 to 11

Wrapping counters of allocated cells. The number of currently used cells is the allocated minus the deallocated modulo the counter size.

Number of Entries : 96
Type of Operation : Read Only
Addressing : 8*(Source port) + Traffic class
Address Space : 1129199 to 1129294

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Number of cells	0x0

38.9.5 Port FFA Used

Number of cells used from the common pool for this source port

Number of Entries : 12
Type of Operation : Read Only
Addressing : Source port
Address Space : 1129391 to 1129402

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Number of cells	0x0

38.9.6 Port Pause Settings

Pause settings per source port.

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Source port
Address Space : 1129425 to 1129436

Field Description

Bits	Field Name	Description	Default Value
0	enable	0 = Pausing disabled 1 = Pausing enabled	0x0
1	mode	On a port where both pausing and tail-drop is enabled the modes must match for the calculation of used FFA to be correct. 0 = Priority mode 1 = Port mode	0x0
3:2	reserved	Reserved.	0x0
11:4	force	Each bit refers to one traffic class (bit 0 = TC 0) 0 = No force 1 = Force the pause state to that set in the pattern field Only valid if pausing is enabled.	0x0
19:12	pattern	Each bit refers to one traffic class (bit 0 = TC 0) 0 = Not paused 1 = Paused	0x0

38.9.7 Port Reserved

Number of cells reserved in the buffer memory for this source port. Shall be set to zero for prio-mode ports Note that this setting can only be changed for an empty port.

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Source port
Address Space : 1129187 to 1129198

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Number of cells	0x9

38.9.8 Port Tail-Drop FFA Threshold

Settings for the Port Tail-Drop FFA Threshold

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Source port
Address Space : 1129500 to 1129511

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Tail-drop threshold in number of cells. When the FFA cells used by the source port reaches this threshold no further packets will be accepted for this source port	0x400
11	enable	0 = This tail-drop threshold is disabled 1 = This tail-drop threshold is enabled	0x0
12	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this tail-drop threshold is enabled.	0x0

38.9.9 Port Tail-Drop Settings

Tail-drop settings per source port.

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Source port
Address Space : 1129437 to 1129448

Field Description

Bits	Field Name	Description	Default Value
0	enable	0 = Tail-drop is disabled for this source port 1 = Tail-drop is enabled for this source port	0x0
1	mode	On a port where both pausing and tail-drop is enabled the modes must match for the calculation of used FFA to be correct. 0 = Priority mode 1 = Port mode	0x0

38.9.10 Port Used

Total number of cells used for this source port

Number of Entries : 12
Type of Operation : Read Only
Addressing : Source port
Address Space : 1129403 to 1129414

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Number of cells	0x0

38.9.11 Port Xoff FFA Threshold

Settings for Port Xoff FFA Threshold

Number of Entries :	12
Type of Operation :	Read/Write
Addressing :	Source port
Address Space :	1129488 to 1129499

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Xoff threshold for the number of used FFA cells for this source port	0x0
11	enable	0 = This Xoff threshold is disabled 1 = This Xoff threshold is enabled	0x0
12	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this Xoff threshold is enabled.	0x0

38.9.12 Port Xon FFA Threshold

Settings for Port Xon FFA Threshold

Number of Entries :	12
Type of Operation :	Read/Write
Addressing :	Source port
Address Space :	1129476 to 1129487

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Xon threshold for the number of used FFA cells for this source port	0x0

38.9.13 Port/TC Reserved

Number of cells reserved in the buffer memory for this source port and traffic class. For ports set to port-mode this should be 0 for all queues. Note that this setting can only be changed for an empty port.

Number of Entries :	96
Type of Operation :	Read/Write
Addressing :	8 * Source port + Traffic class
Address Space :	1129091 to 1129186

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Number of cells	0x0

38.9.14 Port/TC Tail-Drop Total Threshold

Settings for Port/TC Tail-Drop Total Threshold

Number of Entries : 96
Type of Operation : Read/Write
Addressing : 8 * Source port + Traffic class
Address Space : 1129704 to 1129799

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Tail-drop threshold in number of cells. When the sum of reserved and FFA cells used by this specific source port and traffic class combination reaches this threshold no further packets will be accepted for this source port and traffic class	0x400
11	enable	0 = This tail-drop threshold is disabled 1 = This tail-drop threshold is enabled	0x0
12	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this tail-drop threshold is enabled.	0x0

38.9.15 Port/TC Xoff Total Threshold

Settings for Port/TC Xoff Total Threshold

Number of Entries : 96
Type of Operation : Read/Write
Addressing : 8 * Source port + Traffic class
Address Space : 1129608 to 1129703

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Xoff threshold for the sum of reserved and FFA cells used for this source port and traffic class combination	0x0
11	enable	0 = This Xoff threshold is disabled 1 = This Xoff threshold is enabled	0x0
12	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this Xoff threshold is enabled.	0x0

38.9.16 Port/TC Xon Total Threshold

Settings for Port/TC Xon Total Threshold

Number of Entries : 96
Type of Operation : Read/Write
Addressing : 8 * Source port + Traffic class
Address Space : 1129512 to 1129607

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Xon threshold for the sum of reserved and FFA cells used for this source port and traffic class combination	0x0

38.9.17 TC FFA Used

Number of cells used from the common pool for this traffic class.

Number of Entries : 8
Type of Operation : Read Only
Addressing : Traffic class
Address Space : 1129415 to 1129422

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Number of cells	0x0

38.9.18 TC Tail-Drop FFA Threshold

Settings for TC Tail-Drop FFA Threshold

Number of Entries : 8
Type of Operation : Read/Write
Addressing : Traffic class
Address Space : 1129468 to 1129475

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Tail-drop threshold in number of cells. When the FFA cells used by the traffic class reaches this threshold no further packets will be accepted for this traffic class	0x400
11	enable	0 = This tail-drop threshold is disabled 1 = This tail-drop threshold is enabled	0x0

Bits	Field Name	Description	Default Value
12	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this tail-drop threshold is enabled.	0x0

38.9.19 TC Xoff FFA Threshold

Settings for TC Xoff FFA Threshold

Number of Entries : 8
Type of Operation : Read/Write
Addressing : Traffic class
Address Space : 1129460 to 1129467

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Xoff threshold for the number of used FFA cells for this traffic class	0x0
11	enable	0 = This Xoff threshold is disabled 1 = This Xoff threshold is enabled	0x0
12	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this Xoff threshold is enabled.	0x0

38.9.20 TC Xon FFA Threshold

Settings for TC Xon FFA Threshold

Number of Entries : 8
Type of Operation : Read/Write
Addressing : Traffic class
Address Space : 1129452 to 1129459

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Xon threshold for the number of used FFA cells for this traffic class	0x0

38.9.21 Tail-Drop FFA Threshold

Settings for Tail-Drop FFA Threshold

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1129451

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Tail-drop threshold in number of cells. When the total number of FFA cells used reaches this threshold no further packets will be accepted.	0x394
11	enable	0 = This tail-drop threshold is disabled 1 = This tail-drop threshold is enabled	0x0
12	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this tail-drop threshold is enabled.	0x0

38.9.22 Xoff FFA Threshold

Settings for Xoff FFA Threshold

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1129450

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Xoff threshold for the total number of used FFA cells	0x0
11	enable	0 = This Xoff threshold is disabled 1 = This Xoff threshold is enabled	0x0
12	trip	0 = Normal operation 1 = Force this threshold to be counted as exceeded Only valid if this Xoff threshold is enabled.	0x0

38.9.23 Xon FFA Threshold

Settings for Xon FFA Threshold

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1129449

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Xon threshold for the total number of used FFA cells	0x0

38.10 Global Configuration

38.10.1 Core Tick Configuration

Global register for setting the frequency of the core tick

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 2

Field Description

Bits	Field Name	Description	Default Value
19:0	clkDivider	The master Core Tick will be issued once every $rg\_tick\_div.clkDivider/4$ core clock cycles. If set to zero, there will be no tick.	0x271
23:20	stepDivider	The four ticks derived from the master core tick are issued once every $rg\_tick\_div.stepDivider^{tick\_number+1}$ master ticks. The master tick is tick number 0. If stepDivider is set to zero, there will be no ticks except possibly the master tick.	0xa

38.10.2 Core Tick Select

Global register for setting clock input to the core tick divider

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 3

Field Description

Bits	Field Name	Description	Default Value
1:0	clkSelect	Select the source clock for the Core Tick divider. 0: disabled, 1: core clock, 2: debug_write_data[0], 3: reserved	0x1

38.10.3 MAC RX Maximum Packet Length

Packets with length above this value will be dropped.

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Ingress Port
Address Space : 72 to 83

Field Description

Bits	Field Name	Description	Default Value
31:0	bytes	Number of bytes.	0x4003

38.10.4 Scratch

Scratch Register

Number of Entries : 1
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 4

Field Description

Bits	Field Name	Description	Default Value
63:0	scratch	scratch field.	0x0

38.11 Ingress Packet Processing

38.11.1 AH Header Packet Decoder Options

The L4 protocol number which is used to determine if the packet has a Authentical Header, the underlaying packet must be a IPv4 or IPv6 packet.. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121221

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
8:1	l4Proto	The value to be used to find this packet type.	0x33
19:9	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
30:20	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

38.11.2 ARP Packet Decoder Options

The Ethernet type used to determine if a packet is a ARP packet.. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	1122483

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	eth	The value to be used to find this packet type.	0x806
27:17	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
38:28	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

38.11.3 Aging Data FIFO

This register exposes the output of a FIFO which is holding all aging requests from the aging unit. Under hardware aging writeback mode, the entry pushed to this FIFO is in sync with the FIB. If hardware aging writeback is turned off, the final aging decision should be issued from software injected learning packet and what is pushed to this FIFO is not updated to L2 tables.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read Only
Address Space :	16769

Field Description

Bits	Field Name	Description	Default Value
7:0	hashClearValid	One bit per bucket, each bit set to 1 means the aging unit has requested to change the corresponding hash bucket valid bit from 1 to 0 hence clear out this entry.	0x0
15:8	hashClearHit	One bit per bucket, each bit set to 1 means the aging unit has requested to change corresponding hash bucket hit bit from 1 to 0.	0x0
26:16	hashValue	Hash of GID, MAC.	0x0
31:27	reserved	Reserved.	0x0
32	camClearValid	When this field is 1, the aging unit has requested to change the corresponding cam entry valid bit from 1 to 0 hence clear out this entry.	0x0

Bits	Field Name	Description	Default Value
33	camClearHit	When this field is 1, the aging unit has requested to change the corresponding cam entry hit bit from 1 to 0.	0x0
38:34	camIndex	Index to the entry in **L2 Aging Collision Table**.	0x0
39	valid	0 = Empty FIFO, entry is not valid 1 = Valid entry	0x0

38.11.4 Aging Data FIFO High Watermark Level

The High Watermark Interrupt will occur when a push to **Aging Data FIFO** is done and the number of existing entries after the push is larger than this setting.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 322

Field Description

Bits	Field Name	Description	Default Value
5:0	level	Number of used entries.	0x0

38.11.5 Allow Special Frame Check For L2 Action Table

The result in **L2 Action Table** is a pointer field **allowPtr** which allows result from the L2 SA Action Table to setup rules of which types of packets/frames are allowed to be sent in on a port. If any of there is a match and packet is not allowed then all instances are dropped of this packet. The drop counter **L2 Action Table Special Packet Type Drop** is updated.

Number of Entries : 4
Type of Operation : Read/Write
Addressing : Result from **L2 Action Table**
Address Space : 1119455 to 1119458

Field Description

Bits	Field Name	Description	Default Value
0	dontAllowBPDU	Allow BPDU frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
1	dontAllow8021X_EAPOL	Allow 802.1X EAPOL frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
2	dontAllowCAPWAP	Allow CAPWAP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0

Bits	Field Name	Description	Default Value
3	dontAllowARP	Allow ARP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
4	dontAllowRARP	Allow RARP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
5	dontAllowDNS	Allow DNS frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
6	dontAllowBOOTP_DHCP	Allow BOOTP_DHCP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
7	dontAllowSCTP	Allow STCP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
8	dontAllowLLDP	Allow LLDP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
9	dontAllowGRE	Allow GRE frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
10	dontAllowESP	Allow ESP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
11	dontAllowAH	Allow AH frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
12	dontAllowL2_1588	Allow L2 1588 frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
13	dontAllowL4_1588	Allow L4 1588 frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
14	dontAllowICMP	Allow ICMP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
15	dontAllowIGMP	Allow IGMP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
16	dontAllowL2McReserved	Allow L2 Reserved Da frames, see register **L2 Reserved Multicast Address Base**. 0 = Allow frame. 1 = Do not allow frame.	0x0
17	dontAllowIPV4	Allow IPV4 frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
18	dontAllowIPV6	Allow IPV6 frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
19	dontAllowUDP	Allow UDP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0

Bits	Field Name	Description	Default Value
20	dontAllowTCP	Allow TCP frames. 0 = Allow frame. 1 = Do not allow frame.	0x0
21	dontAllowMPLS	Allow MPLS frames. 0 = Allow frame. 1 = Do not allow frame.	0x0

38.11.6 BOOTP and DHCP Packet Decoder Options

The UDP port 1 number used by the BOOTP protocol, the underlaying packet must be a IPv4 packet. If L4 Source Port is this value then L4 Destination Port must be egisterbootpUdpPort2 value and vice versa. . If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	1122495

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	udp1	The value to be used to find this packet type.	0x43
32:17	udp2	The value to be used to find this packet type.	0x44
43:33	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
54:44	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

38.11.7 CAPWAP Packet Decoder Options

The fields needs to determine if a packet is a CAPWAP packet the underlaying packet must be a IPv4 or IPv6 packet. . If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	1122497

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	udp1	The value to be used to find this packet type.	0x147e
32:17	udp2	The value to be used to find this packet type.	0x147f
43:33	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
54:44	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

38.11.8 CPU Reason Code Operation

When a packet raises a send to CPU action during the ingress packet process, follow-up operations can be performed based on the reason code. In this table 16 ranges are searched in order and the same action hit in the latter range overrides the previous hit.

Number of Entries :	16
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	1121275 to 1121306

Field Description

Bits	Field Name	Description	Default Value
0	mutableCpu	Force the packet to another port instead of the CPU port when the CPU reason code hit in the range.	0x0
4:1	port	The new destination to replace the CPU port.	0x0
5	forceQueue	Force the packet to the CPU port with a new egress queue when the CPU reason code hit in the range.	0x0
8:6	eQueue	Egress queue	0x0
9	forceUpdateOrigCpuPkt	If this reason code is hit shall the origCpuPkt field be updated? 0 = No, no update. 1 = Yes, update.	0x0
10	origCpuPkt	Force the packet to the CPU to be the orig-inial,unmodified, packet. 0 = No, modification will happen to packet. 1 = Yes, force the packet to be unmodified.	0x0
26:11	start	Start of CPU reason code.	0x0
42:27	end	End of CPU reason code.	0x0

38.11.9 Check IPv4 Header Checksum

This register provides an option to drop the IPv4 packet if its header checksum field has an incorrect value.The option is only for not routed IPv4 packet. For a routed IPv4 packet, the checksum check is

always performed.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121232

Field Description

Bits	Field Name	Description	Default Value
0	dropErrorChkSum	If set, always calculate the checksum of the received IPv4 packet. If the calculated value does not match the IPv4 checksum field, the packet is dropped.	0x0

38.11.10 DNS Packet Decoder Options

The TCP/UDP destination port number used to determine if a packet is a DNS packet, the underlaying packet must be a IPv4 or IPv6 packet.. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries : 1
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 1122493

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	l4Port	The value to be used to find this packet type.	0x35
27:17	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
38:28	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

38.11.11 Debug Counter debugMatchIPP0 Setup

Packet processing debug setup for
registerDebug debugMatchIPP0.

Number of Entries : 1
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 1122507

Field Description

Bits	Field Name	Description	Default Value
21:0	mask	Mask for comparison to update debug counter.	0x0
43:22	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.11.12 Debug Counter dstPortmask Setup

Packet processing debug setup for
registerDebug dstPortmask.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121248

Field Description

Bits	Field Name	Description	Default Value
10:0	mask	Mask for comparison to update debug counter.	0x0
21:11	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.11.13 Debug Counter finalVid Setup

Packet processing debug setup for
registerDebug finalVid.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121227

Field Description

Bits	Field Name	Description	Default Value
12:0	mask	Mask for comparison to update debug counter.	0x0
25:13	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.11.14 Debug Counter l2DaHash Setup

Packet processing debug setup for
registerDebug l2DaHash.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121237

Field Description

Bits	Field Name	Description	Default Value
10:0	mask	Mask for comparison to update debug counter.	0x0
21:11	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.11.15 Debug Counter l2DaHashHitAndBucket Setup

Packet processing debug setup for
registerDebug l2DaHashHitAndBucket.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121238

Field Description

Bits	Field Name	Description	Default Value
3:0	mask	Mask for comparison to update debug counter.	0x0
7:4	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.11.16 Debug Counter l2DaHashKey Setup

Packet processing debug setup for
registerDebug l2DaHashKey.

Number of Entries : 1
Number of Addresses per Entry : 4
Type of Operation : Read/Write
Address Space : 1122675

Field Description

Bits	Field Name	Description	Default Value
59:0	mask	Mask for comparison to update debug counter.	0x0
119:60	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.11.17 Debug Counter l2DaTcamHitsAndCast Setup

Packet processing debug setup for
registerDebug l2DaTcamHitsAndCast.

Number of Entries : 1
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 1122501

Field Description

Bits	Field Name	Description	Default Value
16:0	mask	Mask for comparison to update debug counter.	0x0
33:17	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.11.18 Debug Counter nextHopPtrFinal Setup

Packet processing debug setup for
registerDebug nextHopPtrFinal.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121231

Field Description

Bits	Field Name	Description	Default Value
10:0	mask	Mask for comparison to update debug counter.	0x0
21:11	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.11.19 Debug Counter nextHopPtrHash Setup

Packet processing debug setup for
registerDebug nextHopPtrHash.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121230

Field Description

Bits	Field Name	Description	Default Value
10:0	mask	Mask for comparison to update debug counter.	0x0
21:11	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.11.20 Debug Counter nextHopPtrLpm Setup

Packet processing debug setup for
registerDebug nextHopPtrLpm.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121229

Field Description

Bits	Field Name	Description	Default Value
10:0	mask	Mask for comparison to update debug counter.	0x0
21:11	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.11.21 Debug Counter nrVlans Setup

Packet processing debug setup for
registerDebug nrVlans.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121219

Field Description

Bits	Field Name	Description	Default Value
1:0	mask	Mask for comparison to update debug counter.	0x0
3:2	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.11.22 Debug Counter spVidOp Setup

Packet processing debug setup for
registerDebug spVidOp.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121223

Field Description

Bits	Field Name	Description	Default Value
2:0	mask	Mask for comparison to update debug counter.	0x0
5:3	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.11.23 Debug Counter srcPort Setup

Packet processing debug setup for
registerDebug srcPort.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121216

Field Description

Bits	Field Name	Description	Default Value
3:0	mask	Mask for comparison to update debug counter.	0x0
7:4	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.11.24 Debug Counter vlanVidOp Setup

Packet processing debug setup for
registerDebug vlanVidOp.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121228

Field Description

Bits	Field Name	Description	Default Value
2:0	mask	Mask for comparison to update debug counter.	0x0
5:3	hitValue	Value to compare to update debug counter. Both the incoming value and this value is ANDed with the mask before comparsion is carried out. If comparsion results in true the counter is updated	0x0

38.11.25 Default Packet To CPU Modification

Shall packets which are sent to the CPU be modified or original incoming packets. If a packet is switch / routed the to the CPU port then it will come out as the modified packet. This register only is relevant when a packet is sent to the cpu using Send-to-CPU flag (ie. when reason code != 0).

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Source Port
Address Space : 1120681 to 1120691

Field Description

Bits	Field Name	Description	Default Value
0	origCpuPkt	Force the packet to the CPU to be the originial,unmodified, packet. The exception to this is rule is the tunnel exit which will still be carried out. 0 = No, modification will happen to packet. 1 = Yes, force the packet to be unmodified.	0x0

38.11.26 ESP Header Packet Decoder Options

The L4 protocol number which is used to determine if the packet has a Authentical Header, the underlaying packet must be a IPv4 or IPv6 packet.. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121222

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
8:1	l4Proto	The value to be used to find this packet type.	0x32
19:9	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
30:20	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

38.11.27 Egress ACL Rule Pointer Large Table

D-left search that determines which ACL rule pointer to use when building the search key for the egress ACL lookup.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries : 128
Number of Addresses per Entry : 2
Type of Operation : Read/Write

Addressing :

address[4:0]	:	hash of { destPortMask routed vrf flooded ucSwitched mcSwitched vid l3Type l4Type srcPort fromCrypto }
address[6:5]	:	bucket number

Address Space : 1044400 to 1044655

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
11:1	destPortMask	This is a field which is used as search data. The packets egress ports, one bit per port.	0x0
12	routed	This is a field which is used as search data. The packet was routed.	0x0
14:13	vrf	This is a field which is used as search data. The VRF used when routed.	0x0
15	flooded	This is a field which is used as search data. The packet was flooded due to L2 table miss.	0x0
16	ucSwitched	This is a field which is used as search data. The packet was L2 switched to a unicast destination port.	0x0
17	mcSwitched	This is a field which is used as search data. The packet was L2 switched to a multicast group.	0x0
29:18	vid	This is a field which is used as search data. The index used in the VLAN table lookup.	0x0
31:30	l3Type	This is a field which is used as search data. The packets L3 Type. abFourIPv4IPv6MPLSOther	0x0

Bits	Field Name	Description	Default Value
34:32	l4Type	This is a field which is used as search data. The packets L4 Type. abEightNot known.Is IPv4 or IPv6 but type is not any L4 type in this list.UDPTCPIGMPICMPICMPv6MLD	0x0
38:35	srcPort	This is a field which is used as search data. The packets source port.	0x0
39	fromCrypto	This is a field which is used as search data. The packet came from Crypto Engine.	0x0
42:40	rulePtr	This is a result field used when this entry is hit. Rule Pointer.	0x0

38.11.28 Egress ACL Rule Pointer Search Mask

Before the hashing and searching is done in the **Egress ACL Rule Pointer Large Table** and **Egress ACL Rule Pointer Small Table** The search data is AND:ed with this mask. If a bit in the mask is set to zero then this bit in the lookup will be viewed as do not care. Seperate masks exists for both small and large tables.

Number of Entries : 1
Number of Addresses per Entry : 4
Type of Operation : Read/Write
Address Space : 1122679

Field Description

Bits	Field Name	Description	Default Value
10:0	destPortMask_mask_small	Which bits to compare in the field destPortMask in **Egress ACL Rule Pointer Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x7ff
21:11	destPortMask_mask_large	Which bits to compare in the field destPortMask **Egress ACL Rule Pointer Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x7ff
22	routed_mask_small	Which bits to compare in the field routed in **Egress ACL Rule Pointer Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x1
23	routed_mask_large	Which bits to compare in the field routed **Egress ACL Rule Pointer Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x1
25:24	vrf_mask_small	Which bits to compare in the field vrf in **Egress ACL Rule Pointer Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x3

Bits	Field Name	Description	Default Value
27:26	vrf_mask_large	Which bits to compare in the field vrf **Egress ACL Rule Pointer Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x3
28	flooded_mask_small	Which bits to compare in the field flooded in **Egress ACL Rule Pointer Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x1
29	flooded_mask_large	Which bits to compare in the field flooded **Egress ACL Rule Pointer Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x1
30	ucSwitched_mask_small	Which bits to compare in the field ucSwitched in **Egress ACL Rule Pointer Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x1
31	ucSwitched_mask_large	Which bits to compare in the field ucSwitched **Egress ACL Rule Pointer Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x1
32	mcSwitched_mask_small	Which bits to compare in the field mcSwitched in **Egress ACL Rule Pointer Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x1
33	mcSwitched_mask_large	Which bits to compare in the field mcSwitched **Egress ACL Rule Pointer Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x1
45:34	vid_mask_small	Which bits to compare in the field vid in **Egress ACL Rule Pointer Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0xfff
57:46	vid_mask_large	Which bits to compare in the field vid **Egress ACL Rule Pointer Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0xfff
59:58	l3Type_mask_small	Which bits to compare in the field l3Type in **Egress ACL Rule Pointer Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x3

Bits	Field Name	Description	Default Value
61:60	l3Type_mask_large	Which bits to compare in the field l3Type **Egress ACL Rule Pointer Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x3
64:62	l4Type_mask_small	Which bits to compare in the field l4Type in **Egress ACL Rule Pointer Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x7
67:65	l4Type_mask_large	Which bits to compare in the field l4Type **Egress ACL Rule Pointer Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x7
71:68	srcPort_mask_small	Which bits to compare in the field srcPort in **Egress ACL Rule Pointer Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0xf
75:72	srcPort_mask_large	Which bits to compare in the field srcPort **Egress ACL Rule Pointer Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0xf
76	fromCrypto_mask_small	Which bits to compare in the field fromCrypto in **Egress ACL Rule Pointer Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x1
77	fromCrypto_mask_large	Which bits to compare in the field fromCrypto **Egress ACL Rule Pointer Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	0x1

38.11.29 Egress ACL Rule Pointer Small Table

D-left search that determines which ACL rule pointer to use when building the search key for the egress ACL lookup.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries : 64
Number of Addresses per Entry : 2
Type of Operation : Read/Write

Addressing :

address[3:0] :	hash of { destPortMask routed vrf flooded ucSwitched mcSwitched vid l3Type l4Type srcPort fromCrypto }
address[5:4] :	bucket number

Address Space : 1044656 to 1044783

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
11:1	destPortMask	This is a field which is used as search data. The packets egress ports, one bit per port.	0x0
12	routed	This is a field which is used as search data. The packet was routed.	0x0
14:13	vrf	This is a field which is used as search data. The VRF used when routed.	0x0
15	flooded	This is a field which is used as search data. The packet was flooded due to L2 table miss.	0x0
16	ucSwitched	This is a field which is used as search data. The packet was L2 switched to a unicast destination port.	0x0
17	mcSwitched	This is a field which is used as search data. The packet was L2 switched to a multicast group.	0x0
29:18	vid	This is a field which is used as search data. The index used in the VLAN table lookup.	0x0
31:30	l3Type	This is a field which is used as search data. The packets L3 Type. abFourIPv4IPv6MPLSOther	0x0
34:32	l4Type	This is a field which is used as search data. The packets L4 Type. abEightNot known.Is IPv4 or IPv6 but type is not any L4 type in this list.UDPTCPIGMPICMPICMPv6MLD	0x0
38:35	srcPort	This is a field which is used as search data. The packets source port.	0x0
39	fromCrypto	This is a field which is used as search data. The packet came from Crypto Engine.	0x0
42:40	rulePtr	This is a result field used when this entry is hit. Rule Pointer.	0x0

38.11.30 Egress ACL Rule Pointer TCAM

D-left search that determines which ACL rule pointer to use when building the search key for the egress ACL lookup.

Number of Entries :	16
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	1122555 to 1122618

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
11:1	destPortMask_mask	Mask for destPortMask.	0x7ff
22:12	destPortMask	The packets egress ports, one bit per port.	0x0
23	routed_mask	Mask for routed.	0x1
24	routed	The packet was routed.	0x0
26:25	vrf_mask	Mask for vrf.	0x3

Bits	Field Name	Description	Default Value
28:27	vrf	The VRF used when routed.	0x0
29	flooded_mask	Mask for flooded.	0x1
30	flooded	The packet was flooded due to L2 table miss.	0x0
31	ucSwitched_mask	Mask for ucSwitched.	0x1
32	ucSwitched	The packet was L2 switched to a unicast destination port.	0x0
33	mcSwitched_mask	Mask for mcSwitched.	0x1
34	mcSwitched	The packet was L2 switched to a multicast group.	0x0
46:35	vid_mask	Mask for vid.	0xfff
58:47	vid	The index used in the VLAN table lookup.	0x0
60:59	l3Type_mask	Mask for l3Type.	0x3
62:61	l3Type	The packets L3 Type. abFourIPv4IPv6MPLSOther	0x0
65:63	l4Type_mask	Mask for l4Type.	0x7
68:66	l4Type	The packets L4 Type. abEightNot known.Is IPv4 or IPv6 but type is not any L4 type in this list.UDPTCPIGMPICMPICMPv6MLD	0x0
72:69	srcPort_mask	Mask for srcPort.	0xf
76:73	srcPort	The packets source port.	0x0
77	fromCrypto_mask	Mask for fromCrypto.	0x1
78	fromCrypto	The packet came from Crypto Engine.	0x0

38.11.31 Egress ACL Rule Pointer TCAM Answer

This is the table holding the answer for the **Egress ACL Rule Pointer TCAM**.

Number of Entries : 16
Type of Operation : Read/Write
Addressing : **Egress ACL Rule Pointer TCAM** hit index
Address Space : 1044784 to 1044799

Field Description

Bits	Field Name	Description	Default Value
2:0	rulePtr	Rule Pointer.	0x0

38.11.32 Egress Configurable ACL Large Table

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries : 8192
Number of Addresses per Entry : 8
Type of Operation : Read/Write
Addressing :

address[10:0] : hash of {compareData }
address[12:11] : bucket number

Address Space : 1044800 to 1110335

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
135:1	compareData	The data which shall be compared in this entry.	0x0
136	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
137	forceSendToCpuOrigPkt	This is a result field used when this entry is hit. If packet shall be sent to CPU then setting this bit will force the packet to be the incoming originial packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
138	metaDataValid	This is a result field used when this entry is hit. Is the meta_data field valid.	0x0
154:139	metaData	This is a result field used when this entry is hit. Meta data for packets going to the CPU.	0x0
155	dropEnable	This is a result field used when this entry is hit. If set, the packet shall be dropped and the **Egress Configurable ACL Drop** counter is incremented.	0x0
156	sendToPort	This is a result field used when this entry is hit. Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
160:157	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
161	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
167:162	counter	This is a result field used when this entry is hit. Which counter in **Egress Configurable ACL Match Counter** to update.	0x0
168	natOpValid	This is a result field used when this entry is hit. NAT operation pointer is valid.	0x0
181:169	natOpPtr	This is a result field used when this entry is hit. NAT operation pointer.	0x0
184:182	natVersion	This is a result field used when this entry is hit. NAT Entry Version	0x0
185	tunnelEntry	This is a result field used when this entry is hit. Shall all of these packets enter into a tunnel.	0x0
186	tunnelEntryUcMc	This is a result field used when this entry is hit. Shall this entry point to the **Tunnel Entry Instruction Table** with or without a egress port offset. 0 = Unicast **Tunnel Entry Instruction Table** without offset for each port 1 = Multicast **Tunnel Entry Instruction Table** with offset for each port.	0x0
191:187	tunnelEntryPtr	This is a result field used when this entry is hit. The tunnel entry which this packet shall enter upon exiting the switch.	0x0

Bits	Field Name	Description	Default Value
192	cancelCryptoOp	This is a result field used when this entry is hit. Cancel the crypto operation. No crypto operations will be done on this packet. 0 = No. 1 = Yes.	0x0
193	sendToCrypto	This is a result field used when this entry is hit. Do a crypto operation on this packet. 0 = No. 1 = Yes.	0x0
195:194	cryptoProto	This is a result field used when this entry is hit. Crypto protocol. 0 = AH 1 = ESP 2 = MACsec 3 = Reserved.	0x0
196	cryptoOp	This is a result field used when this entry is hit. Crypto operation. 0 = Encrypt 1 = Decrypt	0x0
202:197	secPtr	This is a result field used when this entry is hit. Pointer into Security Association Database. Valid if sendToCrypto is set.	0x0
206:203	cryptoPort	This is a result field used when this entry is hit. Crypto modification port. Before the packet is sent for encryption/decryption there can be packet modifications which are based on the egress port, this is the egress port which will be used for these packet modifications.	0x0
207	forceQueue	This is a result field used when this entry is hit. If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
210:208	eQueue	This is a result field used when this entry is hit. The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0

38.11.33 Egress Configurable ACL Rules Setup

The rules are setup by selecting which fields shall be used in the ACL search. Each rule has a fixed number of fields. The fieldSelectBitmask has one bit for each field. The first 7 fields (bits) which are set to one are selected to build the lookup key for this ACL. It is not allowed to set more than 7 bit in the bitmask. The fields are described in ACL Fields

Number of Entries : 8
Type of Operation : Read/Write
Addressing : ACL rule pointer
Address Space : 1119447 to 1119454

Field Description

Bits	Field Name	Description	Default Value
20:0	fieldSelectBitmask	Bitmask of which fields to select. Set a bit to one to select this specific field, set zero to not select field. At Maximum 7 bits should be set.	0x0

38.11.34 Egress Configurable ACL Search Mask

Before the hashing and searching is done in the **Egress Configurable ACL Large Table** and **Egress Configurable ACL Small Table**. The search data is AND:ed with this mask. If a bit in the mask is set to zero then this bit in the lookup will be viewed as do not care. Seperate masks exists for both small and large tables.

Number of Entries : 1
Number of Addresses per Entry : 16
Type of Operation : Read/Write
Address Space : 1124299

Field Description

Bits	Field Name	Description	Default Value
134:0	mask_small	Which bits to compare in the **Egress Configurable ACL Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	$2^{135} - 1$
269:135	mask_large	Which bits to compare in the **Egress Configurable ACL Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	$2^{135} - 1$

38.11.35 Egress Configurable ACL Selection

This register selects which result to use when there are multiple hits.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121243

Field Description

Bits	Field Name	Description	Default Value
0	selectTcamOrTable	If set to zero then TCAM answer is selected. If set to one then hash table answer is selected.	0x0
1	selectSmallOrLarge	If set to zero then small hash table is selected. If set to one then large hash table is selected.	0x0

38.11.36 Egress Configurable ACL Small Table

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries :	1024
Number of Addresses per Entry :	8
Type of Operation :	Read/Write
Addressing :	address[7:0] : hash of {compareData } address[9:8] : bucket number
Address Space :	1110336 to 1118527

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
135:1	compareData	The data which shall be compared in this entry.	0x0
136	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
137	forceSendToCpuOrigPkt	This is a result field used when this entry is hit. If packet shall be sent to CPU then setting this bit will force the packet to be the incoming originial packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
138	metaDataValid	This is a result field used when this entry is hit. Is the meta_data field valid.	0x0
154:139	metaData	This is a result field used when this entry is hit. Meta data for packets going to the CPU.	0x0
155	dropEnable	This is a result field used when this entry is hit. If set, the packet shall be dropped and the **Egress Configurable ACL Drop** counter is incremented.	0x0
156	sendToPort	This is a result field used when this entry is hit. Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
160:157	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
161	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
167:162	counter	This is a result field used when this entry is hit. Which counter in **Egress Configurable ACL Match Counter** to update.	0x0
168	natOpValid	This is a result field used when this entry is hit. NAT operation pointer is valid.	0x0
181:169	natOpPtr	This is a result field used when this entry is hit. NAT operation pointer.	0x0
184:182	natVersion	This is a result field used when this entry is hit. NAT Entry Version	0x0
185	tunnelEntry	This is a result field used when this entry is hit. Shall all of these packets enter into a tunnel.	0x0

Bits	Field Name	Description	Default Value
186	tunnelEntryUcMc	This is a result field used when this entry is hit. Shall this entry point to the **Tunnel Entry Instruction Table** with or without a egress port offset. 0 = Unicast **Tunnel Entry Instruction Table** without offset for each port 1 = Multicast **Tunnel Entry Instruction Table** with offset for each port.	0x0
191:187	tunnelEntryPtr	This is a result field used when this entry is hit. The tunnel entry which this packet shall enter upon exiting the switch.	0x0
192	cancelCryptoOp	This is a result field used when this entry is hit. Cancel the crypto operation. No crypto operations will be done on this packet. 0 = No. 1 = Yes.	0x0
193	sendToCrypto	This is a result field used when this entry is hit. Do a crypto operation on this packet. 0 = No. 1 = Yes.	0x0
195:194	cryptoProto	This is a result field used when this entry is hit. Crypto protocol. 0 = AH 1 = ESP 2 = MACsec 3 = Reserved.	0x0
196	cryptoOp	This is a result field used when this entry is hit. Crypto operation. 0 = Encrypt 1 = Decrypt	0x0
202:197	secPtr	This is a result field used when this entry is hit. Pointer into Security Association Database. Valid if sendToCrypto is set.	0x0
206:203	cryptoPort	This is a result field used when this entry is hit. Crypto modification port. Before the packet is sent for encryption/decryption there can be packet modifications which are based on the egress port, this is the egress port which will be used for these packet modifications.	0x0
207	forceQueue	This is a result field used when this entry is hit. If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
210:208	eQueue	This is a result field used when this entry is hit. The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0

38.11.37 Egress Configurable ACL TCAM

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.

Number of Entries :	16
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	1123899 to 1124154

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
135:1	mask	Which bits to compare in this entry.	$2^{135}-1$
270:136	compareData	The data which shall be compared in this entry. Observe that this compare data must be AND:ed by software before the entry is searched. The hardware does not do the AND between mask and compareData (In order to save area).	0x0

38.11.38 Egress Configurable ACL TCAM Answer

This is the table holding the answer for the **Egress Configurable ACL TCAM**.

Number of Entries :	16
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	**Egress Configurable ACL TCAM** hit index
Address Space :	1118528 to 1118591

Field Description

Bits	Field Name	Description	Default Value
0	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
1	forceSendToCpuOrigPkt	If packet shall be sent to CPU then setting this bit will force the packet to be the incoming originial packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
2	metaDataValid	Is the meta_data field valid.	0x0
18:3	metaData	Meta data for packets going to the CPU.	0x0
19	dropEnable	If set, the packet shall be dropped and the **Egress Configurable ACL Drop** counter is incremented.	0x0
20	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
24:21	destPort	The port which the packet shall be sent to.	0x0
25	updateCounter	When set the selected statistics counter will be updated.	0x0
31:26	counter	Which counter in **Egress Configurable ACL Match Counter** to update.	0x0
32	natOpValid	NAT operation pointer is valid.	0x0
45:33	natOpPtr	NAT operation pointer.	0x0
48:46	natVersion	NAT Entry Version	0x0
49	tunnelEntry	Shall all of these packets enter into a tunnel.	0x0

Bits	Field Name	Description	Default Value
50	tunnelEntryUcMc	Shall this entry point to the **Tunnel Entry Instruction Table** with or without a egress port offset. 0 = Unicast **Tunnel Entry Instruction Table** without offset for each port 1 = Multicast **Tunnel Entry Instruction Table** with offset for each port.	0x0
55:51	tunnelEntryPtr	The tunnel entry which this packet shall enter upon exiting the switch.	0x0
56	cancelCryptoOp	Cancel the crypto operation. No crypto operations will be done on this packet. 0 = No. 1 = Yes.	0x0
57	sendToCrypto	Do a crypto operation on this packet. 0 = No. 1 = Yes.	0x0
59:58	cryptoProto	Crypto protocol. 0 = AH 1 = ESP 2 = MACsec 3 = Reserved.	0x0
60	cryptoOp	Crypto operation. 0 = Encrypt 1 = Decrypt	0x0
66:61	secPtr	Pointer into Security Association Database. Valid if sendToCrypto is set.	0x0
70:67	cryptoPort	Crypto modification port. Before the packet is sent for encryption/decryption there can be packet modifications which are based on the egress port, this is the egress port which will be used for these packet modifications.	0x0
71	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
74:72	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0

38.11.39 Egress Port NAT State

At end of ingress processing a check is done to determine what to do with packets which has different port states and what the ingress and egress ACLs says what shall be done with the packets. The table needs to be enabled in the **natActionTableEnable**.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121244

Field Description

Bits	Field Name	Description	Default Value
10:0	portState	Egress Port NAT state (Bit 0 is port 0, bit 1 is port 1 etc.). 0 = Private 1 = Public	0x0

38.11.40 Egress Spanning Tree State

Spanning tree state for each egress port. The state Disabled implies that spanning tree protocol is not enabled and hence frames will be forwarded on this egress port.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	1122503

Field Description

Bits	Field Name	Description	Default Value
32:0	sptState	State of the spanning tree protocol. Bit[2:0] is port #0, bit[5:3] is port #1 etc. 0 = Disabled 1 = Blocking 2 = Listening 3 = Learning 4 = Forwarding	0x0

38.11.41 Enable Enqueue To Ports And Queues

This register is used to control if a particular port and queue shall be able to enqueue new packets. One queue mask exists for each port, setting a bit in the queue mask means packet is allowed to be queued on the respective queue. Packets that are directed to a queue that is turned off will be dropped and counted in **Queue Off Drop**.

Number of Entries :	11
Type of Operation :	Read/Write
Addressing :	Egress Port
Address Space :	1118635 to 1118645

Field Description

Bits	Field Name	Description	Default Value
7:0	q_on	If a bit is set, the corresponding queue is on.	0xff

38.11.42 Flooding Action Send to Port

If a packet is flooded and this function is enabled on the source port then the packet is send to a single egress port instead of being flooded to all ports part of the packets VLAN membership.

Number of Entries :	11
Type of Operation :	Read/Write
Addressing :	Source Port
Address Space :	1118646 to 1118656

Field Description

Bits	Field Name	Description	Default Value
0	enable	Enable sent to port instead of flooding. 0 = Disable 1 = Enable	0x0
4:1	destPort	Once enabled this is the destination port to sent the packet to in case of flooding.	0x0

38.11.43 Force Non VLAN Packet To Specific Color

If a packet is non-VLAN tagged, there is an option to force these packets to a certain initial color.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121235

Field Description

Bits	Field Name	Description	Default Value
0	forceColor	When set, packets which are non-VLAN tagged are forced to a color.	0x0
2:1	color	Initial color of the packet	0x0

38.11.44 Force Non VLAN Packet To Specific Queue

If a packet is non-VLAN tagged, there is an option to force these packets to a certain ingress/egress queue.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121233

Field Description

Bits	Field Name	Description	Default Value
0	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
3:1	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0

38.11.45 Force Unknown L3 Packet To Specific Color

If a packet does not contain IPv4, IPv6, MPLS or PPPoE carrying IPv4/IPv6 field there is an option to force the packet to a certain initial color.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121236

Field Description

Bits	Field Name	Description	Default Value
0	forceColor	When set, unknown L3 packet types are forced to a color.	0x0
2:1	color	Initial color of the packet	0x0

38.11.46 Force Unknown L3 Packet To Specific Egress Queue

If a packet does not contain IPv4, IPv6, MPLS or PPPoE carrying IPv4/IPv6 field there is an option to force the packet to a certain egress queue.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121234

Field Description

Bits	Field Name	Description	Default Value
0	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
3:1	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0

38.11.47 Forward From CPU

Indicates if all frames received on the CPU port shall be forwarded while ignoring the egress port's spanning tree status.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121239

Field Description

Bits	Field Name	Description	Default Value
0	enable	If set, any frame received on the CPU port is forwarded without consideration of the egress port's spanning tree state.	0x0

38.11.48 GRE Packet Decoder Options

The L4 protocol number which is used to detemine if the packet has a GRE header. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries : 1
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 1122491

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
8:1	l4Proto	The value to be used to find this packet type.	0x2f
24:9	udp1	The value to be used to find this packet type.	0x1292
40:25	udp2	The value to be used to find this packet type.	0x1293
51:41	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
62:52	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

38.11.49 Hairpin Enable

Decide if the L2 switching allows a packet to be switched back on the same port it entered the switch. There are separate controls for flooding due to unknown MAC DA, multicast and unicast.

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Ingress port
Address Space : 1119519 to 1119529

Field Description

Bits	Field Name	Description	Default Value
0	allowFlood	Allow flooding to source port.	0x0
1	allowMc	Allow multicast to source port.	0x0
2	allowUc	Allow unicast to source port.	0x1

38.11.50 Hardware Learning Configuration

Configure default status for a newly learned entry, learning limits and learning exceptions.

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Ingress Port
Address Space : 324 to 334

Field Description

Bits	Field Name	Description	Default Value
0	valid	For a new packet which is to be learned what value shall the valid bit have?	0x1

Bits	Field Name	Description	Default Value
1	stat	For a new packet which is to be learned what value shall the static bit have?	0x0
2	hit	For a new packet which is to be learned what value shall the hit bit have?	0x1
17:3	learnLimit	Maximum number of entries can be learned on this port. 0 means no limit.	0x0
18	portMoveException	When the hardware learning unit is turned on and the ingress packet processing determines to bypass the hardware learning check, set this field to one to still perform the port move action.	0x0
19	saHitException	When the hardware learning unit is turned on and the ingress packet processing determines to bypass the hardware learning check, set this field to one to still perform the SA hit update action.	0x0

38.11.51 Hardware Learning Counter

Number of MAC addresses learned by the hardware learning unit. Write 0 to clear.

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Ingress Port
Address Space : 369 to 379

Field Description

Bits	Field Name	Description	Default Value
14:0	cnt	Number of learned L2 entries.	0x0

38.11.52 Hash Based L3 Routing Table

This is the routing table used to determine the next hop. The IP lookup is done by hashing the VRF and the destination address extracted from the incoming packet. The hash is used to index this table. For each hash value the table has 4 buckets. The incoming IP address is compared with the destIPAddr field in all the buckets for the selected hash value. The packets assigned VRF is compared with the vrf fields and the protocol type is compares against the entries protocol. If there is a match in any bucket then the other fields in the matched bucket will be used for next hop processing. If ECMP is enabled for this entry an offset is added to the **nextHopPointer** and used when indexing the **Next Hop Table**.

Number of Entries : 65536
Number of Addresses per Entry : 8
Type of Operation : Read/Write
Addressing :

address[0:13] :	hash of {VRF, IP destination address} or {Source port and outermost MPLS label}
address[14:15] :	bucket number

Address Space : 347632 to 871919

Field Description

Bits	Field Name	Description	Default Value
0	ipVersion	Select if this is an IPv4 or IPv6 entry. 0 = IPv4 entry. 1 = IPv6 entry.	0x0
1	mpls	This is an MPLS entry, 0 = IP entry. 1 = MPLS entry.	0x0
3:2	vrf	This entries VRF. The packets assigned VRF will be compared with this field.	0x0
131:4	destIPAddr	The IP or MPLS address to be matched. If the entry is an IPv4 entry then only bits [31:0] is used. If the entry is a MPLS entry then bits [4-1:0] contains the source port while bits [4+19:4] contains the MPLS label to match.	0x0
142:132	nextHopPointer	Index into the **Next Hop Table** for this destination.	0x0
143	useECMP	Enables the use of ECMP hash to calculate the next hop pointer. 0 = Use ECMP hash. 1 = Do not use ECMP hash.	0x0
149:144	ecmpMask	How many bits of the ECMP hash will be used when calculating the ECMP offset. This byte is AND:ed with the ECMP hash to determine which bits shall be used as offset.	0x0
152:150	ecmpShift	How many bits the masked ECMP hash will be right shifted.	0x0
155:153	entryVersion	The version of this entry. All other tables which points from this table must have same version.	0x0

38.11.53 Hit Update Data FIFO

This register exposes the output of a FIFO which is holding all hit update requests to refresh the hit state. Under hardware hit writeback mode, the entry pushed to this FIFO is in sync with the FIB. If hardware hit writeback is turned off, the final hit update decision should be issued from software injected learning packet and what is pushed to this FIFO is not updated to L2 tables.

Number of Entries : 1
Type of Operation : Read Only
Address Space : 16771

Field Description

Bits	Field Name	Description	Default Value
0	hashRefreshHit	When this field is 1, the learning and aging engine has requested to refresh the hit state from 0 to 1 for the hash table in FIB.	0x0
11:1	hashValue	Hash of GID, MAC.	0x0
14:12	hashBucket	Bucket number of the hash lookup table.	0x0
15	camRefreshHit	When this field is 1, the learning and aging engine has requested to refresh the hit state from 0 to 1 for the cam entry.	0x0
20:16	camIndex	Index to the entry in **L2 Aging Collision Table**.	0x0

Bits	Field Name	Description	Default Value
21	valid	0 = Empty FIFO, entry is not valid 1 = Valid entry	0x0

38.11.54 Hit Update Data FIFO High Watermark Level

The High Watermark Interrupt will occur when a push to **Hit Update Data FIFO** is done and the number of existing entries after the push is larger than this setting.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 323

Field Description

Bits	Field Name	Description	Default Value
5:0	level	Number of used entries	0x0

38.11.55 IEEE 1588 L2 Packet Decoder Options

The Ethernet type used to determine if a packet is a IEEE 1588 L2 Packet. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries : 1
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 1122487

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	eth	The value to be used to find this packet type.	0x88f7
27:17	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
38:28	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0
39	ptp	If a packet is sent to the CPU and this bit is set and the packet has a timestamp then it will show having a valid timestamp in the CPU-header.	0x0

38.11.56 IEEE 1588 L4 Packet Decoder Options

IEEE 1588 L4 packet is determined by this register. Fields from L2/L3/L4 are required for the comparison, including two optional DA MAC, five optional IPv4 DA, two optional IPv6 DA with the first one maskable, and two optional UDP destination ports. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries :	1
Number of Addresses per Entry :	32
Type of Operation :	Read/Write
Address Space :	1123707

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
48:1	da_mac1	DA MAC to match.	0x11b19000000
96:49	da_mac2	DA MAC to match.	0x180c200000e
128:97	da_ipv4_addr1	IPv4 DA to match.	0xe0000181
160:129	da_ipv4_addr2	IPv4 DA to match.	0xe0000182
192:161	da_ipv4_addr3	IPv4 DA to match.	0xe0000183
224:193	da_ipv4_addr4	IPv4 DA to match.	0xe0000184
256:225	da_ipv4_addr5	IPv4 DA to match.	0xe000016b
384:257	da_ipv6_addr1	IPv6 DA to match. This address is maskable.	0x181000000000000000000000000000ff0
512:385	da_ipv6_mask1	Bit mask for da_ipv6_addr1. For each bit of the mask, 1 means valid for comparison.	0xfff0ffffffffffffffffffffffffffff
640:513	da_ipv6_addr2	IPv6 DA to match.	0x6b000000000000000000000000ff02
656:641	udp1	UDP destination to match.	0x13f
672:657	udp2	UDP destination to match.	0x140
683:673	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
694:684	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0
695	ptp	If a packet is sent to the CPU and this bit is set and the packet has a timestamp then it will show having a valid timestamp in the CPU-header.	0x0

38.11.57 IEEE 802.1X and EAPOL Packet Decoder Options

The Ethernet type used to determine if a packet is a 802.1X or EAPOL packet. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	1122489

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	eth	The value to be used to find this packet type.	0x888e
27:17	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
38:28	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

38.11.58 IKE Packet Decoder Options

The UDP ports used to detect a IKE packet the underlaying packet must be a IPv4 or IPv6 packet.. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	1122499

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	udp1	The value to be used to find this packet type.	0x1f4
32:17	udp2	The value to be used to find this packet type.	0x1194
43:33	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
54:44	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

38.11.59 IPP Debug debugMatchIPP0

Packet processing pipeline status for debugMatchIPP0.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	1121270

Field Description

Bits	Field Name	Description	Default Value
21:0	value	Status from last processed packet.	0x0

38.11.60 IPP Debug doL2Lookup

Packet processing pipeline status for doL2Lookup.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121268

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.11.61 IPP Debug dropPktAfterL2Decode

Packet processing pipeline status for dropPktAfterL2Decode.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121250

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.11.62 IPP Debug dropPktAfterL3Decode

Packet processing pipeline status for dropPktAfterL3Decode.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121252

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.11.63 IPP Debug dstPortmask

Packet processing pipeline status for dstPortmask.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121269

Field Description

Bits	Field Name	Description	Default Value
10:0	value	Status from last processed packet.	0x0

38.11.64 IPP Debug finalVid

Packet processing pipeline status for finalVid.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121254

Field Description

Bits	Field Name	Description	Default Value
12:0	value	Status from last processed packet.	0x0

38.11.65 IPP Debug isBroadcast

Packet processing pipeline status for isBroadcast.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121267

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.11.66 IPP Debug isFlooding

Packet processing pipeline status for isFlooding.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121266

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.11.67 IPP Debug I2DaHash

Packet processing pipeline status for I2DaHash.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121262

Field Description

Bits	Field Name	Description	Default Value
10:0	value	Status from last processed packet.	0x0

38.11.68 IPP Debug I2DaHashHitAndBucket

Packet processing pipeline status for I2DaHashHitAndBucket.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121263

Field Description

Bits	Field Name	Description	Default Value
3:0	value	Status from last processed packet.	0x0

38.11.69 IPP Debug I2DaHashKey

Packet processing pipeline status for I2DaHashKey.

Number of Entries : 1
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 1122509

Field Description

Bits	Field Name	Description	Default Value
59:0	value	Status from last processed packet.	0x0

38.11.70 IPP Debug l2DaTcamHitsAndCast

Packet processing pipeline status for l2DaTcamHitsAndCast.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121264

Field Description

Bits	Field Name	Description	Default Value
16:0	value	Status from last processed packet.	0x0

38.11.71 IPP Debug nextHopPtrFinal

Packet processing pipeline status for nextHopPtrFinal.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121261

Field Description

Bits	Field Name	Description	Default Value
10:0	value	Status from last processed packet.	0x0

38.11.72 IPP Debug nextHopPtrHash

Packet processing pipeline status for nextHopPtrHash.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121258

Field Description

Bits	Field Name	Description	Default Value
10:0	value	Status from last processed packet.	0x0

38.11.73 IPP Debug nextHopPtrHashHit

Packet processing pipeline status for nextHopPtrHashHit.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121260

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.11.74 IPP Debug nextHopPtrLpm

Packet processing pipeline status for nextHopPtrLpm.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121257

Field Description

Bits	Field Name	Description	Default Value
10:0	value	Status from last processed packet.	0x0

38.11.75 IPP Debug nextHopPtrLpmHit

Packet processing pipeline status for nextHopPtrLpmHit.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121259

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.11.76 IPP Debug nrVlans

Packet processing pipeline status for nrVlans.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121251

Field Description

Bits	Field Name	Description	Default Value
1:0	value	Status from last processed packet.	0x0

38.11.77 IPP Debug routed

Packet processing pipeline status for routed.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121265

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.11.78 IPP Debug routerHit

Packet processing pipeline status for routerHit.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121256

Field Description

Bits	Field Name	Description	Default Value
0	value	Status from last processed packet.	0x0

38.11.79 IPP Debug spVidOp

Packet processing pipeline status for spVidOp.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121253

Field Description

Bits	Field Name	Description	Default Value
2:0	value	Status from last processed packet.	0x0

38.11.80 IPP Debug srcPort

Packet processing pipeline status for srcPort.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121249

Field Description

Bits	Field Name	Description	Default Value
3:0	value	Status from last processed packet.	0x0

38.11.81 IPP Debug vlanVidOp

Packet processing pipeline status for vlanVidOp.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121255

Field Description

Bits	Field Name	Description	Default Value
2:0	value	Status from last processed packet.	0x0

38.11.82 IPSec Table

Which IPSec protocol the router shall use.

Number of Entries : 64
Type of Operation : Read/Write
Addressing : SA Pointer from Next Hop Table
Address Space : 962096 to 962159

Field Description

Bits	Field Name	Description	Default Value
1:0	protocol	AH or ESP operation. 0 = AH 1 = ESP	0x0

38.11.83 IPv4 TOS Field To Egress Queue Mapping Table

Mapping table from TOS in the IPv4 header to an egress queue.

Number of Entries : 256
Type of Operation : Read/Write
Addressing : Incoming IPv4 packets TOS
Address Space : 1120370 to 1120625

Field Description

Bits	Field Name	Description	Default Value
2:0	pQueue	Egress queue.	0x1

38.11.84 IPv4 TOS Field To Packet Color Mapping Table

Mapping table from TOS in the IPv4 header to a packet inital color.

Number of Entries : 256
Type of Operation : Read/Write
Addressing : Incoming IPv4 packets TOS pointer
Address Space : 1119834 to 1120089

Field Description

Bits	Field Name	Description	Default Value
1:0	color	Packet initial color.	0x0

38.11.85 IPv6 Class of Service Field To Egress Queue Mapping Table

Mapping table from Class of Service in the IPv6 header to an egress queue.

Number of Entries : 256
Type of Operation : Read/Write
Addressing : Incoming IPv6 packets Class of Service
Address Space : 1120114 to 1120369

Field Description

Bits	Field Name	Description	Default Value
2:0	pQueue	Egress queue.	0x1

38.11.86 IPv6 Class of Service Field To Packet Color Mapping Table

Mapping table from Class of service in the IPv6 header to a packet inital color.

Number of Entries : 256
Type of Operation : Read/Write
Addressing : Incoming IPv6 packets Class os Service pointer
Address Space : 1119578 to 1119833

Field Description

Bits	Field Name	Description	Default Value
1:0	color	Packet initial color.	0x0

38.11.87 Ingress Admission Control Current Status

Number of tokens currently in the token bucket.

Number of Entries : 64
Type of Operation : Read/Write
Addressing : Meter Pointer
Address Space : 1128942 to 1129005

Field Description

Bits	Field Name	Description	Default Value
15:0	tokens_0	Number of tokens after the last visit for token bucket 0.	0x0
31:16	tokens_1	Number of tokens after the last visit for token bucket 1.	0x0

38.11.88 Ingress Admission Control Initial Pointer

Initial ingress admission control pointer based on source port number and L2 priority. L2 priority is from either the outermost VLAN PCP field or **defaultPcp**. Further processes may overwrite the initial pointer by comparing the order of the pointer.

Number of Entries : 128
Type of Operation : Read/Write
Addressing :

address[3:0] :	Ingress Port
address[6:4] :	L2 Priority

Address Space : 17172 to 17299

Field Description

Bits	Field Name	Description	Default Value
0	mmpValid	If set, this entry contains a valid MMP pointer	0x0
6:1	mmpPtr	Initial pointer to the ingress MMP.	0x0
8:7	mmpOrder	Order of the initial ingress MMP pointer.	0x0

38.11.89 Ingress Admission Control Mark All Red

Blocking status of the MMP entry due to packet drops in the MMP.

Number of Entries : 64
Type of Operation : Read/Write
Addressing : Meter Pointer
Address Space : 1128558 to 1128621

Field Description

Bits	Field Name	Description	Default Value
0	markAllRed	When this field is set to 1 by the core, the corresponding MMP entry is under the blocking status. As a consequence, all packets with this MMP pointer will be dropped. Clear this field to allow packets enter the MMP entry again.	0x0

38.11.90 Ingress Admission Control Mark All Red Enable

Option to block metering after MMP packet drops.

Number of Entries : 64
Type of Operation : Read/Write
Addressing : Meter Pointer
Address Space : 1128494 to 1128557

Field Description

Bits	Field Name	Description	Default Value
0	markAllRedEn	After setting this field to 1, if a packet is dropped by a MMP entry, this MMP entry will stop metering and drop all packets with the corresponding MMP pointer.	0x0

38.11.91 Ingress Admission Control Reset

Reset token buckets so that it is back to the inital status. The reset will be kept high till new traffic arrives, then the traffic is metered with a bucket full of tokens and the reset is deactivated. It is helpful when the token bucket configuration is changed during runtime.

Number of Entries : 64
Type of Operation : Read/Write
Addressing : Meter Pointer
Address Space : 1128878 to 1128941

Field Description

Bits	Field Name	Description	Default Value
0	bucketReset	if set, reload with full tokens for token buckets in this entry.	0x1

38.11.92 Ingress Admission Control Token Bucket Configuration

Configuration options for token buckets used by Ingress Admission Control. Each entry refers to either a single rate three color marker (srTCM) or a two rate three color marker (trTCM) with two token buckets. For each token bucket the rate is configured by filling in a certain number of tokens at one of the available

frequencies. Token bucket 0 shall always use the committed information rate (CIR). Runtime configuration update requires writting 1 to the **Ingress Admission Control Reset** first.

Number of Entries :	64
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	Meter Pointer
Address Space :	1128622 to 1128877

Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity_0	Capacity for token bucket 0.	0x0
27:16	tokens_0	Number of tokens added each tick for token bucket 0.	0x0
30:28	tick_0	Select one of the 5 available ticks for token bucket 0. The tick frequencies are configured globaly in the Core Tick Configuration register.	0x0
46:31	bucketCapacity_1	Capacity for token bucket 1.	0x0
58:47	tokens_1	Number of tokens added each tick for token bucket 1.	0x0
61:59	tick_1	Select one of the 5 available ticks for token bucket 1. The tick frequencies are configured globaly in the Core Tick Configuration register.	0x0
62	bucketMode	0 = srTCM 1 = trTCM	0x0
63	colorBlind	0 = color-aware: The metering result is based on the initial coloring from the ingress process pipeline. 1 = color-blind: The metering ignores any pre-coloring.	0x0
66:64	dropMask	Drop mask for the three colors obtained from the metering result. For each bit set to 1 the corresponding color shall drop the packet. Bit 0, 1, 2 represents drop or not for green, yellow and red respectively	0x4
81:67	maxLength	Maximum allowed packet length in bytes. Packets with bytes larger than this value will be dropped before metering.	0x7fff
83:82	tokenMode	0 = Count in bytes and add extra bytes for metering. 1 = Count in bytes and substract extra bytes for metering. 2 = Count in packets. 3 = No tokens are counted.	0x0
91:84	byteCorrection	Extra bytes per packet for IFG correction, only valid under byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

38.11.93 Ingress Configurable ACL 0 Large Table

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.. If multiple buckets match then the result

from the highest entry is selected.

Number of Entries :	8192
Number of Addresses per Entry :	32
Type of Operation :	Read/Write
Addressing :	address[10:0] : hash of {compareData } address[12:11] : bucket number
Address Space :	17316 to 279459

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
430:1	compareData	The data which shall be compared in this entry.	0x0
431	forceRoute	This is a result field used when this entry is hit. Shall the packet do a forced Routing? 0 = No. 1 = Yes.	0x0
442:432	nextHopPtr	This is a result field used when this entry is hit. Which next hop entry shall the forced routing used?	0x0
444:443	vrf	This is a result field used when this entry is hit. Which vrf shall the forced routing used?	0x0
447:445	nextHopVersion	This is a result field used when this entry is hit. Which version does this force route table entry have?	0x0
448	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
449	forceSendToCpuOrigPkt	This is a result field used when this entry is hit. If packet shall be sent to CPU then setting this bit will force the packet to be the incoming originial packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
450	metaDataValid	This is a result field used when this entry is hit. Is the meta_data field valid.	0x0
466:451	metaData	This is a result field used when this entry is hit. Meta data for packets going to the CPU.	0x0
467	metaDataPrio	This is a result field used when this entry is hit. If multiple ACLs hit this meta_data shall take priority.	0x0
468	dropEnable	This is a result field used when this entry is hit. If set, the packet shall be dropped and the **Ingress Configurable ACL Drop** counter is incremented.	0x0
469	sendToPort	This is a result field used when this entry is hit. Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
473:470	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0

Bits	Field Name	Description	Default Value
474	inputMirror	This is a result field used when this entry is hit. If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
478:475	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
479	imPrio	This is a result field used when this entry is hit. If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
480	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
486:481	counter	This is a result field used when this entry is hit. Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
487	updateTosExp	This is a result field used when this entry is hit. Force TOS/EXP update.	0x0
495:488	newTosExp	This is a result field used when this entry is hit. New TOS/EXP value.	0x0
503:496	tosMask	This is a result field used when this entry is hit. Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
504	enableUpdateIp	This is a result field used when this entry is hit. If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this is exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0
505	updateSaOrDa	This is a result field used when this entry is hit. Update the SA or DA IPv4 address. The Destiantion IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0
537:506	newIpValue	This is a result field used when this entry is hit. Update the SA or DA IPv4 address value.	0x0
538	enableUpdateL4	This is a result field used when this entry is hit. If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0

Bits	Field Name	Description	Default Value
539	updateL4SpOrDp	This is a result field used when this entry is hit. Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
555:540	newL4Value	This is a result field used when this entry is hit. Update the L4 SP or DP with this value	0x0
556	natOpValid	This is a result field used when this entry is hit. NAT operation pointer is valid.	0x0
569:557	natOpPtr	This is a result field used when this entry is hit. NAT operation pointer.	0x0
570	natOpPrio	This is a result field used when this entry is hit. If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
573:571	natVersion	This is a result field used when this entry is hit. NAT Entry Version.	0x0
574	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
576:575	color	This is a result field used when this entry is hit. Initial color of the packet if the forceColor field is set.	0x0
577	forceColorPrio	This is a result field used when this entry is hit. If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
578	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
584:579	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0
586:585	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
587	forceQueue	This is a result field used when this entry is hit. If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
590:588	eQueue	This is a result field used when this entry is hit. The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
591	forceQueuePrio	This is a result field used when this entry is hit. If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0

38.11.94 Ingress Configurable ACL 0 Pre Lookup

The pre ACL lookup allows the user to defined a specific rules for certain packet types in the ACL engine 0. Setting the valid bit and a new rule will override the default rule pointer from the source port table.

Number of Entries : 16

Type of Operation : Read/Write

Addressing :

Address bits [1:0]	Value from **preLookupAclBits**.
Address bits [3:2]	L3 Type Of Packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4, IPv6 or MPLS

Address Space : 17300 to 17315

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. If not then use default port rule.	0x0
3:1	rulePtr	If the valid is entry then this rule pointer will be used.	0x0

38.11.95 Ingress Configurable ACL 0 Rules Setup

The rules are setup by selecting which fields shall be used in the ACL search. Each rule has a fixed number of fields. The fieldSelectBitmask has one bit for each field. The first 7 fields (bits) which are set to one are selected. It is not allowed to set more than 7 bit in the bitmask. The fields are described in ACL Fields

Number of Entries : 8
Type of Operation : Read/Write
Addressing : ACL rule pointer
Address Space : 1120654 to 1120661

Field Description

Bits	Field Name	Description	Default Value
15:0	fieldSelectBitmask	Bitmask of which fields to select. Set a bit to one to select this specific field, set zero to not select field. At Maximum 7 bits should be set.	0x0

38.11.96 Ingress Configurable ACL 0 Search Mask

Before the hashing and searching is done in the **Ingress Configurable ACL 0 Large Table** and **Ingress Configurable ACL 0 Small Table**. The search data is AND:ed with this mask. If a bit in the mask is set to zero then this bit in the lookup will be viewed as do not care. Seperate masks exists for both small and large tables.

Number of Entries : 1
Number of Addresses per Entry : 32
Type of Operation : Read/Write
Address Space : 1123739

Field Description

Bits	Field Name	Description	Default Value
429:0	mask_small	Which bits to compare in the **Ingress Configurable ACL 0 Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	$2^{430} - 1$

Bits	Field Name	Description	Default Value
859:430	mask_large	Which bits to compare in the **Ingress Configurable ACL 0 Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	$2^{430} - 1$

38.11.97 Ingress Configurable ACL 0 Selection

This register selects which result to use when there are multiple hits.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121224

Field Description

Bits	Field Name	Description	Default Value
0	selectTcamOrTable	If set to zero then TCAM answer is selected. If set to one then hash table answer is selected.	0x0
1	selectSmallOrLarge	If set to zero then small hash table is selected. If set to one then large hash table is selected.	0x0

38.11.98 Ingress Configurable ACL 0 Small Table

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries : 1024
Number of Addresses per Entry : 32
Type of Operation : Read/Write
Addressing :

address[7:0] :	hash of {compareData }
address[9:8] :	bucket number

Address Space : 279460 to 312227

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
430:1	compareData	The data which shall be compared in this entry.	0x0
431	forceRoute	This is a result field used when this entry is hit. Shall the packet do a forced Routing? 0 = No. 1 = Yes.	0x0

Bits	Field Name	Description	Default Value
442:432	nextHopPtr	This is a result field used when this entry is hit. Which next hop entry shall the forced routing used?	0x0
444:443	vrf	This is a result field used when this entry is hit. Which vrf shall the forced routing used?	0x0
447:445	nextHopVersion	This is a result field used when this entry is hit. Which version does this force route table entry have?	0x0
448	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
449	forceSendToCpuOrigPkt	This is a result field used when this entry is hit. If packet shall be sent to CPU then setting this bit will force the packet to be the incoming originial packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
450	metaDataValid	This is a result field used when this entry is hit. Is the meta_data field valid.	0x0
466:451	metaData	This is a result field used when this entry is hit. Meta data for packets going to the CPU.	0x0
467	metaDataPrio	This is a result field used when this entry is hit. If multiple ACLs hit this meta_data shall take priority.	0x0
468	dropEnable	This is a result field used when this entry is hit. If set, the packet shall be dropped and the **Ingress Configurable ACL Drop** counter is incremented.	0x0
469	sendToPort	This is a result field used when this entry is hit. Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
473:470	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
474	inputMirror	This is a result field used when this entry is hit. If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
478:475	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
479	imPrio	This is a result field used when this entry is hit. If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
480	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
486:481	counter	This is a result field used when this entry is hit. Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
487	updateTosExp	This is a result field used when this entry is hit. Force TOS/EXP update.	0x0
495:488	newTosExp	This is a result field used when this entry is hit. New TOS/EXP value.	0x0

Bits	Field Name	Description	Default Value
503:496	tosMask	This is a result field used when this entry is hit. Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
504	enableUpdateIp	This is a result field used when this entry is hit. If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this is exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0
505	updateSaOrDa	This is a result field used when this entry is hit. Update the SA or DA IPv4 address. The Destiantion IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0
537:506	newIpValue	This is a result field used when this entry is hit. Update the SA or DA IPv4 address value.	0x0
538	enableUpdateL4	This is a result field used when this entry is hit. If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0
539	updateL4SpOrDp	This is a result field used when this entry is hit. Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
555:540	newL4Value	This is a result field used when this entry is hit. Update the L4 SP or DP with this value	0x0
556	natOpValid	This is a result field used when this entry is hit. NAT operation pointer is valid.	0x0
569:557	natOpPtr	This is a result field used when this entry is hit. NAT operation pointer.	0x0
570	natOpPrio	This is a result field used when this entry is hit. If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
573:571	natVersion	This is a result field used when this entry is hit. NAT Entry Version.	0x0
574	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
576:575	color	This is a result field used when this entry is hit. Initial color of the packet if the forceColor field is set.	0x0
577	forceColorPrio	This is a result field used when this entry is hit. If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
578	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0

Bits	Field Name	Description	Default Value
584:579	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0
586:585	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
587	forceQueue	This is a result field used when this entry is hit. If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
590:588	eQueue	This is a result field used when this entry is hit. The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
591	forceQueuePrio	This is a result field used when this entry is hit. If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0

38.11.99 Ingress Configurable ACL 0 TCAM

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.

Number of Entries : 16
Number of Addresses per Entry : 32
Type of Operation : Read/Write
Addressing : All entries are read out in parallel
Address Space : 1123195 to 1123706

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
430:1	mask	Which bits to compare in this entry.	$2^{430}-1$
860:431	compareData	The data which shall be compared in this entry. Observe that this compare data must be AND:ed by software before the entry is searched. The hardware does not do the AND between mask and compareData (In order to save area).	0x0

38.11.100 Ingress Configurable ACL 0 TCAM Answer

This is the table holding the answer for the **Ingress Configurable ACL 0 TCAM**.

Number of Entries : 16
Number of Addresses per Entry : 8
Type of Operation : Read/Write
Addressing : **Ingress Configurable ACL 0 TCAM** hit index
Address Space : 312228 to 312355

Field Description

Bits	Field Name	Description	Default Value
0	forceRoute	Shall the packet do a forced Routing? 0 = No. 1 = Yes.	0x0
11:1	nextHopPtr	Which next hop entry shall the forced routing used?	0x0
13:12	vrf	Which vrf shall the forced routing used?	0x0
16:14	nextHopVersion	Which version does this force route table entry have?	0x0
17	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
18	forceSendToCpuOrigPkt	If packet shall be sent to CPU then setting this bit will force the packet to be the incoming originial packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
19	metaDataValid	Is the meta_data field valid.	0x0
35:20	metaData	Meta data for packets going to the CPU.	0x0
36	metaDataPrio	If multiple ACLs hit this meta_data shall take priority.	0x0
37	dropEnable	If set, the packet shall be dropped and the **Ingress Configurable ACL Drop** counter is incremented.	0x0
38	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
42:39	destPort	The port which the packet shall be sent to.	0x0
43	inputMirror	If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
47:44	destInputMirror	Destination physical port for input mirroring.	0x0
48	imPrio	If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
49	updateCounter	When set the selected statistics counter will be updated.	0x0
55:50	counter	Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
56	updateTosExp	Force TOS/EXP update.	0x0
64:57	newTosExp	New TOS/EXP value.	0x0
72:65	tosMask	Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
73	enableUpdateIp	If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this is exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0

Bits	Field Name	Description	Default Value
74	updateSaOrDa	Update the SA or DA IPv4 address. The Destiantion IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0
106:75	newIpValue	Update the SA or DA IPv4 address value.	0x0
107	enableUpdateL4	If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0
108	updateL4SpOrDp	Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
124:109	newL4Value	Update the L4 SP or DP with this value	0x0
125	natOpValid	NAT operation pointer is valid.	0x0
138:126	natOpPtr	NAT operation pointer.	0x0
139	natOpPrio	If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
142:140	natVersion	NAT Entry Version.	0x0
143	forceColor	If set, the packet shall have a forced color.	0x0
145:144	color	Initial color of the packet if the forceColor field is set.	0x0
146	forceColorPrio	If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
147	mmpValid	If set, this entry contains a valid MMP pointer	0x0
153:148	mmpPtr	Ingress MMP pointer.	0x0
155:154	mmpOrder	Ingress MMP pointer order.	0x0
156	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
159:157	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
160	forceQueuePrio	If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0

38.11.101 Ingress Configurable ACL 1 Large Table

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries : 256
Number of Addresses per Entry : 16
Type of Operation : Read/Write
Addressing :

address[6:0]	:	hash of {compareData }
address[7:7]	:	bucket number

Address Space : 313380 to 317475

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
235:1	compareData	The data which shall be compared in this entry.	0x0
236	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
237	forceSendToCpuOrigPkt	This is a result field used when this entry is hit. If packet shall be sent to CPU then setting this bit will force the packet to be the incoming originial packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
238	metaDataValid	This is a result field used when this entry is hit. Is the meta_data field valid.	0x0
254:239	metaData	This is a result field used when this entry is hit. Meta data for packets going to the CPU.	0x0
255	metaDataPrio	This is a result field used when this entry is hit. If multiple ACLs hit this meta_data shall take priority.	0x0
256	dropEnable	This is a result field used when this entry is hit. If set, the packet shall be dropped and the **Ingress Configurable ACL Drop** counter is incremented.	0x0
257	sendToPort	This is a result field used when this entry is hit. Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
261:258	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
262	inputMirror	This is a result field used when this entry is hit. If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
266:263	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
267	imPrio	This is a result field used when this entry is hit. If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
268	noLearning	This is a result field used when this entry is hit. If set this packets MAC SA will not be learned.	0x0
269	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
275:270	counter	This is a result field used when this entry is hit. Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
276	updateTosExp	This is a result field used when this entry is hit. Force TOS/EXP update.	0x0
284:277	newTosExp	This is a result field used when this entry is hit. New TOS/EXP value.	0x0

Bits	Field Name	Description	Default Value
292:285	tosMask	This is a result field used when this entry is hit. Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
293	updateCfiDei	This is a result field used when this entry is hit. The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
294	newCfiDeiValue	This is a result field used when this entry is hit. The value to update to.	0x0
295	updatePcp	This is a result field used when this entry is hit. The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
298:296	newPcpValue	This is a result field used when this entry is hit. The PCP value to update to.	0x0
299	updateVid	This is a result field used when this entry is hit. The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
311:300	newVidValue	This is a result field used when this entry is hit. The VID value to update to.	0x0
312	updateEType	This is a result field used when this entry is hit. The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
314:313	newEthType	This is a result field used when this entry is hit. Select which TPID to use in the outer VLAN header. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register **Egress Ethernet Type for VLAN tag**.	0x0
315	cfiDeiPrio	This is a result field used when this entry is hit. If multiple updateCfiDei are set and this prio bit is set then this updateCfiDei will be selected.	0x0
316	pcpPrio	This is a result field used when this entry is hit. If multiple updatePcp are set and this prio bit is set then this updatePcp will be selected.	0x0
317	vidPrio	This is a result field used when this entry is hit. If multiple updateVid are set and this prio bit is set then this updateVid will be selected.	0x0
318	ethPrio	This is a result field used when this entry is hit. If multiple updateEType are set and this prio bit is set then this updateEType will be selected.	0x0

Bits	Field Name	Description	Default Value
319	enableUpdateIp	This is a result field used when this entry is hit. If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this is exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0
320	updateSaOrDa	This is a result field used when this entry is hit. Update the SA or DA IPv4 address. The Destiantion IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0
352:321	newIpValue	This is a result field used when this entry is hit. Update the SA or DA IPv4 address value.	0x0
353	enableUpdateL4	This is a result field used when this entry is hit. If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0
354	updateL4SpOrDp	This is a result field used when this entry is hit. Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
370:355	newL4Value	This is a result field used when this entry is hit. Update the L4 SP or DP with this value	0x0
371	natOpValid	This is a result field used when this entry is hit. NAT operation pointer is valid.	0x0
384:372	natOpPtr	This is a result field used when this entry is hit. NAT operation pointer.	0x0
385	natOpPrio	This is a result field used when this entry is hit. If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
388:386	natVersion	This is a result field used when this entry is hit. NAT Entry Version.	0x0
389	ptp	This is a result field used when this entry is hit. When the packet is sent to the CPU the packet will have the PTP bit in the To CPU Tag set to one. The timestamp in the To CPU Tag will also be set to the timestamp from the incoming packet.	0x0
390	tunnelEntry	This is a result field used when this entry is hit. Shall all of these packets enter into a tunnel.	0x0
391	tunnelEntryUcMc	This is a result field used when this entry is hit. Shall this entry point to the **Tunnel Entry Instruction Table** with or without a egress port offset. 0 = Unicast **Tunnel Entry Instruction Table** without offset for each port 1 = Multicast **Tunnel Entry Instruction Table** with offset for each port.	0x0

Bits	Field Name	Description	Default Value
396:392	tunnelEntryPtr	This is a result field used when this entry is hit. The tunnel entry which this packet shall enter upon exiting the switch.	0x0
397	tunnelEntryPrio	This is a result field used when this entry is hit. If multiple tunnelEntry are set and this prio bit is set then this tunnelEntryPtr will be selected.	0x0
398	tunnelExit	This is a result field used when this entry is hit. Shall this packet do a tunnel exit. 0 = No 1 = Yes	0x0
403:399	tunnelExitPtr	This is a result field used when this entry is hit. Pointer to tunnel exit described in **Egress Tunnel Exit Table**.	0x0
404	tunnelExitPrio	This is a result field used when this entry is hit. If multiple tunnelExit are set and this prio bit is set then this tunnelExitPtr will be selected.	0x0
405	cancelCryptoOp	This is a result field used when this entry is hit. Cancel the crypto operation. No crypto operations will be done on this packet. 0 = No. 1 = Yes.	0x0
406	sendToCrypto	This is a result field used when this entry is hit. Do a crypto operation on this packet. 0 = No. 1 = Yes.	0x0
408:407	cryptoProto	This is a result field used when this entry is hit. Crypto protocol. 0 = AH 1 = ESP 2 = MACsec 3 = Reserved.	0x0
409	cryptoOp	This is a result field used when this entry is hit. Crypto operation. 0 = Encrypt 1 = Decrypt	0x0
415:410	secPtr	This is a result field used when this entry is hit. Pointer into Security Association Database. Valid if sendToCrypto is set.	0x0
419:416	cryptoPort	This is a result field used when this entry is hit. Crypto modification port. Before the packet is sent for encryption/decryption there can be packet modifications which are based on the egress port, this is the egress port which will be used for these packet modifications.	0x0
420	cryptoPrio	This is a result field used when this entry is hit. If multiple sendToCrypto actions are set and this prio bit is set then those secPtr,cryptoProto and cryptoOp value will be selected.	0x0
421	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
423:422	color	This is a result field used when this entry is hit. Initial color of the packet if the forceColor field is set.	0x0

Bits	Field Name	Description	Default Value
424	forceColorPrio	This is a result field used when this entry is hit. If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
425	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
431:426	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0
433:432	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
434	forceQueue	This is a result field used when this entry is hit. If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
437:435	eQueue	This is a result field used when this entry is hit. The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
438	forceQueuePrio	This is a result field used when this entry is hit. If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0
439	forceVidValid	This is a result field used when this entry is hit. Override the Ingress VID, see chapter VLAN Processing.	0x0
451:440	forceVid	This is a result field used when this entry is hit. The new Ingress VID.	0x0
452	forceVidPrio	This is a result field used when this entry is hit. If multiple forceVid are set and this prio bit is set then this forceVid value will be selected.	0x0

38.11.102 Ingress Configurable ACL 1 Pre Lookup

The pre ACL lookup allows the user to defined a specific rules for certain packet types in the ACL engine 1. Setting the valid bit and a new rule will override the default rule pointer from the source port table.

Number of Entries : 1024

Type of Operation : Read/Write

Addressing :

Address bits [1:0]	Value from **preLookupAclBits**.
Address bits [3:2]	Number of VLANs in incoming Packet.
Address bits [4:4]	L2 Type Of Packet. 0 = Others - Not listed in this list. 1 = MACsec
Address bits [6:5]	L3 Type Of Packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4, IPv6 or MPLS
Address bits [9:7]	L4 Type Of Packet. 0 = Not known. 1 = Is IPv4 or IPv6 but type is not any L4 type in this list. 2 = UDP 3 = TCP 4 = IGMP 5 = ICMP 6 = ICMPv6 7 = MLD

Address Space : 312356 to 313379

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. If not then use default port rule.	0x0
3:1	rulePtr	If the valid is entry then this rule pointer will be used.	0x0

38.11.103 Ingress Configurable ACL 1 Rules Setup

The rules are setup by selecting which fields shall be used in the ACL search. Each rule has a fixed number of fields. The fieldSelectBitmask has one bit for each field. The first 7 fields (bits) which are set to one are selected. It is not allowed to set more than 7 bit in the bitmask. The fields are described in ACL Fields

Number of Entries :	8
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	ACL rule pointer
Address Space :	1122435 to 1122450

Field Description

Bits	Field Name	Description	Default Value
38:0	fieldSelectBitmask	Bitmask of which fields to select. Set a bit to one to select this specific field, set zero to not select field. At Maximum 7 bits should be set.	0x0

38.11.104 Ingress Configurable ACL 1 Search Mask

Before the hashing and searching is done in the **Ingress Configurable ACL 1 Large Table** and **Ingress Configurable ACL 1 Small Table**. The search data is AND:ed with this mask. If a bit in the mask is set to zero then this bit in the lookup will be viewed as do not care. Seperate masks exists for both small and large tables.

Number of Entries :	1
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Address Space :	1124283

Field Description

Bits	Field Name	Description	Default Value
234:0	mask_small	Which bits to compare in the **Ingress Configurable ACL 1 Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	$2^{235} - 1$

Bits	Field Name	Description	Default Value
469:235	mask_large	Which bits to compare in the **Ingress Configurable ACL 1 Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	$2^{235} - 1$

38.11.105 Ingress Configurable ACL 1 Selection

This register selects which result to use when there are multiple hits.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121225

Field Description

Bits	Field Name	Description	Default Value
0	selectTcamOrTable	If set to zero then TCAM answer is selected. If set to one then hash table answer is selected.	0x0
1	selectSmallOrLarge	If set to zero then small hash table is selected. If set to one then large hash table is selected.	0x0

38.11.106 Ingress Configurable ACL 1 Small Table

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries : 512
Number of Addresses per Entry : 16
Type of Operation : Read/Write
Addressing :

address[7:0] :	hash of {compareData }
address[8:8] :	bucket number

Address Space : 317476 to 325667

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
235:1	compareData	The data which shall be compared in this entry.	0x0
236	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0

Bits	Field Name	Description	Default Value
237	forceSendToCpuOrigPkt	This is a result field used when this entry is hit. If packet shall be sent to CPU then setting this bit will force the packet to be the incoming originial packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
238	metaDataValid	This is a result field used when this entry is hit. Is the meta_data field valid.	0x0
254:239	metaData	This is a result field used when this entry is hit. Meta data for packets going to the CPU.	0x0
255	metaDataPrio	This is a result field used when this entry is hit. If multiple ACLs hit this meta_data shall take priority.	0x0
256	dropEnable	This is a result field used when this entry is hit. If set, the packet shall be dropped and the **Ingress Configurable ACL Drop** counter is incremented.	0x0
257	sendToPort	This is a result field used when this entry is hit. Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
261:258	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
262	inputMirror	This is a result field used when this entry is hit. If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
266:263	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
267	imPrio	This is a result field used when this entry is hit. If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
268	noLearning	This is a result field used when this entry is hit. If set this packets MAC SA will not be learned.	0x0
269	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
275:270	counter	This is a result field used when this entry is hit. Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
276	updateTosExp	This is a result field used when this entry is hit. Force TOS/EXP update.	0x0
284:277	newTosExp	This is a result field used when this entry is hit. New TOS/EXP value.	0x0
292:285	tosMask	This is a result field used when this entry is hit. Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0

Bits	Field Name	Description	Default Value
293	updateCfiDei	This is a result field used when this entry is hit. The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
294	newCfiDeiValue	This is a result field used when this entry is hit. The value to update to.	0x0
295	updatePcp	This is a result field used when this entry is hit. The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
298:296	newPcpValue	This is a result field used when this entry is hit. The PCP value to update to.	0x0
299	updateVid	This is a result field used when this entry is hit. The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
311:300	newVidValue	This is a result field used when this entry is hit. The VID value to update to.	0x0
312	updateEType	This is a result field used when this entry is hit. The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
314:313	newEthType	This is a result field used when this entry is hit. Select which TPID to use in the outer VLAN header. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register **Egress Ethernet Type for VLAN tag**.	0x0
315	cfiDeiPrio	This is a result field used when this entry is hit. If multiple updateCfiDei are set and this prio bit is set then this updateCfiDei will be selected.	0x0
316	pcpPrio	This is a result field used when this entry is hit. If multiple updatePcp are set and this prio bit is set then this updatePcp will be selected.	0x0
317	vidPrio	This is a result field used when this entry is hit. If multiple updateVid are set and this prio bit is set then this updateVid will be selected.	0x0
318	ethPrio	This is a result field used when this entry is hit. If multiple updateEType are set and this prio bit is set then this updateEType will be selected.	0x0
319	enableUpdateIp	This is a result field used when this entry is hit. If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this is exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0

Bits	Field Name	Description	Default Value
320	updateSaOrDa	This is a result field used when this entry is hit. Update the SA or DA IPv4 address. The Destiantion IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0
352:321	newIpValue	This is a result field used when this entry is hit. Update the SA or DA IPv4 address value.	0x0
353	enableUpdateL4	This is a result field used when this entry is hit. If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0
354	updateL4SpOrDp	This is a result field used when this entry is hit. Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
370:355	newL4Value	This is a result field used when this entry is hit. Update the L4 SP or DP with this value	0x0
371	natOpValid	This is a result field used when this entry is hit. NAT operation pointer is valid.	0x0
384:372	natOpPtr	This is a result field used when this entry is hit. NAT operation pointer.	0x0
385	natOpPrio	This is a result field used when this entry is hit. If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
388:386	natVersion	This is a result field used when this entry is hit. NAT Entry Version.	0x0
389	ptp	This is a result field used when this entry is hit. When the packet is sent to the CPU the packet will have the PTP bit in the To CPU Tag set to one. The timestamp in the To CPU Tag will also be set to the timestamp from the incoming packet.	0x0
390	tunnelEntry	This is a result field used when this entry is hit. Shall all of these packets enter into a tunnel.	0x0
391	tunnelEntryUcMc	This is a result field used when this entry is hit. Shall this entry point to the **Tunnel Entry Instruction Table** with or without a egress port offset. 0 = Unicast **Tunnel Entry Instruction Table** without offset for each port 1 = Multicast **Tunnel Entry Instruction Table** with offset for each port.	0x0
396:392	tunnelEntryPtr	This is a result field used when this entry is hit. The tunnel entry which this packet shall enter upon exiting the switch.	0x0
397	tunnelEntryPrio	This is a result field used when this entry is hit. If multiple tunnelEntry are set and this prio bit is set then this tunnelEntryPtr will be selected.	0x0

Bits	Field Name	Description	Default Value
398	tunnelExit	This is a result field used when this entry is hit. Shall this packet do a tunnel exit. 0 = No 1 = Yes	0x0
403:399	tunnelExitPtr	This is a result field used when this entry is hit. Pointer to tunnel exit described in **Egress Tunnel Exit Table**.	0x0
404	tunnelExitPrio	This is a result field used when this entry is hit. If multiple tunnelExit are set and this prio bit is set then this tunnelExitPtr will be selected.	0x0
405	cancelCryptoOp	This is a result field used when this entry is hit. Cancel the crypto operation. No crypto operations will be done on this packet. 0 = No. 1 = Yes.	0x0
406	sendToCrypto	This is a result field used when this entry is hit. Do a crypto operation on this packet. 0 = No. 1 = Yes.	0x0
408:407	cryptoProto	This is a result field used when this entry is hit. Crypto protocol. 0 = AH 1 = ESP 2 = MACsec 3 = Reserved.	0x0
409	cryptoOp	This is a result field used when this entry is hit. Crypto operation. 0 = Encrypt 1 = Decrypt	0x0
415:410	secPtr	This is a result field used when this entry is hit. Pointer into Security Association Database. Valid if sendToCrypto is set.	0x0
419:416	cryptoPort	This is a result field used when this entry is hit. Crypto modification port. Before the packet is sent for encryption/decryption there can be packet modifications which are based on the egress port, this is the egress port which will be used for these packet modifications.	0x0
420	cryptoPrio	This is a result field used when this entry is hit. If multiple sendToCrypto actions are set and this prio bit is set then those secPtr,cryptoProto and cryptoOp value will be selected.	0x0
421	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
423:422	color	This is a result field used when this entry is hit. Initial color of the packet if the forceColor field is set.	0x0
424	forceColorPrio	This is a result field used when this entry is hit. If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
425	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
431:426	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0

Bits	Field Name	Description	Default Value
433:432	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
434	forceQueue	This is a result field used when this entry is hit. If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
437:435	eQueue	This is a result field used when this entry is hit. The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
438	forceQueuePrio	This is a result field used when this entry is hit. If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0
439	forceVidValid	This is a result field used when this entry is hit. Override the Ingress VID, see chapter VLAN Processing.	0x0
451:440	forceVid	This is a result field used when this entry is hit. The new Ingress VID.	0x0
452	forceVidPrio	This is a result field used when this entry is hit. If multiple forceVid are set and this prio bit is set then this forceVid value will be selected.	0x0

38.11.107 Ingress Configurable ACL 1 TCAM

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.

Number of Entries :	8
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	1124155 to 1124282

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
235:1	mask	Which bits to compare in this entry.	$2^{235}-1$
470:236	compareData	The data which shall be compared in this entry. Observe that this compare data must be AND:ed by software before the entry is searched. The hardware does not do the AND between mask and compareData (In order to save area).	0x0

38.11.108 Ingress Configurable ACL 1 TCAM Answer

This is the table holding the answer for the **Ingress Configurable ACL 1 TCAM**.

Number of Entries : 8
Number of Addresses per Entry : 8
Type of Operation : Read/Write
Addressing : **Ingress Configurable ACL 1 TCAM** hit index
Address Space : 325668 to 325731

Field Description

Bits	Field Name	Description	Default Value
0	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
1	forceSendToCpuOrigPkt	If packet shall be sent to CPU then setting this bit will force the packet to be the incoming originial packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
2	metaDataValid	Is the meta_data field valid.	0x0
18:3	metaData	Meta data for packets going to the CPU.	0x0
19	metaDataPrio	If multiple ACLs hit this meta_data shall take priority.	0x0
20	dropEnable	If set, the packet shall be dropped and the **Ingress Configurable ACL Drop** counter is incremented.	0x0
21	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
25:22	destPort	The port which the packet shall be sent to.	0x0
26	inputMirror	If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
30:27	destInputMirror	Destination physical port for input mirroring.	0x0
31	imPrio	If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
32	noLearning	If set this packets MAC SA will not be learned.	0x0
33	updateCounter	When set the selected statistics counter will be updated.	0x0
39:34	counter	Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
40	updateTosExp	Force TOS/EXP update.	0x0
48:41	newTosExp	New TOS/EXP value.	0x0
56:49	tosMask	Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
57	updateCfiDei	The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
58	newCfiDeiValue	The value to update to.	0x0
59	updatePcp	The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0

Bits	Field Name	Description	Default Value
62:60	newPcpValue	The PCP value to update to.	0x0
63	updateVid	The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
75:64	newVidValue	The VID value to update to.	0x0
76	updateEType	The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
78:77	newEthType	Select which TPID to use in the outer VLAN header. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register **Egress Ethernet Type for VLAN tag**.	0x0
79	cfiDeiPrio	If multiple updateCfiDei are set and this prio bit is set then this updateCfiDei will be selected.	0x0
80	pcpPrio	If multiple updatePcp are set and this prio bit is set then this updatePcp will be selected.	0x0
81	vidPrio	If multiple updateVid are set and this prio bit is set then this updateVid will be selected.	0x0
82	ethPrio	If multiple updateEType are set and this prio bit is set then this updateEType will be selected.	0x0
83	enableUpdateIp	If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this is exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0
84	updateSaOrDa	Update the SA or DA IPv4 address. The Destiantion IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0
116:85	newIpValue	Update the SA or DA IPv4 address value.	0x0
117	enableUpdateL4	If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0
118	updateL4SpOrDp	Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
134:119	newL4Value	Update the L4 SP or DP with this value	0x0
135	natOpValid	NAT operation pointer is valid.	0x0
148:136	natOpPtr	NAT operation pointer.	0x0
149	natOpPrio	If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
152:150	natVersion	NAT Entry Version.	0x0

Bits	Field Name	Description	Default Value
153	ptp	When the packet is sent to the CPU the packet will have the PTP bit in the To CPU Tag set to one. The timestamp in the To CPU Tag will also be set to the timestamp from the incoming packet.	0x0
154	tunnelEntry	Shall all of these packets enter into a tunnel.	0x0
155	tunnelEntryUcMc	Shall this entry point to the **Tunnel Entry Instruction Table** with or without a egress port offset. 0 = Unicast **Tunnel Entry Instruction Table** without offset for each port 1 = Multicast **Tunnel Entry Instruction Table** with offset for each port.	0x0
160:156	tunnelEntryPtr	The tunnel entry which this packet shall enter upon exiting the switch.	0x0
161	tunnelEntryPrio	If multiple tunnelEntry are set and this prio bit is set then this tunnelEntryPtr will be selected.	0x0
162	tunnelExit	Shall this packet do a tunnel exit. 0 = No 1 = Yes	0x0
167:163	tunnelExitPtr	Pointer to tunnel exit described in **Egress Tunnel Exit Table**.	0x0
168	tunnelExitPrio	If multiple tunnelExit are set and this prio bit is set then this tunnelExitPtr will be selected.	0x0
169	cancelCryptoOp	Cancel the crypto operation. No crypto operations will be done on this packet. 0 = No. 1 = Yes.	0x0
170	sendToCrypto	Do a crypto operation on this packet. 0 = No. 1 = Yes.	0x0
172:171	cryptoProto	Crypto protocol. 0 = AH 1 = ESP 2 = MACsec 3 = Reserved.	0x0
173	cryptoOp	Crypto operation. 0 = Encrypt 1 = Decrypt	0x0
179:174	secPtr	Pointer into Security Association Database. Valid if sendToCrypto is set.	0x0
183:180	cryptoPort	Crypto modification port. Before the packet is sent for encryption/decryption there can be packet modifications which are based on the egress port, this is the egress port which will be used for these packet modifications.	0x0
184	cryptoPrio	If multiple sendToCrypto actions are set and this prio bit is set then those secPtr,cryptoProto and cryptoOp value will be selected.	0x0
185	forceColor	If set, the packet shall have a forced color.	0x0
187:186	color	Initial color of the packet if the forceColor field is set.	0x0
188	forceColorPrio	If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
189	mmpValid	If set, this entry contains a valid MMP pointer	0x0

Bits	Field Name	Description	Default Value
195:190	mmpPtr	Ingress MMP pointer.	0x0
197:196	mmpOrder	Ingress MMP pointer order.	0x0
198	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
201:199	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
202	forceQueuePrio	If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0
203	forceVidValid	Override the Ingress VID, see chapter VLAN Processing.	0x0
215:204	forceVid	The new Ingress VID.	0x0
216	forceVidPrio	If multiple forceVid are set and this prio bit is set then this forceVid value will be selected.	0x0

38.11.109 Ingress Configurable ACL 2 Large Table

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries : 128
Number of Addresses per Entry : 32
Type of Operation : Read/Write
Addressing :

address[5:0]	:	hash of {compareData }
address[6:6]	:	bucket number

Address Space : 325860 to 329955

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
560:1	compareData	The data which shall be compared in this entry.	0x0
561	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
562	forceSendToCpuOrigPkt	This is a result field used when this entry is hit. If packet shall be sent to CPU then setting this bit will force the packet to be the incoming originial packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
563	metaDataValid	This is a result field used when this entry is hit. Is the meta_data field valid.	0x0
579:564	metaData	This is a result field used when this entry is hit. Meta data for packets going to the CPU.	0x0
580	metaDataPrio	This is a result field used when this entry is hit. If multiple ACLs hit this meta_data shall take priority.	0x0
581	dropEnable	This is a result field used when this entry is hit. If set, the packet shall be dropped and the **Ingress Configurable ACL Drop** counter is incremented.	0x0

Bits	Field Name	Description	Default Value
582	sendToPort	This is a result field used when this entry is hit. Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
586:583	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
587	inputMirror	This is a result field used when this entry is hit. If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
591:588	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
592	imPrio	This is a result field used when this entry is hit. If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
593	noLearning	This is a result field used when this entry is hit. If set this packets MAC SA will not be learned.	0x0
594	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
600:595	counter	This is a result field used when this entry is hit. Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
601	updateTosExp	This is a result field used when this entry is hit. Force TOS/EXP update.	0x0
609:602	newTosExp	This is a result field used when this entry is hit. New TOS/EXP value.	0x0
617:610	tosMask	This is a result field used when this entry is hit. Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
618	updateCfiDei	This is a result field used when this entry is hit. The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
619	newCfiDeiValue	This is a result field used when this entry is hit. The value to update to.	0x0
620	updatePcp	This is a result field used when this entry is hit. The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
623:621	newPcpValue	This is a result field used when this entry is hit. The PCP value to update to.	0x0

Bits	Field Name	Description	Default Value
624	updateVid	This is a result field used when this entry is hit. The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
636:625	newVidValue	This is a result field used when this entry is hit. The VID value to update to.	0x0
637	updateEType	This is a result field used when this entry is hit. The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
639:638	newEthType	This is a result field used when this entry is hit. Select which TPID to use in the outer VLAN header. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register **Egress Ethernet Type for VLAN tag**.	0x0
640	cfiDeiPrio	This is a result field used when this entry is hit. If multiple updateCfiDei are set and this prio bit is set then this updateCfiDei will be selected.	0x0
641	pcpPrio	This is a result field used when this entry is hit. If multiple updatePcp are set and this prio bit is set then this updatePcp will be selected.	0x0
642	vidPrio	This is a result field used when this entry is hit. If multiple updateVid are set and this prio bit is set then this updateVid will be selected.	0x0
643	ethPrio	This is a result field used when this entry is hit. If multiple updateEType are set and this prio bit is set then this updateEType will be selected.	0x0
644	enableUpdateIp	This is a result field used when this entry is hit. If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this is exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0
645	updateSaOrDa	This is a result field used when this entry is hit. Update the SA or DA IPv4 address. The Destiantion IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0
677:646	newIpValue	This is a result field used when this entry is hit. Update the SA or DA IPv4 address value.	0x0
678	enableUpdateL4	This is a result field used when this entry is hit. If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0

Bits	Field Name	Description	Default Value
679	updateL4SpOrDp	This is a result field used when this entry is hit. Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
695:680	newL4Value	This is a result field used when this entry is hit. Update the L4 SP or DP with this value	0x0
696	natOpValid	This is a result field used when this entry is hit. NAT operation pointer is valid.	0x0
709:697	natOpPtr	This is a result field used when this entry is hit. NAT operation pointer.	0x0
710	natOpPrio	This is a result field used when this entry is hit. If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
713:711	natVersion	This is a result field used when this entry is hit. NAT Entry Version.	0x0
714	ptp	This is a result field used when this entry is hit. When the packet is sent to the CPU the packet will have the PTP bit in the To CPU Tag set to one. The timestamp in the To CPU Tag will also be set to the timestamp from the incoming packet.	0x0
715	tunnelEntry	This is a result field used when this entry is hit. Shall all of these packets enter into a tunnel.	0x0
716	tunnelEntryUcMc	This is a result field used when this entry is hit. Shall this entry point to the **Tunnel Entry Instruction Table** with or without a egress port offset. 0 = Unicast **Tunnel Entry Instruction Table** without offset for each port 1 = Multicast **Tunnel Entry Instruction Table** with offset for each port.	0x0
721:717	tunnelEntryPtr	This is a result field used when this entry is hit. The tunnel entry which this packet shall enter upon exiting the switch.	0x0
722	tunnelEntryPrio	This is a result field used when this entry is hit. If multiple tunnelEntry are set and this prio bit is set then this tunnelEntryPtr will be selected.	0x0
723	cancelCryptoOp	This is a result field used when this entry is hit. Cancel the crypto operation. No crypto operations will be done on this packet. 0 = No. 1 = Yes.	0x0
724	sendToCrypto	This is a result field used when this entry is hit. Do a crypto operation on this packet. 0 = No. 1 = Yes.	0x0
726:725	cryptoProto	This is a result field used when this entry is hit. Crypto protocol. 0 = AH 1 = ESP 2 = MACsec 3 = Reserved.	0x0
727	cryptoOp	This is a result field used when this entry is hit. Crypto operation. 0 = Encrypt 1 = Decrypt	0x0

Bits	Field Name	Description	Default Value
733:728	secPtr	This is a result field used when this entry is hit. Pointer into Security Association Database. Valid if sendToCrypto is set.	0x0
737:734	cryptoPort	This is a result field used when this entry is hit. Crypto modification port. Before the packet is sent for encryption/decryption there can be packet modifications which are based on the egress port, this is the egress port which will be used for these packet modifications.	0x0
738	cryptoPrio	This is a result field used when this entry is hit. If multiple sendToCrypto actions are set and this prio bit is set then those secPtr,cryptoProto and cryptoOp value will be selected.	0x0
739	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
741:740	color	This is a result field used when this entry is hit. Initial color of the packet if the forceColor field is set.	0x0
742	forceColorPrio	This is a result field used when this entry is hit. If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
743	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
749:744	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0
751:750	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0
752	forceQueue	This is a result field used when this entry is hit. If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
755:753	eQueue	This is a result field used when this entry is hit. The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
756	forceQueuePrio	This is a result field used when this entry is hit. If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0
757	forceVidValid	This is a result field used when this entry is hit. Override the Ingress VID, see chapter VLAN Processing.	0x0
769:758	forceVid	This is a result field used when this entry is hit. The new Ingress VID.	0x0
770	forceVidPrio	This is a result field used when this entry is hit. If multiple forceVid are set and this prio bit is set then this forceVid value will be selected.	0x0

38.11.110 Ingress Configurable ACL 2 Pre Lookup

The pre ACL lookup allows the user to defined a specific rules for certain packet types in the ACL engine 2. Setting the valid bit and a new rule will override the default rule pointer from the source port table.

Number of Entries : 128
Type of Operation : Read/Write
Addressing :

Address bits [1:0]	Value from **preLookupAclBits**.
Address bits [3:2]	Number of VLANs in incoming Packet.
Address bits [4:4]	L2 Type Of Packet. 0 = Others - Not listed in this list. 1 = MACsec
Address bits [6:5]	L3 Type Of Packet. 0 = IPv4 1 = IPv6 2 = MPLS 3 = Not IPv4, IPv6 or MPLS

Address Space : 325732 to 325859

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. If not then use default port rule.	0x0
2:1	rulePtr	If the valid is entry then this rule pointer will be used.	0x0

38.11.111 Ingress Configurable ACL 2 Rules Setup

The rules are setup by selecting which fields shall be used in the ACL search. Each rule has a fixed number of fields. The fieldSelectBitmask has one bit for each field. The first 22 fields (bits) which are set to one are selected. It is not allowed to set more than 22 bit in the bitmask. The fields are described in ACL Fields

Number of Entries : 4
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Addressing : ACL rule pointer
Address Space : 1122427 to 1122434

Field Description

Bits	Field Name	Description	Default Value
33:0	fieldSelectBitmask	Bitmask of which fields to select. Set a bit to one to select this specific field, set zero to not select field. At Maximum 22 bits should be set.	0x0

38.11.112 Ingress Configurable ACL 2 Search Mask

Before the hashing and searching is done in the **Ingress Configurable ACL 2 Large Table** and **Ingress Configurable ACL 2 Small Table**. The search data is AND:ed with this mask. If a bit in the mask is set to zero then this bit in the lookup will be viewed as do not care. Seperate masks exists for both small and large tables.

Number of Entries :	1
Number of Addresses per Entry :	64
Type of Operation :	Read/Write
Address Space :	1124451

Field Description

Bits	Field Name	Description	Default Value
559:0	mask_small	Which bits to compare in the **Ingress Configurable ACL 2 Small Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	$2^{560} - 1$
1119:560	mask_large	Which bits to compare in the **Ingress Configurable ACL 2 Large Table** lookup. A bit set to 1 means the corresponding bit in the search data is compared and 0 means the bit is ignored.	$2^{560} - 1$

38.11.113 Ingress Configurable ACL 2 Selection

This register selects which result to use when there are multiple hits.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	1121226

Field Description

Bits	Field Name	Description	Default Value
0	selectTcamOrTable	If set to zero then TCAM answer is selected. If set to one then hash table answer is selected.	0x0
1	selectSmallOrLarge	If set to zero then small hash table is selected. If set to one then large hash table is selected.	0x0

38.11.114 Ingress Configurable ACL 2 Small Table

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.. If multiple buckets match then the result from the highest entry is selected.

Number of Entries :	32
Number of Addresses per Entry :	32
Type of Operation :	Read/Write
Addressing :	address[3:0] : hash of {compareData } address[4:4] : bucket number
Address Space :	329956 to 330979

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
560:1	compareData	The data which shall be compared in this entry.	0x0
561	sendToCpu	This is a result field used when this entry is hit. If set, the packet shall be sent to the CPU port.	0x0
562	forceSendToCpuOrigPkt	This is a result field used when this entry is hit. If packet shall be sent to CPU then setting this bit will force the packet to be the incoming originial packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
563	metaDataValid	This is a result field used when this entry is hit. Is the meta_data field valid.	0x0
579:564	metaData	This is a result field used when this entry is hit. Meta data for packets going to the CPU.	0x0
580	metaDataPrio	This is a result field used when this entry is hit. If multiple ACLs hit this meta_data shall take priority.	0x0
581	dropEnable	This is a result field used when this entry is hit. If set, the packet shall be dropped and the **Ingress Configurable ACL Drop** counter is incremented.	0x0
582	sendToPort	This is a result field used when this entry is hit. Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
586:583	destPort	This is a result field used when this entry is hit. The port which the packet shall be sent to.	0x0
587	inputMirror	This is a result field used when this entry is hit. If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
591:588	destInputMirror	This is a result field used when this entry is hit. Destination physical port for input mirroring.	0x0
592	imPrio	This is a result field used when this entry is hit. If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
593	noLearning	This is a result field used when this entry is hit. If set this packets MAC SA will not be learned.	0x0
594	updateCounter	This is a result field used when this entry is hit. When set the selected statistics counter will be updated.	0x0
600:595	counter	This is a result field used when this entry is hit. Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
601	updateTosExp	This is a result field used when this entry is hit. Force TOS/EXP update.	0x0
609:602	newTosExp	This is a result field used when this entry is hit. New TOS/EXP value.	0x0

Bits	Field Name	Description	Default Value
617:610	tosMask	This is a result field used when this entry is hit. Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
618	updateCfiDei	This is a result field used when this entry is hit. The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
619	newCfiDeiValue	This is a result field used when this entry is hit. The value to update to.	0x0
620	updatePcp	This is a result field used when this entry is hit. The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
623:621	newPcpValue	This is a result field used when this entry is hit. The PCP value to update to.	0x0
624	updateVid	This is a result field used when this entry is hit. The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
636:625	newVidValue	This is a result field used when this entry is hit. The VID value to update to.	0x0
637	updateEType	This is a result field used when this entry is hit. The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
639:638	newEthType	This is a result field used when this entry is hit. Select which TPID to use in the outer VLAN header. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register **Egress Ethernet Type for VLAN tag**.	0x0
640	cfiDeiPrio	This is a result field used when this entry is hit. If multiple updateCfiDei are set and this prio bit is set then this updateCfiDei will be selected.	0x0
641	pcpPrio	This is a result field used when this entry is hit. If multiple updatePcp are set and this prio bit is set then this updatePcp will be selected.	0x0
642	vidPrio	This is a result field used when this entry is hit. If multiple updateVid are set and this prio bit is set then this updateVid will be selected.	0x0
643	ethPrio	This is a result field used when this entry is hit. If multiple updateEType are set and this prio bit is set then this updateEType will be selected.	0x0

Bits	Field Name	Description	Default Value
644	enableUpdateIp	This is a result field used when this entry is hit. If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this is exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0
645	updateSaOrDa	This is a result field used when this entry is hit. Update the SA or DA IPv4 address. The Destiantion IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0
677:646	newIpValue	This is a result field used when this entry is hit. Update the SA or DA IPv4 address value.	0x0
678	enableUpdateL4	This is a result field used when this entry is hit. If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0
679	updateL4SpOrDp	This is a result field used when this entry is hit. Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
695:680	newL4Value	This is a result field used when this entry is hit. Update the L4 SP or DP with this value	0x0
696	natOpValid	This is a result field used when this entry is hit. NAT operation pointer is valid.	0x0
709:697	natOpPtr	This is a result field used when this entry is hit. NAT operation pointer.	0x0
710	natOpPrio	This is a result field used when this entry is hit. If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
713:711	natVersion	This is a result field used when this entry is hit. NAT Entry Version.	0x0
714	ptp	This is a result field used when this entry is hit. When the packet is sent to the CPU the packet will have the PTP bit in the To CPU Tag set to one. The timestamp in the To CPU Tag will also be set to the timestamp from the incoming packet.	0x0
715	tunnelEntry	This is a result field used when this entry is hit. Shall all of these packets enter into a tunnel.	0x0
716	tunnelEntryUcMc	This is a result field used when this entry is hit. Shall this entry point to the **Tunnel Entry Instruction Table** with or without a egress port offset. 0 = Unicast **Tunnel Entry Instruction Table** without offset for each port 1 = Multicast **Tunnel Entry Instruction Table** with offset for each port.	0x0

Bits	Field Name	Description	Default Value
721:717	tunnelEntryPtr	This is a result field used when this entry is hit. The tunnel entry which this packet shall enter upon exiting the switch.	0x0
722	tunnelEntryPrio	This is a result field used when this entry is hit. If multiple tunnelEntry are set and this prio bit is set then this tunnelEntryPtr will be selected.	0x0
723	cancelCryptoOp	This is a result field used when this entry is hit. Cancel the crypto operation. No crypto operations will be done on this packet. 0 = No. 1 = Yes.	0x0
724	sendToCrypto	This is a result field used when this entry is hit. Do a crypto operation on this packet. 0 = No. 1 = Yes.	0x0
726:725	cryptoProto	This is a result field used when this entry is hit. Crypto protocol. 0 = AH 1 = ESP 2 = MACsec 3 = Reserved.	0x0
727	cryptoOp	This is a result field used when this entry is hit. Crypto operation. 0 = Encrypt 1 = Decrypt	0x0
733:728	secPtr	This is a result field used when this entry is hit. Pointer into Security Association Database. Valid if sendToCrypto is set.	0x0
737:734	cryptoPort	This is a result field used when this entry is hit. Crypto modification port. Before the packet is sent for encryption/decryption there can be packet modifications which are based on the egress port, this is the egress port which will be used for these packet modifications.	0x0
738	cryptoPrio	This is a result field used when this entry is hit. If multiple sendToCrypto actions are set and this prio bit is set then those secPtr,cryptoProto and cryptoOp value will be selected.	0x0
739	forceColor	This is a result field used when this entry is hit. If set, the packet shall have a forced color.	0x0
741:740	color	This is a result field used when this entry is hit. Initial color of the packet if the forceColor field is set.	0x0
742	forceColorPrio	This is a result field used when this entry is hit. If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
743	mmpValid	This is a result field used when this entry is hit. If set, this entry contains a valid MMP pointer	0x0
749:744	mmpPtr	This is a result field used when this entry is hit. Ingress MMP pointer.	0x0
751:750	mmpOrder	This is a result field used when this entry is hit. Ingress MMP pointer order.	0x0

Bits	Field Name	Description	Default Value
752	forceQueue	This is a result field used when this entry is hit. If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
755:753	eQueue	This is a result field used when this entry is hit. The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
756	forceQueuePrio	This is a result field used when this entry is hit. If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0
757	forceVidValid	This is a result field used when this entry is hit. Override the Ingress VID, see chapter VLAN Processing.	0x0
769:758	forceVid	This is a result field used when this entry is hit. The new Ingress VID.	0x0
770	forceVidPrio	This is a result field used when this entry is hit. If multiple forceVid are set and this prio bit is set then this forceVid value will be selected.	0x0

38.11.115 Ingress Configurable ACL 2 TCAM

This table is used for the configurable ACL lookup. A hash is calculated on the selected fields from the packet header. The hash is then used as index into this table.

Number of Entries :	16
Number of Addresses per Entry :	64
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	1124515 to 1125538

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
560:1	mask	Which bits to compare in this entry.	$2^{560}-1$
1120:561	compareData	The data which shall be compared in this entry. Observe that this compare data must be AND:ed by software before the entry is searched. The hardware does not do the AND between mask and compareData (In order to save area).	0x0

38.11.116 Ingress Configurable ACL 2 TCAM Answer

This is the table holding the answer for the **Ingress Configurable ACL 2 TCAM**.

Number of Entries :	16
Number of Addresses per Entry :	8
Type of Operation :	Read/Write
Addressing :	**Ingress Configurable ACL 2 TCAM** hit index
Address Space :	330980 to 331107

Field Description

Bits	Field Name	Description	Default Value
0	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
1	forceSendToCpuOrigPkt	If packet shall be sent to CPU then setting this bit will force the packet to be the incoming originial packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
2	metaDataValid	Is the meta_data field valid.	0x0
18:3	metaData	Meta data for packets going to the CPU.	0x0
19	metaDataPrio	If multiple ACLs hit this meta_data shall take priority.	0x0
20	dropEnable	If set, the packet shall be dropped and the **Ingress Configurable ACL Drop** counter is incremented.	0x0
21	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
25:22	destPort	The port which the packet shall be sent to.	0x0
26	inputMirror	If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
30:27	destInputMirror	Destination physical port for input mirroring.	0x0
31	imPrio	If multiple input mirror are set and this prio bit is set then this input mirror will be selected.	0x0
32	noLearning	If set this packets MAC SA will not be learned.	0x0
33	updateCounter	When set the selected statistics counter will be updated.	0x0
39:34	counter	Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
40	updateTosExp	Force TOS/EXP update.	0x0
48:41	newTosExp	New TOS/EXP value.	0x0
56:49	tosMask	Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
57	updateCfiDei	The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
58	newCfiDeiValue	The value to update to.	0x0
59	updatePcp	The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
62:60	newPcpValue	The PCP value to update to.	0x0
63	updateVid	The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
75:64	newVidValue	The VID value to update to.	0x0

Bits	Field Name	Description	Default Value
76	updateEType	The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
78:77	newEthType	Select which TPID to use in the outer VLAN header. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register **Egress Ethernet Type for VLAN tag**.	0x0
79	cfiDeiPrio	If multiple updateCfiDei are set and this prio bit is set then this updateCfiDei will be selected.	0x0
80	pcpPrio	If multiple updatePcp are set and this prio bit is set then this updatePcp will be selected.	0x0
81	vidPrio	If multiple updateVid are set and this prio bit is set then this updateVid will be selected.	0x0
82	ethPrio	If multiple updateEType are set and this prio bit is set then this updateEType will be selected.	0x0
83	enableUpdateIp	If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this is exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0
84	updateSaOrDa	Update the SA or DA IPv4 address. The Destiantion IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0
116:85	newIpValue	Update the SA or DA IPv4 address value.	0x0
117	enableUpdateL4	If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0
118	updateL4SpOrDp	Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
134:119	newL4Value	Update the L4 SP or DP with this value	0x0
135	natOpValid	NAT operation pointer is valid.	0x0
148:136	natOpPtr	NAT operation pointer.	0x0
149	natOpPrio	If multiple natOpValid are set and this prio bit is set then this natOpPtr value will be selected.	0x0
152:150	natVersion	NAT Entry Version.	0x0
153	ptp	When the packet is sent to the CPU the packet will have the PTP bit in the To CPU Tag set to one. The timestamp in the To CPU Tag will also be set to the timestamp from the incoming packet.	0x0
154	tunnelEntry	Shall all of these packets enter into a tunnel.	0x0

Bits	Field Name	Description	Default Value
155	tunnelEntryUcMc	Shall this entry point to the **Tunnel Entry Instruction Table** with or without a egress port offset. 0 = Unicast **Tunnel Entry Instruction Table** without offset for each port 1 = Multicast **Tunnel Entry Instruction Table** with offset for each port.	0x0
160:156	tunnelEntryPtr	The tunnel entry which this packet shall enter upon exiting the switch.	0x0
161	tunnelEntryPrio	If multiple tunnelEntry are set and this prio bit is set then this tunnelEntryPtr will be selected.	0x0
162	cancelCryptoOp	Cancel the crypto operation. No crypto operations will be done on this packet. 0 = No. 1 = Yes.	0x0
163	sendToCrypto	Do a crypto operation on this packet. 0 = No. 1 = Yes.	0x0
165:164	cryptoProto	Crypto protocol. 0 = AH 1 = ESP 2 = MACsec 3 = Reserved.	0x0
166	cryptoOp	Crypto operation. 0 = Encrypt 1 = Decrypt	0x0
172:167	secPtr	Pointer into Security Association Database. Valid if sendToCrypto is set.	0x0
176:173	cryptoPort	Crypto modification port. Before the packet is sent for encryption/decryption there can be packet modifications which are based on the egress port, this is the egress port which will be used for these packet modifications.	0x0
177	cryptoPrio	If multiple sendToCrypto actions are set and this prio bit is set then those secPtr,cryptoProto and cryptoOp value will be selected.	0x0
178	forceColor	If set, the packet shall have a forced color.	0x0
180:179	color	Initial color of the packet if the forceColor field is set.	0x0
181	forceColorPrio	If multiple forceColor are set and this prio bit is set then this forceVid value will be selected.	0x0
182	mmpValid	If set, this entry contains a valid MMP pointer	0x0
188:183	mmpPtr	Ingress MMP pointer.	0x0
190:189	mmpOrder	Ingress MMP pointer order.	0x0
191	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
194:192	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
195	forceQueuePrio	If multiple forceQueue are set and this prio bit is set then this forceQueue value will be selected.	0x0
196	forceVidValid	Override the Ingress VID, see chapter VLAN Processing.	0x0
208:197	forceVid	The new Ingress VID.	0x0

Bits	Field Name	Description	Default Value
209	forceVidPrio	If multiple forceVid are set and this prio bit is set then this forceVid value will be selected.	0x0

38.11.117 Ingress Drop Options

Options to enable or disable learning when the the L2 forwarding process drops the packet.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1125539

Field Description

Bits	Field Name	Description	Default Value
0	learnL2DestDrop	Allow learning when L2 Destination Table drops the packet.	0x0
1	learnL2FloodDrop	Allow learning when the packet is dropped due to unknown DA.	0x0
2	learnL2DestVlanMemberDrop	Allow learning when the packt is dropped due to destination VLAN membership check.	0x1
3	learnL2HairpinDrop	Allow learning when the packet is dropped due to hairpin configurations.	0x0

38.11.118 Ingress Egress Port Packet Type Filter

This sets up which packets are to be dropped or allowed to be transmitted on each of the egress ports. This filtering is done after the source port tables VLAN operation and the VLAN tables VLAN operation. Notice this filter applies to L2 L3 forwarding result only, any other special rules could bypass it (traffic to/from CPU port, classifications, etc). Packets dropped due to this filter will be counted in **Ingress-Egress Packet Filtering Drop**.

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 1118924 to 1118934

Field Description

Bits	Field Name	Description	Default Value
0	dropCtaggedVlans	Drop or allow customer VLAN tagged packets on this egress port. Will only drop packets that has exactly one VLAN tag. Must set moreThanOneVlans when this is used. Note that after a VLAN push operation the pushed VLAN will be regarded as a C-VLAN. 0 = Allow C-VLANs. 1 = Drop C-VLANs.	0x0

Bits	Field Name	Description	Default Value
1	dropStaggedVlans	Drop or allow service VLAN tagged packets on this egress port. Must set moreThanOneVlans when this is used. Note that after a VLAN push operation the pushed VLAN will be regarded as a C-VLAN. 0 = Allow S-VLANs. 1 = Drop S-VLANs.	0x0
2	moreThanOneVlans	When filtering with dropCtaggedVlans or dropStaggedVlans then this field must be set to 1.	0x0
3	dropSingleTaggedVlans	Drop or Allow packets that are VLAN untagged on this egress port. 0 = Allow untagged packets. 1 = Drop untagged packets.	0x0
4	dropUntaggedVlans	Drop or Allow packets that are VLAN untagged on this egress port. 0 = Allow untagged packets. 1 = Drop untagged packets.	0x0
5	dropIPv4Packets	Drop or allow IPv4 packets on this egress port. 0 = Allow IPv4 packets. 1 = Drop IPv4 packets.	0x0
6	dropIPv6Packets	Drop or allow IPv6 packets on this egress port. 0 = Allow IPv6 packets. 1 = Drop IPv6 packets.	0x0
7	dropMPLSPackets	Drop or allow MPLS packets on this source port. 0 = Allow MPLS packets. 1 = Drop MPLS packets.	0x0
8	dropIPv4MulticastPackets	Drop or allow IPv4 Multicast packets on this egress port. 0 = Allow IPv4 MC packets. 1 = 1 = Drop IPv4 MC packets.	0x0
9	dropIPv6MulticastPackets	Drop or allow IPv6 Multicast packets on this egress port. 0 = Allow IPv6 MC packets. 1 = Drop IPv6 MC packets.	0x0
10	dropL2BroadcastFrames	Drop or allow L2 broadcast packets on this egress port. 0 = Allow L2 broadcast packets. 1 = Drop L2 broadcast packets.	0x0
11	dropL2FloodingFrames	Drop or allow L2 flooding packets on this egress port. Observe that this rule takes the **unknownL2McFilterRule** into account. 0 = Allow L2 flooding packets. 1 = Drop L2 flooding packets.	0x0
12	dropL2MulticastFrames	Drop or allow L2 multicast packets on this egress port. Observe that this L2 multicast bit takes the register **L2 Multicast Handling** into account to determine if this packet is a L2 multicast packet or not. 0 = Allow L2 multicast packets 1 = Drop L2 multicast packets.	0x0

Bits	Field Name	Description	Default Value
13	dropDualTaggedVlans	Drop or allow packets with has more than one VLAN tag on this egress port. 0 = Allow packets which has more than one VLAN tag. 1 = Drop packets which has more than one VLAN tag.	0x0
14	dropCStaggedVlans	Drop or allow packets with has a C-VLAN followed by a S-VLAN tagged on this egress port. Note that after a VLAN push operation the pushed VLAN will be regarded as a C-VLAN. 0 = Allow packets which has a C-VLAN tag followed by a S-VLAN tag. 1 = Drop packets which has a C-VLAN tag followed by a S-VLAN tag.	0x0
15	dropSCtaggedVlans	Drop or allow packets with has a S-VLAN followed by a C-VLAN tagged on this egress port. Note that after a VLAN push operation the pushed VLAN will be regarded as a C-VLAN. 0 = Allow packets which has a S-VLAN followed by a C-VLAN tag. 1 = Drop packets which has a S-VLAN tag followed by a C-VLAN tag.	0x0
16	dropCCtaggedVlans	Drop or allow packets with has a C-VLAN followed by a C-VLAN tagged on this egress port. Note that after a VLAN push operation the pushed VLAN will be regarded as a C-VLAN. 0 = Allow packets which has a C-VLAN tag followed by a C-VLAN tag. 1 = Drop packets which has a C-VLAN tag followed by a C-VLAN tag.	0x0
17	dropSStaggedVlans	Drop or allow packets with has a S-VLAN followed by a S-VLAN tagged on this egress port. Note that after a VLAN push operation the pushed VLAN will be regarded as a C-VLAN. 0 = Allow packets which has a S-VLAN tag followed by a S-VLAN tag. 1 = Drop packets which has a S-VLAN tag followed by a S-VLAN tag.	0x0
18	dropRouted	Drop or allow packets which has been routed on this egress port. 0 = Allow packets which has been routed. 1 = Drop packets which has been routed.	0x0
29:19	srcPortFilter	Each egress port has an optional way of ensuring that a specific source port does not send out a packet on a specific egress port. By setting a bit in this port mask, the packets originating from that source port will be dropped and not be allowed to reach this egress port.	0x0

38.11.119 Ingress Ethernet Type for VLAN tag

When decoding VLAN tags, if the Ethernet Type matches the **typeValue** it will be considered to be a VLAN tag in addition to the standard values of 0x8100 and 0x88A8. The **type** field determines if the VLAN should be regarded as a Service VLAN or Customer VLAN.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121218

Field Description

Bits	Field Name	Description	Default Value
15:0	typeValue	Ethernet Type value.	0xffff
16	type	User defined VLAN type. 0 = Customer VLAN. 1 = Service VLAN.	0x0
17	valid	User defined VLAN is valid. 0 = Not Valid. 1 = Valid.	0x0

38.11.120 Ingress Function Control

This register controls which functions a packet shall execute in the ingress packet processing pipeline.

Number of Entries : 2
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Addressing : See **Ingress Function Pointer Source Port** and see function control chapter.
Address Space : 1121271 to 1121274

Field Description

Bits	Field Name	Description	Default Value
0	doL2L3Lookup	Shall this packet do L2 and L3 Lookups? 0 = No. 1 = Yes.	0x1
1	noLearning	Shall this packet not be learned. Default is to learn the packet (=0)? 0 = No. 1 = Yes.	0x0
2	drop	Force the packet to be dropped, updates the **Ingress Functional Control Drops** counter. 0 = No. 1 = Yes.	0x0
3	bypassSendToCpu	In the L2,L3,L4 packet decoding there are a number of registers which can send packets to the CPU, this bit allows these send-to-cpu to be ignored / bypassed. Shall the packet decoders send-to-cpu options be bypassed? 0 = No. 1 = Yes.	0x0
4	allowRouting	Shall the packet be allowed to be routed? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
5	enableReservedDmac	Shall the reserved DMAC range check be performed? 0 = No. 1 = Yes.	0x1
6	enableReservedSmac	Shall the reserved SMAC range check be performed? 0 = No. 1 = Yes.	0x1
7	enableSrcPortVlanOps	Shall the source port vlan operation be carried out? 0 = No. 1 = Yes.	0x1
8	enableTunnelExit	Shall this packet do the tunnel exit lookup? 0 = No. 1 = Yes.	0x1
9	allowSmon	Shall this packet be allowed to update the SMON counters? 0 = No. 1 = Yes.	0x1
10	enableIngressPortFilter	Shall the packet be subjected to the ingress port filter? 0 = No. 1 = Yes.	0x1
11	ingressAclEnabled	Shall the ingress ACL operation be done? 0 = No. 1 = Yes.	0x1
12	checkIngressSpt	Shall the ingress spanning tree operation be done? 0 = No. 1 = Yes.	0x1
13	checkEgressSpt	Shall the egress spanning tree operation be done? 0 = No. 1 = Yes.	0x1
14	allowVlanPortMembershipDrop	Shall the VLAN table drop packets due to packets not being packets not being part of VLAN PortMembership, this affects both the source port and egress port(s) being checked on the vlan-port-membership mask. 0 = No. 1 = Yes.	0x1
15	enableVidVlanOps	Shall the VLAN Table VID operation be carried out? 0 = No. 1 = Yes.	0x1
16	checkIngressMspt	Shall the ingress multiple spanning tree check be carried out? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
17	checkEgressMspt	Shall the egress multiple spanning tree check be carried out? 0 = No. 1 = Yes.	0x1
18	checksumCheck	Shall the IPv4 header checksum check be done? 0 = No. 1 = Yes.	0x1
19	checkL2ActionTable	Shall the L2 Action Table be checked? 0 = No. 1 = Yes.	0x1
20	allowPortMove	Shall the packet be allowed to do a L2 Table port move? 0 = No. 1 = Yes.	0x1
21	checkIngressMmp	Shall the ingress meter-marker-policer be updated? 0 = No. 1 = Yes.	0x1
22	doVrfStat	Shall VRF statistics be updated? 0 = No. 1 = Yes.	0x1
23	doNhHitUpdate	Shall this packet do next hop hit statistics updates? 0 = No. 1 = Yes.	0x1
24	routerVops	If the packet is routed then shall the VLAN updates come from the Next Hop Packet Modifications table? 0 = No. 1 = Yes.	0x1
25	egressAclEnabled	Shall the egress ACL lookup be done? 0 = No. 1 = Yes.	0x1
26	allowIngressNat	Shall the ingress NAT operation be allowed? 0 = No. 1 = Yes.	0x1
27	allowEgressNat	Shall the egress NAT operation be allowed? 0 = No. 1 = Yes.	0x1
28	natActionTable	Shall the NAT action table be done? 0 = No. 1 = Yes.	0x1
29	allowMbsc	Shall MBSC operation be allowed? 0 = No. 1 = Yes.	0x1
30	checkInputMirror	Shall the input mirror operation be done on this packet? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
31	checkEgressQueueOn	Check if the egress queue is turned on? 0 = No. 1 = Yes.	0x1
32	updateStatPortMib	Update the hyperref[reg:Statistics: IPP Ingress Port Receive]IPP Ingress Port Receive statistics be updated? 0 = No. 1 = Yes.	0x1
33	usePmFromCryAfterEncrypt	Once a packet has been encrypted shall the processing of the packet use the portmask from IPP before it was sent to Crypto Engine? 0 = No. 1 = Yes.	0x1
34	useQueueFromCryAfterEncrypt	Once a packet has been encrypted shall the processing of the packet use the queue from IPP before it was sent to Crypto Engine? 0 = No. 1 = Yes.	0x1
35	cryptoInputMirroring	When a packet goes to Crypto Engine shall input mirroring be turned off? 0 = No. 1 = Yes.	0x1
36	enableIngressEgressPortFilter	Shall the ingress-egress port filter operation be done? 0 = No. 1 = Yes.	0x1

38.11.121 Ingress Function Control Packet From CPU Port

This register controls which functions a packet shall execute in the ingress packet processing pipeline when a packet comes from the CPU port.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	1122467

Field Description

Bits	Field Name	Description	Default Value
0	doL2L3Lookup	Shall this packet do L2 and L3 Lookups? 0 = No. 1 = Yes.	0x1
1	noLearning	Shall this packet not be learned. Default is to learn the packet (=0)? 0 = No. 1 = Yes.	0x0

Bits	Field Name	Description	Default Value
2	drop	Force the packet to be dropped, updates the **Ingress Functional Control Drops** counter. 0 = No. 1 = Yes.	0x0
3	bypassSendToCpu	In the L2,L3,L4 packet decoding there are a number of registers which can send packets to the CPU, this bit allows these send-to-cpu to be ignored / bypassed. Shall the packet decoders send-to-cpu options be bypassed? 0 = No. 1 = Yes.	0x0
4	allowRouting	Shall the packet be allowed to be routed? 0 = No. 1 = Yes.	0x1
5	enableReservedDmac	Shall the reserved DMAC range check be performed? 0 = No. 1 = Yes.	0x1
6	enableReservedSmac	Shall the reserved SMAC range check be performed? 0 = No. 1 = Yes.	0x1
7	enableSrcPortVlanOps	Shall the source port vlan operation be carried out? 0 = No. 1 = Yes.	0x1
8	enableTunnelExit	Shall this packet do the tunnel exit lookup? 0 = No. 1 = Yes.	0x1
9	allowSmon	Shall this packet be allowed to update the SMON counters? 0 = No. 1 = Yes.	0x1
10	enableIngressPortFilter	Shall the packet be subjected to the ingress port filter? 0 = No. 1 = Yes.	0x1
11	ingressAclEnabled	Shall the ingress ACL operation be done? 0 = No. 1 = Yes.	0x1
12	checkIngressSpt	Shall the ingress spanning tree operation be done? 0 = No. 1 = Yes.	0x1
13	checkEgressSpt	Shall the egress spanning tree operation be done? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
14	allowVlanPortMembershipDrop	Shall the VLAN table drop packets due to packets not being packets not being part of VLAN PortMembership, this affects both the source port and egress port(s) being checked on the vlan-port-membership mask. 0 = No. 1 = Yes.	0x1
15	enableVidVlanOps	Shall the VLAN Table VID operation be carried out? 0 = No. 1 = Yes.	0x1
16	checkIngressMspt	Shall the ingress multiple spanning tree check be carried out? 0 = No. 1 = Yes.	0x1
17	checkEgressMspt	Shall the egress multiple spanning tree check be carried out? 0 = No. 1 = Yes.	0x1
18	checksumCheck	Shall the IPv4 header checksum check be done? 0 = No. 1 = Yes.	0x1
19	checkL2ActionTable	Shall the L2 Action Table be checked? 0 = No. 1 = Yes.	0x1
20	allowPortMove	Shall the packet be allowed to do a L2 Table port move? 0 = No. 1 = Yes.	0x1
21	checkIngressMmp	Shall the ingress meter-marker-policer be updated? 0 = No. 1 = Yes.	0x1
22	doVrfStat	Shall VRF statistics be updated? 0 = No. 1 = Yes.	0x1
23	doNhHitUpdate	Shall this packet do next hop hit statistics updates? 0 = No. 1 = Yes.	0x1
24	routerVops	If the packet is routed then shall the VLAN updates come from the Next Hop Packet Modifications table? 0 = No. 1 = Yes.	0x1
25	egressAclEnabled	Shall the egress ACL lookup be done? 0 = No. 1 = Yes.	0x1
26	allowIngressNat	Shall the ingress NAT operation be allowed? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
27	allowEgressNat	Shall the egress NAT operation be allowed? 0 = No. 1 = Yes.	0x1
28	natActionTable	Shall the NAT action table be done? 0 = No. 1 = Yes.	0x1
29	allowMbsc	Shall MBSC operation be allowed? 0 = No. 1 = Yes.	0x1
30	checkInputMirror	Shall the input mirror operation be done on this packet? 0 = No. 1 = Yes.	0x1
31	checkEgressQueueOn	Check if the egress queue is turned on? 0 = No. 1 = Yes.	0x1
32	updateStatPortMib	Update the hyperref[reg:Statistics: IPP Ingress Port Receive]IPP Ingress Port Receive statistics be updated? 0 = No. 1 = Yes.	0x1
33	usePmFromCryAfterEncrypt	Once a packet has been encrypted shall the processing of the packet use the portmask from IPP before it was sent to Crypto Engine? 0 = No. 1 = Yes.	0x1
34	useQueueFromCryAfterEncrypt	Once a packet has been encrypted shall the processing of the packet use the queue from IPP before it was sent to Crypto Engine? 0 = No. 1 = Yes.	0x1
35	cryptoInputMirroring	When a packet goes to Crypto Engine shall input mirroring be turned off? 0 = No. 1 = Yes.	0x1
36	enableIngressEgressPortFilter	Shall the ingress-egress port filter operation be done? 0 = No. 1 = Yes.	0x1

38.11.122 Ingress Function Control Packet From CPU Tag

This register controls which functions a packet shall execute in the ingress packet processing pipeline when a packet comes from the CPU port and had a from-CPU tag.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	1122477

Field Description

Bits	Field Name	Description	Default Value
0	doL2L3Lookup	Shall this packet do L2 and L3 Lookups? 0 = No. 1 = Yes.	0x1
1	noLearning	Shall this packet not be learned. Default is to learn the packet (=0)? 0 = No. 1 = Yes.	0x0
2	drop	Force the packet to be dropped, updates the **Ingress Functional Control Drops** counter. 0 = No. 1 = Yes.	0x0
3	bypassSendToCpu	In the L2,L3,L4 packet decoding there are a number of registers which can send packets to the CPU, this bit allows these send-to-cpu to be ignored / bypassed. Shall the packet decoders send-to-cpu options be bypassed? 0 = No. 1 = Yes.	0x0
4	allowRouting	Shall the packet be allowed to be routed? 0 = No. 1 = Yes.	0x1
5	enableReservedDmac	Shall the reserved DMAC range check be performed? 0 = No. 1 = Yes.	0x1
6	enableReservedSmac	Shall the reserved SMAC range check be performed? 0 = No. 1 = Yes.	0x1
7	enableSrcPortVlanOps	Shall the source port vlan operation be carried out? 0 = No. 1 = Yes.	0x1
8	enableTunnelExit	Shall this packet do the tunnel exit lookup? 0 = No. 1 = Yes.	0x1
9	allowSmon	Shall this packet be allowed to update the SMON counters? 0 = No. 1 = Yes.	0x1
10	enableIngressPortFilter	Shall the packet be subjected to the ingress port filter? 0 = No. 1 = Yes.	0x1
11	ingressAclEnabled	Shall the ingress ACL operation be done? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
12	checkIngressSpt	Shall the ingress spanning tree operation be done? 0 = No. 1 = Yes.	0x1
13	checkEgressSpt	Shall the egress spanning tree operation be done? 0 = No. 1 = Yes.	0x1
14	allowVlanPortMembershipDrop	Shall the VLAN table drop packets due to packets not being packets not being part of VLAN PortMembership, this affects both the source port and egress port(s) being checked on the vlan-port-membership mask. 0 = No. 1 = Yes.	0x1
15	enableVidVlanOps	Shall the VLAN Table VID operation be carried out? 0 = No. 1 = Yes.	0x1
16	checkIngressMspt	Shall the ingress multiple spanning tree check be carried out? 0 = No. 1 = Yes.	0x1
17	checkEgressMspt	Shall the egress multiple spanning tree check be carried out? 0 = No. 1 = Yes.	0x1
18	checksumCheck	Shall the IPv4 header checksum check be done? 0 = No. 1 = Yes.	0x1
19	checkL2ActionTable	Shall the L2 Action Table be checked? 0 = No. 1 = Yes.	0x1
20	allowPortMove	Shall the packet be allowed to do a L2 Table port move? 0 = No. 1 = Yes.	0x1
21	checkIngressMmp	Shall the ingress meter-marker-policer be updated? 0 = No. 1 = Yes.	0x1
22	doVrfStat	Shall VRF statistics be updated? 0 = No. 1 = Yes.	0x1
23	doNhHitUpdate	Shall this packet do next hop hit statistics updates? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
24	routerVops	If the packet is routed then shall the VLAN updates come from the Next Hop Packet Modifications table? 0 = No. 1 = Yes.	0x1
25	egressAclEnabled	Shall the egress ACL lookup be done? 0 = No. 1 = Yes.	0x1
26	allowIngressNat	Shall the ingress NAT operation be allowed? 0 = No. 1 = Yes.	0x1
27	allowEgressNat	Shall the egress NAT operation be allowed? 0 = No. 1 = Yes.	0x1
28	natActionTable	Shall the NAT action table be done? 0 = No. 1 = Yes.	0x1
29	allowMbsc	Shall MBSC operation be allowed? 0 = No. 1 = Yes.	0x1
30	checkInputMirror	Shall the input mirror operation be done on this packet? 0 = No. 1 = Yes.	0x1
31	checkEgressQueueOn	Check if the egress queue is turned on? 0 = No. 1 = Yes.	0x1
32	updateStatPortMib	Update the hyperref[reg:Statistics: IPP Ingress Port Receive]IPP Ingress Port Receive statistics be updated? 0 = No. 1 = Yes.	0x1
33	usePmFromCryAfterEncrypt	Once a packet has been encrypted shall the processing of the packet use the portmask from IPP before it was sent to Crypto Engine? 0 = No. 1 = Yes.	0x1
34	useQueueFromCryAfterEncrypt	Once a packet has been encrypted shall the processing of the packet use the queue from IPP before it was sent to Crypto Engine? 0 = No. 1 = Yes.	0x1
35	cryptoInputMirroring	When a packet goes to Crypto Engine shall input mirroring be turned off? 0 = No. 1 = Yes.	0x1
36	enableIngressEgressPortFilter	Shall the ingress-egress port filter operation be done? 0 = No. 1 = Yes.	0x1

38.11.123 Ingress Function Control Packet From CPU Tag Do Not Modify

This register controls which functions a packet shall execute in the ingress packet processing pipeline when a packet comes from the CPU port and had a from-CPU tag where the do not change bit set to one.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	1122479

Field Description

Bits	Field Name	Description	Default Value
0	doL2L3Lookup	Shall this packet do L2 and L3 Lookups? 0 = No. 1 = Yes.	0x1
1	noLearning	Shall this packet not be learned. Default is to learn the packet (=0)? 0 = No. 1 = Yes.	0x0
2	drop	Force the packet to be dropped, updates the **Ingress Functional Control Drops** counter. 0 = No. 1 = Yes.	0x0
3	bypassSendToCpu	In the L2,L3,L4 packet decoding there are a number of registers which can send packets to the CPU, this bit allows these send-to-cpu to be ignored / bypassed. Shall the packet decoders send-to-cpu options be bypassed? 0 = No. 1 = Yes.	0x0
4	allowRouting	Shall the packet be allowed to be routed? 0 = No. 1 = Yes.	0x0
5	enableReservedDmac	Shall the reserved DMAC range check be performed? 0 = No. 1 = Yes.	0x1
6	enableReservedSmac	Shall the reserved SMAC range check be performed? 0 = No. 1 = Yes.	0x1
7	enableSrcPortVlanOps	Shall the source port vlan operation be carried out? 0 = No. 1 = Yes.	0x0
8	enableTunnelExit	Shall this packet do the tunnel exit lookup? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
9	allowSmon	Shall this packet be allowed to update the SMON counters? 0 = No. 1 = Yes.	0x1
10	enableIngressPortFilter	Shall the packet be subjected to the ingress port filter? 0 = No. 1 = Yes.	0x1
11	ingressAclEnabled	Shall the ingress ACL operation be done? 0 = No. 1 = Yes.	0x0
12	checkIngressSpt	Shall the ingress spanning tree operation be done? 0 = No. 1 = Yes.	0x1
13	checkEgressSpt	Shall the egress spanning tree operation be done? 0 = No. 1 = Yes.	0x1
14	allowVlanPortMembershipDrop	Shall the VLAN table drop packets due to packets not being packets not being part of VLAN PortMembership, this affects both the source port and egress port(s) being checked on the vlan-port-membership mask. 0 = No. 1 = Yes.	0x1
15	enableVidVlanOps	Shall the VLAN Table VID operation be carried out? 0 = No. 1 = Yes.	0x0
16	checkIngressMspt	Shall the ingress multiple spanning tree check be carried out? 0 = No. 1 = Yes.	0x1
17	checkEgressMspt	Shall the egress multiple spanning tree check be carried out? 0 = No. 1 = Yes.	0x1
18	checksumCheck	Shall the IPv4 header checksum check be done? 0 = No. 1 = Yes.	0x1
19	checkL2ActionTable	Shall the L2 Action Table be checked? 0 = No. 1 = Yes.	0x1
20	allowPortMove	Shall the packet be allowed to do a L2 Table port move? 0 = No. 1 = Yes.	0x1
21	checkIngressMmp	Shall the ingress meter-marker-policer be updated? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
22	doVrfStat	Shall VRF statistics be updated? 0 = No. 1 = Yes.	0x1
23	doNhHitUpdate	Shall this packet do next hop hit statistics updates? 0 = No. 1 = Yes.	0x1
24	routerVops	If the packet is routed then shall the VLAN updates come from the Next Hop Packet Modifications table? 0 = No. 1 = Yes.	0x1
25	egressAclEnabled	Shall the egress ACL lookup be done? 0 = No. 1 = Yes.	0x0
26	allowIngressNat	Shall the ingress NAT operation be allowed? 0 = No. 1 = Yes.	0x1
27	allowEgressNat	Shall the egress NAT operation be allowed? 0 = No. 1 = Yes.	0x1
28	natActionTable	Shall the NAT action table be done? 0 = No. 1 = Yes.	0x1
29	allowMbsc	Shall MBSC operation be allowed? 0 = No. 1 = Yes.	0x1
30	checkInputMirror	Shall the input mirror operation be done on this packet? 0 = No. 1 = Yes.	0x1
31	checkEgressQueueOn	Check if the egress queue is turned on? 0 = No. 1 = Yes.	0x1
32	updateStatPortMib	Update the hyperref[reg:Statistics: IPP Ingress Port Receive]IPP Ingress Port Receive statistics be updated? 0 = No. 1 = Yes.	0x1
33	usePmFromCryAfterEncrypt	Once a packet has been encrypted shall the processing of the packet use the portmask from IPP before it was sent to Crypto Engine? 0 = No. 1 = Yes.	0x1
34	useQueueFromCryAfterEncrypt	Once a packet has been encrypted shall the processing of the packet use the queue from IPP before it was sent to Crypto Engine? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
35	cryptoInputMirroring	When a packet goes to Crypto Engine shall input mirroring be turned off? 0 = No. 1 = Yes.	0x1
36	enableIngressEgressPortFilter	Shall the ingress-egress port filter operation be done? 0 = No. 1 = Yes.	0x1

38.11.124 Ingress Function Control Packet From Crypto Engine Decrypted

This register controls which functions a packet shall execute in the ingress packet processing pipeline when a packet comes from the crypto engine and has been decrypted.

Number of Entries : 1
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 1122469

Field Description

Bits	Field Name	Description	Default Value
0	doL2L3Lookup	Shall this packet do L2 and L3 Lookups? 0 = No. 1 = Yes.	0x1
1	noLearning	Shall this packet not be learned. Default is to learn the packet (=0)? 0 = No. 1 = Yes.	0x0
2	drop	Force the packet to be dropped, updates the **Ingress Functional Control Drops** counter. 0 = No. 1 = Yes.	0x0
3	bypassSendToCpu	In the L2,L3,L4 packet decoding there are a number of registers which can send packets to the CPU, this bit allows these send-to-cpu to be ignored / bypassed. Shall the packet decoders send-to-cpu options be bypassed? 0 = No. 1 = Yes.	0x0
4	allowRouting	Shall the packet be allowed to be routed? 0 = No. 1 = Yes.	0x1
5	enableReservedDmac	Shall the reserved DMAC range check be performed? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
6	enableReservedSmac	Shall the reserved SMAC range check be performed? 0 = No. 1 = Yes.	0x1
7	enableSrcPortVlanOps	Shall the source port vlan operation be carried out? 0 = No. 1 = Yes.	0x1
8	enableTunnelExit	Shall this packet do the tunnel exit lookup? 0 = No. 1 = Yes.	0x1
9	allowSmon	Shall this packet be allowed to update the SMON counters? 0 = No. 1 = Yes.	0x1
10	enableIngressPortFilter	Shall the packet be subjected to the ingress port filter? 0 = No. 1 = Yes.	0x1
11	ingressAclEnabled	Shall the ingress ACL operation be done? 0 = No. 1 = Yes.	0x1
12	checkIngressSpt	Shall the ingress spanning tree operation be done? 0 = No. 1 = Yes.	0x1
13	checkEgressSpt	Shall the egress spanning tree operation be done? 0 = No. 1 = Yes.	0x1
14	allowVlanPortMembershipDrop	Shall the VLAN table drop packets due to packets not being packets not being part of VLAN PortMembership, this affects both the source port and egress port(s) being checked on the vlan-port-membership mask. 0 = No. 1 = Yes.	0x1
15	enableVidVlanOps	Shall the VLAN Table VID operation be carried out? 0 = No. 1 = Yes.	0x1
16	checkIngressMspt	Shall the ingress multiple spanning tree check be carried out? 0 = No. 1 = Yes.	0x1
17	checkEgressMspt	Shall the egress multiple spanning tree check be carried out? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
18	checksumCheck	Shall the IPv4 header checksum check be done? 0 = No. 1 = Yes.	0x1
19	checkL2ActionTable	Shall the L2 Action Table be checked? 0 = No. 1 = Yes.	0x1
20	allowPortMove	Shall the packet be allowed to do a L2 Table port move? 0 = No. 1 = Yes.	0x1
21	checkIngressMmp	Shall the ingress meter-marker-policer be updated? 0 = No. 1 = Yes.	0x1
22	doVrfStat	Shall VRF statistics be updated? 0 = No. 1 = Yes.	0x1
23	doNhHitUpdate	Shall this packet do next hop hit statistics updates? 0 = No. 1 = Yes.	0x1
24	routerVops	If the packet is routed then shall the VLAN updates come from the Next Hop Packet Modifications table? 0 = No. 1 = Yes.	0x1
25	egressAclEnabled	Shall the egress ACL lookup be done? 0 = No. 1 = Yes.	0x1
26	allowIngressNat	Shall the ingress NAT operation be allowed? 0 = No. 1 = Yes.	0x1
27	allowEgressNat	Shall the egress NAT operation be allowed? 0 = No. 1 = Yes.	0x1
28	natActionTable	Shall the NAT action table be done? 0 = No. 1 = Yes.	0x1
29	allowMbsc	Shall MBSC operation be allowed? 0 = No. 1 = Yes.	0x1
30	checkInputMirror	Shall the input mirror operation be done on this packet? 0 = No. 1 = Yes.	0x0
31	checkEgressQueueOn	Check if the egress queue is turned on? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
32	updateStatPortMib	Update the hyperref[reg:Statistics: IPP Ingress Port Receive]IPP Ingress Port Receive statistics be updated? 0 = No. 1 = Yes.	0x1
33	usePmFromCryAfterEncrypt	Once a packet has been encrypted shall the processing of the packet use the portmask from IPP before it was sent to Crypto Engine? 0 = No. 1 = Yes.	0x1
34	useQueueFromCryAfterEncrypt	Once a packet has been encrypted shall the processing of the packet use the queue from IPP before it was sent to Crypto Engine? 0 = No. 1 = Yes.	0x1
35	cryptoInputMirroring	When a packet goes to Crypto Engine shall input mirroring be turned off? 0 = No. 1 = Yes.	0x0
36	enableIngressEgressPortFilter	Shall the ingress-egress port filter operation be done? 0 = No. 1 = Yes.	0x1

38.11.125 Ingress Function Control Packet From Crypto Engine Encrypted

This register controls which functions a packet shall execute in the ingress packet processing pipeline when a packet comes from the crypto engine and has been encrypted.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	1122471

Field Description

Bits	Field Name	Description	Default Value
0	doL2L3Lookup	Shall this packet do L2 and L3 Lookups? 0 = No. 1 = Yes.	0x1
1	noLearning	Shall this packet not be learned. Default is to learn the packet (=0)? 0 = No. 1 = Yes.	0x0
2	drop	Force the packet to be dropped, updates the **Ingress Functional Control Drops** counter. 0 = No. 1 = Yes.	0x0

Bits	Field Name	Description	Default Value
3	bypassSendToCpu	In the L2,L3,L4 packet decoding there are a number of registers which can send packets to the CPU, this bit allows these send-to-cpu to be ignored / bypassed. Shall the packet decoders send-to-cpu options be bypassed? 0 = No. 1 = Yes.	0x0
4	allowRouting	Shall the packet be allowed to be routed? 0 = No. 1 = Yes.	0x0
5	enableReservedDmac	Shall the reserved DMAC range check be performed? 0 = No. 1 = Yes.	0x1
6	enableReservedSmac	Shall the reserved SMAC range check be performed? 0 = No. 1 = Yes.	0x1
7	enableSrcPortVlanOps	Shall the source port vlan operation be carried out? 0 = No. 1 = Yes.	0x0
8	enableTunnelExit	Shall this packet do the tunnel exit lookup? 0 = No. 1 = Yes.	0x1
9	allowSmon	Shall this packet be allowed to update the SMON counters? 0 = No. 1 = Yes.	0x1
10	enableIngressPortFilter	Shall the packet be subjected to the ingress port filter? 0 = No. 1 = Yes.	0x1
11	ingressAclEnabled	Shall the ingress ACL operation be done? 0 = No. 1 = Yes.	0x1
12	checkIngressSpt	Shall the ingress spanning tree operation be done? 0 = No. 1 = Yes.	0x1
13	checkEgressSpt	Shall the egress spanning tree operation be done? 0 = No. 1 = Yes.	0x1
14	allowVlanPortMembershipDrop	Shall the VLAN table drop packets due to packets not being packets not being part of VLAN PortMembership, this affects both the source port and egress port(s) being checked on the vlan-port-membership mask. 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
15	enableVidVlanOps	Shall the VLAN Table VID operation be carried out? 0 = No. 1 = Yes.	0x0
16	checkIngressMspt	Shall the ingress multiple spanning tree check be carried out? 0 = No. 1 = Yes.	0x1
17	checkEgressMspt	Shall the egress multiple spanning tree check be carried out? 0 = No. 1 = Yes.	0x1
18	checksumCheck	Shall the IPv4 header checksum check be done? 0 = No. 1 = Yes.	0x1
19	checkL2ActionTable	Shall the L2 Action Table be checked? 0 = No. 1 = Yes.	0x1
20	allowPortMove	Shall the packet be allowed to do a L2 Table port move? 0 = No. 1 = Yes.	0x1
21	checkIngressMmp	Shall the ingress meter-marker-policer be updated? 0 = No. 1 = Yes.	0x1
22	doVrfStat	Shall VRF statistics be updated? 0 = No. 1 = Yes.	0x1
23	doNhHitUpdate	Shall this packet do next hop hit statistics updates? 0 = No. 1 = Yes.	0x1
24	routerVops	If the packet is routed then shall the VLAN updates come from the Next Hop Packet Modifications table? 0 = No. 1 = Yes.	0x1
25	egressAclEnabled	Shall the egress ACL lookup be done? 0 = No. 1 = Yes.	0x1
26	allowIngressNat	Shall the ingress NAT operation be allowed? 0 = No. 1 = Yes.	0x1
27	allowEgressNat	Shall the egress NAT operation be allowed? 0 = No. 1 = Yes.	0x1
28	natActionTable	Shall the NAT action table be done? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
29	allowMbsc	Shall MBSC operation be allowed? 0 = No. 1 = Yes.	0x1
30	checkInputMirror	Shall the input mirror operation be done on this packet? 0 = No. 1 = Yes.	0x0
31	checkEgressQueueOn	Check if the egress queue is turned on? 0 = No. 1 = Yes.	0x1
32	updateStatPortMib	Update the hyperref[reg:Statistics: IPP Ingress Port Receive]IPP Ingress Port Receive statistics be updated? 0 = No. 1 = Yes.	0x1
33	usePmFromCryAfterEncrypt	Once a packet has been encrypted shall the processing of the packet use the portmask from IPP before it was sent to Crypto Engine? 0 = No. 1 = Yes.	0x1
34	useQueueFromCryAfterEncrypt	Once a packet has been encrypted shall the processing of the packet use the queue from IPP before it was sent to Crypto Engine? 0 = No. 1 = Yes.	0x1
35	cryptoInputMirroring	When a packet goes to Crypto Engine shall input mirroring be turned off? 0 = No. 1 = Yes.	0x0
36	enableIngressEgressPortFilter	Shall the ingress-egress port filter operation be done? 0 = No. 1 = Yes.	0x1

38.11.126 Ingress Function Control Packet To Crypto Engine

This register controls which functions a packet shall execute in the ingress packet processing pipeline when a packet shall be sent to the crypto engine.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	1122505

Field Description

Bits	Field Name	Description	Default Value
0	doL2L3Lookup	Shall this packet do L2 and L3 Lookups? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
1	noLearning	Shall this packet not be learned. Default is to learn the packet (=0)? 0 = No. 1 = Yes.	0x0
2	drop	Force the packet to be dropped, updates the **Ingress Functional Control Drops** counter. 0 = No. 1 = Yes.	0x0
3	bypassSendToCpu	In the L2,L3,L4 packet decoding there are a number of registers which can send packets to the CPU, this bit allows these send-to-cpu to be ignored / bypassed. Shall the packet decoders send-to-cpu options be bypassed? 0 = No. 1 = Yes.	0x0
4	allowRouting	Shall the packet be allowed to be routed? 0 = No. 1 = Yes.	0x1
5	enableReservedDmac	Shall the reserved DMAC range check be performed? 0 = No. 1 = Yes.	0x1
6	enableReservedSmac	Shall the reserved SMAC range check be performed? 0 = No. 1 = Yes.	0x1
7	enableSrcPortVlanOps	Shall the source port vlan operation be carried out? 0 = No. 1 = Yes.	0x1
8	enableTunnelExit	Shall this packet do the tunnel exit lookup? 0 = No. 1 = Yes.	0x1
9	allowSmon	Shall this packet be allowed to update the SMON counters? 0 = No. 1 = Yes.	0x1
10	enableIngressPortFilter	Shall the packet be subjected to the ingress port filter? 0 = No. 1 = Yes.	0x1
11	ingressAclEnabled	Shall the ingress ACL operation be done? 0 = No. 1 = Yes.	0x1
12	checkIngressSpt	Shall the ingress spanning tree operation be done? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
13	checkEgressSpt	Shall the egress spanning tree operation be done? 0 = No. 1 = Yes.	0x1
14	allowVlanPortMembershipDrop	Shall the VLAN table drop packets due to packets not being packets not being part of VLAN PortMembership, this affects both the source port and egress port(s) being checked on the vlan-port-membership mask. 0 = No. 1 = Yes.	0x1
15	enableVidVlanOps	Shall the VLAN Table VID operation be carried out? 0 = No. 1 = Yes.	0x1
16	checkIngressMspt	Shall the ingress multiple spanning tree check be carried out? 0 = No. 1 = Yes.	0x1
17	checkEgressMspt	Shall the egress multiple spanning tree check be carried out? 0 = No. 1 = Yes.	0x1
18	checksumCheck	Shall the IPv4 header checksum check be done? 0 = No. 1 = Yes.	0x1
19	checkL2ActionTable	Shall the L2 Action Table be checked? 0 = No. 1 = Yes.	0x1
20	allowPortMove	Shall the packet be allowed to do a L2 Table port move? 0 = No. 1 = Yes.	0x1
21	checkIngressMmp	Shall the ingress meter-marker-policer be updated? 0 = No. 1 = Yes.	0x1
22	doVrfStat	Shall VRF statistics be updated? 0 = No. 1 = Yes.	0x1
23	doNhHitUpdate	Shall this packet do next hop hit statistics updates? 0 = No. 1 = Yes.	0x1
24	routerVops	If the packet is routed then shall the VLAN updates come from the Next Hop Packet Modifications table? 0 = No. 1 = Yes.	0x1
25	egressAclEnabled	Shall the egress ACL lookup be done? 0 = No. 1 = Yes.	0x1

Bits	Field Name	Description	Default Value
26	allowIngressNat	Shall the ingress NAT operation be allowed? 0 = No. 1 = Yes.	0x1
27	allowEgressNat	Shall the egress NAT operation be allowed? 0 = No. 1 = Yes.	0x1
28	natActionTable	Shall the NAT action table be done? 0 = No. 1 = Yes.	0x1
29	allowMbsc	Shall MBSC operation be allowed? 0 = No. 1 = Yes.	0x1
30	checkInputMirror	Shall the input mirror operation be done on this packet? 0 = No. 1 = Yes.	0x1
31	checkEgressQueueOn	Check if the egress queue is turned on? 0 = No. 1 = Yes.	0x1
32	updateStatPortMib	Update the hyperref[reg:Statistics: IPP Ingress Port Receive]IPP Ingress Port Receive statistics be updated? 0 = No. 1 = Yes.	0x1
33	usePmFromCryAfterEncrypt	Once a packet has been encrypted shall the processing of the packet use the portmask from IPP before it was sent to Crypto Engine? 0 = No. 1 = Yes.	0x1
34	useQueueFromCryAfterEncrypt	Once a packet has been encrypted shall the processing of the packet use the queue from IPP before it was sent to Crypto Engine? 0 = No. 1 = Yes.	0x1
35	cryptoInputMirroring	When a packet goes to Crypto Engine shall input mirroring be turned off? 0 = No. 1 = Yes.	0x1
36	enableIngressEgressPortFilter	Shall the ingress-egress port filter operation be done? 0 = No. 1 = Yes.	0x1

38.11.127 Ingress Function Pointer Source Port

This register controls which basic function a port shall be using by pointing to the entry in the **Ingress Function Control** which a source port shall use.

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Source Port
Address Space : 1118624 to 1118634

Field Description

Bits	Field Name	Description	Default Value
0	ptr	Which functional setting shall be used for this source port.	0x0

38.11.128 Ingress MMP Drop Mask

This register provides an option to let ingress MMP not drop packets on certain ports after metering.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121247

Field Description

Bits	Field Name	Description	Default Value
10:0	dropMask	Each bit in this mask refers to if ingress MMP drop is allowed on the corresponding egress port.	0x7ff

38.11.129 Ingress Multiple Spanning Tree State

Table of ingress Multiple Spanning Tree Protocol Instances. For routed packets the pointer used to address this table is from the **msptPtr** field in the **Next Hop Packet Modifications** table. For switched packets is is from the **msptPtr** field in the **VLAN Table**. Each entry contains the ingress spanning tree states for all ports in this MSTI.

Number of Entries : 16
Type of Operation : Read/Write
Addressing : msptPtr from VLAN Table or Next Hop Packet Modifications Table
Address Space : 347580 to 347595

Field Description

Bits	Field Name	Description	Default Value
21:0	portSptState	The ingress spanning tree state for this MSTI. Bit[1:0] is the state for port #0, bit[3:2] is the state for port #1, etc. 0 = Forwarding 1 = Discarding 2 = Learning	0x0

38.11.130 Ingress Port Packet Type Filter

This configures which packet types that are to be dropped or allowed on each source port. Each entry corresponds to one ingress port. Packets dropped due to the filter are counted in **Ingress Packet Filtering Drop**.

Number of Entries :	11
Type of Operation :	Read/Write
Addressing :	Ingress port
Address Space :	1120662 to 1120672

Field Description

Bits	Field Name	Description	Default Value
0	dropCtaggedVlans	Drop or allow customer VLAN tagged packet on this ingress port. Will only drop packets that has exactly one VLAN tag. Must set moreThanOneVlans when this is used. 0 = Allow C-VLANs. 1 = Drop C-VLANs.	0x0
1	dropStaggedVlans	Drop or allow service VLANs tagged packets on this ingress port. Will only drop packets that has exactly one VLAN tag. Must set moreThanOneVlans when this is used. 0 = Allow S-VLANs. 1 = Drop S-VLANs.	0x0
2	moreThanOneVlans	When filtering with dropCtaggedVlans or dropStaggedVlans then this field must be set to 1.	0x0
3	dropUntaggedVlans	Drop or Allow packets that are VLAN untagged on this ingress port. 0 = Allow untagged packets. 1 = Drop untagged packets.	0x0
4	dropSingleTaggedVlans	Drop or Allow packets that are VLAN untagged on this ingress port. 0 = Allow untagged packets. 1 = Drop untagged packets.	0x0
5	dropIPv4Packets	Drop or allow IPv4 packets on this ingress port. 0 = Allow IPv4 packets. 1 = Drop IPv4 packets.	0x0
6	dropIPv6Packets	Drop or allow IPv6 packets on this ingress port. 0 = Allow IPv6 packets. 1 = Drop IPv6 packets.	0x0
7	dropMPLSPackets	Drop or allow MPLS packets on this ingress port. 0 = Allow MPLS packets. 1 = Drop MPLS packets.	0x0
8	dropIPv4MulticastPackets	Drop or allow IPv4 multicast packets on this ingress port. 0 = Allow IPv4 MC packets. 1 = Drop IPv4 MC packets.	0x0
9	dropIPv6MulticastPackets	Drop or allow IPv6 multicast packets on this ingress port. 0 = Allow IPv6 MC packets. 1 = Drop IPv6 MC packets.	0x0

Bits	Field Name	Description	Default Value
10	dropL2BroadcastFrames	Drop or allow L2 broadcast packets on this ingress port. 0 = Drop L2 broadcast packets. 1 = Allow L2 broadcast packets.	0x0
11	dropL2MulticastFrames	Drop or allow L2 multicast packets on this ingress port. Observe that this L2 multicast bit takes the register **L2 Multicast Handling** into account to determine if this packet is a L2 multicast packet or not. 0 = Allow L2 multicast packets 1 = Drop L2 multicast packets.	0x0
12	dropDualTaggedVlans	Drop or allow packets which has more than one VLAN tag on this ingress port. 0 = Allow packets which has dual tags. 1 = Drop packets which has dual tags.	0x0
13	dropCStaggedVlans	Drop or allow packets which has a C-VLAN followed by a S-VLAN tagged on this ingress port. 0 = Allow packets which has a C-VLAN tag followed by a S-VLAN tag. 1 = Drop packets which has a C-VLAN tag followed by a S-VLAN tag.	0x0
14	dropSCtaggedVlans	Drop or allow packets which has a S-VLAN followed by a C-VLAN tagged on this ingress port. 0 = Allow packets which has a S-VLAN followed by a C-VLAN tag. 1 = Drop packets which has a S-VLAN tag followed by a C-VLAN tag.	0x0
15	dropCCtaggedVlans	Drop or allow packets which has a C-VLAN followed by a C-VLAN tagged on this ingress port. 0 = Allow packets which has a C-VLANs tag followed by a C-VLAN tag. 1 = Drop packets which has a C-VLAN tag followed by a C-VLAN tag.	0x0
16	dropSStaggedVlans	Drop or allow packets which has a S-VLAN followed by a S-VLAN tagged on this source port. 0 = Allow packets which has a S-VLAN tag followed by a S-VLAN tag. 1 = Drop packets which has a S-VLAN tag followed by a S-VLAN tag.	0x0

38.11.131 Ingress Router Table

The ingress router table or the Virtual Router Function (VRF), controls which packets are allowed to get access to this router. If a packet is dropped due to the settings of **Ingress Router Table** accept fields then the **Invalid Routing Protocol Drop** will be incremented. Updates for the **Next Hop Hit Status** is also controlled in this table.

Number of Entries : 4
Type of Operation : Read/Write
Addressing : **vrf**
Address Space : 347596 to 347599

Field Description

Bits	Field Name	Description	Default Value
0	acceptIPv4	Accept IPv4 packets. If disabled and an IPv4 packet reaches the router the packet will be dropped and the **Invalid Routing Protocol Drop** incremented. 0 = Deny 1 = Accept	0x0
1	acceptIPv6	Accept IPv6 packets. If disabled and an IPv6 packet reaches the router the packet will be dropped and the **Invalid Routing Protocol Drop** incremented. 0 = Deny 1 = Accept	0x0
2	acceptMPLS	Accept MPLS packets. If disabled and an MPLS packet reaches the router the packet will be dropped and the **Invalid Routing Protocol Drop** incremented. 0 = Deny 1 = Accept	0x0
10:3	minTTL	Minimum TTL. Packets with a TTL below this value will not be accepted. The packet will be dropped and the **Expired TTL Drop** counter incremented. If the **minTtlToCpu** is set the packet will be sent to CPU instead of being dropped. The TTL check is done for IPv4, IPv6 and MPLS routed packets.	0x0
11	minTtlToCpu	If this is set then packets below minimum TTL will be send to CPU instead of dropped.	0x0
12	ipv4HitUpdates	Enable updates of the **Next Hop Hit Status** for routed IPv4 packets. 0 = Disable 1 = Enable	0x0
13	ipv6HitUpdates	Enable updates of the **Next Hop Hit Status** for routed IPv6 packets. 0 = Disable 1 = Enable	0x0
14	mplsHitUpdates	Enable updates of the **Next Hop Hit Status** for routed MPLS packets. 0 = Disable 1 = Enable	0x0
15	ecmpUseIpDa	Use IP destination address as part of ECMP hash key.	0x1
16	ecmpUseIpSa	Use IP source address as part of ECMP hash key.	0x1
17	ecmpUseIpTos	Use IP TOS/Traffic Class as part of ECMP hash key.	0x0
18	ecmpUseIpProto	Use IP Protocol/Next Header as part of ECMP hash key.	0x1
19	ecmpUseIpL4Sp	Use TCP/UDP source port as part of ECMP hash key.	0x1
20	ecmpUseIpL4Dp	Use TCP/UDP destination port as part of ECMP hash key.	0x1
21	mmpValid	If set, this entry contains a valid MMP pointer. Only valid when packets get routed	0x0
27:22	mmpPtr	Ingress MMP pointer.	0x0
29:28	mmpOrder	Ingress MMP pointer order.	0x0
30	sendToCpuOrDrop	When a check if the packet protocols are allowed on this Ingress Router Table shall the packets be dropped or sent-to-CPU? 0 = Dropped. 1 = Sent-To-CPU	0x0

38.11.132 Ingress VID Ethernet Type Range Assignment Answer

The ingress VID to be assigned when the corresponding range matched.

Number of Entries :	4
Type of Operation :	Read/Write
Addressing :	**Ingress VID Ethernet Type Range Search Data** hit index
Address Space :	1120638 to 1120641

Field Description

Bits	Field Name	Description	Default Value
11:0	ingressVid	Ingress VID.	0x0
13:12	order	Order for this assignment. If the ingress VID can be assigned from other packet field ranges, the one with the highest order wins.	0x0

38.11.133 Ingress VID Ethernet Type Range Search Data

This Ethernet type range can be used to assign the ingress VID. The search starts from entry 0 and returns the first match to lookup in the **Ingress VID Ethernet Type Range Assignment Answer** table.

Number of Entries :	4
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	1122403 to 1122410

Field Description

Bits	Field Name	Description	Default Value
10:0	ports	Ports that this range search is activated on.	0x0
26:11	start	Start of Ethernet type range.	0x0
42:27	end	End of Ethernet type range.	0x0

38.11.134 Ingress VID Inner VID Range Assignment Answer

The ingress VID to be assigned when the corresponding range matched.

Number of Entries :	4
Type of Operation :	Read/Write
Addressing :	**Ingress VID Inner VID Range Search Data** hit index
Address Space :	1120642 to 1120645

Field Description

Bits	Field Name	Description	Default Value
11:0	ingressVid	Ingress VID.	0x0
13:12	order	Order for this assignment. If the ingress VID can be assigned from other packet field ranges, the one with the highest order wins.	0x0

38.11.135 Ingress VID Inner VID Range Search Data

If a packet has an inner VLAN tag, this inner VID range can be used to assign the ingress VID. The search starts from entry 0 and returns the first match to lookup in the **Ingress VID Inner VID Range Assignment Answer** table.

Number of Entries : 4
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Addressing : All entries are read out in parallel
Address Space : 1122411 to 1122418

Field Description

Bits	Field Name	Description	Default Value
10:0	ports	Ports that this range search is activated on.	0x0
11	vtype	Shall this entry match S-Type or C-Type VLAN. 0 = C-Type 1 = S-Type	0x0
23:12	start	Start of VID range.	0x0
35:24	end	End of VID range.	0x0

38.11.136 Ingress VID MAC Range Assignment Answer

The ingress VID to be assigned when the corresponding range matched.

Number of Entries : 4
Type of Operation : Read/Write
Addressing : **Ingress VID MAC Range Search Data** hit index
Address Space : 1120650 to 1120653

Field Description

Bits	Field Name	Description	Default Value
11:0	ingressVid	Ingress VID.	0x0
13:12	order	Order for this assignment. If the ingress VID can be assigned from other packet field ranges, the one with the highest order wins.	0x0

38.11.137 Ingress VID MAC Range Search Data

This MAC address range can be used to assign the ingress VID. The search starts from entry 0 and returns the first match to lookup in the **Ingress VID MAC Range Assignment Answer** table.

Number of Entries :	4
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	1122619 to 1122634

Field Description

Bits	Field Name	Description	Default Value
10:0	ports	Ports that this range search is activated on.	0x0
11	saOrDa	Is this rule for source or destination MAC address. 0 = Source MAC 1 = Destination MAC	0x0
59:12	start	Start of MAC address range.	0x0
107:60	end	End of MAC address range.	0x0

38.11.138 Ingress VID Outer VID Range Assignment Answer

The ingress VID to be assigned when the corresponding range matched.

Number of Entries :	4
Type of Operation :	Read/Write
Addressing :	**Ingress VID Outer VID Range Search Data** hit index
Address Space :	1120646 to 1120649

Field Description

Bits	Field Name	Description	Default Value
11:0	ingressVid	Ingress VID.	0x0
13:12	order	Order for this assignment. If the ingress VID can be assigned from other packet field ranges, the one with the highest order wins.	0x0

38.11.139 Ingress VID Outer VID Range Search Data

If a packet has an outer VLAN tag, this outer VID range can be used to assign the ingress VID. The search starts from entry 0 and returns the first match to lookup in the **Ingress VID Outer VID Range Assignment Answer** table.

Number of Entries :	4
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	1122419 to 1122426

Field Description

Bits	Field Name	Description	Default Value
10:0	ports	Ports that this range search is activated on.	0x0
11	vtype	Shall this entry match S-Type or C-Type VLAN. 0 = C-Type 1 = S-Type	0x0
23:12	start	Start of VID range.	0x0
35:24	end	End of VID range.	0x0

38.11.140 L2 Action Table

The L2 action table can be used to limit what type of traffic shall be able to enter a port depending on which port its coming from and going to. There are three table results which can be taken into consideration, the l2 destination MAC lookup, the l2 source MAC lookup and finally the ingress ACL lookup. The **L2 Action Table Egress Port State** defines the highest bit in the address. This table is looked up for each of the destiantion ports which the packet is going to. If a packet is dropped then it is recorded in the drop counter **L2 Action Table Drop**.

Number of Entries : 128

Type of Operation : Read/Write

Addressing :

Address Bit 0:	Source Port State Bit from **Source Port Table** field **l2ActionTablePortState**.
Address Bit 1:	L2 SA Table was a hit. 0 = Miss. 1 = Hit.
Address Bit 2:	L2 SA Table - L2 Action Table Status bit. If this table was a miss then this bit will be zero.
Address Bit 3:	L2 DA Table - L2 Action Table Status bit. If this table was a miss then this bit will be zero.
Address Bit [5:4]:	L2 Packet Type. 0 = L2 Dest Table was a Unicast. 1 = L2 Dest Table was Multicast. 2 = L2 DA table was a miss and packet is being flooded. 3 = Packet was a Broadcast packet and L2 Dest Table did not hit. If both flooded and L2 Broadcast packet then this option will be selected.
Address Bit 6:	Destiantion Port State Bit comes from the **L2 Action Table Egress Port State**.

Address Space : 1044144 to 1044271

Field Description

Bits	Field Name	Description	Default Value
0	noLearningUc	The packet shall not be learned. This is applied to L2 DA MAC unicast packets.	0x0
1	noLearningMc	If the packet is a L2 Multicast then the packet shall not be learned. If a packet is a L2 Multicast depends on if the SA MAC MC bit is set.	0x0

Bits	Field Name	Description	Default Value
2	dropAll	The packet shall drop all instances and update counter **L2 Action Table Drop**. However special packets which are allowed will still be allowed into the switch (using the field **useSpecialAllow** set to one and register **Allow Special Frame Check For L2 Action Table**)	0x0
3	drop	The packet shall only drop on the ports which hits this action.	0x0
4	dropPortMove	The packet shall be dropped if the result from the learning lookup is port-move.	0x0
5	sendToCpu	The packet shall be send to the CPU.	0x0
6	forceSendToCpuOrigPkt	Force the packet to the CPU to be the original,unmodified, packet. The exception to this is rule is the tunnel exit which will still be carried out.	0x0
7	noPortMove	No port move is allowed for this packet.	0x0
8	useSpecialAllow	Use the special frame checks on this port. 0 = No. 1 = Yes.	0x0
10:9	allowPtr	Pointer to allow special packets defined in **Allow Special Frame Check For L2 Action Table**.	0x0
11	mmpValid	If set, this entry contains a valid MMP pointer	0x0
17:12	mmpPtr	Ingress MMP pointer.	0x0
19:18	mmpOrder	Ingress MMP pointer order.	0x0
20	forceQueue	Shall this packet be forced to a queue? 0 = No. 1 = Yes.	0x0
23:21	dstQueue	The new queue for this packet(s).	0x0

38.11.141 L2 Action Table Egress Port State

The egress port state for the L2 Action Table Lookup.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121241

Field Description

Bits	Field Name	Description	Default Value
10:0	state	What is the egress port status bits in the L2 Action Table for the egress port. Bit [0] are used for port 0, Bits [1] are used for port 1 and so on.	0x0

38.11.142 L2 Action Table Source Port

The L2 action table for source port is looked up at the same time as the **L2 Action Table** and its result is merged with the lookup from the **L2 Action Table** table, this lookup is active when enabled in the **Source Port Table** field **enableL2ActionTable** is set to one. The **L2 Action Table** is enabled for each of the destination ports the packet is going to, this table is looked up based on the source port and even if

the packet is going to no destination ports this lookup is still carried out. Another difference between **L2 Action Table** and this table is that the highest address bit (bit 6) which uses the status from the L2 SA Lookup and if the packet is going to do a port move then this address bit is high.

Number of Entries : 128

Type of Operation : Read/Write

Addressing :

Address Bit 0:	Source Port State Bit from **Source Port Table** field **l2ActionTablePortState**.
Address Bit 1:	L2 SA Table was a hit. 0 = Miss. 1 = Hit.
Address Bit 2:	L2 SA Table - L2 Action Table Status bit.
Address Bit 3:	L2 DA Table - L2 Action Table Status bit. If this table was a miss then this bit will be zero.
Address Bit [5:4]:	L2 Packet Type. 0 = L2 Dest Table was a Unicast. 1 = L2 Dest Table was Multicast. 2 = L2 DA table was a miss and packet is being flooded. 3 = Packet was a Broadcast packet and L2 Dest Table did not hit. If both flooded and L2 Broadcast packet then this option will be selected.
Address Bit [6]:	Port Move. Result bit from L2 SA lookup if the packet shall do a port move or not.

Address Space : 1044272 to 1044399

Field Description

Bits	Field Name	Description	Default Value
0	noLearningUc	The packet shall not be learned. This is applied to L2 DA MAC unicast packets.	0x0
1	noLearningMc	If the packet is a L2 Multicast then the packet shall not be learned. If a packet is a L2 Multicast depends on if the SA MAC MC bit is set.	0x0
2	dropAll	The packet shall drop all instances and update counter **L2 Action Table Drop**. However special packets which are allowed will still be allowed into the switch (using the field **useSpecialAllow** set to one and register **Allow Special Frame Check For L2 Action Table**)	0x0
3	drop	The packet shall only drop on the ports which hits this action.	0x0
4	dropPortMove	The packet shall be dropped if the result from the learning lookup is port-move.	0x0
5	sendToCpu	The packet shall be send to the CPU.	0x0
6	forceSendToCpuOrigPkt	Force the packet to the CPU to be the orig-inial,unmodified, packet. The exception to this is rule is the tunnel exit which will still be carried out.	0x0
7	noPortMove	No port move is allowed for this packet.	0x0
8	useSpecialAllow	Use the special frame checks on this port. 0 = No. 1 = Yes.	0x0
10:9	allowPtr	Pointer to allow special packets defined in **Allow Special Frame Check For L2 Action Table**.	0x0

Bits	Field Name	Description	Default Value
11	mmpValid	If set, this entry contains a valid MMP pointer	0x0
17:12	mmpPtr	Ingress MMP pointer.	0x0
19:18	mmpOrder	Ingress MMP pointer order.	0x0
20	forceQueue	Shall this packet be forced to a queue? 0 = No. 1 = Yes.	0x0
23:21	dstQueue	The new queue for this packet(s).	0x0

38.11.143 L2 Aging Collision Shadow Table

This table traces the **valid** field of the **L2 Aging Collision Table** and is used by L2 forwarding to check if a hit in the **L2 Lookup Collision Table** is valid. Any software write to this table shall be updated to the **valid** field of the **L2 Aging Collision Table**.

Number of Entries : 32
Type of Operation : Read/Write
Addressing : **L2 Lookup Collision Table** hit index
Address Space : 1119530 to 1119561

Field Description

Bits	Field Name	Description	Default Value
0	valid	If this is set, then the corresponding **L2 Lookup Collision Table** entry is valid.	0x0

38.11.144 L2 Aging Collision Table

This table holds the status of the entries in the **L2 Lookup Collision Table**. Any software write to the **valid** field in this table shall be done in the **L2 Aging Collision Shadow Table**.

Number of Entries : 32
Type of Operation : Read/Write
Addressing : **L2 Lookup Collision Table** hit index
Address Space : 337 to 368

Field Description

Bits	Field Name	Description	Default Value
0	valid	If this is set, then the corresponding **L2 Lookup Collision Table** entry is valid.	0x0
1	stat	If this is set, then the corresponding **L2 Lookup Collision Table** entry will not be aged out.	0x0
2	hit	If this is set, then the corresponding **L2 Lookup Collision Table** entry has a L2 SA/DA search hit since the last aging scan.	0x0

38.11.145 L2 Aging Status Shadow Table

This table traces the **valid** field of the **L2 Aging Table** and is used by L2 forwarding to check if a hit in the **L2 DA Hash Lookup Table** is valid. Any software write to this table shall be updated to the **valid** field of the **L2 Aging Table**. Any software write to this table shall be copied to the **L2 Aging Status Shadow Table - Replica**

Number of Entries :	16384
Type of Operation :	Read/Write
Addressing :	address[0:10] : hash of {GID, destination MAC} address[11:13] : bucket number
Address Space :	880112 to 896495

Field Description

Bits	Field Name	Description	Default Value
0	valid	If this is set, then the corresponding hash table entry is valid.	0x0

38.11.146 L2 Aging Status Shadow Table - Replica

This table traces the **valid** field of the **L2 Aging Table** and is used by L2 forwarding to check if a hit in the **L2 SA Hash Lookup Table** is valid. Content of this table shall be identical as the **L2 Aging Status Shadow Table**.

Number of Entries :	16384
Type of Operation :	Read/Write
Addressing :	address[0:10] : hash of {GID, source MAC} address[11:13] : bucket number
Address Space :	994928 to 1011311

Field Description

Bits	Field Name	Description	Default Value
0	valid	If this is set, then the corresponding hash table entry is valid.	0x0

38.11.147 L2 Aging Table

This table uses the same addressing as the **L2 DA Hash Lookup Table** to show the status of each entries in that table. Any software write to any valid field in this table shall be done in the **L2 Aging Status Shadow Table**. Any software write to this table shall be copied to the **L2 Aging Status Shadow Table - Replica**

Number of Entries :	16384
Type of Operation :	Read/Write
Addressing :	address[0:10] : hash of {GID, destination MAC} address[11:13] : bucket number
Address Space :	380 to 16763

Field Description

Bits	Field Name	Description	Default Value
0	valid	If set, then the corresponding hash table entry is valid.	0x0
1	stat	If set, then the corresponding hash table entry will not be aged out.	0x0
2	hit	If set, then the corresponding hash table entry has a L2 DA search hit since the last aging scan.	0x0

38.11.148 L2 DA Hash Lookup Table

The L2 table is used for hash search based on the destination MAC address and a GID from the **VLAN Table**. When performing a L2 destination port lookup, {GID, destination MAC} is used as key for a hash calculation (see Section MAC Table Hashing). The hash is then used as index into this table to read out the 8 buckets. The incoming {GID, destination MAC} are compared to all the buckets. If any of the buckets match then address was known. The result of the lookup will be read from the **L2 Destination Table** at the same address as the matching hash index and bucket. Any software write to this table shall be copied to the **L2 SA Hash Lookup Table**.

Number of Entries :	16384
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	address[0:10] : hash of {GID, destination MAC} address[11:13] : bucket number
Address Space :	896496 to 929263

Field Description

Bits	Field Name	Description	Default Value
47:0	macAddr	MAC address.	0x0
59:48	gid	Global identifier from the VLAN Table.	0x0

38.11.149 L2 Destination Table

This table contains either a destination port or a pointer to the L2 multicast table.Any software write to this table shall be copied to the **L2 Destination Table - Replica**.

Number of Entries :	16416
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	address 0 to16383 : **L2 DA Hash Lookup Table** address address 16384 to16415 : **L2 Lookup Collision Table** address
Address Space :	929264 to 962095

Field Description

Bits	Field Name	Description	Default Value
0	uc	Unicast if set; multicast if cleared. Multicast means that a lookup to the **L2 Multicast Table** will occur and determine a list of destination ports.	0x0
9:1	destPort_or_mcAddr	Destination port number or pointer into the **L2 Multicast Table**.	0x0
10	pktDrop	If set, the packet will be dropped and the **L2 Lookup Drop** incremented.	0x0
11	l2ActionTableDaStatus	The status DA bit to be used in the addressing for the table **L2 Action Table** Lookup.	0x0
12	l2ActionTableSaStatus	The status SA bit to be used in the addressing for the table **L2 Action Table** Lookup.	0x0
13	tunnelEntry	Shall this packet enter into a tunnel.	0x0
18:14	tunnelEntryPtr	The tunnel entry which this packet shall enter upon exiting the switch. If a multicast l2 entry is selected then this field points to **Tunnel Entry Instruction Table** using the destiantion port as a offset into the table.	0x0
19	tunnelExit	Shall this packet do a tunnel exit? 0 = No 1 = Yes	0x0
23:20	tunnelExitPtr	Pointer to tunnel exit description in **Egress Tunnel Exit Table**	0x0
39:24	metaData	Meta data for to CPU tag.	0x0

38.11.150 L2 Destination Table - Replica

This table is replicated from the **L2 Destination Table** and used by the learning engine allowing the learning engine and packet forwarding to process in parallel. Content of this table shall be identical as the **L2 Destination Table**.

Number of Entries :	16416
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	address 0 to **L2 SA Hash Lookup Table** address 16383 : address 16384 to **L2 Lookup Collision Table** address 16415 :
Address Space :	1011312 to 1044143

Field Description

Bits	Field Name	Description	Default Value
0	uc	Unicast if set; multicast if cleared. Multicast means that a lookup to the **L2 Multicast Table** will occur and determine a list of destination ports.	0x0
9:1	destPort_or_mcAddr	Destination port number or pointer into the **L2 Multicast Table**.	0x0
10	pktDrop	If set, the packet will be dropped and the **L2 Lookup Drop** incremented.	0x0

Bits	Field Name	Description	Default Value
11	l2ActionTableDaStatus	The status DA bit to be used in the addressing for the table **L2 Action Table** Lookup.	0x0
12	l2ActionTableSaStatus	The status SA bit to be used in the addressing for the table **L2 Action Table** Lookup.	0x0
13	tunnelEntry	Shall this packet enter into a tunnel.	0x0
18:14	tunnelEntryPtr	The tunnel entry which this packet shall enter upon exiting the switch. If a multicast l2 entry is selected then this field points to **Tunnel Entry Instruction Table** using the destiantion port as a offset into the table.	0x0
19	tunnelExit	Shall this packet do a tunnel exit? 0 = No 1 = Yes	0x0
23:20	tunnelExitPtr	Pointer to tunnel exit description in **Egress Tunnel Exit Table**	0x0
39:24	metaData	Meta data for to CPU tag.	0x0

38.11.151 L2 Lookup Collision Table

Collision table for the **L2 DA Hash Lookup Table**. If there is a hash collision and all the buckets for that hash index are occupied then additional entries can be stored in the collision table. When searching this table, all entries are compared in parallel and the matching entry with the lowest address will be used as a match result. Chapter Learning and Aging describes how to search and write to this table.

Number of Entries :	32
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	1122339 to 1122402

Field Description

Bits	Field Name	Description	Default Value
47:0	macAddr	MAC address	0x0
59:48	gid	Global identifier for learning	0x0

38.11.152 L2 Lookup Collision Table Masks

Masks for collision memory for the MAC address and the global identifier. Only the first 4entries has masks on them.

Number of Entries :	4
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	1122331 to 1122338

Field Description

Bits	Field Name	Description	Default Value
47:0	macAddr	MAC address mask	$2^{48}-1$
59:48	gid	Global identifier for learning mask	0xfff

38.11.153 L2 Multicast Handling

Exceptions for L2 multicast flag handling, only valid for the Multicast Broadcast Storm Control and the Ingress Egress Port Packet Type Filter. The switch sets by default a L2 multicast flag when DA is an Ethernet multicast address (i.e. DA with the least-significant bit of the first octet equals 1 (e.g. 01:80:c2:00:00:00) but not equal to ff:ff:ff:ff:ff:ff).

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121242

Field Description

Bits	Field Name	Description	Default Value
0	exclIPv4Mc	If set, IPv4 packets with IPv4 multicast MAC address will NOT have a L2 multicast flag.	0x0
1	exclIPv6Mc	If set, IPv6 packets with IPv6 multicast MAC address will NOT have a L2 multicast flag.	0x0
2	inclL2McLut	If set, packets that are forwarded by **L2 Multicast Table** will internally be treated as the L2 multicast bit in the L2 DA address would have been set to one.	0x1
3	inclMultiPorts	If set, packets that end up in more than one destination port but not due to broadcast or flooding will have a L2 multicast flag. Observe that mirroring is not a valid multiport destination.	0x0
4	unknownL2McFilterRule	Select the filtering rules for unknown L2 multicast MAC DA in the **Ingress Egress Port Packet Type Filter**. 0 = **dropL2FloodingFrames** 1 = **dropL2MulticastFrames**	0x0

38.11.154 L2 Multicast Table

L2 multicast table.

Number of Entries : 512
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Addressing : mcAddr field from **L2 Destination Table** or from **Next Hop Table**
Address Space : 1121307 to 1122330

Field Description

Bits	Field Name	Description	Default Value
10:0	mcPortMask	L2 portmask entry members. If set, the port is part of multicast group and shall be transmitted to.	0x7ff
11	tunnelEntry	Shall this packet enter into a tunnel.	0x0
16:12	tunnelEntryPtr	The tunnel entry which this packet shall enter upon exiting the switch. If a multicast l2 entry is selected then this field points to **Tunnel Entry Instruction Table**. The destination port is used as a offset from this number.	0x0
17	tunnelExit	Shall this packet do a tunnel exit. 0 = No 1 = Yes	0x0
21:18	tunnelExitPtr	Pointer to tunnel exit described in **Egress Tunnel Exit Table**.	0x0
37:22	metaData	Meta data for to CPU tag.	0x0

38.11.155 L2 Reserved Multicast Address Action

If the higher bits of the incoming packets MAC DA address matches the **L2 Reserved Multicast Address Base** then the lower bits are used as index into this table. The action can be to drop the packet, send the packet to the CPU or just process the packet in the normal L2 pipeline.

Number of Entries : 256
Type of Operation : Read/Write
Addressing : MAC DA[7:0]
Address Space : 1120692 to 1120947

Field Description

Bits	Field Name	Description	Default Value
10:0	dropMask	Determines which source ports that are not allowed to receive this multicast address. Each bit set to 1 will result in dropping this multicast address on that source port. Bit 0 is port 0, bit 1 is port 1 etc. Each drop will be counted in **L2 Reserved Multicast Address Drop**.	0x0
21:11	sendToCpuMask	Received packets on these source ports will be sent to the CPU. Bit 0 represents port 0, bit 1 represents port 1 etc. LLDP frames sent to the CPU takes priority over this.	0x0

38.11.156 L2 Reserved Multicast Address Base

Certain L2 Destination MAC addresses shall be treated special when entering the switch. If the first 40 bits of the Destination MAC address matches the macBase field then the lowest 8 bits are used as index into the **L2 Reserved Multicast Address Action** table.

Number of Entries : 1
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 1122481

Field Description

Bits	Field Name	Description	Default Value
39:0	macBase	The first 40 bits of the reserved MAC address, and the lower 16 bits of it can be masked. The default is 01:80:c2:00:00	0x180c20000
55:40	mask	Bit comparison mask for the lower 2 bytes in macBase (marked with XX as in 01:80:c2:XX:XX). If a bit is set in the mask then the corresponding bit will be compared. Otherwise the bits are dont care.	0xffff

38.11.157 L2 SA Hash Lookup Table

L2 table used for hash search based on the source MAC and a GID from the **VLAN Table**. When performing a SA MAC learning check {GID, Source MAC} is used as key for a hash function (see Section MAC Table Hashing) which calculates a hash index. The hash index points to this table that has 8 buckets. The incoming {GID, source MAC} are compared to all the 8 buckets. If any of the buckets match then address was known. The result of the lookup will be read from the **L2 Destination Table - Replica** at the same address as the matching hash index and bucket. Content of this table shall be identical as the **L2 DA Hash Lookup Table**.

Number of Entries :	16384
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	address[0:10] : hash of {GID, source MAC} address[11:13] : bucket number
Address Space :	962160 to 994927

Field Description

Bits	Field Name	Description	Default Value
47:0	macAddr	MAC address.	0x0
59:48	gid	Global identifier from the VLAN Table.	0x0

38.11.158 L2 Tunnel Decoder Setup

The tunnel TPID setup is setup in this register. This is used by the tunnel packet decoder. Besides the configurable values the default Ethernet Type values of 0x8100 is detected as a C-type VLAN ID while 0x9100, 0x9200 and 0x88A8 is descoverable as S-type VLAN IDs.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	1122473

Field Description

Bits	Field Name	Description	Default Value
0	defaultEthCTypeValid	The configurable Ethernet Type C-type is valid.	0x0
16:1	defaultEthCType	A configurable Ethernet Type which shall be used to determine a C-Type VLAN.	0x0
17	defaultEthSTypeValid	The configurable Ethernet Type S-type is valid.	0x0
33:18	defaultEthSType	A configurable Ethernet Type which shall be used to determine a S-Type VLAN.	0x0

38.11.159 L3 LPM Result

This is the longest prefix routing table result. The index into the table is the hit index from the **L3 Routing TCAM**.

Number of Entries : 32
Type of Operation : Read/Write
Addressing : **L3 Routing TCAM** hit index
Address Space : 347600 to 347631

Field Description

Bits	Field Name	Description	Default Value
0	useECMP	Enables the use of ECMP hash to calculate the next hop pointer. 0 = Use ECMP hash. 1 = Do not use ECMP hash.	0x0
6:1	ecmpMask	How many bits of the ECMP hash will be used when calculating the ECMP offset. This byte is AND:ed with the ECMP hash to determine which bits shall be used as offset.	0x0
9:7	ecmpShift	How many bits the masked ECMP hash will be right shifted.	0x0
20:10	nextHopPointer	Index into the **Next Hop Table** for this destination.	0x0
23:21	entryVersion	The version of this entry. All other tables which points from this table must have same version.	0x0

38.11.160 L3 Routing Default

The default router to be used if the destination lookup in L3 tables fails, i.e does not match either the LPM or the hash tables.

Number of Entries : 4
Type of Operation : Read/Write
Addressing : **vrf**
Address Space : 1120634 to 1120637

Field Description

Bits	Field Name	Description	Default Value
10:0	nextHop	The default next hop to be used. Index into the **Next Hop Table**.	0x0
11	pktDrop	If set the packet will be drop and the **L3 Lookup Drop** counter incremented.	0x0
12	sendToCpu	If set then the packet will be sent to the CPU.	0x0
13	mmpValid	If set, this entry contains a valid MMP pointer	0x0
19:14	mmpPtr	Ingress MMP pointer.	0x0
21:20	mmpOrder	Ingress MMP pointer order.	0x0
24:22	entryVersion	The version of this entry. All other tables which points from this table must have same version.	0x0

38.11.161 L3 Routing TCAM

This is the longest prefix match routing table used to determine the next hop. This table is compared from the highest address and downwards. The match which has the highest entry number is selected. The selected entry number is used to index the**L3 LPM Result** table to provide the next hop result. For each entry the mask determines which bits that shall be compared. An entry contains three parts: valid flag, comparison fields and field masks. Each comparison field is associated with a mask to optionally ignore several bits or even the entire field during comparison. To allow any value on a certain bit, the corresponding bit in the mask shall be set to 1. As a consequence, the field will have that bit nailed to 0 if read and ignored during lookup. Hit in multiple entries will return the first hit index (lowest address/index) to lookup in the result table.

Number of Entries :	32
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Addressing :	Entry number
Address Space :	1125540 to 1126051

Field Description

Bits	Field Name	Description	Default Value
1:0	proto	Select if this is an IPv4, IPv6 or MPLS entry. 0 = Reserved 1 = MPLS Entry. 2 = IPv4 entry. 3 = IPv6 entry. protoMaskN determines the bits in the field that can be ignored for comparison.	0x0
3:2	vrf	This entries VRF. The packets assigned VRF will be compared with this field. vrfMaskN determines the bits in the field that can be ignored for comparison.	0x0
131:4	destIPAddr	The IP or MPLS address to be matched. If the entry is an IPv4 entry then bits [31:0] should be set to the IPv4 address. If the entry is an MPLS entry then bits [4-1:0] should contain the source port while bits [4+19:4] should contain the MPLS label. destIPAddrMaskN determines the bits in the field that can be ignored for comparison.	0x0

Bits	Field Name	Description	Default Value
133:132	protoMaskN	Mask for the proto field. For each bit in the mask, 0 means the bit is valid for comparison, 1 means the comparison ignores this bit.	0x0
135:134	vrfMaskN	Mask for the vrf field. For each bit in the mask, 0 means the bit is valid for comparison, 1 means the comparison ignores this bit.	0x0
263:136	destIPAddrMaskN	Mask for the destIPAddr field. For each bit in the mask, 0 means the bit is valid for comparison, 1 means the comparison ignores this bit.	0x0
264	valid	If set, this entry is valid	0x0

38.11.162 LACP Packet Decoder Options

This is the MAC address used to determine that a packet is a LACP packet. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries : 1
Number of Addresses per Entry : 4
Type of Operation : Read/Write
Address Space : 1122671

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
48:1	mac	The value to be used to find this packet type.	0x180c2000002
59:49	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
70:60	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

38.11.163 LLDP Configuration

A LLDP packet is identified as a LLDP frame if the packets MAC DA matches one of the mac1-mac3 fields and the packets EtherType matches eth. The portmask field determines if an identified LLDP packet will bypass the normal packet processing and instead be sent to the CPU or if the packet should pass through normal packet processing.

Number of Entries : 1
Number of Addresses per Entry : 8
Type of Operation : Read/Write
Address Space : 1124315

Field Description

Bits	Field Name	Description	Default Value
47:0	mac1	DA MAC address to match for LLDP packet.	0x180c200000e
95:48	mac2	DA MAC address to match for LLDP packet.	0x180c2000003
143:96	mac3	DA MAC address to match for LLDP packet.	0x180c2000000
159:144	eth	The Ethernet Type for a LLDP	0x88cc
160	bpduOption	If both LLDP and BPDU are valid, because the BPDU has same MAC address as LLDP, then this option allows the BPDU identification to be turned off 0 = Don't do anything. Both LLDP and BPDU can be valid at same time. 1 = Remove BPDU valid causing that the packet will only be seen as a LLDP packet and not a BPDU frame and the new frame will not be sent to the CPU because the switch will no longer consider it a BPDU frame, this includes Rapid Spanning Tree BPDUs also.	0x0
171:161	portmask	One bit per source port, bit 0 for port 0, bit 1 for port 1 etc. 0 = Do not sent a matched LLDP packet to the CPU from this port. Packet will pass through normal packet processing. 1 = Send a matched LLDP packet to CPU from this source port and hence bypassing normal processing.	0x3ff

38.11.164 Learning And Aging Enable

Enable/Disable the learning and aging function. If software needs to take fully control over learning and aging tables by writting to the FIB directly, the learning and aging units should be completely turned off, which means all fields in this register have to be cleared to 0, partly reset is not allowed. When the learning and aging units are turned on, software still has controllablity over learning and aging by injecting user defined learning packets.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 318

Field Description

Bits	Field Name	Description	Default Value
0	learningEnable	If set, the learning unit will be activated.	0x1
1	agingEnable	If set, the aging unit will be activated.	0x1
2	daHitEnable	If set, MAC DA hit in the forwarding information base will update the hit bit for non-static entries.	0x1
3	lru	If set, the learning unit will try to overwrite a least recently used non-static entry in either the hash table or the collision table when there is no free entry to use. Otherwise the learning unit will try to overwrite a non-static entry in the collision table.	0x0

38.11.165 Learning And Aging Writeback Control

Determine how the hardware learning and aging engine act on injected learning packets. By default all the hardware and software learning/aging/hit results can be updated to the FIB. If software needs more controllability, the learning/aging/hit decisions from hardware can be configured to only send to corresponding writeback FIFOs but not write to the FIB.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 320

Field Description

Bits	Field Name	Description	Default Value
0	hwLearningWriteBack	If set, the hardwrae learning result from unknown or port moved source MAC will be pushed to the **Learning Data FIFO** and written to the FIB simultaneously. Otherwise the result is only pushed to the FIFO.	0x1
1	hwAgingWriteBack	If set, the aging result will be pushed to the **Aging Data FIFO** and written to the FIB simultaneously. Otherwise the result is only pushed to the FIFO and software has to read it out then send in a corresponding learning packet if this aging result should be written to the tables.	0x1
2	hwHitWriteBack	If set, the hit update of a learned destination MAC will be pushed to **Hit Update Data FIFO** and written to the FIB simulatneously. Otherwise the result is only pushed to the FIFO and then software decides the FIB writes.	0x1
3	adfPushOption	By default the **Aging Data FIFO** contains all hardware aging requests, including modifing the hit state and clearing the entry. Set this field to 1 to only push when an entry needs to be cleared/aged out.	0x0

38.11.166 Learning Conflict

Status register for the failed port move operation. A valid status means the L2 Forwarding Information Base cannot bind the existing GID, MAC to a new port. Once the status register is updated from the hardware, no more fails can be updated untill the software clears the valid field.

Number of Entries : 1
Number of Addresses per Entry : 4
Type of Operation : Read/Write
Address Space : 310

Field Description

Bits	Field Name	Description	Default Value
0	valid	Indicates hardware has written a learning conflict to this status register. Write 0 to clear.	0x0
48:1	macAddr	MAC address.	0x0
60:49	gid	Global identifier from the VLAN Table.	0x0
64:61	port	Port number.	0x0

38.11.167 Learning DA MAC

The MAC address to be used by packets which are injected by software to be learned.

Number of Entries : 1
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 1122475

Field Description

Bits	Field Name	Description	Default Value
47:0	mac	The destination MAC address to be used by software when injecting new addresses to be learned	0x0
48	enable	Shall the switch accept learning packets? 0 = No 1 = Yes	0x0

38.11.168 Learning Data FIFO

This register exposes the output of a FIFO which is holding all learning requests that have been processed by the learning unit. A read from this register will pop one entry from the fifo. Under hardware learning writeback mode, all valid entries have been updated to the FIB regardless of hardware or software learning. When hardware learning writeback is turned off, software takes full control of the learning unit, hardware learning result will only be pushed to this FIFO but not update the related L2 tables.

Number of Entries : 1
Number of Addresses per Entry : 4
Type of Operation : Read Only
Address Space : 16765

Field Description

Bits	Field Name	Description	Default Value
47:0	mac	MAC address for a learning request.	0x0
63:48	gid	Global IDentifier from the **gid** field in the **VLAN Table**.	0x0
78:64	destAddress	The **L2 Destination Table** address decided by the learning unit.	0x0
79	uc	The **uc** field in the **L2 Destination Table** decided by the learning unit.	0x0

Bits	Field Name	Description	Default Value
88:80	port_or_ptr	The **destPort or mcAddr** field in the **L2 Destination Table** decided by the learning unit.	0x0
89	drop	The **pktDrop** field in the **L2 Destination Table** decided by the learning unit.	0x0
90	l2ActionTableDaStatus	**l2ActionTableDaStatus** field in the **L2 Destination Table**	0x0
91	l2ActionTableSaStatus	**l2ActionTableSaStatus** field in the **L2 Destination Table**	0x0
92	tunnelEntry	**tunnelEntry** field in the **L2 Destination Table**	0x0
97:93	tunnelEntryPtr	**tunnelEntryPtr** field in the **L2 Destination Table**	0x0
98	tunnelExit	**tunnelExit** field in the **L2 Destination Table**	0x0
102:99	tunnelExitPtr	**tunnelExitPtr** field in the **L2 Destination Table**	0x0
118:103	metaData	**metaData** field in the **L2 Destination Table**	0x0
121:119	status	Entry status either refers to the **L2 Aging Table** or the **L2 Aging Collision Table** based on the **destAddress** field.	0x0
122	type	Type of the learning request. 0 = Hardware learning result 1 = Software learning result from a injected learning packet	0x0
123	valid	0 = FIFO is empty 1 = FIFO is not empty and the data is valid	0x0

38.11.169 Learning Data FIFO High Watermark Level

The High Watermark Interrupt will occur when a push to **Learning Data FIFO** is done and the number of existing entries after the push is larger than this setting.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 321

Field Description

Bits	Field Name	Description	Default Value
5:0	level	Number of used entries.	0x0

38.11.170 Learning Overflow

Status register for the failed hardware learning operation. A valid status means the L2 Forwarding Information Base cannot find an available slot for the unknown GID, MAC. Once the status register is updated from the hardware, no more fails can be updated untill the software clears the valid field.

Number of Entries : 1
Number of Addresses per Entry : 4
Type of Operation : Read/Write
Address Space : 314

Field Description

Bits	Field Name	Description	Default Value
0	valid	Indicates hardware has written a learning overflow to this status register, Write 0 to clear.	0x0
48:1	macAddr	MAC address.	0x0
60:49	gid	Global identifier from the VLAN Table.	0x0
64:61	port	Port number.	0x0

38.11.171 Link Aggregate Weight

The link aggregate hash will index into this table to determine which physical port within the aggregate that a packet should be output to. The number of bits set for a port will determine the ratio of packets that will go out on that port. For each hash index only one of the ports that belong to the same link aggregate must be set. The number of bits set divided by number of hash values determines the ratio of traffic going to that port. All link aggregates share this table since each physical port can only belong to one link aggregate. When a link aggregate only has one port then all bits for that port must be set.

Number of Entries : 256
Type of Operation : Read/Write
Addressing : The link aggregate hash.
Address Space : 1118668 to 1118923

Field Description

Bits	Field Name	Description	Default Value
10:0	ports	One bit per physical port.	0x0

38.11.172 Link Aggregation Ctrl

This register controls whether link aggregation is enabled and which packet header fields that will be used to calculate the link aggregate hash value.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121215

Field Description

Bits	Field Name	Description	Default Value
0	enable	Is Link aggregation enabled or not. 0 = Link Aggregation is disabled 1 = Link Aggregation is enabled	0x0
1	useSaMacInHash	The packets source MAC address shall be part of the hash key when calculating the link aggregate hash value	0x0

Bits	Field Name	Description	Default Value
2	useDaMacInHash	The packets destination MAC addresses shall be part of the hash key when calculating the link aggregate hash value	0x0
3	useIpInHash	The packets IP source and destination addresses shall be part of the hash key when calculating the link aggregate hash value	0x0
4	useL4InHash	The packets L4 SP / DP and L4 protocol byte shall be part of the hash key when calculating the link aggregate hash value	0x0
5	useTosInHash	The incoming packets TOS byte shall be part of the hash key when calculating the link aggregate hash value	0x0
6	useNextHopInHash	For routed packets the next hop entry shall be part of the hash key when calculating the link aggregate hash value.	0x0
7	useVlanIdInHash	The packets VLAN Identifier tag shall be part of the hash key when calculating the link aggregate hash value.	0x0

38.11.173 Link Aggregation Membership

This register is used to determine which link aggregation a specific source port is membership of. If link aggregation is enabled then this port number is used for all source lookups instead of the port where the packet enterned the switch.

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Ingress port
Address Space : 1121204 to 1121214

Field Description

Bits	Field Name	Description	Default Value
3:0	la	The Link aggregation which this port is a member of	0x0

38.11.174 Link Aggregation To Physical Ports Members

This link aggregate portmasks are setup to determine which physical ports are members of each link aggregate.

Number of Entries : 11
Type of Operation : Read/Write
Addressing : The link aggregate number.
Address Space : 1118657 to 1118667

Field Description

Bits	Field Name	Description	Default Value
10:0	members	Physical ports that are members of this link aggregate. One bit per port.	0x0

38.11.175 MACsec Port

Configurations to enforce MACsec encryption on a specific port. Packets dropped by this functionality are counted in **MACsec Drops**.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121246

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is the MACsec function enabled? 0 = Disabled 1 = Enabled	0x0
4:1	portId	Which port shall use MACsec on.	0x0
5	dropNonEncryptedPackets	If an unencrypted packet comes in on this port it will be dropped. 0 = No, Do not drop packets 1 = Yes, Drop packets.	0x0
11:6	encryptSaPtr	Which Security Association shall be used for encryption of packets.	0x0
12	encryptBPDU	Encrypt BPDU frames. 0 = Encrypt frame. 1 = Do not encrypt frame.	0x0
13	dropBPDU	Drop Incoming Non encrypted BPDU frames. 0 = Allow frames. 1 = Drop them.	0x0
14	encryptLACP	Encrypt LACP frames. 0 = Encrypt frame. 1 = Do not encrypt frame.	0x0
15	dropLACP	Drop Incoming Non encrypted LACP frames. 0 = Allow frames. 1 = Drop them.	0x0
16	encryptLLDP	Encrypt LLDP frames. 0 = Encrypt frame. 1 = Do not encrypt frame.	0x0
17	dropLLDP	Drop Incoming Non encrypted LLDP frames. 0 = Allow frames. 1 = Drop them.	0x0
18	encryptL2_1588	Encrypt L2 1588 frames. 0 = Encrypt frame. 1 = Do not encrypt frame.	0x0
19	dropL2_1588	Drop Incoming Non encrypted L2_1588 frames. 0 = Allow frames. 1 = Drop them.	0x0

Bits	Field Name	Description	Default Value
20	encryptEAPOL	Encrypt EAPOL frames. 0 = Encrypt frame. 1 = Do not encrypt frame.	0x0
21	dropEAPOL	Drop Incoming Non encrypted EAPOL frames. 0 = Allow frames. 1 = Drop them.	0x0
22	encryptL2McReserved	Encrypt L2 Reserved Da frames, both LLDP and BPDU frames have been excluded from this setting, see register **L2 Reserved Multicast Address Base**. 0 = Encrypt frame. 1 = Do not encrypt frame.	0x0
23	dropL2McReserved	Drop Incoming Non encrypted L2McReserved frames, both LLDP and BPDU frames have been excluded from this setting. 0 = Allow frames. 1 = Drop them.	0x0

38.11.176 MPLS EXP Field To Egress Queue Mapping Table

Mapping table from MPLS EXP priority fields to egress queues.

Number of Entries : 8
Type of Operation : Read/Write
Addressing : Incoming packets MPLS EXP priority bits
Address Space : 1120106 to 1120113

Field Description

Bits	Field Name	Description	Default Value
2:0	pQueue	Egress queue	0x1

38.11.177 MPLS EXP Field To Packet Color Mapping Table

Mapping table from MPLS EXP priority fields to packet initial color.

Number of Entries : 8
Type of Operation : Read/Write
Addressing : Incoming packets MPLS EXP priority bits
Address Space : 1119570 to 1119577

Field Description

Bits	Field Name	Description	Default Value
1:0	color	Packet initial color	0x0

38.11.178 NAT Action Table

At end of ingress processing a check is done to determine what to do with the packets. This table is used to setup operations based on the port states **Egress Port NAT State** and **natPortState**

Number of Entries : 512
Type of Operation : Read/Write

Addressing :

Address Bit 0 :	Source Port NAT State **natPortState**
Address Bit 1 :	Egress Port NAT State **Egress Port NAT State**
Address Bit [3:2] :	Was packet Switched or Routed 0 = Other - sendToCpu,sendToPort from classification. 1 = Routed. 2 = Flooded. 3 = Switched - The packet hit a DA table entry and was not routed.
Address Bit 4 :	Was ingress ACL NAT operation enabled. 0 = No 1 = Yes
Address Bit 5 :	Was egress ACL NAT operation enabled. 0 = No 1 = Yes
Address Bit 6 :	Was the packet switched/routed unicast/multicast. Only valid for switching and routing. Otherwise set to zero. 0 = Multicast. 1 = Unicast.
Address Bit [8:7] :	L3 Packet Type 0 = IPv4 1 = IPv6 2 = MPLS 3 = Other

Address Space : 1118935 to 1119446

Field Description

Bits	Field Name	Description	Default Value
1:0	action	What to do with the packet depending on what port states are. 0 = No Operation 1 = Send to CPU Reason NAT Action Table Code 1 2 = Send to CPU Reason NAT Action Table Code 2 3 = Drop the packet. Update counter **NAT Action Table Drop**.	0x0

38.11.179 NAT Action Table Force Original Packet

If the NAT Action Table forces packets to be send to the CPU then they can either be the processed packet or the original packet. This register sets up for each reason what the packet to the CPU shall be.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121245

Field Description

Bits	Field Name	Description	Default Value
0	reasonOne	Force the packet to the CPU from the NAT action table Reason type 1 to be the original packet.	0x0
1	reasonTwo	Force the packet to the CPU from the NAT action table Reason type 2 to be the original packet.	0x0

38.11.180 Next Hop Packet Modifications

Determines the VLAN operations to perform on the packet exiting the router. One or two VLAN headers can be added to the outgoing packet.

Number of Entries : 2048
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Addressing : **nextHopPacketMod**
Address Space : 876016 to 880111

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this a valid entry. If the router points to an entry with this field cleared the packet will be sent to CPU. 0 = Invalid 1 = Valid	0x0
1	outerVlanAppend	Insert/push an outer VLAN header in the packet. The information used to create the new VLAN header is controlled by the fields **outerVid**, **outerPcpSel**, **outerCfiDeiSel** and **outerEthType**. If the selected outermost VLAN header field doesn't exist in the packet then the new VLAN header field will be taken from **Router Egress Queue To VLAN Data**. 0 = No operation. 1 = Insert/push an outer VLAN tag.	0x0
3:2	outerPcpSel	Selects which PCP bits to use when building an outer VLAN header. 0 = From outermost VLAN header in the original packet (if any). 1 = From this entrie's **outerPcp** field. 2 = From **Router Egress Queue To VLAN Data**.	0x0
5:4	outerCfiDeiSel	Selects which CFI/DEI bit to use when building an outer VLAN header. 0 = From outermost VLAN header in the original packet (if any). 1 = From this entrie's **outerCfiDei** field. 2 = From **Router Egress Queue To VLAN Data**.	0x0
7:6	outerEthType	Pointer to the VLAN type. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN.	0x0
19:8	outerVid	The VID used when building an outer VLAN header.	0x0

Bits	Field Name	Description	Default Value
22:20	outerPcp	The PCP bits to use when building an outer VLAN header. If selected by **outerPcpSel**.	0x0
23	outerCfiDei	The CFI/DEI bit to use when building an outer VLAN header. If selected by **outerCfiDeiSel**.	0x0
24	innerVlanAppend	Insert/push an inner VLAN header in the packet. The information used to create the new VLAN header is controlled by the fields **innerVid**, **innerPcpSel**, **innerCfiDeiSel** and **innerEthType**. If the selected innermost VLAN header field doesn't exist in the packet then the new VLAN header field will be taken from **Router Egress Queue To VLAN Data**. 0 = No operation 1 = Insert/push an inner VLAN tag.	0x0
26:25	innerPcpSel	Selects which PCP bits to use when building an inner VLAN header. 0 = From innermost VLAN header in the original packet (if any). 1 = From this entrie's **innerPcp** field. 2 = From **Router Egress Queue To VLAN Data**.	0x0
28:27	innerCfiDeiSel	Selects which CFI/DEI bit to use when building an inner VLAN header. 0 = From innermost VLAN header in the original packet (if any). 1 = From this entrie's **innerCfiDei** field. 2 = From **Router Egress Queue To VLAN Data**.	0x0
30:29	innerEthType	Pointer to the VLAN type. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN.	0x0
42:31	innerVid	The VID used when building an inner VLAN header.	0x0
45:43	innerPcp	The PCP bits to use when building an inner VLAN header. If selected by **innerPcpSel**.	0x0
46	innerCfiDei	The CFI/DEI bit to use when building an inner VLAN header. If selected by **innerCfiDeiSel**.	0x0
50:47	msptPtr	The multiple spanning tree to be used by packets for egress spanning tree check for this next hop. Points to an entry in **Egress Multiple Spanning Tree State**	0x0
53:51	entryVersion	The version of this entry. Must be same version as the entry pointing to it.	0x0

38.11.181 Next Hop Table

Forwarding decision for a routed packet including destination port(s), or if packet shall be dropped or sent to the CPU port.

Number of Entries :	2048
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	Next Hop Pointer
Address Space :	871920 to 876015

Field Description

Bits	Field Name	Description	Default Value
0	validEntry	Is this a valid entry or not. If the entry is not valid then the packet shall be sent to the CPU for further processsing	0x0
1	srv6Sid	If set, this entry is a locally instantiated SRv6 segment identifier	0x0
12:2	nextHopPacketMod	Pointer into the **Next Hop Packet Modifications** table and the **Next Hop DA MAC** table.	0x0
13	l2Uc	L2 unicast or multicast. A multicast means that a lookup in the **L2 Multicast Table** will take place to determine the destination portmask. 0 = L2 multicast. 1 = L2 unicast.	0x0
22:14	destPort_or_mcAddr	Destination port number or a pointer into the **L2 Multicast Table**	0x0
23	pktDrop	If set then the packet will be dropped and the **L3 Lookup Drop** incremented.	0x0
24	sendToCpu	If set then the packet will be sent to the CPU.	0x0
25	tunnelEntry	Shall this packet enter into a tunnel.	0x0
30:26	tunnelEntryPtr	The tunnel entry which this packet shall enter upon exiting the router. If field l2Uc is set to L2 multicast then **Tunnel Entry Instruction Table** uses the egress port as a offset from this base pointer.	0x0
31	tunnelExit	Shall this packet do a tunnel exit. 0 = No 1 = Yes	0x0
35:32	tunnelExitPtr	Pointer to tunnel exit described in **Egress Tunnel Exit Table**.	0x0
36	sendToCrypto	Send packet for IPSec processing in Crypto Engine using the security association in **saPtr**	0x0
42:37	saPtr	Security Association entry to be used.	0x0
43	cryptoOp	Which Crypto Operation to perform. 0 = Decrypt 1 = Encrypt	0x0
59:44	metaData	Meta data for to CPU tag.	0x0
62:60	entryVersion	The version of this entry. All other tables which points from this table must have same version.	0x0

38.11.182 Port Move Options

Determine if port move is allowed on static entries.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121240

Field Description

Bits	Field Name	Description	Default Value
10:0	allowPortMoveOnStatic	This field configures which source ports that are allowed to change their static GID and MAC to other ports. One bit for each port where bit 0 corresponds to port 0. When the L2 forwarding information base identifies a GID, MAC SA and source port combination that conflicts with a existing static entry, if the previous binded port has a coressponding bit set to 1 in this field, it allows the learning engine to update the GID and MAC to the current source port.	0x7ff

38.11.183 RARP Packet Decoder Options

The Ethernet type used to determine if a packet is a RARP packet.. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries :	1
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Address Space :	1122485

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
16:1	eth	The value to be used to find this packet type.	0x8035
27:17	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
38:28	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

38.11.184 Reserved Destination MAC Address Range

The mac addresses ranges that the packets destination MAC address are compared with and the corresponding actions. A range is matched if the packets MAC address is $\geq$ *startAddr* and the address is $\leq$ *stopAddr*. The table is searched starting from entry 0. When a range is matched the corresponding actions (drop, send to cpu, force egress queue) will be activated. If multiple ranges are matched, any matching range that sets drop will cause a drop. Any match that sets sendToCpu will cause send to CPU (this has priority over drop). When multiple ranges that match has set the forceQueue field then the highest numbered entry will determine the value.

Number of Entries :	4
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	1122651 to 1122666

Field Description

Bits	Field Name	Description	Default Value
47:0	startAddr	The start MAC address of the range. A packets destination MAC address must be equal or greater than this value to match the range.	0x0
95:48	stopAddr	The end MAC address of the range. A packets destination MAC address must be equal or less than this value to match the range.	0x0
96	dropEnable	If the MAC address was within the range the packet shall be dropped and the **Reserved MAC DA Drop** counter incremented.	0x0
97	sendToCpu	If the MAC address was within the range the packet shall be sent to the CPU.	0x0
98	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
101:99	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
103:102	color	Inital color of the packet.	0x0
104	forceColor	If set, the packet shall have a forced color.	0x0
105	mmpValid	If set, this entry contains a valid MMP pointer	0x0
111:106	mmpPtr	Ingress MMP pointer.	0x0
113:112	mmpOrder	Ingress MMP pointer order.	0x0
124:114	enable	Enable the reserved MAC DA check per source port. One bit for each port where bit 0 corresponds to port 0. If a bit is set to one, the reserved MAC DA range is activated for that source port.	0x0

38.11.185 Reserved Source MAC Address Range

The mac addresses ranges that the packets source MAC address are compared with and the corresponding actions. A range is matched if the packets MAC address is $\geq$ *startAddr* and the address is $\leq$ *stopAddr*. The table is searched starting from entry 0. When a range is matched the corresponding actions (drop, send to cpu, force egress queue) will be activated. If multiple ranges are matched, any matching range that sets drop will cause a drop. Any match that sets sendToCpu will cause send to CPU (this has priority over drop). When multiple ranges that match has set the forceQueue then the highest numbered entry will determine the value.

Number of Entries :	4
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	1122635 to 1122650

Field Description

Bits	Field Name	Description	Default Value
47:0	startAddr	The start MAC address of the range. A packets source MAC address must be equal or greater than this value to match the range.	0x0
95:48	stopAddr	The end MAC address of the range. A packets source MAC address must be equal or less than this value to match the range.	0x0
96	dropEnable	If the MAC address was within the range the packet shall be dropped and the **Reserved MAC SA Drop** counter incremented.	0x0
97	sendToCpu	If the MAC address was within the range the packet shall be sent to the CPU.	0x0
98	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
101:99	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
103:102	color	Inital color of the packet.	0x0
104	forceColor	If set, the packet shall have a forced color.	0x0
105	mmpValid	If set, this entry contains a valid MMP pointer	0x0
111:106	mmpPtr	Ingress MMP pointer.	0x0
113:112	mmpOrder	Ingress MMP pointer order.	0x0
124:114	enable	Enable the reserved source MAC check per source port. One bit for each port where bit 0 corresponds to port 0. If a bit is set to one, the reserved source MAC range is activated for that source port.	0x0

38.11.186 Router Egress Queue To VLAN Data

Map from egress queue number to VLAN PCP and CFI/DEI values to be used in router VLAN operations selected by **Next Hop Packet Modifications**.

Number of Entries : 8
Type of Operation : Read/Write
Addressing : Egress Queue
Address Space : 1119562 to 1119569

Field Description

Bits	Field Name	Description	Default Value
0	cfiDei	Map from egress queue to CFI/DEI	0x0
3:1	pcp	Map from egress queue to PCP	0x0

38.11.187 Router MTU Table

An MTU check is done on each routed packet by comparing the IPv4 Total Length field with the **maxIPv4MTU** limit. Correspondingly IPv6 Payload Length field is compared with **maxIPv6MTU**. If the length field exceeds the limit the packet will be sent to the CPU. Each router VRF has a MTU limit for each port.

Number of Entries : 44
Type of Operation : Read/Write
Addressing : destination-port * 4 + VRF
Address Space : 1119475 to 1119518

Field Description

Bits	Field Name	Description	Default Value
15:0	maxIPv4MTU	The maximum MTU allowed for IPv4 packets	0xffff
31:16	maxIPv6MTU	The maximum MTU allowed for IPv6 packets	0xffff

38.11.188 Router Port MAC Address

The incoming packets destination MAC address is compared against all the entries in the table. If there is a match after the macMask has been applied the packet will enter the routing function with the VRF identifier assigned from the matching entry. The table must be configured so that only one match is possible.

Number of Entries : 16
Number of Addresses per Entry : 8
Type of Operation : Read/Write
Addressing : All entries are read out in parallel
Address Space : 1124323 to 1124450

Field Description

Bits	Field Name	Description	Default Value
47:0	macAddress	The base destination MAC address that is used to identify packets to the router.	0x0
95:48	macMask	Each bit says if the bit in the DA MAC shall be compared. 0 = Dont compare bit. 1 = Compare bit.	0x0
106:96	selectMacEntryPortMask	Portmask to select which MAC address to use. One bit per source port. 0 = use macAddress. 1 = use altMacAddress.	0x0
154:107	altMacAddress	The alternative base destination MAC address that is used to identify packets to the router.	0x0
155	valid	If set, this entry is valid for comparison.	0x0
157:156	vrf	The VRF to use for this router	0x0

38.11.189 SCTP Packet Decoder Options

The L4 protocol number which is used to detemine if the packet has a SCTP header. If both the send to cpu option and drop packet option is selected on same source port then the packet will be dropped.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121220

Field Description

Bits	Field Name	Description	Default Value
0	enabled	Is this decoding enabled. 0 = No 1 = Yes	0x1
8:1	l4Proto	The value to be used to find this packet type.	0x84
19:9	drop	If a packet comes in on this source port then drop the packet. 0 = Do not drop this packet. 1 = Drop this packet and update the drop counter.	0x0
30:20	toCpu	If a packet comes in on this source port then send the packet to the CPU port. 0 = Do not sent to CPU. Normal Processing of packet. 1 = Send to CPU , bypass normal packet processing.	0x0

38.11.190 SMON Set Search

If both source port and VLAN ID match one of the entries, the corresponding SMON counter will be updated.

Number of Entries : 8
Type of Operation : Read/Write
Addressing : SMON set number
Address Space : 1120673 to 1120680

Field Description

Bits	Field Name	Description	Default Value
3:0	srcPort	Source port	0x0
15:4	vid	VLAN ID	0x0

38.11.191 SNAP LLC Decoding Options

When a SNAP/LLC packet is received there are some options which allows the packet if not recoginzed to be sent to the CPU. The packet will have a special reason code

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1121217

Field Description

Bits	Field Name	Description	Default Value
15:0	ethSize	What maximum size of packet shall be interpreted as SNAP packet.	0x5dc
26:16	sendToCpu	When a LLC is not equal to (dsap==0xAA and ssap==0xAA and ctrl==0x03) then packet will be sent to cpu. Bit 0 is from port 0, bit 1 is for port 1, etc.	0x0

38.11.192 Second Tunnel Exit Lookup TCAM

The extracted key from packet which is described in the tunnel exit lookup.

Number of Entries :	8
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	1123771 to 1123898

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
128:1	pktKey_mask	Mask for pktKey.	$2^{128}-1$
256:129	pktKey	The extracted key from the packet according to the first lookup.	0x0
264:257	tabKey_mask	Mask for tabKey.	0xff
272:265	tabKey	The key from the first tunnel exit lookup result table.	0x0

38.11.193 Second Tunnel Exit Lookup TCAM Answer

This is the table holding the answer for the **Second Tunnel Exit Lookup TCAM**.

Number of Entries :	8
Number of Addresses per Entry :	2
Type of Operation :	Read/Write
Addressing :	**Second Tunnel Exit Lookup TCAM** hit index
Address Space :	1122451 to 1122466

Field Description

Bits	Field Name	Description	Default Value
7:0	howManyBytesToRemove	How many bytes to remove.	0x0
8	updateEthType	If packet is removed after L2+VLAN headers then update the Ethernet Header Type Field	0x0
24:9	ethType	If packet is removed after L2+VLAN headers then the New Ethernet Type which will overwrite the existing lowest 16 bits after the removal operation.	0x0
25	removeVlan	If packet is removed after L2+VLAN headers then remove the VLAN headers on the incoming packet.	0x0
26	updateL4Protocol	If packet is removed after L3 headers then update the L4 Protocol in IP header.	0x0
34:27	l4Protocol	If packet is removed after L3 headers then this new L4 Protocol will be written.	0x0
36:35	whereToRemove	Where to do the tunnel exit from 0 = At Byte Zero 1 = After L2 and up to two VLAN headers. 2 = After L3 IPv4/IPv6 headers. 3 = Reserved.	0x0
37	dropPkt	Drop the packet.	0x0

Bits	Field Name	Description	Default Value
38	dontExit	Do not do a tunnel exit on this packet.	0x0
39	replaceVid	Replace the assigned VID. This is the VID which shall be used in the VLAN table lookup. This forces a new VID into this packet and bypassing all but the ACL force VID operation.	0x0
51:40	newVid	The new to be used VID.	0x0
56:52	tunnelExitEgressPtr	Tunnel Exit Egress Pointer. Shall point to same tunnel / packet decapsulation operation but setup in egress pipeline in **Egress Tunnel Exit Table**	0x0
57	removeFromCpuTag	If packet came in and had a From CPU Tag does the tunnel exit lookup remove the From CPU Tag or should the design remove this TAG? 0 = Ignore the FROM CPU Tag, the tunnel exit will remove this. 1 = The hardware should remove the From CPU Tag.	0x0

38.11.194 Second Tunnel Exit Miss Action

When a packet misses in the tunnel second lookup table shall this packet be dropped or not?

Number of Entries : 256
Type of Operation : Read/Write
Addressing : The tabIndex result field from the first tunnel exit lookup
Address Space : 1120948 to 1121203

Field Description

Bits	Field Name	Description	Default Value
0	dropIfMiss	If miss in this table then drop packet 0 = No 1 = Yes	0x0

38.11.195 Send to CPU

Configuration of MAC addresses used to redirect packets to CPU.

Number of Entries : 1
Number of Addresses per Entry : 4
Type of Operation : Read/Write
Address Space : 1122667

Field Description

Bits	Field Name	Description	Default Value
10:0	allowBpdu	Send to CPU portmask, bit 0 port 0, bit 1 port 1 etc. If source port bit is set then packets that have the destination MAC address equal to 01:80:C2:00:00:00 are sent to the CPU port.	0x7ff

Bits	Field Name	Description	Default Value
21:11	allowRstBpdu	Send to CPU portmask, bit 0 port 0, bit 1 port 1 etc. If the source port bit is set then packets that have the destination MAC address equal to 01:00:0C:CC:CC:CD are sent to the CPU port.	0x7ff
32:22	uniqueCpuMac	If set then unicast packets can not be switched or routed to the CPU port. Other mechanism for sending to the CPU port are not affected (e.g. ACL's). This also enables detection of a specific MAC address, **cpuMacAddr**, that will be sent to the CPU.	0x0
80:33	cpuMacAddr	Packets with this destination MAC address will be sent to the CPU. Only valid if **uniqueCpuMac** on the source port is set.	0x0

38.11.196 Software Aging Enable

If set, the hardware aging unit will stop the countdown - age out loop, instead software is responsible for counting down and triggering an age out round by **Software Aging Start Latch**.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 319

Field Description

Bits	Field Name	Description	Default Value
0	enable	Enable software aging.	0x0

38.11.197 Software Aging Start Latch

This is used under software aging mode when **Software Aging Enable** is set.

Number of Entries : 1
Type of Operation : Write Only
Address Space : 16764

Field Description

Bits	Field Name	Description	Default Value
0	start	When register is written with start bit set an age out process is started.	0x0

38.11.198 Source Port Default ACL Action

The default ACL action which will be taken on a source port if the **enableDefaultPortAcl** is set and the ACL lookup misses. The action will also be taken if the **forcePortAclAction** is set and then it will override the result from the ACL even if the ACL was hit or not.

Number of Entries :	11
Number of Addresses per Entry :	8
Type of Operation :	Read/Write
Addressing :	Source Port
Address Space :	331108 to 331195

Field Description

Bits	Field Name	Description	Default Value
0	metaDataValid	Is the meta_data field valid.	0x0
16:1	metaData	Meta data for packets going to the CPU.	0x0
17	forceRoute	Shall the packet do a forced Routing? 0 = No. 1 = Yes.	0x0
28:18	nextHopPtr	Which next hop entry shall the forced routing used?	0x0
30:29	vrf	Which vrf shall the forced routing used?	0x0
33:31	nextHopVersion	Which version does this force route table entry have?	0x0
34	inputMirror	If set, input mirroring is enabled for this rule. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the destination Input Mirror port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
38:35	destInputMirror	Destination physical port for input mirroring.	0x0
39	noLearning	If set this packets MAC SA will not be learned.	0x0
40	updateCounter	When set the selected statistics counter will be updated.	0x0
46:41	counter	Which counter in **Ingress Configurable ACL Match Counter** to update.	0x0
47	updateTosExp	Force TOS/EXP update.	0x0
55:48	newTosExp	New TOS/EXP value.	0x0
63:56	tosMask	Mask for TOS value. Setting a bit to one means this bit will be selected from the newTosExp field , while setting this bit to zero means that the bit will be selected from the packets already existing TOS byte bit.	0x0
64	forceVidValid	Override the Ingress VID, see chapter VLAN Processing.	0x0
76:65	forceVid	The new Ingress VID.	0x0
77	updateCfiDei	The CFI/DEI value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
78	newCfiDeiValue	The value to update to.	0x0

Bits	Field Name	Description	Default Value
79	updatePcp	The PCP value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
82:80	newPcpValue	The PCP value to update to.	0x0
83	updateVid	The VID value of the packets outermost VLAN should be updated. 0 = Do not update the value. 1 = Update the value.	0x0
95:84	newVidValue	The VID value to update to.	0x0
96	updateEType	The VLANs TPID type should be updated. 0 = Do not update the TPID. 1 = Update the TPID.	0x0
98:97	newEthType	Select which TPID to use in the outer VLAN header. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register **Egress Ethernet Type for VLAN tag**.	0x0
99	enableUpdateIp	If this entry is hit then update SA or DA IPv4 address in ingress packet processing, this value will be used by the routing function and egress ACL if this is exists, this only works for IPv4. 0 = Disable 1 = Enable	0x0
100	updateSaOrDa	Update the SA or DA IPv4 address. The Destiantion IP address updated will be used in the routing functionality and Egress ACL functionality. If the source IP address is updated then the updated value will be used in the egress ACL keys. 0 = Source IP Address 1 = Destination IP Address	0x0
132:101	newIpValue	Update the SA or DA IPv4 address value.	0x0
133	enableUpdateL4	If this entry is hit then update L4 Source Port or Destination port in ingress packet processing, this value will be used in the Egress ACL. 0 = Disable 1 = Enable	0x0
134	updateL4SpOrDp	Update the source or destination L4 port. 0 = Source L4 Port 1 = Destination L4 Port	0x0
150:135	newL4Value	Update the L4 SP or DP with this value	0x0
151	dropEnable	If set, the packet shall be dropped and the **Ingress Configurable ACL Drop** counter is incremented.	0x0
152	sendToCpu	If set, the packet shall be sent to the CPU port.	0x0
153	forceSendToCpuOrigPkt	If packet shall be sent to CPU then setting this bit will force the packet to be the incoming originial packet. The exception to this is rule is the tunnel exit which will still be carried out..	0x0
154	sendToPort	Send the packet to a specific port. 0 = Disabled. 1 = Send to port configured in destPort.	0x0
158:155	destPort	The port which the packet shall be sent to.	0x0

Bits	Field Name	Description	Default Value
159	ptp	When the packet is sent to the CPU the packet will have the PTP bit in the To CPU Tag set to one. The timestamp in the To CPU Tag will also be set to the timestamp from the incoming packet.	0x0
160	tunnelEntry	Shall all of these packets enter into a tunnel.	0x0
161	tunnelEntryUcMc	Shall this entry point to the **Tunnel Entry Instruction Table** with or without a egress port offset. 0 = Unicast **Tunnel Entry Instruction Table** without offset for each port 1 = Multicast **Tunnel Entry Instruction Table** with offset for each port.	0x0
166:162	tunnelEntryPtr	The tunnel entry which this packet shall enter upon exiting the switch.	0x0
167	tunnelExit	Shall this packet do a tunnel exit. 0 = No 1 = Yes	0x0
172:168	tunnelExitPtr	Pointer to tunnel exit described in **Egress Tunnel Exit Table**.	0x0
173	cancelCryptoOp	Cancel the crypto operation. No crypto operations will be done on this packet. 0 = No. 1 = Yes.	0x0
174	sendToCrypto	Do a crypto operation on this packet. 0 = No. 1 = Yes.	0x0
176:175	cryptoProto	Crypto protocol. 0 = AH 1 = ESP 2 = MACsec 3 = Reserved.	0x0
177	cryptoOp	Crypto operation. 0 = Encrypt 1 = Decrypt	0x0
183:178	secPtr	Pointer into Security Association Database. Valid if sendToCrypto is set.	0x0
187:184	cryptoPort	Crypto modification port. Before the packet is sent for encryption/decryption there can be packet modifications which are based on the egress port, this is the egress port which will be used for these packet modifications.	0x0
188	forceColor	If set, the packet shall have a forced color.	0x0
190:189	color	Initial color of the packet if the forceColor field is set.	0x0
191	mmpValid	If set, this entry contains a valid MMP pointer	0x0
197:192	mmpPtr	Ingress MMP pointer.	0x0
199:198	mmpOrder	Ingress MMP pointer order.	0x0
200	forceQueue	If set, the packet shall have a forced egress queue. Please see Egress Queue Selection Diagram in Figure 23.1	0x0
203:201	eQueue	The egress queue to be assigned if the forceQueue field in this entry is set to 1.	0x0
204	natOpValid	NAT operation pointer is valid.	0x0
217:205	natOpPtr	NAT operation pointer.	0x0

Bits	Field Name	Description	Default Value
220:218	natVersion	NAT Entry Version.	0x0

38.11.199 Source Port Table

This table configures various functions that are dependent on which port the packet enters the switch. A VLAN operation (e.g. push, pop, swap) to be performed can be selected by the **vlanSingleOp** field in **Source Port Table**. For the push and swap operations the information used to create the new VLAN header is controlled by the fields **vidSel**, **cfiDeiSel**, **pcpSel** and **typeSel**. Other configurations are VLAN LUT index, input mirroring, spanning tree state, Ingress VID offset, special VID treatment, multicast learning, min/max number of VLANs and L3 priority selection.

Number of Entries :	11
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	Ingress port
Address Space :	1122511 to 1122554

Field Description

Bits	Field Name	Description	Default Value
0	learningEn	If hardware learning is turned on and this is set to one, the unknown source MAC address from this port will be learned.	0x1
1	dropUnknownDa	If set to one packets with unknown destination MAC address from this port will be dropped.	0x0
2	prioFromL3	If the packet is IP/MPLS and this is set the egress queue will be selected from Layer 3 decoding described in Determine Egress Queue.	0x0
3	colorFromL3	If the packet is IP/MPLS and this bit is set the packet initial color will be selected from Layer 3 decoding.	0x0
4	useAcl0	Use ACL on this source port. 0 = No. No ACL lookup is done 1 = Yes. The aclRule0 pointer selects which fields that are part of the lookup .	0x0
7:5	aclRule0	Pointer into the **Ingress Configurable ACL 0 Rules Setup** table selecting which ACL fields to select to do the ACL lookup with.	0x0
8	useAcl1	Use ACL on this source port. 0 = No. No ACL lookup is done 1 = Yes. The aclRule1 pointer selects which fields that are part of the lookup .	0x0
11:9	aclRule1	Pointer into the **Ingress Configurable ACL 1 Rules Setup** table selecting which ACL fields to select to do the ACL lookup with.	0x0

Bits	Field Name	Description	Default Value
12	useAcl2	Use ACL on this source port. 0 = No. No ACL lookup is done 1 = Yes. The aclRule2 pointer selects which fields that are part of the lookup .	0x0
15:13	aclRule2	Pointer into the **Ingress Configurable ACL 2 Rules Setup** table selecting which ACL fields to select to do the ACL lookup with.	0x0
18:16	vlanSingleOp	The source port VLAN operation to perform on the packet. 0 = No operation. 1 = Swap. 2 = Push. 3 = Pop. 4 = Penultimate pop(remove all VLAN headers).	0x0
20:19	vidSel	Selects which VID to use when building a new VLAN header in a source port push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's **defaultVid** will be used. 0 = From outermost VLAN in the original packet (if any). 1 = From this table entry's **defaultVid**. 2 = From the second VLAN in the original packet (if any).	0x0
22:21	cfiDeiSel	Selects which CFI/DEI to use when building a new VLAN header in a source port push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's **defaultCfiDei** will be used. 0 = From outermost VLAN in the original packet (if any). 1 = From this table entry's **defaultCfiDei**. 2 = From the second VLAN in the original packet (if any).	0x0
24:23	pcpSel	Selects which PCP to use when building a new VLAN header in a source port push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's **defaultPcp** will be used. 0 = From outermost VLAN in the original packet. (if any) 1 = From this table entry's **defaultPcp**. 2 = From the second VLAN in the original packet (if any).	0x0

Bits	Field Name	Description	Default Value
26:25	nrVlansVidOperationIf	This alternative VID operation for port VLAN operation is selected if the following operation is true. 0 = Nr of VLANS in incoming packet is zero. 1 = Nr of VLANS in incoming packet is one. 2 = Nr of VLANS in incoming packet is two. 3 = Reserved and Disabled	0x3
29:27	vlanSingleOpIf	If the field **nrVlansVidOperationIf** is true then this operation will override the default port vid operation **vlanSingleOp**. The source port VLAN operation to perform on the packet. 0 = No operation. 1 = Swap. 2 = Push. 3 = Pop. 4 = Penultimate pop(remove all VLAN headers).	0x0
31:30	vidSelIf	If the field **nrVlansVidOperationIf** is true then this operation will override the default port vid operation **vidSel**. Selects which VID to use when building a new VLAN header in a source port push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's **defaultVidIf** will be used. 0 = From outermost VLAN in the original packet (if any). 1 = From this table entry's **defaultVid**. 2 = From the second VLAN in the original packet (if any).	0x0
33:32	cfiDeiSelIf	If the field **nrVlansVidOperationIf** is true then this operation will override the default port vid operation **cfiDeiSel**. Selects which CFI/DEI to use when building a new VLAN header in a source port push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's **defaultCfiDeiIf** will be used. 0 = From outermost VLAN in the original packet (if any). 1 = From this table entry's **defaultCfiDei**. 2 = From the second VLAN in the original packet (if any).	0x0

Bits	Field Name	Description	Default Value
35:34	pcpSelIf	If the field **nrVlansVidOperationIf** is true then this operation will override the default port vid operation **pcpSel**. Selects which PCP to use when building a new VLAN header in a source port push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's **defaultPcpIf** will be used. 0 = From outermost VLAN in the original packet. (if any) 1 = From this table entry's **defaultPcp**. 2 = From the second VLAN in the original packet (if any).	0x0
37:36	typeSelIf	If the field **nrVlansVidOperationIf** is true then this operation will override the default port vid operation **typeSel**. Selects which TPID to use when building a new VLAN header in a source port push or swap operation. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register **Egress Ethernet Type for VLAN tag**.	0x0
49:38	defaultVidIf	The default VID if **nrVlansVidOperationIf** is true. This is used in source port VLAN operations (see **vidSel**). It is used to assign Ingress VID (see **vlanAssignment**). It is used when creating an internal VLAN header for incoming packets that has no VLAN header.	0x0
50	defaultCfiDeiIf	The default CFI / DEI bit if **nrVlansVidOperationIf** is true. This is used in source port VLAN operations (see **cfiDeiSel**). It is used when creating an internal VLAN header for incoming packets that has no VLAN header.	0x0
53:51	defaultPcpIf	The default PCP bits if **nrVlansVidOperationIf** is true. This is used in source port VLAN operations (see .**pcpSel**). It is used when creating an internal VLAN header for incoming packets that has no VLAN header.	0x0
55:54	typeSel	Selects which TPID to use when building a new VLAN header in a source port push or swap operation. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register **Egress Ethernet Type for VLAN tag**.	0x0

Bits	Field Name	Description	Default Value
57:56	vlanAssignment	Controls how a packets Ingress VID is assigned. If the selected source is from a VLAN header in the incoming packet and the packet doesn't have that header, then this table entry's **defaultVid** will be used. 0 = packet based - the Ingress VID is assigned from the incoming packets outermost VLAN header. 1 = port-based - the packets Ingress VID is assigned from this table entry's **defaultVid** 2 = mixed - if there are two VLANs in the incoming packet, the inner VLAN is chosen. If the incoming packet has only 0 or 1 VLAN, then it will select this table entry's **defaultVid**	0x0
69:58	defaultVid	The default VID. This is used in source port VLAN operations (see **vidSel**). It is used to assign Ingress VID (see **vlanAssignment**). It is used when creating an internal VLAN header for incoming packets that has no VLAN header.	0x0
70	defaultCfiDei	The default CFI / DEI bit. This is used in source port VLAN operations (see **cfiDeiSel**). It is used when creating an internal VLAN header for incoming packets that has no VLAN header.	0x0
73:71	defaultPcp	The default PCP bits. This is used in source port VLAN operations (see .**pcpSel**). It is used when creating an internal VLAN header for incoming packets that has no VLAN header.	0x0
75:74	defaultVidOrder	When a new hit is done in the result in the L2,L3,L4 VID range checks the ingress VID will only be changed if the result has a higher order value.	0x0
77:76	minAllowedVlans	The minimum number of VLAN headers a packet must have to be allowed on this port. Otherwise the packet will be dropped and the **Minimum Allowed VLAN Drop** will be incremented. 0 = All packets are accepted. 1 = 1 or more tags are accepted. 2 = 2 or more tags are accepted. 3 = No packets are accepted.	0x0
79:78	maxAllowedVlans	The maximum number of VLAN headers a packet is allowed to have to enter on this port. Otherwise the packet will be dropped and the **Maximum Allowed VLAN Drop** will be incremented. 0 = Only untagged packets are accepted. 1 = 0 to 1 tags are accepted. 2 = Any number of VLANs are accepted. 3 = Any number of VLANs are accepted.	0x2

Bits	Field Name	Description	Default Value
80	ignoreVlanMembership	By default packets on non-VLAN member source port are dropped before entering the L2 lookup process. Set this field to one to ignore the VLAN membership check on the source port. However L2 lookup can never forward packets to non-VLAN member destinations.	0x0
81	learnMulticastSaMac	If set, the learning engine allows Ethernet multicast source MAC addresses to be learned.	0x0
82	inputMirrorEnabled	If set, input mirroring is enabled on this port. In addition to the normal processing of the packet a copy of the unmodified input packet will be send to the **destInputMirror** port and exit on that port. The copy will be subject to the normal resource limitations in the switch.	0x0
83	imUnderVlanMembership	If set, input mirroring to a destination that not a member of the VLAN will be ignored.	0x0
84	imUnderPortIsolation	If set, input mirroring to a destination that isolated the source port in the **srcPortFilter** will be ignored.	0x0
88:85	destInputMirror	Destination physical port for input mirroring. Only valid if **inputMirrorEnabled** is set.	0x0
91:89	spt	The spanning tree state for this ingress port. The state Disabled implies that spanning tree protocol is not enabled and hence frames will be forwarded on this egress port. 0 = Disabled. 1 = Blocking. 2 = Listening. 3 = Learning. 4 = Forwarding.	0x0
92	enablePriorityTag	An outer VLAN tag with VID matching **priorityVid** will have PCP bits extracted and used to determine output queue but in remaining VLAN processing this tag will not be treated as a VLAN tag. If the packet has an inner VLAN tag this will be treated as an outer VLAN tag in the following VLAN processing. The VID will only be matched in a VLAN header located immediately after DA and SA MAC, i.e. no custom tags allowed. In egress processing the outer VLAN tag will be removed. 0 = Disable comparison. 1 = Enable comparison.	0x0
104:93	priorityVid	The VID used in the outer VLAN tag comparison, see **enablePriorityTag**.	0x0

Bits	Field Name	Description	Default Value
105	enableFromCpuTag	This option can validate the from CPU tag decoding on packets from non-CPU ports. The CPU port is not affected by this field and always decode the from CPU tag.	0x0
106	disableTunnelExit	On this source port are the packets allowed to do a tunnel exit. 0 = Yes 1 = No	0x0
107	firstHitSecondMissSendToCpu	If first tunnel lookup exit hit but second tunnel exit lookup fails then send the packet to the CPU. 0 = Do nothing. 1 = Send the packet to the CPU.	0x0
108	disableRouting	On this source port are the packets allowed to do L3 routing. 0 = No 1 = Yes	0x0
109	natActionTableEnable	Packets coming in on this source port should be checked in the **NAT Action Table**. 0 = No. 1 = Yes.	0x0
110	natPortState	What is this ports NAT status. 0 = Private 1 = Public	0x0
111	enableL2ActionTable	On packets coming in on this port should be checked with the **L2 Action Table** and **L2 Action Table Source Port**. 0 = No, Do not lookup on the **L2 Action Table** and **L2 Action Table Source Port**. 1 = Yes. Do Lookup in the **L2 Action Table** and **L2 Action Table Source Port**	0x0
112	l2ActionTablePortState	What is the source port status bit. Used in table **L2 Action Table** and **L2 Action Table Source Port**.	0x0
113	enableDefaultPortAcl	If enabled then the default acl for this port will be done if the ACL misses in its lookup. 0 = Disabled. No default action taken. 1 = Enabled. If ACL lookup misses then this ACL actil will be carried out instead.	0x0
114	forcePortAclAction	If enabled then the default acl for this port will always be done, if the ACL is hit then the port ACL will overwrite the ACL result. 0 = Disabled. Not action forced. 1 = Enabled. The port ACL overwrites and result from the ingress ACL.	0x0

Bits	Field Name	Description	Default Value
116:115	preLookupAclBits	Pre lookup bits which is used by this port in the pre-lookup tables in the ingress ACLS. Same value is used for all pre ACL lookups which has the source port bits in it.	0x0

38.11.200 Time to Age

Interval period after which FIB entries are aged out.

Number of Entries : 1
Number of Addresses per Entry : 2
Type of Operation : Read/Write
Address Space : 335

Field Description

Bits	Field Name	Description	Default Value
31:0	tickCnt	Number of ticks (see Chapter Tick) between starts of the aging process.	$2^{32}-1$
34:32	tick	Select one of the 5 available ticks. The tick frequencies are configured globaly in the **Core Tick Configuration** register.	0x0

38.11.201 Tunnel Entry MTU Length Check

If a packet is routed and if the tunnel entry updates the IPv4 or IPv6 packet length then this table shall be setup to enable the too long packets to be sent to the CPU for fragmentation.

Number of Entries : 32
Type of Operation : Read/Write
Addressing : Tunnel entry pointer
Address Space : 1118592 to 1118623

Field Description

Bits	Field Name	Description	Default Value
5:0	length	The added length of a IPv4 or IPv6 packet.	0x0

38.11.202 Tunnel Exit Lookup TCAM

The tunnel exit lookup which is performed on the incoming original packet

Number of Entries :	16
Number of Addresses per Entry :	32
Type of Operation :	Read/Write
Addressing :	All entries are read out in parallel
Address Space :	1122683 to 1123194

Field Description

Bits	Field Name	Description	Default Value
0	valid	Is this entry valid. 0 = No 1 = Yes	0x0
48:1	saMac_mask	Mask for saMac.	$2^{48}-1$
96:49	saMac	Source MAC address of the incoming packet	0x0
144:97	daMac_mask	Mask for daMac.	$2^{48}-1$
192:145	daMac	Destination MAC address of the incoming packet	0x0
193	snapLlc_mask	Mask for snapLlc.	0x1
194	snapLlc	This is a SNAP and LLC packet.	0x0
195	hasOuterVid_mask	Mask for hasOuterVid.	0x1
196	hasOuterVid	Does the incoming packet need a outer VLAN	0x0
208:197	outerVid_mask	Mask for outerVid.	0xfff
220:209	outerVid	The outer VLAN header	0x0
221	hasInnerVid_mask	Mask for hasInnerVid.	0x1
222	hasInnerVid	Does the incoming packet need a inner VLAN	0x0
234:223	innerVid_mask	Mask for innerVid.	0xfff
246:235	innerVid	The inner VLAN header	0x0
262:247	ethType_mask	Mask for ethType.	0xffff
278:263	ethType	Ethernet Type for the incoming packet.	0x0
281:279	l3Type_mask	Mask for l3Type.	0x7
284:282	l3Type	The L3 type which shall be matched. If unknown L3 type then this will set to 7. 0 = IPv4 1 = IPv6 2 = One MPLS Label 3 = Two MPLS Labels 4 = Three MPLS labels 5 = Four MPLS labels	0x0
287:285	frag_mask	Mask for frag.	0x7
290:288	frag	IPv4 header fragments bits, if IPv6/MPLS then these bits are set to zero. The bit 0 is the dont-fragment flag (DF bit), bit 1 is the multi-fragment bit (MF bit), bit 2 is if fragment offset is non-zero.	0x0
418:291	ipSa_mask	Mask for ipSa.	$2^{128}-1$
546:419	ipSa	The IP Source Address. IPv4 is located in bits [31:0].	0x0
674:547	ipDa_mask	Mask for ipDa.	$2^{128}-1$

Bits	Field Name	Description	Default Value
802:675	ipDa	The IP Destination or MPLS Address. IPv4 is located in bits [31:0]. First MPLS bits are located at [19:0], second MPLS label [39:20],third MPLS label is [59:40] and forth label is at [79:60].	0x0
804:803	l4Type_mask	Mask for l4Type.	0x3
806:805	l4Type	The L4 type which shall be matched. If not UDP or TCP value 2 will be set in this register. 0 = TCP 1 = UDP 2 = Others 3 = Reserved.	0x0
814:807	l4Protocol_mask	Mask for l4Protocol.	0xff
822:815	l4Protocol	The L4 protocol from the IPv4 or IPv6 headers which shall be matched.	0x0
838:823	l4Sp_mask	Mask for l4Sp.	0xffff
854:839	l4Sp	L4 Source port, if packet is a TCP or UDP, otherwise set to zero.	0x0
870:855	l4Dp_mask	Mask for l4Dp.	0xffff
886:871	l4Dp	L4 destination port, if packet is a TCP or UDP, otherwise set to zero.	0x0
887	fromCpuTag_mask	Mask for fromCpuTag.	0x1
888	fromCpuTag	This packet contains a From CPU Tag. 0 = No. 1 = Yes.	0x0

38.11.203 Tunnel Exit Lookup TCAM Answer

This is the table holding the answer for the **Tunnel Exit Lookup TCAM**.

Number of Entries :	16
Number of Addresses per Entry :	16
Type of Operation :	Read/Write
Addressing :	**Tunnel Exit Lookup TCAM** hit index
Address Space :	16916 to 17171

Field Description

Bits	Field Name	Description	Default Value
10:0	srcPortMask	Which source ports shall this tunnel exit be done on? The portmask which has one bit per source port. 0 = No, do not do tunnel exit 1 = Yes, if second tunnel lookup is a hit then do tunnel exit.	0x0
11	sendToCpu	This packet shall be sent to the CPU. 0 = No. 1 = Yes.	0x0
19:12	secondShift	Second tunnel exit lookup shift to get the data for the second lookup, this value is in number of bytes, this value can at maximum be 142.	0x0

Bits	Field Name	Description	Default Value
20	secondIncludeVlan	Shall second tunnel exit lookup shift be updated according to how many VLANs the packet has? 0 = No 1 = Yes	0x0
21	direct	Use direct value in this table in the Second Tunnel Exit Lookup Table Lookup. 0 = False 1 = True	0x0
149:22	key	Direct Value to use in instead of value from packet.	0x0
277:150	lookupMask	Mask for second tunnel exit lookup data. Before the lookup in the second lookup takes place this value from first lookup/packet data is AND:ed with this value.	0x0
285:278	tabIndex	Index to be used in second tunnel exit dleft lookup. This is used in conjunciton with the key extracted from this table or from packet data.	0x0

38.11.204 VLAN PCP And DEI To Color Mapping Table

Mapping table from VLAN PCP and DEI field to packet initial color.

Number of Entries : 16
Type of Operation : Read/Write
Addressing :

address[0:2] :	PCP
address[3] :	DEI

Address Space : 1120090 to 1120105

Field Description

Bits	Field Name	Description	Default Value
1:0	color	Packet initial color.	0x0

38.11.205 VLAN PCP To Queue Mapping Table

Mapping table from VLAN PCP priority bits to ingress/egress queues.

Number of Entries : 8
Type of Operation : Read/Write
Addressing : Incoming packets VLAN priority bits
Address Space : 1120626 to 1120633

Field Description

Bits	Field Name	Description	Default Value
2:0	pQueue	Egress queue.	0x1

38.11.206 VLAN Table

Defines the VLAN port membership, which GID to use in L2 lookups, the MSPT to use, if routing is allowed and a VLAN operation (e.g. push, pop, swap) to be performed.
The VLAN operation is selected by the **vlanSingleOp** field. For the push and swap operations the information used to create the new VLAN header is controlled by the fields **vidSel**, **cfiDeiSel**, **pcpSel** and **typeSel**.

Number of Entries :	4096
Number of Addresses per Entry :	4
Type of Operation :	Read/Write
Addressing :	The packet's Ingress VID plus offset as defined in **Source Port Table**.
Address Space :	331196 to 347579

Field Description

Bits	Field Name	Description	Default Value
10:0	vlanPortMask	VLAN membership portmask. The packets source port must be a member of the VLAN, otherwise the packet will be dropped and the **VLAN Member Drop** will be incremented. The membership mask will also limit the destination ports for L2 unicast, multicast, broadcast and flooding. If this results in an empty destination port mask then the packet is dropped and the **Empty Mask Drop** will be incremented.	0x7ff
22:11	gid	The packet will be assigned a global identifier that is used during L2 lookup to allow multiple VLANs to share the same L2 tables.	0x0
23	mmpValid	If set, this entry contains a valid MMP pointer	0x0
29:24	mmpPtr	Ingress MMP pointer.	0x0
31:30	mmpOrder	Ingress MMP pointer order.	0x0
35:32	msptPtr	The multiple spanning tree to be used by packets on this VLAN. Points to entries in the **Ingress Multiple Spanning Tree State** and **Egress Multiple Spanning Tree State** tables	0x0
38:36	vlanSingleOp	The ingress VLAN operation to perform on the packet. 0 = No operation. 1 = Swap. 2 = Push. 3 = Pop. 4 = Penultimate Pop(remove all VLANS).	0x0
40:39	vidSel	Selects which VID to use when building a new VLAN header in a push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's **vid** will be used. 0 = From the outermost VLAN in the original packet (if any). 1 = From this table entry's **vid**. 2 = From the second VLAN in the original packet (if any).	0x0

Bits	Field Name	Description	Default Value
42:41	cfiDeiSel	Selects which CFI/DEI to use when building a new VLAN header in a push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's **cfiDei** will be used. 0 = From outermost VLAN in the original packet (if any). 1 = From this table entry's **cfiDei**. 2 = From the second VLAN in the original packet (if any).	0x0
44:43	pcpSel	Selects which PCP to use when building a new VLAN header in a push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's **pcp** will be used. 0 = From outermost VLAN in the original packet. (if any) 1 = From this table entry's **pcp**. 2 = From the second VLAN in the original packet (if any).	0x0
56:45	vid	The VID used in VLAN push or swap operation if selected by **vidSel**.	0x0
59:57	pcp	The PCP used in VLAN push or swap operation if selected by **pcpSel**.	0x0
60	cfiDei	The CFI/DEI used in VLAN push or swap operation if selected by **cfiDeiSel**	0x0
62:61	typeSel	Selects which TPID to use when building a new VLAN header in a push or swap operation. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register **Egress Ethernet Type for VLAN tag** field **typeValue**.	0x0
84:63	nrVlansVidOperationIf	A per source port setting. Port 0 uses bits [1:0], port 2 uses bits [3:2] and so on. If the packet coming in on the source port has this amount of VLANs then this operation will override the VLAN Tables VID operation and all associated data. This operation does take into account what operation the source port VID operation performed on the packet. If a already has 2 VLANs and a push operation is done it will still be counted as a packet with two vlans. If a packet has zero vlans and a pop operation is carried out it will still have zero VLANs. Swap operations does not change the number of VLANs on the packet. 0 = Incoming packet after source port VID op has zero VLANs 1 = Incoming packet after source port VID op has one VLAN 2 = Incoming packet after source port VID op has Two VLANs 3 = Reserved and Disabled	$2^{22}-1$

Bits	Field Name	Description	Default Value
87:85	vlanSingleOpIf	This operation depends on if the **nrVlansVidOperationIf** is done on this port. Then the default operation is overriden with this value. The ingress VLAN operation to perform on the packet. 0 = No operation. 1 = Swap. 2 = Push. 3 = Pop. 4 = Penultimate Pop(remove all VLANS).	0x0
89:88	vidSelIf	This operation depends on if the **nrVlansVidOperationIf** is done on this port. Then the default operation is overriden with this value. Selects which VID to use when building a new VLAN header in a push or swap operation. this table entry's **pcp** will be used. 0 = From outermost VLAN in the original packet. (if any) 1 = From this table entry's **pcp**. 2 = From the second VLAN in the original packet (if any).	0x0
91:90	cfiDeiSelIf	This operation depends on if the **nrVlansVidOperationIf** is done on this port. Selects which CFI/DEI to use when building a new VLAN header in a push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's **cfiDei** will be used. 0 = From outermost VLAN in the original packet (if any). 1 = From this table entry's **cfiDei**. 2 = From the second VLAN in the original packet (if any).	0x0
93:92	pcpSelIf	This operation depends on if the **nrVlansVidOperationIf** is done on this port. Selects which PCP to use when building a new VLAN header in a push or swap operation. If the selected VLAN header doesn't exist in the packet then this table entry's **pcp** will be used. 0 = From outermost VLAN in the original packet. (if any) 1 = From this table entry's **pcp**. 2 = From the second VLAN in the original packet (if any).	0x0
95:94	typeSelIf	This operation depends on if the **nrVlansVidOperationIf** is done on this port. Then the default operation is overriden with this value. Selects which TPID to use when building a new VLAN header in a push or swap operation. 0 = C-VLAN - 0x8100. 1 = S-VLAN - 0x88A8. 2 = User defined VLAN type from register **Egress Ethernet Type for VLAN tag** field **typeValue**.	0x0

Bits	Field Name	Description	Default Value
107:96	vidIf	If this data is used depends on if the **nrVlansVidOperationIf** is done on this port. Then the default operation is overriden with this value. The VID used in VLAN push or swap operation if selected by **vidSel**.	0x0
110:108	pcpIf	If this data is used depends on if the **nrVlansVidOperationIf** is done on this port. Then the default operation is overriden with this value. The PCP used in VLAN push or swap operation if selected by **pcpSel**.	0x0
111	cfiDeiIf	If this data is used depends on if the **nrVlansVidOperationIf** is done on this port. Then the default operation is overriden with this value. The CFI/DEI used in VLAN push or swap operation if selected by **cfiDeiSel**	0x0
112	allowRouting	Allow routing. 0 = The router will not process the packet but L2 processing will be done normally. 1 = Packet will be processed by the router.	0x1
113	sendIpMcToCpu	Send all IPv4 and IPv6 multicast packets to CPU, bypassing L2 processing and L3 routing.	0x0
114	tunnelEntry	For broadcast and flooding packets this tunnel entry will be used on this VLAN.	0x0
119:115	tunnelEntryPtr	The tunnel entry which this packet shall enter upon exiting the vlan. This field points to **Tunnel Entry Instruction Table** using the destination port as a offset from this number.	0x0
120	tunnelExit	Shall this packet do a tunnel exit. 0 = No 1 = Yes	0x0
124:121	tunnelExitPtr	Pointer to tunnel exit described in **Egress Tunnel Exit Table**.	0x0

38.12 MBSC

38.12.1 L2 Broadcast Storm Control Bucket Capacity Configuration

Token Bucket Capacity Configuration for L2 Broadcast Storm Control

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 219 to 229

Field Description

Bits	Field Name	Description	Default Value	
15:0	bucketCapacity	Capacity of the token bucket	Index	Value
			0-1	0x154
			2-10	0x35c

38.12.2 L2 Broadcast Storm Control Bucket Threshold Configuration

Token Bucket Threshold Configuration for L2 Broadcast Storm Control

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 230 to 240

Field Description

Bits	Field Name	Description	Default Value
15:0	threshold	Minimum number of tokens in bucket for the status to be set to accept.	Index: 0-1, Value: 0xaa Index: 2-10, Value: 0x1ae

38.12.3 L2 Broadcast Storm Control Enable

Bitmask to turn L2 Broadcast Storm Control ON/OFF (1/0) for Egress Ports

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 241

Field Description

Bits	Field Name	Description	Default Value
10:0	enable	Bitmask where the index is the Egress Ports	0x0

38.12.4 L2 Broadcast Storm Control Rate Configuration

Token Bucket rate Configuration for L2 Broadcast Storm Control

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 208 to 218

Field Description

Bits	Field Name	Description	Default Value
0	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x1
12:1	tokens	The number of tokens added each tick	Index: 0-1, Value: 0x11 Index: 2-10, Value: 0x2b

Bits	Field Name	Description	Default Value
15:13	tick	Select one of the five available core ticks. The tick frequencies are configured globaly in the core Tick Configuration register.	Index Value 0-1 0x1 2-10 0x2
23:16	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

38.12.5 L2 Flooding Storm Control Bucket Capacity Configuration

Token Bucket Capacity Configuration for L2 Flooding Storm Control

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 287 to 297

Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity	Capacity of the token bucket	Index Value 0-1 0x154 2-10 0x35c

38.12.6 L2 Flooding Storm Control Bucket Threshold Configuration

Token Bucket Threshold Configuration for L2 Flooding Storm Control

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 298 to 308

Field Description

Bits	Field Name	Description	Default Value
15:0	threshold	Minimum number of tokens in bucket for the status to be set to accept.	Index Value 0-1 0xaa 2-10 0x1ae

38.12.7 L2 Flooding Storm Control Enable

Bitmask to turn L2 Flooding Storm Control ON/OFF (1/0) for Egress Ports

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 309

Field Description

Bits	Field Name	Description	Default Value
10:0	enable	Bitmask where the index is the Egress Ports	0x0

38.12.8 L2 Flooding Storm Control Rate Configuration

Token Bucket rate Configuration for L2 Flooding Storm Control

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 276 to 286

Field Description

Bits	Field Name	Description	Default Value
0	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x1
12:1	tokens	The number of tokens added each tick	Index Value 0-1 0x11 2-10 0x2b
15:13	tick	Select one of the five available core ticks. The tick frequencies are configured globaly in the core Tick Configuration register.	Index Value 0-1 0x1 2-10 0x2
23:16	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

38.12.9 L2 Multicast Storm Control Bucket Capacity Configuration

Token Bucket Capacity Configuration for L2 Multicast Storm Control

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 253 to 263

Field Description

Bits	Field Name	Description	Default Value
15:0	bucketCapacity	Capacity of the token bucket	Index Value 0-1 0x154 2-10 0x35c

38.12.10 L2 Multicast Storm Control Bucket Threshold Configuration

Token Bucket Threshold Configuration for L2 Multicast Storm Control

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 264 to 274

Field Description

Bits	Field Name	Description	Default Value
15:0	threshold	Minimum number of tokens in bucket for the status to be set to accept.	Index: Value 0-1: 0xaa 2-10: 0x1ae

38.12.11 L2 Multicast Storm Control Enable

Bitmask to turn L2 Multicast Storm Control ON/OFF (1/0) for Egress Ports

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 275

Field Description

Bits	Field Name	Description	Default Value
10:0	enable	Bitmask where the index is the Egress Ports	0x0

38.12.12 L2 Multicast Storm Control Rate Configuration

Token Bucket rate Configuration for L2 Multicast Storm Control

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 242 to 252

Field Description

Bits	Field Name	Description	Default Value
0	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x1
12:1	tokens	The number of tokens added each tick	Index: Value 0-1: 0x11 2-10: 0x2b

Bits	Field Name	Description	Default Value
15:13	tick	Select one of the five available core ticks. The tick frequencies are configured globaly in the core Tick Configuration register.	Index Value 0-1 0x1 2-10 0x2
23:16	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

38.13 Scheduling

38.13.1 DWRR Bucket Capacity Configuration

Token Bucket Capacity Configuration for DWRR

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 1129921 to 1129932

Field Description

Bits	Field Name	Description	Default Value
17:0	bucketCapacity	Capacity of the byte bucket	$2^{18} - 1$

38.13.2 DWRR Bucket Misc Configuration

Bucket Configurations for DWRR

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Egress Ports
Address Space : 1129933 to 1129944

Field Description

Bits	Field Name	Description	Default Value
4:0	threshold	When the number of bytes in any bucket goes below 2**thr, all buckets mapped to the same prio will be replenished.	0xf
5	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x0
13:6	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode.	0x14

38.13.3 DWRR Weight Configuration

Weight Configuration for DWRR

Number of Entries :	96
Type of Operation :	Read/Write
Addressing :	Egress port * 8 + queue
Address Space :	1129945 to 1130040

Field Description

Bits	Field Name	Description	Default Value
7:0	weight	The relative weight of the queue. A queue with weight 0 is not part of the round robin scheduling but will always be selected last.	0x1

38.13.4 Map Queue to Priority

Map from egress queue to egress priority. Note that this setting must not be changed for any queue with packets queued.

Number of Entries :	12
Type of Operation :	Read/Write
Addressing :	Egress port
Address Space :	1129078 to 1129089

Field Description

Bits	Field Name	Description	Default Value
2:0	prio0	The priority for queue 0	0x0
5:3	prio1	The priority for queue 1	0x1
8:6	prio2	The priority for queue 2	0x2
11:9	prio3	The priority for queue 3	0x3
14:12	prio4	The priority for queue 4	0x4
17:15	prio5	The priority for queue 5	0x5
20:18	prio6	The priority for queue 6	0x6
23:21	prio7	The priority for queue 7	0x7

38.13.5 Output Disable

Bitmask for disabling the egress queues on egress ports.

Number of Entries :	12
Type of Operation :	Read/Write
Addressing :	Egress port
Address Space :	1129909 to 1129920

Field Description

Bits	Field Name	Description	Default Value
0	egressQueue0Disabled	If set, stop scheduling new packets for output from queue 0 on this egress port.	0x0
1	egressQueue1Disabled	If set, stop scheduling new packets for output from queue 1 on this egress port.	0x0
2	egressQueue2Disabled	If set, stop scheduling new packets for output from queue 2 on this egress port.	0x0
3	egressQueue3Disabled	If set, stop scheduling new packets for output from queue 3 on this egress port.	0x0
4	egressQueue4Disabled	If set, stop scheduling new packets for output from queue 4 on this egress port.	0x0
5	egressQueue5Disabled	If set, stop scheduling new packets for output from queue 5 on this egress port.	0x0
6	egressQueue6Disabled	If set, stop scheduling new packets for output from queue 6 on this egress port.	0x0
7	egressQueue7Disabled	If set, stop scheduling new packets for output from queue 7 on this egress port.	0x0

38.14 Shapers

38.14.1 Port Shaper Bucket Capacity Configuration

Token Bucket Capacity Configuration for Port Shaper

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Egress Port
Address Space : 1130637 to 1130648

Field Description

Bits	Field Name	Description	Default Value	
16:0	bucketCapacity	Capacity of the token bucket	Index	Value
			0-1	0x15306
			2-11	0x54ba

38.14.2 Port Shaper Bucket Threshold Configuration

Token Bucket Threshold Configuration for Port Shaper

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Egress Port
Address Space : 1130649 to 1130660

Field Description

Bits	Field Name	Description	Default Value
16:0	threshold	Minimum number of tokens in bucket for the status to be set to accept.	Index: Value 0-1: 0x7102 2-11: 0x1c3e

38.14.3 Port Shaper Enable

Bitmask to turn Port Shaper ON/OFF (1/0) for Egress Port

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1130661

Field Description

Bits	Field Name	Description	Default Value
11:0	enable	Bitmask where the index is the Egress Port	0x0

38.14.4 Port Shaper Rate Configuration

Token Bucket rate Configuration for Port Shaper

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Egress Port
Address Space : 1130625 to 1130636

Field Description

Bits	Field Name	Description	Default Value
0	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x0
13:1	tokens	The number of tokens added each tick	Index: Value 0-1: 0xb4d 2-11: 0x2d3
16:14	tick	Select one of the five available core ticks. The tick frequencies are configured globaly in the core Tick Configuration register.	0x0
24:17	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

38.14.5 Prio Shaper Bucket Capacity Configuration

Token Bucket Capacity Configuration for Prio Shaper

Number of Entries : 96
Type of Operation : Read/Write
Addressing : Egress Port * 8 + Egress Prio
Address Space : 1130429 to 1130524

Field Description

Bits	Field Name	Description	Default Value	
			Index	Value
16:0	bucketCapacity	Capacity of the token bucket	0-15	0x15306
			16-95	0x54ba

38.14.6 Prio Shaper Bucket Threshold Configuration

Token Bucket Threshold Configuration for Prio Shaper

Number of Entries : 96
Type of Operation : Read/Write
Addressing : Egress Port * 8 + Egress Prio
Address Space : 1130525 to 1130620

Field Description

Bits	Field Name	Description	Default Value	
			Index	Value
16:0	threshold	Minimum number of tokens in bucket for the status to be set to accept.	0-15	0x7102
			16-95	0x1c3e

38.14.7 Prio Shaper Enable

Bitmask to turn Prio Shaper ON/OFF (1/0) for Egress Port * 8 + Egress Prio

Number of Entries : 1
Number of Addresses per Entry : 4
Type of Operation : Read/Write
Address Space : 1130621

Field Description

Bits	Field Name	Description	Default Value
95:0	enable	Bitmask where the index is the Egress Port * 8 + Egress Prio	0x0

38.14.8 Prio Shaper Rate Configuration

Token Bucket rate Configuration for Prio Shaper

Number of Entries : 96
Type of Operation : Read/Write
Addressing : Egress Port * 8 + Egress Prio
Address Space : 1130333 to 1130428

Field Description

Bits	Field Name	Description	Default Value
0	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x0
13:1	tokens	The number of tokens added each tick	Index: 0-15 Value: 0xb4d; Index: 16-95 Value: 0x2d3
16:14	tick	Select one of the five available core ticks. The tick frequencies are configured globaly in the core Tick Configuration register.	0x0
24:17	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

38.14.9 Queue Shaper Bucket Capacity Configuration

Token Bucket Capacity Configuration for Queue Shaper

Number of Entries : 96
Type of Operation : Read/Write
Addressing : Egress Port * 8 + Egress Queue
Address Space : 1130137 to 1130232

Field Description

Bits	Field Name	Description	Default Value
16:0	bucketCapacity	Capacity of the token bucket	Index: 0-15 Value: 0x15306; Index: 16-95 Value: 0x54ba

38.14.10 Queue Shaper Bucket Threshold Configuration

Token Bucket Threshold Configuration for Queue Shaper

Number of Entries : 96
Type of Operation : Read/Write
Addressing : Egress Port * 8 + Egress Queue
Address Space : 1130233 to 1130328

Field Description

Bits	Field Name	Description	Default Value
16:0	threshold	Minimum number of tokens in bucket for the status to be set to accept.	Index: 0-15, 16-95 Value: 0x7102, 0x1c3e

38.14.11 Queue Shaper Enable

Bitmask to turn Queue Shaper ON/OFF (1/0) for Egress Port * 8 + Egress Queue

Number of Entries : 1
Number of Addresses per Entry : 4
Type of Operation : Read/Write
Address Space : 1130329

Field Description

Bits	Field Name	Description	Default Value
95:0	enable	Bitmask where the index is the Egress Port * 8 + Egress Queue	0x0

38.14.12 Queue Shaper Rate Configuration

Token Bucket rate Configuration for Queue Shaper

Number of Entries : 96
Type of Operation : Read/Write
Addressing : Egress Port * 8 + Egress Queue
Address Space : 1130041 to 1130136

Field Description

Bits	Field Name	Description	Default Value
0	packetsNotBytes	If set the bucket will count packets, if cleared bytes	0x0
13:1	tokens	The number of tokens added each tick	Index: 0-15, 16-95 Value: 0xb4d, 0x2d3
16:14	tick	Select one of the five available core ticks. The tick frequencies are configured globaly in the core Tick Configuration register.	0x0
24:17	ifgCorrection	Extra bytes per packet to correct for IFG in byte mode. Default is 4 byte FCS plus 20 byte IFG.	0x18

38.15 Shared Buffer Memory

38.15.1 Buffer Free

The number of cells available in the buffer memory for incoming packets.

Number of Entries : 1
Type of Operation : Read Only
Address Space : 1

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Number of free cells.	0x400

38.15.2 Egress Port Depth

Number of packets available in the buffer memory for each egress port.

Number of Entries : 12
Type of Operation : Read Only
Addressing : Egress Port
Address Space : 1129800 to 1129811

Field Description

Bits	Field Name	Description	Default Value
10:0	packets	Number of packet currently queued.	0x0

38.15.3 Egress Queue Depth

Number of packets available in the buffer memory for each egress queue.

Number of Entries : 96
Type of Operation : Read Only
Addressing : Global queue number
Address Space : 1129812 to 1129907

Field Description

Bits	Field Name	Description	Default Value
10:0	packets	Number of packets currently queued.	0x0

38.15.4 Minimum Buffer Free

Minimum number of cells available in the buffer memory

Number of Entries : 1
Type of Operation : Read Only
Address Space : 1129908

Field Description

Bits	Field Name	Description	Default Value
10:0	cells	Number of cells.	0x400

38.15.5 Packet Buffer Status

Queue status of the packet buffer

Number of Entries : 1
Type of Operation : Read Only
Address Space : 1129075

Field Description

Bits	Field Name	Description	Default Value
11:0	empty	Empty flags for the egress ports	0xfff

38.16 Statistics: ACL

38.16.1 Egress Configurable ACL Match Counter

Number of packets hit in entries from Egress configurable ACL lookup.

Number of Entries : 64
Type of Operation : Read/Write
Addressing : Index from result of Egress configurable ACL.
Address Space : 1128296 to 1128359

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.16.2 Ingress Configurable ACL Match Counter

Number of packets hit in entries from Ingress configurable ACL lookup.

Number of Entries :	64
Type of Operation :	Read/Write
Addressing :	Index from result of Ingress configurable ACL.
Address Space :	1126180 to 1126243

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.17 Statistics: Debug

38.17.1 Debug EPP Counter

Number of packets hit in entries from Debug points in EPP.

Number of Entries :	15
Type of Operation :	Read/Write
Addressing :	Epp Debug Counter table.
Address Space :	1198588 to 1198602

Field Description

Bits	Field Name	Description	Default Value
15:0	packets	Number of packets.	0x0

38.17.2 Debug IPP Counter

Number of packets hit in entries from Debug points in IPP.

Number of Entries :	23
Type of Operation :	Read/Write
Addressing :	Ipp Debug Counter table.
Address Space :	1128470 to 1128492

Field Description

Bits	Field Name	Description	Default Value
15:0	packets	Number of packets.	0x0

38.17.3 EPP PM Drop

Number of drops due to FIFO overflows in EPP PM.
In Figure 32.1, **epmOverflow** with process sequence **22** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1130815

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.17.4 IPP PM Drop

Number of drops due to FIFO overflows in IPP PM.
In Figure 32.1, **ipmOverflow** with process sequence **12** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16864

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.17.5 PS Error Counter

Number of errors occured in the PS-converter.
In Figure 32.1, **psError** with process sequence **25** represents the internal location of this counter.

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 1198642 to 1198653

Field Description

Bits	Field Name	Description	Default Value
7:0	underrun	Number of packets which have empty cycles caused by the internal PS-converter but not the external halt during packet transmissions.	0x0
15:8	overflow	Number of FIFO overflows in the PS-converter. This error will cause packet corruptions.	0x0

38.17.6 SP Overflow Drop

Number of packets dropped due to: FIFO overflow in the SP-converter.
In Figure 32.1, **spOverflow** with process sequence **5** represents the internal location of this counter.

Number of Entries : 12
Type of Operation : Read Only
Addressing : Ingress port
Address Space : 16816 to 16827

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets on this ingress port.	0x0

38.18 Statistics: EPP Egress Port Drop

38.18.1 Egress Cell Size Drop

Egress processing exceeds cell size.
In Figure 32.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 1130803 to 1130814

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.18.2 Egress Functional Control Drops

Number of packets dropped due to egress functional control.
In Figure 32.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 1130791 to 1130802

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.18.3 Egress Port Disabled Drop

Number of packets dropped due to egress port disabled.
In Figure 32.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries :	12
Type of Operation :	Read/Write
Addressing :	Egress port
Address Space :	1130731 to 1130742

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.18.4 Egress Port Filtering Drop

Number of packets dropped due to egress port filtering.
In Figure 32.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries :	12
Type of Operation :	Read/Write
Addressing :	Egress port
Address Space :	1130743 to 1130754

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.18.5 Egress Table Not In Sync Drop

Egress table entry was not in sync with ingress table ID.
In Figure 32.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries :	12
Type of Operation :	Read/Write
Addressing :	Egress port
Address Space :	1130755 to 1130766

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.18.6 Minimum and Maximum Packet Size Drops

The number of packets dropped due to larger than 16,387or smalled that 60and being sent to the crypto block.
In Figure 32.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 1130779 to 1130790

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.18.7 Tunnel Exit Too Small Packet Modification To Small Drop

The packet modification after the tunnel exit resulted in a packet size that was less than zero.
In Figure 32.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 1130767 to 1130778

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.18.8 Unknown Egress Drop

Number of packets dropped during egress packet processing due to unknown reasons. Internal error caused by packet drop with an invalid Drop ID.
In Figure 32.1, **epppDrop** with process sequence **19** represents the internal location of this counter.

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 1130719 to 1130730

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.19 Statistics: IPP Egress Port Drop

38.19.1 Egress Spanning Tree Drop

Number of packets dropped due to egress spanning tree check configured in **Egress Spanning Tree State** and **Egress Multiple Spanning Tree State**
In Figure 32.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	11
Type of Operation :	Read/Write
Addressing :	Egress Port (not aggregated)
Address Space :	1128371 to 1128381

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.19.2 Ingress-Egress Packet Filtering Drop

Number of packets dropped due to ingress-egress packet filtering configured in **Ingress Egress Port Packet Type Filter**.
In Figure 32.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	11
Type of Operation :	Read/Write
Addressing :	Egress Port (not aggregated)
Address Space :	1128393 to 1128403

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.19.3 L2 Action Table Per Port Drop

Number of packets dropped due to L2 Action Table per egress port drop configured in **L2 Action Table Drop**.
In Figure 32.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries :	11
Type of Operation :	Read/Write
Addressing :	Egress Port (not aggregated)
Address Space :	1128404 to 1128414

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.19.4 MBSC Drop

Number of packets dropped due to MBSC. When the egress port exceeds the multicast/broadcast traffic limits any multicast/broadcast packets will be dropped.
In Figure 32.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Egress Port (not aggregated)
Address Space : 1128382 to 1128392

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.19.5 Queue Off Drop

Number of packets dropped due to the queue being turned off.
In Figure 32.1, **preEppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Egress Port (not aggregated)
Address Space : 1128360 to 1128370

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20 Statistics: IPP Ingress Port Drop

38.20.1 AH Decoder Drop

Number of packets dropped due to setting in register **AH Header Packet Decoder Options**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16899

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.2 ARP Decoder Drop

Number of packets dropped due to setting in register **ARP Packet Decoder Options**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16892

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.3 BOOTP and DHCP Decoder Drop

Number of packets dropped due to setting in register **BOOTP and DHCP Packet Decoder Options**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16902

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.4 CAPWAP Decoder Drop

Number of packets dropped due to setting in register **CAPWAP Packet Decoder Options**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16903

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.5 Crypto Drops

Number of packets dropped due to crypto block says drop.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16907

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.6 DNS Decoder Drop

Number of packets dropped due to setting in register **DNS Packet Decoder Options**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16901

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.7 ESP Decoder Drop

Number of packets dropped due to setting in register **ESP Header Packet Decoder Options**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16900

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.8 Egress Configurable ACL Drop

Number of packets dropped due to matching an Egress Configurable ACL with drop.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16891

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.9 Empty Mask Drop

Number of packets dropped due to an empty destination port mask.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16867

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.10 Expired TTL Drop

Number of packets dropped due to expired TTL.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16880

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.11 GRE Decoder Drop

Number of packets dropped due to setting in register **GRE Packet Decoder Options**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16905

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.12 IEEE 802.1X and EAPOL Decoder Drop

Number of packets dropped due to setting in register **IEEE 802.1X and EAPOL Packet Decoder Options**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16896

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.13 IKE Decoder Drop

Number of packets dropped due to setting in register **IKE Packet Decoder Options**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16904

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.14 IP Checksum Drop

Number of packets dropped due to incorrect IP checksum.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16882

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.15 Ingress Configurable ACL Drop

Number of packets dropped due to matching an Ingress Configurable ACL with drop.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16890

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.16 Ingress Functional Control Drops

Number of packets dropped due to ingress functional control packet drops.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16913

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.17 Ingress Packet Filtering Drop

Number of packets dropped due to ingress port packet type filtering as configured in **Ingress Port Packet Type Filter**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16873

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.18 Ingress Spanning Tree Drop: Blocking

Number of packets dropped due to that a ports's ingress spanning tree protocol state was **Blocking** or that port and packet VLAN's ingress multiple spanning tree instance state was **Discarding**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16870

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.19 Ingress Spanning Tree Drop: Learning

Number of packets dropped due to that a port's ingress spanning tree protocol state was **Learning** or that port and packet VLAN's ingress multiple spanning tree instance state was **Learning**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16869

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.20 Ingress Spanning Tree Drop: Listen

Number of packets dropped due to that a port's ingress spanning tree protocol state was **Listening**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16868

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.21 Ingress Table Not In Sync Drop

Ingress tables entry was not in sync with ingress table ID.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16872

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.22 Invalid Routing Protocol Drop

Number of packets dropped due to invalid routing protocol. This occurs when a packet enters the router port but the protocol type is not allowed to be routed as configured in **Ingress Router Table**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16879

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.23 L2 Action Table Drop

Number of packets dropped due to the **L2 Action Table** says drop all instances.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16910

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.24 L2 Action Table Port Move Drop

Number of packets dropped due to the **L2 Action Table** says drop due to port move packet.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16911

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.25 L2 Action Table Special Packet Type Drop

Number of packets dropped due to the **Allow Special Frame Check For L2 Action Table** dit not allow a certain packet/frame type.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16909

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.26 L2 Decoder Packet Drop

Number of packets dropped due to L2 decoding error. This error occurs when the L2 decoding can not complete due to packet being shorter than the required headers.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16887

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.27 L2 IEEE 1588 Decoder Drop

Number of packets dropped due to setting in register **IEEE 1588 L4 Packet Decoder Options**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16894

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.28 L2 Lookup Drop

Number of packets dropped in the L2 destination port lookup process. Either due to a drop flag in an **L2 Destination Table** entry, or due to destination port not being member of the VLAN or due to not allowing destination port being the same as the source port.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16871

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.29 L2 Reserved Multicast Address Drop

Number of packets dropped due to the L2 Reserved Multicast Addresses on counter 0
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16889

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.30 L3 Decoder Packet Drop

Number of packets dropped due to L3 decoding error. This error occurs when the L3/L4 decoding can not complete due to packet being shorter than the required headers.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16888

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.31 L3 Lookup Drop

Number of packets dropped due to a drop flag in **L3 Routing Default** or **Next Hop Table**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16881

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.32 L4 IEEE 1588 Decoder Drop

Number of packets dropped due to setting in register **IEEE 1588 L4 Packet Decoder Options**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16895

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.33 LACP Decoder Drop

Number of packets dropped due to setting in register **LACP Packet Decoder Options**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16898

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.34 Learning Packet Drop

Number of learning packets dropped. After learning information is extracted all learning packets are dropped.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16886

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.35 MACsec Drops

Number of packets dropped due to MACsec drop.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16908

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.36 Maximum Allowed VLAN Drop

Number of packets dropped due to too many VLAN tags. Packets are dropped if number of VLANS is above the limit setup in the **Source Port Table**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16878

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.37 Minimum Allowed VLAN Drop

Number of packets dropped due to insufficient VLAN tags. Packets are dropped if number of VLANS is below the limit setup in the **Source Port Table**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16877

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.38 NAT Action Table Drop

Number of packets dropped due to the **NAT Action Table**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16906

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.39 RARP Decoder Drop

Number of packets dropped due to setting in register **RARP Packet Decoder Options**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16893

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.40 Reserved MAC DA Drop

Number of packets dropped due to the packets destination MAC address match a **Reserved Destination MAC Address Range** that is configured to be dropped.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16874

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.41 Reserved MAC SA Drop

Number of packets dropped due to the packets source MAC address match a **Reserved Source MAC Address Range** that is configured to be dropped.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16875

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.42 SCTP Decoder Drop

Number of packets dropped due to setting in register **SCTP Packet Decoder Options**.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16897

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.43 Second Tunnel Exit Drop

Number of packets dropped due to second tunnel exit lookup says drop packet.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16883

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.44 Source Port Default ACL Action Drop

Number of packets dropped due to the table **Source Port Default ACL Action** says drop.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16912

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.45 Tunnel Exit Miss Action Drop

Number of packets dropped due to second tunnel exit lookup was a miss while the tunnel exit table **Second Tunnel Exit Miss Action** says that the second tunnel table must be a hit.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16884

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.46 Tunnel Exit Too Small Packet Modification Drop

The packet modification after the tunnel exit resulted in a packet size that was less than zero.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16885

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.47 Unknown Ingress Drop

Number of packets dropped during ingress packet processing due to unknown reasons. Internal error caused by packet drop with an invalid Drop ID.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16866

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.20.48 VLAN Member Drop

Number of packets dropped due to the packets source port notbeing part of the packets VLAN membership.
In Figure 32.1, **ipppDrop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16876

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.21 Statistics: IPP Ingress Port Receive

38.21.1 IP Multicast ACL Drop Counter

Number of IP multicast packets received and hit in ACL drop rules. IP multicast packets are counted for IPv4 packets with destination MAC in range 01:00:5e:00:00:00 to 01:00:5e:7f:ff:ff or IPv6 packets with destination MAC matches 33:33:xx:xx:xx:xx.
In Figure 32.1, **ip** with process sequence **11** represents the internal location of this counter.

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Ingress port
Address Space : 1128459 to 1128469

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.21.2 IP Multicast Received Counter

Number of IP multicast packets received on ingress. IP multicast packets are counted for IPv4 packets with destination MAC in range 01:00:5e:00:00:00 to 01:00:5e:7f:ff:ff or IPv6 packets with destination MAC matches 33:33:xx:xx:xx:xx.
In Figure 32.1, **ip** with process sequence **11** represents the internal location of this counter.

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Ingress port
Address Space : 1128426 to 1128436

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.21.3 IP Multicast Routed Counter

Number of IP multicast packets received and routed on ingress. IP multicast packets are counted for IPv4 packets with destination MAC in range 01:00:5e:00:00:00 to 01:00:5e:7f:ff:ff or IPv6 packets with destination MAC matches 33:33:xx:xx:xx:xx.
In Figure 32.1, **ip** with process sequence **11** represents the internal location of this counter.

Number of Entries : 11
Type of Operation : Read/Write
Addressing : Ingress port
Address Space : 1128448 to 1128458

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.21.4 IP Unicast Received Counter

Number of IP unicast packets received on ingress. Any IP packet with destination MAC not in IP multicast range (01:00:5e:00:00:00 to 01:00:5e:7f:ff:ff for IPv4 and 33:33:xx:xx:xx:xx for IPv6) are counted as IP unicast packets.
In Figure 32.1, **ip** with process sequence **11** represents the internal location of this counter.

Number of Entries :	11
Type of Operation :	Read/Write
Addressing :	Ingress port
Address Space :	1128415 to 1128425

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.21.5 IP Unicast Routed Counter

Number of IP unicast packets received and routed on ingress. Any IP packet with destination MAC not in IP multicast range (01:00:5e:00:00:00 to 01:00:5e:7f:ff:ff for IPv4 and 33:33:xx:xx:xx:xx for IPv6) are counted as IP unicast packets.
In Figure 32.1, **ip** with process sequence **11** represents the internal location of this counter.

Number of Entries :	11
Type of Operation :	Read/Write
Addressing :	Ingress port
Address Space :	1128437 to 1128447

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.22 Statistics: Misc

38.22.1 Buffer Overflow Drop

Counter for the number of packets dropped due to the shared buffer memory being full.
In Figure 32.1, **bmOverflow** with process sequence **16** represents the internal location of this counter.

Number of Entries :	1
Type of Operation :	Read/Write
Address Space :	1129076

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.22.2 Drain Port Drop

Number of packets dropped due to the port is drained.
In Figure 32.1, **drain** with process sequence **21** represents the internal location of this counter.

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Egress port
Address Space : 1130707 to 1130718

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.22.3 Egress Resource Manager Drop

Number of packets dropped by the egress resource manager.
In Figure 32.1, **erm** with process sequence **15** represents the internal location of this counter.

Number of Entries : 12
Type of Operation : Read/Write
Addressing : Egress Port
Address Space : 1129063 to 1129074

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.22.4 Flow Classification And Metering Drop

Number of packets dropped due to flow classification and metering.
In Figure 32.1, **mmp** with process sequence **14** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1128493

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.22.5 IPP Empty Destination Drop

Number of drops due to the determined destination is cleared during post-ingress packet processing and causing no cell to be enqueued in the buffer memory. This happens on single cell packet with end-of-packet drop actions.
In Figure 32.1, **eopDrop** with process sequence **14** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16865

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.22.6 Ingress Resource Manager Drop

Counter for the number of packets dropped due to exeeding thresholds set up in the ingress resource manager.
In Figure 32.1, **irm** with process sequence **16** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1129077

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets.	0x0

38.22.7 MAC RX Broken Packets

Number of broken packets dropped (packets with last=1 and valid_bytes=0).
In Figure 32.1, **macBrokenPkt** with process sequence **3** represents the internal location of this counter.

Number of Entries : 12
Type of Operation : Read Only (unreliable)
Addressing : Ingress Port
Address Space : 84 to 95

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.22.8 MAC RX Long Packet Drop

Number of packets dropped due to length above **MAC RX Maximum Packet Length**.
In Figure 32.1, **macRxMax** with process sequence **4** represents the internal location of this counter.

Number of Entries : 12
Type of Operation : Read Only (unreliable)
Addressing : Ingress Port
Address Space : 108 to 119

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.22.9 MAC RX Short Packet Drop

Number of packets dropped due to length below 60 bytes.
In Figure 32.1, **macRxMin** with process sequence **4** represents the internal location of this counter.

Number of Entries : 12
Type of Operation : Read Only (unreliable)
Addressing : Ingress Port
Address Space : 96 to 107

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.22.10 Re-queue Overflow Drop

Counter for the number of packets dropped due to a FIFO overflow in re-queue.
In Figure 32.1, **rqOverflow** with process sequence **24** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1129090

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of dropped packets	0x0

38.23 Statistics: NAT

38.23.1 Egress NAT Hit Status

Status bit is set if there was a hit in the **Egress NAT Operation**.
In Figure 32.1, **nat** with process sequence **19** represents the internal location of this counter.

Number of Entries :	8192
Type of Operation :	Read/Write
Addressing :	Egress NAT pointer + Egress Port
Address Space :	1190396 to 1198587

Field Description

Bits	Field Name	Description	Default Value
0	hit	If set, the corresponding entry in the **Egress NAT Operation** is hit.	0x0

38.23.2 Ingress NAT Hit Status

Status bit is set if there was a hit in the **Ingress NAT Operation**.
In Figure 32.1, **nat** with process sequence **19** represents the internal location of this counter.

Number of Entries :	8192
Type of Operation :	Read/Write
Addressing :	Ingress NAT pointer + Egress Port
Address Space :	1182204 to 1190395

Field Description

Bits	Field Name	Description	Default Value
0	hit	If set, the corresponding entry in the **Ingress NAT Operation** is hit.	0x0

38.24 Statistics: Packet Datapath

38.24.1 EPP Packet Head Counter

Number of packet first cells through the Egress Packet Process module.
In Figure 32.1, **eppTxPkt** with process sequence **24** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1130816

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packet headers.	0x0

38.24.2 EPP Packet Tail Counter

Number of packet last cells through the Egress Packet Process module.
In Figure 32.1, **eppTxPkt** with process sequence **24** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1130817

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packet tails.	0x0

38.24.3 IPP Packet Head Counter

Number of packet first cells through the Ingress Packet Process module.
In Figure 32.1, **ippTxPkt** with process sequence **13** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16914

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packet headers.	0x0

38.24.4 IPP Packet Tail Counter

Number of packet last cells through the Ingress Packet Process module.
In Figure 32.1, **ippTxPkt** with process sequence **13** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 16915

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packet tails.	0x0

38.24.5 MAC Interface Counters For RX

Counters for the interface protocol checkers. The counters wrap.
In Figure 32.1, **rxIf** with process sequence **1** represents the internal location of this counter.

Number of Entries : 12
Number of Addresses per Entry : 2
Type of Operation : Read Only (unreliable)
Addressing : Ingress Port
Address Space : 48 to 71

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Correct packets completed	0x0
63:32	error	Bus protocol errors.	0x0

38.24.6 MAC Interface Counters For TX

Counters for the interface protocol checkers. The counters wrap.
In Figure 32.1, **txIf** with process sequence **28** represents the internal location of this counter.

Number of Entries : 12
Number of Addresses per Entry : 4
Type of Operation : Read Only
Addressing : Egress Port
Address Space : 120 to 167

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Correct packets completed	0x0
63:32	error	Bus protocol errors.	0x0
95:64	halt	Halt errors. Incremented if first, last or valid_bytes is non-zero when halt is high.	0x0

38.24.7 PB Packet Head Counter

Number of packet first cells through the Shared Buffer Memory module.
In Figure 32.1, **pbTxPkt** with process sequence **18** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1130704

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packet headers.	0x0

38.24.8 PB Packet Tail Counter

Number of packet last cells through the Shared Buffer Memory module.
In Figure 32.1, **pbTxPkt** with process sequence **18** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1130705

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packet tails.	0x0

38.24.9 PS Packet Head Counter

Number of packet first cells through the Parallel to Serial module.
In Figure 32.1, **psTxPkt** with process sequence **25** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1198640

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packet headers.	0x0

38.24.10 PS Packet Tail Counter

Number of packet last cells through the Parallel to Serial module.
In Figure 32.1, **psTxPkt** with process sequence **25** represents the internal location of this counter.

Number of Entries : 1
Type of Operation : Read/Write
Address Space : 1198641

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packet tails.	0x0

38.25 Statistics: Routing

38.25.1 Next Hop Hit Status

Status bit is set if a packet was routed using the corresponding entry in the **Next Hop Table**.
In Figure 32.1, **nextHop** with process sequence **11** represents the internal location of this counter.

Number of Entries : 2048
Type of Operation : Read/Write
Addressing : Next Hop
Address Space : 1126248 to 1128295

Field Description

Bits	Field Name	Description	Default Value
0	ipv4	The next hop entry was hit with an IPv4 packet.	0x0
1	ipv6	The next hop entry was hit with an IPv6 packet.	0x0
2	mpls	The next hop entry was hit with an MPLS packet.	0x0

38.25.2 Received Packets on Ingress VRF

Number of packets enter a VRF on ingress.
In Figure 32.1, **vrfIn** with process sequence **11** represents the internal location of this counter.

Number of Entries : 4
Type of Operation : Read/Write
Addressing : vrf
Address Space : 1126244 to 1126247

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.25.3 Transmitted Packets on Egress VRF

Number of packets leave a VRF on egress.
In Figure 32.1, **vrfOut** with process sequence **19** represents the internal location of this counter.

Number of Entries :	4
Type of Operation :	Read/Write
Addressing :	vrf
Address Space :	1182200 to 1182203

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.26 Statistics: SMON

38.26.1 SMON Set 0 Byte Counter

Number of bytes counted in SMON Set 0.
In Figure 32.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space :	1126116 to 1126123

Field Description

Bits	Field Name	Description	Default Value
31:0	bytes	Number of bytes.	0x0

38.26.2 SMON Set 0 Packet Counter

Number of packets counted in SMON Set 0.
In Figure 32.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space :	1126052 to 1126059

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.26.3 SMON Set 1 Byte Counter

Number of bytes counted in SMON Set 1.
In Figure 32.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space :	1126124 to 1126131

Field Description

Bits	Field Name	Description	Default Value
31:0	bytes	Number of bytes.	0x0

38.26.4 SMON Set 1 Packet Counter

Number of packets counted in SMON Set 1.
In Figure 32.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space :	1126060 to 1126067

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.26.5 SMON Set 2 Byte Counter

Number of bytes counted in SMON Set 2.
In Figure 32.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space :	1126132 to 1126139

Field Description

Bits	Field Name	Description	Default Value
31:0	bytes	Number of bytes.	0x0

38.26.6 SMON Set 2 Packet Counter

Number of packets counted in SMON Set 2.
In Figure 32.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space :	1126068 to 1126075

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.26.7 SMON Set 3 Byte Counter

Number of bytes counted in SMON Set 3.
In Figure 32.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space :	1126140 to 1126147

Field Description

Bits	Field Name	Description	Default Value
31:0	bytes	Number of bytes.	0x0

38.26.8 SMON Set 3 Packet Counter

Number of packets counted in SMON Set 3.
In Figure 32.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space :	1126076 to 1126083

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.26.9 SMON Set 4 Byte Counter

Number of bytes counted in SMON Set 4.
In Figure 32.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space :	1126148 to 1126155

Field Description

Bits	Field Name	Description	Default Value
31:0	bytes	Number of bytes.	0x0

38.26.10 SMON Set 4 Packet Counter

Number of packets counted in SMON Set 4.
In Figure 32.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space :	1126084 to 1126091

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.26.11 SMON Set 5 Byte Counter

Number of bytes counted in SMON Set 5.
In Figure 32.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space :	1126156 to 1126163

Field Description

Bits	Field Name	Description	Default Value
31:0	bytes	Number of bytes.	0x0

38.26.12 SMON Set 5 Packet Counter

Number of packets counted in SMON Set 5.
In Figure 32.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space :	1126092 to 1126099

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.26.13 SMON Set 6 Byte Counter

Number of bytes counted in SMON Set 6.
In Figure 32.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space :	1126164 to 1126171

Field Description

Bits	Field Name	Description	Default Value
31:0	bytes	Number of bytes.	0x0

38.26.14 SMON Set 6 Packet Counter

Number of packets counted in SMON Set 6.
In Figure 32.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space :	1126100 to 1126107

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0

38.26.15 SMON Set 7 Byte Counter

Number of bytes counted in SMON Set 7.
In Figure 32.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space :	1126172 to 1126179

Field Description

Bits	Field Name	Description	Default Value
31:0	bytes	Number of bytes.	0x0

38.26.16 SMON Set 7 Packet Counter

Number of packets counted in SMON Set 7.
In Figure 32.1, **smon** with process sequence **11** represents the internal location of this counter.

Number of Entries :	8
Type of Operation :	Read/Write
Addressing :	VLAN PCP
Address Space :	1126108 to 1126115

Field Description

Bits	Field Name	Description	Default Value
31:0	packets	Number of packets.	0x0