June 23, 2010

Data Path Acceleration Architecture (DPAA) Deep Dive

FTF-NET-F0446

Sam Siu

Systems and Applications Engineer

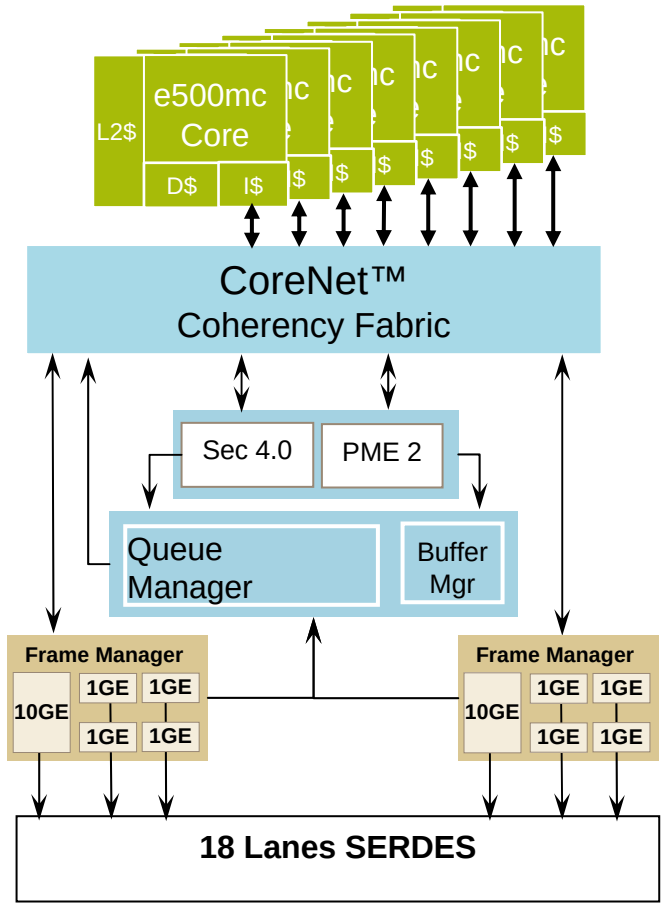
freescale™ semiconductor

Agenda

- DPAA
- BMAN Deep Dive
- FMAN Deep Dive
- QMAN Deep Dive
- SEC Deep Dive
- PME Deep Dive
- Conclusion

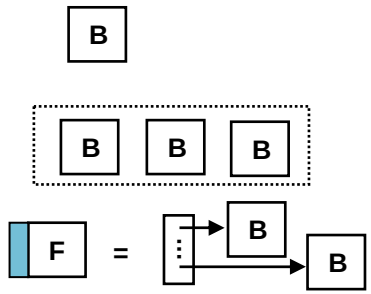
DPAA Components

- "Infrastructure" components
  - Queue Manager (QMan)
  - Buffer Manager (BMan)
- Network I/O
  - Frame Manager (FMan)
- Hardware accelerators
  - SEC – cryptographic accelerator
  - PME – Pattern matching engine
- Cores
- CoreNet is not part of the DPAA but it provides the interconnect between the cores and the DPAA infrastructure as well as access to memory (DRAM)



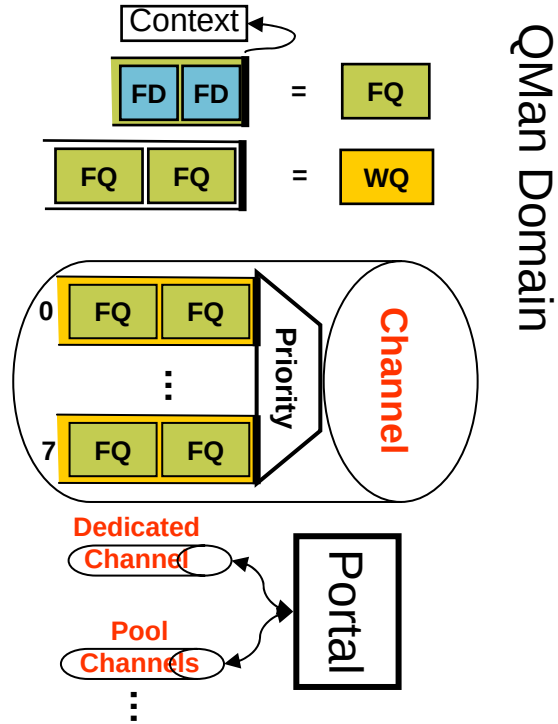
DPAA Terminology

- **Buffer Descriptor:**
- **Buffer: Unit of contiguous memory, allocated by software.**
- **Frame: Buffer(s) that hold a data element (generally a packet)**
  - **Frames can be single buffers or multiple buffers (scatter/gather lists)**
  - **A "simple frame" has one delimited data element**
  - **A "Compound frame" has more than one buffer**



- **Frame Descriptor: Proxy structure used to represent frames.**
- **Frame Queue: FIFO of related Frames Descriptor.**

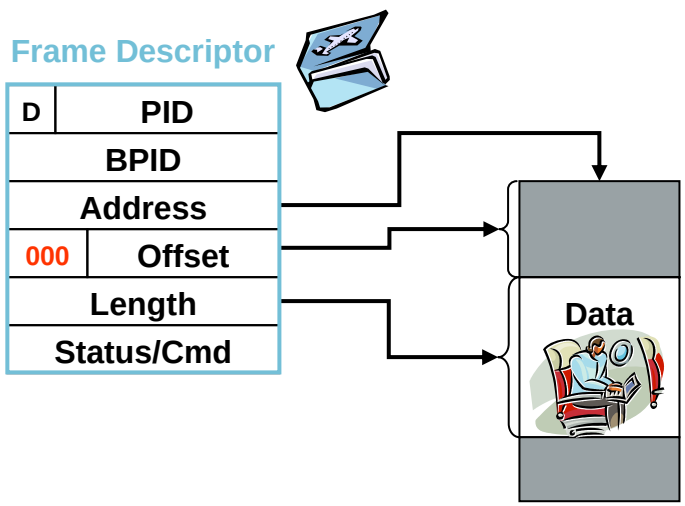
  The basic queuing structure supported by QMan
- **Frame Queue Descriptor: Structure used to manage Frame Queues**
- **Work Queue: FIFO of Frame Queues**
- **Channel: Set of 8 prioritized Work Queues, with HW scheduling**
  - **Dedicated Channel: supplies FQs to a single consumer.**
  - **Pool Channel: can be shared by multiple consumers**
- **Portal: HW interface used to access QMan facilities (e.g. Enqueue or Dequeue) for possibly multiple channels**



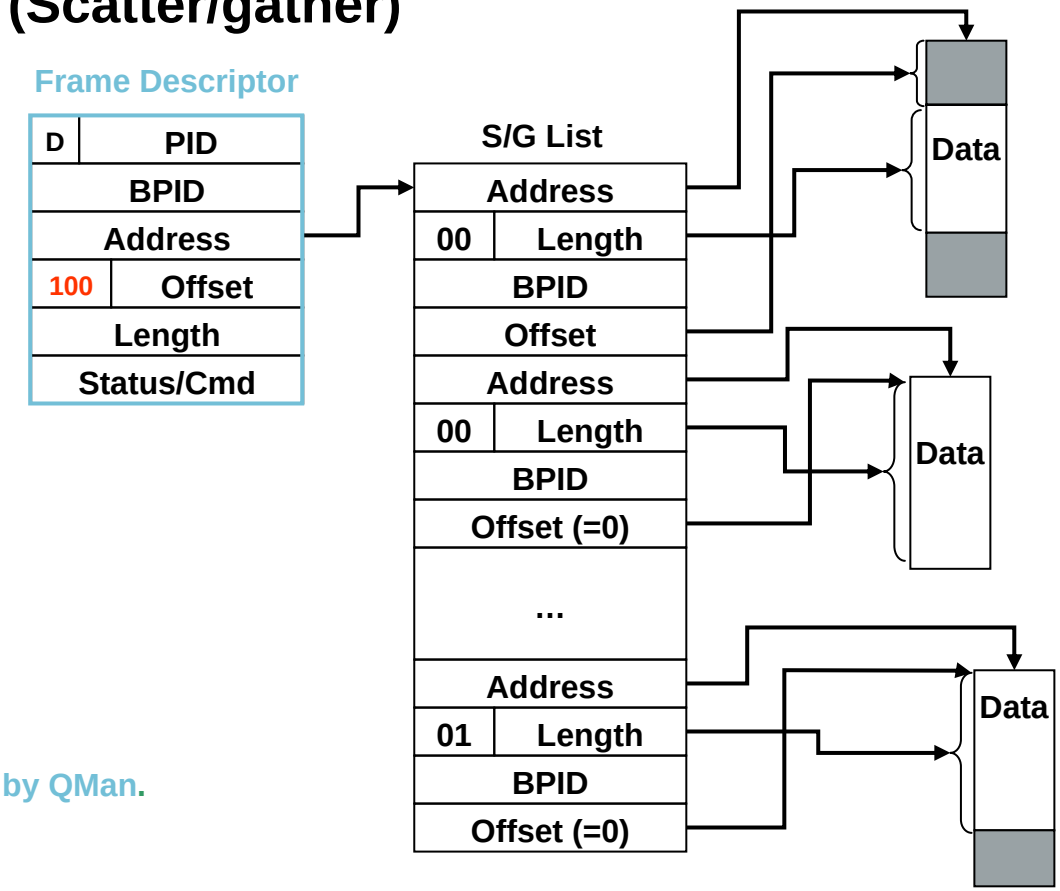
freescale semiconductor

DPAA Building Block: Frame Descriptor

Simple Frame



Multi-buffer Frame (Scatter/gather)



Frame Descriptor (FD) is the basic element queued by QMan.

Frame Format Code

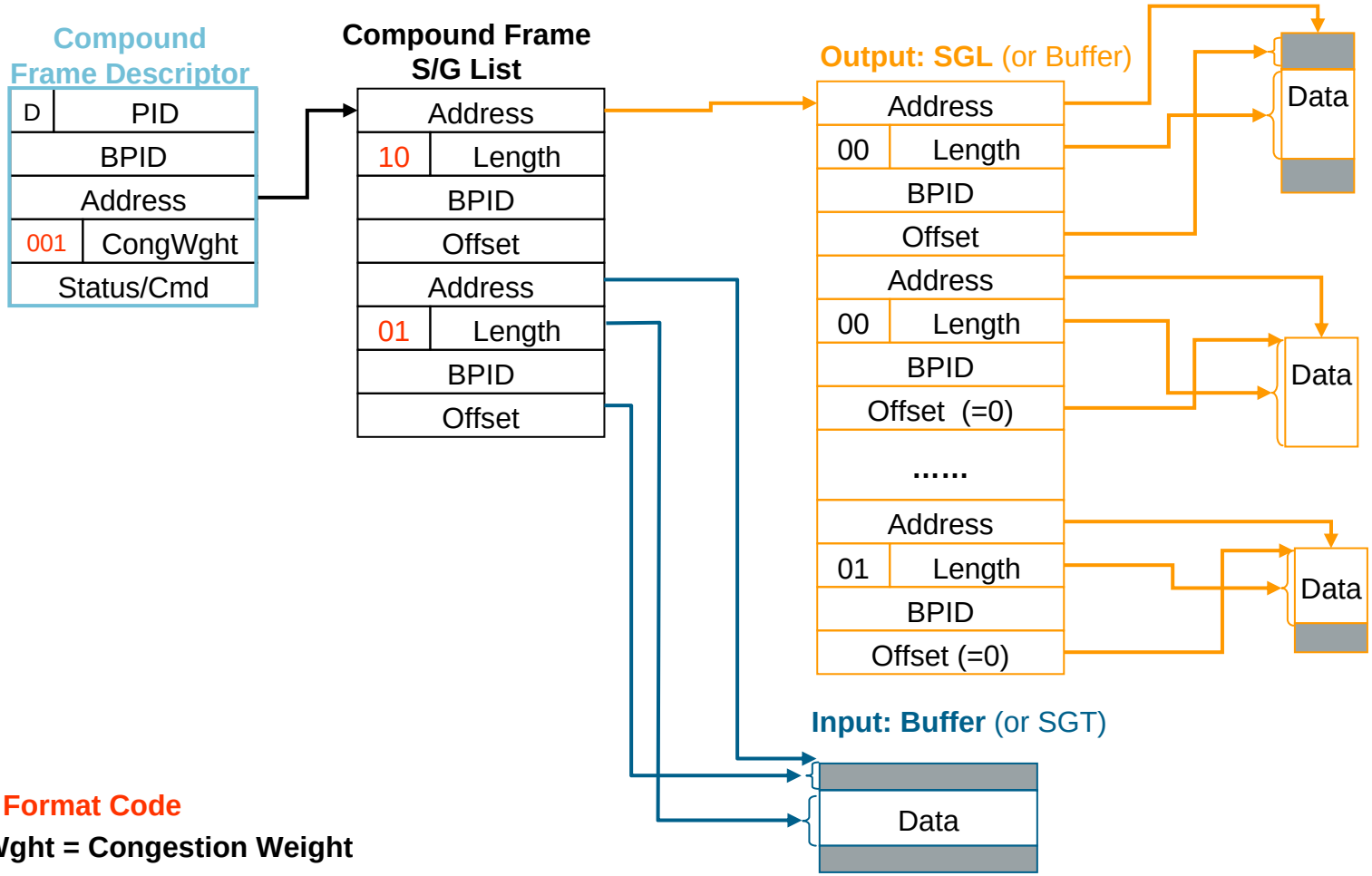
PID = Frame Partition ID

BPID = Buffer Pool ID

freescale semiconductor

Compound Frame

Compound frames describe multiple simple frames
Variation of the S/G list is used as the “top level” of a compound frame

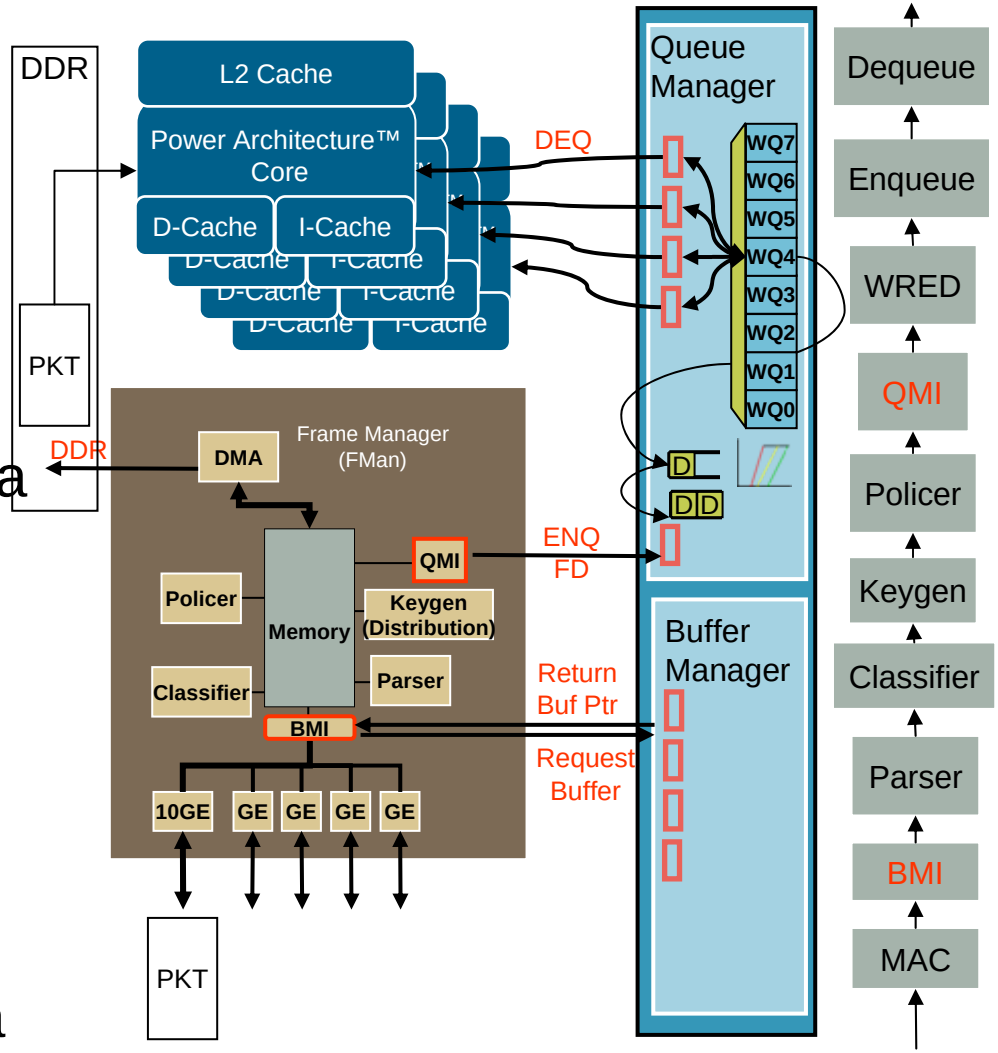


Frame Format Code

CongWght = Congestion Weight

Life of an Ingress Packet

- FMAN receives packets
  - allocates internal buffers
  - retrieves data from MAC
- BMI
  - acquires a buffer from BMan
  - uses DMA to store data in it
- Parse+classify+keygen select a queue and policer profile
- Policer “colors” and optionally discards frame
- QMan applies active queue management and enqueues frame
- Frame is enqueued to one of a pool of cores

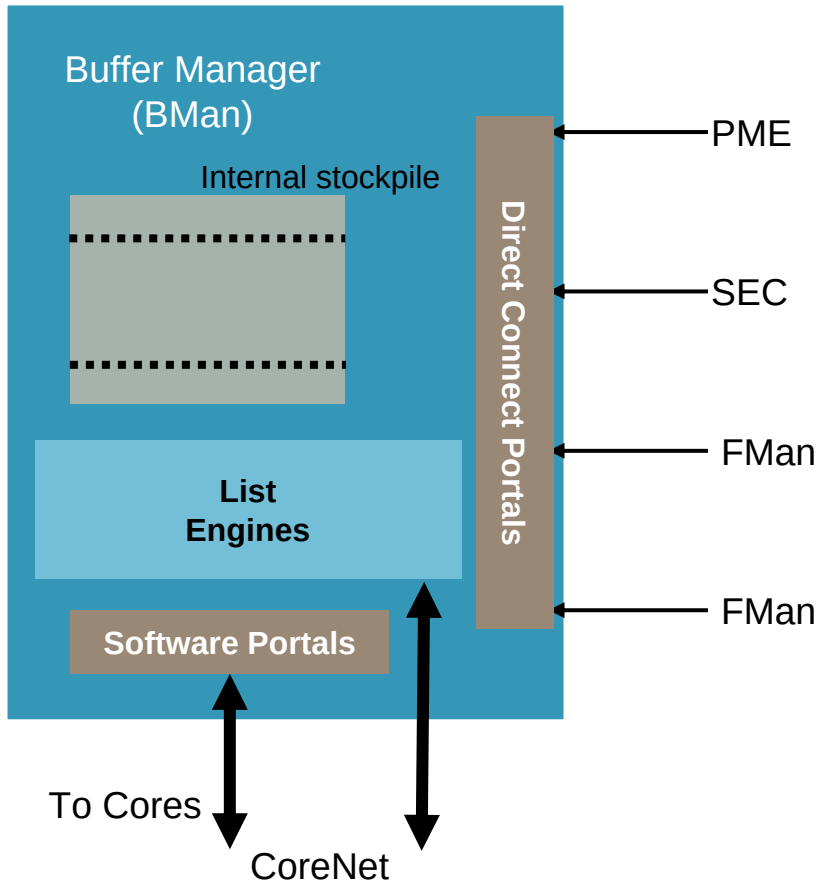


Agenda

- DPAA
- BMAN Deep Dive
- FMAN Deep Dive
- QMAN Deep Dive
- SEC Deep Dive
- PME Deep Dive
- Conclusion

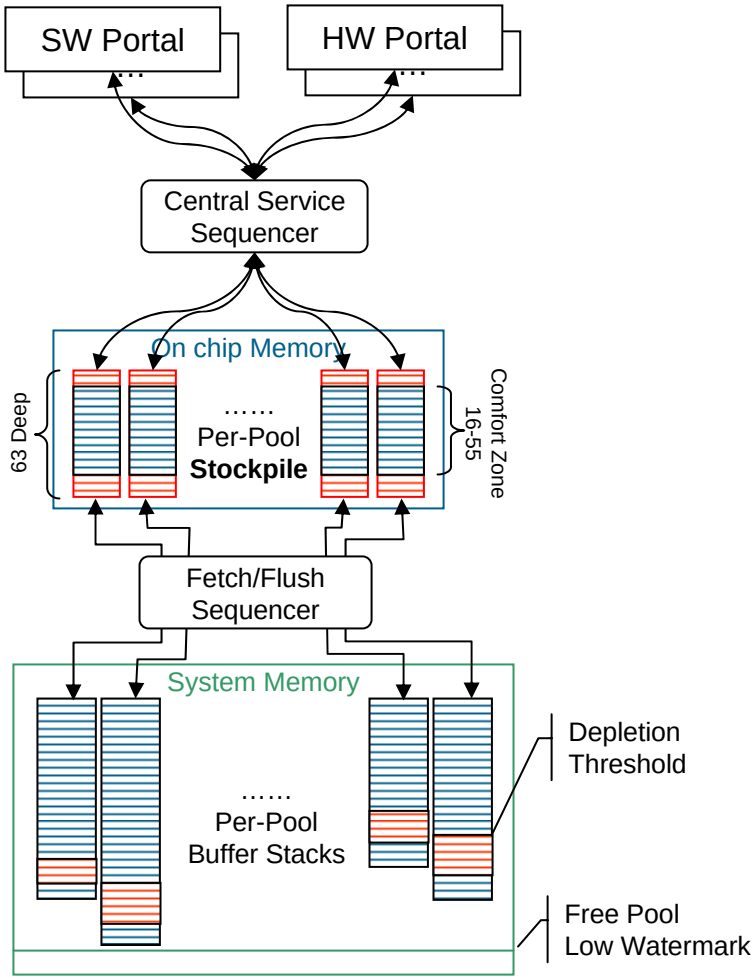
BMan Functions and Features

- Buffer Manager (BMan) supports:
- Hardware (and software) acquire and release of buffer pointers from/to pools
  - BMan is primarily intended to reduce the buffer management load on SW
  - It does not read or write to the actual buffers
- Support 64 buffer pools
  - Different partitions allow HW to use the right partition memory
  - Different sizes allow buffer allocation per packet size
  - Maximum 2G buffers pointers, average 32M buffer pointers per pool
- Pool depletion thresholds for pool replenishment and lossless flow control
  - All thresholds have hysteresis
- Hysteresis on all thresholds

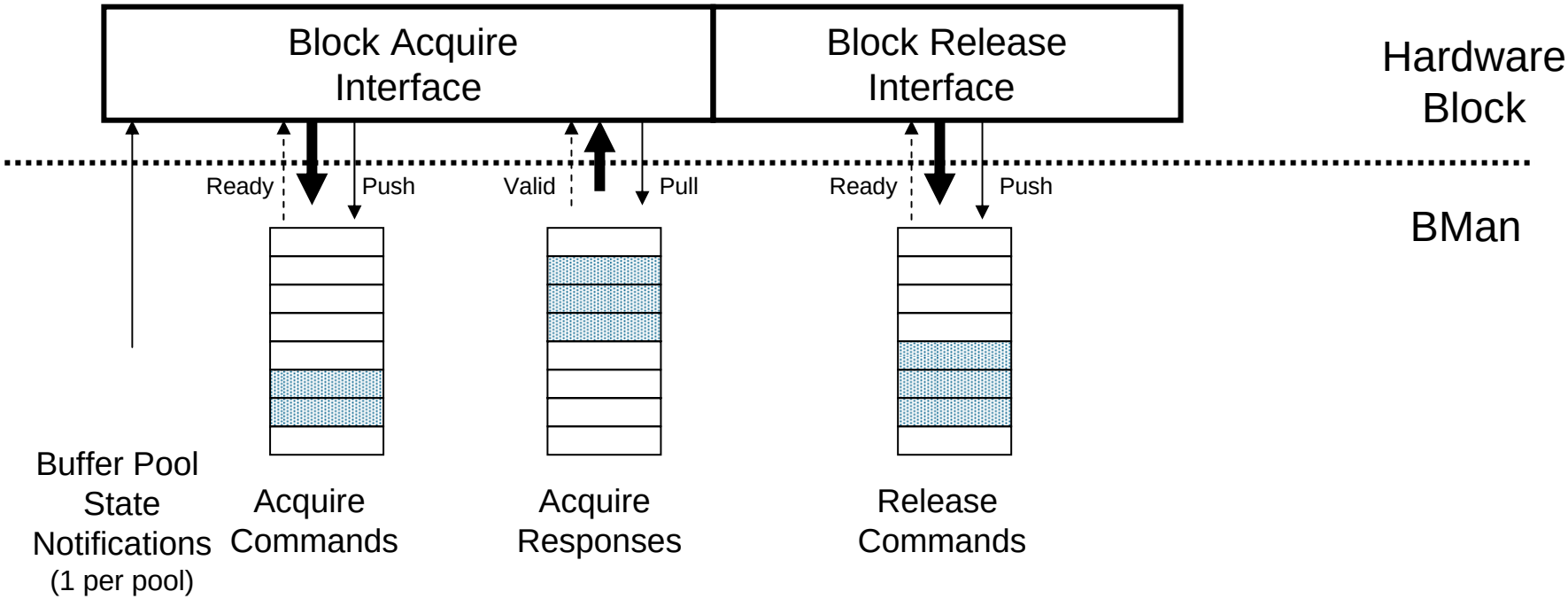


Buffer Manager

- Standardized command interface to SW and HW
  - SW portal per core resolves by HW any Multi Core race scenario
  - HW portal per HW block allows simplified command for HW IP's
  - Dedicated HW portal interface to FM
- BMan keeps a small per-pool stockpile of buffer pointers in internal memory
  - stockpile of 64 buffers per pool
  - Absorbs "bursts" of acquire/release without external memory access
  - minimized access to memory for buffer pool management.
  - Reduces acquire latency
- Pools (pointers) overflow into DRAM
- LIFO (last in first out) buffer allocation policy
  - optimizes cache usage and allocation
  - A released buffer is immediately used for receiving new data, using cache lines previously allocated



BMan HW Portal

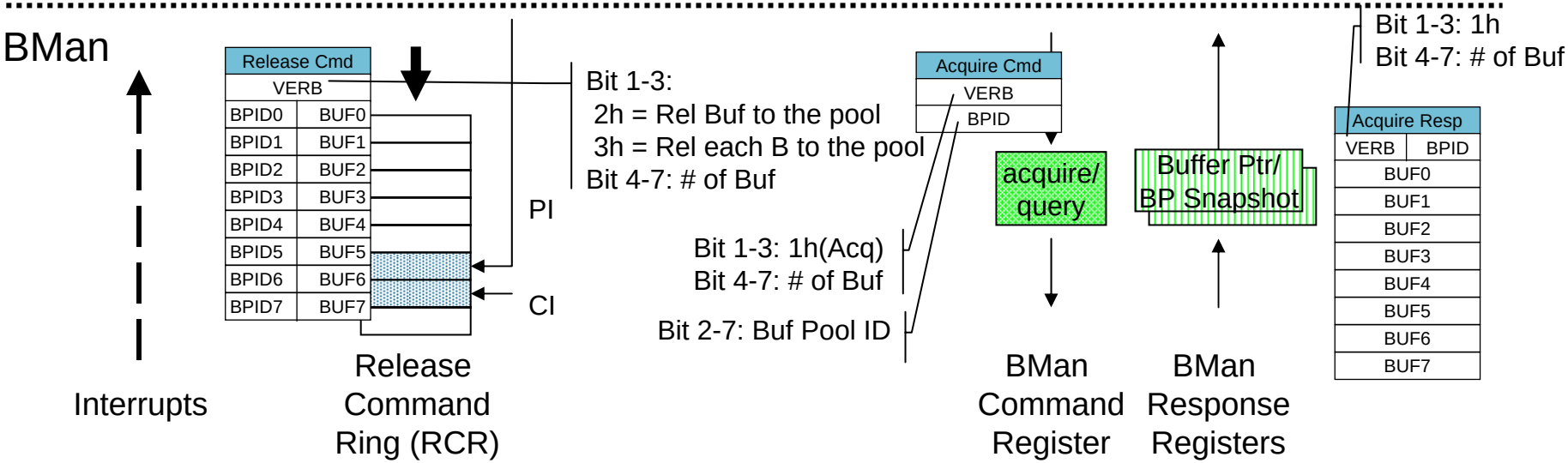


- HW portals consist of 4 sub-links
  - Acquire Command sub-link
  - Acquire Response sub-link
  - Buffer Pool State sub-link
  - Release Command sub-link
- Sub-link handshake signals define the HW block as the "master" but allow BMan to control flow

freescale semiconductor

BMan SW Portal components

Core

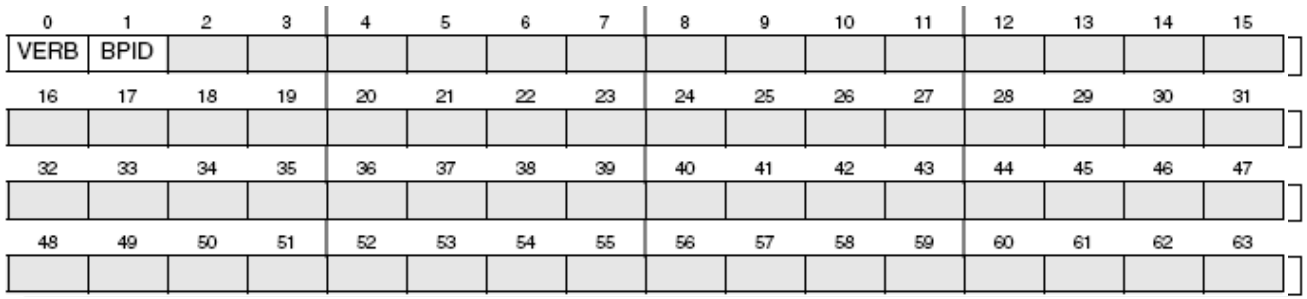


- Software portals have 2 components
  - Management commands:
    - Command Registers: acquire 1-8 buffers OR query availability
    - Response Registers: buffer address OR Buffer Pool Availability and Depletion state
  - Buffer Release: Release Command Ring RCR
    - Circular FIFO
- Interrupts can be used to signal availability of space (in RCR) and that pools are depleted and require replenishment

freescale semiconductor

BMAN Command and Response

- BMan Command Type
  - BMan command registers are 64B long.
  - Command Verb (1B) + Buffer Pool ID (1B)
    - Bit 1-3: Response Type. Valid encodings are:
      - 1h = Acquire buffers (Acquire).
      - 2h = Release buffers to the pool identified in byte field 1 (Release).
      - 3h = Release each buffer to the pool identified in byte field immediately preceding its buffer field (Release).
      - 6h = Invalid command (Response)
      - 7h = Stockpile ECC Error (Response)
    - Bit 4-7: Number of buffers associated with command type, maximum 8.
      - 0h = Zero buffers.
      - 1h = One buffer.
      - ....
      - 8h = Eight buffers.
    - Returns up to eight 48bit buffer addresses.



freescale semiconductor

Buffer Pool Depletion Thresholds

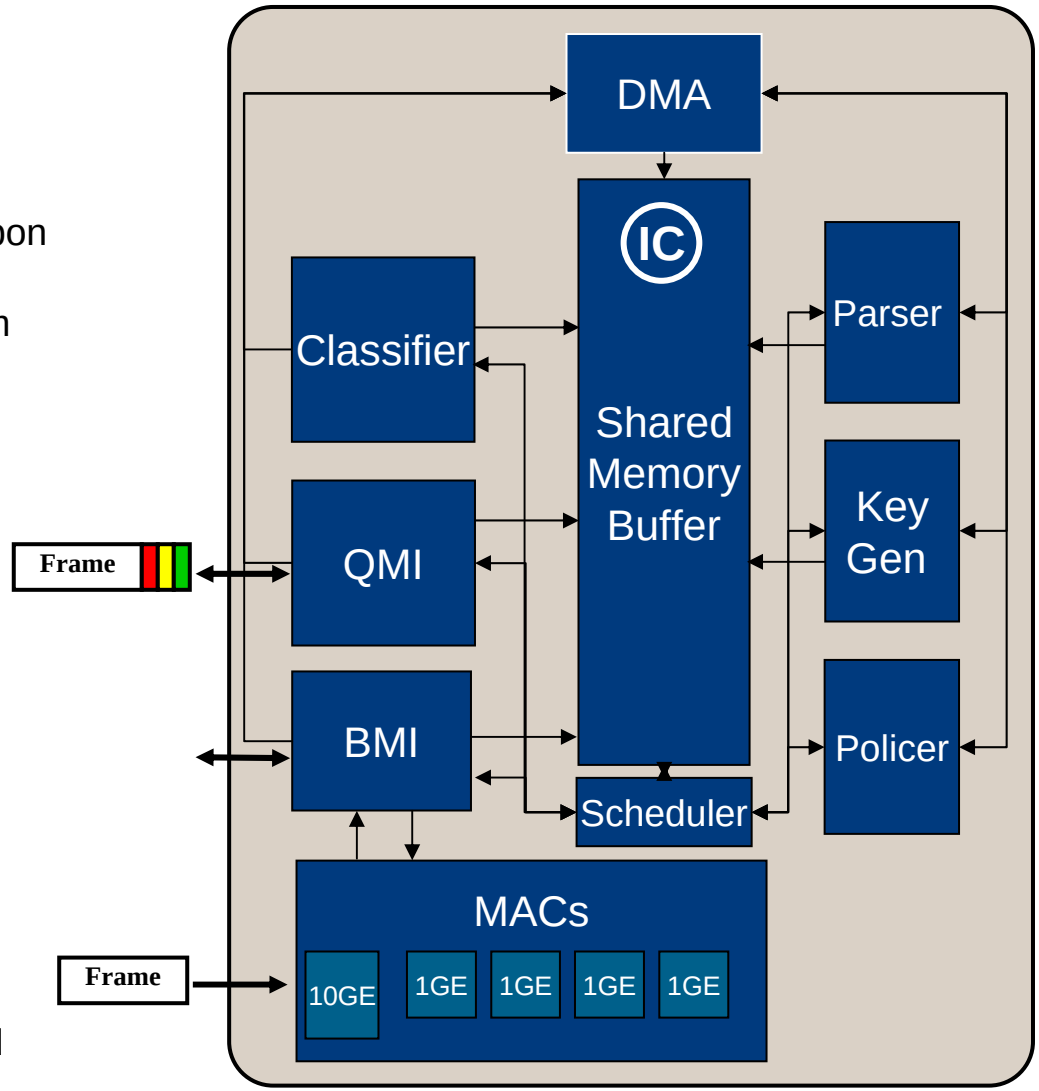
- BMan tracks the number of buffers in each pool
- Each time buffers are acquired from or released to a pool the length is compared against two per pool thresholds
  - One threshold is used to determine whether a pool should be **replenished** by software
    - When the pool falls below this threshold an interrupt can be asserted
  - The second threshold is used to determine whether a pool has too few buffers to be able to continue servicing incoming network traffic – so that the network interface can issue its **flow control signal**
- Both thresholds have a second hysteresis value which is used to determine when the pool has exited that depletion state

Agenda

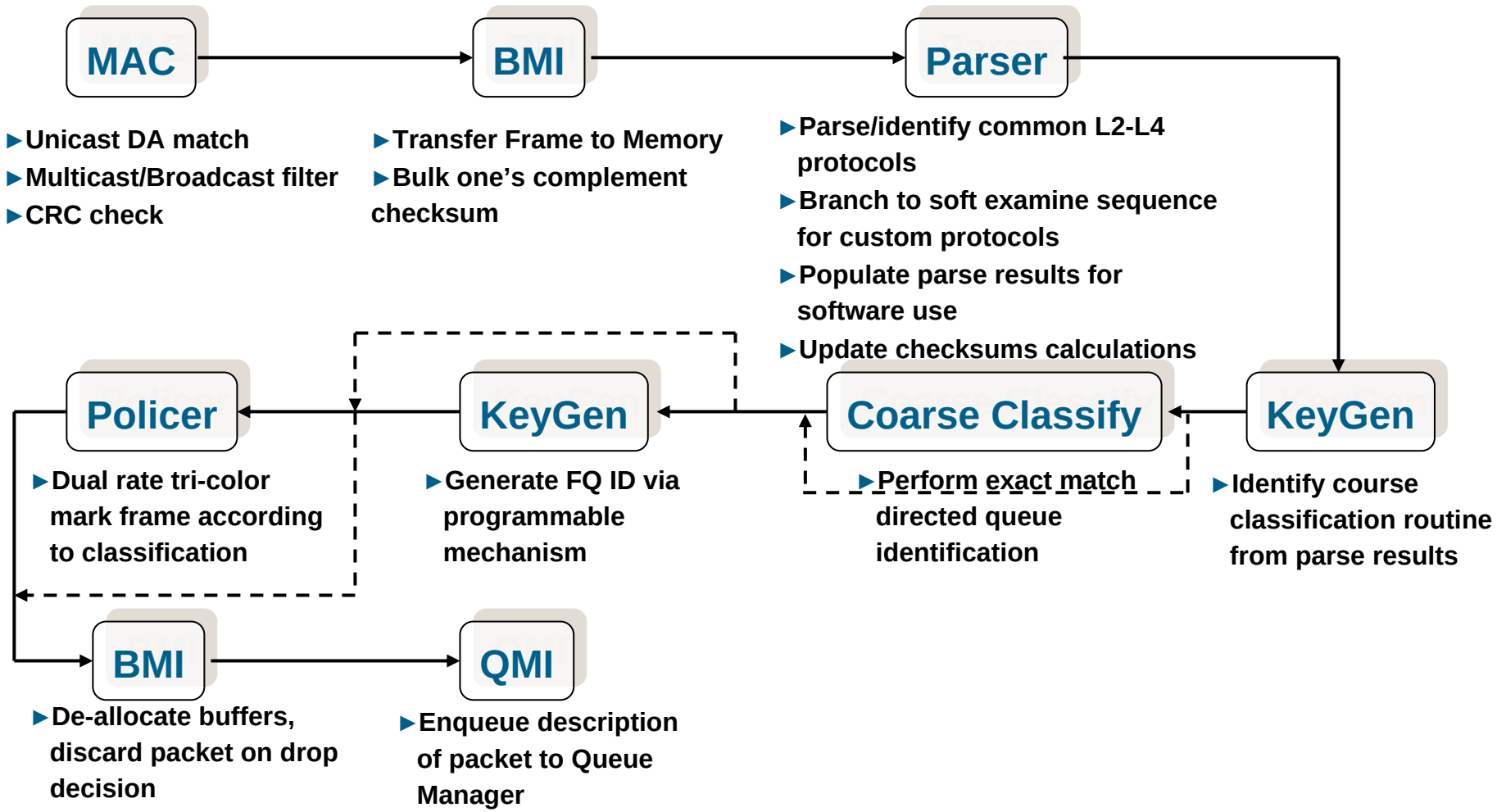
- DPAA
- BMAN Deep Dive
- FMAN Deep Dive
- QMAN Deep Dive
- SEC Deep Dive
- PME Deep Dive
- Conclusion

Frame Manager

- **Ingress (RX)**
  - Timestamp incoming packet (IEEE 1588 representative)
  - Acquisition of “right sized” buffer based upon Ethernet packet size
  - L2 CRC validation, IPv4 header checksum and TCP/UDP checksum validation
  - L2/L3/L4 protocol parsing (hard and soft examination sequences)
  - Classification based queue distribution
    - Exact match to queue
    - Configurable flow hash to queue
  - Color marked policing decision (RFC2698, RFC4115)
  - Direct hardware enqueue to QMan queue defined via classification
- **Egress (TX)**
  - Direct hardware dequeue from QMan
  - IPv4, TCP/UDP checksum update
  - Transmit rate based metering per port
  - Buffer constituting the packet are returned



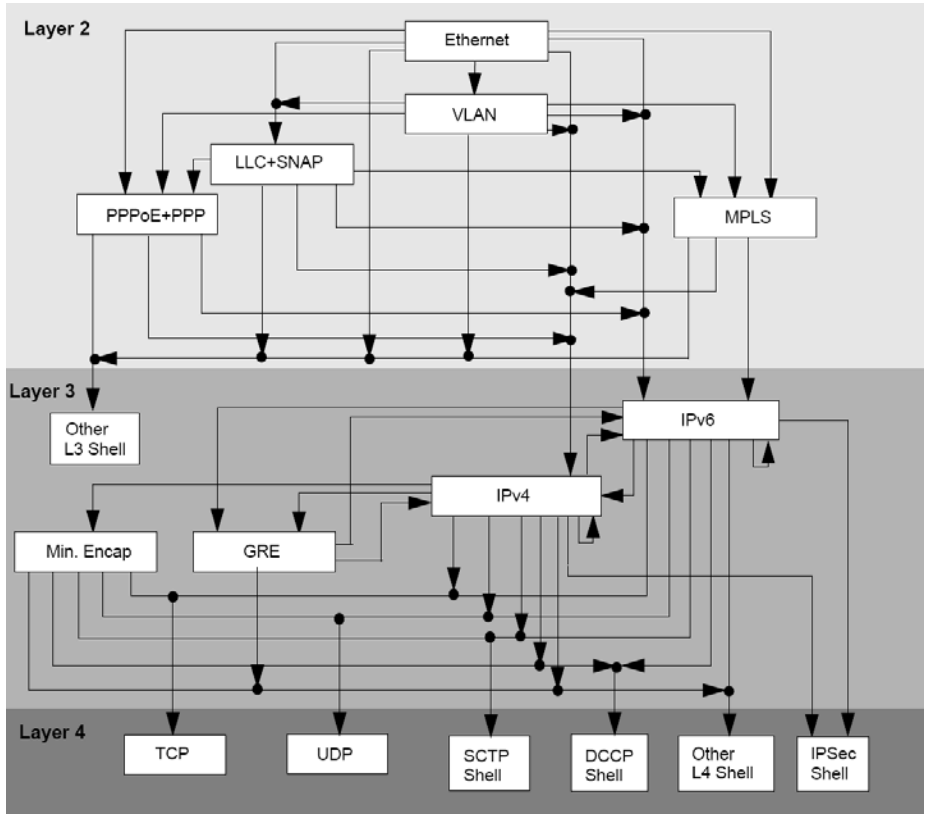
Frame Manager Flow Chart



freescale semiconductor

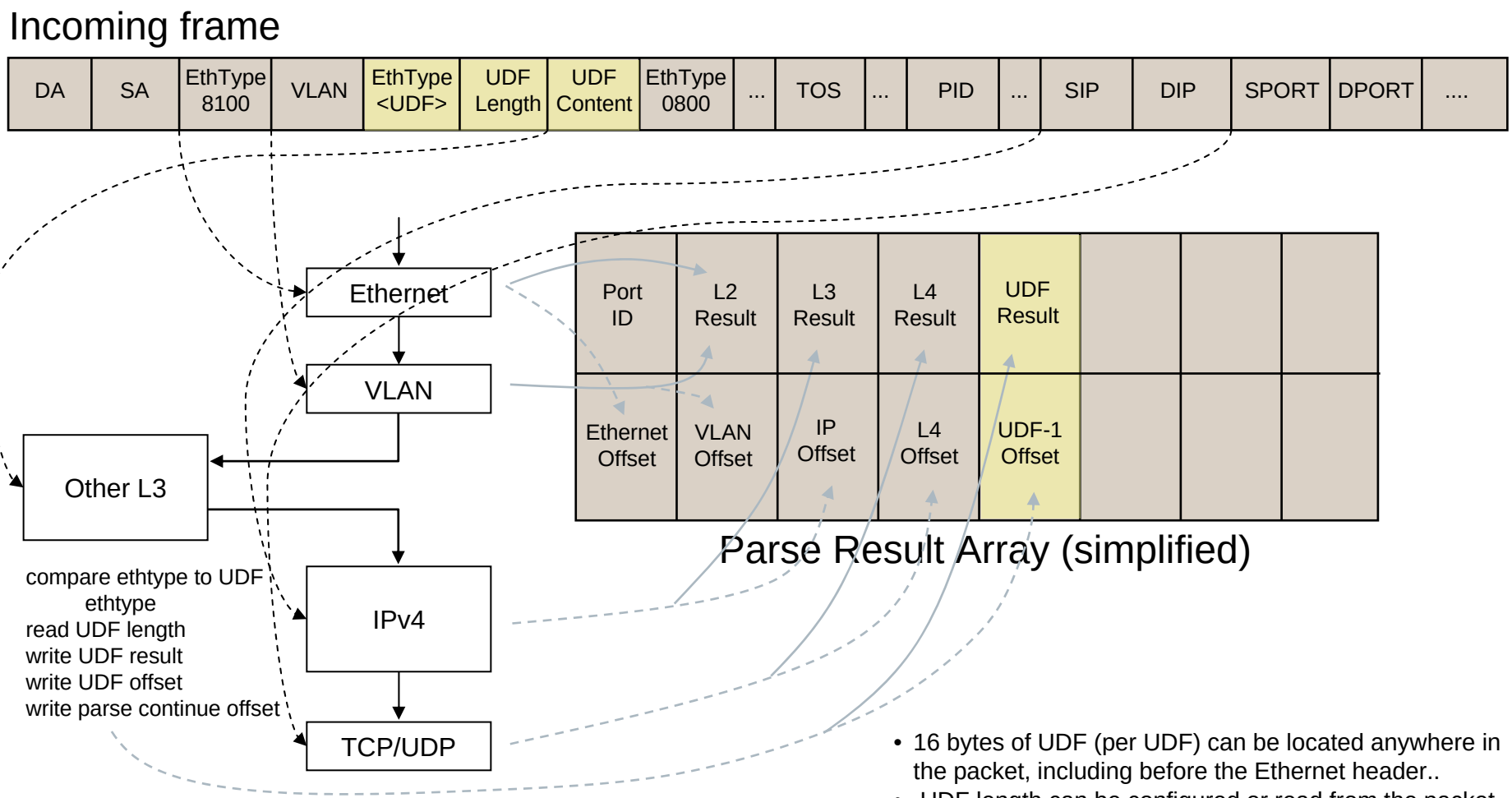
Parser

- Determine protocol correctness
  - IP/TCP/UDP checksum
  - Version number check
  - Length field check
  - Frame too short, long
- Give protocol stack indication to KeyGen and/or software
- Support for hard and soft parse algorithms
  - Soft parse can detect ≤3 proprietary/user defined fields and extract offset to Parser Result
  - Soft parse can do ‘classification’ by e.g. address comparison instructions (limited functionality)



freescale semiconductor

Flexible Parsing of Proprietary User-Defined Fields (UDF)



- 16 bytes of UDF (per UDF) can be located anywhere in the packet, including before the Ethernet header..
- UDF length can be configured or read from the packet.
- UDF information is available to the classifier and s/w.
- Standard parsing can continue after UDF parsing.

freescale semiconductor

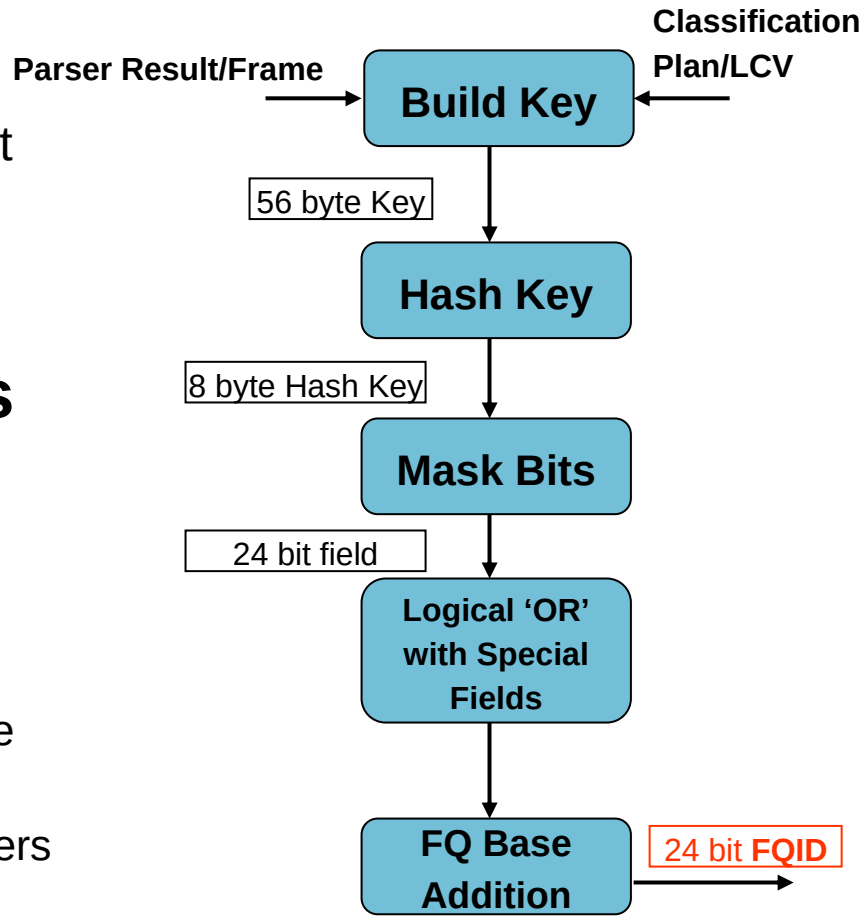
KeyGen – Key Generator

256 Classification Plans

- Indicates which fields of a parsed packet are of interest for key generation

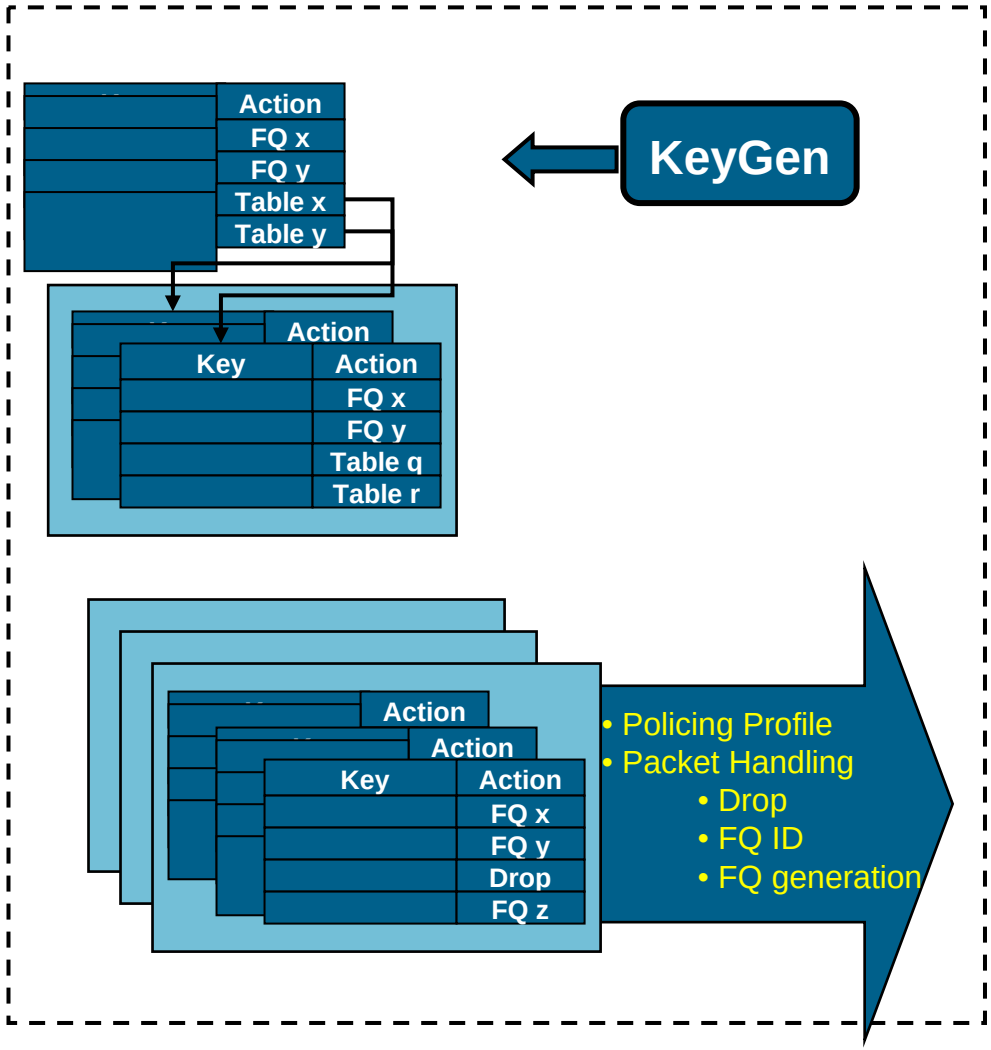
32 Key Generation schemes

- Direct Method
  - Used for port based or post coarse classification
- Indirect Method
  - Based on the parsed protocol stack of the frame and the source port
  - The presence (or absence) of valid headers can direct the scheme used



Classifier

- **Up to three level tree search at line rate**
  - Up to 256 bytes per tree level
  - Up to 512 bytes of total key data for the three level
  - i.e. 32 entries with 4 byte key is considered as 128 bytes table
- **Each table is an ordered list of entries, returning the first match.**
  - Keys for the initial table are generated from parse results/KeyGen
  - Keys for subsequent tables are generated from parse and previous lookup results
  - Keys can be generated from any fields in the frame, including proprietary user-defined fields
  - Each entry is individually maskable. Masks occupy key data space
- **Each table entry has Action description**
  - Queue ID, next table, hash and distribute, drop.
- **Output of classifier**
  - Single queue
  - Set of queues and a KeyGen scheme if distributing.
  - Policing Profile
- **The two FM's provide total performance of 36MPPs**
  - System running less then this rate can use more levels or larger tables



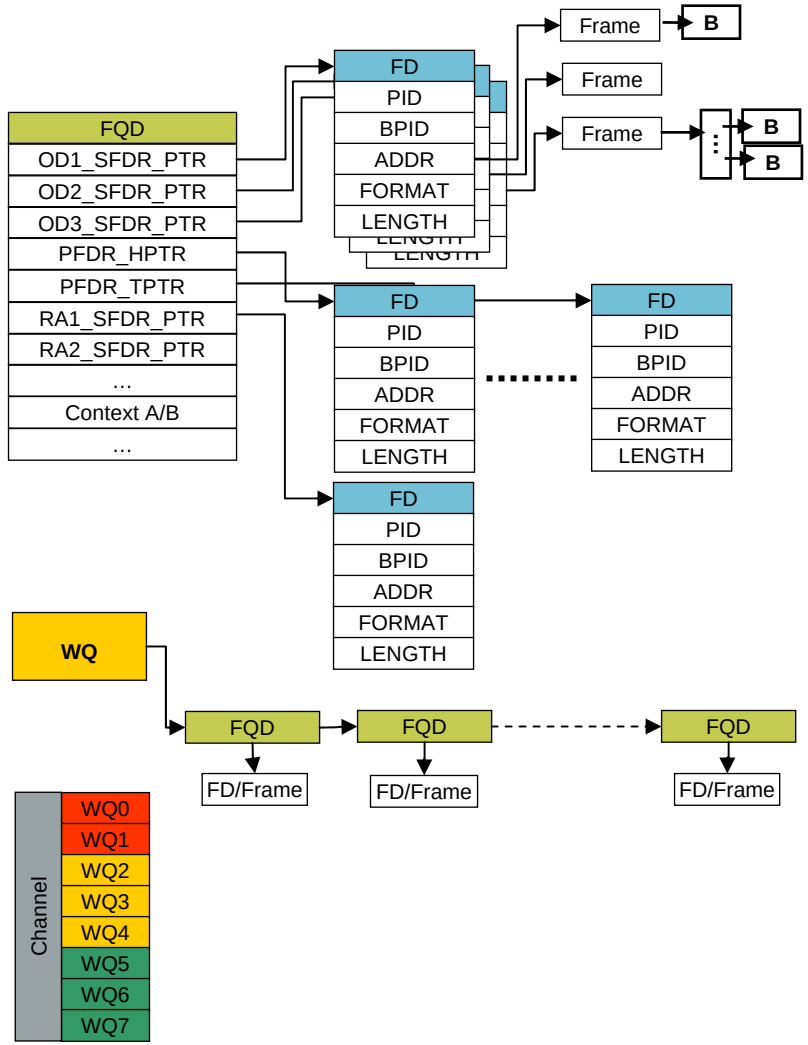
freescale semiconductor

Agenda

- DPAA
- BMAN Deep Dive
- FMAN Deep Dive
- QMAN Deep Dive
  - QMAN Building Blocks
  - QMAN Functions
  - QMAN Scheduling
  - Order Restoration, Order Preservation and Atomicity
  - Congestion management and avoidance
- SEC Deep Dive
- PME Deep Dive
- Conclusion

Queuing Structure

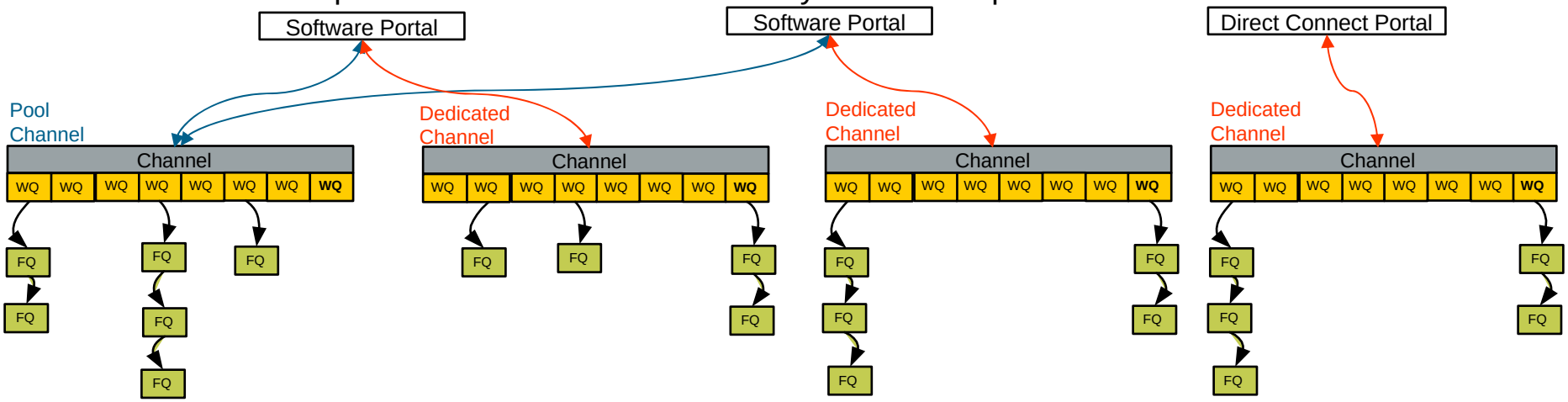
- Frame Descriptor (FD)
  - The basic queue Element that describe a frame
  - Usually a single IO packet will use a single frame
  - Other scenarios: commands with no buffer
- Frame Queue Descriptor (FQD)
  - A linked list of FD's
  - Usually a frame queue is associated with flow.
  - Head of frame queue can be associated to ODP
  - Enqueue operation must include the target FQ as a parameter
  - Dequeue operation may use FQ as a parameter for operation
- Work Queue Structure
  - Linked list of FQD
  - Hold flows of the same priority and designation
  - Dequeue operation may use WQ as a parameter for operation
- Channel
  - Set of eight WQ Channel served by a single type of entity
  - Dequeue from channel can be configured to be:
    - strict priority
    - round robin (Simple, Weighted or Deficit)
  - Dequeue may use Channel as a parameter for operation



freescale semiconductor

QMAN Communicates with Portals

- Portals are the interface between QMan and the blocks which use them
  - **Direct Connect Portals** has direct connect signals to Dedicated Channel
    - Dedicated channels are always serviced by a single entity, e.g. FMAN, etc.
  - **Software Portals** use CoreNet as the physical interconnect to the processor core.
    - Each Software Portal serves a dedicated channel, and optionally services Pool Channels.
    - Software and QMan interact by “reading” and “writing” data across CoreNet
- Each channel consists of 8 WQs, and thus there are 8 possible priorities
- QMan contains a total of 51 Channels
  - 12 Dedicated Channels per direct connect Portal FMan
  - 1 for SEC, 1 for PME
  - 10 for CoreNet dedicated channels for software portals
  - 15 for CoreNet pool channels which are shared by all software portals



freescale semiconductor

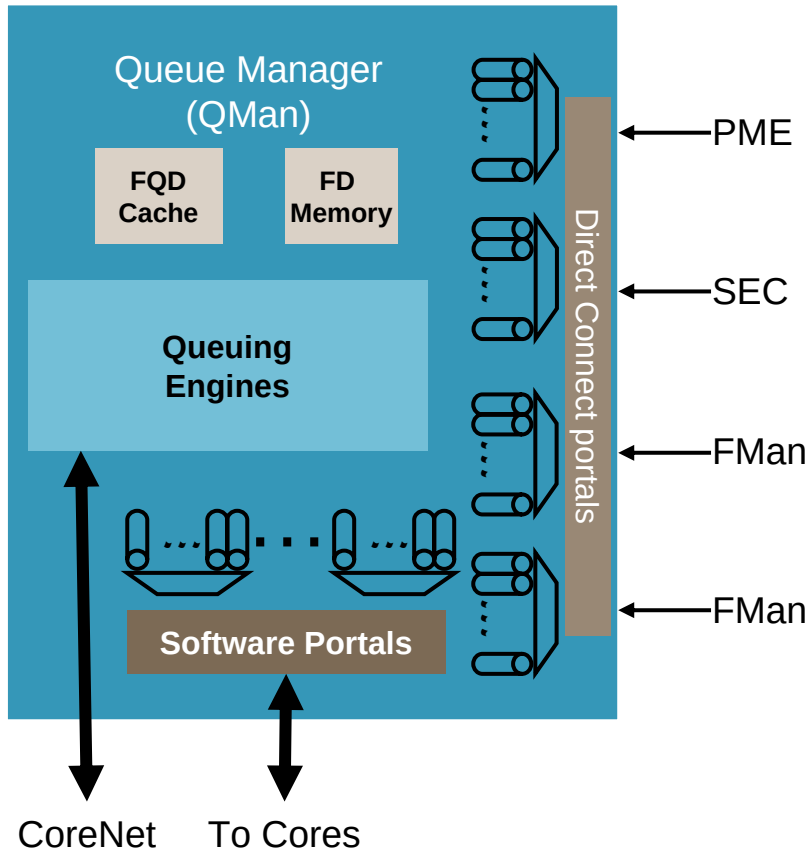
Agenda

- DPAA Overview
- FMAN Deep Dive
- QMAN Deep Dive
  - QMAN Building Blocks
  - QMAN Functions
  - QMAN Scheduling
  - Order Restoration, Order Preservation and Atomicity
  - Congestion management and avoidance
- BMAN Deep Dive
- SEC4.x Deep Dive
- PME Deep Dive

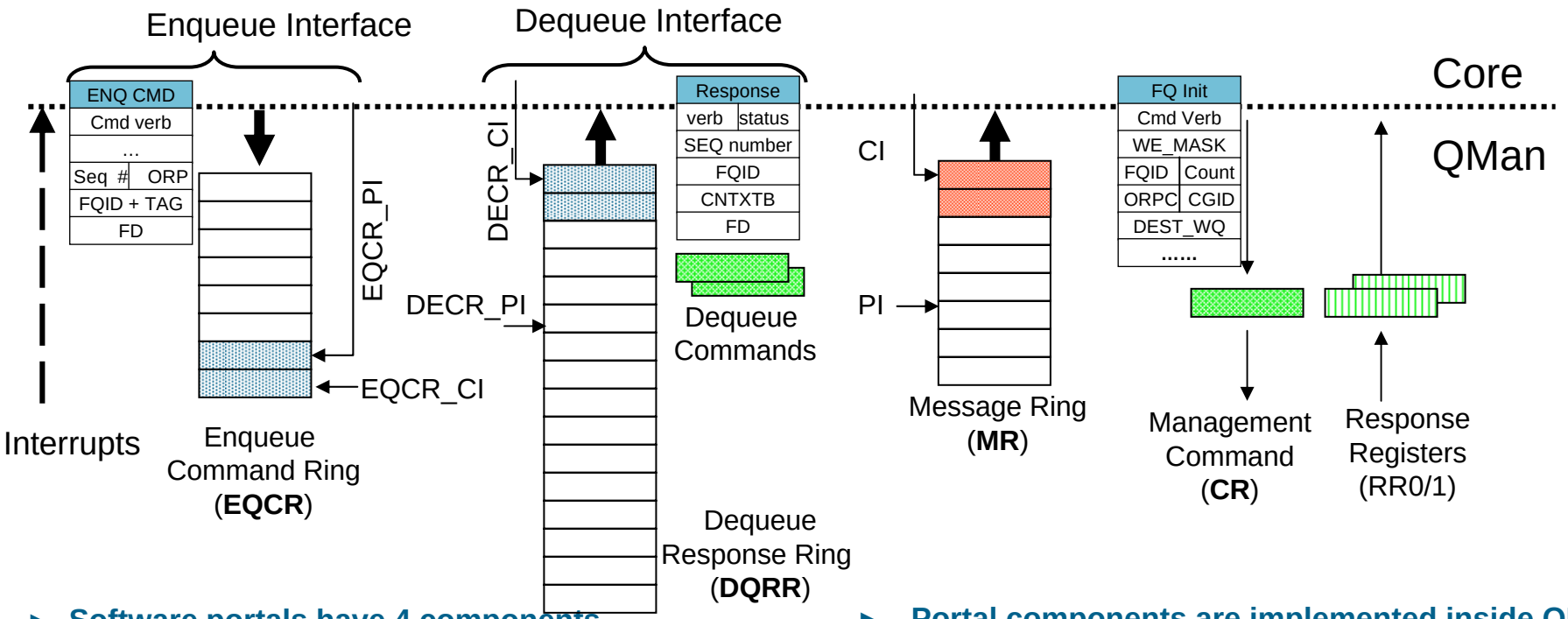
freescale semiconductor

Queue Management

- QMan provides a way to inter-connect DPAA components
  - Cores (including IPC)
  - Hardware offload accelerators
  - Network interfaces – Frame Manager
- Queue management
  - High performance interfaces (“portals”) for enqueue/dequeue
  - Internal buffering of queue/frame data to enhance performance
- Congestion avoidance and management
  - RED/WRED
  - Tail drop for single queues and aggregates of queues
  - Congestion notification for “loss-less” flow control
- Load spreading across processing engines (cores, HW accelerators)
  - Order restoration
  - Order preservation/atomicity
- Delivery to cache/HW accelerators of per queue context information with the data (Frames)
  - This is an important offload for software using hardware accelerators



QMan Software Portal Components



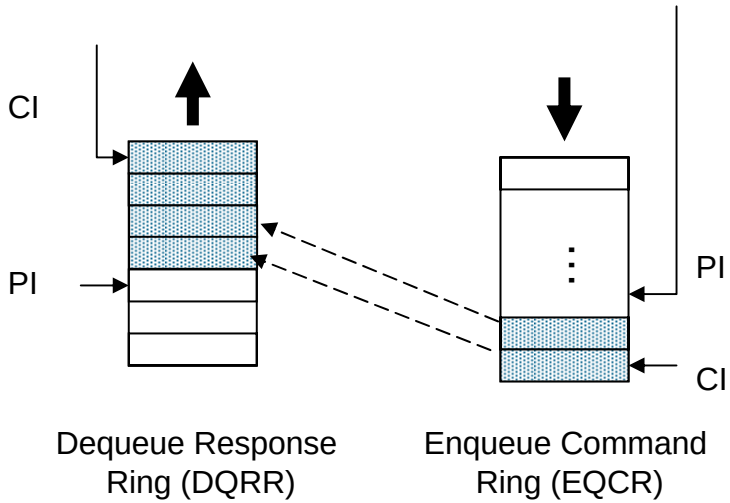
- **Software portals have 4 components**
  - Enqueue: EQCR
  - Dequeue: Command registers + DQRR
  - Messages: MR
    - Asynchronous error messages (e.g. enqueue rejections)
  - Management commands: command/response registers
- **Interrupts can be used to signal availability of data or space (in EQCR)**
- **Rings provide finite size FIFOs**
  - Up to 16 entries for DQRR, 8 entries for EQCR and MR
- **Portal components are implemented inside QMan to reduce access latency**
  - Unlike traditional BD rings which are in "memory" and "registers"
- **QMan can "push" (stash) DQRR entries across CoreNet into the appropriate core's cache**
- **PI and CI are the basic mechanisms used with rings but other forms of notification of data availability and data consumption are supported**
- **When these other mechanisms are used QMan maintains PI/CI**

freescale semiconductor

Dequeue modes

- QMan supports 2 modes on software portals:
  - Push Mode:
    - In this mode, QMan will continue to push entries into DQRR in attempt to keep it “full”
    - QMan provides 2 command registers
      - One register is “static” and QMan repeatedly executes this command
      - One register is “volatile” and QMan executes that command a limited number of times
    - Push mode is “just like” a BD ring
  - Pull Mode:
    - QMan provides a single command register
    - Software must issue a new command for each dequeue operation
- Push mode is the most common mode
- Pull mode offer more control to applications

Discrete Consumption Acknowledgement



- Each entry in the TX ring (EQCR) can contain a DCA for the corresponding entry on the RX (DQRR) ring

- DCA is a form of “indirect” CI updating
- DCA is targeted at forwarding applications where most frames which are received are then transmitted
- A EQCR entry can acknowledge multiple DQRR entries

freescale semiconductor

QMan cache warming

- In addition to stashing DQRR entries into cache, QMan's software portals can also "warm" a core's (L1 or L2) cache with frame and queue related data
  - Actual frame data for single buffer frames
  - Scatter gather list for multi-buffer frames
  - Frame "annotations"
    - Data between Address and Offset at start of frame
    - Used to pass additional information about the frame which is not "frame data" e.g. FM parse results
  - Data referenced by FQ Context
- These stashing options can be configured on per FQ basis
- Operations are performed at the time that the frame is dequeued (i.e. DQRR entry is created)

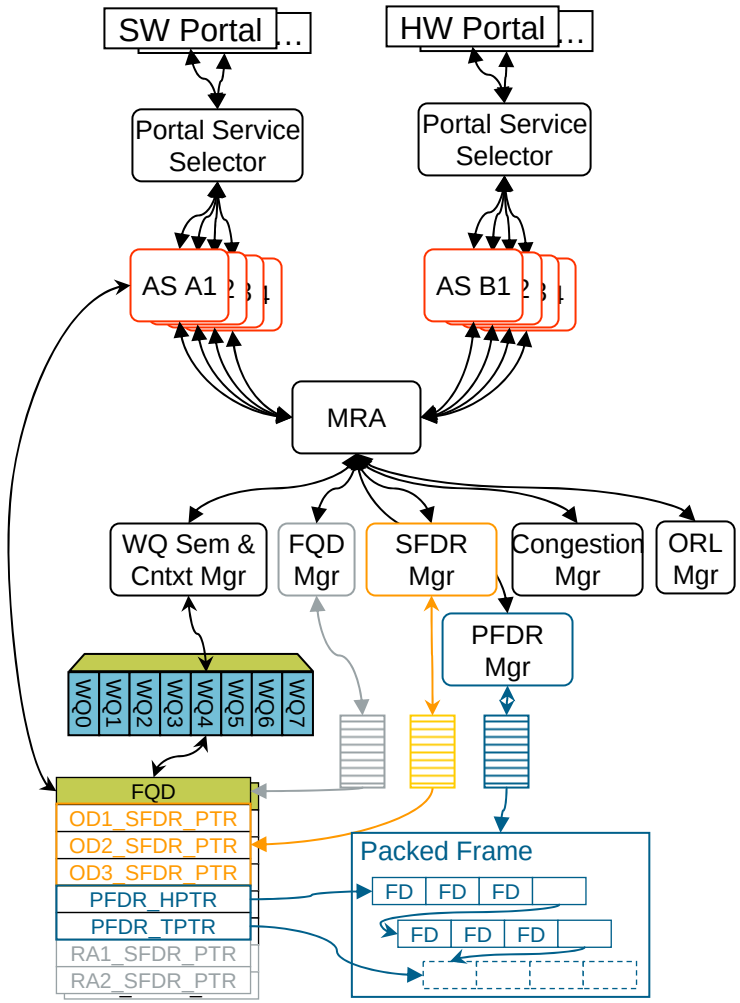
Agenda

- DPAA Overview
- FMAN Deep Dive
- QMAN Deep Dive
  - QMAN Building Blocks
  - QMAN Functions
  - QMAN Scheduling
  - Order restoration, order preservation and atomicity
  - Congestion management and avoidance
- BMAN Deep Dive
- SEC4.x Deep Dive
- PME Deep Dive

freescale semiconductor

Scheduling: The Big Picture

- Portal Service Selector
  - select the next portal that has command to be executed,
  - Assign commands to the AS in the pool
  - Simple RR selection between the portals
- Algorithmic Sequencer (AS)
  - used within QMan to execute the commands received across the various portals
  - Isolate commands from HW and SW Portals
  - Up to 2 commands per cycle, one in each of the AS pools
- Multiway Resource Arbiter (MRA)
  - Independent arbiter between each of the Algorithmic Sequencers (AS) and the resource managers
  - Selection is base on work conserving RR
- Single Frame Descriptor Record Manager
  - Build Frame Queues
- Packed Frame Descriptor Record Manager
  - Acquire and Release PFDR
- Order Restoration List Manager
  - Enable frames from same FQ to processed in parallel while restore the temporal order before enqueue to one or multiple destination Frame Queue(s)



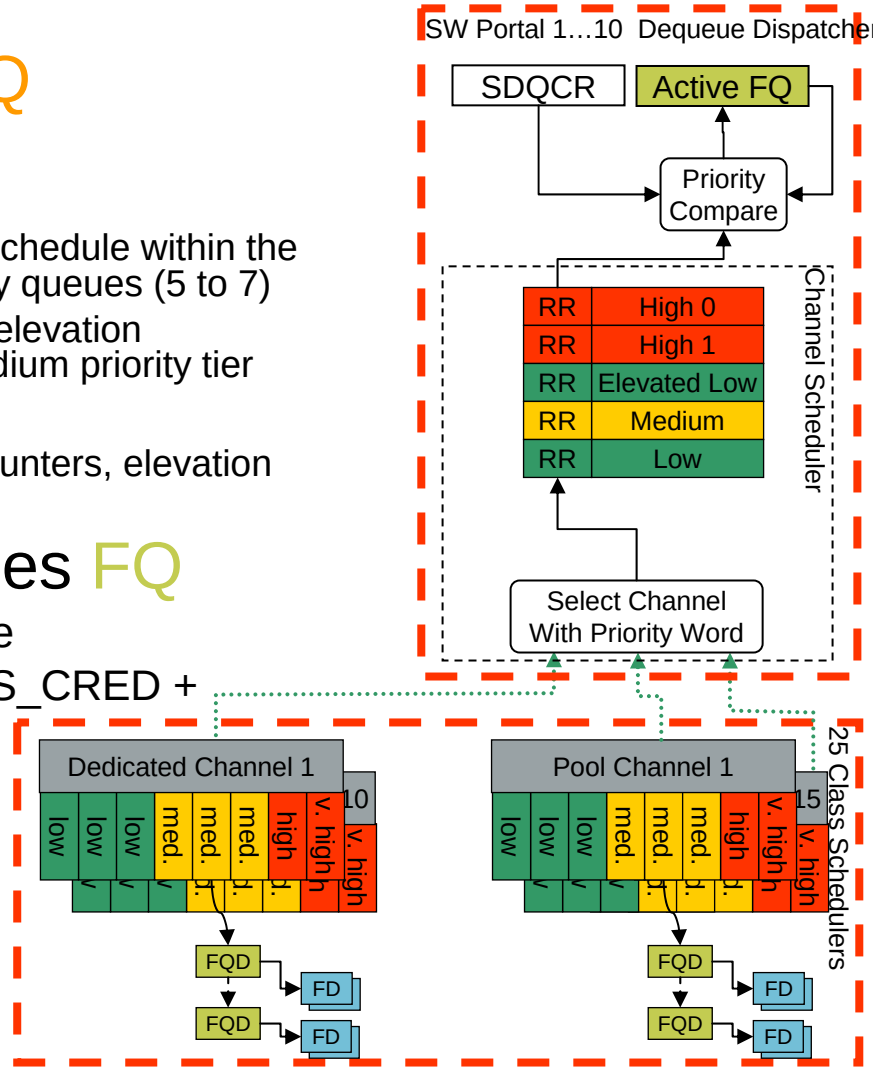
freescale semiconductor

Dequeue Scheduling

- Class Scheduler schedules WQ
  - A Class scheduler per channel
  - Two levels of scheduling:
    - Use **Weighted Interleaved Round Robin** to schedule within the medium priority queues (2 to 4) and low priority queues (5 to 7)
    - Strict priority of all 8 WQs with programmable elevation (CS_ELEV) of the low priority tier over the medium priority tier
  - Maintain active FQ states transition
    - keeps track of the last RR winner, selection counters, elevation pending counter
- Intra-Class scheduling schedules FQ
  - Schedule a frame queue within a work queue
  - Use **Modified Deficit Round Robin** with ICS_CRED + ICS_SURP

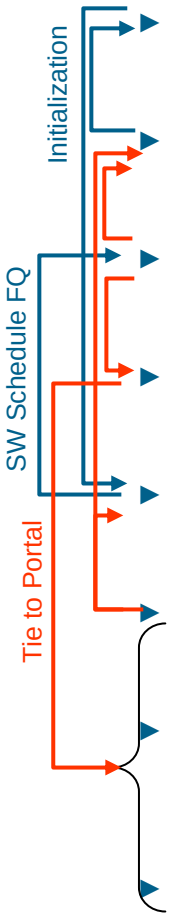
15 bit Credit	16 bit Surplus

  - First Dequeue surplus = surplus + credit
  - Dequeue 1-3 frames,
  - subtract frame length(s) bytes from surplus
  - If surplus > 0, dequeue 1-3 frames more
  - If surplus <=0, reschedule Frame Queue



freescale semiconductor

Frame Queue Life Cycle



0 Out of Service State:
- FQ ready for Initialization to a Park state.
- Software can transform Retired FQ to Out of Service.

1 Retired State:
- This state is a precursor to the Out of Service State.
- Placed in this state upon command at the earliest possible opportunity.

2 Tentatively Scheduled State
- consumer has given scheduling control of the FQ to the QMan.
- the eligibility criteria is not currently met.

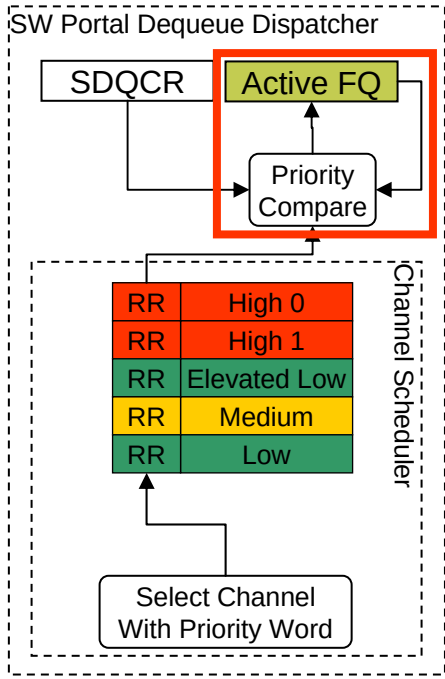
3 Truly Scheduled State
- eligibility criteria is currently met
- FQ is queued in a WQ waiting to become the active or held active.

4 Parked State:
- The consumer (rather than the QMan) has scheduling control of the FQ.
- Producers may enqueue to FQs in this state.

5 Active State:
- FQ is temporarily coupled to a single portal.
- Consumers may dequeue and Producers may also enqueue to this FQ.

Held Active State:
- FQ cannot be scheduled or consumed via any other portal.
- Consumers may dequeue and Producers may also enqueue to this FQ.
- Released from the portal after all frames have been consumed.

Held Suspended State:
- A FQ moves to this state when it is no longer eligible for dequeue.
- Consumers cannot dequeue. Producers may still enqueue to it.

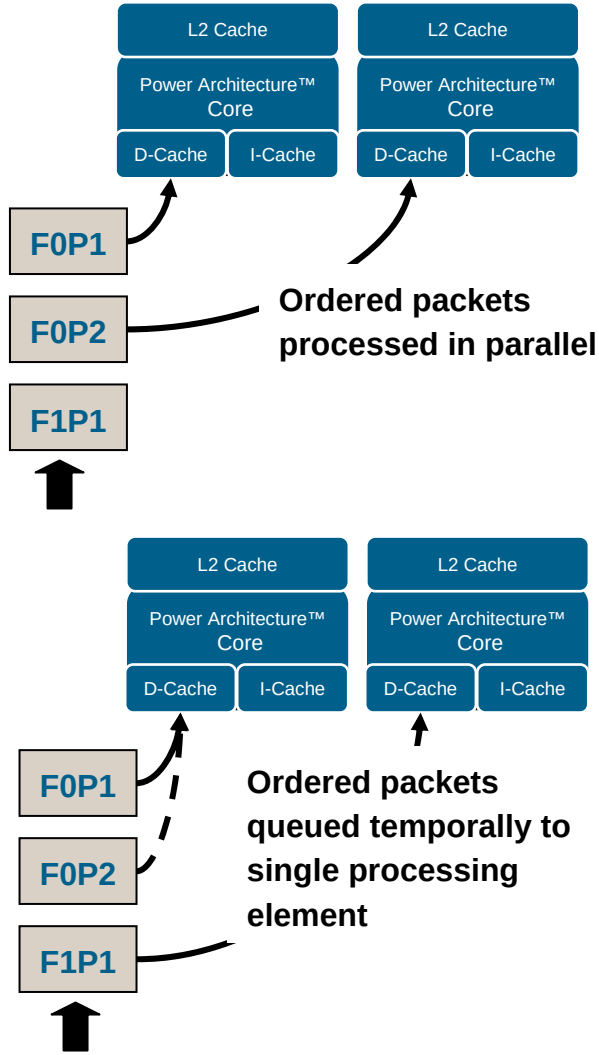


Agenda

- DPAA Overview
- FMAN Deep Dive
- QMAN Deep Dive
  - QMAN Building Blocks
  - QMAN Functions
  - QMAN Scheduling
  - Order Restoration, Order Preservation and Atomicity
  - Congestion management and avoidance
- BMAN Deep Dive
- SEC4.x Deep Dive
- PME Deep Dive

Addressing Ordering Requirements

- **There are two basic approaches to addressing this requirement:**
  - **Order restoration**
    - Take note of the correct order (or sequence) of packets before processing starts and restore the packets to that order before they are transmitted
  - **Order preservation**
    - Ensure that related packets are processed in order (and typically one at a time)
    - Order preservation can also provide “atomicity” – atomic access to data used in processing the frame
- **QMan requires that related frames (which must be transmitted in order) be placed on the same frame queue for both of these approaches**
  - This does not mean that **only** related frames are placed on a given FQ
  - Many sets of related frames can be placed on an FQ
  - Frame Manager is responsible for achieving this

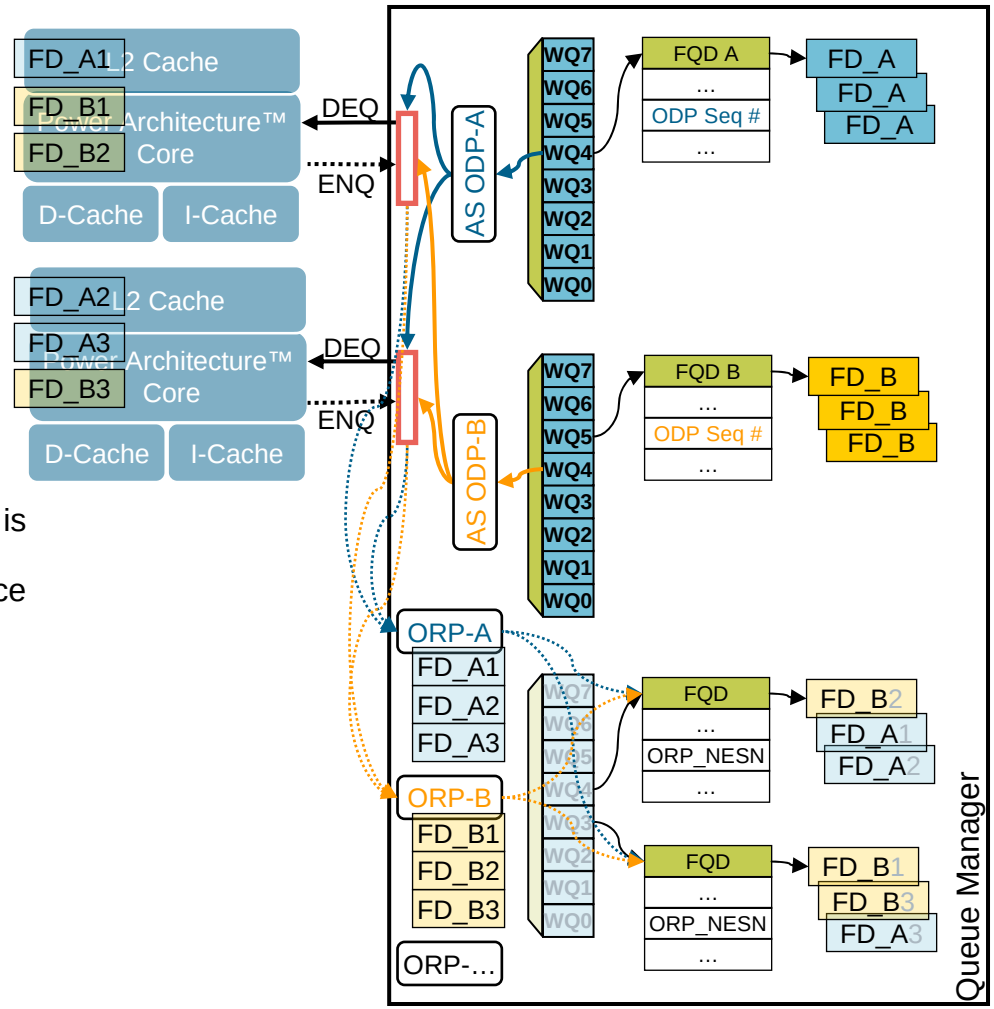


freescale semiconductor

QMan Order Restoration

- QMan's order restoration support has two components:
- Order Definition Point (ODP)
  - A point Defined relative sequence to each Frames pass
  - ODP id is associated to FQ-ID
  - assigning a monotonically increasing 14 bits sequence number to a series of frames
  - QM supports single ODP on head of queue
  - ODP can be made anywhere in the system i.e. SW can be an ODP
- Order Restoration Point (ORP)
  - A point relative sequence associated to single ODP is restored
  - Allows frames insertion into the flow (single sequence number with more/last indication)
- Behavior highlights
  - Configurable number of "in-flight" packets per ORP
  - resection of ORP is part of enqueue command but Queue tail is not associated to ORP i.e. enqueue to single destination queue can respect many ORP's

  Note: Frames are not "marked"



ODP-A

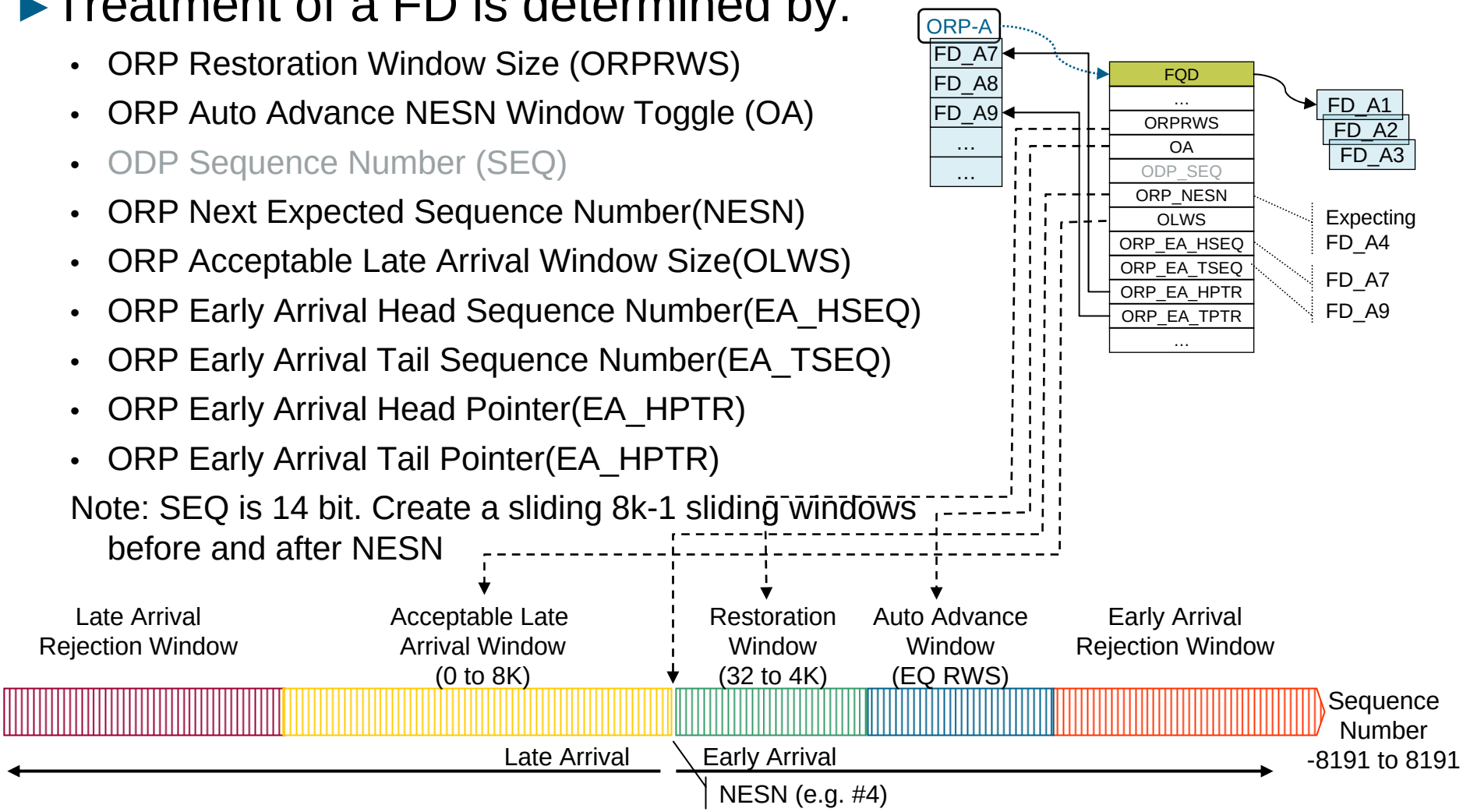
ODP-B

freescale semiconductor

Order Restoration Configuration

- Treatment of a FD is determined by:
  - ORP Restoration Window Size (ORPRWS)
  - ORP Auto Advance NESN Window Toggle (OA)
  - ODP Sequence Number (SEQ)
  - ORP Next Expected Sequence Number(NESN)
  - ORP Acceptable Late Arrival Window Size(OLWS)
  - ORP Early Arrival Head Sequence Number(EA_HSEQ)
  - ORP Early Arrival Tail Sequence Number(EA_TSEQ)
  - ORP Early Arrival Head Pointer(EA_HPTR)
  - ORP Early Arrival Tail Pointer(EA_HPTR)

  Note: SEQ is 14 bit. Create a sliding 8k-1 sliding windows before and after NESN



freescale semiconductor

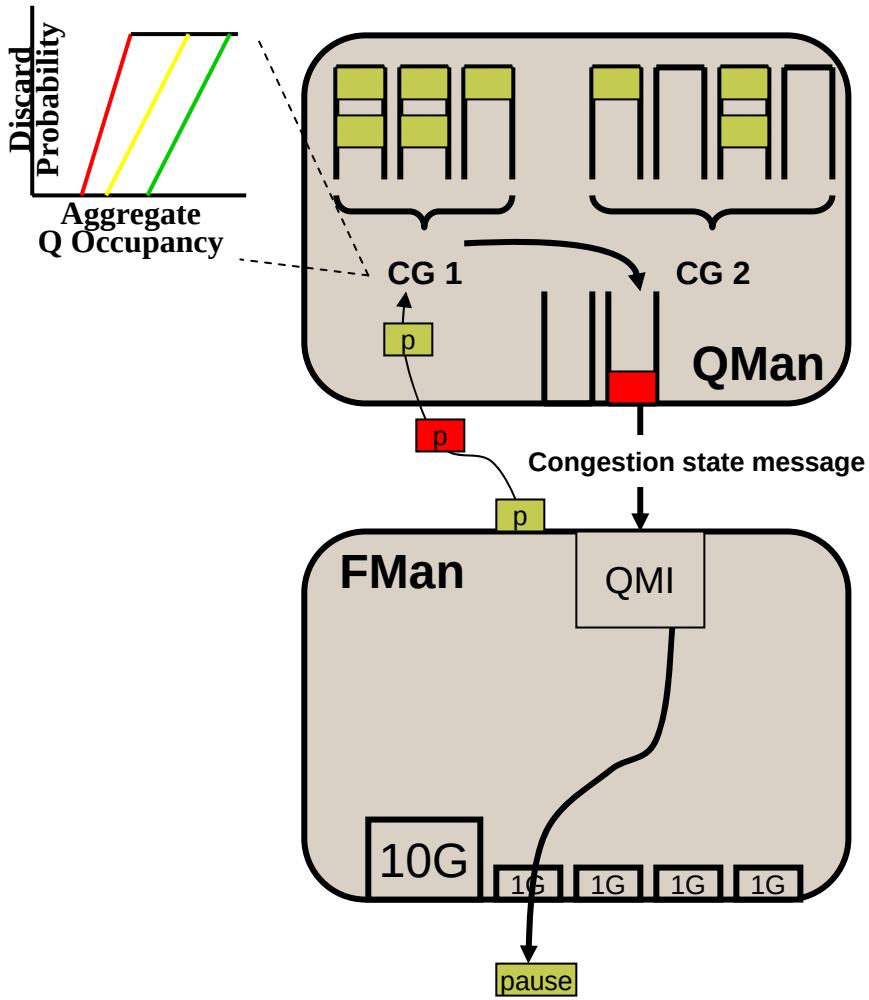
Agenda

- DPAA Overview
- FMAN Deep Dive
- QMAN Deep Dive
  - QMAN Building Blocks
  - QMAN Functions
  - QMAN Scheduling
  - Order Restoration, Order Preservation and Atomicity
  - Congestion management and avoidance
- BMAN Deep Dive
- SEC4.x Deep Dive
- PME Deep Dive

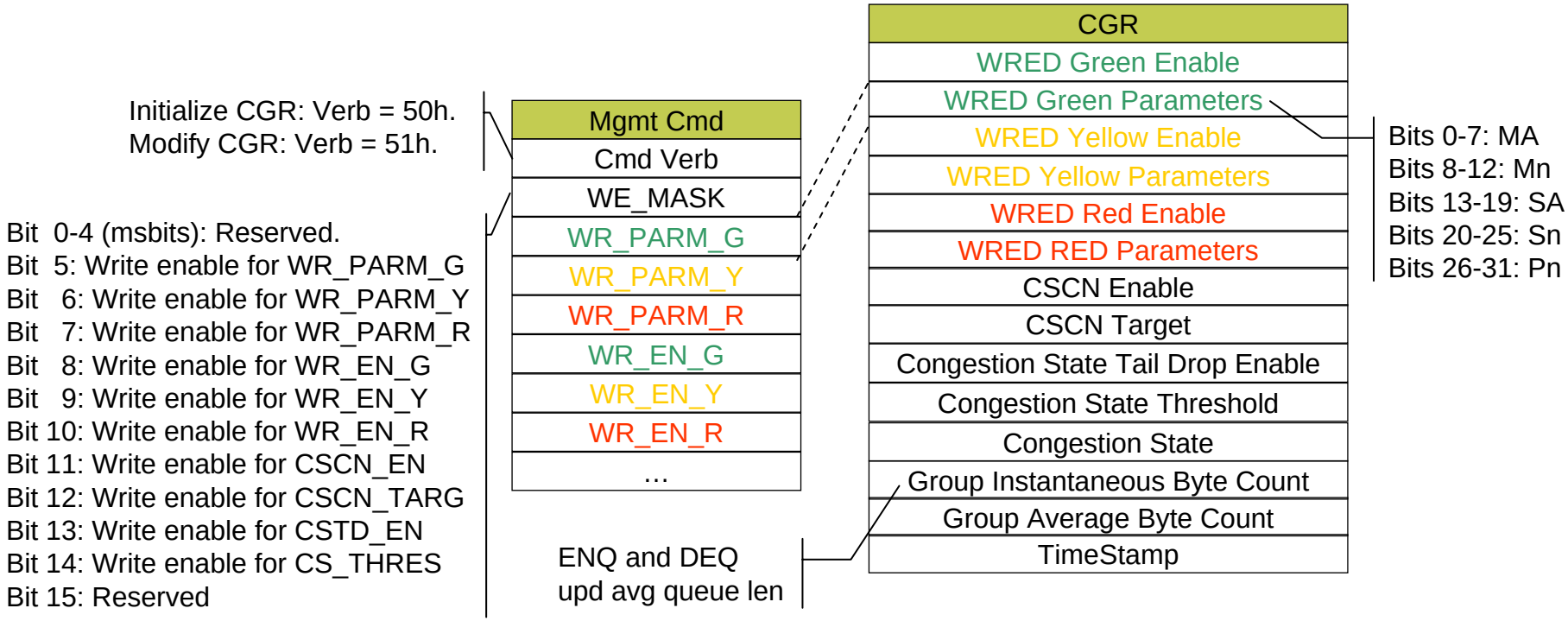
freescale semiconductor

Congestion Management and Avoidance

- **Both FM and QMan are involved in supporting congestion management and avoidance**
- **QMan provides the following support**
  - Congestion management (loss-less flow control, threshold/tail drop)
  - Congestion avoidance (RED/WRED)
- **Congestion Groups (CG) define granularity**
  - Every frame queue has a tail drop threshold and configured to a congestion group
  - Congestion calculations are done on groups of queues
  - Congestion avoidance/management calculations configured in the CG
- **Congestion Group Details**
  - 256 Congestion Groups
  - Time aware weighted average queue depth of all queues in the CG
  - 3 color configurable WRED curves
  - Enqueued packet may be rejected (discarded) due to WRED policy
  - Instantaneous CG depth +/- hysteresis value can initiate congestion state messages to enqueue sources
    - Lossless flow control on interfaces supporting PAUSE semantics



Setup Congestion Group Record



- On enqueue:
  - The color for the frame being enqueued is used to select a probability curve
  - The frame may be selected for random discard

Congestion Detection: Loss-less Flow Control and Tail Drop

- Enqueues and dequeues update an instantaneous queue length stored in the CG
  - This is actually the aggregate instantaneous length for all queues in the group
- On enqueue this instantaneous length is compared against a threshold
  - If configured the enqueue may be rejected
  - If it is greater than the threshold and the group is not already in congestion then it is entering congestion
  - If configured notification may be sent to one or more portals that the group has entered congestion
- On dequeue this instantaneous length is compared against the threshold minus a hysteresis value
  - If it is less than this value and the group is in congestion then the group has just exited congestion
  - If configured notification may be sent to a one or more portals that group has exited congestion
- FM must track congestion state changes and generate PAUSE frames to implement loss-less flow control

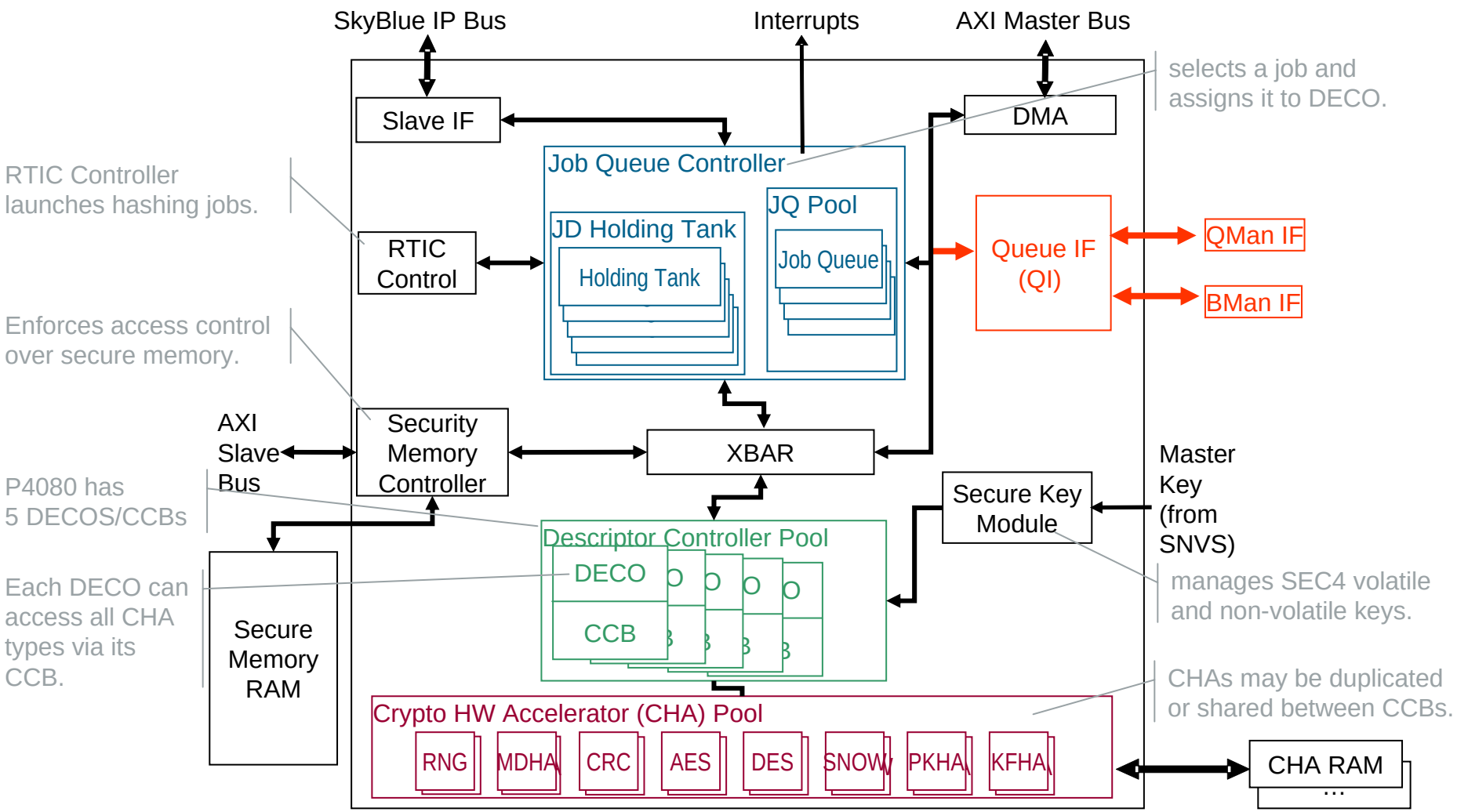
Agenda

- DPAA
- BMAN Deep Dive
- FMAN Deep Dive
- QMAN Deep Dive
- SEC Deep Dive
- PME Deep Dive
- Conclusion

SEC4.0 Features

- Public Key Cryptography
  - Modular Arithmetic
  - Elliptic Curve Cryptographic Functions
  - RSA Encryption, Decryption Public Key
- Cryptographic Authentication
  - MD5
  - SHA-1
  - SHA-224, SHA-256, SHA-384, SHA-512
  - Message Authentication Codes (MAC)
    - HMAC - all hashing algorithms
    - SSL 3.0 MAC (SMAC) - MD5, SHA-1 only
    - CMAC (AES)
    - XCBC-MAC (AES)
    - Kasumi f9
    - SNOW 3G f9
- Advanced Protocol Support
  - IPSec
  - SSL/TLS
  - SRTP
  - IEEE 802.11i WiFi
  - IEEE 802.16 WiMAX
  - IEEE 802.1AE MacSec / LinkSec
  - Support for protocol-specific padding
- Authenticated Encryption Algorithms
  - AES-CCM (Counter with CBC-MAC)
  - AES-GCM (Galois Counter Mode)
- Symmetric key block ciphers
  - AES (128, 192 or 256 bit keys)
  - DES (56 bit keys plus optional key parity)
  - 3DES (112 or 168 bit keys plus optional key parity)
  - Cipher modes
    - ECB, CBC, CFB, OFB for all block ciphers
    - CTR, XTS for AES
- Symmetric key stream ciphers
  - ArcFour (Alleged RC4, with 40 - 128 bit keys)
  - Kasumi f8
    support for 3GPP, ECSD/EDGE and GSM
  - Snow 3G f8
- Random Number Generation
  - NIST-compliant pseudo RNG
- Run Time Integrity Checking
  - SHA-256 message authentication
  - Segmented data gathering to support non-contiguous data blocks in memory
  - Support for up to 4 independent memory blocks

SEC4 Block Diagram

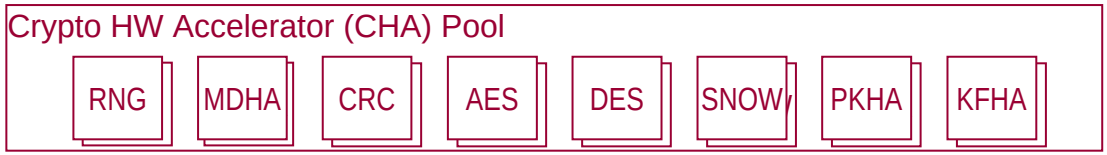


freescale semiconductor

SEC Functional Blocks

- Queue Interface
  - Enqueues and dequeues work from the Queue Manager
  - Acquire and free buffers from the Buffer Manager
  - Converts the information from DPAA data structures (e.g. Frame Descriptors, Shared Descriptors, and Buffers) into simpler Job Descriptors for processing by the lower levels of the SEC
- Job Queue Controller
  - SEC 4.0 Job Queue Controller has 4 Job Queues and 5 Job Descriptor Holding Tanks
  - The QI or Job Queues can put a Job Descriptor into any Hold Tank
  - Job Queues are analogous to the Crypto Channel Fetch FIFOs used in the SEC 3.x
  - Users can bypass the Queue Interface and place descriptor pointers on a Job Queue
- DEscriptor COntroller :
  - DECOs are analogous to the Crypto Channel State Machines in the SEC 3.x
    - DECO decodes Job Descriptors
    - DECO load keys, context, and data into the Crypto Hardware Accelerators
    - DECO process security protocol header & trailer processing and security session state maintenance

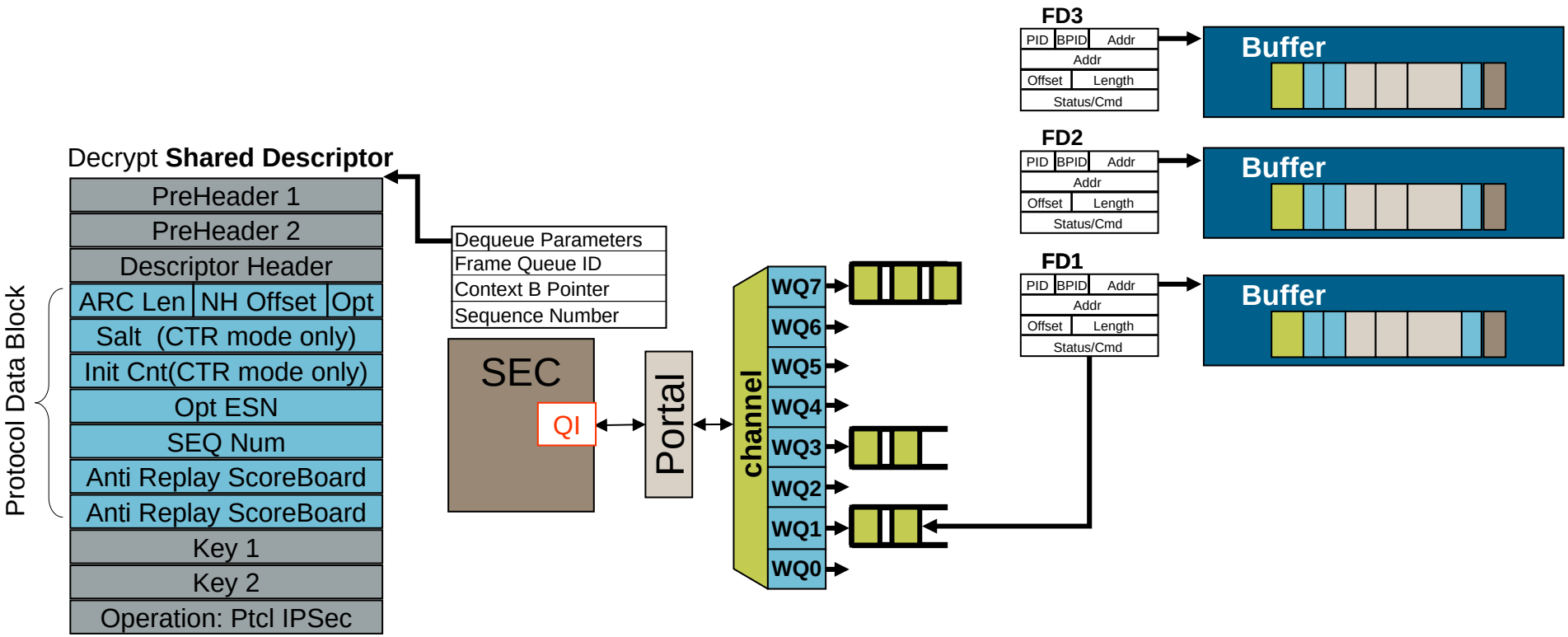
Crypto Hardware Accelerator (CHA)



- Modular and scalable with simplified device driver
- Header and trailer offload for the following security protocols:
  - IPSec, 802.1ae, SSL/TLS, SRTP, 802.11i, 802.16e
- Random number generator, random IV generation (RNG)
- Message Digest Hardware Accelerators (MDHA)
  - SHA-1, SHA-2 256-, 384-, 512-bit digests
  - MD5 128-bit digest
  - HMAC with all algorithms
- CRC Unit
  - CRC32, CRC32C, 802.16e OFDMA CRC
- Advanced Encryption Standard (AES)
  - Key lengths of 128-, 192-, and 256-bit
  - ECB, CBC, CTR, CCM, GCM, CMAC,
  - OFB, CFB, and XTS
- Data Encryption Standard (DES)
  - DES, 3DES (2K, 3K)
  - ECB, CBC, OFB modes
- Snow 3G Hardware Accelerators
  - Implements Snow 3.0
- Public Key Hardware Accelerators (PKHA)
  - RSA and Diffie-Hellman (to 4096b)
  - Elliptic curve cryptography (1023b)
  - Supports runtime equalization
- Kasumi/F8 Hardware Accelerators(KFHA)
  - F8, F9 as required for 3GPP
  - A5/3 for GSM and EDGE
  - GEA-3 for GPRS

freescale semiconductor

SEC 4.0 Inputs: DataPath Mode

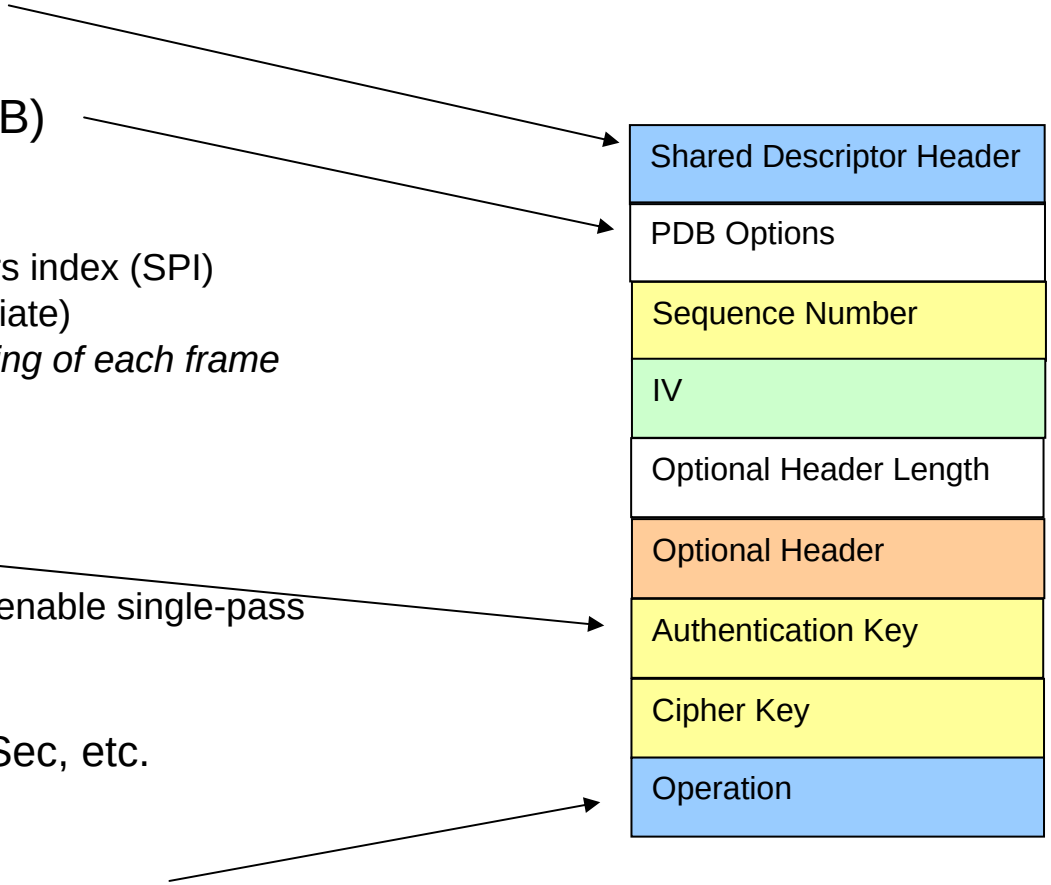


SEC 4.0 Inputs: Direct Mode

- Direct Mode bypasses the Queue Interface by creating a descriptor specific to a given operation (i.e. Job Descriptor)
  - It puts a pointer to the Job Descriptor on one of the Job Queue descriptor rings
  - Job Descriptors point to the data to be processed, and to a context block, which may be created for a single Job Descriptor, or it may be a Shared Descriptor that is used for multiple Job Descriptors
- Direct Mode can be used simultaneously with Datapath Mode
  - the SEC manages resources when work arrives on both the QI and JQ interfaces
  - E.g. control software can use Direct Mode for public key operations
- Direct Mode has higher software overheads than Datapath Mode,
  - these overheads are comparable of using the SEC 3.x and earlier cores
  - It is possible, but not practical, for more than 4 CPUs/multi-core partitions to interact with the SEC 4.x via the Job Queues
  - A typical scenario would be for CPUs/partitions with a need to perform ‘one-off’ jobs to be given a dedicated Job Queue. The interrupts from the dedicated JQ would only go to the ‘owning’ CPU/partition

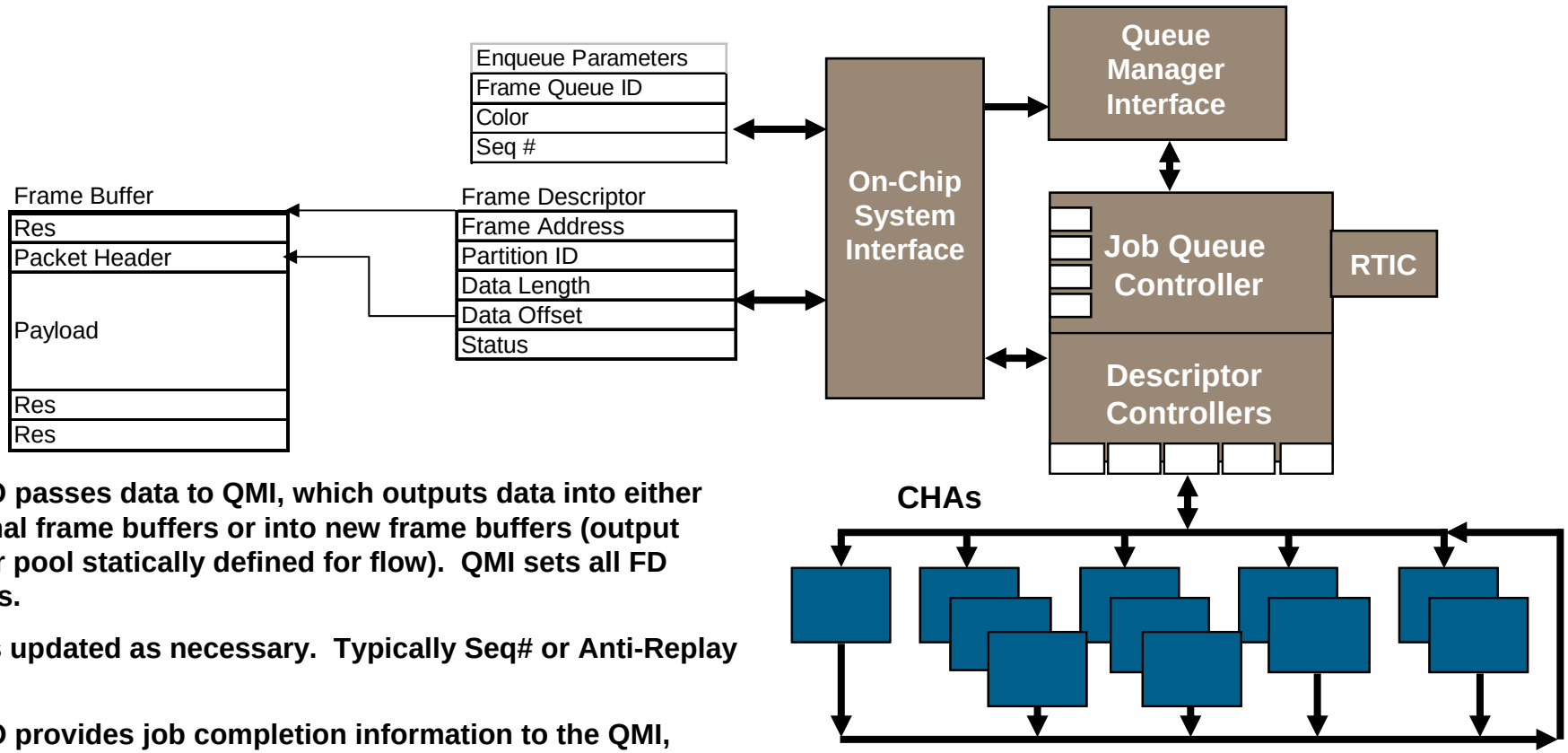
Shared Descriptor Example (single-pass ESP-CBC tunnel)

- Descriptor Header
  - Desc length, attributes
- Protocol Data Block (PDB)
  - Could hold:
    - Sequence Numbers
    - Association/Parameters index (SPI)
    - Blockcipher IV (immediate)
    - *Updated after processing of each frame (where appropriate)*
- Key Blocks
  - HMAC Key (Class 2)
  - Cipher Key (Class 1)
    - Both classes together enable single-pass operation
- Protocol Operation
  - e.g. WiMax, 802.11, IPSec, etc.



freescale semiconductor

SEC 4.0 Outputs



DECO passes data to QMI, which outputs data into either original frame buffers or into new frame buffers (output buffer pool statically defined for flow). QMI sets all FD values.

Ctx is updated as necessary. Typically Seq# or Anti-Replay state.

DECO provides job completion information to the QMI, which uses the status word in the frame descriptor to inform software of success or failure.

When last data is processed by an EU, DECO releases EUs and next DECO grabs them.

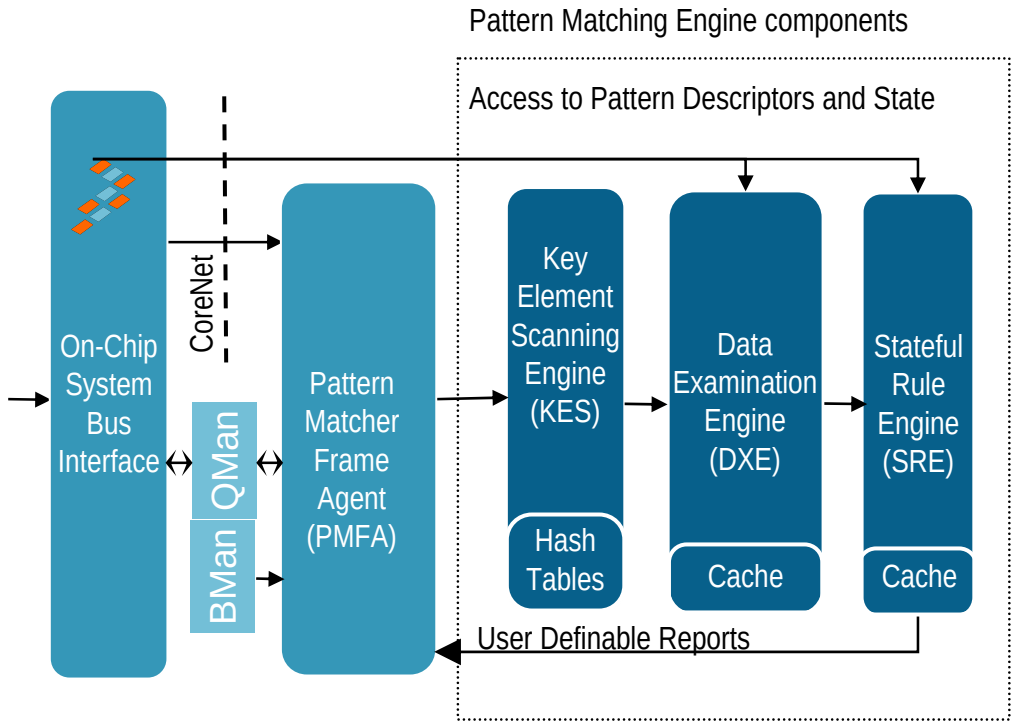
freescale semiconductor

Agenda

- DPAA
- BMAN Deep Dive
- FMAN Deep Dive
- QMAN Deep Dive
- SEC Deep Dive
- PME Deep Dive
- Conclusion

Pattern Matching Engine (PME) 2.x Overview

- Regex support plus significant extensions:
  - Patterns can be split into 256 sets each of which can contain 16 subsets
  - 32K patterns of up to 128B length
  - 9.6 Gbps raw performance
- Combined hash/NFA technology
  - No “explosion” in number of patterns due to wildcards
  - Low system memory utilization
  - Fast pattern database compiles and incremental updates
- Matching across “work units” finds patterns in streamed data
- The Pattern Matching Engine utilizes a pipeline of processing blocks to provide a complete pattern matching solution



Frame Manager Interface (PME 2.X)

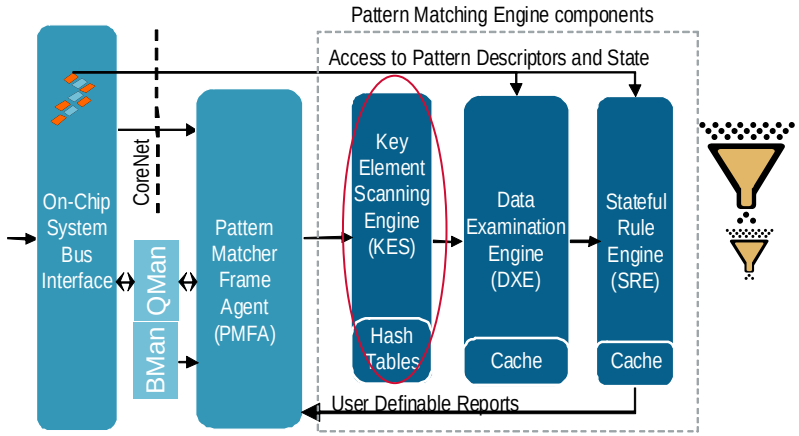
- Pattern Matcher Frame Agent (PMFA)
  - Provides an interface to the Frame Queue datapath environment (Queue Manager and Buffer Manager) to receive work requests (scan data and control messages) and to send notifications/reports (if required) of completed work requests
- Frame Queue
  - A Frame Queue represents IO data to and from the Pattern Matcher
  - The input data typically corresponds to the data of a flow to be acted on by the Pattern Matcher
  - The output data corresponds to the pattern matching reports
- Exclusive Frame Queue Control (EFQC)
  - Force the PMFA block to service a specified Frame Queue exclusively
  - This capability allows a batch of Pattern Matcher configuration commands (read, write, clear session context) that spans multiple Frames to be treated atomically by the Pattern Matcher

Input Frame Processing (PME 2.X)

- Direct Access Mode
  - For single input frame
  - No Flow Context
  - No residue to be applied across Frames
  - No Stateful Rule
  - The special CLIM (Compare Limit) and MLIM (Match Limit) values of 0xFFFF will be applied by the DXE, indicating no compare or match limit
- Flow Mode
  - For compound input frame
  - The Flow Context contains information related to input/output data handling and optionally a Stateful Rule Engine Session ID
  - Supports an inter-Frame sequence number and optional residue data handling
  - Support the Stateful Rule Engine
  - The FCP memory location may be cached internally by the Pattern Matcher

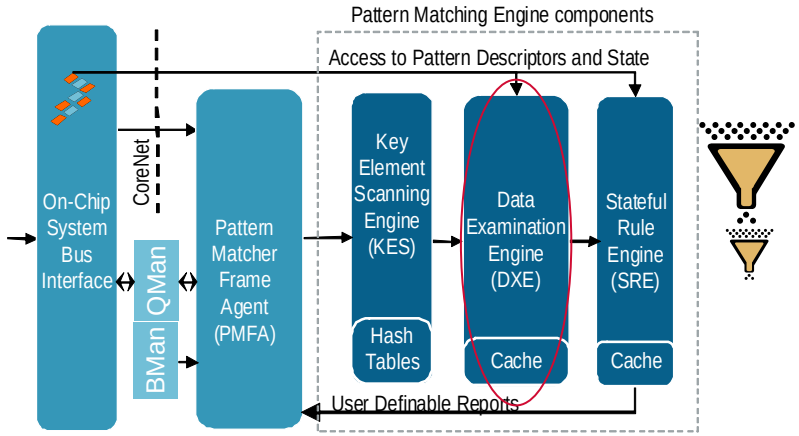
PM 1/3: Key Element Scanning Engine (KES)

- KES scans for possible matches and filters work to be performed by DXE
- All work performed using on-chip hash tables – no external memory access required
- Creates multiple formats of each incoming data byte
  - Original, translated to equivalent, pre-defined category, user-defined category
- Computes a hash for different fingerprint lengths and looks up hash tables
- A “hit” on one of these hashes results in a second level hash (“confidence” hash) being performed
  - If all levels of hash “hit” then data window is passed to data examination engine
- Scan engine continues as data examination engine checks “possible” matches for a definite match



PM 2/3: Data Examination Engine (DXE)

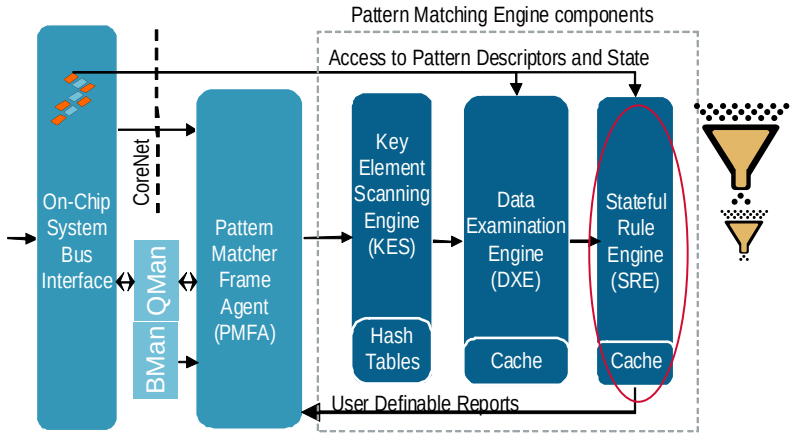
- DXE invoked only when needed to confirm a match
- Performs complete match for each “possible” match found by KES
- Pattern definitions stored in DRAM
  - Incremental updates only affect the changed pattern records
  - Implements a significant subset of the regex pattern definition syntax plus many constructs which cannot be expressed in regex.
    - Supports Perl meta-characters and Freescale extensions (for capture)
    - Content can be extracted from data for later comparison
    - ASCII-encoded length fields and counts embedded in data can be extracted and converted to numeric form for later use



freescale semiconductor

PM 3/3: Stateful Rule Engine (SRE)

- Stateful rules are FSM that operate on a per session basis
- User-defined logic reacts to pattern matches detected by the DXE



- A session is defined here as a logical grouping of one or more flows. e.g. a pair of flows representing both directions of communication in a connection
- Can be used to further qualify the pattern match. For example:
  - all patterns making up the signature are found, or
  - matched only within a certain portion of the data (e.g. URL), or
  - matched only within a certain portion of the data within the data unit
- Other uses:
  - Protocol state tracking (e.g. track the “normal” transitions of SMTP)
  - Provide support for “greedy” wildcards (e.g. ABC.*DEF i.e. two patterns tied together by a stateful rule)

Agenda

- DPAA
- BMAN Deep Dive
- FMAN Deep Dive
- QMAN Deep Dive
- SEC Deep Dive
- PME Deep Dive
- Conclusion

Conclusion

- The QorIQ™ Datapath Acceleration Architecture components include:
  - Queue Manager
  - Buffer Manager
  - Frame Manager
  - Hardware accelerators such as SEC and PME
  - Cores
- Together these components address multicore requirements including:
  - Load spreading
  - Packet ordering
  - Device virtualization
  - Inter-core communication
  - HW buffer management


freescale™
semiconductor