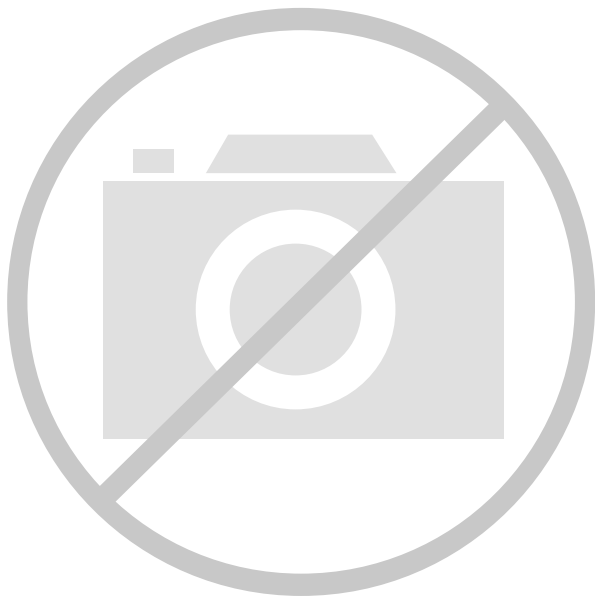




Welcome to [E-XFL.COM](https://www.e-xfl.com)

Understanding [Embedded - Microprocessors](#)

Embedded microprocessors are specialized computing chips designed to perform specific tasks within an embedded system. Unlike general-purpose microprocessors found in personal computers, embedded microprocessors are tailored for dedicated functions within larger systems, offering optimized performance, efficiency, and reliability. These microprocessors are integral to the operation of countless electronic devices, providing the computational power necessary for controlling processes, handling data, and managing communications.

Applications of [Embedded - Microprocessors](#)

Embedded microprocessors are utilized across a broad spectrum of applications, making them indispensable in

Details

Product Status	Active
Core Processor	-
Number of Cores/Bus Width	-
Speed	-
Co-Processors/DSP	-
RAM Controllers	-
Graphics Acceleration	-
Display & Interface Controllers	-
Ethernet	-
SATA	-
USB	-
Voltage - I/O	-
Operating Temperature	-
Security Features	-
Package / Case	-
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=t4240nse7qtb

2 Summary of benefits

The T4 family of processors are ideal for combined control and data plane processing. A wide variety of applications can benefit from the processing, I/O integration, and power management capabilities. Similar to other QorIQ devices, the T4 family of processors' high level of integration offers significant space, weight, and power benefits compared to multiple discrete devices. Examples include:

- Service provider networking: RNC, metro networking, gateway, core/edge router, EPC, CRAN, ATCA, and AMC solutions.
- Enterprise equipment: router, switch services, and UTM appliances.
- Data centers: NFV, SDN, ADC, WOC, UTM, proxy, server appliance, and PCI Express (PCIe) offload.
- Storage controllers: FCoE bridging, iSCSI controller, and SAN controller.
- Aerospace, defense, and government: radar imaging, ruggedized network appliance, and cockpit display.
- Industrial computing: single-board computers and test equipment.

2.1 e6500 CPU core

The T4 family of processors are based on the Power Architecture® e6500 core. The e6500 core uses a seven-stage pipeline for low latency response while also boosting single-threaded performance. The e6500 core also offers high aggregate instructions per clock at lower power with an innovative "fused core" approach to threading. The e6500 core's fully resourced dual threads provide 1.7 times the performance of a single thread.

The e6500 cores are clustered in banks of four cores sharing a 2 MB L2 cache, allowing efficient sharing of code and data within a multicore cluster. Each e6500 core implements the Freescale AltiVec technology SIMD engine, dramatically boosting performance of heavy math algorithms with DSP-like performance.

The e6500 core features include:

- Up to 1.8 GHz dual threaded operation
- 7 DMIPS/MHz per core
- Advanced power saving modes, including state retention power gating

2.2 Virtualization

The T4 family of processors includes support for hardware-assisted virtualization. The e6500 core offers an extra core privilege level (hypervisor) and hardware offload of logical-to-real address translation. In addition, the T4 family of processors includes platform-level enhancements supporting I/O virtualization with DMA memory protection through IOMMUs and configurable "storage profiles" that provide isolation of I/O buffers between guest environments. Virtualization software for the T4 family includes kernel virtualization machine (KVM), Linux containers, and Freescale hypervisor and commercial virtualization software from vendors such as Enea®, Greenhills Software®, Mentor Graphics®, and Wind River.

2.3 Data Path Acceleration Architecture (DPAA)

The T4 family of processors enhance the QorIQ DPAA, an innovative multicore infrastructure for scheduling work to cores (physical and virtual), hardware accelerators, and network interfaces.

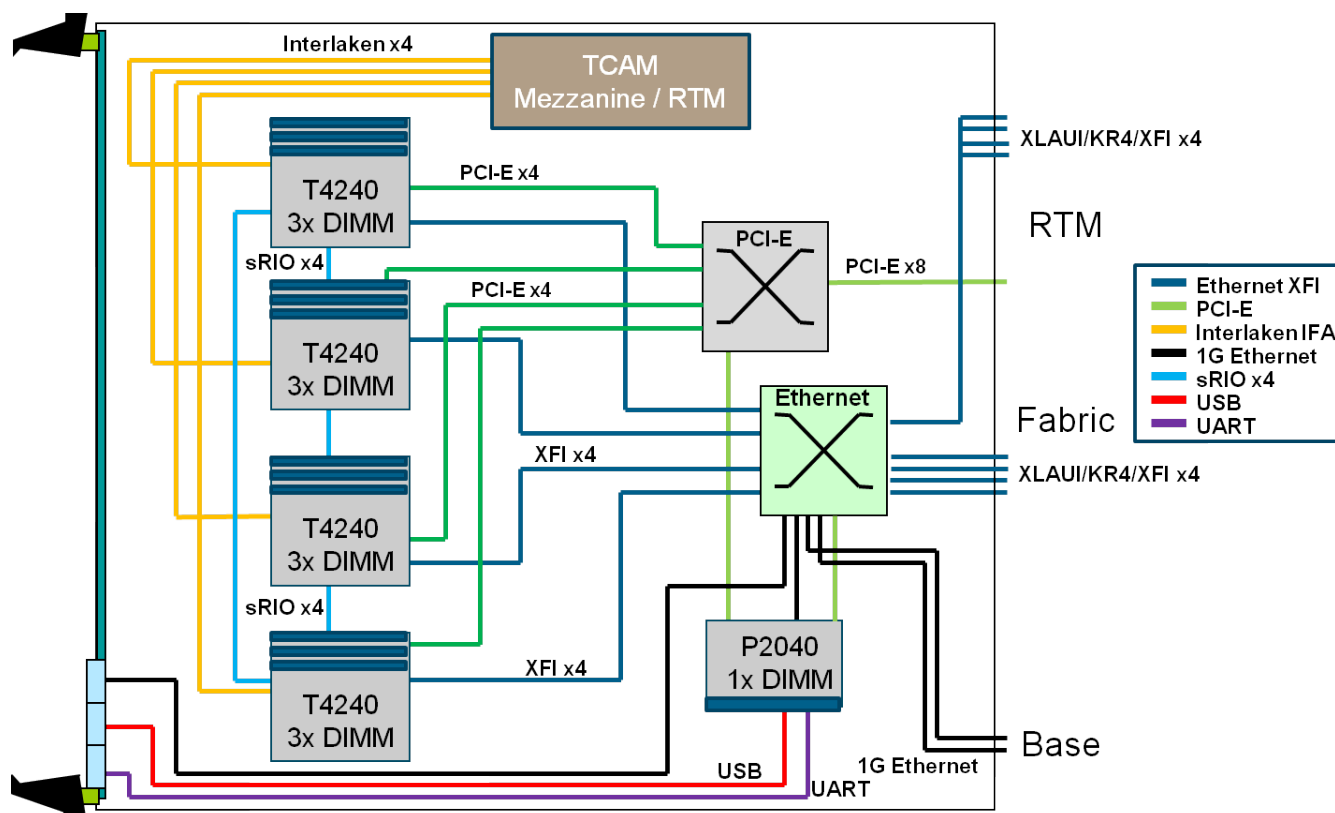


Figure 2. Network services ATCA blade

3.3 Radio node controller

Some of the more demanding packet-processing applications are found in the realm of wireless infrastructure. These systems have to interwork between wireless link layer protocols and IP networking protocols. Wireless protocol complexity is high, and includes scheduling, retransmission, and encryption with algorithms specific to cellular wireless access networks. Connecting to the IP network offers wireless infrastructure tremendous cost savings, but introduces all the security threats found in the IP world. The chip's network and peripheral interfaces provide it with the flexibility to connect to DSPs, and to wireless link layer framing ASICs/FPGAs (not shown). While the Data Path Acceleration Architecture offers encryption acceleration for both wireless and IP networking protocols, in addition to packet filtering capability on the IP networking side, multiple virtual CPUs may be dedicated to data path processing in each direction.

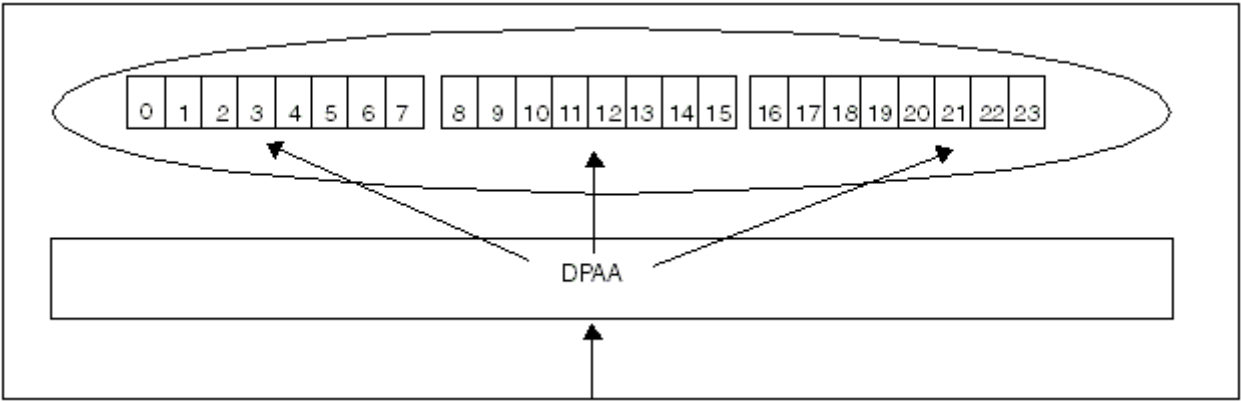


Figure 5. 24 vCPU AMP or SMP with affinity

4.2 Symmetric multiprocessing

Figure 5 also presents 24 vCPU SMP, where it is typical for data processing to involve some level of task affinity.

4.3 Mixed symmetric and asymmetric multiprocessing

This figure shows one possibility for a mixed SMP and AMP processing. Two physical CPUs (vCPUs 0-3) are combined in an SMP cluster for control processing, with the Datapath using exact match classification to send only control packets to the SMP cluster. The remaining virtual cores could run 20 instances of datapath software.

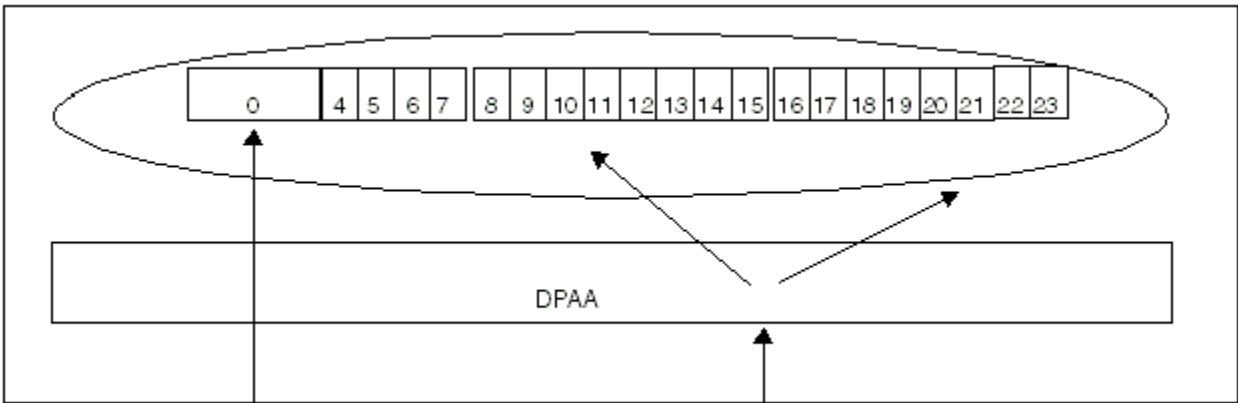


Figure 6. Mixed SMP and AMP option 1

This figure shows another possibility for mixed SMP and AMP processing. Two of the physical cores are run in single threaded mode; the remaining physical cores operate as four virtual CPUs. The Datapath directs traffic to specific software partitions based on physical Ethernet port, classification, or some combination.

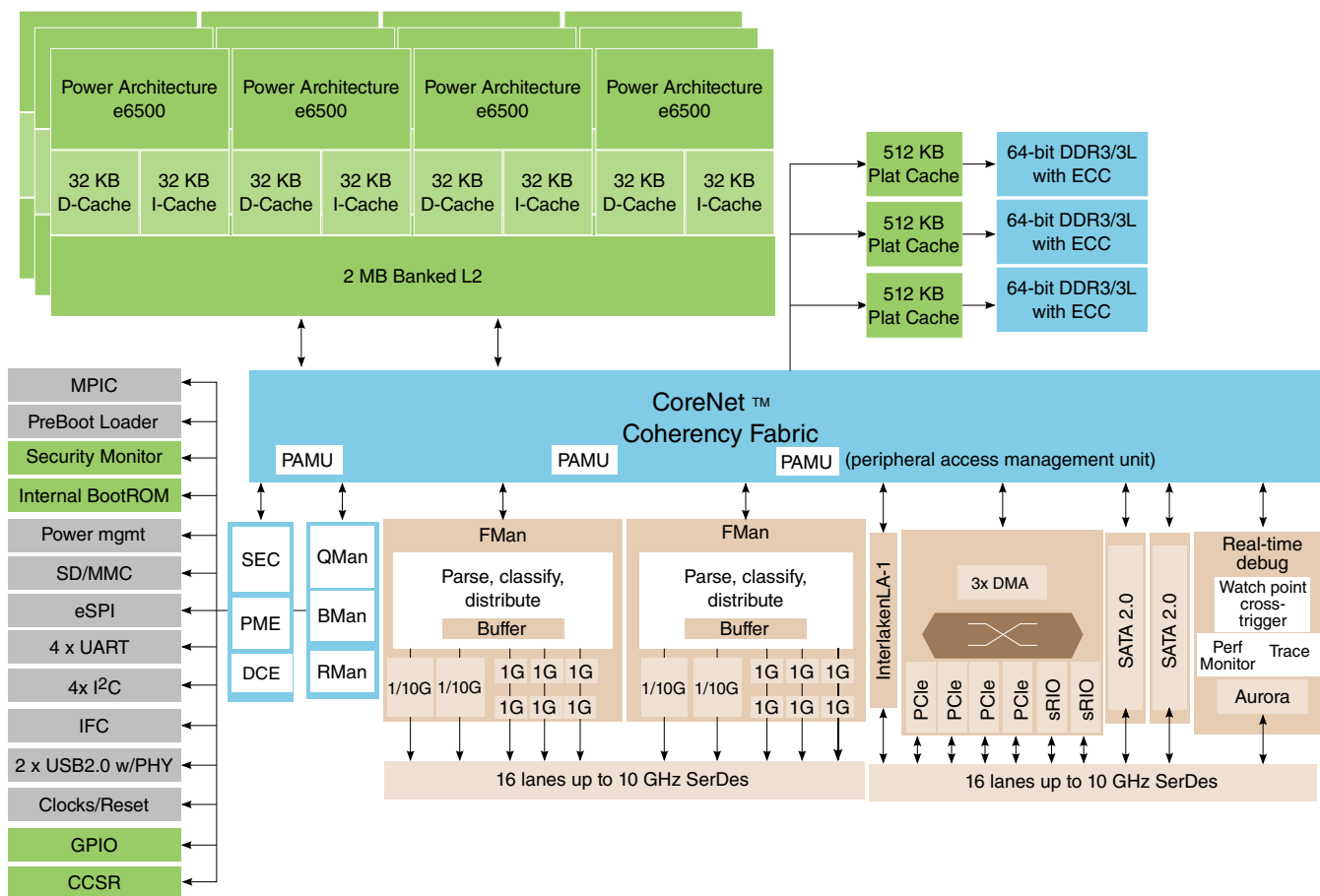


Figure 8. T4240 block diagram

5.2 Features summary

This chip includes the following functions and features:

- 12, dual-threaded e6500 cores for a total of 24/16/8 threads (T4240/T4160/T4080) built on Power Architecture® technology
 - Arranged as three clusters of four cores sharing a 2 MB L2 cache, 6 MB L2 cache total.
 - Up to 1.8 GHz with 64-bit ISA support (Power Architecture v2.06-compliant)
 - Three privilege levels of instruction: user, supervisor, and hypervisor
- Up to 1.5 MB CoreNet Platform Cache (CPC)
- Hierarchical interconnect fabric
 - CoreNet fabric supporting coherent and non-coherent transactions with prioritization and bandwidth allocation amongst CoreNet end-points
 - 1.46 Tbps coherent read bandwidth
- Up to three 64-bit DDR3/3L SDRAM memory controllers with ECC and interleaving support
 - Up to 1.867 GT/s data transfer rate
 - 64 GB per DDR controller
- Data Path Acceleration Architecture (DPAA) incorporating acceleration for the following functions:
 - Packet parsing, classification, and distribution (Frame Manager 1.1) up to 50 Gbps
 - Queue management for scheduling, packet sequencing, and congestion management (Queue Manager 1.1)
 - Queue Manager (QMan) fabric supporting packet-level queue management and quality of service scheduling
 - Hardware buffer management for buffer allocation and de-allocation (BMan 1.1)
 - Cryptography acceleration (SEC 5.0) at up to 40 Gbps

5.7.1.1 DDR bandwidth optimizations

Multicore SoCs are able to increase CPU and network interface bandwidths faster than commodity DRAM technologies are improving. As a result, it becomes increasingly important to maximize utilization of main memory interfaces to avoid a memory bottleneck. The T4 family's DDR controllers are Freescale-developed IP, optimized for the QorIQ SoC architecture, with the goal of improving DDR bandwidth utilization by fifty percent when compared to first generation QorIQ SoCs.

Most of the WRITE bandwidth improvement and approximately half of the READ bandwidth improvement is met through target queue enhancements; in specific, changes to the scheduling algorithm, improvements in the bank hashing scheme, support for more transaction re-ordering, and additional proprietary techniques.

The remainder of the READ bandwidth improvement is due to the addition of an intelligent data prefetcher in the memory subsystem.

5.7.1.2 Prefetch Manager (PMan)

NOTE

All transactions to DDR pass through the CPC; this means the CPC can miss (and trigger prefetching) even on data that is not intended for allocation into the CPC.

The PMAN monitors CPC misses for opportunities to prefetch, using a "confidence"-based algorithm to determine its degree of aggressiveness. It can be configured to monitor multiple memory regions (each of different size) for prefetch opportunities. Multiple CPC misses on accesses to a tracked region for consecutive cache blocks increases confidence to start prefetching, and a CPC miss of a tracked region with same stride will instantly cause prefetching.

The PMan uses feedback to increase or decrease its aggressiveness. When the data it prefetches is being used, it prefetches further ahead. If the request stride length changes or previously prefetched data isn't consumed, prefetching slows or stops (at least for that region/requesting device/transaction type).

5.7.2 PreBoot Loader and nonvolatile memory interfaces

The PreBoot Loader (PBL) operates similarly to an I²C boot sequencer but on behalf of a large number of interfaces.

It supports IFC, I²C, eSPI, eSDHC.

The PBL's functions include the following:

- Simplifies boot operations, replacing pin strapping resistors with configuration data loaded from nonvolatile memory
- Uses the configuration data to initialize other system logic and to copy data from low speed memory interfaces (I²C, IFC, eSPI, and SD/MMC) into fully initialized DDR or the 2 MB front-side cache

5.7.2.1 Integrated Flash Controller

The SoC incorporates an Integrated Flash Controller similar to the one used in some previous generation QorIQ SoCs. The IFC supports both NAND and NOR flash, as well as a general purpose memory mapped interface for connecting low speed ASICs and FPGAs.

5.7.2.1.1 NAND Flash features

- x8/x16 NAND Flash interface
- Optional ECC generation/checking
- Flexible timing control to allow interfacing with proprietary NAND devices
- SLC and MLC Flash devices support with configurable page sizes of up to 4 KB
- Support advance NAND commands like cache, copy-back, and multiplane programming

Chip features

- Boot chip-select (CS0) available after system reset, with boot block size of 8 KB, for execute-in-place boot loading from NAND Flash
- Up to terabyte Flash devices supported

5.7.2.1.2 NOR Flash features

- Data bus width of 8/16/32
- Compatible with asynchronous NOR Flash
- Directly memory mapped
- Supports address data multiplexed (ADM) NOR device
- Flexible timing control allows interfacing with proprietary NOR devices
- Boot chip-select (CS0) available at system reset

5.7.2.1.3 General-purpose chip-select machine (GPCM)

The IFC's GPCM supports the following features:

- Normal GPCM
 - Support for x8/16/32-bit device
 - Compatible with general purpose addressable device, for example, SRAM and ROM
 - External clock is supported with programmable division ratio (2, 3, 4, and so on, up to 16)
- Generic ASIC Interface
 - Support for x8/16/32-bit device
 - Address and Data are shared on I/O bus
 - Following address and data sequences are supported on I/O bus:
 - 32-bit I/O: AD
 - 16-bit I/O: AADD
 - 8-bit I/O: AAAADDDD

5.7.2.2 Serial memory controllers

In addition to the parallel NAND and NOR flash supported by the IFC, the SoC supports serial flash using eSPI, I²C and SD/MMC/eMMC card and device interfaces. The SD/MMC/eMMC controller includes a DMA engine, allowing it to move data from serial flash to external or internal memory following straightforward initiation by software.

Detailed features of the eSDHC include the following:

- Conforms to the SD Host Controller Standard Specification version 2.0, including Test event register support
- Compatible with the MMC System Specification version 4.2
- Compatible with the SD Memory Card Specification version 2.0, and supports the high capacity SD memory card
- Designed to work with SD memory, SD combo, MMC, and their variants like mini and micro.
- Card bus clock frequency up to 52 MHz
- Supports 1-/4-bit SD, 1-/4-/8-bit MMC modes
- Supports single-block and multi-block read, and write data transfer
- Supports block sizes of 1-2048 bytes
- Supports the mechanical write protect detection. In the case where write protect is enabled, the host will not initiate any write data command to the card
- Supports both synchronous and asynchronous abort
- Supports pause during the data transfer at block gap
- Supports Auto CMD12 for multi-block transfer
- Host can initiate command that do not use data lines, while data transfer is in progress
- Embodies a configurable 128x32-bit FIFO for read/write data
- Supports SDMA, ADMA1, and ADMA2 capabilities

5.9.2 Serial RapidIO

The Serial RapidIO interface is based on the *RapidIO Interconnect Specification, Revision 2.1*. RapidIO is a high-performance, point-to-point, low-pin-count, packet-switched system-level interconnect that can be used in a variety of applications as an open standard. The rich feature set includes high data bandwidth, low-latency capability, and support for high-performance I/O devices as well as message-passing and software-managed programming models. Receive and transmit ports operate independently, and with 2 x 4 Serial RapidIO controllers, the aggregate theoretical bandwidth is 32 Gbps.

The chip offers two Serial RapidIO controllers, muxed onto the SerDes blocks. The Serial RapidIO interface is based on the *RapidIO Interconnect Specification, Revision 2.1*. Receive and transmit ports operate independently and with 2 x 4 Serial RapidIO controllers; the aggregate theoretical bandwidth is 32 Gbps. The Serial RapidIO controllers can be used in conjunction with "Rapid IO Message Manager (RMAN), as described in [RapidIO Message Manager \(RMan\)](#)."

Key features of the Serial RapidIO interface unit include the following:

- Support for *RapidIO Interconnect Specification, Revision 2.1* (All transaction flows and priorities.)
- 2x, and 4x LP-serial link interfaces, with transmission rates of 2.5, 3.125, or 5.0 Gbaud (data rates of 1.0, 2.0, 2.5, or 4.0 Gbps) per lane
- Auto-detection of 1x, 2x, or 4x mode operation during port initialization
- 34-bit addressing and up to 256-byte data payload
- Support for SWRITE, NWRITE, NWRITE_R and Atomic transactions
- Receiver-controlled flow control
- RapidIO error injection
- Internal LP-serial and application interface-level loopback modes

The Serial RapidIO controller also supports the following capabilities, many of which are leveraged by the RMan to efficient chip-to-chip communication through the DPAA:

- Support for RapidIO Interconnect Specification 2.1, "Part 2: Message Passing Logical Specification"
- Supports RapidIO Interconnect Specification 2.1, "Part 10: Data Streaming Logical Specification"
- Supports RapidIO Interconnect Specification 2.1, "Annex 2: Session Management Protocol"
 - Supports basic stream management flow control (XON/XOFF) using extended header message format
- Up to 16 concurrent inbound reassembly operations
 - One additional reassembly context is reservable to a specific transaction type
- Support for outbound Type 11 messaging
- Support for outbound Type 5 NWRITE and Type 6 SWRITE transactions
- Support for inbound Type 11 messaging
- Support for inbound Type 9 data streaming transactions
- Support for outbound Type 9 data streaming transactions
 - Up to 64 KB total payload
- Support for inbound Type 10 doorbell transactions
 - Transaction steering through doorbell header classification
- Support for outbound Type 10 doorbell transactions
 - Ordering can be maintained with respect to other types of traffic.
- Support for inbound and outbound port-write transactions
 - Data payloads of 4 to 64 bytes

5.9.3 SATA

Each of the SoC's two SATA controllers is compliant with the *Serial ATA 2.6 Specification*. Each of the SATA controllers has the following features:

- Supports speeds: 1.5 Gbps (first-generation SATA), and 3Gbps (second-generation SATA)
- Supports advanced technology attachment packet interface (ATAPI) devices
- Contains high-speed descriptor-based DMA controller
- Supports native command queuing (NCQ) commands

- Up to 20 Gbps for IPsec @ Imix
- 10x performance increase for public key algorithms
- Support for 3GPP Confidentiality and Integrity Algorithms 128-EEA3 & 128-EIA3 (ZUC)
- DCE 1.0, new accelerator for compression/decompression
- RMan (Serial RapidIO Manager)
- DPAA overall capabilities
 - Data Center Bridging
 - Egress Traffic Shaping

5.10.4 DPAA terms and definitions

The QorIQ Platform's Data Path Acceleration Architecture (henceforth DPAA) assumes the existence of network flows, where a flow is defined as a series of network datagrams, which have the same processing and ordering requirements. The DPAA prescribes data structures to be initialized for each flow. These data structures define how the datagrams associated with that flow move through the DPAA. Software is provided a consistent interface (the software portal) for interacting with hardware accelerators and network interfaces.

All DPAA entities produce data onto frame queues (a process called enqueueing) and consume data from frame queues (dequeuing). Software enqueues and dequeues through a software portal (each vCPU has two software portals), and the FMan, RMan, and DPAA accelerators enqueue/dequeue through hardware portals. This figure illustrates this key DPAA concept.

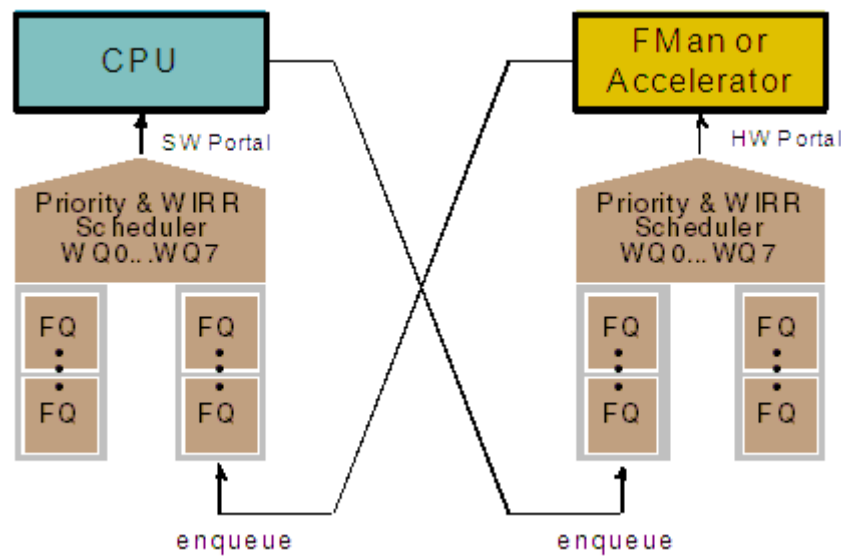


Figure 9. DPAA enqueueing and dequeuing

This table lists common DPAA terms and their definitions.

Table 5. DPAA terms and definitions

Term	Definition	Graphic representation
Buffer	Region of contiguous memory, allocated by software, managed by the DPAA BMan	

Table continues on the next page...

5.10.5.1.1 Receiver functionality: parsing, classification, and distribution

Each Frame Manager matches its 25 Gbps Ethernet connectivity with 25 Gbps (37.2 Mpps) of Parsing, Classification, and Distribution (PCD) performance. PCD is the process by which the Frame Manager identifies the frame queue on which received packets should be enqueued. The consumer of the data on the frame queues is determined by Queue Manager configuration; however, these activities are closely linked and managed by the FMan Driver and FMan Configuration Tool, as in previous QorIQ SoCs.

This figure provides a logical view of the FMan's processing flow, illustrating the PCD features.

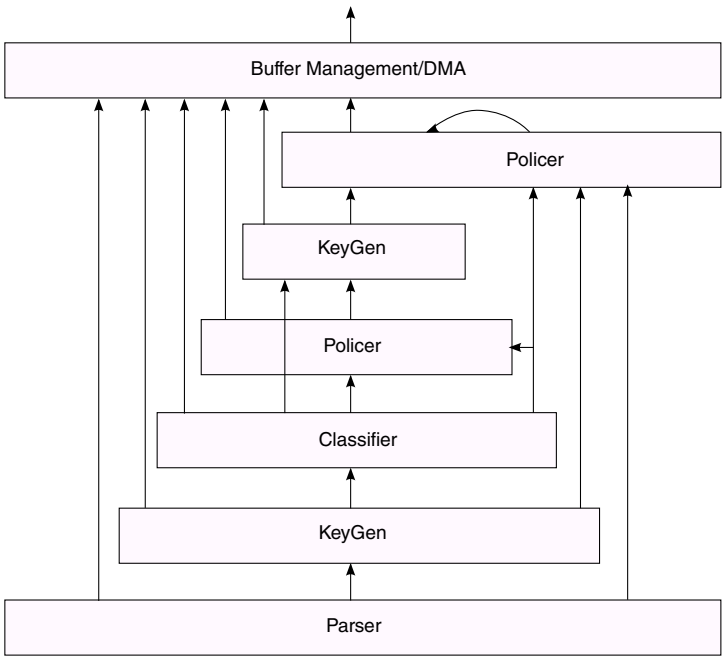


Figure 11. Logical view of FMan processing

Each frame received by the FMan is buffered internally while the Parser, KeyGen, and Classification functions operate.

The parse function can parse many standard protocols, including options and tunnels, and it supports a generic configurable capability to allow proprietary or future protocols to be parsed. Hard parsing of the standard protocol headers can be augmented with user-defined soft parsing rules to handle proprietary header fields. Hard and soft parsing occurs at wire speed.

This table defines several types of parser headers.

Table 6. Parser header types

Header type	Definition
Self-describing	Announced by proprietary values of Ethertype, protocol identifier, next header, and other standard fields. They are self-describing in that the frame contains information that describes the presence of the proprietary header.
Non-self-describing	Does not contain any information that indicates the presence of the header.

Table continues on the next page...

The SEC 5.0 can perform full protocol processing for the following security protocols:

- IPsec
- SSL/TLS
- 3GPP RLC encryption/decryption
- LTE PDCP
- SRTP
- IEEE 802.1AE MACSec
- IEEE 802.16e WiMax MAC layer

The SEC 5.0 supports the following algorithms, modes, and key lengths as raw modes, or in combination with the security protocol processing described above.

- Public Key Hardware Accelerators (PKHA)
 - RSA and Diffie-Hellman (to 4096b)
 - Elliptic curve cryptography (1023b)
- Data Encryption Standard Accelerators (DESA)
 - DES, 3DES (2-key, 3-key)
 - ECB, CBC, OFB, and CFB modes
- Advanced Encryption Standard Accelerators (AESA)
 - Key lengths of 128-bit, 192-bit, and 256-bit
 - ECB, CBC, CTR, CCM, GCM, CMAC, OFB, CFB, xcbc-mac, and XTS
- ARC Four Hardware Accelerators (AFHA)
 - Compatible with RC4 algorithm
- Message Digest Hardware Accelerators (MDHA)
 - SHA-1, SHA-256, 384, 512-bit digests
 - MD5 128-bit digest
 - HMAC with all algorithms
- Kasumi/F8 Hardware Accelerators (KFHA)
 - F8, F9 as required for 3GPP
 - A5/3 for GSM and EDGE, GEA-3 for GPRS
- Snow 3G Hardware Accelerators (SNOWf8 and SNOWf9)
 - Implements Snow 3.0, F8 and F9 modes
- ZUC Hardware Accelerators (ZUCE and ZUCA)
 - Implements 128-EEA3 & 128-EIA3
- CRC Unit
 - Standard and user-defined polynomials
- Random Number Generator
 - Incorporates TRNG entropy generator for seeding and deterministic engine (SHA-256)
 - Supports random IV generation

The SEC 5.0 is designed to support bulk encryption at up to 40 Gbps, large packet/record IPsec/SSL at up to 30 Gbps, and 20 Gbps for IPsec ESP at Imix packet sizes. 3G and LTE algorithms are supported at 10 Gbps or more.

The SEC dequeues data from its QMan hardware portal and, based on FQ configuration, also dequeues associated instructions and operands in the Shared Descriptor. The SEC processes the data then enqueues it to the configured output FQ. The SEC uses the Status/CMD word in the output Frame Descriptor to inform the next consumer of any errors encountered during processing (for example, received packet outside the anti-replay window.)

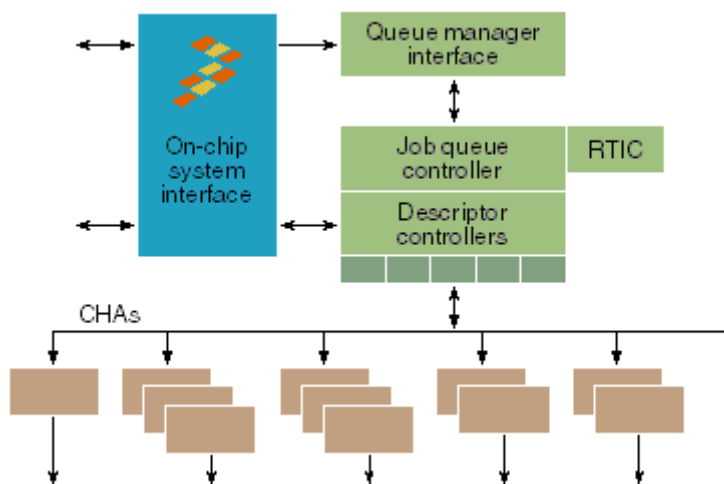


Figure 12. SEC 5.0 block diagram

The SEC 5.0 is also part of the QorIQ Platform's Trust Architecture, which gives the SoC the ability to perform secure boot, runtime code integrity protection, and session key protection. The Trust Architecture is described in [Resource partitioning and QorIQ Trust Architecture](#).

5.10.5.5 Pattern Matching Engine (PME 2.1)

The PME 2.1 is Freescale's second generation of extended NFA style pattern matching engine. Unchanged from the first generation QorIQ products, it supports ~10 Gbps data scanning.

Key benefits of a NFA pattern matching engine:

- No pattern "explosion" to support "wildcarding" or case-insensitivity
 - Comparative compilations have shown 300,000 DFA pattern equivalents can be achieved with ~8000 extended NFA patterns
- Pattern density much higher than DFA engines.
 - Patterns can be stored in on-chip tables and main DDR memory
 - Most work performed solely with on-chip tables (external memory access required only to confirm a match)
 - No need for specialty memories; for example, QDR SRAM, RLDRAM, and so on.
- Fast compilation of pattern database, with fast incremental additions
 - Pattern database can be updated without halting processing
 - Only affected pattern records are downloaded
 - DFA style engines can require minutes to hours to recompile and compress database

Freescale's basic NFA capabilities for byte pattern scanning are as follows:

- The PME's regex compiler accepts search patterns using syntax similar to that in software-based regex engines, such as Perl-Compatible Regular Expression (PCRE).
 - Supports Perl meta-characters including wildcards, repeats, ranges, anchors, and so on.
 - Byte patterns are simple matches, such as gabcd123h, existing in both the data being scanned and in the pattern specification database.
- Up to 32 KB patterns of length 1-128 bytes

Freescale's extensions to NFA style pattern matching are principally related to event pattern scanning. Event patterns are sequences of byte patterns linked by 'stateful rules.' Freescale uses event pattern scanning and stateful rule processing synonymously. Stateful rules are hardware instructions by which users define reactions to pattern match events, such as state changes, assignments, bitwise operations, addition, subtraction, and comparisons.

Some key characteristics and benefits of the Stateful Rule extensions include:

- Ability to match patterns across data "work units" or packet boundaries
 - Can be used to correlate patterns, qualify matches (for example, contextual match), or to track protocol state change
- Easily support "greedy" wildcards
 - For example, ABC.*DEF == two patterns tied together by a stateful rule
- Delays the need for software post-processing. Software is alerted after all byte patterns are detected in the proper sequence, rather than any time a byte pattern is detected.
- Implements a significant subset of the regex pattern definition syntax as well as many constructs which cannot be expressed in standard PCRE
- PME 2.1 supports up to 32K stateful rules, linking multiple byte patterns

The PME 2.1 dequeues data from its QMan hardware portal and, based on FQ configuration, scans the data against one of 256 pattern sets, 16 subsets per pattern set.

When the PME finds a byte pattern match, or a final pattern in a stateful rule, it generates a report.

5.10.5.6 Decompression and Compression Engine (DCE 1.0)

The Decompression and Compression Engine (DCE 1.0) is an accelerator compatible with Datapath Architecture providing lossless data decompression and compression for the QorIQ family of SoCs. The DCE supports the raw DEFLATE algorithm (RFC1951), GZIP format (RFC1952) and ZLIB format (RFC1950). The DCE also supports Base 64 encoding and decoding (RFC4648).

The DEFLATE algorithm is a basic building block for data compression in most modern communication systems. It is used by HTTP to compress web pages, by SSL to compress records, by gzip to compress files and email attachments, and by many other applications.

Deflate involves searching for repeated patterns previously seen in a Frame, computing the length and the distance of the pattern with respect to the current location in the Frame, and encoding the resulting information into a bitstream.

The decompression algorithm involves decoding the bitstream and replaying past data. The Decompression and Compression Engine is architected to minimize the system memory bandwidth required to do decompression and compression of Frames while providing multi-gigabits per second of performance.

Detailed features include the following:

- Deflate; as specified as in RFC1951
- GZIP; as specified in RFC1952
- Zlib; as specified in RFC1950
 - Interoperable with the zlib 1.2.5 compression library
- Compression
 - ZLIB, GZIP and DEFLATE header insertion
 - ZLIB and GZIP CRC computation and insertion
 - Zlib sync flush and partial flush for chunked compression (for example, for HTTP1.1)
 - Four modes of compression
 - No compression (just add DEFLATE header)
 - Encode only using static/dynamic Huffman codes
 - Compress and encode using static Huffman codes
 - Compress and encode using dynamic Huffman codes
 - Uses a 4KB sliding history window
 - Supports Base 64 encoding (RFC4648) after compression
 - Provides at least 2.5:1 compression ratio on the Calgary Corpus
- Decompression supports:
 - ZLIB, GZIP and DEFLATE header removal
 - ZLIB and GZIP CRC validation
 - 32KB history
 - Zlib flush for chunked decompression (for HTTP1.1 for example)

- Channels #6, #7, #8 and #9 have been configured to be scheduled by the channel scheduler for LNI#3 (for example, all the packets from these channels are directed to the physical network interface configurably coupled to LNI#3).
- Channels #6 and #7 have been configured to be "unshaped." Packets from these channels will not be subjected to shaping at the channel level and will feed the top priority level within the LNI, which is also not subjected to shaping. Their class schedulers will not distinguish between CR and ER opportunities.
- Channels #8 and #9 have been configured to be "shaped." Their class schedulers will distinguish between CR and ER opportunities. The CR/ER packets to be sent from each channel shall be subjected to a pair of CR/ER token bucket shapers specific to that channel. The aggregate of CR/ER packets from these channels are subject to a pair of CR/ER token bucket shapers specific to LNI#3.
- Channel #6 has only one class in use. That class queue behaves as if it were a channel queue and as a peer to Channel #7. Unused classes do not have to be configured as such; they are simply not used.
- Channel #7 has all 16 classes in use.
 - The group classes have been configured as two groups (A and B) of four classes.
 - The priority of the groups A and B have both been set to be immediately below independent class 5. In a case of similar configuration group A has higher priority than group B.
- Channel #8 has three independent classes and two groups of four grouped classes in use.
 - The priorities of the class groups A and B have been set to be immediately below independent class 0 and class 2 respectively.
 - Independent class 0 and class group A have been configured to request and fulfill only CR packet opportunities.
 - Independent class 1 has been configured to request and fulfill both CR and ER packet opportunities.
 - Independent class 2 and class group B have been configured to request and fulfill only ER packet opportunities.
- Channels #9 has four independent classes and one group of eight grouped classes in use.
 - The group classes have been configured as one group (A) of eight classes.
 - All independent classes and the class group (A) have been configured to request and fulfill both CR and ER packet opportunities.

Benefits of the CEETM include the following:

- Provides "virtual" ports for multiple applications or users with different QoS/CoS requirements which are sharing an egress interface
- Supports DSCP capable scheduling for the following virtual link with configurable combinations of strict priority and weighted scheduling
 - Weighted scheduling closely approximating WFQ
- Supports traffic shaping
 - dual rate shaping of the virtual links
- Supports aggregating traffic from multiple virtual links and shaping this aggregate
- Hierarchical scheduling and shaping
- Class-based scheduling and dual rate shaping
- Supports a subset of the IEEE Data Center Bridging (DCB) standards

5.10.6.3 Data Center Bridging (DCB)

Data Center Bridging (DCB) refers to a series of inter-related IEEE specifications collectively designed to enhance Ethernet LAN traffic prioritization and congestion management. Although the primary objective is the data center environment (consisting of servers and storage arrays), some aspects of DCB are applicable to more general uses of Ethernet, within and between network nodes.

The SoC DPAA is compliant with the following DCB specifications :

- IEEE Std. 802.1Qbb: Priority-based flow control (PFC)
 - PAUSE frame per Ethernet priority code point (8)
 - Prevents single traffic class from throttling entire port
- IEEE Std. 802.1Qaz: Enhanced transmission selection (ETS)
 - Up to three Traffic Class Groups (TCG), where a TCG is composed of one or more priority code points
 - Bandwidth allocation and transmit scheduling (1% granularity) by traffic class group
 - If one of the TCGs does not consume its allocated bandwidth, unused bandwidth is available to other TCGs

5.11 Resource partitioning and QorIQ Trust Architecture

Consolidation of discrete CPUs into a single, multicore chip introduces many opportunities for unintended resource contentions to arise, particularly when multiple, independent software entities reside on a single chip. A system may exhibit erratic behavior if multiple software partitions cannot effectively partition resources. Device consolidation, combined with a trend toward embedded systems becoming more open (or more likely to run third-party or open-source software on at least one of the cores), creates opportunities for malicious code to enter a system.

This chip offers a new level of hardware partitioning support, allowing system developers to ensure software running on any CPU only accesses the resources (memory, peripherals, and so on) that it is explicitly authorized to access. This section provides an overview of the features implemented in the chip that help ensure that only trusted software executes on the CPUs, and that the trusted software remains in control of the system with intended isolation.

5.11.1 Core MMU, UX/SX bits, and embedded hypervisor

The chip's first line of defense against unintended interactions amongst the multiple CPUs/OSes is each core vCPU's MMU. A vCPU's MMU is configured to determine which addresses in the global address map the CPU is able to read or write. If a particular resource (memory region, peripheral device, and so on) is dedicated to a single vCPU, that vCPU's MMU is configured to allow access to those addresses (on 4 KB granularity); other vCPU MMUs are not configured for access to those addresses, which makes them private. When two vCPUs need to share resources, their MMUs are both configured so that they have access to the shared address range.

This level of hardware support for partitioning is common today; however, it is not sufficient for many core systems running diverse software. When the functions of multiple discrete CPUs are consolidated onto a single multicore chip, achieving strong partitioning should not require the developer to map functions onto vCPUs that are the exclusive owners of specific platform resources. The alternative, a fully open system with no private resources, is also unacceptable. For this reason, the core's MMU also includes three levels of access permissions: user, supervisor (OS), and hypervisor. An embedded hypervisor (for example, KVM, XEN, QorIQ ecosystem partner hypervisor) runs unobtrusively beneath the various OSes running on the vCPUs, consuming CPU cycles only when an access attempt is made to an embedded hypervisor-managed shared resource.

The embedded hypervisor determines whether the access should be allowed and, if so, proxies the access on behalf of the original requestor. If malicious or poorly tested software on any vCPU attempts to overwrite important device configuration registers (including vCPU's MMU), the embedded hypervisor blocks the write. High and low-speed peripheral interfaces (PCI Express, UART), when not dedicated to a single vCPU/partition, are other examples of embedded hypervisor managed resources. The degree of security policy enforcement by the embedded hypervisor is implementation-dependent.

In addition to defining regions of memory as being controlled by the user, supervisor, or hypervisor, the core MMU can also configure memory regions as being non-executable. Preventing CPUs from executing instructions from regions of memory used as data buffers is a powerful defense against buffer overflows and other runtime attacks. In previous generations of Power Architecture, this feature was controlled by the NX (no execute) attribute. In new Power Architecture cores such as the e6500 core, there are separate bits controlling execution for user (UX) and supervisor (SX).

5.11.2 Peripheral access management unit (PAMU)

MMU-based access control works for software running on CPUs; however, these are not the only bus masters in the SoC. Internal components with bus mastering capability (FMan, RMan, PCI Express controller, PME, SEC, and so on) also need to be prevented from reading and writing to certain memory regions. These components do not spontaneously generate access attempts; however, if programmed to do so by buggy or malicious software, any of them could read or write sensitive data registers and crash the system. For this reason, the SoC also includes a distributed function referred to as the peripheral access management unit (PAMU).

PAMUs provide address translation and access control for all non-CPU initiators in the system. PAMU access control is based on the logical I/O device number (LIODN) advertised by a bus master for a given transaction. LIODNs can be static (for example, PCI Express controller #1 always uses LIODN 123) or they can be dynamic, based on the ID of the CPU that programmed the initiator (for example, the SEC uses LIODN 456 because it was given a descriptor by vCPU #2). In the dynamic example, the SoC architecture provides positive identification of the vCPU programming the SEC, preventing LIODN spoofing.

5.11.3 IO partitioning

The simplest IO configuration in chips running multiple independent software partitions is to dedicate specific IO controllers (PCI Express, SATA, Serial RapidIO controllers) to specific vCPUs. The core MMUs and PAMUs can enforce these access permissions to insure that only the software partition owning the IO is able to use it. The obvious problem with this approach is that there are likely to be more software partitions wanting IO access than there are IO controllers to dedicate to each.

Safe IO sharing can be accomplished through the use of a hypervisor; however, there is a performance penalty associated with virtual IO, as the hypervisor must consume CPU cycles to schedule the IO requests and get the results back to the right software partition.

The DPAA (described in [Data Path Acceleration Architecture \(DPAA\)](#)) was designed to allow multiple partitions to efficiently share accelerators and IOs, with its major capabilities centered around sharing Ethernet ports. These capabilities were enhanced in the chip with the addition of FMan storage profiles. The chip's FMans perform classification prior to buffer pool selection, allowing Ethernet frames arriving on a single port to be written to the dedicated memory of a single software partition. This capability is fully described in [Receiver functionality: parsing, classification, and distribution](#)."

The addition of the RMan extends the chip's IO virtualization by allowing many types of traffic arriving on Serial RapidIO to enter the DPAA and take advantage of its inherent virtualization and partitioning capabilities.

The PCI Express protocol lacks the PDU semantics found in Serial RapidIO, making it difficult to interwork between PCI Express controllers and the DPAA; however, PCI Express has made progress in other areas of partition. The Single Root IO Virtualization specification, which the chip supports as an endpoint, allows external hosts to view the chip as multiple two physical functions (PFs), where each PF supports up to 64 virtual functions (VFs). Having multiple VFs on a PCI Express port effectively channelizes it, so that each transaction through the port is identified as belonging to a specific PF/VF combination (with associated and potentially dedicated memory regions). Message signalled interrupts (MSIs) allow the external Host to generate interrupts associated with a specific VF.

5.11.4 Secure boot and sensitive data protection

The core MMUs and PAMU allow the SoC to enforce a consistent set of memory access permissions on a per-partition basis. When combined with an embedded hypervisor for safe sharing of resources, the SoC becomes highly resilient to poorly tested or malicious code. For system developers building high reliability/high security platforms, rigorous testing of code of known origin is the norm.

For this reason, the SoC offers a secure boot option, in which the system developer digitally signs the code to be executed by the CPUs, and the SoC insures that only an unaltered version of that code runs on the platform. The SoC offers both boot time and run time code authenticity checking, with configurable consequences when the authenticity check fails. The SoC also supports protected internal and external storage of developer-provisioned sensitive instructions and data. For example, a system developer may provision each system with a number of RSA private keys to be used in mutual authentication and key exchange. These values would initially be stored as encrypted blobs in external non-volatile memory; but, following secure boot, these values can be decrypted into on-chip protected memory (portion of platform cache dedicated as SRAM). Session keys, which may number in the thousands to tens of thousands, are not good candidates for on-chip storage, so the SoC offers session key encryption. Session keys are stored in main memory, and are decrypted (transparently to software and without impacting SEC throughput) as they are brought into the SEC 5.0 for decryption of session traffic.

5.12.2 Non-transparent power management

Many load-based power savings are use-case specific static configurations (thereby software transparent), and were described in the previous section. This section focuses on SoC power management mechanisms, which software can dynamically leverage to reduce power when the system is lightly loaded. The most important of these mechanisms involves the cores.

A full description of core low-power states with proper names is provided in the SoC reference manual. At a high level, the most important of these states can be viewed as "PH10" and "PH20," described as follows. Note that these are relative terms, which do not perfectly correlate to previous uses of these terms in Power Architecture and other ISAs:

- In PH10 state CPU stops instruction fetches but still performs L1 snoops. The CPU retains all state, and instruction fetching can be restarted instantly.
- In PH20 state CPU stops instruction fetches and L1 snooping, and turns off all clocks. Supply voltage is reduced, using a technique Freescale calls State Retention Power Gating (SRPG). In the "napping" state, a CPU uses ~75% less power than a fully operational CPU, but can still return to full operation quickly (~100 platform clocks).

The core offers two ways to enter these (and other) low power states: registers and instructions.

As the name implies, register-based power management means that software writes to registers to select the CPU and its low power state. Any CPU with write access to power management registers can put itself, or another CPU, into a low power state; however, a CPU put into a low power state by way of register write cannot wake itself up.

Instruction-based power management means that software executes special WAIT instruction to enter a low power state. CPUs exit the low power state in response to external triggers, interrupts, doorbells, stashes into L1-D cache, or clear reservation on snoop. Each vCPU can independently execute WAIT instructions; however, the physical CPU enters PH20 state after the second vCPU executes its wait. The instruction-based "enters PH20 state" state is particularly well-suited for use in conjunction with Freescale's patented Cascade Power Management, which is described in the next section.

While significant power savings can be achieved through individual CPU low power states, the SoC also supports a register-based cluster level low power state. After software puts all CPUs in a cluster in a PH10 state, it can additionally flush the L2 cache and have the entire cluster enter PH20 state. Because the L2 arrays have relatively low static power dissipation, this state provides incremental additional savings over having four napping CPUs with the L2 on.

5.12.3 Cascade power management

Cascade power management refers to the concept of allowing SoC load, as defined by the depth of queues managed by the Queue Manager, to determine how many vCPUs need to be awake to handle the load. Recall from [Queue Manager](#) that the QMan supports both dedicated and pool channels. Pool channels are channels of frame queues consumed by parallel workers (vCPUs), where any worker can process any packet dequeued from the channel.

Cascade Power Management exploits the QMan's awareness of vCPU membership in a pool channel and overall pool channel queue depth. The QMan uses this information to tell vCPUs in a pool channel (starting with the highest numbered vCPU) that they can execute instructions to "take a nap." When pool channel queue depth exceeds configurable thresholds, the QMan wakes up the lowest numbered vCPU.

The SoC's dynamic power management capabilities, whether using the Cascade scheme or a master control CPU and load to power matching software, enable up to a 75% reduction to each core in power consumption versus data sheet max power.

5.13 Debug support

The reduced number of external buses enabled by the move to multicore chips greatly simplifies board level lay-out and eliminates many concerns over signal integrity. Even though the board designer may embrace multicore CPUs, software engineers have real concerns over the potential to lose debug visibility. Despite the problems external buses can cause for the hardware engineer, they provide software developers with the ultimate confirmation that the proper instructions and data are passing between processing elements.

Processing on a multicore chip with shared caches and peripherals also leads to greater concurrency and an increased potential for unintended interactions between device components. To ensure that software developers have the same or better visibility into the device as they would with multiple discrete communications processors, Freescale developed an Advanced Multicore Debug Architecture.

The debugging and performance monitoring capability enabled by the device hardware coexists within a debug ecosystem that offers a rich variety of tools at different levels of the hardware/software stack. Software development and debug tools from Freescale (CodeWarrior), as well as third-party vendors, provide a rich set of options for configuring, controlling, and analyzing debug and performance related events.

6 Conclusion

Featuring 24 virtual cores, and based on the dual-threaded e6500 Power Architecture core, the T4240 processor, along with its 16 (T4160) and 8 (T4080) virtual-core variants, offers frequencies up to 1.8 GHz, large caches, hardware acceleration, and advanced system peripherals. All three devices target applications that benefit from consolidation of control and data plane processing in a single chip. In addition, each e6500 core implements the Freescale AltiVec technology SIMD engine, dramatically boosting the performance of math-intensive algorithms without using additional DSP components on the board. A wide variety of applications can benefit from the processing, I/O integration, and power management offered for the T4 series processors. Similar to other QorIQ devices, the T4 family processors' high level of integration offers significant space, weight, and power benefits compared to multiple discrete devices. Freescale also offers fully featured development support, which includes the QorIQ T4240 QDS Development System, QorIQ T4240 Reference Design Board, Linux SDK for QorIQ Processors, as well as popular operating systems and development tools from a variety of vendors. See the Freescale website for the latest information on tools and SW availability.

For more information about the QorIQ T4 family, contact your Freescale sales representative.

Appendix A T4160

A.1 Introduction

The T4160 is a lower power version of the T4240. The T4160 combines eight dual threaded Power Architecture e6500 cores and two memory complexes (CoreNet platform cache and DDR3 memory controller) with the same high-performance datapath acceleration, networking, and peripheral bus interfaces.

This figure shows the major functional units within the chip.

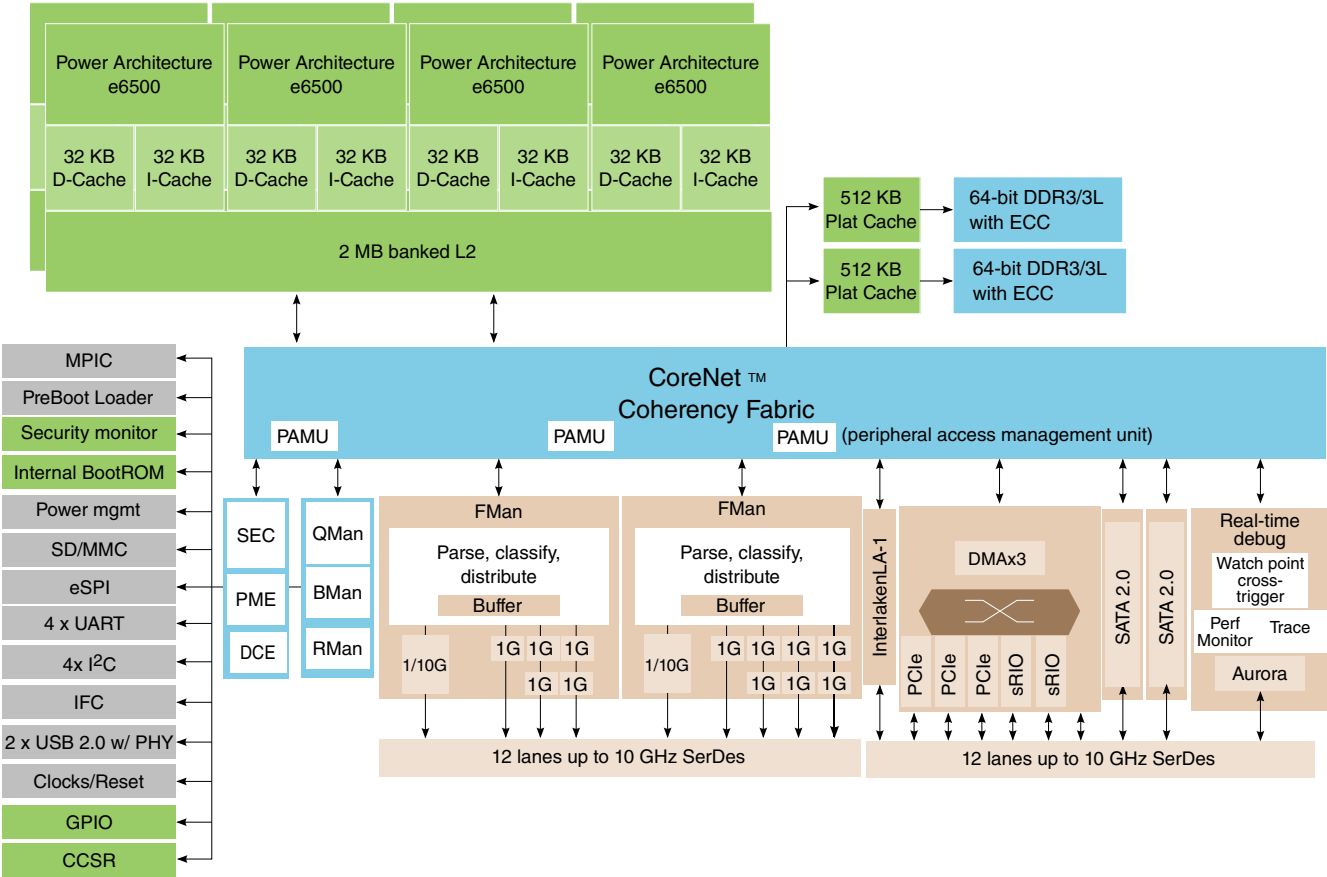


Figure A-1. T4160 block diagram

A.2 Overview of differences between T4240 and T4160

Table A-1. Differences between T4240 and T4160

Feature	T4240	T4160
Cores		
Number of physical cores	12	8
Number of threads	24	16
Number of clusters	3	2
Memory subsystem		
Total CPC memory	3 x 512 KB	2 x 512 KB
Number of DDR controllers	3	2
Peripherals		
Number of Frame Managers	2	2
Total number of Anyspeed MACs	8 per Frame Manager	6 (FMan1) and 8 (FMan2)

Table continues on the next page...

Table A-1. Differences between T4240 and T4160 (continued)

Feature	T4240	T4160
Max number of Anyspeed MACs configured for 10 GE operation	2 per Frame Manager	1 per Frame Manager
SerDes and pinout		
Total number of SerDes lanes	4 x 8	2 x 4 and 2 x 8
High-speed IO		
PCIe	4	3 (PCIe 3 is disabled)

Appendix B T4080

B.1 Introduction

The T4080 is a low power version of the T4160. The T4080 has four dual threaded Power Architecture e6500 cores with the same two memory complexes (CoreNet platform cache and DDR3 memory controller) with the same high-performance datapath acceleration, networking, and peripheral bus interfaces.

This figure shows the major functional units within the chip.



How to Reach Us:

Home Page:

freescale.com

Web Support:

freescale.com/support

Information in this document is provided solely to enable system and software implementers to use Freescale products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document.

Freescale reserves the right to make changes without further notice to any products herein.

Freescale makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages.

“Typical” parameters that may be provided in Freescale data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. Freescale does not convey any license under its patent rights nor the rights of others. Freescale sells products pursuant to standard terms and conditions of sale, which can be found at the following address: freescale.com/SalesTermsandConditions .

Freescale, the Freescale logo, AltiVec, CodeWarrior, Energy Efficient Solutions logo, and QorIQ are trademarks of Freescale Semiconductor, Inc., Reg. U.S. Pat. & Tm. Off. CoreNet is a trademark of Freescale Semiconductor, Inc. All other product or service names are the property of their respective owners. The Power Architecture and Power.org word marks and the Power and Power.org logos and related marks are trademarks and service marks licensed by Power.org.

© 2013–2014 Freescale Semiconductor, Inc.