Freescale Semiconductor
Product Brief

Document Number: P4080PB
Rev. 1, 09/2008

QorIQ™ P4080 Communications Processor Product Brief

The QorIQ™ P4080 Communications Processor combines eight Power Architecture™ processor cores with high-performance datapath acceleration logic and network and peripheral bus interfaces required for networking, telecom/datacom, wireless infrastructure, and mil/aerospace applications.

The P4080 can be used for combined control, datapath, and application layer processing in routers, switches, base station controllers, and general-purpose embedded computing systems. Its high level of integration offers significant performance benefits compared to multiple discrete devices, while also greatly simplifying board design.

This product brief provides an overview of the P4080's features, with expanded explanations of multicore and data path areas of innovation. Power management and the developer's environment created by Freescale's enablement ecosystem also receive an expanded explanation.

Contents

1 P4080 Features

The P4080 SoC includes the following function and features:

- Eight e500mc cores built on Power Architecture technology, each with a private 128-kbyte L2 Cache
  - 3 levels of instructions: User, Supervisor, and Hypervisor
  - Independent boot and reset
  - Secure boot capability
- 2-Mbyte shared L3 CoreNet Platform Cache (CPC)
- Hierarchical interconnect fabric
  - CoreNet fabric supporting coherent and non-coherent transactions with prioritization and bandwidth allocation amongst CoreNet end-points
  - .8Tbps coherent read bandwidth
  - Queue Manager fabric supporting packet-level queue management and quality of service scheduling
- Two 64-bit DDR2/DDR3 SDRAM memory controllers with ECC and interleaving support
- Datapath Acceleration Architecture incorporating acceleration for the following functions:
  - Packet parsing, classification, and distribution
  - Queue Management for scheduling, packet sequencing, and congestion management
  - Hardware Buffer Management for buffer allocation and de-allocation
  - Encryption/decryption (SEC 4.0)
  - RegEx Pattern Matching (PME 2.0)
- Ethernet Interfaces
  - Two 10 Gbps Ethernet (XAUI) controllers
  - Eight 1 Gbps Ethernet controllers
- High Speed Peripheral Interfaces
  - Three PCI Express 2.0 controllers/ports running at up to 5 GHz
  - Two serial RapidIO® 1.2 controllers/ports running at up to 3.125 GHz
- Additional Peripheral Interfaces
  - Two USB Controllers with ULPI interface to external PHY
  - SD/MMC
  - SPI Controller
  - Four I^{2}C controllers
  - Two Dual UARTs
  - Enhanced Local Bus Controller (eLBC)
- Multicore Programmable Interrupt Controller (PIC)
- Two 4-channel DMA engines

1.1 Block Diagram

Figure 1 shows the major functional units within the P4080.

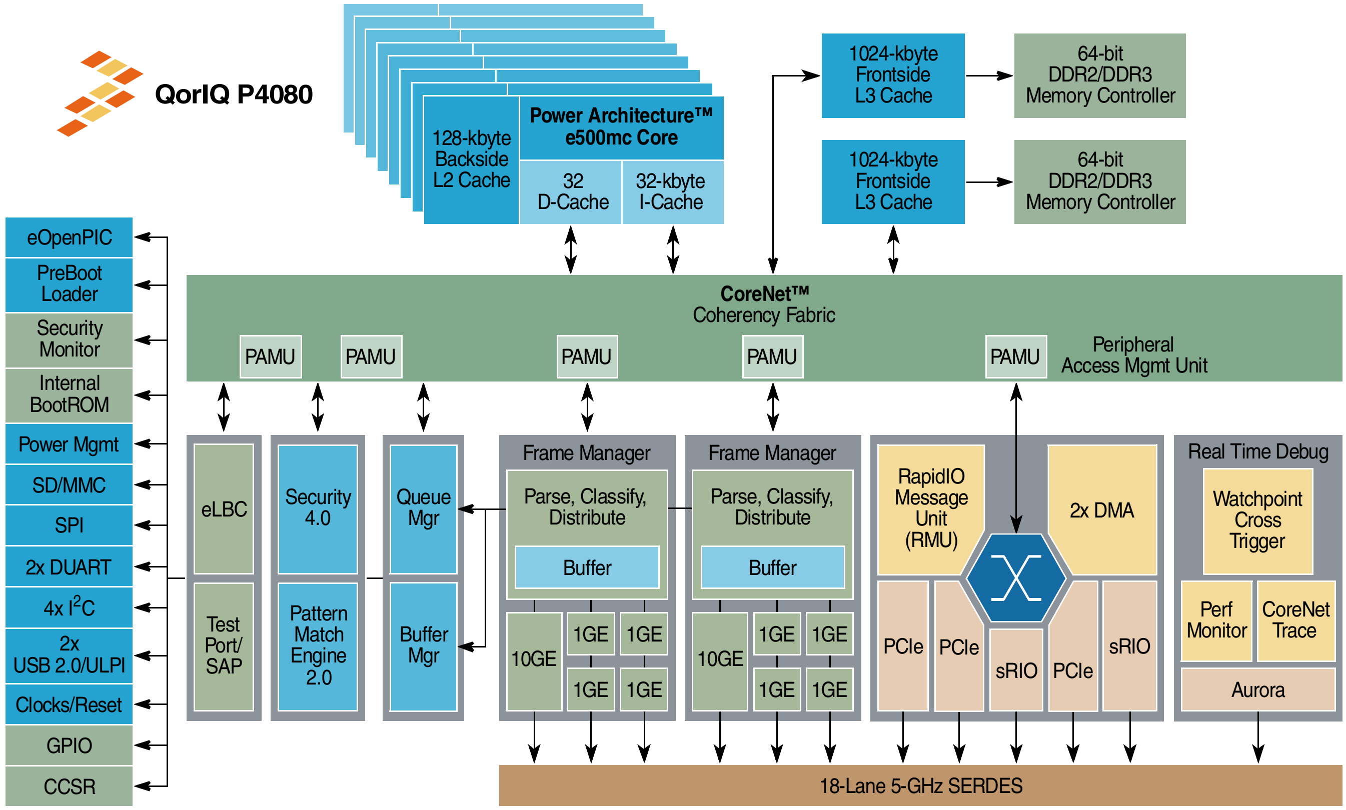


Figure 1. P4080 Preliminary Block Diagram

1.2 Operational Parameters

- Top speed bin e500mc core frequency of 1.5 GHz at 1.0 V
- Core supply voltage options:
  - — All cores 1.0-V
  - — Two cores up to 1.0-V to achieve higher frequencies
  - — Up to 6 cores down to .9-V for lower power consumption at lower frequencies
- Maximum memory data rates of 1066 MHz (DDR2), 1600 MHz (DDR3)
  - — DDR: 1.8-V for DDR2, 1.5-V for DDR3 (conforms to JEDEC standard)
- Local bus: 3.3-, 2.5-, or 1.8-V
- Operating junction temperature range is 0–105C.
- Max Power dissipation target <30 W: 8 x 1.5 GHz, 1.0-V operation
- Package: 1295-pin FC-PBGA (flip-chip plastic ball grid array)

2 Application Examples

The P4080 is a very flexible device that can be configured to meet many system application needs.

The P4080's e500mc cores can be combined as a fully-symmetric, multi-processing, system-on-a-chip, or they can be operated with varying degrees of independence to perform asymmetric multi-processing. Full processor independence, including the ability to independently boot and reset each e500mc core, is a defining characteristic of the P4080. The ability of the cores to run different operating systems, or run OS-less, provides the user with significant flexibility in partitioning between control, datapath, and applications processing. It also simplifies consolidation of functions previously spread across multiple discrete processors onto a single device.

While the eight Power Architecture cores offer a major leap in available processor performance, in many throughput intensive, packet processing networking applications, raw processing power isn't enough to achieve multi-Gbps data rates. To address this, the P4080 introduces Freescale's Datapath Acceleration Architecture (DPAA), which significantly reduces data plane instructions per packet, enabling more CPU cycles to work on value-added services, rather than repetitive low-level tasks. Combined with specialized accelerators for cryptography and pattern matching, the P4080 allows the user's software to perform complex packet processing at high data rates.

2.1 Multicore Processing Scenarios

There are several ways to map operating systems to the eight P4080 cores:

- Eight core asymmetric multi-processing
  - Eight copies of the same uni-processor OS
  - Up to eight different uni-processor OSes
- Eight-core symmetric multi-processing
- Mixed symmetric and asymmetric multi-processing
  - N cores running in SMP mode, while the remainder of the cores operate asymmetrically with up to 8-N different OSes.

It is also possible for one or more cores to run OS-less, using a simple scheduler. This is a likely scenario when cores are performing datapath operations with bounded real-time requirements. This use case is greatly enhanced by the provisioning of a 128-KByte private back-side L2 cache for each e500mc core. These caches can operate as a traditional unified cache, or be set to operate as Instruction Only, Data Only, or even locked and used as memory-mapped SRAM.

CPU cores operating asymmetrically can be run at asynchronous clock rates. Each processor can source its input clock from one of the multiple PLLs inside the P4080. This allows each core to operate at the minimum frequency required to perform its assigned function, saving power. The cores are also capable of running at half and quarter ratios of their input PLL frequency, and can switch between PLLs and ratios nearly instantaneously. This allows lightly utilized CPUs to be slowed (under software control) for power savings, rather than performing more complex task migration operations.

Figure 2 shows several CPU usage scenarios, along with potential interaction with the datapath acceleration architecture.

In A, all CPUs are running a single operating system, with any specialization of CPU function occurring through OS techniques such as Task Affinity. The I/Os and acceleration hardware are under the control of the SMP OS. Typically all CPUs operate at the same frequency.

In B, some number of the cores are operated as an SMP cluster, most likely running high complexity control plane operations. The control plane configures and manages the remaining processors, which are running individual copies of an RTOS or scheduler to perform dataplane operations. In this scenario, the SMP CPUs typically operate at the same frequency, the remaining CPUs can run at a different frequency from the SMP CPUs, and even from each other.

In C, a single CPU is used as the control processor, configuring and managing the other seven processors, which are running individual copies of an RTOS or scheduler, as in B. As in B, CPU operating frequencies are an independent parameter.

In D, all CPUs are used for datapath operations, here shown as two sets of pipelined functions, each interacting independently with the I/Os and accelerators. Operating frequencies for each CPU in the pipeline can be set independently, and the provision of a 128-kbyte back-side L2 provides significant flexibility in partitioning and rebalancing the pipeline as processing requirements change.

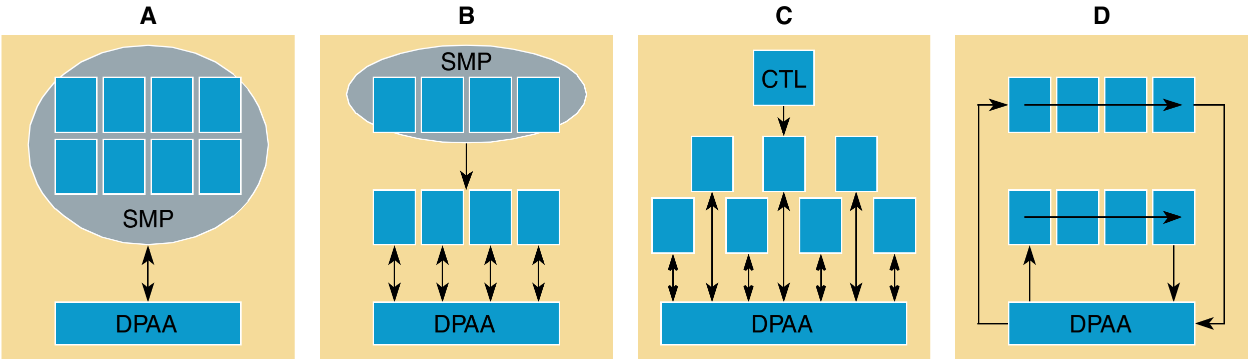


Figure 2. MultiCore Processing Scenarios

A fifth scenario, shown in Figure 3, involves the use of one of the CPUs as an I/O Processor. The datapath architecture (described in more detail later in this document) can greatly simplify and accelerate processing for packets entering the system by means of the Ethernet interfaces. For systems requiring external ASICs or legacy network interface cards in the high performance datapath, system developers can allocate a CPU to help interwork between the native data buffers used by PCIe- or serial RapidIO-based network interfaces and the data buffers used by the datapath acceleration hardware.

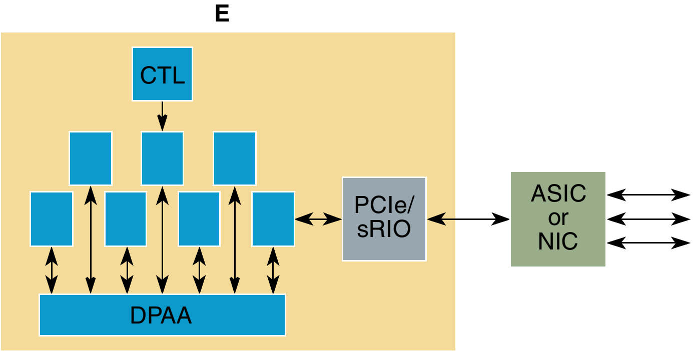


Figure 3. IO Processor Managing PCIe/Serial RapidIO-Based Network Interfaces

2.2 P4080 Applications

The P4080 is well suited for applications that are highly compute-intensive, I/O-intensive, or both. Some examples of each are shown in the following sections.

2.2.1 Virtual Private Network (VPN)/ IP Services Router

Figure 4 shows a virtual private network (VPN)/IP Services router enabled through PCIe and Ethernet. The QorIQ DPAA accelerates packet classification, filtering, and packet queuing, while the crypto accelerator (SEC 4.0) and pattern matching engine (PME 2.0) perform high throughput encryption/decryption and regex payload scanning security under control of stacks running on the CPUs. Session establishment, policy enforcement, and, potentially, application processing are executed by the control processor(s). Security appliances would use the P4080 in a similar manner, however without direct connections to the WAN and DMZ server.

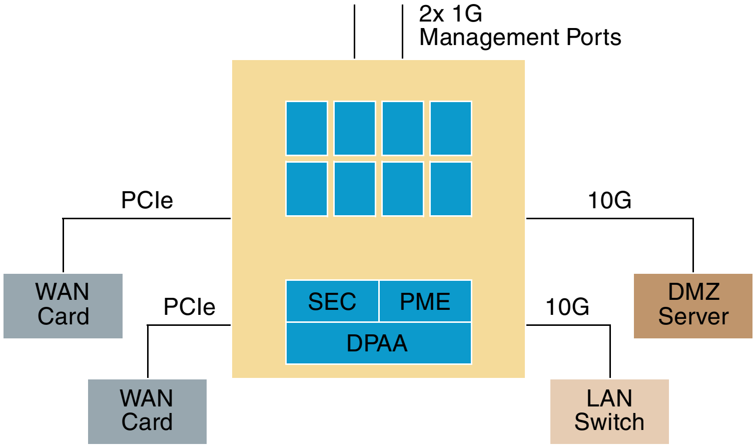


Figure 4. VPN/IP Services Router

2.2.2 Security Services Blade for Switch or Server

Figure 5 shows a P4080 operating as a packet processing engine in a security services blade. Due to the presence of an external control processor, all of the P4080's processors are dedicated to complex content-oriented packet processing, assisting the DPAA. Functions performed might include security protocol termination (MACSec, IPSec, SSL/TLS) and content inspection.

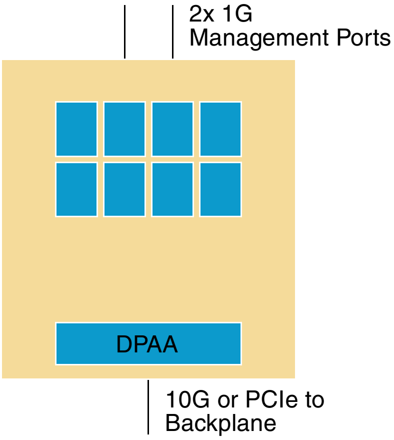


Figure 5. Security Services Blade for Switch, Router, or Server

2.2.3 Wireless Infrastructure/Radio Node Controller

Some of the more demanding packet-processing applications are found in the realm of wireless infrastructure. Systems have to interwork between wireless link layer protocols and IP networking protocols. Wireless link layer termination typically involves decryption, using algorithms such as Kasumi which are very 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 P4080's network and peripheral interfaces provide it with the flexibility to connect to DSPs, and wireless link layer framing ASICs/FPGAs. Multiple processors may be dedicated to data path processing in each direction, while the data path acceleration architecture offers encryption acceleration for both wireless and IP networking protocols, plus packet filtering capability on the IP networking side. A representative wireless infrastructure application is shown in Figure 6.

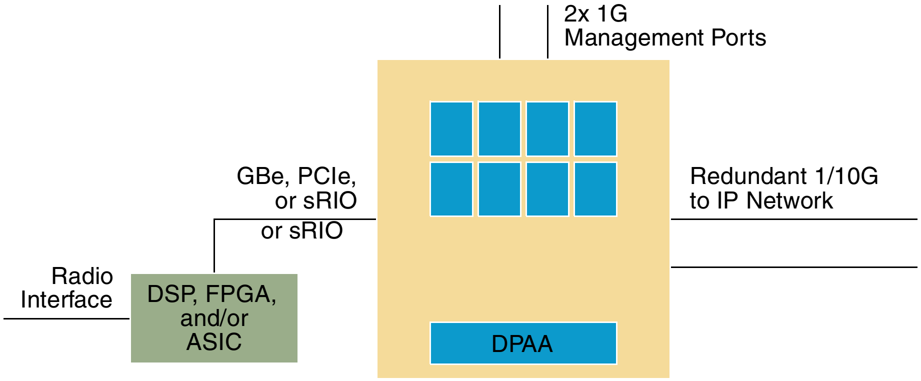


Figure 6. Wireless Infrastructure/Radio Node Controller

2.2.4 High-Performance Compute Blade

Figure 7 shows likely scenarios for eight core symmetric multi-processing on the P4080. The eight processors may provide all the computing power needed for a stand-alone application, or multiple QorIQ P4080s may be placed on a single board to create an extremely high-density computing platform suitable for military/aerospace, high-performance test and measurement, and other compute-intensive applications. In a grid computing application, data shared among multiple computing platforms may use network transport protocols as the interconnect, and depending on the grid elements' physical proximity and security requirements, shared application data may be encrypted or at least cryptographically authenticated. In such a case, the datapath acceleration architecture assists by performing TCP or even SSL termination.

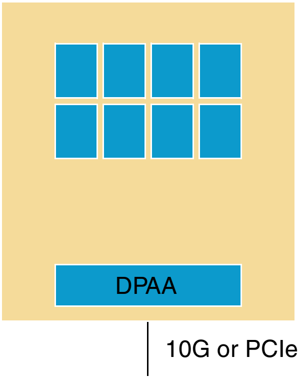


Figure 7. High-Performance Compute Blade

Figure 8 shows the potential scaling for very high performance compute blade for the P4080.

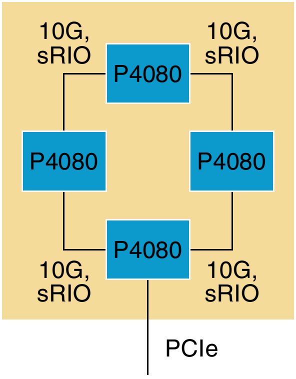


Figure 8. Potential Scaling for Very High Performance Compute Blade

2.3 Subsystem Features

This section highlights the subsystems or complexes of the P4080 SoC and describes their features.

2.3.1 e500 Core and Cache Memory Complex

The P4080 offers eight high-performance 32-bit Power Architecture Book E-compliant e500mc cores. Like previous e500 cores, each e500mc is a superscalar dual issue processor, supporting out-of-order execution and in-order completion, which allows the Power Architecture e500mc to perform more instructions per clock than other RISC and CISC architectures.

Some of the important features of the e500mc are listed below:

- Up to 1.5 GHz at 1.0 V
- 36 bit physical addressing
- 64 TLB SuperPages
- 512-entry 4-kbyte pages
- 3 Integer Units: 2 simple, 1 complex (integer multiply and divide)
- 64-byte cache line size
- L1 caches, running at same frequency of CPU
  - 32-kbyte Instruction, 8-way
  - 32-kbyte Data, 8-way
  - Both with data and tag parity protection
- Supports Datapath Acceleration Architecture (DPAA) data and context “stashing” into L1 cache
- User, Supervisor, and Hypervisor instruction level privileges
- New processor facilities
  - Hypervisor APU
  - Classic Double Precision Floating Point Unit

        - Uses 32 64-bit floating-point registers (FPRs) for scalar single- and double-precision floating-point arithmetic.
        - Replaces the embedded floating-point facility (SPE) implemented on the e500v1 and e500v2.
    - "Decorated Storage" APU for improved statistics support
        - Provides additional atomic operations, including a "fire-and-forget" atomic update of up to two 64-bit quantities by a single access.
    - Expanded Interrupt Model
        - Improved Programmable Interrupt Controller (PIC) automatically ACKs interrupts
        - Implements message send and receive functions for interprocessor communication, including receive filtering
    - External PID load and store facility
        - Provides system software with an efficient means to move data and perform cache operations between two disjoint address spaces
        - Eliminates the need to copy data from a source context into a kernel context, change to destination address space, then copy the data to the destination address space or alternatively to map the user space into the kernel address space

Each e500mc core features a 128-kbyte private L2 cache running at the same frequency of CPU. The L2 caches support the following:

- Write Back, pseudo LRU replacement algorithm
- Tag parity and ECC data protection
- Eight-way, with arbitrary partitioning between instruction and data. For example, 3 ways instruction, 5 ways data, etc.
- Supports direct stashing of datapath architecture data into L2

The P4080 also contains 2 Mbytes of shared L3 CoreNet Platform Cache (CPC), with the following features:

- Configurable as Write Back or Write Through
- Pseudo LRU replacement algorithm
- ECC protection
- 64 B coherency granule
- Two cache line reads (1024 bytes) per cycle at 800 MHz, 0.8 terabits/sec read bandwidth
- 32-way cache array configurable to any of several modes on a per-way basis.
    - Unified cache, I-only, D-only
    - I/O stash (configurable portion of each packet copied to CPC on write to main memory)
        - stashing of all transactions and sizes supported
        - explicit (CoreNet signalled) and implicit (address range based) stash allocation
    - Addressable SRAM (32-kbyte granularity)

2.3.2 CoreNet Fabric and Address Map

The CoreNet fabric is Freescale's next generation Front-side Interconnect Standard for multicore products. CoreNet is a highly concurrent, fully cache coherent, multi-ported fabric. CoreNet's point-to-point connectivity with flexible protocol architecture allows for pipelined interconnection between CPUs, platform caches, memory controllers, and I/O and accelerators at up to 800 MHz.

CoreNet has been designed to overcome bottlenecks associated with shared bus architectures, particularly address issue and data bandwidth limitations. The P4080's multiple, parallel address paths allow for high address bandwidth, which is a key performance indicator for large coherent multicore processors.

CoreNet also eliminates address retries, triggered by CPUs being unable to snoop within the narrow snooping window of a shared bus. This results in the P4080 having lower average memory latency.

The flexible P4080 36-bit physical address map consists of local space and external address space. For the local address map, thirty-two local access windows define mapping within the local 36-bit (64-Gbyte) address space. Inbound and outbound translation windows can map the P4080 into a larger system address space such as the RapidIO or PCIe 64-bit address environment. This functionality is included in the address translation and mapping units (ATMUs).

2.3.3 Memory Complex

The P4080 memory complex consists of the 2 DDR controllers for main memory, and the memory controllers associated with the Enhanced Local Bus Controller.

2.3.3.1 DDR Memory Controllers

The two DDR memory controllers support DDR2 and DDR3 SDRAM. The memory interface controls main memory accesses and together the two controllers support a maximum of 64 Gbyte of main memory. The P4080 also supports chip-select interleaving within a controller as well as interleaving across controllers on bank, page, or cache line boundaries.

The P4080 can be configured to retain the currently active SDRAM page for pipelined burst accesses. Page mode support of up to 64 simultaneously open pages can dramatically reduce access latencies for page hits. Depending on the memory system design and timing parameters, page mode can save up to 10 memory clock cycles for subsequent burst accesses that hit in an active page.

Using ECC, the P4080 detects and corrects all single-bit errors and detects all double-bit errors and all errors within a nibble.

Upon detection of a loss of power signal from external logic, the DDR controllers can put compliant DDR SDRAM DIMMs into self-refresh mode, allowing systems to implement battery-backed main memory protection. In addition, the DDR controllers offer an initialization bypass feature for use by system designers to prevent re-initialization of main memory during system power-on after an abnormal shutdown. The DDR controllers also support active zeroization of system memory upon detection of a user-defined security violation.

2.3.3.2 PreBoot Loader and Nonvolatile Memory Interfaces

The PreBoot Loader is a new logic block which operates similarly to an I2C boot sequencer, but on behalf of a larger number of interfaces. The PBL exists to simplify boot operations, replacing pin strapping resistors with configuration data loaded from nonvolatile memory. The PBL uses the configuration data to initialize other system logic and to copy data from low speed memory interfaces (I2C, eLBC, SPI, and SD/MMC) into fully initialized DDR or the 2-Mbyte front-side cache. The PBL then releases CPU 0 from reset, allowing the boot processes to begin from fast system memory.

The nonvolatile memory interfaces accessible by the PBL are described in the subsequent sections. Note that these interfaces may be accessed by software running on the CPUs following boot; they are not dedicated to the PBL. Also note that the Enhanced Local Bus Controller can be used for both volatile (SRAM) and nonvolatile memory, as well as a control and low performance data port for external memory-mapped devices.

2.3.3.2.1 Enhanced Local Bus Controllers

The enhanced local bus controller (eLBC) port connects to a variety of external memories, DSPs, and ASICs. Three separate state machines share the same external pins and can be programmed separately to access different types of devices. The general-purpose chip select machine (GPCM) controls accesses to asynchronous devices using a simple handshake protocol. The user-programmable machine (UPM) can be programmed to interface to synchronous devices or custom ASIC interfaces. The NAND flash control machine (FCM) further extends interface options. Each chip select can be configured so that the associated chip interface is controlled by the GPCM, UPM, or FCM controller. All controllers can be enabled simultaneously. The eLBC internally arbitrates among the controllers, allowing each to read or write a limited amount of data before allowing another controller to use the bus.

Features of the local bus controller are as follows:

- Multiplexed 26-bit address and 16-bit data bus operating at up to 100 MHz
- Eight chip selects for eight external slaves
- Up to eight-beat burst transfers
- 8- and 16-bit port sizes controlled by an internal memory controller
- Three protocol engines on a per-chip-select basis
- Parity support
- Default boot ROM chip select with configurable bus width (8- or 16-bit)
- Support for parallel NAND and NOR flash

2.3.3.2.2 Serial Memory Controllers

In addition to the parallel NAND and NOR flash supported by means of the eLBC, the P4080 supports serial flash using SPI and SD/MMC card interfaces. The SD/MMC controller includes a DMA engine, allowing it to move data from serial flash to external or internal memory following straightforward initiation by software.

2.3.4 Universal Serial Bus (USB) 2.0

The two USB 2.0 controllers provide point-to-point connectivity complying with the USB specification, Rev. 2.0. Each of the USB controllers can be configured to operate as a stand-alone host, and one of the controllers (USB #2) can be configured as a stand-alone device, or with both host and device functions operating simultaneously.

USB 2.0 controller highlights:

- Complies with USB specification, Rev. 2.0
- Supports high-speed (480 Mbps), full-speed (12 Mbps), and low-speed (1.5 Mbps) operations
- Supports external PHY with UTMI+ low-pin interface (ULPI)
    - ULPI interfaces are muxed with RGMII. Requires system configuration choice between use of USB and use of 10/100/1000 Ethernet MAC (dTSEC) by means of RGMII.
- Both controllers support operation as a stand-alone USB host controller
    - Supports USB root hub with one downstream-facing port
    - Enhanced host controller interface (EHCI)-compatible
- One controller supports operation as a stand-alone USB device
    - Supports one upstream-facing port
    - Supports six programmable USB endpoints

The host and device functions are both configured to support all four USB transfer types: Bulk, Control, Interrupt, and Isochronous.

2.3.5 High-Speed Peripheral Interface Complex

All high-speed peripheral interfaces connect to a common crossbar switch referred to as Ocean. Two high-speed I/O interface standards are supported: PCI Express (PCIe), and serial RapidIO (sRIO). The P4080 integrates three PCIe controllers and two serial RapidIO controllers plus a RapidIO Messaging Unit (RMU). See Appendix A for more information on SERDES usage for Ethernet and high-speed peripheral interfaces.

The features of each controller are described in the subsequent sections.

2.3.5.1 PCI Express Controllers

Each of the three PCIe interfaces is compliant with the *PCI Express Base Specification Revision 2.0*. Power-on reset configuration options allow root complex or endpoint functionality. The physical layer operates at 2.5 or 5 Gbaud data rate per lane. Receive and transmit ports operate independently, with an aggregate theoretical bandwidth of 32 Gbps. Other features of the PCIe interfaces include the following:

- x8, x4, x2, and x1 link widths supported
- Both 32- and 64-bit addressing and 256-byte maximum payload size
- Full 64-bit decode with 36-bit-wide windows
- Inbound INTx transactions
- Message Signaled Interrupt (MSI) transactions

2.3.5.2 Serial RapidIO

The serial RapidIO interface is based on the *RapidIO Interconnect Specification, Revision 1.2*. 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 x4 serial RapidIO controllers, the aggregate theoretical bandwidth is 16 Gbps. Key features of the serial RapidIO interface unit include:

- Support for *RapidIO Interconnect Specification, Revision 1.2* (all transaction flows and priorities)
- Both 1x, 4x LP-serial link interfaces, with transmission rates of 2.5 or 3.125 Gbaud (data rates of 2.0 or 2.5 Gbps) per lane.
- Auto-detection of 1x, 4x mode operation during port initialization
- 34-bit addressing and up to 256-byte data payload
- Receiver-controlled flow control
- RapidIO error injection
- Internal LP-serial and application interface-level loopback modes

The RapidIO Messaging Unit (RMU) manages two inbox/outbox mailboxes (queues) for data and one doorbell message structure. The outbox operates in both chaining and direct modes, and messages can hold up to 16 packets of 256 bytes, or a total of 4 kbytes. The RMU can also multi-cast a single-segment 156-byte message to up to 32 different destination DevIDs. The RMU supports type 11 message formats.

2.3.6 Datapath Acceleration Architecture (DPAA)

The P4080 includes the first implementation of the PowerQUICC Datapath Acceleration Architecture (DPAA). This architecture provides the infrastructure to support simplified sharing of networking interfaces and accelerators by multiple CPU cores. These resources are abstracted into enqueue/dequeue operations by means of a common DPAA Queue Manager Driver. Beyond enabling multicore sharing of resources, the DPAA significantly reduces software overheads associated with high-touch packet-forwarding operations. Examples of the types of packet-processing services this architecture is optimized to support include traditional routing and bridging, firewall, VPN termination for both IPsec and SSL VPNs, Intrusion Detection/Prevention (IDS/IPS), and network anti-virus (AV).

The DPAA generally leaves software in control of protocol processing, while reducing CPU overheads through off-load functions which fall into two broad categories, as follows:

1. Packet Distribution and Queue/Congestion Management. Off-load functions in this category include:
   - Data Buffer Management: Supports allocation and deallocation of buffers belonging to pools originally created by software with configurable depletion thresholds. Implemented in a block called the Buffer Manager (BMan).
   - Queue Management: Supports queuing and quality-of-service scheduling of frames to CPUs, network interfaces and DPAA logic blocks, maintains packet ordering within flows. Implemented in a block called the Queue Manager (QMan). The Queue Manager, besides

providing flow-level queuing, is also responsible for congestion management functions such as RED/WRED, Congestion Notifications and Tail Discards.

- Packet Distribution: Supports in-line packet parsing and general classification to enable policing and QoS-based packet distribution to the CPUs for further processing of the packets. This function is implemented in the block called Frame Manager (FMan).
- Policing: Supports in-line rate-limiting by means of Two-Rate-Three Color Marking (RFC2698). Up to 256 policing profiles are supported. This function is also implemented in the Frame Manager.

2. Content Processing Acceleration. Properly implemented acceleration logic can provide significant performance advantages over the most optimized software, with acceleration factors on the order of 10-100x. Accelerators in this category typically touch most of the bytes of a packet, not just headers. To avoid consuming CPU cycles to move data to the accelerators, these engines include well-pipelined DMAs. Specific Content Processing Accelerators in the P4080 include:
   - SEC 4.0—Crypto-Acceleration for protocols such as IPsec, SSL, and 802.16
   - PME 2.0—Regex style pattern matching for unanchored searches, including cross-packet stateful patterns

Prior versions of the SEC and PME are integrated into multiple members of the PowerQUICC family. Both of these engines have been enhanced to work within the DPAA, and also upgraded in both features and performance.

2.3.6.1 Datapath Acceleration Architecture Programming Model

The DPAA assumes the existence of network flows, each of which is a series of packets that have the same packet processing and ordering requirements. Software has full flexibility in how each flow is defined, from a broad grouping of packets down to an individual session that is more synonymous with the standard networking definition of a flow. In general, packets arriving from the network are categorized into broader flows to be distributed to the appropriate cores as defined by packet parsing and classification rules (for example, control plane traffic), while packets being sent to the hardware accelerators are categorized into more granular flows (for example, a specific IPSec tunnel). Control software would set up these flows through the SoC by updating or creating data structures describing how packets in the flow should be treated. The DPAA prescribes specific data structures to be created by the control processor at flow establishment time, and also defines how datapath processors should interact with these data structures in order to move packets through the off-load engines and outbound network interfaces with the least CPU overhead. The DPAA also provides for a uniform programming interface for the hardware accelerators and network interfaces. Rather than porting multiple distinct device drivers to control and datapath software, the user ports a unified Queue Manager Driver.

It is important to note that the Datapath Acceleration Architecture is not an all-or-nothing programming model. It is possible to use legacy implementations of packet classification and buffer management and still take advantage of the Queue Interface Driver’s simplified, software-friendly interface. Operations are encoded in architected messages which are placed on queues, and responses (normal and error) flow back to software using the same queue structures.

Figure 9 shows a diagram of queuing messages to channels.

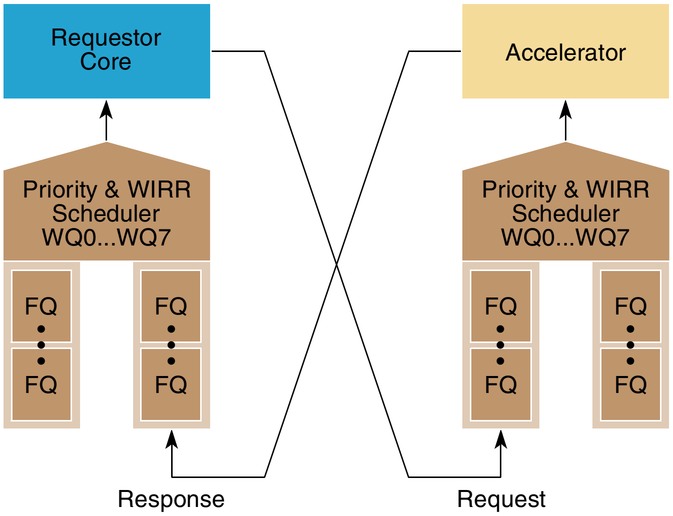
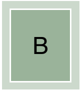

Requestor Core
Accelerator
Priority & WIRR Scheduler WQ0...WQ7
Priority & WIRR Scheduler WQ0...WQ7
FQ
FQ
FQ
FQ
FQ
FQ
FQ
FQ
Response
Request

Figure 9. Queuing Messages to Channels

2.3.6.2 Definitions

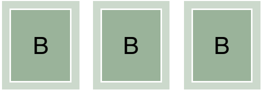
Buffer

Region of contiguous memory, allocated by software, managed by the DPAA Buffer Manager.



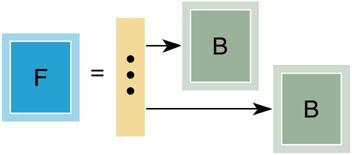
Buffer Pool

Set of buffers with common characteristics (mainly size, alignment, access control)



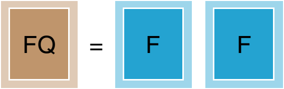
Frame

Single buffer or list of buffers that hold data, for example, packet payload, header, and other control information.



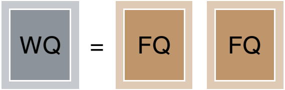
Frame Queue

FIFO of Frames



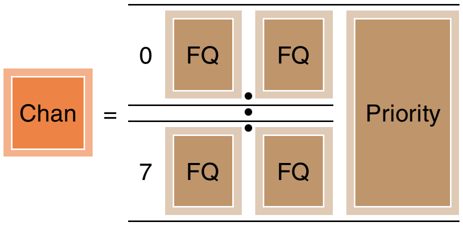
Work Queue

FIFO of Frame Queues



Channel

Set of eight Work Queues, with HW provided prioritized access



Dedicated Channel

Channel statically assigned to a particular end point, from which that end point can dequeue frames. End point may be a CPU, FMan, PME, or SEC.

Pool Channel

A channel statically assigned to a group of end points, from which any of the end points may dequeue frames.

2.3.6.3 Packet Walk Through

This section walks a packet through the P4080 to illustrate how the DPAA off-loads and accelerates packet forwarding while leaving key decisions under software control.

Datapath processing begins when Ethernet frames arrive at a network interface, assumed to be one of the Ethernet MACs within a Frame Manager. Alternatively, packets can arrive across a peripheral bus from an external network interface, in which case, the same packet walk-through stages can be applied with the help of a CPU acting as an I/O processor.

After the FMan receives the Ethernet frame, it requests one or more buffers (from the HW Buffer Manager) to store the frame. The BMan maintains pools of buffers, each with software-defined characteristics, and the FMan is initialized to request a buffer from the most appropriate pool. If a sufficiently large buffer cannot be found for the incoming frame, the FMan stores the frame across several smaller buffers and create a scatter/gather list for these buffers.

The FMan’s configurable parsing and filing capabilities can perform initial classification, sufficient to steer a packet toward a control processor, or toward one of the datapath processors for flow-specific processing (or additional classification by means of software). The steering of a packet towards a processor or a group of processors could be also based on differentiating flows by means of the Quality of Service attributes of the flow (for example, DSCP, IP precedence, or by user-defined proprietary headers).

Steering is accomplished by the FMan issuing an enqueue command to the Queue Manager with the designated Frame Queue ID, along with frame parameters such as Frame Queue ID, Color Marking, and optional Sequence Number.

In this example, the FMan's initial classification causes it to select a Frame Queue ID which the Queue Manager uses to steer the Frame Queue to the dedicated channel of logical CPU#3, a CPU operating in a datapath role. Potentially, the Frame Queue could be selected based on the QoS requirements of the flow with a policing profile associated with the selection.

Continuing the example, the Queue Manager places the Frame Queue onto Work Queue#5 of CPU#3's dedicated channel. The Frame Queue was queued to CPU#3 due to user configuration decisions that all packets belonging to this specific flow should be processed by CPU#3, possibly to take advantage of special processing instructions locked in CPU#3's private cache, or due to user-defined load balancing. The Frame Queue was placed in Work Queue#5 for QoS reasons, as the amount of traffic processed from each Work Queue relative to other work queues is also user-configurable. All Frame Queues on a Work Queue have equal priority, but the amount of traffic that the CPU draws from each is user-configurable to allow fairness and appropriate bandwidth allocation. The relative priority of a Work Queue with respect to other Work Queues in the Channel is user-configurable.

Once configured, the Queue Manager can take care of the packet/frame level scheduling requirements of the CPU, that is, the Queue Manager would appropriately schedule the Work Queue and Frame Queue within the Work Queue. The Queue Manager can be configured to perform a stashing operation in response to a CPU (or co-processor) accessing a Frame Queue.

CPU#3 performs protocol processing on the packet, including modification of the packet data in the buffer. Any modifications which change the length of the packet would be noted by means of changes to the Frame. The CPU determines that the packet requires IPsec ESP processing, and enqueues the frame back to the QMan using the FQ ID associated with the packet's specific ESP tunnel.

The Queue Manager uses this Frame Queue ID to determine that the next consumer of the Frame Queue is the SEC 4.0, and places the Frame Queue onto Work Queue#5 of the SEC's dedicated channel. The SEC pulls Frame Queues from the Work Queues in its dedicated channel according to user-defined weights in a WFQ model.

The SEC dequeues and processes data from the Frame Queue, adding the tunnel IP header, ESP header, IV, trailer, and HMAC, and writing encrypted data to either the original buffers, or new buffers which the SEC requests from the HW Buffer Manager. The SEC updates the Frame description (length change due to the addition of the headers, trailers, and HMAC) and enqueues the FQ back to the Queue Manager using a configured FQ ID. In this case, the FQ ID causes the QMan to enqueue the FQ to Work Queue #5 of logical CPU#4's dedicated channel. CPU#4 dequeues the FQ, determines the outbound interface for the newly encrypted packet, updates the tunnel IP header, and enqueues the Frame back to the QMan on a new FQ ID. This FQ ID causes the QMan to enqueue the FQ to a channel serviced by Frame Manager #2, which dequeues the FQ, transmits one or more packets from that FQ out the appropriate dTSEC, and releases the buffers back to the Buffer Manager.

The processing pipeline used in this example is not required by the P4080 DPAA. The initial classification could have caused the packet to be steered toward a CPU dedicated to fine-grained classification, or to a pool channel of CPUs, any of which could have performed the operations described. CPU#3 could have added the ESP header and trailer to the packet and sent it to the SEC for crypto-only processing. Following

SEC processing, the flow was steered to CPU#4, however it could have just as easily been steered back to CPU#3, or to a pool channel. At any FQ ID transition, the relative priority of the flow could have been elevated or reduced by enqueuing it to a different work queue.

2.3.7 Major DPAA Components

The PowerQUICC Datapath Acceleration Architecture, shown in Figure 10, includes the following major components:

- Frame Manager
- Queue Manager
- Buffer Manager
- SEC 4.0
- PME 2.0

The role of most of these blocks is described in Section 2.3.6.3, “Packet Walk Through.” Each is described in more detail below.

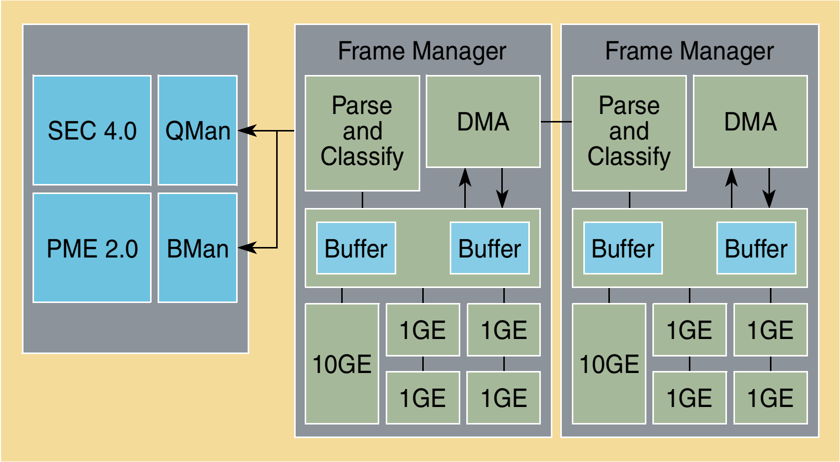


Figure 10. PowerQUICC Datapath Acceleration Architecture

2.3.7.1 Frame Manager

The P4080 is the first product in the PowerQUICC family to incorporate a Frame Manager. A Frame Manager is a functional unit which combines the Ethernet network interfaces with Packet Distribution logic to provide intelligent distribution and queuing decisions for incoming traffic at line rate (18 Mpps). This integration allows the Frame Manager to perform configurable parsing and classification of the incoming frame with the purpose of selecting the appropriate input frame queue for expedited processing by a CPU or pool of CPUs.

2.3.7.1.1 Network Interfaces

Each of the two Frame Managers in the P4080 integrates 4 datapath tri-speed Ethernet controllers (dTSECs) and one 10-Gbit Ethernet controller. The more basic parsing and filing capability found in prior

PowerQUICC eTSECs is removed from the MACs themselves, and aggregated in the more flexible and robust parsing and classification logic described in the next section.

The Ethernet controllers support programmable CRC generation and checking, RMON statistics, and jumbo frames of up to 9.6 kbytes. They are designed to comply with IEEE Std. 802.3™, 802.3u™, 802.3x™, 802.3z™, 802.3ac™, 802.3ab™, and additionally the 1 Gbps MACs support IEEE Std. 1588™ v2 (clock synchronization over Ethernet).

The dTSECS are capable of full- and half-duplex Ethernet support (1000 Mbps supports only full duplex); the 10-Gbit MACs are single-speed full duplex. Both support IEEE 802.3 full-duplex flow control (automatic PAUSE frame generation or software-programmed PAUSE frame generation and recognition)

SERDES flexibility makes it possible to enable up to 14 Gbps of Ethernet traffic on a single Frame Manager, however each Frame Manager can support line rate parsing and classification at 12 Gbps. In this situation, it may be desirable to configure one FMan to process frames from the 10 Gbps interface, and the second FMan to process frames from the four individual GE interfaces. The Frame Managers are designed to process up to 12Gbps each.

See Appendix A, “SERDES Options,” for more information on SERDES usage for Ethernet and high speed peripheral interfaces.

When all SERDES are otherwise allocated, it is possible to enable two of dTSECs by means of RGMII or RMII physical interfaces.

2.3.7.1.2 Parse Function

The primary function of the Packet Parse logic is to identify the incoming frame for the purpose of determining the desired treatment to apply. This Parse function can parse many standard protocols, including options and tunnels, and supports a generic configurable capability to allow proprietary or future protocols to be parsed.

Proprietary headers can be defined as being self-describing or non-self-describing. Self-describing headers are 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. In contrast, non-self-describing proprietary headers do not contain any information that indicates the presence of the header. For example, a frame that always contains a proprietary header before the Ethernet header would be non-self-describing. Both self-describing and non-self-describing headers are supported by means of parsing rules in the Frame Manager.

The underlying notion is that different frames may require different treatment, and only through detailed parsing of the frame can proper treatment be determined. Parse results can (optionally) be passed to software.

2.3.7.1.3 Distribution and Policing

Once parsing is complete, the treatment can be identified. That treatment can be to hash selected fields in the frame as part of a spreading mechanism or that treatment can be to look up certain fields in the frame to determine subsequent action to take, including policing.

The hash capability allows a hash key to be built from many different fields in the frame to provide differentiation between flows. Default values can be used if fields are not present in the frame. Alternatively, a key can be built exclusively from fields present in the frame based on what the Parse Function has detected. The result of the hash is a specific Frame Queue identifier. To support added control, this FQID can be indexed by values found in the frame, such as TOS or p-bits or any other desired field(s). This is useful when it is required to spread traffic while obeying QoS constraints.

In some situations, a hash distribution based on parse results may be insufficient, and a more detailed examination of the frame is required. Therefore, instead of, or prior to the hash, the Frame Manager supports a classification capability to look up fields in the frame to determine the action to take. The Frame Manager contains internal memory that holds small tables for this purpose.

Classification lookups are performed based on the combination of user configuration and what fields the parser actually encountered. That is, the user configures the sets of lookups to perform, and the parse results dictate which one of those sets to use. Lookups can be chained together such that a successful lookup can provide key information for a subsequent lookup. After all the lookups are complete, the final classification result provides either a hash key to use for spreading, or a FQ ID directly.

In addition to selecting the FQ ID, classification can determine whether policing is required and the policing context to use.

This choice of Frame Queue could depend on the flow being either directed to a particular CPU, quality of service consideration, control plane/data plane traffic, etc. For example, the most obvious one is the distribution of flows on Frame Queues based on their DSCP or IP precedence bits. Thus, up to 8 different Frame Queues would be created. CPUs which service those Frame Queues can then preferentially schedule these queues or let the Queue Manager perform preferential scheduling for the CPUs. Since the Frame Manager has up to 256 policing profiles, any Frame Queue or group of Frame Queues can be policed to either drop or mark packets if the flow exceeds a preconfigured rate. The policing function in conjunction with classification can be used for mitigating Distributed Denial of Service Attack (DDOS).

The policing is based on Two-Rate-Three-Color Marking algorithm (RFC 2698). The sustained and peak rates as well as the burst sizes are user-configurable. Hence, the policing function can rate-limit traffic to conform to the rate the flow is mapped to at flow setup time. By prioritizing and policing traffic prior to software processing, CPU cycles can be focused on the important and urgent traffic ahead of other traffic.

2.3.7.2 Queue Manager

The Queue Manager (QMan) is the main component in the DPAA that allows for simplified sharing of network interfaces and hardware accelerators by multiple CPU cores. It also provides a simple and consistent message and data passing mechanism for dividing processing tasks amongst multiple CPU cores.

The Queue Manager offers the following features:

- Common interface between software and all hardware
  - controls the prioritized queuing of data between multiple processor cores, network interfaces, and hardware accelerators
  - supports both dedicated and pool channels, allowing both push and pull models of multicore load spreading

- Atomic access to common queues without software locking overhead
- Mechanisms to guarantee order preservation with atomicity and order restoration following parallel processing on multiple CPUs
- Two level queuing hierarchy with one or more Channels per Endpoint, eight Work Queues per Channel, and numerous Frame Queues per Work Queue
- Priority and work conserving fair scheduling between the Work Queues and the Frame Queues
- Loss-less flow control for ingress network interfaces
- Congestion avoidance (RED/WRED) and congestion management with tail discard and up to 256 congestion groups. Each congestion group is composed of a user-configured number of Frame Queues. Congestion notification is sent to an Endpoint when the sum of the buffer occupancies of the group's Frame Queues reaches a pre-determined congestion threshold. Congestion group status has hysteresis built in.

2.3.7.3 Buffer Manager

The Buffer Manager (BMan) manages pools of buffers on behalf of software for both hardware (accelerators and network interfaces) and software use.

The Buffer Manager offers the following features:

- Common interface for software and hardware
- Guarantees atomic access to shared buffer pools
- Supports 64 buffer pools
  - SW, HW buffer consumers can request different size buffers and buffers in different memory partitions
- Supports depletion thresholds with congestion notifications
- On-chip per pool buffer stockpile to minimize access to memory for buffer pool management
- LIFO (last in first out) buffer allocation policy
  - Optimizes cache usage and allocation
  - A released buffer is immediately used for receiving new data

2.3.7.4 Security Engine (SEC 4.0)

The SEC 4.0 is the PowerQUICC's fourth generation crypto-acceleration engine. In addition to off-loading cryptographic algorithms, the SEC 4.0 offers header and trailer processing for several established security protocols. The SEC 4.0 includes 5 Descriptor Controllers (DECOs), which are updated versions of the previous SEC crypto-channels. DECOs are responsible for header and trailer processing, and managing context and data flow into the CHAs assigned to it for the length of an operation.

The DECOs can perform header and trailer processing, as well as single pass encryption/integrity checking for the following security protocols:

- IPsec
- SSL/TLS
- SRTP

- IEEE 802.1AE MACSec
- IEEE 802.16e WiMax MAC layer
- 3GPP RLC encryption/decryption

In prior versions of the SEC, the individual algorithm accelerators were referred to as Execution Units (EUs). In the SEC 4.0, these are referred to as Crypto Hardware Accelerators (CHAs) to distinguish them from prior implementations. Specific CHAs available to the DECOs are listed below.

- Advanced Encryption Standard unit (AESA)
- ARC four execution unit (AFHA)
- Cyclic Redundancy Check Accelerator (CRCA)
- Data Encryption Standard execution unit (DESA)
- Kasumi execution unit (KFHA)
- SNOW 3G Hardware Accelerator (STHA)
- Message digest execution unit (MDHA)
- Public key execution unit (PKHA)
- Random number generator (RNGB)

Depending on the security protocol and specific algorithms, the SEC 4.0’s aggregate symmetric encryption/integrity performance is 10 Gbps, while asymmetric encryption (RSA public key) performance is ~10,000 1024b RSA operations per second.

The SEC 4.0 is also part of the PowerQUICC Trust Architecture, which gives the P4080 the ability to perform secure boot, runtime code integrity protection, and session key protection. The Trust Architecture is described in Section 2.4, “Resource Partitioning and QorIQ Trust Architecture.”

Figure 11 shows the block diagram for SEC 4.0.

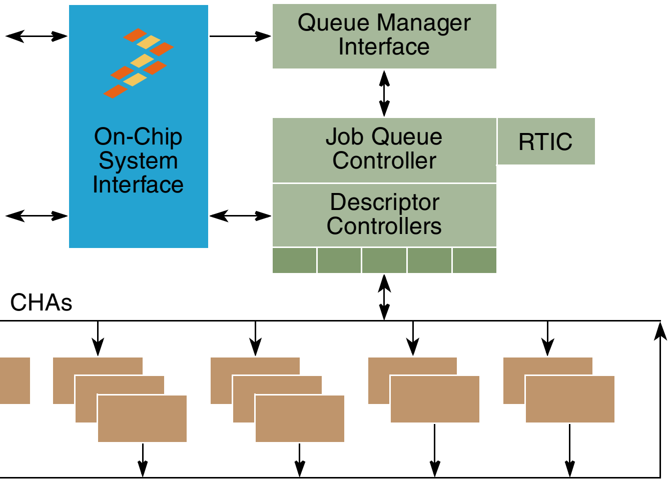


Figure 11. SEC 4.0 Block Diagram

2.3.7.5 Pattern Matching Engine (PME) 2.0

The Pattern Matching Engine is a self-contained hardware block capable of autonomously scanning data from streams for patterns that match a specification in a database dedicated to it. The PME 2.0 is an updated version of the PME used in previous members of the PowerQUICC family. Specific updates include a Queue Manager Interface, supporting the DPAA Queue Interface Driver, and a 4x increase in the number of patterns (16K →64K) and stateful rules (8K→32K) supported. Raw scanning performance has also increased 4x to ~10 Gbps.

Patterns that can be recognized (matched) by the PME are of two general forms, byte patterns and event patterns. Byte patterns are simple matches such as “abcd123” existing in both the data being scanned and in the pattern specification database. Event patterns are a sequence of multiple byte patterns. In the PME, event patterns are defined by stateful rules.

The PME specifies patterns of bytes as regular expressions (Regex). The P4080 (by means of an on-line or off-line process) converts Regex patterns into the PME’s pattern specification database. Generally there is a one-to-one mapping between a regular expression and a PME byte pattern. The PME’s use of regular expression pattern-matching offers built-in case-insensitivity and wildcard support with no pattern explosion, while the PME’s NFA-style architecture offers fast pattern database compilation and fast incremental updates. Up to 64,000 regular expression patterns are supported, each up to 128 bytes long. The 64,000 regular expression patterns can be combined by means of stateful rules to detect a far larger set of event patterns. Comparative compilations against DFA style regular expression engines have shown that 300,000 DFA pattern equivalents can be achieved with ~8000 PME regular expressions with stateful rules.

Within the PME, match detection precedes in stages. The Key Element Scanner performs initial byte pattern matching, with handoff to the Data Examination Engine for elimination of false positives through more complex comparisons. As the name implies, the Stateful Rule Engine receives confirmed basic matches from the earlier stages, and monitors a stream for addition for subsequent matches that define an event pattern.

Figure 12 shows the block diagram for PME 2.0.

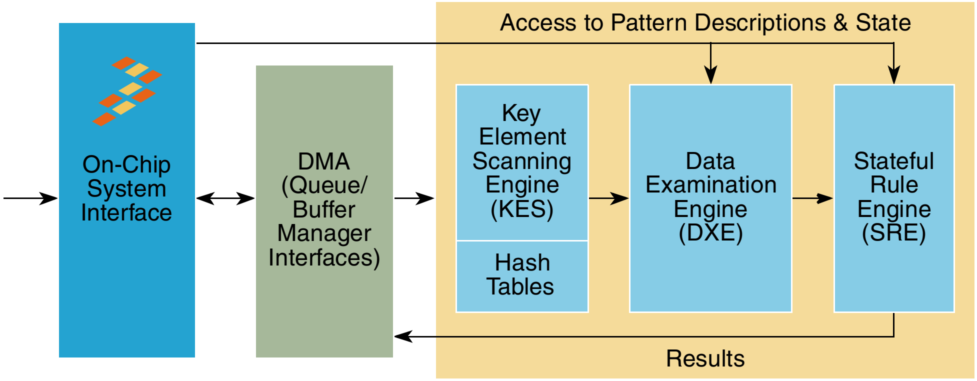

Access to Pattern Descriptions & State
On-Chip System Interface
DMA (Queue/ Buffer Manager Interfaces)
Key Element Scanning Engine (KES)
Hash Tables
Data Examination Engine (DXE)
Stateful Rule Engine (SRE)
Results

Figure 12. PME 2.0 Block Diagram

2.4 Resource Partitioning and QorIQ Trust Architecture

Consolidation of discrete CPUs into a single multicore SoC and potential repartitioning of legacy software on those cores introduces many opportunities for unintended resource contentions to arise. A system may exhibit erratic behavior if the multiple CPUs don't effectively partition and share system resources. While it can be challenging to prevent unintended resource contention, stopping malicious software is much more difficult. 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.

The P4080 offers a new level of hardware partitioning support, allowing system developers to ensure software running on any CPU only accesses the resources (memory, peripherals, etc.) that it is explicitly authorized to access. This may not seem like a challenge in a SMP environment, as the OS performs resource allocation for the applications running on it, but it is a very difficult problem in AMP environments, where there may be multiple instances of the same OS, or even different OSes running on the various CPU cores. Even OS protections in an SMP system may be insufficient in the presence of malicious software.

2.4.1 e500mc MMU and Embedded Hypervisor

The P4080's first line of defense against unintended interactions amongst the multiple CPUs/OSes is each e500mc core's MMU. Each e500mc core'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 (portion of memory, peripheral device) is dedicated to a single CPU, that CPU's MMU is configured to allow access to those addresses (on 4-kbyte granularity); other CPU MMUs are not configured for access to the other CPU's private memory range. When two CPUs 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 SoC, achieving strong partitioning shouldn't require the developer to map functions onto cores 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 e500mc MMU also includes embedded Hypervisor extensions. Each e500mc MMU supports three levels of instructions: User, Supervisor (OS), and Hypervisor. An embedded Hypervisor micro-kernel (provided by Freescale as source code) runs unobtrusively beneath the various OSes running on the CPUs, 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 core attempts to overwrite important device configuration registers (including CPU MMUs), the embedded Hypervisor blocks the write. Other examples of embedded Hypervisor managed resources are high- and low-speed peripheral interfaces (PCIe, UART) if those resources are not dedicated to a single CPU/partition.

2.4.2 Peripheral Access Management Unit (PAMU)

Being MMU-based, the embedded Hypervisor is only able to stop unauthorized software access attempts. Internal components with bus mastering capability (PME, SEC, Frame Manager, etc.) also need to be prevented from reading and writing to specific memory regions. These devices won't spontaneously generate access attempts, but if programmed to do so by buggy or malicious software, any of them could overwrite sensitive configuration registers and crash the system. For this reason, the P4080 also includes a distributed function, collectively referred to as the Peripheral Access Management Unit (PAMU), which provides address translation and access control for all bus masters in the system. PAMU access control can be absolute (FMan, PME, SEC, other bus masters can never access memory range XYZ), or it can be conditional, based on the Partition ID of the CPU that programmed the bus master.

2.4.3 Secure Boot and Sensitive Data protection

The e500mc MMUs and PAMU allow the P4080 to enforce a consistent set of memory access permissions on a per-partition basis. When combined with embedded Hypervisor for safe sharing of resources, the P4080 becomes highly resilient when poorly tested or malicious code is run. For system developers building high reliability/high security platforms, rigorous testing of code of known origin is the norm.

For this reason, the P4080 offers a secure boot option, in which the system developer digitally signs the code to be executed by the CPU coming out of reset, and the P4080 insures that only an unaltered version of that code runs on the platform. The P4080 offers both boot time and run time code authenticity checking, and offers configurable consequences when the authenticity check fails. The P4080 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 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 P4080 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 4.0 for decryption of session traffic.

2.5 Advanced Power Management

Power management is always a major design consideration in embedded applications, and for this reason, the P4080 is designed to dissipate <30 W Max with all 8 CPUs and platform logic running at maximum frequency (8x 1.5 GHz CPUs, 800 MHz CoreNet). <30 W max power is aligned with the power envelopes of typical network and wireless infrastructure system chassis.

The P4080 does not rely on statistical processing loads or forced voltage/frequency reductions to achieve its <30 W max operation. Dynamic voltage and frequency scaling (DVFS) are useful techniques for reducing typical/average power and maximizing battery life in laptop environments, but embedded applications have to be designed for rapid response to bursts of traffic and max power under worst case environmental conditions. While not implementing DVFS in the PC sense, the P4080 does actively manage internal clocks to avoid wasting energy. Clock signals are disabled to idle components, reducing dynamic power. These blocks can return to full operating frequency on the clock cycle after work is dispatched to them. Based on transistor-switching rates and clock gating opportunities associated with high-throughput packet forwarding applications, a more typical power dissipation is ~23 W.

In addition to internal clock gating, the P4080’s advanced power management capabilities are based around fine-grained static clock control, and SW-controlled dynamic frequency management.

Fine-grained static control allows developers to turn off the clocks to individual logic blocks within the SoC that the system has no need for. Based on a finite number of SERDES, it is expected that any given application has some Ethernet MACs, PCIe, or serial RapidIO controllers inactive. These blocks can be disabled by means of the DEVDIS Register. Re-enabling clocks to a logic block requires an SoC reset, which makes this type of power management operation infrequent (effectively static).

Another aspect of the P4080’s Advanced Power Management is the amount of control over voltage and frequency available to the system developer. The eight e500 cores draw their power from three power rails, each of which can be set to a different supply voltage. When the usage model calls for AMP or mixed AMP/SMP, two, four, or six of the P4080’s e500mc cores can be turned off, or run at .9-V. Two cores can be supplied with 1.0-V, and run at higher frequencies. System developers can use their knowledge of their specific application to determine the P4080’s static power management configurations (CPU supply voltage, max operating frequency, unneeded components) to enable operating at the minimum power consumption for that application.

In addition to fine-grained static power management, the P4080 platform supports (under software control) dynamic changes to CPU operating frequencies and voltages. Each CPU sources its input clock from one of three independent PLLs inside the P4080. Each CPU can also source its input clock from an integer frequency divider from two of the three independent PLLs. CPUs can switch their source PLL, and their frequency divider glitchlessly and nearly instantaneously. This allows each core to operate at the minimum frequency required to perform its assigned function, saving power.

Changing PLL frequency dividers (/2, /4) can be used to achieve large and rapid reductions in dynamic power consumptions, and with the help of external temperature detection circuitry, can serve as a thermal overload protection scheme. If the junction temperature of the P4080 (or system ambient temperature) achieves some critical level, external temperature detection circuitry can drive a high-priority interrupt into the P4080, causing it to reduce selected CPU frequencies by half or more. This allows the system to continue to function in a degraded mode, rather than failing entirely. This technique is much simpler than turning off selected CPUs, which can involve complex task migration in an AMP system. When system temperatures have been restored to safe ranges, all CPUs can be returned to normal frequency within a few clock cycles.

When less drastic frequency changes are desired, software can switch the CPU to a slower speed PLL, such as 1 GHz versus 1.5 GHz. Many cores could be switched to a slower PLL during periods of light traffic, with the ability to immediately return those cores to the full rate PLL should traffic suddenly increase. The more traditional Power Architecture single core power management modes (Core Doze, Core Nap, Core Sleep) are also available in the e500mc.

To change a CPU’s voltage, the two or four CPUs on a particular power rail are placed in Nap mode, and a signal is sent to an external voltage regulator. Once the new voltage is stable, the CPUs can be brought out of Nap mode without a reset. Because Nap mode requires a CPU to flush its caches and reach a completely idle state, changing voltages is more of a quasi-static change, likely used in association with a longer term change in CPU role.

2.6 Debug Support

The reduced number of external buses enabled by the move to multicore SoCs greatly simplifies board level lay-out and eliminates many concerns over signal integrity. While 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 HW engineer, they provide software developers with the ultimate confirmation that the proper instructions and data are passing between processing elements.

Processing on a multicore SOC with shared caches and peripherals also leads to greater concurrency and an increased potential for unintended CPU interactions. To ensure that software developers have the same or better visibility into the P4080 as they would with multiple discrete Freescale communications processors, Freescale developed the debug architecture shown in Figure 13.

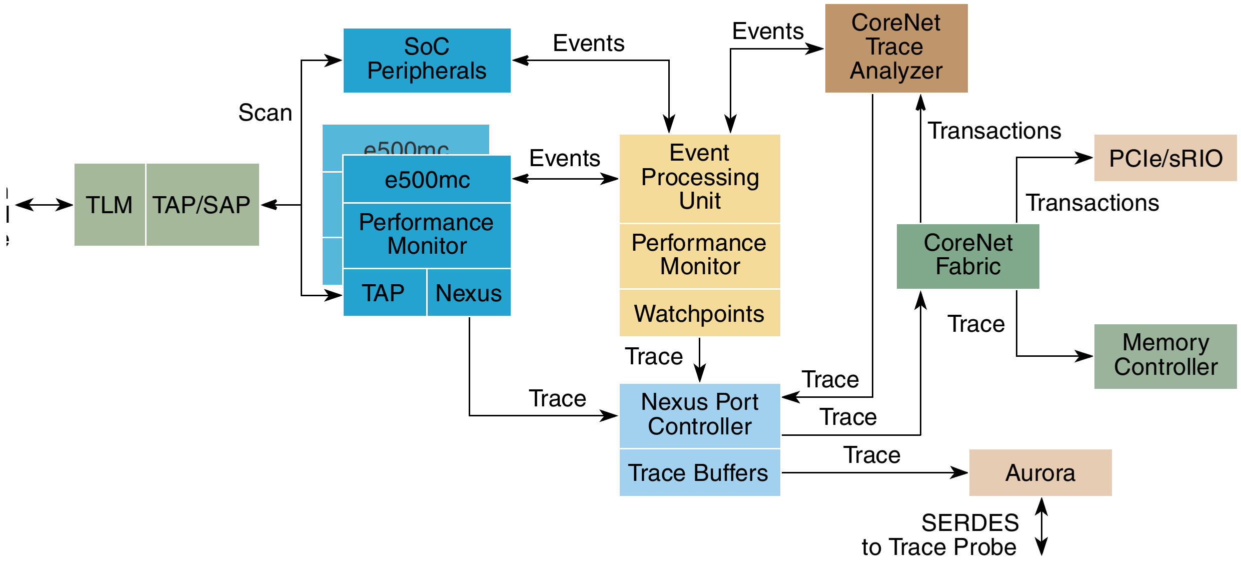


Figure 13. P4080 Debug Architecture

Debug features include the following:

- Debug and performance monitoring registers in both the e500mc and platform
  — Accessible by target resident debug software and non-resident debug tools
  — Capable of generating debug interrupts and trace event messages
- Run control with enhancements
  — Classic
  — Cross-core and SoC watchpoint triggering
- High speed trace port (Aurora-based)
  — Supports Nexus class 2 instruction trace including timestamps

    - Process id trace, watchpoint trace
  - Supports “light” subset of Nexus class 3 data trace
    - Enabled by cores, by event triggers, by Instruction Address Compare/Data Address Compare events
  - Data Acquisition Trace
    - Compatible with Nexus class 3
    - Instrumented code can generate data trace messages for values of interest
    - Performed by writing values to control registers within each e500mc core
  - Watchpoint Trace
    - Can generate cross-core correlated breakpoints
    - Breakpoint on any core can halt execution of selected additional cores with minimal skid
- CoreNet transaction analyzer
  - Provides visibility to transactions across CoreNet (CoreNet fabric is otherwise transparent to software)
  - Generates trace messages to Nexus Port Controller
  - Supports filtering of accesses of interest
    - Data Address Compare (4)
    - Data Value Compare (2)
    - Transaction Attribute Compare (2)

3 Developer Environment

Software developers creating solutions with the Power Architecture have long benefited from a vibrant support ecosystem, including high quality tools, OSes, and network protocol stacks. Freescale is working with our ecosystem partners to insure that this remains the case for multicore Power architecture products, led by the P4080. The various levels of the developer’s environment are shown in Figure 14, with the more broadly used tools and boards at the base of the pyramid, and increasingly application-specific enablement items at the top.

The foundation of the pyramid consists of the traditional tools available to Power architecture software developers, such as development systems, compilers, and debuggers. Development systems are available from both Freescale and our partners, with some partner systems being offered with form factors and BOMs to support use as reference designs.

Current Freescale development systems are supported by the open source GNU tool set, including compilers, linkers, and debuggers. In active partnership with the open source community and Linux™ distribution and support suppliers, these tools are updated to fully and efficiently support the P4080. Open source tools are part of an overall P4080 development board Linux support package, which includes AMP and SMP versions of the Linux OS, and device drivers for the accelerators and networking and peripheral interfaces featured in the P4080. AMP Linux support includes the ability to boot multiple instances of Linux on different cores. Freescale’s dual-core Power Architecture products (MPC8572, MPC8641) already enjoy the support of industry-leading RTOS suppliers, and multiple Power Architecture ecosystem partners are committed to providing board support packages for the P4080.

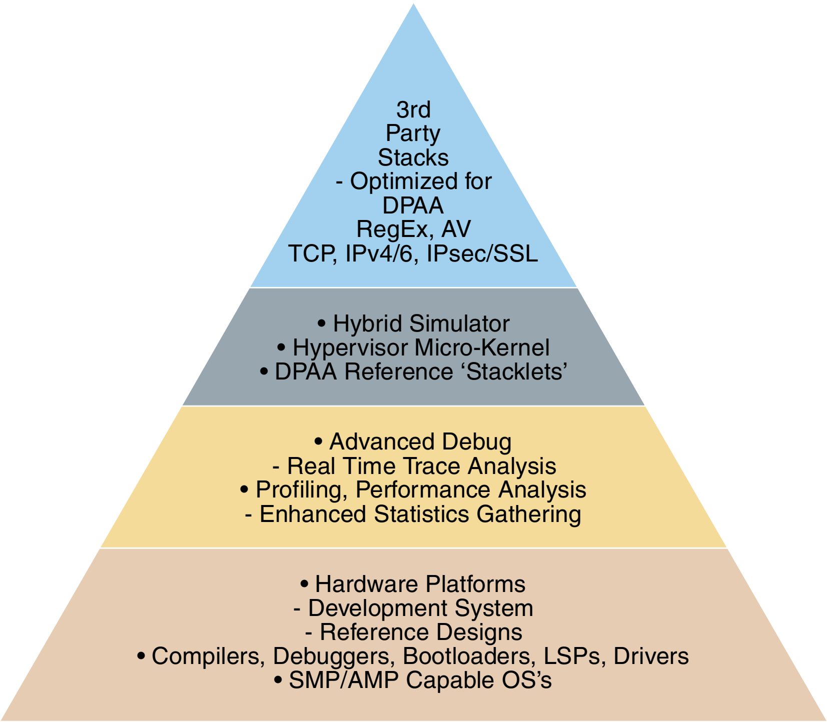

3rd
Party
Stacks
- Optimized for
DPAA
RegEx, AV
TCP, IPv4/6, IPsec/SSL
• Hybrid Simulator
• Hypervisor Micro-Kernel
• DPAA Reference 'Stacklets'
• Advanced Debug
- Real Time Trace Analysis
• Profiling, Performance Analysis
- Enhanced Statistics Gathering
• Hardware Platforms
- Development System
- Reference Designs
• Compilers, Debuggers, Bootloaders, LSPs, Drivers
• SMP/AMP Capable OS's

Figure 14. P4080 Multicore Enablement Pyramid

While the first level of the enablement pyramid allows the developer to perform initial system bring-up and development, the next level is often required to deal with the special challenges of software debugging and performance analysis in multicore systems. As highlighted in Section 2.6, "Debug Support," the P4080 offers sophisticated new capabilities in these areas. Freescale will bring tools support for these features to the market both by means of our CodeWarrior line of tools and in partnership with industry standard tools suppliers.

Without a single operating system performing coordination and access control functions, managing shared resources in an efficient manner can be a challenge. The P4080's e500mc cores offer an embedded Hypervisor capability to address this need. The embedded Hypervisor unobtrusively provides the software layer needed to manage the operating systems and supervisor-level applications as they access shared resources. Recognizing that each developer's system design may call for a different partitioning of resources, and involve different combinations of OSes and RTOSes, Freescale and our ecosystem partners will provide reference implementations of the embedded Hypervisor's peripheral virtualization and access control which the developer can modify to match unique system requirements.

Many of the expected applications for the P4080 involve network protocol processing. It is expected that some CPUs will be dedicated as datapath processors, working closely with the Datapath Acceleration Architecture. Freescale will provide reference protocol "stacklets," optimizing performance critical regions of protocol processing and their interaction with the DPAA hardware.

A simulator can be an extremely valuable tool in multicore code development and partitioning. Simulator capabilities include the following:

- Global visibility
- Determinism
- Bug reproducibility
- Reverse execution
- Special abilities to detect race conditions
- Ability to detect race conditions

In conjunction with Virtutech®, Freescale provides a hybrid simulator that combines both functional and performance measurement models of the P4080. The hybrid simulator allows the user to switch between "Fast Functional Mode" and "Detailed Performance Mode."

To support customers needing more that reference implementations of fast path network forwarding paths, Partitioning between control CPUs and datapath CPUs, and developing the protocol processing firmware which runs on the datapath CPUs is an area for significant value added services for Freescale partners at the top level of the enablement pyramid. OEMs wishing to engage with these partners can realize significant 'time to performance' advantages.

4 Document Revision History

Table 1 provides a revision history for this product brief.

Table 1. Revision History

Revision	Date	Substantive Change(s)
1	08/2008	• Changed 1.1 V to 1.0 V throughout. • Changed 1023 pins to 1295 pins.
0	07/2008	Initial public release

Appendix A SERDES Options

The P4080 integrates 18 high-speed SERDES, which can be assigned to network (SGMII, XAUI) or peripheral (PCIe, serial RapidIO) interfaces. Two dedicated lanes of SERDES are available for debug. The SERDES lanes can be assigned to these interfaces as shown in Table 2.

Table 2. SERDES Configuration Options

<table>
<tr><th colspan="7">Bank 1</th><th colspan="4">Bank 2</th><th colspan="4">Bank 3</th></tr>
<tr><td colspan="6">PCIe</td><td>Debug</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td colspan="6">PCIe</td><td>Debug</td><td colspan="4">XAUI</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td colspan="6">PCIe</td><td>Debug</td><td colspan="4">XAUI</td><td colspan="4">XAUI</td></tr>
<tr><td colspan="2">PCIe</td><td colspan="4">PCIe</td><td>Debug</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td colspan="2">PCIe</td><td colspan="4">PCIe</td><td>Debug</td><td colspan="4">XAUI</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td colspan="2">PCIe</td><td colspan="4">PCIe</td><td>Debug</td><td colspan="4">XAUI</td><td colspan="4">XAUI</td></tr>
<tr><td>PCIe</td><td>PCIe</td><td colspan="4">PCIe</td><td>Debug</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td>PCIe</td><td>PCIe</td><td colspan="4">PCIe</td><td>Debug</td><td colspan="4">XAUI</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td>PCIe</td><td>PCIe</td><td colspan="4">PCIe</td><td>Debug</td><td colspan="4">XAUI</td><td colspan="4">XAUI</td></tr>
<tr><td colspan="2">PCIe</td><td colspan="4">PCIe</td><td>Debug</td><td colspan="4">PCIe</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td colspan="2">PCIe</td><td colspan="4">PCIe</td><td>Debug</td><td colspan="4">PCIe</td><td colspan="4">XAUI</td></tr>
<tr><td colspan="2">PCIe</td><td colspan="2">PCIe</td><td>SGMII</td><td>SGMII</td><td>Debug</td><td colspan="4">PCIe</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td colspan="2">PCIe</td><td colspan="2">PCIe</td><td>SGMII</td><td>SGMII</td><td>Debug</td><td colspan="4">PCIe</td><td colspan="4">XAUI</td></tr>
<tr><td colspan="2">PCIe</td><td colspan="2">PCIe</td><td>SGMII</td><td>SGMII</td><td>Debug</td><td colspan="4">XAUI</td><td colspan="4">XAUI</td></tr>
<tr><td>PCIe</td><td>PCIe</td><td colspan="2">PCIe</td><td>SGMII</td><td>SGMII</td><td>Debug</td><td colspan="4">XAUI</td><td colspan="4">XAUI</td></tr>
<tr><td colspan="2">PCIe</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>Debug</td><td colspan="4">XAUI</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td>PCIe</td><td>PCIe</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>Debug</td><td colspan="4">XAUI</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td colspan="2">SRIO</td><td colspan="4">SRIO</td><td>Debug</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td colspan="2">SRIO</td><td colspan="4">SRIO</td><td>Debug</td><td colspan="4">XAUI</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td colspan="2">SRIO</td><td colspan="4">SRIO</td><td>Debug</td><td colspan="4">XAUI</td><td colspan="4">XAUI</td></tr>
<tr><td colspan="2">SRIO</td><td colspan="4">SRIO</td><td>Debug</td><td colspan="4">PCIe</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td colspan="2">SRIO</td><td colspan="4">SRIO</td><td>Debug</td><td colspan="4">PCIe</td><td colspan="4">XAUI</td></tr>
<tr><td>PCIe</td><td>PCIe</td><td>X</td><td>SRIO</td><td>X</td><td>SRIO</td><td>Debug</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td>PCIe</td><td>PCIe</td><td>X</td><td>SRIO</td><td>X</td><td>SRIO</td><td>Debug</td><td colspan="4">XAUI</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td>PCIe</td><td>PCIe</td><td>X</td><td>SRIO</td><td>X</td><td>SRIO</td><td>Debug</td><td colspan="4">XAUI</td><td colspan="4">XAUI</td></tr>
<tr><td colspan="2">SRIO</td><td colspan="4">PCIe</td><td>Debug</td><td colspan="4">PCIe</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td colspan="2">SRIO</td><td colspan="4">PCIe</td><td>Debug</td><td colspan="4">PCIe</td><td colspan="4">XAUI</td></tr>
<tr><td colspan="2">PCIe</td><td colspan="4">SRIO</td><td>Debug</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
<tr><td colspan="2">PCIe</td><td colspan="4">SRIO</td><td>Debug</td><td colspan="4">XAUI</td><td>SGMII</td><td>SGMII</td><td>SGMII</td><td>SGMII</td></tr>
</table>

Table 2. SERDES Configuration Options (continued)

Bank 1				Bank 2				Bank 3			
PCIe		SRIO	Debug	XAUI				XAUI			
PCIe	PCIe	SRIO	Debug	SGMII	SGMII	SGMII	SGMII	SGMII	SGMII	SGMII	SGMII
PCIe	PCIe	SRIO	Debug	XAUI				SGMII	SGMII	SGMII	SGMII
PCIe	PCIe	SRIO	Debug	XAUI				XAUI			

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor China Ltd.
Exchange Building 23F
No. 118 Jianguo Road
Chaoyang District
Beijing 100022
China
+86 10 5879 8000
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150

Document Number: P4080PB
Rev. 1
09/2008

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

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor 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 Semiconductor 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 Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.