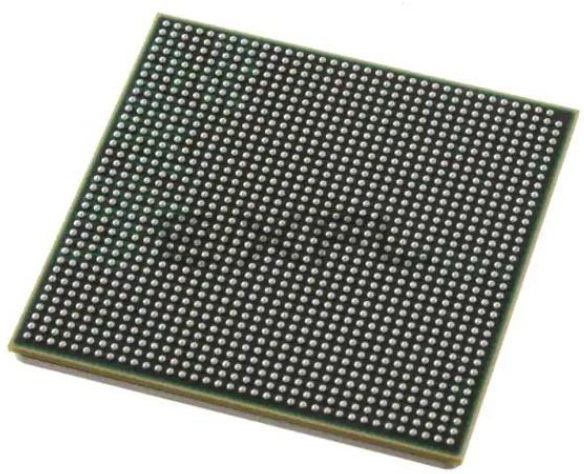

Welcome to E-XFL.COM

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	Obsolete
Core Processor	PowerPC e5500
Number of Cores/Bus Width	2 Core, 64-Bit
Speed	2.0GHz
Co-Processors/DSP	Security; SEC 4.2
RAM Controllers	DDR3, DDR3L
Graphics Acceleration	No
Display & Interface Controllers	-
Ethernet	1Gbps (5), 10Gbps (1)
SATA	SATA 3Gbps (2)
USB	USB 2.0 + PHY (2)
Voltage - I/O	-
Operating Temperature	-40°C ~ 105°C (TA)
Security Features	Boot Security, Cryptography, Secure Fusebox, Secure JTAG, Secure Memory, Tamper Detection
Package / Case	1295-BBGA, FCBGA
Supplier Device Package	1295-FCPBGA (37.5x37.5)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/p5020nxe1qmb

1 P5020 Application Use Cases

1.1 Router Control Processor

The following figure shows the P5020 in a linecard control plane application, where the linecard is part of a high-end network router.

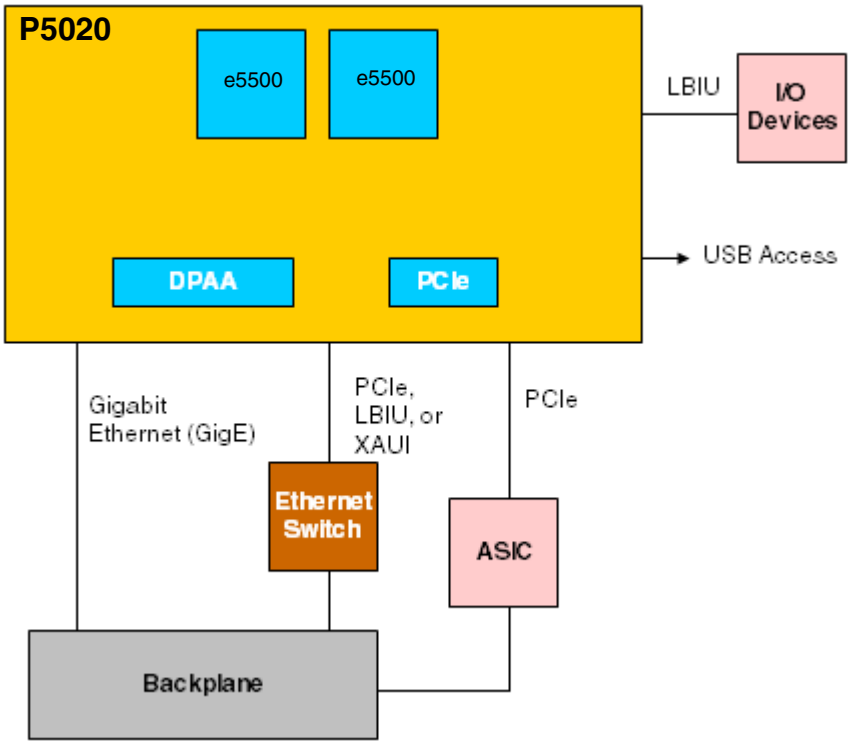

Figure 1. Control Plane Processor for a Router

1.2 DSP Farm Control Processor

The following figure shows a DSP farm enabled by the P5020 utilizing serial RapidIO.

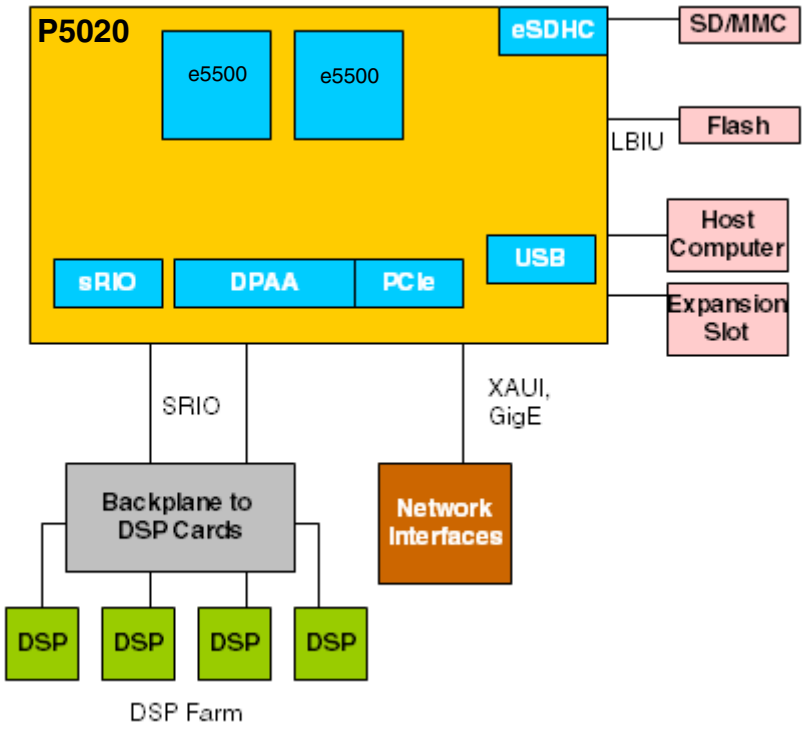

Figure 2. Control Plane Processor for a DSP Farm

1.3 SAN RAID 6 Controller

The following figure shows a RAID-enabled Disk Array Controller in an redundant active-active system for block-oriented storage systems. The P5020 Data Path Acceleration Architecture (DPAA) accelerates RAID 5/6 calculations and low-overhead data movement while optionally supporting data-at rest encryption and Data Integrity Field support.

nearly instantaneously. This allows lightly utilized CPUs to be slowed (under software control) for power savings, rather than performing more complex task migration operations.

3 Features

3.1 Block Diagram

The following figure shows the major functional units within the P5020.

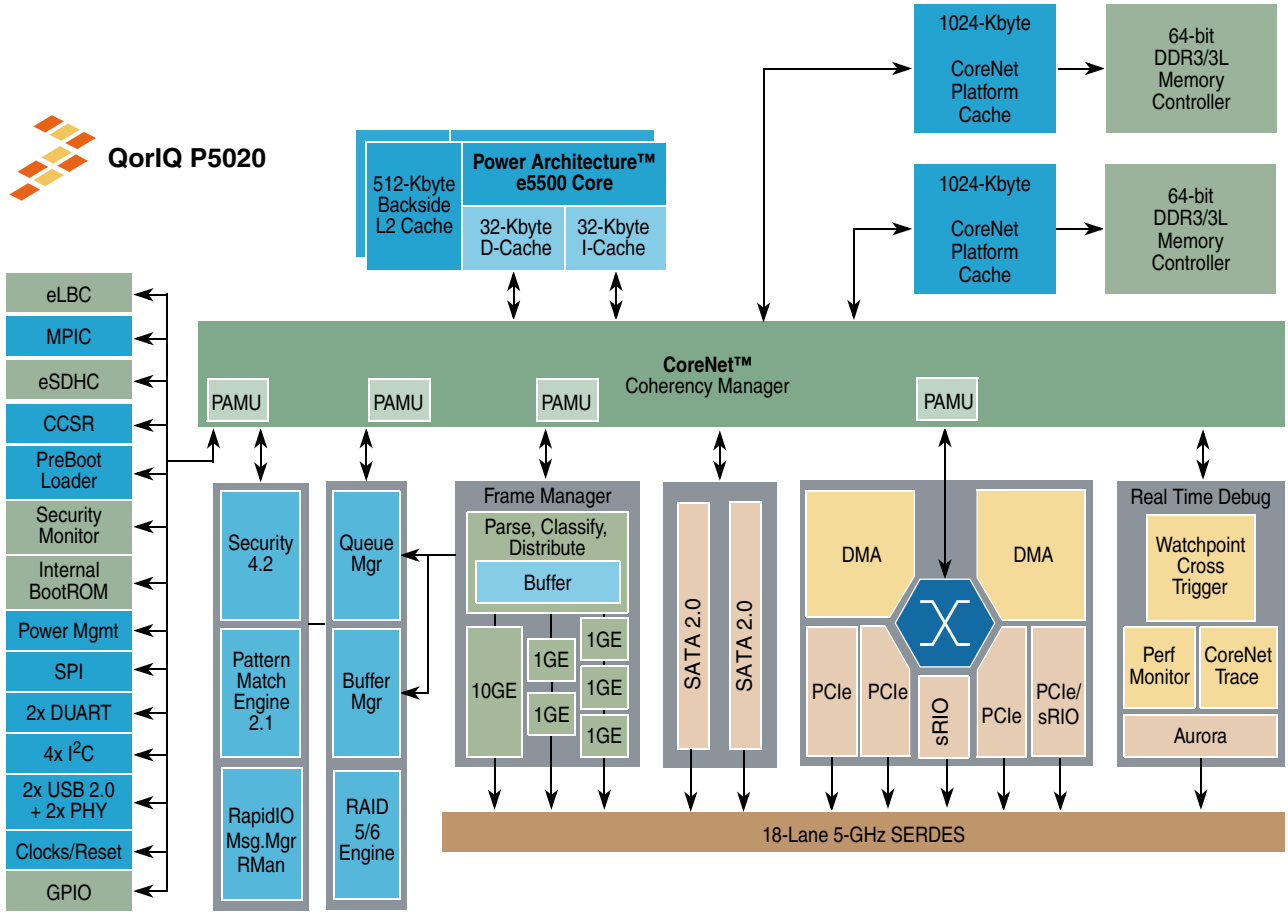

Figure 4. P5020 Preliminary Block Diagram

3.2 P5020 Features Summary

The P5020 SoC includes the following functions and features:

- Two e5500 cores built on Power Architecture technology, each with a private 512-Kbyte private backside cache
  - Up to 2 GHz
  - Three levels of instructions:
    - User

- Two 4-channel DMA engines

3.3 P5020 Benefits

The P5020's e5500 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 e5500 core, is a defining characteristic of the device. 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.

3.4 Data Path Acceleration Architecture (DPAA) Benefits

While the two Power Architecture cores offer a major leap in available processor performance in many throughput-intensive, packet-processing networking applications, raw processing power is not enough to achieve multi-Gbps data rates. To address this, the P5020 uses Freescale's Data Path Acceleration Architecture (DPAA) (see Section 3.9, "Data Path 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 P5020 allows the user's software to perform complex packet processing at high data rates.

3.5 Critical Performance Parameters

The following table lists key performance indicators that define a set of values used to measure P5020 operation.

Table 1. P5020 Critical Performance Parameters

Indicator	Values(s)
Top speed bin core frequency	2.0 GHz
Maximum memory data rate	1.3 GHz (DDR3/3L)[1] • 1.5-V for DDR3 • 1.35-V for DDR3L
Local bus	• 3.3 V • 2.5 V • 1.8 V
Operating junction temperature range	0–105 C
Package	1295-pin FC-PBGA (flip-chip plastic ball grid array)

Notes:
1 Conforms to JEDEC standard

3.6 e5500 Core and Cache Memory Complex

Each e5500 is a superscalar dual issue processor, supporting out-of-order execution and in-order completion, which allows the Power Architecture e5500 to perform more instructions per clock than other RISC and CISC architectures.

3.6.1 e5500 Core Features

- Up to 2.0 GHz core clock speed
- 36 bit physical addressing
- 64 TLB SuperPages
- 512-entry, 4-Kbyte pages front end
- 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 data path acceleration architecture (DPAA) data and context “stashing” into the L1 data cache and the backside L2 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
    - Designed to comply with IEEE Std. 754™1985 FPU for both single- and double-precision operations
  - “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

3.6.2 512-Kbyte Private Backside Cache

- Each e5500 core features a 512-Kbyte private backside L2 cache running at the same frequency of CPU. The caches support: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, and so on.
- Supports direct stashing of datapath architecture data into cache

3.6.3 CoreNet Platform Cache (CPC)

The QorIQ P5020 also contains 2x1-Mbyte of shared CoreNet platform cache, with the following features:

- Configurable as write back or write through
- Pseudo LRU replacement algorithm
- ECC protection
- 64-byte coherency granule
- Two cache line read 1024 bits per cycle at 800 MHz, 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)

3.6.4 CoreNet Fabric and Address Map

The CoreNet fabric is Freescale's next generation Interconnect Standard for multicore products, and provides the following:

- A highly concurrent, fully cache coherent, multi-ported fabric
- 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
- The CoreNet fabric has been designed to overcome bottlenecks associated with shared bus architectures, particularly address issue and data bandwidth limitations. The P5020's multiple, parallel address paths allow for high address bandwidth, which is a key performance indicator for large coherent multicore processors
- Eliminates address retries, triggered by CPUs being unable to snoop within the narrow snooping window of a shared bus. This results in the device having lower average memory latency

The 36-bit, physical address map consists of local space and external address space. For the local address map, 32 local access windows (LAWs) define mapping within the local 36-bit (64-Gbyte) address space. Inbound and outbound translation windows can map the device 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).

3.6.5 Memory Complex

The P5020 memory complex consists of the two DDR controllers for main memory, and the memory controllers associated with the enhanced local bus controller (eLBC).

3.6.5.1 DDR Memory Controllers

The two DDR memory controllers have the following functionalities:

- Supports DDR3/3L SDRAM. The P5020 also supports chip-select interleaving within a controller. The memory interface controls main memory accesses and together the two controllers support a maximum of 64 Gbytes of main memory.
- Supports interleaving across controllers on bank, page, or cache line boundaries.
- The P5020 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 P5020 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.
- Supports initialization bypass feature for use by system designers to prevent re-initialization of main memory during system power-on after an abnormal shutdown.
- Supports active zeroization of system memory upon detection of a user-defined security violation.

3.6.6 PreBoot Loader (PBL) and Nonvolatile Memory Interfaces

The PreBoot Loader (PBL) is a new logic module that operates similarly to an I^2C boot sequencer but on behalf of a larger number of interfaces.

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 (I^2C, eLBC, SPI, and SD/MMC) into fully initialized DDR or the 2-Mbyte CPC.
- Releases CPU 0 from reset, allowing the boot processes to begin from fast system memory.

- 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

3.8.2 Serial RapidIO

The Serial RapidIO interface is based on the *RapidIO Interconnect Specification, Revision 1.3,* with features from 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.

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

- Support for *RapidIO Interconnect Specification, Revision 1.3* (all transaction flows and priorities)
- 1x, 2x, and 4x LP-serial link interfaces, with transmission rates of 2.5, 3.125, or 5.0 Gbaud (data rates of 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

3.8.2.1 RapidIO Message Manager (RMan)

The key features of the RapidIO message manager (RMan) include the following:

- Manages two inbox/outbox mailboxes (queues) for data and one doorbell message structure
- Can multi-cast a single-segment 256-byte message to up to 32 different destination DevIDs
- Has four outbound segmentation units supporting RapidIO Type 5–6 and Type 8–11

3.8.3 Serial ATA (SATA) 2.0 Controllers

The key features of each of the two SATA include the following:

- Designed to comply with Serial ATA 2.6 Specification
- Supports host SATA I per spec Rev 1.0a
  - OOB
  - Port multipliers
  - ATAPI 6+

  - Spread spectrum clocking on receive
- Support for SATA II extensions
  - Asynchronous notification
  - Hot plug including asynchronous signal recovery
  - Link power management
  - Native command queuing
  - Staggered spin-up and port multiplier support
- Support for SATA I and II data rates (1.5 and 3.0 Gbaud)
- Standard ATA master-only emulation
- Includes ATA shadow registers
- Implements SATA superset registers (SError, SControl, SStatus)
- Interrupt driven
- Power management support
- Error handling and diagnostic features
  - Far end/near end loopback
  - Failed CRC error reporting
  - Increased ALIGN insertion rates
  - Scrambling and CONT override

3.9 Data Path Acceleration Architecture (DPAA)

The DPAA 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 (QMan) driver. Beyond enabling multicore resource sharing, 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 are as follows:

- Traditional routing and bridging
- Firewall
- VPN termination for both IPsec and SSL VPNs
- Intrusion detection/prevention (IDS/IPS)
- 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:

- Packet Distribution and Queue/Congestion Management
- Accelerating Content Processing

3.9.1 Packet Distribution and Queue/Congestion Management

The following table lists some packet distribution and queue/congestion management offload functions.

Table 2. Offload Functions

Function Type	Definition
Data buffer management	Supports allocation and deallocation of buffers belonging to pools originally created by software with configurable depletion thresholds. Implemented in a module 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 module called the Queue Manager (QMan). The QMan, 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 the Frame Manager (FMan).
Policing	Supports in-line rate-limiting by means of two-rate, three-color marking (RFC 2698). Up to 256 policing profiles are supported. This function is also implemented in the FMan.

3.9.2 Accelerating Content Processing

Properly implemented acceleration logic can provide significant performance advantages over 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 in order to move data to the accelerators, these engines include well-pipelined DMAs. The following table lists some specific content-processing accelerators on the P5020.

Table 3. Content-Processing Accelerators

Interface	Definition
SEC 4.2	Crypto-acceleration for protocols such as IPsec, SSL, and 802.16
PME 2.1	Regex style pattern matching for unanchored searches, including cross-packet stateful patterns

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

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

3.9.4.1.2 FMan 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.

There are several types of parser headers, shown in the following table.

Table 5. 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. 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 FMan.
Proprietary	Can be defined as being self-describing or non-self-describing

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.

3.9.4.1.3 FMan Distribution and Policing

After parsing is complete, there are two options for treatment (see Table 6).

Table 6. Post-Parsing Treatment Options

Treatment	Function	Benefits
Hash	• Hashes selected fields in the frame as part of a spreading mechanism • The result 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).	Useful when spreading traffic while obeying QoS constraints is required
Classification look-up	• Looks up certain fields in the frame to determine subsequent action to take, including policing • The FMan contains internal memory that holds small tables for this purpose. • 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 look-up can provide key information for a subsequent look-up. After all the look-ups are complete, the final classification result provides either a hash key to use for spreading, or a FQ ID directly.	• Useful when hash distribution is insufficient and a more detailed examination of the frame is required • Can determine whether policing is required and the policing context to use

Key benefits of the FMan policing function are as follows:

3.9.4.4 Security Engine (SEC 4.2)

The SEC 4.2 is QorIQ's fourth generation crypto-acceleration engine. In addition to off-loading cryptographic algorithms, the SEC 4.2 offers header and trailer processing for several established security protocols. The SEC 4.2 includes several 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 Std 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 4.2, 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 3 G 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 4.2's aggregate symmetric encryption/integrity performance is 5 Gbps, while asymmetric encryption (RSA public key) performance is ~5,000 1024b RSA operations per second.

The SEC 4.2 is also part of the QorIQ Trust Architecture, which gives the P5020 the ability to perform secure boot, runtime code integrity protection, and session key protection. The Trust Architecture is described in Section 3.10, "Avoiding Resource Contentions Using the QorIQ Trust Architecture."

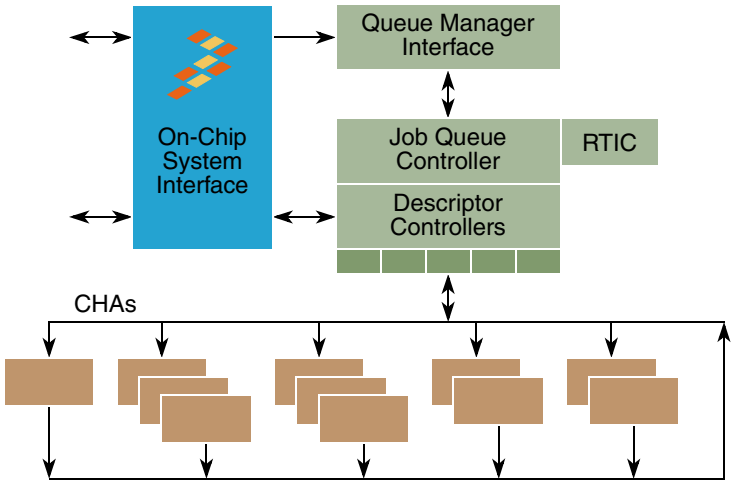

Figure 6. SEC 4.2 Block Diagram

3.9.4.5 Pattern Matching Engine (PME 2.1)

The PME is a self-contained hardware module capable of autonomously scanning data from streams for patterns that match a specification in a database dedicated to it. The PME 2.1 is an updated version of the PME used in previous members of the PowerQUICC family. Specific updates include the following:

- QMan interface supporting the DPAA Queue Interface Driver
- 2x increase in the number of patterns supported (16 Kbytes to 32 Kbytes)
- Increase in number of stateful rules supported (8 Kbytes to 16 Kbytes)
- Raw scanning performance is ~ 5 Gbps.

Patterns that can be recognized, or "matched," by the PME are of two general forms:

- 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.

3.9.4.5.1 PME Regular Expressions (Regex)

The PME specifies patterns of bytes as regular expressions (regex). The P5020 (by means of an online or offline process) converts Regex patterns into the PME's pattern specification database. Generally, there is a one-to-one mapping between a regex and a PME byte pattern. The PME's use of regex 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 32,000 regex patterns are supported, each up to 128 bytes long. The 32,000 regex patterns can be combined by means of stateful rules to detect a far larger set of event patterns. Comparative compilations against DFA style regex engines have shown that 300,000 DFA pattern equivalents can be achieved with ~8000 PME regexes with stateful rules.

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 should not 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 core MMU also includes embedded Hypervisor extensions.

Each core MMU supports three levels of instructions:

- User
- Supervisor (OS)
- 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 P5020 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.

3.10.3 Peripheral Access Management Unit (PAMU)

The P5020 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 (PME, SEC, FMan, and so on). The PAMU access control can be one of the following:

- Absolute—The FMan, PME, SEC, and other bus masters can never access memory range XYZ.
- Conditional—Based on the Partition ID of the CPU that programmed the bus master

Being MMU-based, the embedded Hypervisor is only able to stop unauthorized software access attempts. Internal components with bus mastering capability also need to be prevented from reading and writing to specific memory regions. These devices do not 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.

3.10.4 Secure Boot and Sensitive Data Protection

The core MMUs and PAMU allow the device 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 P5020 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.

3.10.4.1 Secure Boot Option

The system developer digitally signs the code to be executed by the CPU coming out of reset, and the device ensures that only an unaltered version of that code runs on the platform. The P5020 offers both boot time and run time code authenticity checking and configurable consequences when the authenticity check fails.

3.10.4.2 Sensitive Data Protection Option

The P5020 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 device 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 4.2 for decryption of session traffic.

3.11 Advanced Power Management

The P5020's advanced power management capabilities are based around fine-grained static clock control and software-controlled dynamic frequency management.

3.11.1 Saving Power by Managing Internal Clocks

Dynamic voltage and frequency scaling (DVFS) are useful techniques for reducing typical/average power and maximizing battery life in laptop environments, but embedded applications must be designed for rapid response to bursts of traffic and max power under worst-case environmental conditions. While the P5020 does not implement DVFS in the PC sense, it 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.

The P5020 also supports (under software control) dynamic changes to CPU operating frequencies and voltages. Each CPU sources its input clock from one of two independent PLLs inside the device. 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.

3.11.2 Turning Off Unneeded Clocks

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 will have 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).

3.11.3 Avoiding Full System Failure Due to Thermal Overload

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 or system ambient temperature of the device achieves some critical level, external temperature detection circuitry can drive a high-priority interrupt into the P5020, 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 G Hz 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 (such as Core Doze, Core Nap, and Core Sleep) are also available in the core.

3.12 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 P5020 as they would with multiple discrete Freescale communications processors, Freescale developed the debug architecture shown in the following figure.

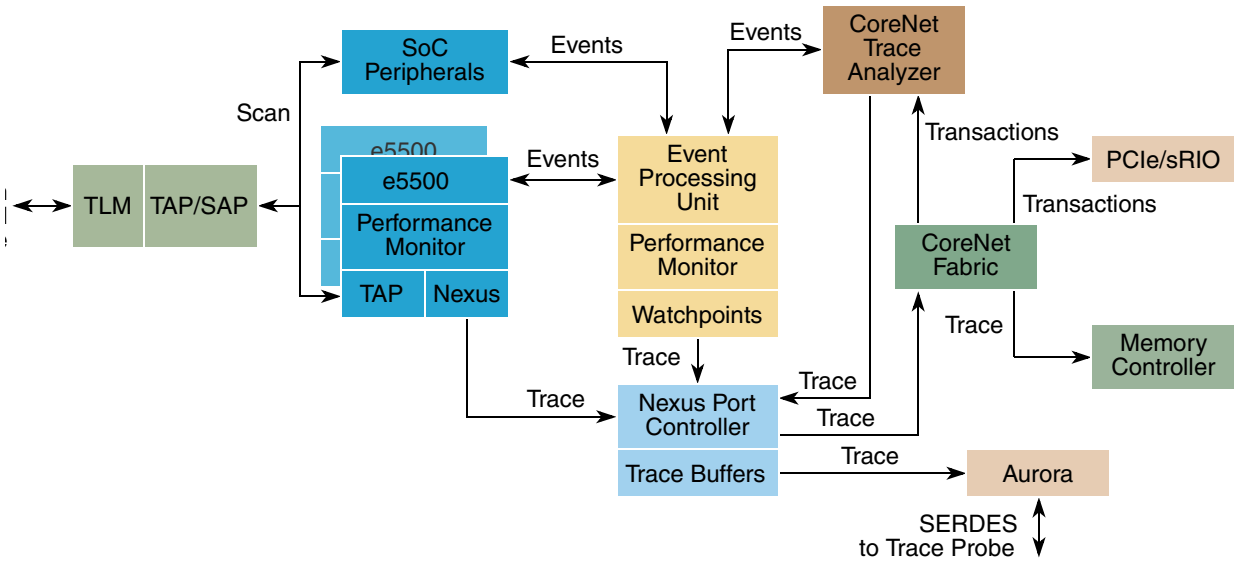

Figure 8. Debug Architecture

Debug features include the following:

- Debug and performance monitoring registers in both the core 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 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)

4 Developer Environment

Software developers creating solutions with the Power Architecture technology 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 ensure that this remains the case for multicore, Power Architecture-based products, including the P5020.

The various levels of the developer environment are shown in Figure 9, with the more broadly used tools and boards at the base of the pyramid, and increasingly application-specific enablement items at the top. Each level is described further, as follows:

- Section 4.1, "Base of the Pyramid: Broadly-Used Tools and Boards
- Section 4.2, "First Level of the Pyramid: Debug and Performance Analysis
- Section 4.3, "Second Level of the Pyramid: Simulation, Hypervisor, and DPAA Reference "Stacklets"
- Section 4.4, "Top Level of the Pyramid: Application-Specific Enablement

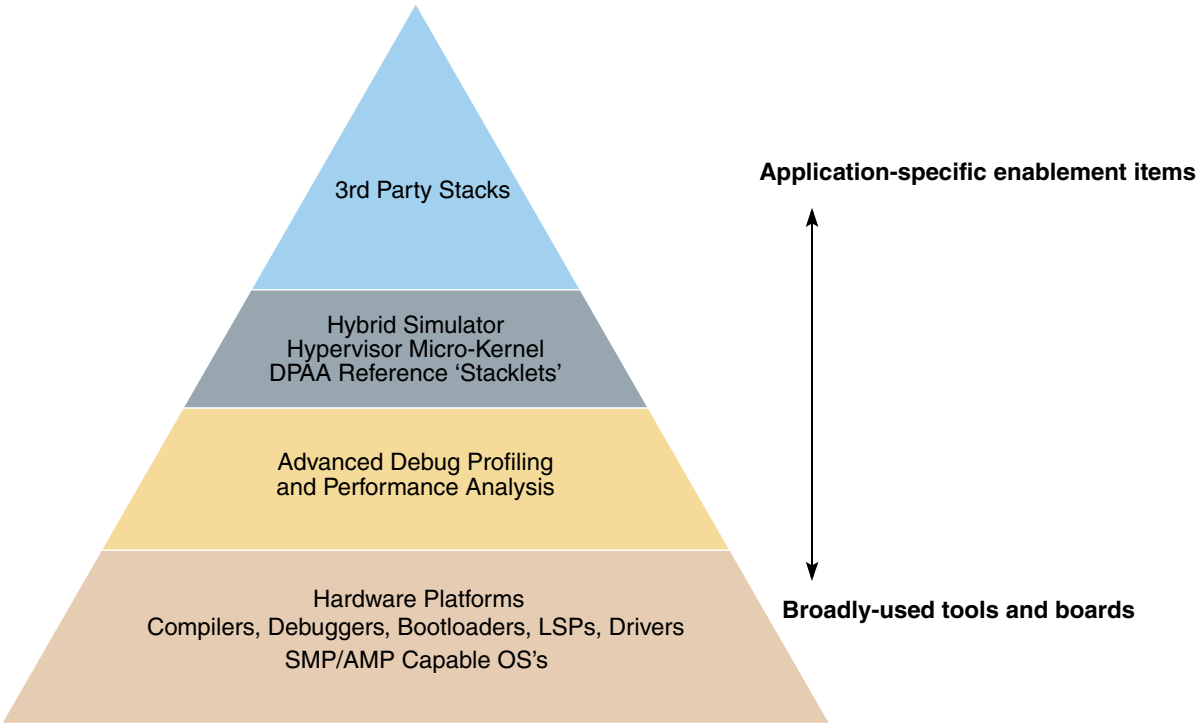

Figure 9. Levels of Developer Environment

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

4.3.2 Hypervisor Micro-Kernel

The P5020's e5500 cores offer a new embedded Hypervisor capability to address the need for a single operating system performing coordination and access control functions, managing shared resources in an efficient manner. The embedded Hypervisor 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.

4.3.3 DPAA Reference "Stacklets"

It is expected that some CPUs will be dedicated as datapath processors, working closely with the DPAA. Freescale will provide reference protocol "stacklets," optimizing performance critical regions of protocol processing and their interaction with the DPAA hardware.

4.4 Top Level of the Pyramid: Application-Specific Enablement

This category includes 3rd-party stacks optimized for DPAA, RegEx, AV TCP, IPv4/6, IPsec/SSL.

Many of the expected applications for the P5020 involve network protocol processing. 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.

5 Document Revision History

The following table provides a revision history for this product brief.

Table 8. Revision History

Revision	Date	Substantive Change(s)
1	02/2013	Modified USB Specification, Section 3.7, "Universal Serial Bus (USB) 2.0."
0	12/2011	Initial public release.

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: P5020PB
Rev. 1
02/2013

freescale™