

## Application Note

AN2662  
Rev. 0, 05/2004

Migrating from  
PowerQUICC II to  
PowerQUICC III



Nigel Dick  
NCSD Applications  
East Kilbrade,  
Scotland

This application note outlines the considerations that are relevant to migrating from the MPC826x (PowerQUICC II) family of devices to the higher performance MPC85xx (PowerQUICC III) family. This document, explains the differences between the main system blocks and internal cores, explores the initialization and reset sequences, and details the minor modifications required to run existing PowerQUICC II software on the PowerQUICC III.

It is assumed that the reader has a basic understanding of the system blocks and operation of the PowerQUICC II microprocessor family and HiP7 derivatives. Therefore, it is very important that users are familiar with the following technical references in order to understand the correct terminology and general MPC85xx and e500 programming model. At a minimum, the user should refer to the *MPC8560 PowerQUICC™ III Integrated Communications Processor Reference Manual*.

The following Motorola documents are referred to throughout this application note:

- *MPC8560 PowerQUICC™ III Integrated Communications Processor Reference Manual* (MPC8560RM)
- *PowerPC™ e500 Core Complex Reference Manual* (E500CORERM)
- *MPC603e and e500 Register Model Comparison* (AN2490)

The products described in this document are PowerPC™ microprocessor cores. This application note outlines differences between the register models defined by the Apple/IBM/Motorola (AIM) and Book E versions of the PowerPC architecture and the registers implemented in the MPC603e processor and the e500 microprocessor core.

The following topics are addressed:

Topic	Page
Section 1, "System Overview"	2
Section 2, "Core Differences"	3
Section 3, "Communications Processor Module (CPM)"	12
Section 4, "New Features on PowerQUICC III"	15
Section 5, "Initialization, Reset and Boot Procedures"	28
Section 6, "Software Considerations"	30
Section 7, "Summary"	32

# 1 System Overview

The PowerQUICC III is the latest addition to Motorola's PowerQUICC line of integrated communications processors, offering the end-user greater flexibility, extended capabilities and high levels of integration. Figure 1 illustrates the internal blocks of the MPC8560.

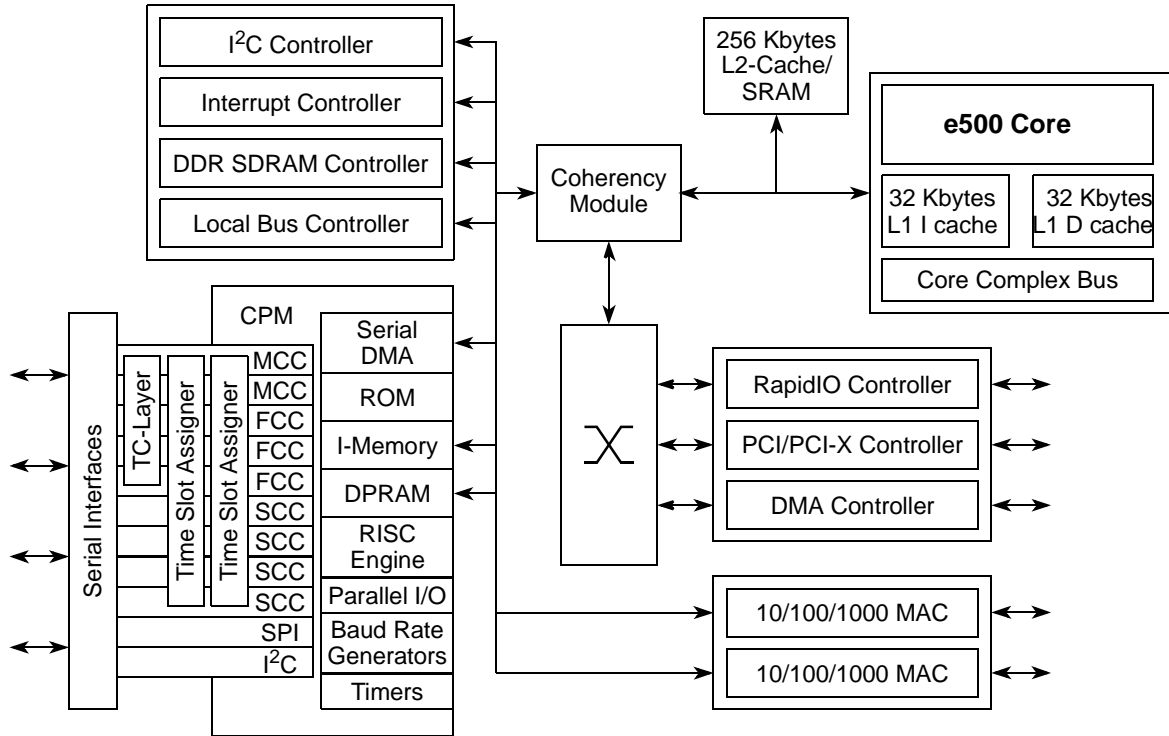


Figure 1. System Block Diagram of MPC8560

Like the other members of the PowerQUICC family, the MPC8560 is split into three main system blocks:

- Microprocessor core: e500 Book E core with 32 Kbytes of L1 instruction/data cache and 256 Kbytes of configurable L2 cache or SRAM.
- Communications Processing Module (CPM): (similar to the CPM block on the PowerQUICC II, HiP 7).
- Peripheral Modules: used to provide a number primary and auxiliary functions such as DDR SDRAM, RapidIO, PCI, PCI-X, three-speed Ethernet controllers. These are covered in detail in Section 4.6, "Three-Speed Ethernet Controller."

The MPC8540, another member of the PowerQUICC III family, is identical to the system block diagram of the MPC8560 except that the CPM block has been replaced by a dual UART and an additional 10/100 Ethernet MAC. As a feature subset of the MPC8560, the MPC8540 uses the same package as the MPC8560 (783PBGA) and is footprint compatible. The redundant CPM pins on the MPC8540 package are replaced by no connects.

The MPC8555 and MPC8541 devices expand the PowerQUICC III roadmap with the inclusion of a 32-bit PCI controller, a lite version of the MPC8560 CPM, and an integrated security core. Higher performance HiP8 derivatives are planned for release by Freescale in 2005. However, this document refers only to the performance and features of HiP7 PowerQUICC III designs.

## 2 Core Differences

The MPC8540 and MPC8560 devices implement the e500 core, which is based on a new embedded version of the PowerPC architecture known as Book E. The following section describes the differences between the e500 core and the 603e core used in PowerQUICC II devices.

### 2.1 Introduction

The e500 processor core is a low-power implementation of the family of reduced instruction set computing (RISC) embedded processors that implement the Book E definition of the PowerPC architecture. The e500 implements the 32-bit portion of the Book E architecture using the lower words in the 64-bit general-purpose registers (GPRs).

Book E allows processors to provide auxiliary processing units (APUs), which are extensions to the architecture. The most significant of these on the e500 is the signal processing engine APU (SPE APU), which includes a suite of vector instructions that use the upper and lower halves of the GPRs as a single two-element operand. Most APUs implemented on the e500 are defined by the Motorola Book E implementation standards (EIS).

Like the 603e, the e500 provides 32-bit effective addresses, integer data types of 8, 16, and 32 bits, and a 32-bit floating-point data type. The SPE APU on the e500 also supports signed and unsigned fractional data types of 32 bits. The core complex is a superscalar processor that can issue two instructions and complete two instructions per clock cycle. A block diagram of the core complex is shown in Figure 2. See Section 2.3.1, “Instruction Set Differences” and Section 5.3, “Boot Sequencer,” for more information on e500 APUs.

### 2.2 Key Features

The key features of the e500 core complex are detailed in the *PowerPC™ e500 Core Complex Reference Manual* and are not discussed in this application note.

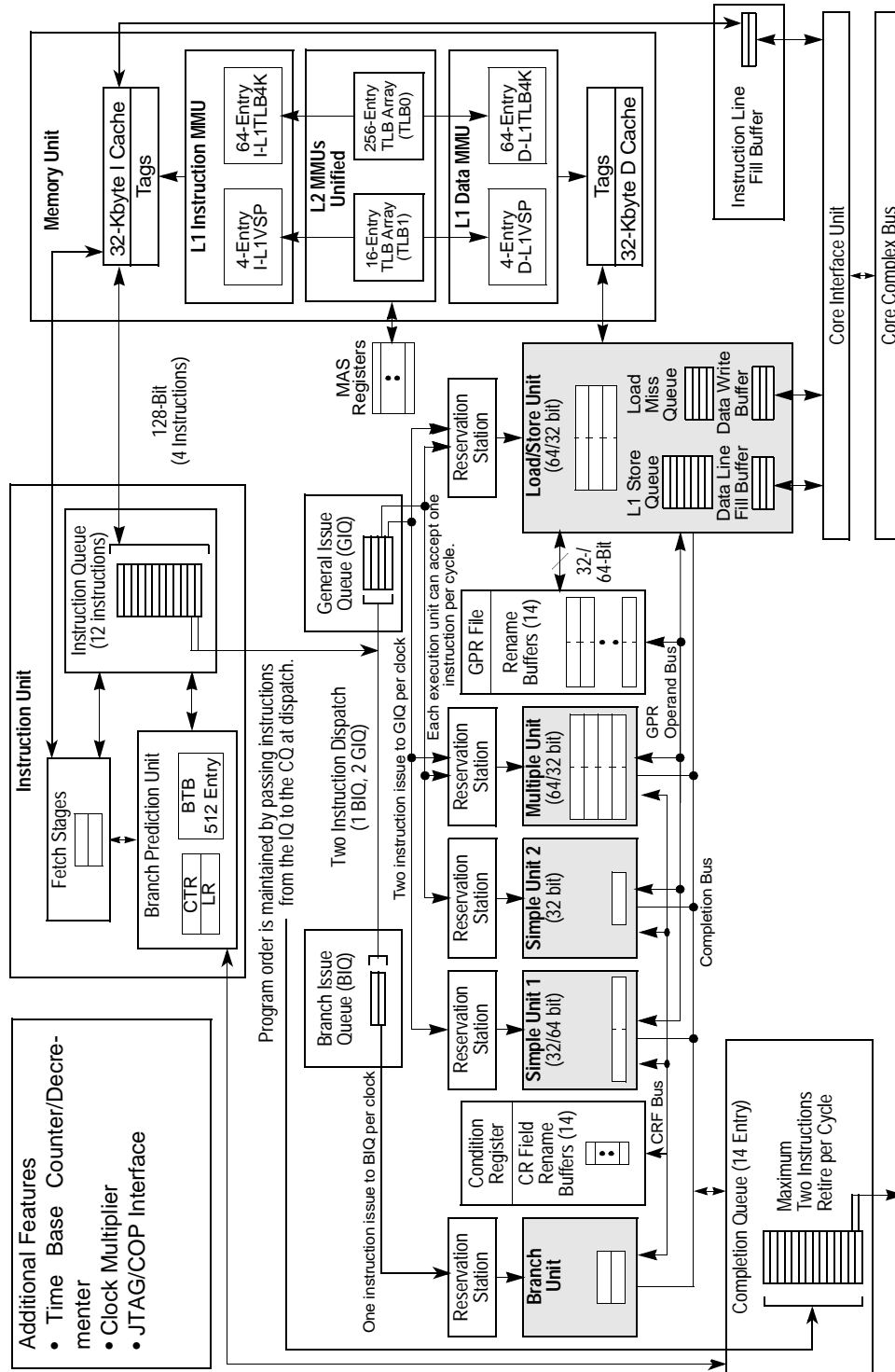


Figure 2. e500 Core Complex Block Diagram'

## 2.3 e500 Core Difference Comparison

This section contains a summary of the Book E architecture and its compatibility with the original version of the PowerPC architecture as it was defined by Apple, IBM, and Motorola and implemented in the 603e core (referred to as the AIM version of the PowerPC architecture).

The e500 core is Motorola's 32-bit implementation of the Book E architecture.

### 2.3.1 Instruction Set Differences

In user-mode the e500 core executes legacy user-mode binaries and object files with the following exceptions:

- The e500 core uses GPRs for floating point operations instead of separate floating point registers (FPRs) used in the AIM architecture. The floating point instructions on the e500 only support single precision operations and have different encoding from those on AIM processors. These changes are generally handled by recompiling the source code with an e500 compiler.
- The e500 architecture is a 32-bit implementation and 64-bit operation instructions are not supported. Any 64-bit instructions force an illegal instruction exception.
- External control input/output word index (**eciwx** and **ecowx**) instructions are not supported.
- No string instructions are implemented in the e500 architecture. To support legacy binaries, it is necessary to add some trap emulation code for the missing instructions.

The supervisor mode instruction set defined by the AIM version of the PowerPC architecture is compatible with the e500 with the following exceptions:

- The MMU architecture is different, and some TLB manipulation instructions have different semantics—**tlbsync**, **tlbivax**, **tlbre**, **tlbwe**.
- Instructions that support BATs and segment registers are not implemented due to the difference in the MMU architecture of the e500. See Section 2.3.5, “Memory Management,” for more information.
- New instructions, such as **rftci** (Return from Critical Interrupt), **bbelr/bblels** (Branch Buffer Entry Lock Reset/Branch Buffer Load Entry and Lock Set), **wrttee[i]** (Write MSR External Enable)

Figure 3 shows the e500 user instruction set to be a mnemonics instruction subset of the AIM instruction set with some additional instructions to support the new APU features of the e500 architecture.

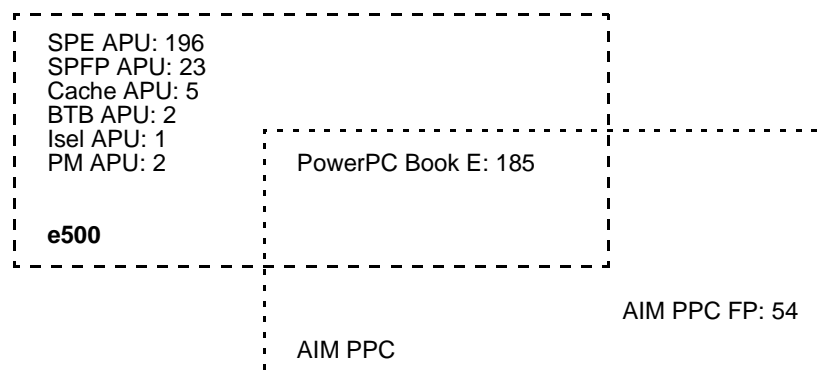


Figure 3. PowerPC User Instructions

## Core Differences

The e500 core implements new instructions related to the signal processing engine (SPE) APU and to support single-precision floating-point operations (SPFP APU).

### NOTE

The SPE APU is supported only through the use of the libmoto library. Section 5.3, “Boot Sequencer,” contains more information on the SPE and the libmoto library.

Another, new APU is **isel** (Integer Select). This instruction performs a conditional register move operation which helps to reduce the size of the code footprint and can avoid having to predict the outcome of a condition and the use of conditional branches. It is also useful for conditional clearing of a register. In the Book E architecture the **isel** is treated as an APU.

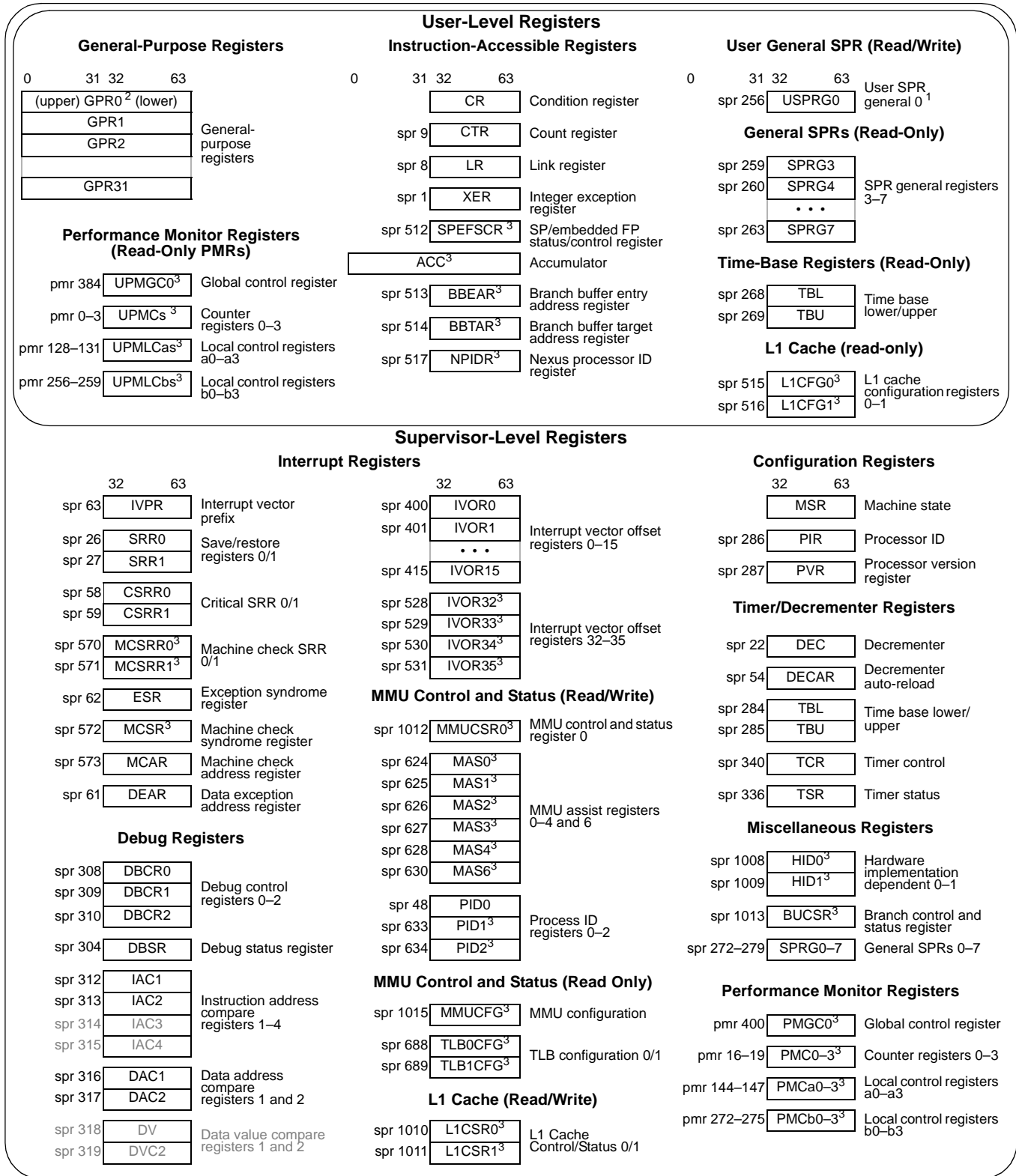
Another useful new supervisor instruction is **wrtee[i]**. This instruction manipulates the EE bit of the MSR register and replaces several instructions required by the AIM architecture to perform the same task. The advantage here is that **wrtee[i]** is used to update MSR[EE] without affecting any other MSR bits that may have changed.

for example:

<u>AIM architecture</u>		<u>e500 architecture</u>
mfmsr rn	; get MSR	mfmsr rn ; get MSR
mr ra,rn	;take a copy	wrteei 0 ;clear interrupt
andi ra,ra,0x7FFF	;clear EE bit	:
mtmsr ra	;clear interrupt	:
:		wrtee rn ;restore MSR
:		
mtsr rn	; restore MSR	

## 2.3.2 Programming Model

The programming model of the e500 core is shown in Figure 4.



<sup>1</sup> USPRG0 is a separate physical register from SPRG0.

<sup>2</sup> The 64-bit GPR registers are accessed by the SPE as separate 32-bit registers by SPE instructions. Only SPE vector instructions can access the upper word.

<sup>3</sup> These registers are e500-specific and are not part of the Book E architecture.

Figure 4. e500 Core Programming Model

## NOTE

Differences between the e500 and the 603e core register sets are described in the *MPC603e and e500 Register Model Comparison* (AN2490).

Unlike the AIM version of the PowerPC architecture, where little-endian mode is controlled on a system basis, Book E allows control of endian mode on a memory page basis. The e500 core also affects true little-endian mode by byte swapping.

### 2.3.3 Internal Memory Subsystem

Both Book E and AIM versions of the PowerPC architecture provide separate instruction and data memory resources. In addition to the internal 32-Kbyte L1 instruction and 32-Kbyte L1 data cache on the e500 core, the PowerQUICC III has 256 Kbytes of L2 cache, which is often referred to as the memory complex. This structure is configurable and its function can be switched between SRAM or cache, or a combination of both SRAM and cache.

Configured as memory-mapped SRAM, the memory complex can either be split into two separate blocks of 128 Kbytes each, or one large 256 Kbyte SRAM block. In full cache mode, all 256 Kbytes of the memory complex are used as L2 cache. Finally, in half-SRAM/half-cache mode, 128 Kbytes of memory-mapped SRAM and 128 Kbytes of cache are created.

The L2 cache is an 8-way, set-associative, write-through, front-side cache that supports locking on either a line-per-line basis or on the complete block, unlike the PowerQUICC II, where only the entire cache could be locked or up to a maximum of three ways. Locking cache blocks allows code in specific areas to be locked so that it is not modified or updated. For example, this feature can be used to store a non-changing, but often used subroutine, which can be stored in cache and then locked to enable optimal code efficiency. Both e500 and 603e cores use the same cache control instructions so that data cache coherency instructions can be used on the PowerQUICC III. The L2 cache on the PowerQUICC III also has hit under miss capability, parity checking and generation and a feature called cache stashing.

Stashing allows external masters that drive transactions into the chip to force certain data into the L2 cache to increase performance. For example, an Ethernet routing application where the processor is simply receiving an IP header, interrogating the recipient address and forwarding the packet back out, without looking at the actual payload. Stashing, for example, allows the Gigabit Ethernet controller on the PowerQUICC III to stash a copy of the header into the L2 cache, so that when the cache gets a hit, it retrieves this information rather than wasting valuable cycles retrieving the information from DDR SDRAM.

### 2.3.4 Interrupt Handling

Interrupt handling is generally the same as defined in the AIM version of the PowerPC architecture for the e500, with the following differences:

- Book E processors use the IVPR and IVORs to set exception vectors individually. They can also be set to the address offsets defined in the OEA to provide compatibility with AIM processors. See Section 6.1, “Exception Handling,” for additional information).
- Book E defines a new critical interrupt, that provides an extra level of interrupt nesting. The critical interrupt includes critical and watchdog timer time-out inputs.
- The e500 implements the machine check exception differently from the Book E and from the AIM definition. It defines a Return from Machine Check Interrupt instruction, **rfmci**, and two machine check save/restore registers, MCSRR0 and MCSRR1.



### 2.3.5 Memory Management

The e500 core implements a straightforward virtual address space that complies with the Book E MMU definition, eliminating segment registers and block address translation resources. Book E defines resources for fixed 4-Kbyte pages and multiple, variable page sizes that can be configured in a single implementation. TLB management is provided with new instructions and SPRs.

The e500 core contains a two-level MMU structure as shown in Figure 5.

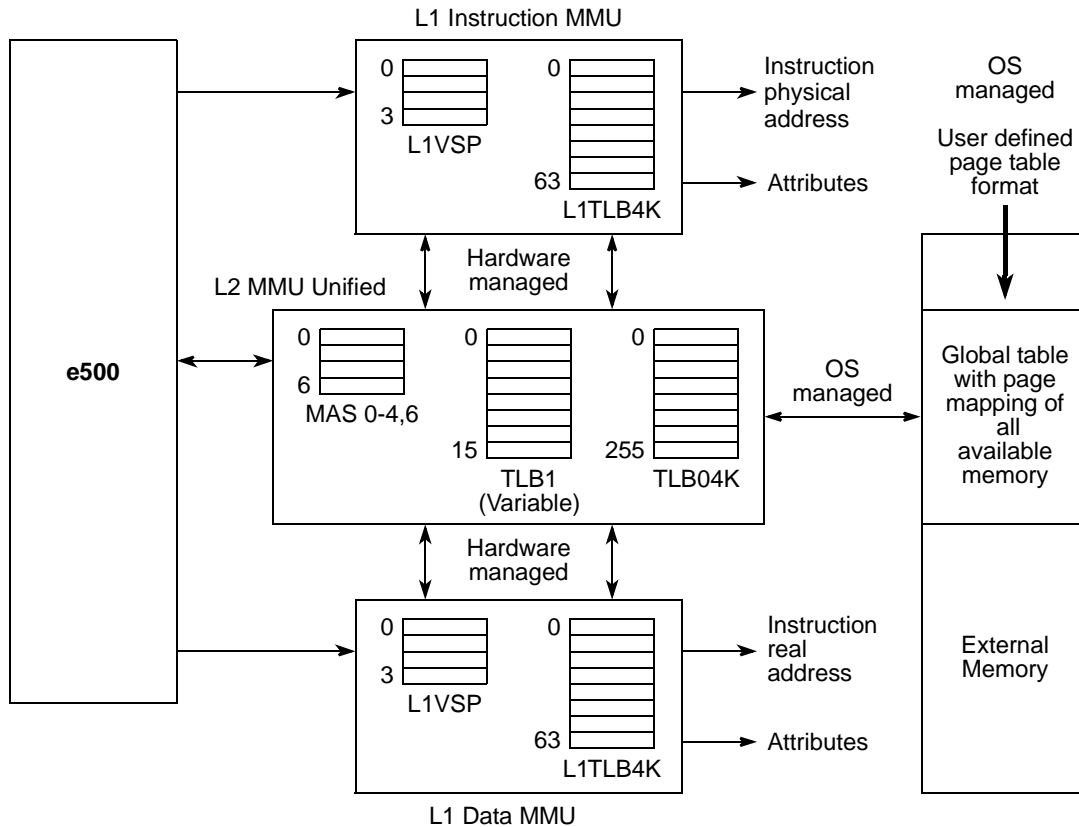


Figure 5. e500 Core MMU Structure

The L1 MMU is maintained automatically by the hardware and is transparent to the software. It consists of two four-entry fully associative TLB arrays. One array for instruction and another for data. These arrays support nine page sizes. There are also two 64-entry, 4-way set associative TLB arrays that support fixed (4 Kbyte) page sizes. As a result, the L1 MMU is structured into two similar MMU blocks, one MMU block is used for instruction and the other MMU block is for data.

The L2 MMU is programmed by the user. It consists of one 16-entry, fully associative unified TLB array, TLB1, which supports nine page sizes. This is similar to the BAT registers in the AIM architecture. There is also a 256-entry, two-way set associative unified TLB array called TLB0 which supports fixed page sizes (4 Kbytes). This is similar to the page tables in the AIM architecture. A key difference between the MMU structures of L1 and L2 is that on the L1 MMU, two separate MMU structures are used to implement data and instruction entries. In the L2 MMU, the MMU is unified and can be shared by both instruction and data entries.

The TLB0 and TLB1 arrays are managed by software using **tlbre**, **tlbwe**, **tlbsx**, **tlbsync**, **tlbivax**, and **mtspr/mfspr** instructions along with six special registers [MAS0–MAS4, MAS6]. These MAS registers in

## Core Differences

conjunction with the TLB arrays, allow the MMU to be set up to provide address translation, access protection, and attribute generation on a page-by-page basis. The master translation table is user definable and is maintained and managed by software with no format requirements.

Unlike the AIM PowerPC architecture, the MMU in the e500 is always on and cannot be disabled. During Reset, all of the TLB entries in the L1 MMU and L2 MMU are invalidated by the hardware and entry 0 of the TLB1 array is initialized to allow supervisor only access to the last 4K page in the memory map. This is address range 0xFFFF\_F000–0xFFFF\_FFFF. This allows access to the reset vector location. The initialization code that runs after reset configures the other entries of the MMU to allow access to all other components in a system.

In the following debugger initialization example, you can see that the MAS registers are re-used to configure different peripherals on the PowerQUICC III. The following code example assumes that the debugger already knows that CCSRBAR is at the default position (0xFF70\_0000) and that the internal SRAM has been configured to reside at the highest memory position (0xFFFFE\_0000) in order to cover the default MMU page (at 0xFFFF\_F000–0xFFFF\_FFFF).

The configuration, control, and status registers on the PowerQUICC III are all memory mapped and occupy a 1 Mbyte region of memory. The starting base address of the 1 Mbyte region is programmable using the CCSR base address register (CCSRBAR). The default base address for the configuration, control, and status registers is 0xFF70\_0000 (CCSRBAR = 0x000F\_F700).

The following code sample is an example of an MMU initialization:

```
#####
# MMU initialization
# First we re-write TLB1 entry 0 using MAS0
writespr 624 0x10000000 # MAS0
# Then we write at 0xFFFFF000 (SRAM) the code to read a TLB entry
writemem.l 0xFFFFF000 0x7C000764 # tlbre
writemem.l 0xFFFFF004 0x7C0004AC # msync
writemem.l 0xFFFFF008 0x48000000 # infinite loop
# Read tlb1 entry 0, execute the code at 0xFFFFF000
writereg PC 0xfffff000
run
sleep 50
stop
# Write at 0xFFFFF000 the necessary code for setting a TLB entry
writemem.l 0xFFFFF000 0x7C0007A4 # tlbwe
writemem.l 0xFFFFF004 0x7C0004AC # msync
writemem.l 0xFFFFF008 0x48000000 # infinite loop
# Re-write tlb entry 0
writereg PC 0xfffff000
run
sleep 50
stop
# OK, TLB0 is now set-up, now we turn to define our peripherals
# define 256MB TLB1 entry 1: 0x00000000 - 0xFFFFFFFF for DDR Memory
writespr 624 0x10010000 # MAS0
writespr 625 0x80000900 # MAS1
writespr 626 0x00000000 # MAS2
writespr 627 0x0000003f # MAS3
# write tlb entry
writereg PC 0xfffff000
```

```

run
sleep 50
stop
# OK DDR is set up, now define an area for 16MB FLASH at
# 0xFF000000 - 0xFFFFFFFF. Note how we re-use the MASx registers
# again, once the previous TLB entry has been written.
writespr 624 0x10030000 # MAS0
writespr 625 0xc0000700 # MAS1
writespr 626 0xff00001A # MAS2
writespr 627 0xff000015 # MAS3
writespr 628 0x00000000 # MAS4
# write tlb entry
writereg PC 0xFFFFFFFF00
run
sleep 50
stop
# The rest of the initialization follows on from here...

```

### 2.3.6 Core Reset

Book E-compliant cores do not share a common reset vector with the AIM version of the PowerPC architecture. Due to the additional features and capabilities of the e500 core, the initialization process is different. Unlike the AIM version of the PowerPC core, once execution begins, the e500 core is in virtual mode with a hardware-initialized TLB entry.

In its default mode of operation, after Reset the e500 core always performs a fetch from address 0xFFFF\_FFFC. The instruction obtained from this address must contain a branch to an address somewhere within the last four Kbytes of the memory map. The most common address will be 0xFFFF\_F000. The initialization code that starts at this address will probably set up other entries in the MMU to enable other address ranges in the memory map. It may also increase the size of the default TLB1 entry from four Kbytes to cover the size of the boot ROM

As part of the power-on reset or hard reset process, some device functions are initialized by sampling a number of signals during  $\overline{\text{HRESET}}$ . These signals are normally pulled high or low by external resistors to select the desired function. See Section 5, “Initialization, Reset and Boot Procedures,” for more information.

#### NOTE

The PowerQUICC III has a number of boot options. These options are user-defined during this  $\overline{\text{HRESET}}$ /configuration phase and allow the device to be booted from RapidIO, PCI, I<sup>2</sup>C Boot Sequencer etc. The reset, initialization and boot operation of the PowerQUICC III is covered in more detail in Section 5, “Initialization, Reset and Boot Procedures.”

### 2.3.7 Branch Prediction

The e500 core does not implement the static branch prediction defined by the PowerPC architecture. It implements dynamic branch prediction using 512-entry, 4-way set associative branch target buffer (BTB) to maintain a history of branches that have been encountered and of how they have or have not been taken, or not taken. Entries can be locked into the BTB under software control. Dynamic branch prediction can be disabled in the branch unit control and status register (BUCSR), in which case, it predicts every branch as not taken.

## 2.4 Summary of Core Differences

Table 1 shows a summary of the differences between the e500 (HiP7) and 603e cores. It is expected that on the current HiP7 silicon, the frequency of the PowerQUICC III device is expected to reach a maximum of 1GHz over time. PowerQUICC III system frequencies in the 667–833 MHz range are currently available and are functional in system designs now. As previously mentioned, PowerQUICC III HiP8 should give a significant performance enhancement over these tabulated HiP7 figures.

**Table 1. Key Characteristic Differences between e500 and 603e Cores**

Features	MPC82xx 603e Core	MPC85xx e500 Core
Frequency (MHz):	200–300 → 450-MHz	667–833 → 1-GHz
Core architecture:	32-bit classic PowerPC	32-bit Book E PowerPC
Instruction issue:	Dual-issue	Dual-issue
Pipeline stages:	4	7
Auxiliary Processing Units (APUs):	Not defined by the classic PowerPC architecture	Signal Processing Engine (SPE), Instruction Select, Performance Monitor, Cache Block Lock/Unlock, BTB Lock, Machine Check
I-Cache/D-Cache: Associativity: Error Detection:	16 Kbyte/16 Kbyte 4-way set-associative None	32K/32K 8-way set-associative Parity generation & checking
Address Translation	Real address mode, Block address translation (BAT), and Page address translation	Page Address Translation Only (More Flexible: 4 Kbyte to 256 Mbyte Variable or 4 kbyte fixed-size pages)
Performance	570 MIPs at 300 MHz (Average 1.9 instruction/cycle)	Est. 1850 MIPs at 800 MHz (Average 2.31 instruction/cycle)

## 3 Communications Processor Module (CPM)

The CPM of the PowerQUICC III (like the CPM on the PowerQUICC II HiP 7) supports three full-duplex, fast communications controllers (FCCs), two multi-channel controllers (MCCs), four full-duplex, serial communications controllers (SCCs), one serial peripheral interface (SPI), and one I<sup>2</sup>C bus interface.

### NOTE

The clock that drives the CPM block on the PowerQUICC III is generated from the core-complex bus clock (CCB).

The following sections highlight the CPM differences between the PowerQUICC II HiP7 and the PowerQUICC III.

Because the PowerQUICC III's CPM architecture is nearly identical to the PowerQUICC II, customers can use existing PowerQUICC II microcode on the PowerQUICC III CPM. However, there are differences in the interrupt handling of the e500. These differences are described in more detail in Section 4.9, "Programmable Interrupt Controller (PIC)" and Section 6.1, "Exception Handling."

### 3.1 CPM Performance & Internal Memory

To provide more available CPM bandwidth on the PowerQUICC III, the CPM operates at a maximum speed of 333 MHz as opposed to 300 MHz on the PowerQUICC II (HiP7). This gives PowerQUICC III applications a notable performance improvement over similar PowerQUICC II (HiP7) designs.

On both the PowerQUICC II (HiP7) and PowerQUICC III, 64 Kbytes of SRAM are available for both instruction RAM and dual-port RAM (DPRAM). Figure 6 illustrates how the 32 Kbytes of instruction RAM can be used to either store microcode patches, or as a general-purpose memory area for the e500 core. Similarly, the 32 Kbytes of DPRAM can be used for CPM-RISC parameter RAM for storing data structures and connection tables. Like the PowerQUICC II (HiP7) the PowerQUICC III can support eight microcode trap registers. Similarly, both the PowerQUICC II (HiP7) and PowerQUICC III have 128 Kbytes of ROM—an additional 68 Kbytes of ROM compared to the PowerQUICC II (HiP4).

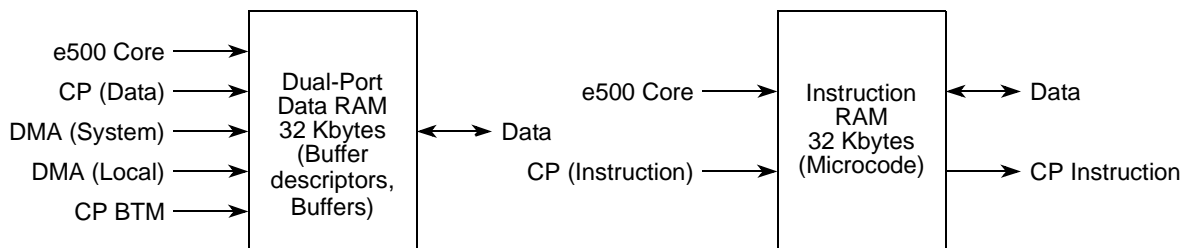


Figure 6. Internal CPM SRAM Configuration

### 3.2 Fast Communication Controllers

The PowerQUICC III CPM, like that of the PowerQUICC II, includes three full-duplex, fast serial communication controllers (FCCs). Each controller can be used to support 10/100 BaseT Ethernet, 45 Mbps HDLC (for example, up to E3/T3 rates) and transparent modes of operation. Both FCC1 and FCC2 can also be used to support up to 155 Mbps (depending on the adaptation layer), 16/8-bit UTOPIA Level II ATM.

For this reason, the functionality and programming model of the FCC has remained unchanged between the PowerQUICC II (HiP7) to PowerQUICC III.

### 3.3 Serial Communication and Management Channels

Figure 1 shows that the serial management channels (SMCs) that existed on the PowerQUICC II have been removed from the PowerQUICC III. Instead, the SMC UART functionality can be implemented on the faster serial communication channels (SCCs).

#### NOTE

SMC GCI functionality is no longer supported on PowerQUICC III.

The function of each of the four SCCs is slightly different on the PowerQUICC III than on the PowerQUICC II. On the PowerQUICC III, the 10 BaseT Ethernet support on the SCC has been removed and that functionality is now provided through one of the three FCCs, or through one of the two three-speed Ethernet controllers (TSECs), as shown in Table 2. Like the PowerQUICC II, the SCCs offer protocol support for UART, HDLC, HDLC bus, Appletalk/Localtalk, transparent, and BISYNC modes. In order to protect existing software legacy, the programming models for all of the supported protocols on the SCCs are the same as the PowerQUICC II.

### 3.4 Independent/Serial Direct Memory Access

In the PowerQUICC II, the CPM contains two physical serial DMA (SDMA) channels. The CPM uses these two physical channels to implement two dedicated SDMA channels for each of the FCC, MCC, SCC, SMC, SPI and I2C. In addition, four virtual SDMA channels can be assigned to the programmable and independent DMA (IDMA) channels. On the PowerQUICC III, PPC Little-Endian SDMA mode is not supported since this is typically not used in a big-endian system.

IDMA emulation is also not supported on the CPM block. The main reason for this is throughput. In order to implement SDMA or IDMA channels on the PowerQUICC II, valuable CPM bandwidth has to be used. On the PowerQUICC III the introduction of a dedicated 4-channel hardware DMA engine means that channels are more efficient and can be quickly set up and removed without wasting CPM bandwidth. The IDMA features of the PowerQUICC II, as well as a number of new features, exist in the new hardware DMA block on the PowerQUICC III. See Section 4.1, “e500 Coherency Module (ECM),” for more information about the 4-channel hardware DMA engine.

### 3.5 CPM Summary

Up to this point the focus has been directed to the various differences between the Communications Processor Module on both the PowerQUICC II and PowerQUICC III. Increased functionality and performance has been achieved while retaining the existing programming model in order to maximize customers existing software legacy. The following Table summarizes the main CPM differences between the PowerQUICC II and PowerQUICC III.

CPM Block	Feature	PQII (HiP7)	PQIII (HiP7)
CPM Core, Memory & Microcodes	CPM Frequency (Max)	300 MHz	333 MHz
	ROM	128K	128K
Fast Communication Controllers	Instruction RAM	32K	32K
	Dual-Port RAM	32K	32K
	# Trap Registers	8	8
	#	3	3
	16-bit ATM UL2	FCC1	FCC1
	8-bit ATM UL2	FCC2	FCC2
	HDLC	45 Mbps	45 Mbps
Serial Communication Controllers	Transparent Mode	Yes	Yes
	10/100 BaseT	Yes (x3)	Yes (x3)
	#	4	4
	10/100 BaseT	Yes (x4)	Not Supported
	UART	Yes	Yes
	Apple/Localtalk	Yes	Yes
	HDLC	Yes	Yes
	Transparent	Yes	Yes
Serial Management Channels	#	2	Not Supported
Multi-Channel Controllers IDMA	#	2	2
	#	4 Virtual Channels (Using CPM B/W)	4 DMA Hardware Implemented Channels
SDMA	#	2	2
Configuration	Modes Programming Model	Big & Little Endian 'Traditional PQII'	Big Endian Only 'Traditional PQII'

**Table 2. CPM Feature Comparison PowerQUICC II(HiP7) versus PowerQUICC III**

## 4 New Features on PowerQUICC III

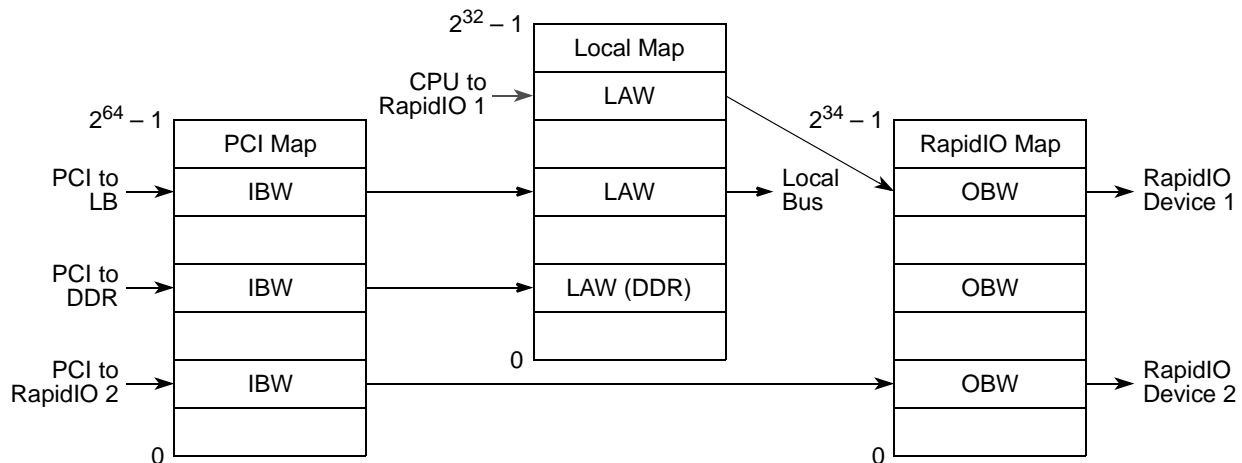
In addition to previously mentioned features, the PowerQUICC III also offers two integrated 10/100/1000 Ethernet controllers, a DDR SDRAM memory controller, a 64-bit PCI-X/PCI controller, and a RapidIO™ interconnect. This high level of integration simplifies board design and offers significant bandwidth and performance for high-end control-plane and data-plane applications. New internal blocks such as 4-channel hardware DMA engine, E500 coherency module (ECM) and non-blocking switch fabric (OCeaN), ensure rapid transfer of data between all of the various peripheral modules and the outside communication network.

### 4.1 e500 Coherency Module (ECM)

At the heart of the PowerQUICC III is the ECM. The primary role of the ECM is to guarantee memory coherency between the external memory interfaces (primarily DDR SDRAM) or memory on the local bus, and the cache memory—either the L1 cache of the e500 core, or the L2 cache of the memory complex. If an address falls into one of the ECM's defined windows it is forwarded to the correct destination, for example, if a buffer on the local bus is marked as snoopable, all traffic to that buffer will be routed using the ECM. This ECM operation highlights the difference in the transaction style architecture of the PowerQUICC III, when compared to the shared bus architecture of the PowerQUICC II.

The implementation of the ECM is such that transactions across it are not implicitly coherent. The internal bus is snooped by e500 L1 and L2 caches to maintain coherency only if the transaction across the ECM has the GBL bit set to mark it as a global transaction. If this is not the case then the ECM is used as a simple conduit for the transaction to get to its destination. Thus, only global transactions across the ECM are coherent transactions; all others (between the CPM and the local bus, and across the on-chip fabric) are non-coherent. In reality, the majority of all internal data transfers are in some way processed by the ECM.

Inside the PowerQUICC III is a 32-bit local address mapping unit than allows 32-bit addressing structures to map to larger addressable devices completely independently, for example, 32-bit PowerQUICC III address to 64-bit PCI address. As a result, the PowerQUICC III system map normally consists of a 32-bit local address space and a larger external address space. Accessing external address space is achieved by using inbound and outbound translation windows, whereas inbound transactions use local access windows (LAWs). It is through these LAWS that the ECM knows where/how to route transactions throughout the internal PowerQUICC III architecture. Only transactions that hit a particular IBW or LAW are actually processed.



**Figure 7. Address Mapping Example**

In Figure 7, four different address mapping examples are shown:

1. PCI to local bus transaction: inbound window (IBW) to local address window (LAW) that the ECM then routes to the local bus.
2. PCI to DDR transaction: inbound window (IBW) on PCI space to local address window (LAW) for DDR. In this example the transaction comes through OCeaN by way of the ECM block and on to the DDR.
3. CPU e500 to RapidIO: direct from CPU local address window (LAW) which the ECM then routes to the RapidIO outbound window (OBW) for the RapidIO device.
4. PCI to RapidIO: from PCI space inbound window (IBW) direct to RapidIO outbound window (OBW) for the RapidIO device. Here the ECM is not involved, the transaction is simply a OCeaN Port-to-Port transaction, local to the OCeaN block.

The coherency module has been optimized for low-latency access to the DDR memory by performing 'speculative reads' using a separate internal bus to the DDR controller. This speculative read starts a transaction in the DDR memory before a response from the cache is received on whether the access is



actually a cache hit or not. If the access is a cache hit then the speculative read is ignored and then discarded; otherwise, if the access is a cache miss, the speculative read transaction completes and a couple of extra clock cycles have been saved.

There are two arbitration mechanisms inside the ECM; the I/O arbiter and the CCB arbiter. An I/O (for example, OCeaN, CPM or either of the two TSECs) must first compete for an internal ECM grant for the CCB with each of the other three I/Os. Arbitration can be won through either a higher priority level access or a lower priority waiting request. Once arbitration is won, the winning bus request must also compete with the e500 core for CCB ownership (using the same arbitration mechanisms) before the transaction is finally entered into the transaction queue. The number of posted transactions by the winning controller is set through the EEBACR[A\_STRM\_CNT], which by default is set to three. As a result, by limiting the number of posted transactions, higher priority transactions or lower priority waiting requests can be arbitrated for the CCB with the minimum of delay.

## 4.2 DMA Controller

The DMA controller on the PowerQUICC III is a dedicated 4-channel, independent, general-purpose controller that can be used by both local and remote masters to transfer data between any memory mapped area of the PowerQUICC III (for example, PCI, Local Bus, or Rio).

The implementation of the DMA controller is again similar to that of the IDMA implementation on the PowerQUICC II with ‘traditional’ direct/auto buffer, and chaining modes being supported. However the PowerQUICC III implementation offers additional features as well as off-loading the main CPM from performing DMA transactions. The result is an efficient, user-configurable solution that can be controlled either internally using software or externally using its external DMA I/O pins.

### 4.2.1 DMA Controller Operation Modes

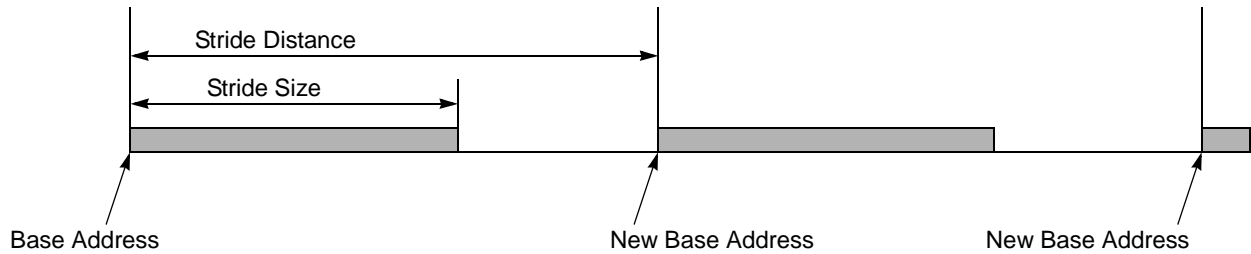
In addition to the direct and chaining modes of operation, the following new features have been implemented into the hardware DMA block:

- Extended mode (Stride capability)
- Bypass ATMU mode (RapidIO)
- External Master mode
- Channel Halt/Abort/Continue mode
- Destination/Source Address Hold Enable mode

#### 4.2.1.1 Extended Mode—Advanced Chaining/Stride Capability

The advanced chaining mode allows the user to set up a series of buffer descriptors all referenced from a linked list. The DMA controller, using this linked list, can then walk through multiple buffer descriptors, allowing complex DMA transactions to be performed.

When operating in stride mode, the PowerQUICC III uses the current base address and adds the user-defined stride distance to determine the next quantity of data to be transferred. The stride size defines the amount of data to be transferred before jumping to the address of the next block of data. This feature is important for DMA transfers to and from fixed memory structures, such as buffer descriptors and buffers. Figure 8 illustrates how the stride mechanism operates.



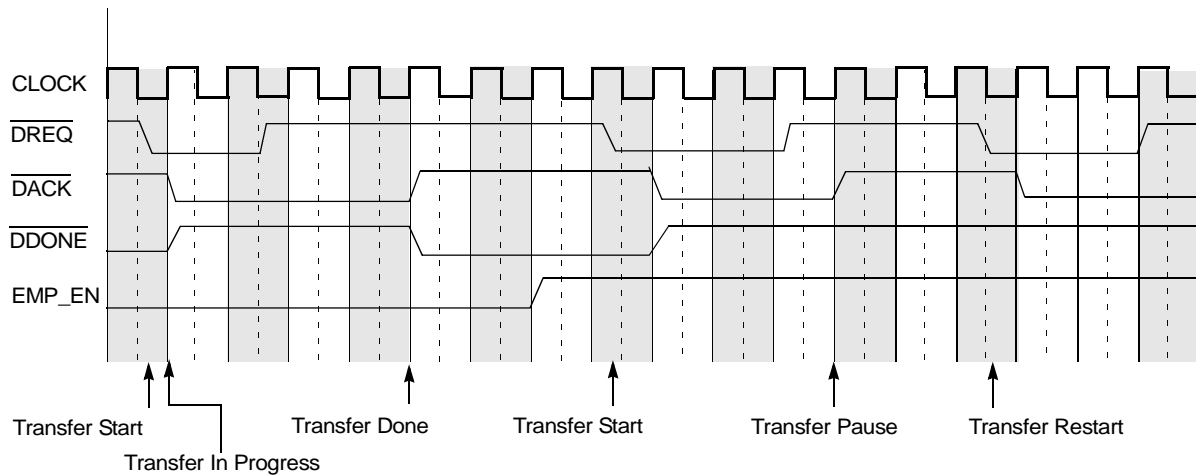
**Figure 8. DMA Stride Mechanism**

### 4.2.1.2 Bypass ATMU Mode

The ATMU bypass mode is a feature that applies specifically to the MPC8560 device and the RapidIO interface. In this mode, the normal address translation mapping unit (ATMU) of the RapidIO is bypassed so that messages and maintenance transactions can be sent across RapidIO switches and endpoints without any mapping or translation overhead. In order to do this, the ATMU parameters (priority, transaction type, targetid, address) normally obtained from the RapidIO ATMU are supplied in bypass mode.

### 4.2.1.3 External Master Mode

The PowerQUICC III DMA block has the ability to either handle local transactions within the device or handle external master initiated DMA transfers through its interface (I/O) ports. As in the PowerQUICC II implementation, external handshake pins are used to request a DMA service or indicate an action: (DMA\_DREQ—data request: transfer start or restart from pause), (DMA\_DACK—data accolade: DMA transfer in progress) and (DMA\_DDONE—data done: DMA transfer complete). Figure 9 illustrates the relationship between the external DMA and the external master pause enable pins.



**Figure 9. External DMA Transfer with Pause**

Data transferred between the PowerQUICC III and external masters can be controlled much more effectively than on the PowerQUICC II due to the new pause mode. As before, the DMA controller on the PowerQUICC III can operate in continuous mode and will DMA data out to the external master. However, for external devices with small FIFO's such as FPGAs the DMA can quickly overload the FIFO and create overruns. The new pause mode prevents devices with small FIFO's filling up as the DMA is paused in between data transfers by setting MR[EMP\_EN]=1 and MR[BWC]=<size of burst>.

#### 4.2.1.4 Channel Continue/Halt/Abort

The DMA controller introduces three new channel control mechanisms, channel continue, halt and abort. Channel continue is used in chaining. In basic mode, the current link address is re-fetched when the channel is continued and in extended mode, the current list address is re-fetched. In the event that the last link or list address has been reached, then the DMA will return to the channel halted state. It should be noted that even when the channel is busy, it is possible to continue building new DMA descriptors by making use of the channel continue feature as follows:

- Set end of links/lists descriptor (EOLND/EOLSD)
- Build new descriptor list
- Clear EOLND/EOLSD
- Restart DMA by issuing a Channel Continue (MR[CC])

Channel abort is recognized by MR[CA]=1 and SR[CB]=0 and is used to finish the current sub-block transfer and halt the relevant channel. Thus, channel halt, determined by SR[CB]=1, indicates that the DMA engine has completed its current DMA transaction.

Another important feature with the PowerQUICC III DMA controller is that it can assign bandwidth on a per channel basis—this prevents any single channel from consuming all of the DMA's available data bandwidth. Using bandwidth control allows a channel only to use its allotted bandwidth, after which, the arbiter grants (round-robin) the next channel access to the shared resources.

#### 4.2.1.5 Destination/Source Address Hold Enable

This feature is typically used in scenarios when reading or writing to some form of FIFO structure. If for example an external device has a FIFO involved in the DMA process then multiple reads/writes will be performed to the same address. In such a case, it is more efficient to hold the current source or destination address and simply transfer a specific chunk of data. The transfer size is user specified through the source and destination transfer size registers (MR[SAHTS] and MR[DAHTS] respectively).

### 4.3 Local Bus Controller

The local bus controller on the PowerQUICC III performs similar functions to the PowerQUICC II implementation. The three main PowerQUICC II memory controller machines are still supported on the PowerQUICC III:

- GPCM—General-Purpose Chip Select Machine for classic memory interfaces such as SRAM memory or FLASH.
- UPM—User Programmable Machine that configures an internal timing generator to create non-standard timings for peripheral devices.
- SDRAM—Single Data Rate SDRAM controller allowing SDRAM devices to be gluelessly connected.

Although functionally similar, the PowerQUICC III implementation of the UPM has three dedicated UPMs unique to the local bus. On the PowerQUICC II, these UPMs also had to be shared with the 60x bus. Similarly, on PowerQUICC II, the twelve available chip selects had to be shared with the 60x bus, on the PowerQUICC III, there are eight dedicated chip selects specific to the local bus. The ability to support bank based interleaving on the SDRAM controller that was on the PowerQUICC II has been removed. On the SDRAM controller on the PowerQUICC III, only page-based interleaving is supported. The reason for this is simple; Page-based interleaving allows a greater degree of flexibility and control when accessing connected SDRAM devices.

The major difference between the two devices is that the local bus controller on the PowerQUICC III can operate up to 166 MHz—the maximum supported frequency on the PowerQUICC II is 100 MHz. The local bus on the PowerQUICC III runs with a fixed frequency referenced from the main system frequency, SYCLK. It should also be noted that the local bus controller runs synchronized with the CCB clock, hence if you have the CCB running at 333 MHz then the frequency of the local bus is divided down from this to either 2:1, 4:1 or 8:1. (Users should note that the 8:1 divide option is the default configuration from reset). The local bus also has optimized state machines which are designed to support the specific transaction size that is generated by the CPM. Hence, a 32-bit configured local bus will have minimal associated overhead when passing traffic between external peripherals and the CPM.

The local bus programming model remains the same as the PowerQUICC II with it supporting 8 chip selects, each individually configurable to either 8, 16 or 32 bits wide. Each individual chip-select is capable of supporting up to 2 Gigabytes of address space. As before, the local bus supports both natural parity and read-modify-write (RMW) data parity checking.

One difference with the local bus controller on the PowerQUICC III is that it is possible to select a wider range of addressable banks within a connected device due to its 34-bit address decoding, as opposed to 32-bit address decoding on the PowerQUICC II. In order to save pins during the design of the PowerQUICC III design, it was decided that a 32-bit multiplexed address/data interface would be implemented. This is an important difference from the PowerQUICC II as any design will require an external buffer and latch combination with which to demultiplex the muxed address/data signals. The logic for controlling these external devices is integrated into the PowerQUICC III design. (On the PowerQUICC II, the local bus had 18 separate address pins and 32 data pins).

## **4.4 Dual Data Rate (DDR) Memory Controller**

The dual data rate (DDR) SDRAM controller is a new feature on the PowerQUICC III and currently supports DDR Type 1 SDRAM up to 333 MHz—as DDR memory is clocked on both edges of the input clock, the actual maximum clock rate is 167 MHz. (Users should note that the MCK# [5:0] signals are simply the inverted clocks from the MCK [5:0] signals and not true differential clock signals.)

On the PowerQUICC II, the 60x bus allowed users to connect to SDRAM, Flash and other 60x bus compatible peripherals. However on the PowerQUICC III, the DDR controller is a dedicated 64-bit data bus (+ 8-bit ECC) with no external master capability and it is used exclusively for DDR memories as the main system bus. Users should note that connection to other external masters can be readily implemented through either the PCI or RapidIO interfaces.

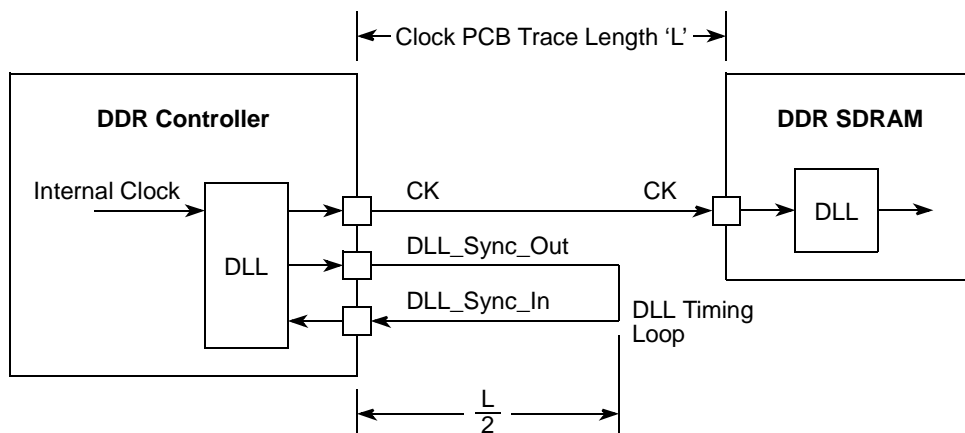
### **4.4.1 Feature Summary**

The DDR controller is capable of supporting discrete or DIMM DDR memories between 64 Mbit and 1 Gbit in size. The DDR controller has four chip selects that can be used to access a (theoretical) maximum of 1 gigabyte addressing space. It should be noted that in reality users cannot use the full 4 gigabytes for DDR, since 4 gigabytes is the maximum addressing that the whole PowerQUICC III can handle. Thus, the DDR controller has been restricted through design to have a maximum addressing limit of 3.5 gigabytes to allow other connected peripherals to have space in the PowerQUICC III memory map.

### **4.4.2 DLL Delay Compensation Loop**

Since the PowerQUICC III operates at much higher frequencies than the PowerQUICC II, the clocks for the memory interface (the Digital Locked Loops or DLLs) have been integrated into the PowerQUICC III

design. In doing so, the AC timing requirements can now be accurately optimized on a per-design basis using a PCB/layout feedback delay loop as shown in Figure 10.



**Figure 10. DDR SDRAM DLL Feedback Delay**

During layout of the final PCB, the layout engineer needs to set the length of this compensation feedback loop to be the same as clock tree on the chip. Thus, the internal DLL clock will compensate for the external clock tree delay and be 100% aligned, allowing maximum margin on the AC timing specifications of the DDR SDRAM. Of course, this allows maximum flexibility as users can create more or less set up and hold time to suit their end system requirements.

#### 4.4.3 Six Clock Pairs

Normally in traditional clock design, you should not supply more than three input pins with one clock otherwise the resulting clock may be degraded under loaded conditions. Due to this reason, the DDR controller supplies six clock pairs for use with DDR memory. (Some degree of external clock buffering would be additionally required if users decide to deviate beyond this clock design guideline).

#### 4.4.4 Sleep Mode Support for Self Refreshing SDRAM

Another feature on the DDR SDRAM controller is the ability to support sleep mode for self refreshing SDRAM. This feature is useful if the clocks or power on the chip have to be stopped for some reason. Prior to shutting down the power to the chip, this mode executes a self refresh command to the DDR SDRAM which ensures that the DDR remain intact when it wakes up from the clock stop mode. This allows the DDR SDRAM to gracefully recover from where it was before the power/clocks were stopped.

#### 4.4.5 Dynamic Power Management

This feature allows users to maximize the bandwidth capabilities of the DDR SDRAM by using on-the-fly power management. When the DDR transaction pipeline begins to empty, the device automatically uses the clock enable to put the device into power conservation mode. As new DDR transactions arrive, the devices comes out of power conservation mode and begins pipelining those new transactions for execution.

## 4.5 On-Chip Network (OCeaN)

The on-chip network (OCeaN) is a 4-port, full crossbar switch fabric with 64-bit inbound/outbound interfaces. Through each of the four ports, a total of 11 Gbyte/sec aggregated bandwidth can be supported on the switch fabric (2.7 Gbyte/sec peak bandwidth per port).

This switch fabric has been designed in such a way as to be non-blocking through the use of pipelining, transaction posting and priority traffic flows. Up to six different transactions can be posted into the fabric before it stalls awaiting a transaction grant on the first one—enabling much more efficient use of the fabric. Figure 11 shows the internal logic arrangement (consisting of wires and muxes) that make up the physical on-chip network.

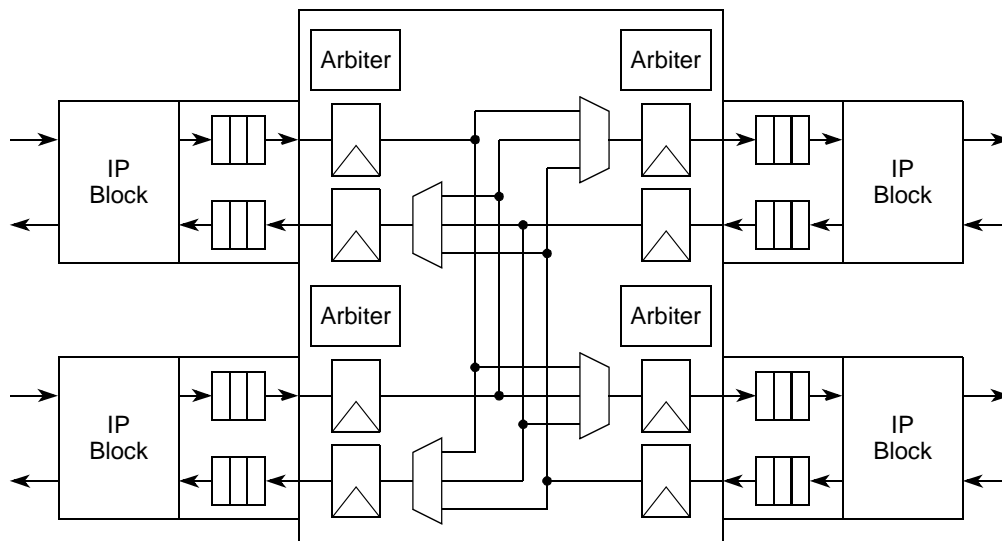


Figure 11. Internal Structure of On-Chip Network (OCeaN)

Under heavily loaded traffic conditions, traffic is prioritized in OceaN using different traffic classes, some of which have a higher priority than others. This allows higher priority traffic to make more forward progress than lower priority traffic.

This latter point is the key mechanism by which processing deadlocks are avoided through the use of priority reordering. For example, a request can be made from a presently blocked port, to another unblocked port in order to pass a blocked transaction ahead. This helps to remove head-of-line blockages when switching traffic between PCI/PCI-X, RapidIO, the DMA controller and the internal system bus.

## 4.6 Three-Speed Ethernet Controller

The three-speed Ethernet controller (TSEC) is a new hardware block on the PowerQUICC III (providing Layer 1 & 2 functionality), allowing 10/100 & 1000BaseT Ethernet traffic to be transmitted and received. Twin TSEC blocks are on both the MPC8540 and MPC8560 and are IEEE 802.3/3u/3x/3z and 802.3ac and 802.3ab specification compliant. Although both TSEC blocks are flexible enough to be programmed in 10/100/1000 BaseT and in MII, RMII, RGMII, TBI, RTBI modes, users should note that both the RGMII and RTBI are 5V interfaces, all of the remaining interfaces are 3.3V. Power supply constraints mean that users can only implement either 5V compliant interfaces or 3.3V interfaces—mixing different 5V/3.3V interfaces is not a valid configuration.

# Freescale Semiconductor, Inc.

## New Features on PowerQUICC III

As on the PowerQUICC II (using the 10/100BaseT, FCC-implemented Ethernet controller), the PowerQUICC III TSEC can support both full and half-duplex modes (although the 1000BaseT implementation of the TSEC only supports full duplex mode of operation). The reader should also note that on the PowerQUICC II HiP7 device, the 10/100BaseT Ethernet can be implemented through a reduced MII interface. Additionally, the TSEC supports a ten bit interface (TBI) which is used to connect to a SERDES (SERialize/DESerialize) for fibre/optical transceiver applications. An Ethernet feature comparison between the PowerQUICC II and PowerQUICC III is summarized in the Table 3.

**Table 3. Ethernet Feature Comparison**

Feature	PowerQUICC II	PowerQUICC III
<b>Supported Interfaces</b>		
10/100 MII	Yes	Yes
10/100 RMII	Yes, Only on HiP7 Devices	Yes
1000TBI	No	Yes
10/100/1000 RGMII	No	Yes
1000 RTBI	No	Yes
10/100/1000 GMII	No	Yes
<b>Capabilities</b>		
CPM Bandwidth Used?	Yes	No, Dedicated H/W Block
Out of Sequence Frames	Yes	Yes
Pause Frame Capability	Yes	Yes
Jumbo Frame Support	No	Yes (9.6KB Max)
Frame Padding	Up to MINFLR	Up to 64-bytes
Polling Frequency	Every 256 Ethernet Clocks	Every 512 Ethernet Clocks
Truncation Capability	No	Yes
Supported Modes	OOS, Pause	OOS, Wait & Pause
Graceful Rx Stop	No	Yes
Internal FIFOs	No – Uses DPRAM	Yes – 2KB on Board

As in the FCC Ethernet controller (100BaseT) on the PowerQUICC II, the TSEC block supports both internal and external loopback. However, users should note that internal loopback on the PowerQUICC III is only supported in GMII, MII or TBI modes—no internal loopback is supported when the controller is configured in RGMII or RTBI mode. Similarly, as on the PowerQUICC II, external loopback is performed at the interface pins. On the PowerQUICC II, 64-bits were made available for address hash matching, on the PowerQUICC III TSEC implementation this has been extended to 256 bits, thereby allowing a greater range of addresses to be decoded.

Due to the high bit rates involved, the TSEC has a number of hidden operational modes that it uses to internally manage its FIFO's to help avert either underrun or overrun conditions. Entering internal TSEC starve mode is an indication that the Tx FIFO is in danger of under-running. When the FIFO nears empty (within 0.5 Kbytes) the starve state is entered and the ECM automatically increases the priority of the internal DMA and also that of the TSEC block, in order to avert the underrun. When the FIFO fills back up to within 1 Kbyte of empty, the starve mode is shutoff. Similarly in the case of frame reception, when the

2-Kbytes internal FIFO reaches the 1-Kbyte level, an internal panic alarm raises the priority of the TSEC block to try to avoid an overrun situation. If the FIFO continues to fill then at the 1.5 Kbytes full level, an emergency panic increases the DMA priority to its maximum and sends a PAUSE frame to allow the data that has already been received, to be processed before other data is received.

Another useful feature is the ability to contour the behavior of the TSEC with regard to frame interrupts on both transmitted and received frames. This feature is known as ‘Interrupt Coalescing’ and can be implemented using either a frame counting or timer threshold method. In the frame counting implementation, the user can configure interrupt coalescing such that frame interrupts are deliberately grouped in order to reduce the number of raised interrupts, thereby avoiding interrupts bandwidth congestion due to frequent consecutive interrupts. Similarly, stale frame interrupts can be avoided using the timer threshold approach, beyond which frame interrupts not yet raised are forced. This is an important feature for helping to optimize packet processing of TSEC frames, by the core.

Although the TSEC is a new hardware block, its programming model is still backwards-compatible with the PowerQUICC II. As in the PowerQUICC II case, the TSEC transfers data into and out off buffers which are pointed to by buffer descriptors and connection tables. This helps ensure that any legacy software is retained.

## **4.7 PCI/PCI-X**

The PCI interface on the PowerQUICC III is the same functional block as on the PowerQUICC II and is version 2.2 compliant, supporting transaction speeds between 16 to 66MHz. As on the PowerQUICC II device, the PowerQUICC III PCI controller can either use its internal arbiter to control PCI transactions or an external arbiter. The PowerQUICC III PCI interface can be configured to operate either as an agent or host or configured as a master or slave PCI device. As a master device, the PowerQUICC III PCI controller manages both memory and I/O transactions, whereas as configured as a slave device, only memory transactions are controlled.

In addition to PCI support, the PowerQUICC III has a new feature which implements the PCI-X standard (version 1.0A) and can support frequencies up to 133MHz. It is perhaps useful to review the standard PCI bus architecture that is available on both the PowerQUICC II and the PowerQUICC III, before looking at the actual differences between the PCI and PCI-X interface.

The PCI bus architecture is a hierarchical, multi-master arbitration scheme that uses either 32 or 64-bit addressing to post transactions onto the PCI bus. Transactions can be either accepted, retried or deferred. In the later two cases, the master repeats any transaction that needs to be retried and deferred transactions can be accepted and started by the target whilst the master retries the transaction. Since the PowerQUICC III has five different pairs of Request/Grant pairs, up to five external PCI masters can be supported. It is also worth noting that the PCI interface on the PowerQUICC III is a dedicated interface and does not multiplex its pins with other functional blocks as it did on the PowerQUICC II. The PowerQUICC II Local Bus pins and PCI were multiplexed and allowed the use of one interface or the other, for example, local bus or PCI, but not both. This restriction has now been removed on PowerQUICC III.

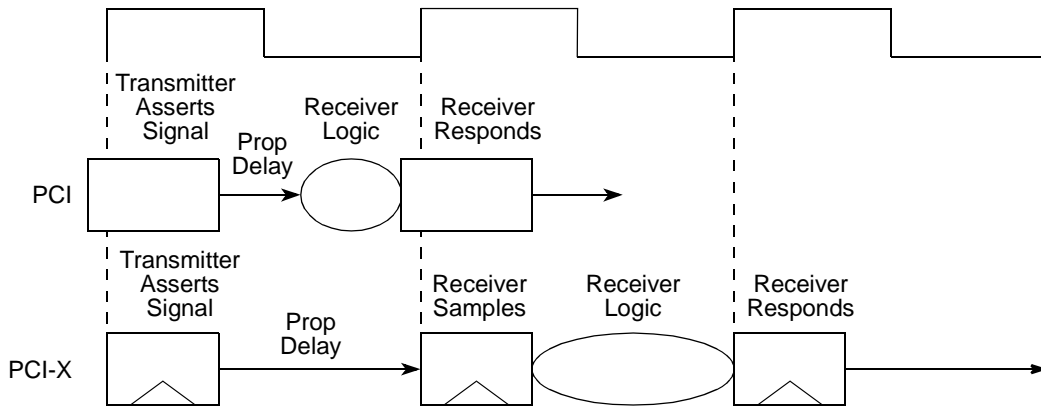
The performance of the PCI interface is enhanced by the two level round-robin arbitration algorithm used in the arbiter and through the ability to do mirror and pre-fetched PCI read accesses. Whilst the PCI interface supports both inbound and outbound data streaming the amount of data that can actually be streamed is limited by both the depth of pre-fetching and the target disconnect limit of the PCI specification. On the PowerQUICC III this disconnect will occur after two cache lines (i.e. after 32 bytes). This helps prevent PCI devices from hogging the bus, thus avoiding system bottlenecks and interface starvation, when operating high speed interfaces such as RapidIO or Gigabit Ethernet.



The main difference between the PCI block implemented on the PowerQUICC II to that of the PowerQUICC III is that the PowerQUICC II implements a 32-bit addressable PCI interface and can only support three Request/Grant signal pairs i.e. only a maximum of three external PCI masters can be supported. Apart from these additional address and Request/Grant signals, all other PCI signals implemented on the PowerQUICC II remain the same on PowerQUICC III.



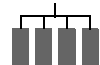


### 4.7.1 PCI & PCI-X System Differences

Perhaps the most obvious difference between PCI and PCI-X is the speed increase of the interface which can now run up to 133MHz. PCI-X is a latch-to-latch based protocol which allows more timing margin for propagation delay and receiver logic. This is shown below in Figure 12.



**Figure 12. PCI/PCI-X Timing Analysis**

Unlike the PCI specification, PCI-X is a split transaction based protocol which has separate arbitration transactions for both bus request and response. Again, this helps to improve bus latency and utilization. Since the maximum transaction size in a PCI-X system is limited to 4K bytes, the worst case system latency dramatically improves. Also, bus utilization is improved by the restriction of both targets and initiators in inserting wait states. Targets cannot insert wait states after the first data beat and both initiators and targets can only end a burst on a naturally aligned 128-byte boundary. Each transaction in a PCI-X system carries a byte count and the identity of the initiator that was not possible in the PCI specification. Hence, PCI-X systems offer better buffer management capabilities and make more efficient use of the available system memory.

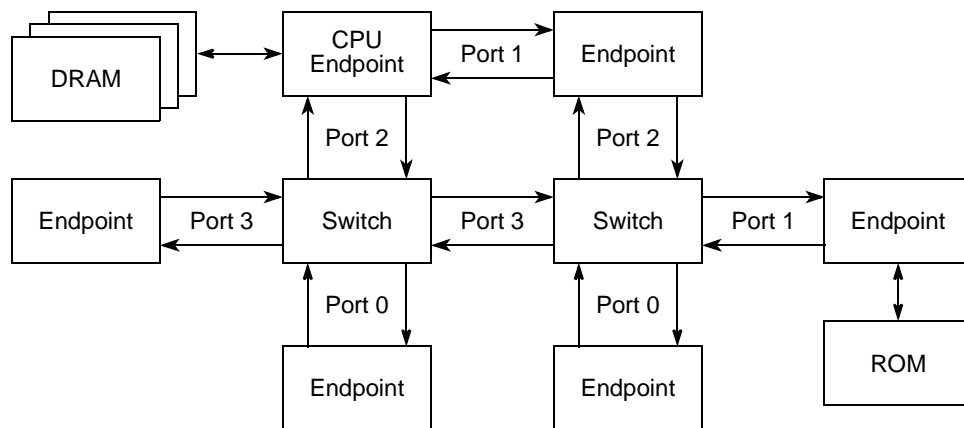
PCI Width	PCI Frequency	PCI Bandwidth	PCI Slots	PCI-X Slots
32-bit	33 MHz	133 MB/s		N/A
64-bit	66 MHz	528 MB/s		
64-bit	100 MHz	800 MB/s	N/A	
64-bit	133 MHz	1066 MB/s	N/A	

**Figure 13. PCI/PCI-X System Comparison**

Figure 13 illustrates the improved system performance and maximum slot capability between PCI and PCI-X systems.

## 4.8 RapidIO

RapidIO is a packet-switched technology used for high speed point-to-point bus interconnect, compliant to Rev 1.2 of the RapidIO specification. The PowerQUICC III is the first PowerQUICC family device (and the first device to market) to incorporate an integrated 8-bit RapidIO controller. Hence, this is a completely new functional block currently implemented only on the PowerQUICC III family of devices, and not a supported feature on PowerQUICC II. The RapidIO specification is divided into three different specification layers: logical, transport and physical. RapidIO networks are built around two basic system building blocks—RapidIO endpoints and switches. RapidIO endpoints as the name suggests create and consume RapidIO messages with the RapidIO switches simply passing packets between its different ports without interpreting them. Figure 14 illustrates a typical RapidIO network.



**Figure 14. A Simple RapidIO Network**

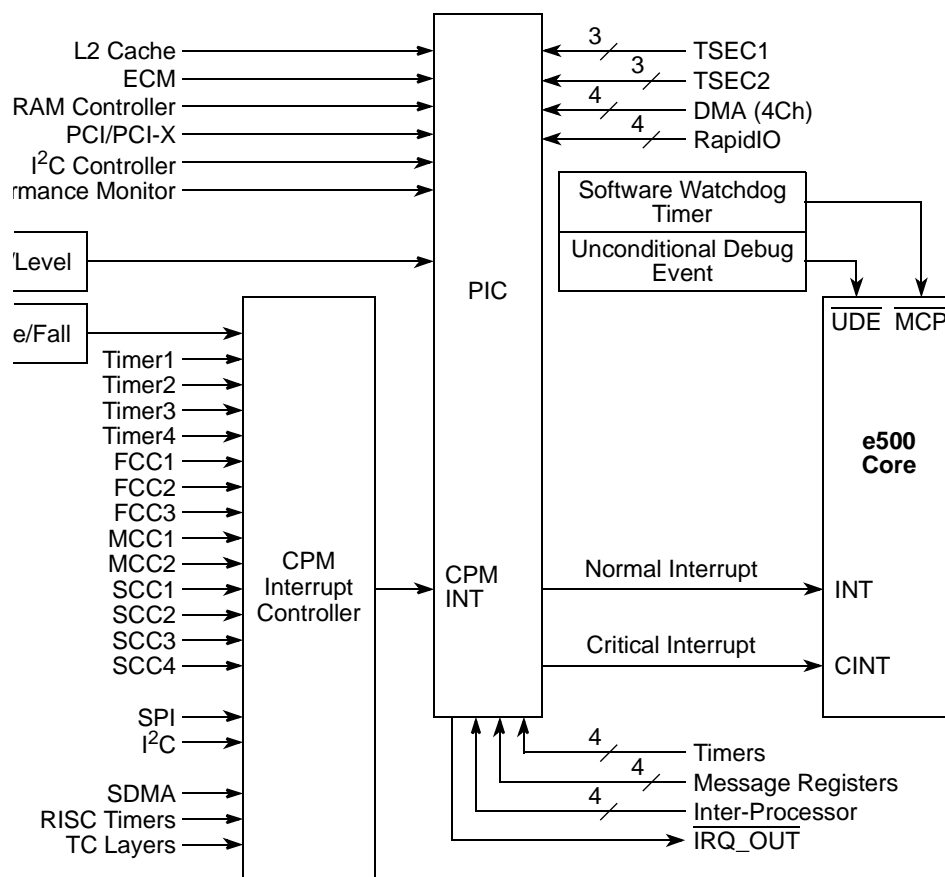
The RapidIO physical specification on the PowerQUICC III has an 8 bit wide bus that uses low-voltage differential signaling (LVDS) I/Os. Since RapidIO uses source synchronous double data rates, it can be run at 500 MHz clock which can generate 1 Gigabit per pin pair i.e. running an 8-bit wide end-point, up to 1 Gigabit per second/per pair can be received/transmitted concurrently. In order to ensure maximum data throughput, the RapidIO implementation on the PowerQUICC III supports four priority levels.

In the transportation specification, each RapidIO endpoint transmits 256-byte transaction packets which can be controlled by atomic packet semaphores i.e. increment, decrement, set and clear. In addition to these functions, RapidIO also has a messaging unit containing one inbound and one outbound data structure. Each message, or telegram can be up to 4 Kbytes in size (i.e. a total of 16 packets at 256 bytes per packet). Efficient system operation can be created using these telegrams to inform endpoints of a particular task to be performed.

## 4.9 Programmable Interrupt Controller (PIC)

On the current version of PowerQUICC III all interrupts are routed via the programmable interrupt controller (PIC) to the e500 core. One of the key design goals of the PIC was to ensure consistency and maximize existing PowerQUICC software legacy by cascading the PowerQUICC II CPM interrupt controller with that of the PowerQUICIII PIC.

A diagram of the PowerQUICC III PIC is shown in Figure 15.



**Figure 15. PowerQUICIII: Programmable Interrupt Controller**

One of the obvious differences between the PowerQUICC II and PowerQUICC III is the inclusion of the interrupt sources from the new PowerQUICC III modules i.e. TSEC, DMA, DDR SDRAM controller, PCI, I<sup>2</sup>C controller, ECM, RapidIO and the performance monitor. A new feature with BookE is the definition of a new critical interrupt which provides an extra level of interrupt nesting from either the critical input, watchdog timer, or from machine check sources. (As on the PowerQUICC II HiP7, critical interrupts can be taken during normal (non-critical) operation or during regular program flow and use their own critical save and store registers to save state when they are taken).

There are 16 interrupt priority levels (0–15) on the PowerQUICC III with level 15 being the highest priority and level 0 disables interrupts from a particular source. This is the opposite to what was on the PowerQUICC II—level 0 was highest priority, level 15 lowest. Additionally, the PowerQUICC III has four new Inter-Processor interrupts and four message registers that will be used on next generation multi-core systems.

On future derivatives, the addition of new or multiple cores will be allowed by the PIC enabling it to be fully configured and able to route interrupts to any individual core. While the concept of interprocessor interrupts apparently makes little sense in a single-core device, this feature can serve as a doorbell type interrupt because external bus masters can write to these interrupt/message registers.

## 4.10 Enhanced Debug Features

The PowerQUICC III has a number of new debug features which include a built-in performance monitoring mechanism. The performance monitoring can be split into two distinct functions: the e500 core performance monitor and the main PowerQUICC III system performance monitor.

The e500 core performance monitor consists of 32-bit counters that can be cascaded to count up to 4 of 89 defined core events, for example, branches, and cache hits/misses. Access to these performance monitor counters is achieved through two new performance monitoring instructions: **mtpmr** and **mfpmr** (move to/from performance monitoring register).

In the case of the PowerQUICC III system performance monitor, eight 32-bit counters are available for general system monitoring with up to 64 system events can be counted on any one of these event counters. An additional 64-bit counter is also available for larger counting cycles. Using these counters, the user can monitor up to 576 system events. Typical examples of such events can be found in Table 18-10 of the PowerQUICC III user's manual, for example, read or writes from DMA/RapidIO/PCI, ECM dispatch events, dropped/accepted frames etc.

Also, the provision of a configurable CLK\_OUT signal allows the user to select whether the platform (CCB) or main system clock (SYSCLK) is made available on an external pin for further debug. Users can select this option via the clock out control register (CLKOCR) as detailed in the MPC85xx Users Manual.

## 5 Initialization, Reset and Boot Procedures

Existing users of the PowerQUICC I and II device will be familiar with the hard reset configuration word (RSTCONF) which is used to determine the device configuration while PORESET changes from assertion to negation. On the PowerQUICC III, the POR configuration stage is a little different. In this section, we will look at the different initialization, reset and boot procedures of the PowerQUICC III.

### 5.1 POR Configuration

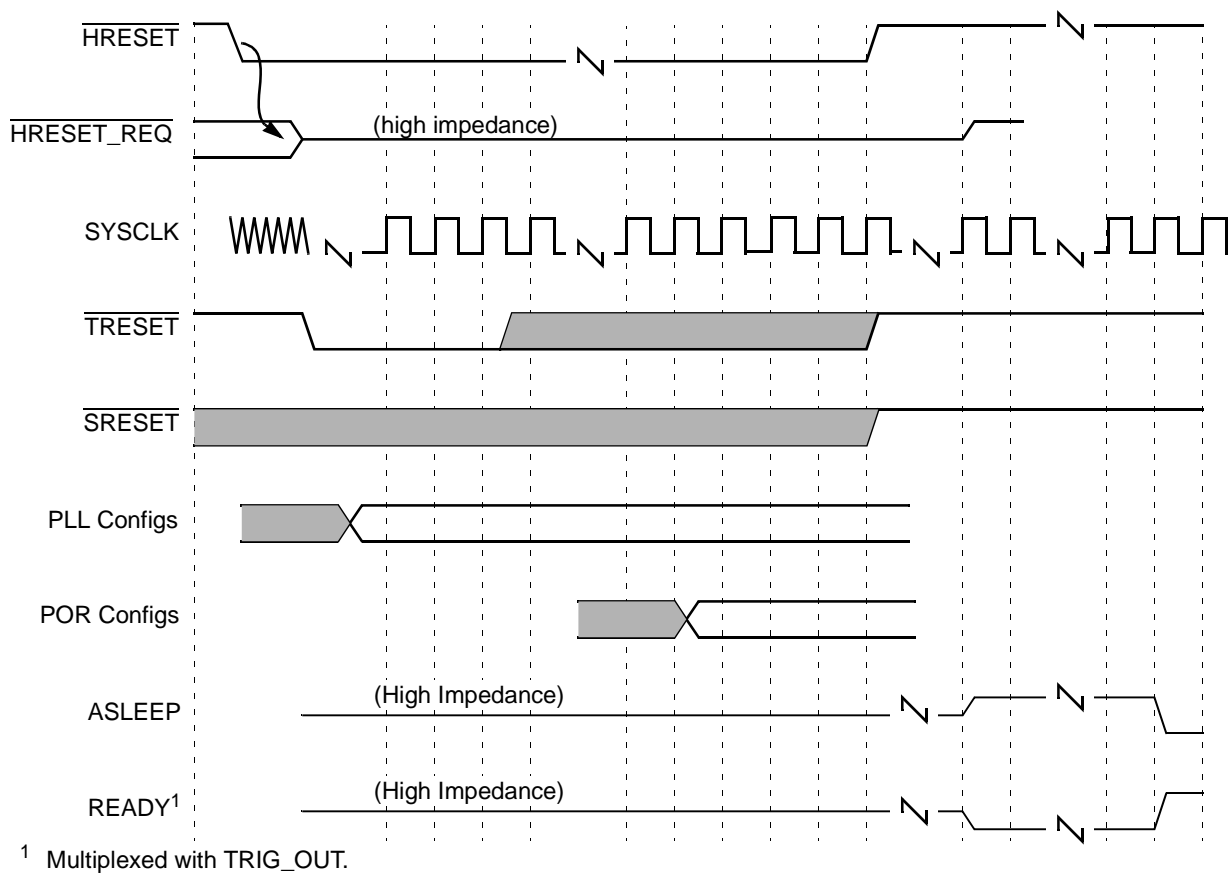
Perhaps the biggest difference between power on reset (POR) between the PowerQUICC II and the PowerQUICC III is the removal of the bi-directional HRESET and PORESET pins on the PowerQUICC III. The SRESET pin on the PowerQUICC II, when asserted, is driven by the chip for 512 clock cycles before it is released and the SRESET flow is exited. On the PowerQUICC III the function of the SRESET pin is different and when asserted, causes a machine check exception (*mcp*) assertion to the core and also a reset to the CPM.

On the PowerQUICC III, when  $\overline{\text{HRESET}}$  is asserted, various device functions are enabled by pulling certain input pins high or low via external pull-up or pull-down resistors, or using configurable tri-state buffers. At this point it is perhaps worth noting that all logic on the chip is powered by VDD, only the I/O's reference G/L/OVDD. (Ideally, VDD and AVDD should be either powered up simultaneously, or before the other chip power supplies).

Once these configuration inputs have been valid for at least four clock cycles, the  $\overline{\text{HRESET}}$  signal is negated by the core and then the CPM PLL and e500 DLLs begin to lock. After the DLLs have completed locking, the boot sequencer (if enabled) is released and can load its configuration data. The PowerQUICC III then enters the ready state and the e500 core indicates this to the external world by the negation of the ASLEEP signal and assertion of the READY signal. Chapter 4 of the MPC8560 (MPC8540) Users Reference Manual gives a list of all the relevant configuration pins and the complete reset procedure. Once the PowerQUICC III has come out of reset and the system DLLs have locked, the boot process can begin as configured by the configuration information supplied by the user. This can take the form of:

- e500 boot—if configured.
- initialization/activation via the PCI/RapidIO—if active.
- Pre configuration of the PowerQUICC III performed by the I2C boot sequencer.

As a number of input pins are required to correctly set up the configuration of the PowerQUICC III, a number of read-only status registers called the Global Utility registers have been added into the core. Using these registers, users can check whether the device has been successfully configured. For example, the PORPLLSR (POR PLL ratio status register), when read, details how the e500 core and CCB (platform) clock ratios have been configured coming out of the POR configuration cycles. This register along with the PORBMSR (POR boot mode status register), PORIMPSCR (POR I/O impedance status and control register), PORDEVSR (POR debug mode status register), PORDBGMSR (POR debug mode status register) and the GPPORCR (general-purpose POR configuration register), provide a useful toolset for debugging hardware initialization.



**Figure 16. POR Configuration Cycles**

## 5.2 System Reset Differences

At reset, things are different on the PowerQUICC III when compared against the PowerQUICC II. First, the time at which the PowerQUICC III is brought out of reset can either be controlled in the normal manner i.e. during initial power-up, or it can be controlled as an agent from either the PCI or RapidIO interface.

Unlike the 603e core on the PowerQUICC II, the e500 core does not have a reset vector. Instead, the e500 core begins its execution at a fixed virtual address in its memory map at location 0xFFFFFFF0C which should be a branch instruction to the beginning address of the page mapped through TLB1. Although the e500 core begins its execution in virtual mode as opposed to the real mode of the 603e core, the MMUs can still be set up to provide real mode addressing where the effective address = the physical address.

(Refer back to Section 2.3.5 for information on how this hardware initialized TLB can be set up. Initialization code in this page should set up more valid TLB entries (and pages) so that the program can actually branch out of this 4-Kbyte page into other pages for booting the operating system. Similarly, the interrupt vector area and pages that contain the interrupt handlers should be set up so that exceptions can be handled early).

## 5.3 Boot Sequencer

On the PowerQUICC III, the boot sequencer is part of the I2C block and is used (before any boot code is loaded), to change the default values of internal or external registers in any memory mapped address. However, in most designs, this block is used to pre-program the PowerQUICC III to boot directly from memory devices such as burstable Flash, DDR or L2 cache (when configured as L2 SRAM). (On the PowerQUICC II, the processor could only boot devices that were attached to the 60x bus, using chip select 0 (CS0).

This block also contains a programmable clock divider which allows users to vary its speed anywhere from 160 to 32768 CCB cycles. When this block is running in boot sequencer mode (which is determined during the initialization procedure via the LGPL3/5 inputs), any register in external memory can be modified by setting the alternate configuration space (ACS) bit and supplying a register's memory address which is then pointed to by the ALTCBAR register. It is possible, by combining the base address in the ALTCBAR with that of the 20 bits of address offset supplied from the serial ROM to generate a 32-bit address that is mapped to the target specified in ALTCAR. Thus, by configuring these registers, the boot sequencer has access to the entire memory map, one 1-Mbyte block at a time.

## 6 Software Considerations

The e500 core contains some differences when compared with the 603e core used in other PowerPC embedded devices. These changes are mainly concerned with supervisor mode resources. Users will need to make small changes to initialization code when moving from the 603e core to the e500 core but their main application code should not need to be modified. Most of this supervisor related code has already been addressed by Motorola's software third parties and often simply re-compiling with an e500 compliant compiler is all that is required.

The main areas where users will need to make changes are around the exception handling, memory subsystem and MMU set up differences of the core. Most user mode software written on the 603e core can run unchanged on the e500. The only differences surround string instructions and APU specific instructions. Generally, porting code to the new e500 core will not involve a major amount of effort.

One of the main differences in running application code on the e500 core is that string instructions are no longer supported, namely: Load String word Immediate (**lswi**), Load String Word Indexed (**lswx**), Store String Word Immediate (**stswi**) and Store String Word indexed (**stswx**). The operating systems of most of the third party software vendors for the PowerQUICC III provide trap emulation when a string instruction is encountered in compiled code which handles this problem for the end user.

## 6.1 Exception Handling

The e500 core uses interrupt prefixing and offset registers to set each individual exception vectors. When migrating code to the e500 core from the 603e core, each exception can be set to the same vector offsets as on 603e to ensure compatibility.

Machine check handlers also need to be re-written as the e500 core uses a new return from machine check instruction (RFMCI) and the two machine check save and restore registers called MCSRR0 and MCSRR1. Normally, the machine check interrupt handler routine should be re-written for the e500 core so that it exits with this new **rfmci** instruction.

Some special purpose registers (SPRs) for acquiring exception information are also different with most of this information being combined into the new e500 exception syndrome register (ESR). A machine check syndrome register (MCSR) is used to log the cause of the machine check, with the ESR being used to determine the status. Software can easily overcome this by pointing the e500 exception vectors to a different address from the 603e vector addresses so that each vector can gather the relevant system information before jumping to the common (e500/603e) code. On the 603e core, the data storage interrupt status register (DSISR) was used for this function.

Timer exceptions are compatible between the e500 and 603e core except that the decremter does not count pass 0 on the e500. The e500 core also has a new decremter auto reload feature which can be disabled in order to be compatible with the 603e core.

## 6.2 e500 ABI

In order to make it easy to link existing 603e (or 'classic' PowerPC) object files, an e500 ABI (application binary interface) has been created and is used by all the leading third parties implementing e500-based tools. (This new ABI was required since there was no support on the 603e ABI for floating point registers, 64-bit SPE registers or embedded support). The e500 ABI specifies low-level system information such as stack usage and calling conventions as well as object file formats, linking, support of standard libraries and program loading.

## 6.3 SPE APU—Floating Point Support

The e500 has floating point capability using the SPE (signal processing engine) APU (auxiliary processing unit). This is an embedded friendly version and makes use of the 64-bit Simple and Multiple Execution Units as well as the 64-bit Load/Store unit to perform all of the necessary arithmetic manipulations with single precision. Users should note that the PowerQUICC III implements floating point operation using general-purpose registers (GPRs) instead of using floating point registers (FPRs) or vector registers (VRs). Refer back to Figure 2 for detailed information on the e500 core complex.

The SPE APU and SPFP (signal processing floating point) functionality is implemented on the MPC8540, the MPC8560 and all future derivatives of PowerQUICC III devices. However, these floating point instructions will not be supported in devices subsequent to PowerQUICC III. It is strongly recommended that use of these floating point instructions is confined to libraries and device drivers. Basically, when using SPE floating point instructions, users should not implement them using assembly code instructions to do this but rather include the libmoto library file. In doing this, customers protect future software legacy by simply referring to a single file rather than complete code rewriting on devices subsequent to the PowerQUICC III.

The libmoto library is a series of hand-optimized library files specifically designed to get 'optimal' performance when running frequently used software routines in different applications. For example, if we

## Summary

compare the Dhrystone benchmark for a standard optimized library then typically a value of 1.693 MIPS/MHz is obtained. Using the libmoto\_e500 this performance can be considerably increased to 2.507 MIPS/MHz.

## 6.4 Migrating Between ADS platforms

Customers using the PowerQUICC I and II should already be familiar with the Application Development Systems (or ADS) that is used for software/system bring-up. The PowerQUICC III continues this tradition with the PQ3 ADS-Pilot board. This board enables all of the features of the PQ3 to be tested, for example, TSEC, RapidIO, PCI, DDR, RS-232, and 10/100BaseT Ethernet.

We have already examined some of the considerations that must be taken into account when migrating software from the PQ2 to the PQ3. In this section we will look at the necessary switch/jumper settings that have to be made when testing code originally running on the PQ2ADS, onto a PQ3ADS-Pilot.

The PQ3ADS-Pilot board has a number of switches and jumpers in order to ensure maximum user configurability. Please refer to the PowerQUICC III Application Development Systems Quick Reference Guide supplied with your ADS board for full details and positioning of each switch.

## 6.5 Fast Ethernet Configuration

In order to enable Fast Ethernet (100BaseT) MII, legacy mode, the following jumpers should be set:

- SW4 [bit 5] Enable FETH2. Set this bit to '1' position.
- SW4 [bit 6] Enable FETH3. Set this bit to '1' position.
- Jumper 22 FETH2. Select 'Legacy mode'. Position jumper between terminals 1 & 2.
- Jumper 23 FETH3. Select 'Legacy mode'. Position jumper between terminals 1 & 2.
- Jumper 31 FETH2. Select 'MII mode'. Position jumper between terminals 1 & 2.
- Jumper 22 FETH3. Select 'MII mode'. Position jumper between terminals 1 & 2.

Each of the two fast Ethernet PHYs have an associated PHY address that must be defined in order to correctly read and write to the MII. The respective PHY addresses for FETH2 and FETH3 are 0x02 and 0x03. In order to read and write to the MII, these addresses need to be set in the 5-bit PHY address field (bits 19–23) of the MII management address register (MIIMADD).

The fast Ethernet PHYs on the ADS-Pilot board use the Davicom DM9161 10/100 Mbps physical layer transceiver chip. In order to allow the transceiver to auto negotiate properly, the following DM9161 registers still need to be set:

reg 16. = 0x0610

Aux. config. set (De)Scrambler for normal operation.the (de)scrambler is configured to "bypass mode" by the power-on reset latch

reg 00. = 0x1200

Control register autonegotiation enable/restart autonegotiation

## 7 Summary

This document has looked at the different aspects that existing PowerQUICC II users must consider when migrating from PowerQUICC II to PowerQUICC III. Although a number of new system blocks, as well as a new processor core, have been added onto the PowerQUICC III, we have hopefully shown that much of



## Summary

the existing PowerQUICC II hardware knowledge and software legacy equally applies to the PowerQUICC III. Whilst most of the new system blocks may seem a little daunting, users should take comfort in knowing that the PowerQUICC III is continuing the PowerQUICC family tradition of delivering high performance, next generation solutions for tomorrows leading edge applications.

THIS PAGE INTENTIONALLY LEFT BLANK

THIS PAGE INTENTIONALLY LEFT BLANK

# Freescale Semiconductor, Inc.

## HOW TO REACH US:

### USA/EUROPE/LOCATIONS NOT LISTED:

Motorola Literature Distribution  
P.O. Box 5405, Denver, Colorado 80217  
1-480-768-2130  
(800) 521-6274

### JAPAN:

Motorola Japan Ltd.  
SPS, Technical Information Center  
3-20-1, Minami-Azabu Minato-ku  
Tokyo 106-8573 Japan  
81-3-3440-3569

### ASIA/PACIFIC:

Motorola Semiconductors H.K. Ltd.  
Silicon Harbour Centre, 2 Dai King Street  
Tai Po Industrial Estate, Tai Po, N.T., Hong Kong  
852-26668334

### TECHNICAL INFORMATION CENTER:

(800) 521-6274

### HOME PAGE:

[www.motorola.com/semiconductors](http://www.motorola.com/semiconductors)

Information in this document is provided solely to enable system and software implementers to use Motorola 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.

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

Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola 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 which may be provided in Motorola 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. Motorola does not convey any license under its patent rights nor the rights of others. Motorola 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 Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola 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 Motorola was negligent regarding the design or manufacture of the part.



Motorola and the Stylized M Logo are registered in the U.S. Patent and Trademark Office. digital dna is a trademark of Motorola, Inc. The described product contains a PowerPC processor core. The PowerPC name is a trademark of IBM Corp. and used under license. All other product or service names are the property of their respective owners. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

© Motorola, Inc. 2004

AN2662

**For More Information On This Product,  
Go to: [www.freescale.com](http://www.freescale.com)**