NXP Semiconductors
Application Note

Document Number: AN5079
Rev. 0, 07/2017

QorIQ P1 Series to T1 Series Migration Guide

Contents

1 About this document

This document provides a summary of the significant differences between QorIQ P1 series and T1 series devices. The QorIQ P1 series devices discussed in this document are P1010, P1020, and P1022. These are compared with QorIQ T1 series devices including T1024, T1014, T1023, T1013, T1040, T1020, T1042, and T1022. Use this document as a recommended resource for migrating products from P1 series to T1 series devices.

2 Introduction

QorIQ P1 series devices combine single or dual e500v2 Power Architecture® core offering excellent combinations of protocol and interface support including DDR2/DDR3/3L memory controller, enhanced three-speed Ethernet controllers (eTSEC), USB 2.0, and PCI Express Gen 2 controllers. However T1 series devices have several enhancements over P1 series devices. As Table 1 shows, these enhancements include:

- Single, dual or quad 64-bit e5500 cores
- DDR3L/DDR4 SDRAM memory controllers
- High performance Data Path Acceleration Architecture (DPAA) to offload packet parsing, classification, traffic management, and quality of service.

- Integrated 8-port gigabit Ethernet switch (T1040/T1020 only)
- Integrated Flash Controller (IFC)
- Support of single source clocking

3 Feature-set comparison

This table shows the chip feature comparison between P1 series and T1 series devices.

Table 1. Detailed chip feature comparison

Features	P1010	P1020	P1022	T1024 / T1014 T1023 / T1013	T1040 / T1020 T1042 / T1022
Core					
PowerPC Core	2 x e500v2 1 x e500v2 (P1010)			2 x e5500 (T1024 / T1023) 1 x e5500 (T1014 / T1013)	4 x e5500 (T1040 / T1042) 2 x e5500 (T1020 / T1022)
Power ISA	32-bit			64-bit	
Core max. frequency (MHz)	1000	800	1067	1400	
Cache					
L1 cache	32 KB I/D per core				
L2 cache	256 KB Platform cache			256 KB Backside cache	
L3 cache	-			256 KB Platform cache	
DDR-SDRAM memory controller					
DDR Type	DDR3 / DDR3L	DDR2 / DDR3		DDR3L / DDR4	
DDR speed	up to 800 MT/s	up to 1333 MT/s		up to 1600 MT/s	
DDR width	16b w/ ECC or 32b w/o ECC	32b w/ ECC	64b w/ ECC	64b w/ ECC 32b w/ 4b-ECC (T1023/ T1013)	
Max memory size	16 GB	8 GB (DDR3) 4 GB (DDR2)	32 GB (DDR3) 16 GB (DDR2)	64 GB (DDR4) 32 GB (DDR3L)	
Ethernet					
Ethernet controller	eTSEC			DPAA	
MAC	3	3	2	4	5
Ethernet Mgmt interface	Clause 22 (through eTSEC)			Clause 22 (EMI1) and Clause 45 (EMI2)	Clause 22 (EMI1)
IEEE 1588	Yes			Yes	
High-speed interfaces					
SerDes Lanes	6 lanes	4 lanes	6 lanes	4 lanes	8 lanes
10GbE XFI	-			1	-
2.5G SGMII	No			Yes	
QSGMII	No			Yes	

Table continues on the next page...

Table 1. Detailed chip feature comparison (continued)

Features	P1010	P1020	P1022	T1024 / T1014 T1023 / T1013	T1040 / T1020 T1042 / T1022
1G SGMII	Yes			Yes	
RGMII	Yes			Yes	
RMII	No	Yes		No	
MII	No	Yes	No	No	
Ethernet Switch	-			-	8 ports (T10x0)
Backplane support	-			1000Base-KX 10GBase-KR	1000Base-KX
SATA	2 (Gen 2, 3Gbps)	-	2 (Gen 2, 3Gbps)	1 (Gen 2, 3Gbps)	2 (Gen 2, 3Gbps)
PCI Express	2 (Gen 1)		3 (Gen 1)	3 (Gen 2)	4 (Gen 2)
Hardware accelerators					
Security engine	SEC 4.x	SEC 3.x		SEC 5.x	
MACSEC	No			all ports	
Pattern Matching Engine	No			Yes	
Other interfaces					
Flash Controller	IFC	enhanced Local Bus Controller (eLBC)		Integrated Flash Controller (IFC)	
USB	ULPI or USB 2.0 w/ on-chip PHY	2 ULPI (USB 2.0)		2 (USB 2.0 w/ on-chip PHY)	
SDHC	MMC 4.2 and SD 2.0			MMC 4.5 and SD 3.0	
TDM	Yes			Supported through QE-TDM (T10x4 only)	Yes
Power management					
Deep sleep	-	-	Supported	Supported (T10x4 only)	Supported
Display controller					
Display Interface Unit	-		24-bit	12-bit Dual data rate (T10x4 only)	12-bit Dual data rate
Clocking					
Single source clocking	No			Yes (DIFF_SYSCLK/ DIFF_SYSCLK_B)	
Technology					
Technology	45 nm			28 nm	
Package					
Package	19mm x 19mm, 425-pin WB-TePBGA I	31mm x 31mm, 689-pin WB-TePBGA		23mm x 23mm, 780-pin FC-PBGA 19mm x 19mm, 525-pin FC-PBGA (T10x3)	

4 PowerPC e500v2 vs e5500 core

QorIQ P1 series devices use a e500v2 core, whereas T1 series devices use a e5500 core.

This table lists the similarities and differences between e500v2 and e5500 cores.

Table 2. Similarities and differences between e500v2 and e5500

Feature	e500v2	e5500	Comments
Architecture	32 bit Power ISA 2.06	64 bit Power ISA 2.06	e5500 provides 64-bit mode and several instructions which are category 64-bit
Instruction and data cache	32 KB	32 KB	Same in both architecture
L1 Instruction MMU	4 entry L1-VSP	8 entry L1-VSP	Enhanced throughput
L1 Data MMU	4 entry L1	8 entry L1-VSP	Enhanced throughput
Number of variable size TLB entries	16	64	e5500 contains a larger number of variable size TLB entries and larger number of available page sizes
Cache block size	32 byte	64 byte	e5500 contains a larger cache block/coherence granule size.
Backside L2 Cache w/ Tag Parity and Data ECC	Not present	256-512 KB	Lower latency than front-side cache without offering any traffic on interconnect network. Larger area more suitable to control plane applications and larger working sets of data.
Performance improvements	-	Over 20% increase in performance (DMIPS/ MHz)	Improved design and processes resulted in higher frequency operation increasing the size marginally.
SPRs	All SPRs 32 bit	LR, DEAR, xSRR0, IACx, DACx, MAS2 , CTR are 64 bit	Register width increased to hold the effective address
Instructions for doubleword	Not present	Additional Instructions for doubleword load, store, compare, shift and rotate	Necessary for 64 bit operation
Integer and logical instructions	32 bit	64 bit	Necessary for 64 bit operation
FPR based floating point	Carried out using SPE instruction	Present and is fully pipelined	Significant performance improvement over e500v2 at same frequency (2X faster on single precision, 4X faster on double precision). Additional instructions that perform convert to/ from integer doubleword to floating point.
SPE and embedded floating point	Present	Not present	SPE and embedded floating point (floating point done in the GPRs) is not present in e5500.
Embedded hypervisor	Not present	Present	A new privilege level and associated instructions and registers are provided in e5500 to support partitioning and virtualization.
External proxy for interrupts controller	Not present	Present	External proxy is a mechanism which allows the core to acknowledge an external input interrupt from the PIC when the interrupt is taken and provide the interrupt vector in a core register.

Table continues on the next page...

Table 2. Similarities and differences between e500v2 and e5500 (continued)

Feature	e500v2	e5500	Comments
Additional level for debug Interrupts	Not present	Present	A separate interrupt level for debug interrupts is provided and the associated save /restore registers DSRR0/DSRR1. This gives user an advantage of prioritizing debug over normal interrupts, which can be proved to be an advantage in many cases.
Processor signaling (door bell instructions)	Not present	Present	The **msgsnd** and **msgclr** instructions are provided to perform topology independent core to core doorbell interrupts. In e500v2 the door-bell interrupts was implemented using PIC which required addition instruction and thereby offer associated latencies
Fixed point	Not Present	Present	Additional instructions are provided in e5500 to access a true single-precision representation in storage, and a fixed-point integer representation in GPRs
Decorated storage	Not preset	Present	Instructions are provided for performing load and store operations to devices that include meta data that is interpreted by the target address. Devices in some SoCs use this facility for performing atomic memory updates like increments and decrements.
Lightweight synchronization	Not present	Adds the **lwsync** instruction	**msync** creates a barrier such that all (regardless of WIMGE attributes) memory accesses that occur before the sync are performed before any accesses after the sync.The **lwsync** instruction is provided for a faster form of memory barrier for load/store ordering to memory that is cached and coherent (WIMGE=0b001xx).
Interconnect bus	Uses Core Complex Bus (CCB) as an interconnect	Uses CoreNet as an interconnect bus	CoreNet is a scalable, non-retry-based fabric used as an interconnect between cores and other devices in the SoC.
Cache stashing	Not present	Present	The capability to have certain SoC devices stash or pre-load data into a designated core L1 or L2 data cache (private to cores) is provided. The core is a passive recipient of such requests.
Machine check	Provides machine check interrupt and HID0[RFXE] to control how the core treats machine check interrupts	Provides error report, asynchronous machine check, and NMI interrupts. HID0[RFXE] is removed	Machine check interrupts are divided into synchronous error reports, asynchronous machine checks, and NMI. How errors are reported are more conducive to a multi-core environment.
Write shadow	Not present	Present	The capability to have all data written to the L1 data cache be written through to the L2 cache (or to memory) is provided. This provides a method of ensuring that any L1 cache error can be recovered from without loss of data.

Table continues on the next page...

Table 2. Similarities and differences between e500v2 and e5500 (continued)

Feature	e500v2	e5500	Comments
Power management	uses MSR[WE] and HID0[DOZE,NAP,SLEEP] to enter power management states	uses SoC programming model to control power management and removes MSR[WE], HID0[DOZE,NAP,SLEEP]. Also adds the wait instruction.	Power management functions are invoked is now mostly controlled by writing SoC registers instead of Core.

The e5500 core supports both 32-bit and 64-bit modes, which is dynamically switchable. The e5500 core is software compatible with the e500v2 core. This compatibility is critical for applications having a large amount of legacy code. The 32-bit mode of e5500 core makes designs possible to seamlessly migrate from previous e500-based products to e5500-based products while taking advantage of the higher frequency and floating point improvements. It also provides a compelling product roadmap for 64-bit processing for next-generation solutions. This is because the design can be started in 32-bit mode and later selected to move to 64-bit mode.

The e5500 core has the following enhancements in its instruction set.

Table 3. Instruction set enhancements in e5500

Instruction	Definition
Doubleword load instructions	
`ld`	Load doubleword
`ldx`	Load doubleword indexed
`ldu`	Load doubleword with update
`ldux`	Load doubleword with update indexed
`ldarx`	Load doubleword and reserved indexed (use with `stdcx.`)
`lddx`	Load doubleword decorated indexed
`ldepx`	Load doubleword by external PID indexed
`ldbrx`	Load doubleword byte-reversed indexed
Doubleword store instructions	
`std`	Store doubleword
`stdx`	Store doubleword indexed
`stdu`	Store doubleword with update
`stdux`	Store doubleword with update indexed
`stdcx.`	Store doubleword conditional indexed
`stddx`	Store doubleword with decoration indexed
`stdepx`	Store doubleword by external PID indexed
`stdbrx`	Store doubleword byte-reversed indexed
Word load instructions	
`lwa`	Load word algebraic
`lwx`	Load word arithmetic indexed
`lwaux`	Load word algebraic with update indexed (sign extends)
Shift instructions	

Table continues on the next page...

Table 3. Instruction set enhancements in e5500 (continued)

Instruction	Definition
`srd[.]`	Shift right doubleword
`srad[.]`	Shift right algebraic doubleword
`sradi[.]`	Shift right algebraic doubleword immediate
`sld[.]`	Shift left doubleword
Rotate instructions	
`rldcl[.]`	Rotate left doubleword then clear left
`rldcr[.]`	Rotate left doubleword then clear right
`rldic[.]`	Rotate left doubleword immediate then clear
`rldicl[.]`	Rotate left doubleword immediate then clear left
`rldicr[.]`	Rotate left doubleword immediate then clear right
`rldimi[.]`	Rotate left doubleword immediate then mask insert
NOTE: Existing word shift and rotate instructions copy the lower 32-bits to the upper 32-bits and perform the operation on the 32-bit pairs	
Multiply instructions	
`mulld[.]`	Multiply low doubleword
`mulldo[.]`	Multiply low doubleword with overflow detection
`mulhd[.]`	Multiply high doubleword
`mulhdu[.]`	Multiply high doubleword unsigned
Divide instructions:	
`divd[.]`	Divide doubleword
`divdo[.]`	Divide doubleword with overflow detection
`divdu[.]`	Divide doubleword unsigned
`divduo[.]`	Divide doubleword unsigned with overflow detection
Compare, trap doubleword instructions:	
`cmp L=1`	Compares 64-bits
`td`	Trap doubleword
`tdi`	Trap doubleword immediate
Compare bytes instructions	
`cmpb`	Can compare up to 8 bytes in a GPR
Population count, leading zeros, extend sign instructions	
`popcntb`	Population count byte. Computes the number of bits set in each byte
`popcntw`	Population count word. Computes the number of bits set in each word
`popcntd`	Population count doubleword. Computes the number of bits set in each doubleword of a GPR
`cntlzd[.]`	Count leading zeros doubleword
`extsw[.]`	Extend sign word
Bit permute and parity instructions	
`bpermd`	Permutes bits selected by a control GPR from bytes in a source GPR producing an 8-bit result
`prtyw`	Exclusive or the least significant bits of each byte in the word in a GPR producing a 1-bit result for each word

Table continues on the next page...

Table 3. Instruction set enhancements in e5500 (continued)

Instruction	Definition
`prtyd`	Exclusive or the least significant bits of each byte in the doubleword in a GPR producing a 1-bit result for each word
NOTE: For more details, refer to the document, EREF: Programmer's Reference Manual for Freescale Power Architecture Processors.	

5 Ethernet controller

This section describes the Ethernet controller used in P1 series and T1 series devices.

P1 series has enhanced three-speed Ethernet controllers (eTSEC). The eTSECs incorporate a media access control (MAC) sublayer that supports 10/100 Mbps and 1 Gbps Ethernet/IEEE 802.3 networks with MII, RMII, SGMII, and RGMII physical interfaces.

T1 series has data path accelerator architecture (DPAA) that enables Ethernet with multi-rate Ethernet MAC (mEMAC) interfaces to 10 Gbps and below Ethernet/IEEE 802.3 networks by means of XFI/ SGMII/ QSGMII using either the high-speed SerDes interfaces or RGMII parallel interface.

5.1 Overview

5.1.1 DPAA overview

DPAA is a comprehensive architecture implemented on T1 series multicore SoCs. The DPAA architecture provides the infrastructure to support simplified sharing of networking interfaces and accelerators to multiple CPUs.

The components of DPAA can be categorized as:

- Network and packet I/O
  - FMAN- 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
- Infrastructure components
  - BMAN- Supports hardware buffer management for buffer allocation and deallocation
  - QMAN- Supports queue management for scheduling, packet sequencing, and congestion management
- Hardware accelerators
  - Cryptographic accelerator, SEC
  - Pattern Matching Engine (PME)
  - MACSEC on DPAA-based Ethernet ports

The following figure shows the block diagram of QorIQ T1 series DPAA.

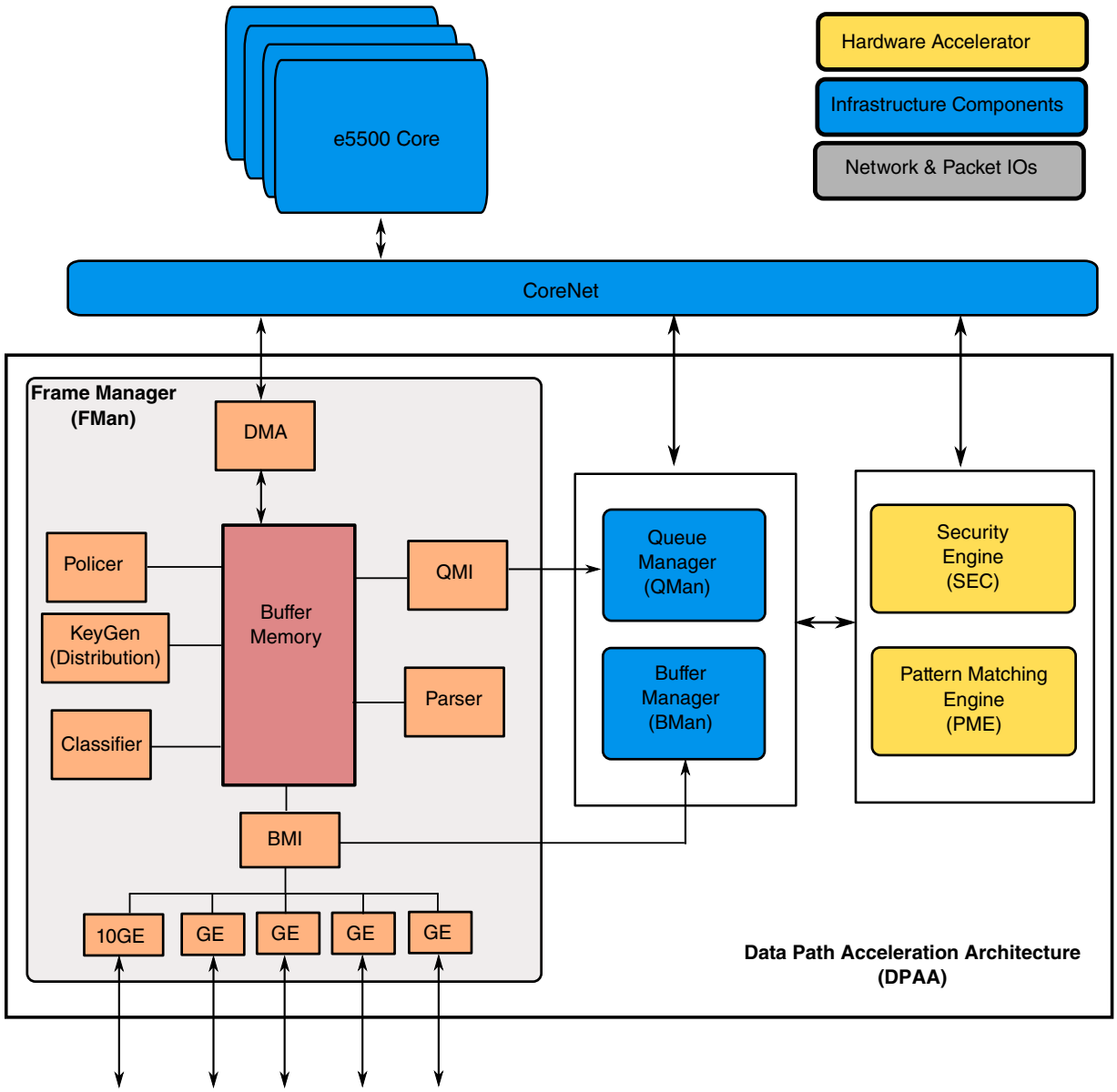

Figure 1. DPAA block diagram

5.1.2 eTSEC overview

The eTSEC uses a software model comprised of a combination of control and status registers (CSRs) as well as buffer descriptors.

eTSEC is organized as shown in the figure below.

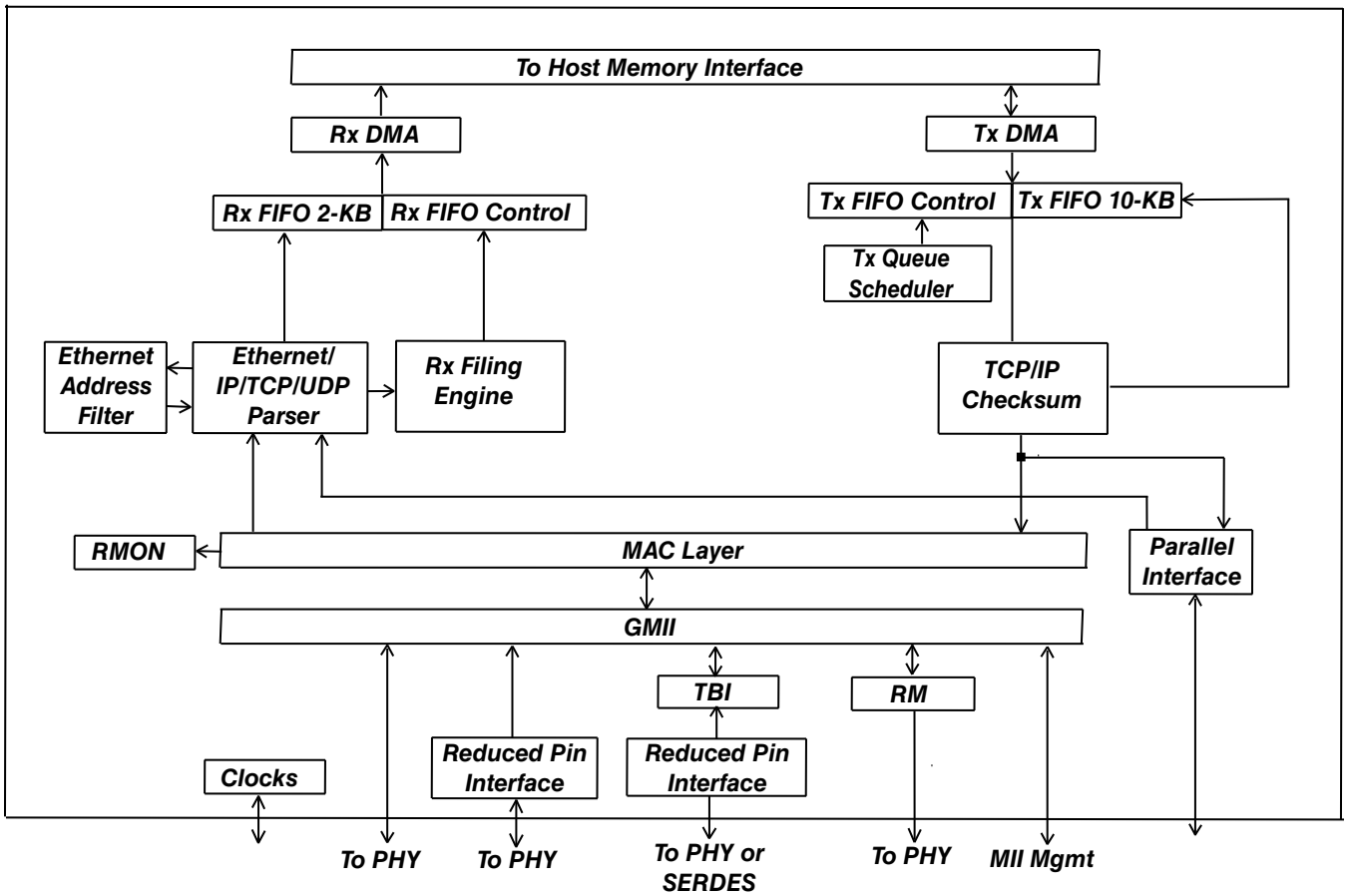

Figure 2. eTSEC block diagram

5.2 Ethernet features comparison

This table shows the Ethernet features comparison between P1 series and T1 series devices.

Table 4. Ethernet features comparison

Ethernet feature	P1 series	T1 series
QoS - Packet classification	Up to 64 virtual queues	Up to 1024 virtual queues
MAC filtering	Yes	Yes
Flow control	Yes	Yes
1588 timestamping	Yes	Yes
Packet statistics	Yes	Yes
Header parsing	Yes	Yes
Clause 22 MII management	Yes	Yes
MII interface	Yes	Yes
RMII interface	Yes	Yes
RGMII interface	Yes	Yes
SGMII interface	Yes	Yes

Table continues on the next page...

Table 4. Ethernet features comparison (continued)

Ethernet feature	P1 series	T1 series
Ethernet pause frame (802.3 Annex 31A) termination	Yes	Yes
CRC-32 checking with optional forwarding of the FCS	Not optional	Yes
CRC-32 generation and append on transmit	Yes	Yes
VLAN tagged frames	Yes	Yes
Magic packet detection	Yes	Yes
Packet parsing at wire speed	Yes	Yes
Packet classification	Yes	Yes
Packet distribution	Yes	Yes
Cryptography acceleration (SEC 5.4)	Yes	Yes
MACSEC	No	Yes
HW buffer management for buffer allocation and deallocation	No	Yes
User defined protocols	No	Yes
Priority flow control (PFC)	No	Yes
Policing - RFC4115 and RFC2968	No	Yes
CEETM	No	Yes
Clause 45 MII management	No	Yes
QSGMII interface	No	Yes
XFI interface	No	Yes
IEEE802.3az (EEE [Energy Efficient Ethernet])	No	Yes
Deficit idle counter (DIC)	No	Yes
Port virtualization	No	Yes
Storage profile selection	No	Yes
Supports drop on tail-drop/wred	No	Yes

5.3 Ethernet packet flow

5.3.1 Ethernet packet flow in DPAA

Frame Manager (FMAN) is the key component of Ethernet packet flow and operates in conjunction with Queue Manager (QMAN), to enqueue frames, and Buffer Manager (BMAN), to allocate and deallocate buffers.

The following figure models one of the basic examples of ingress packet flow for DPAA. The packet flow can be configured in different ways to achieve particular function. Refer to the respective device DPAA reference manual for more details.

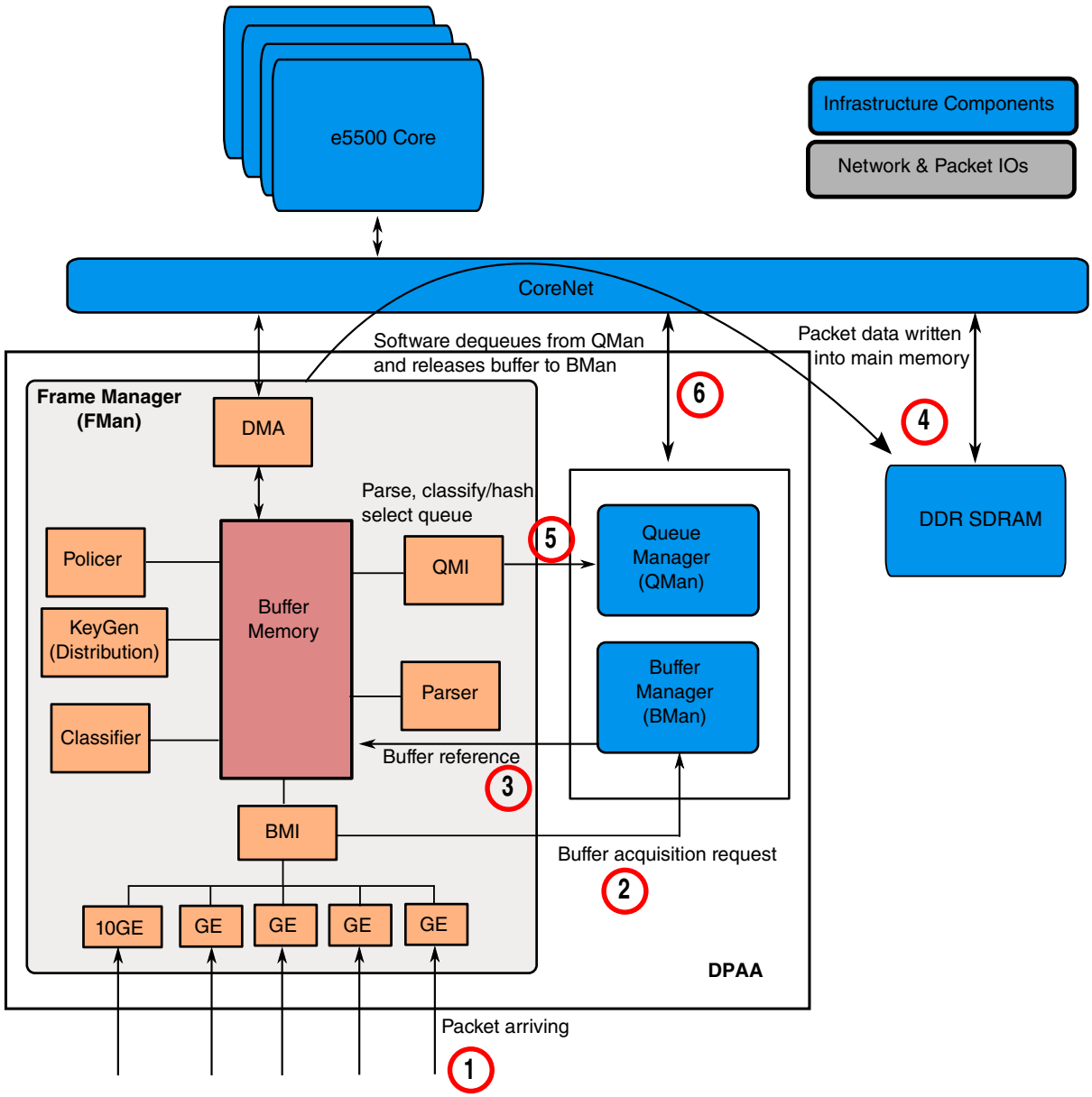

Figure 3. DPAA ingress packet flow

1. The packet arrives at a network interface and enters the Ethernet MAC (mEMAC) within FMAN.
2. After FMAN receives the Ethernet frame, it requests one or more buffers from the hardware buffer manager (BMAN) to store the frame. BMAN maintains pools of buffers, each with software-defined characteristics.
3. FMAN is initialized to request a buffer from the most appropriate pool. If a sufficiently large buffer cannot be found for the incoming frame, FMAN stores the frame across several smaller buffers and creates a scatter/gather list for these buffers.
4. The Ethernet packet, referenced by BMAN, is written into the DDR memory.
5. FMAN's configurable parsing and filing capabilities perform an initial classification. The steering of a packet towards a processor or a group of processors could also be based on differentiating flows by means of the quality of service attributes of the flow.
6. QMAN appropriately schedules the queue for the frames. The core dequeues from QMAN.

This following figure models one of the basic examples of egress packet flow for DPAA.

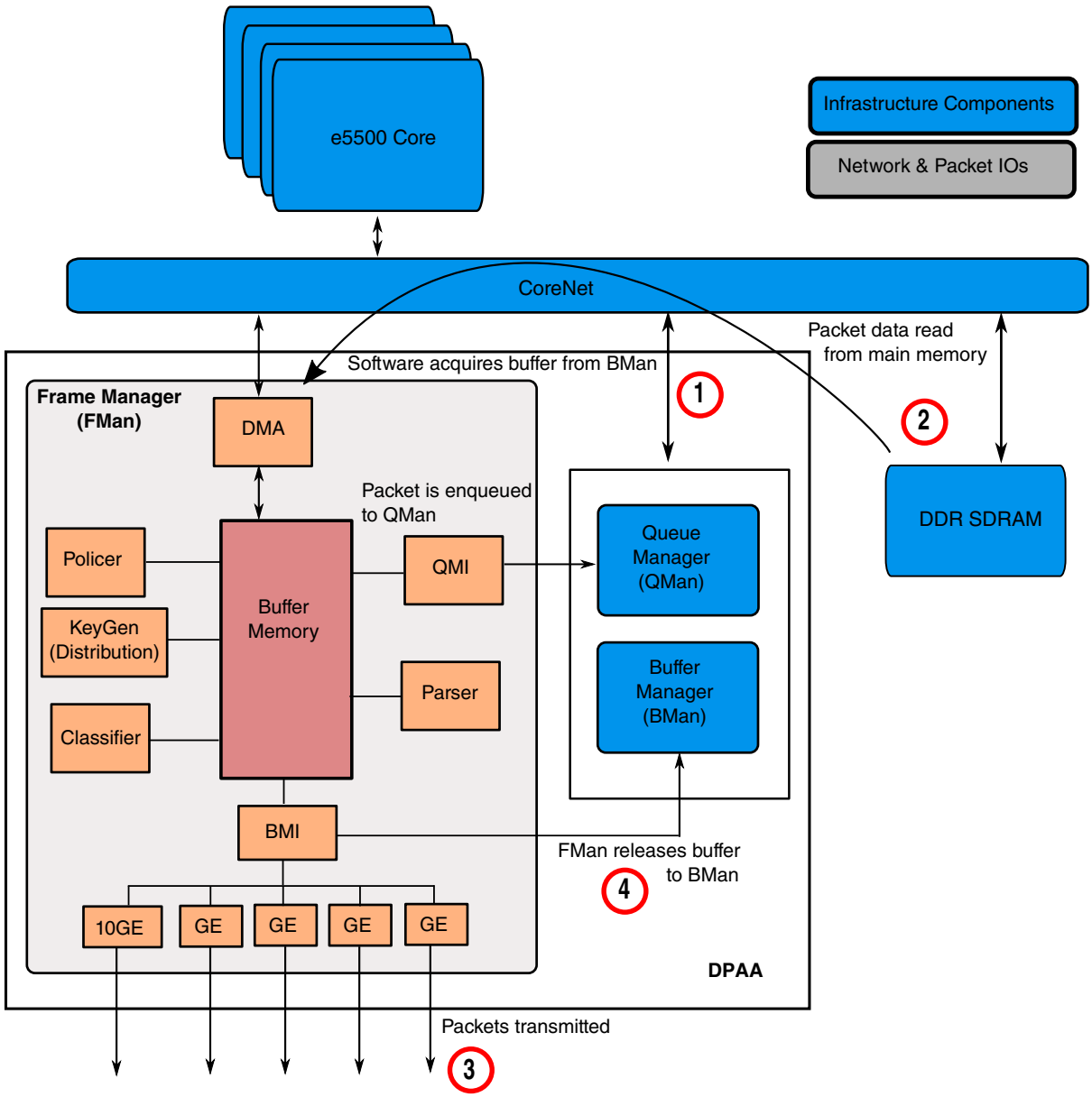

Figure 4. DPAA egress packet flow

1. The core acquires a buffer from BMAN and enqueues the Frame Descriptor (FD) to QMAN.
2. The packet data is read from DDR to FMAN based on the pointer from FD.
3. FMAN inserts checksum and transmits the frame to the interface through mEMAC.
4. FMAN releases the buffer to BMAN acquired at the step 1.

5.3.2 Ethernet packet flow in eTSEC

The eTSEC packet processing requires a software model programmed by a combination of control and status registers (CSRs) and buffer descriptors. The CSRs are used for mode control, interrupts, and to extract status information. The descriptors are used to pass data buffers and related buffer status or frame information between the hardware and software.

The following figure shows the components of eTSEC and Ethernet packet flow.

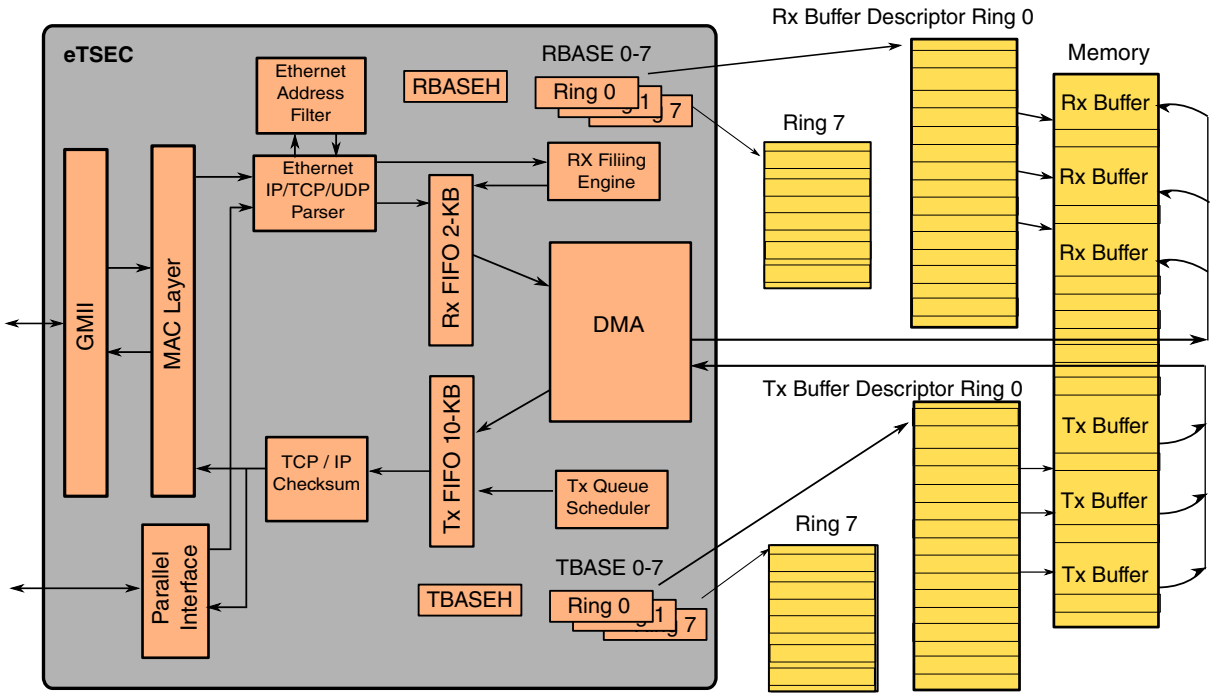

Figure 5. eTSEC general packet flow

5.4 Device driver

The DPAA-Ethernet device driver supported in T1 series devices use a different Linux device driver than eTSEC does.

The eTSECs supported in P1 series devices use a gianfar Linux device driver. The details for the gianfar driver can be found at:

http://cache.nxp.com/files/infocenter/QORIQSDK_INFOCTR.html

Go to >> Linux Kernel Drivers >> Ethernet >> Linux Ethernet for Enhanced Three Speed Ethernet Controller Family

Although DPAA is a powerful networking accelerator, NXP Linux SDK hides its complexity and allows the user to instantiate Ethernet interfaces in simpler ways. There are various use-cases that are discussed in more details at:

http://cache.nxp.com/files/infocenter/QORIQSDK_INFOCTR.html

Go to >> Linux Kernel Drivers >> Ethernet >> Linux Ethernet Driver for DPAA 1.x Family

6 POR sequence

The table below summarizes the similarities and differences in the external reset signals between P1 and T1 series devices.

Table 5. Comparison of external reset signals

Signal	QorIQ P1 series		QorIQ T1 series	
	I/O	Description	I/O	Description
PORESET_B	N/A		I	Power on reset input
HRESET_B	I	Hard reset. Causes the device to abort all current internal and external transactions and set all registers to their default values.	I/O	Hard reset. Functions as an output during initial steps in the power on reset sequence. For reset assertion to the chip, use only PORESET_B.
HRESET_REQ_B	O	Hard reset request. Indicates to the board (system in which the device is embedded) that a condition requiring the assertion of HRESET_B has been detected.	O	Reset request output. An internal block requests that PORESET_B be asserted.
SRESET_B	I	Soft reset. Causes a machine check interrupt to the e500 core. SRESET_B need not be asserted during a hard reset.	N/A	

The following section highlights the similarities and differences in POR sequence between P1 and T1 series devices.

Table 6. Comparison of POR sequence

QorIQ P1 series	QorIQ T1 series
PORESET_B is absent in P1 series devices based system, instead HRESET_B signal does what PORESET_B and HRESET_B signals perform combined. Power is applied to comply with the chip's data sheet.	The external system logic asserts PORESET_B and power is applied to comply with the chip's data sheet.
HRESET_B is asserted resulting all registers to be initialized to their default states and most I/O drivers to be released to high impedance (some clock, clock enables, and system control signals are active). It is recommended that TRST_B is logically tied to PORESET_B in such a way that whenever TRST_B is asserted PORESET_B also appears to be asserted. This ensures that the JTAG scan chain is initialized during the power-on reset flow.	PORESET_B is asserted resulting all registers to be initialized to their default states and most I/O drivers to be released to high impedance (some clock, clock enables, and system control signals are active). It is recommended that TRST_B is logically tied to PORESET_B in such a way that whenever TRST_B is asserted PORESET_B is also appears to be asserted. This ensures that the JTAG scan chain is initialized during the power-on reset flow.
The system applies a stable SYSCLK signal and stable POR configuration inputs.	The system applies a stable SYSCLK signal and stable POR configuration inputs. At this point, SYSCLK is propagated throughout the device; the platform PLL is running in bypass mode
	The device begins driving HRESET_B asserted after sampling the assertion of PORESET_B.
External system logic negates HRESET_B after its required hold time and after POR configuration inputs have been valid for their required setup times.	External system logic negates PORESET_B after its required hold time and after POR configuration inputs have been valid for their required setup times.
The device samples POR config inputs on de-assertion of HRESET_B.	The device samples the RCW source POR configuration inputs (cfg_rcw_src[0:n]) on de-assertion of PORESET_B. Note that the POR configuration inputs are sampled only on a PORESET_B.
	The device "resets" rest of the platform logic.

Table continues on the next page...

Table 6. Comparison of POR sequence (continued)

QorIQ P1 series	QorIQ T1 series
	Note that this platform reset step is the point where the device hard reset process (HRESET_B) begins. In other words, if an external logic asserts HRESET_B, the device starts from here. Assumption is that device has gone through PORESET_B earlier.
	All the required I/O drivers are enabled which are necessary to read RCW data from the source specified in cfg_rcw_src[0:n]. All of the DDR I/Os become enabled at this point (though MCKE, MCK, MODT are enabled from the beginning). The ASLEEP signal is also enabled at this point.
	If the IFC's NAND Flash interface is configured as the RCW source, the reset block instructs the IFC to load a boot block from Flash into the internal buffer RAM of the IFC. When complete, the reset block proceeds to instruct the Pre-Boot Loader to begin reading in RCW data. Note that if the IFC NAND Flash interface reports an ECC error, the device reset sequence is halted indefinitely, waiting for another PORESET_B or hard reset.
	The pre-boot loader (PBL) starts loading the RCW data from the interface specified by cfg_rcw_src[0:n] configuration inputs and stores that 64 bytes of data to the RCWSR registers within the device configuration block. Loading time varies and depends on the source of the RCW. Note that if a hard-coded RCW option is used, this PBL RCW loading process is effectively bypassed and the device is automatically configured according to the specific RCW field encodings pre-assigned for the given hard-coded RCW option.
Platform/device and DDR PLLs begin to lock (CCB clock).	The first few RCW bits have PLL frequency information embedded. Therefore, PLLs begin to lock at the same time that the rest of the RCW is loaded.
	After entire RCW (512 bits) is read from specified source, PBL checks for errors. In case of errors, the reset sequence is halted indefinitely, waiting for another PORESET_B or hard-reset.
The CCB clock is cycled for approximately 100 µs to allow e500v2 core PLL to lock.	The platform clock tree is then switched over and is driven by the output of the platform PLL.
Internal hard reset to cores is negated. The device enables I/O drivers.	The device stops driving HRESET_B at this point. All other I/O drivers are enabled at this point.
If NAND Flash interface is: Configured as boot device (Refer to Boot ROM location - external inputs) The enhanced local bus FCM is released. When the FCM finishes loading the pages from the NAND Flash device, the boot sequencer, if enabled, is allowed to progress, causing it to load configuration data from serial ROMs on the I2C1 interface (Refer to Boot sequencer configuration in P1 series devices reference manual).	If the IFC's NAND Flash interface is: • configured as the pre-boot initialization source OR • the boot device target AND not fused as secure boot AND • the IFC's NAND Flash interface was NOT previously used as the RCW source, then the reset block informs the IFC to load a boot block from Flash into the internal buffer RAM of the IFC. When complete, the IFC signals back to the reset block, and the reset block

Table continues on the next page...

Table 6. Comparison of POR sequence (continued)

QorIQ P1 series	QorIQ T1 series
	can proceed. Note that if the IFC reports an ECC error, the device reset sequence is halted indefinitely, waiting for a hard reset or PORESET_B.
	The PBL performs pre-boot initialization (if enabled by RCW) by reading data from either the eSDHC, QSPI, IFC interface, and writing to CCSR space or local memory space (SRAM, DDR). If the PBL reports an error during its pre-boot initialization process, the device reset sequence is halted indefinitely, waiting for a hard reset or PORESET_B.
	Any external device optionally driving HRESET_B negates it if not done earlier. If other external devices do not release HRESET_B, the device reset sequence stalls at this point.
When the local bus FCM and boot sequencer complete, peripheral interfaces are released to accept external requests, and the boot vectors fetched by the e500 cores are allowed to proceed unless processor booting is further held off by POR configuration inputs as described in Boot sequencer configuration. The device is now in its ready state.	The peripheral interfaces are released to accept external requests, and the boot vector fetches by the cores are allowed to proceed unless processor booting is further held off by the boot release register (BRR) in the device configuration module. The device is now in its ready state.
The ASLEEP signal negates synchronized to a rising edge of SYSCLK, indicating the ready state of the system. After reaching this system ready state, the ASLEEP signal transitions to the asserted state when the device enters sleep mode. The ready state for the e500 core is also indicated by the assertion of READY_P0/TRIG_OUT if TOSR[SEL] = 000.	The ASLEEP signal negates synchronized to a rising edge of SYSCLK, indicating the ready state of the system. After reaching this system ready state, the ASLEEP signal transitions to the asserted state when the device enters sleep mode.

7 DDR memory controller

P1 series devices support DDR2, DDR3/3L memory controllers.

T1 series devices support DDR3L/DDR4 memory controllers. DDR2 memory is not supported on T1 series devices.

When designing DDR subsystem using P1 or T1, the following documents are recommended for respective DDR layout and register programming.

- AN2910, Hardware and Layout Design Considerations for DDR2 SDRAM Memory Interfaces
- AN4039, Hardware and Layout Design Considerations for DDR3 SDRAM Memory Interfaces
- AN5097, Hardware and Layout Design Considerations for DDR4 SDRAM Memory Interfaces
- AN3940, Layout Design Considerations for DDR3 Memory Interface
- AN3939, DDR Interleaving for PowerQUICC and QorIQ Processors

To achieve faster P1 and T1 board bring-up, use NXP's QorIQ Configuration and Validation Suite (QCVS) Tool. QCVS bundles several tools that include DDR Configuration and Validation Tool (DDRv), SerDes Configuration and Validation Tool, PreBootLoader Configuration, Hardware Device Tree Configuration, and Frame Distributor Wizard.

The DDRv tool helps find all the settings that work, display them in table, and allows the user to select between the best working settings.

More details on the DDRv tool can be found at http://www.nxp.com/ddrv

DDR controller in T1 devices can be configured as either DDR3L or DDR4 memory controller. Because DDR3L and DDR4 signals are different, the following table shows DDR3L/DDR4 memory controller signal mapping.

Table 7. DDR3L/DDR4 signal mapping

DDR3L	DDR4
MRAS	MRAS/**MA[16]**
MCAS	MCAS/**MA[15]**
MWE	MWE/**MA[14]**
MA[15]	ACT_n
MA[14]	BG1
MA[2]	BG0
MDM[0:8]	MDM/**DBI**
MAPAR_ERR	ALERT_n
MAPAR_OUT	PAR

8 Local bus - eLBC vs IFC

P1 series devices use enhanced local bus controller (eLBC). However, P1010 has integrated flash controller (IFC). P1010 is the only device in the P1 series family that has IFC.

All T1 series devices use IFC.

8.1 Comparison between eLBC and IFC

eLBC and IFC have different internal controllers and booting options. This table shows the differences between eLBC and IFC internal controllers.

Table 8. Internal controllers of eLBC and IFC

eLBC	IFC
eLBC has three controllers: 1. General purpose chip-select machine (GPCM) • Supports regular NOR flash for booting • Supports FPGA and SRAM 2. NAND flash control machine (FCM) • Supports NAND memory for storage and/or booting 3. User programmable machine (UPM) • Supports FPGA, ZBT RAM • Booting is not supported	**IFC has three controllers:** 1. NOR controller • Supports standard and page mode NOR flash • Supports booting 2. NAND flash control machine (FCM) • Supports NAND memory for storage • Supports booting 3. General purpose chip-select machine (GPCM) • Booting is not supported It operates in either of two modes: a. Normal GPCM mode supports legacy, standard NOR flash b. Generic ASIC mode supports FPGA

This table lists the differences between eLBC and IFC features.

Table 9. Comparison between eLBC and IFC

Machine	Features	eLBC	IFC
NAND	Devices max page size	2KB	8KB
	Error correction	1-bit /512 Bytes	4 bit, 8 bit/512 Byte 24 bit, 40 bit/1 KB
	Flexible timing control allows interfacing with proprietary NAND	Limited Capability	Yes
	Provide cache, copy-back and multi-plane command support	No	Yes
	Programmable command and data transfer sequences	Up to 8	Up to 15
	BBI page position	First two pages of each block	Configurable between second and last page
	Configurable block size constraint to multiple of 32 pages, up to 2048 pages	No	Yes
	Internal SRAM size	5 KB	9 KB
	Max initial boot code size for NAND flash	4 KB	8 KB
	SRAM access when NAND operation is on	Allowed	Not-allowed
NOR	Compatible with page mode NOR flash	No	Yes
	Support true-address/data-muxed devices	No	Yes
	Flexible timing control allows interfacing with variety of NOR devices	Limited capability	Yes
	Synchronous NOR support	No	No
Others	GPCM	Yes	Yes (with enhanced timing control)
	UPM	Yes	No
	GASIC	No	Yes

This table shows that signal name mapping between eLBC and IFC

Table 10. Signal name mapping between eLBC and IFC

Name	eLBC	IFC
Address/Data	LAD	IFC_AD
Address	LA	IFC_ADDR
Address valid	LALE/LFALE	IFC_AVD
Chip select	LCS	IFC_CS_B
Write enable	LWE_B	IFC_WE_B
Command latch enable	LFCLE	IFC_CLE
Output enable	LOE_B	IFC_OE_B
Write protect	LFWP_B	IFC_WP_B
Ready/Busy	LFRB_B	IFC_RB_B
Buffer control	LBCTL	IFC_BCTL
Parity	LDP[0:1]	IFC_PAR[0:1]

Table continues on the next page...

Table 10. Signal name mapping between eLBC and IFC (continued)

Name	eLBC	IFC
Parity error	Not available	IFC_PERR
Clock	LCLK	IFC_CLK

8.2 Device driver

U-Boot drivers for local bus interfaces are configured differently for devices supporting eLBC and IFC. U-Boot drivers for eLBC internal controllers are available in the NXP Linux SDK for QorIQ processors and can be found in the U-Boot folder:

- /drivers/mtd/nand/fsl_elbc_nand.c
- /drivers/mtd/nand/fsl_elbc_spl.c
- /drivers/mtd/nand/fsl_upm.c
- /arch/powerpc/cpu/mpc8xxx/fsl_lbc.c

IFC uses the following U-Boot drivers:

- /drivers/mtd/nand/fsl_ifc_nand.c
- /drivers/mtd/nand/fsl_ifc_spl.c
- /drivers/misc/fsl_ifc.c

NXP Linux SDK for QorIQ processors includes U-Boot and Linux kernel drivers for accessing NOR and NAND flash. The instructions to configure the Linux kernel driver are available within the NXP QorIQ SDK manual:

http://cache.nxp.com/files/infocenter/QORIQSDK_INFOCTR.html

- Go to QorIQ SDK >> Linux Kernel Drivers >> Flash Memory >> JFFS2 on NOR Flash Device Driver User Manual
- Go to QorIQ SDK >> Linux Kernel Drivers >> Flash Memory >> JFFS2 on NAND Flash Device Driver User Manual
- Go to QorIQ SDK >> Linux Kernel Drivers >> Flash Memory >> Integrated Flash Controller NOR Flash User Manual
- Go to QorIQ SDK >> Linux Kernel Drivers >> Flash Memory >> Integrated Flash Controller NAND Flash User Manual

9 USB

This section describes the USB controller used in P1 series and T1 series devices.

9.1 USB compatibility

USB controller in P1 and T1 series devices complies to USB specification Rev 2.0 supporting high-speed (480 Mbps), full-speed (12 Mbps), and low-speed (1.5 Mbps) operation.

P1020 and P1022 support ULPI (UTMI+ low pin interface) and require external PHY. P1010 and T1 series devices support USB 2.0 with on-chip PHY.

NOTE

On all P1 devices low speed (1.5 Mbps) mode is only supported in host mode.

9.2 Device driver

USB in P1 and T1 devices is enhanced host controller interface (EHCI) compatible and can be configured as host and gadget.

NXP provides U-Boot device drivers in the Linux SDK for QorIQ processors.

In U-Boot, the USB drivers can be found in the U-Boot folder:

- drivers/usb/host/ehci-fsl.c
- drivers/usb/host/ehci-generic.c
- drivers/usb/host/ehci-mpc512x.c
- drivers/usb/host/ehci.h

The NXP Linux SDK for QorIQ processors manual (available online at the NXP website) provides more details on USB device drivers:

http://cache.nxp.com/files/infocenter/QORIQSDK_INFOCTR.html

Go to >> Linux Kernel Drivers >> Universal Serial Bus Interfaces >> USB 2.0 Host Driver

Go to >> Linux Kernel Drivers >> Universal Serial Bus Interfaces >> USB Gadget Memory Driver User Manual

Go to >> Linux Kernel Drivers >> Universal Serial Bus Interfaces >> USB Gadget Network Driver User Manual

10 eSDHC

This section describes the SDHC support in P1 series and T1 series devices.

10.1 eSDHC compatibility

P1 series devices support the MMC 4.2 and SD 2.0 specification at maximum bus clock speed up to 52 MHz. The devices only support full-speed/high-speed mode.

T1 series devices support the MMC 4.5 and SD 3.0 specification. The maximum bus clock speed these devices support is up to 175 MHz. They support high-speed mode, UHS-1 speed modes such as SDR12, SDR25, SDR50, SDR104, DDR50, and MMC HS200 and MMC DDR mode. T1 devices also support advanced DMA (ADMA).

10.2 Device driver

NXP provides U-Boot device driver in the Linux SDK for QorIQ processors.

In U-Boot, the eSDHC driver can be found in the U-Boot folder:

- drivers/mmc/fsl_esdhc.c
- drivers/mmc/mmc.c
- drivers/mmc/mmc_write.c

The Linux kernel drivers in P1 and T1 share common eSDHC drivers. For more details see:

http://cache.nxp.com/files/infocenter/QORIQSDK_INFOCTR.html

- Go to QorIQ SDK >> Linux Kernel Drivers >> Enhanced Secured Digital Host Controller (eSDHC) >> eSDHC Driver User Manual

11 Related documentation

Refer to the following documents when considering QorIQ P1 and T1 series devices. Contact your NXP sales representative for access to any documents that are not publicly available on the NXP external website.

QorIQ P1 series device documents:

- *P1020 QorIQ Integrated Processor Reference Manual*
- *P1022 QorIQ Integrated Processor Reference Manual*
- *P1010 QorIQ Integrated Processor Reference Manual*
- *P1020 QorIQ Integrated Processor Hardware Specifications*
- *P1022 QorIQ Integrated Processor Hardware Specifications*
- *P1010 QorIQ Integrated Processor Hardware Specifications*
- *PowerPC™ e500 Core Family Reference Manual*

QorIQ T1 series device documents:

- *QorIQ T1024 Reference Manual*
- *QorIQ T1040 Reference Manual*
- *QorIQ T1024, T1014 Datasheet*
- *QorIQ T1023, T1013 Datasheet*
- *QorIQ T1040, T1020 Datasheet*
- *QorIQ T1042, T1022 Datasheet*
- *QorIQ T1024 Data Path Acceleration Architecture (DPAA) Reference Manual*
- *QorIQ T1040 Data Path Acceleration Architecture (DPAA) Reference Manual*
- *EREF: A Programmer's Reference Manual for Freescale Power Architecture Processors*

12 Revision history

This table summarizes changes to this document.

Table 11. Document revision history

Rev. number	Date	Substantive Change(s)
0	07/2017	• Initial public release

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number AN5079
Revision 0, 07/2017