



XLS[®] Processor Family Programming Reference Manual

XLS616, XLS608,
XLS408-Lite, XLS404-Lite,
XLS416, XLS408, XLS404,
XLS208, XLS204
XLS108, XLS104

| Revision Level 3.40

| NetLogic Confidential

| Document Number 10799V340PM-C
| Released June, 2010

Revision History

See [Appendix B, "Revision History"](#)

Related Documents

- MIPS™ Technologies, "MIPS64™ Architecture for Programmers"
 - Volume I: Introduction to the MIPS64 Architecture
 - Document Number: MD00083, Revision 2.00, June 8, 2003
 - Volume II: The MIPS64 Instruction Set
 - Document Number: MD00087, Revision 2.00, June 9, 2003
 - Volume III: The MIPS64 Privileged Resource Architecture
 - Document Number: MD00091, Revision 2.00, June 9, 2003
- XLS™ Processor Family Data Book

Copyright Information

Specifications subject to change without notice.

NetLogic Microsystems, the NetLogic Microsystems logo, XLS, XLS-based, Fast Messaging Network, and Autonomous Security Acceleration Engine are trademarks or registered trademarks of NetLogic Microsystems Inc. Other names and brands may be claimed as the property of others. All other trademarks are the property of their respective owners.

The information contained herein is the property of NetLogic Microsystems, Inc, and is believed to be accurate at the time of publication. NetLogic Microsystems, Inc, assumes no liability for any errors or omissions in this information, or for the use of this information or products described herein. NetLogic Microsystems, Inc, reserves the right to make changes to its products at any time to improve reliability, functionality, performance, or manufacturability. Disclosure of the information herein does not convey a license or any right in any patent, trademark, or other intellectual property of NetLogic Microsystems, Inc. Copyright © 2010 NetLogic Microsystems, Inc.

All rights reserved.

How to Use This Manual

Mastering the XLS architecture can be achieved systematically. To gain understanding, you should learn about the:

- CPUs
- Internal networks and data management and distribution mechanisms
- External networks interfaces
- Memory interfaces
- Peripheral interfaces
- System control

The following is a guide to reading sequence:

Chapter 1 provides the introductory overview of the XLS architecture, and should be read first by all users.

Chapters 2 through 7 provide understanding of the CPUs, threading, and the caches. Note that you should understand XLS nCPU/thread operation before reading Chapter 10 on the Programmable Interrupt Controller.

Chapter 8 provides orientation to the memory organization and basic register access. All users should read this to understand XLS addressing and how Distributed Interconnects provide internal raw data transport.

Next, gain a basic overview of the Fast Messaging Network (Chapter 12). Then read for a basic first understanding of operation and use of the Networking Accelerators (Chapter 13), the Security Accelerator Engine (Chapter 14) and the DMA Engine (Chapter 9). Following this framework, you will want to reread these chapters to integrate details and tie knowledge together.

To understand memory device control, read Chapter 11 for DRAM and Chapter 22 for Flash memory. Each of these chapters can be read independently.

The chapters on peripheral interfaces may be read in any sequence. Note that some system control is achieved using GPIO registers. See the GPIO chapter for details.

System control can be studied by use of design information in the XLS data book, with referral back to specific registers in the programmer's reference manual as needed.

NetLogic's goal is to help you succeed. We wish you good progress.

Document Conventions

The following documentation notations and conventions are used in this Data Book.

Item	Description
Number multipliers	k = 1000 (10^3), usually used with frequency values K = 1024 (2^{10}), used for memory size references Kbyte = 1024 bytes M = mega, = 1,000,000 when referring to frequency (decimal notation) = 1,048,576 when referring to memory sizes (binary notation)
0x00	0x prefix identifies hex values; Each suffix term is equivalent to a 4-bit binary value (nibble) Legal suffix terms are: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F Formats are right justified
b1010 001: '0' and '1'	A 'b' prefix or a colon following a number are used as binary notation. '0' and '1' are the only legal digits
0 through 9	Decimal numbers (sometimes may have a 'd' prefix)
Byte (B)	Eight bits
LSB / lsb	Least-significant Byte / least-significant bit
MSB / msb	Most-significant Byte / most-significant bit
Word	A word is 32 bits or 4 bytes
Half Word (HW)	A half word is 16 bits or 2 bytes
Double Word (DW)	A double word is 64 bits or 8 bytes The exception is PCI, where double word is 32 bits.
Reserved	1. Reserved bits and words should not be modified by the user. 2. Unless otherwise specified, these bits read as '0' and should be written as '0'.
Active Low signals	Unless otherwise specified, active Low signals are identified by a "_L" appended to the name (for example, RESET_L, HSYNC_L).
" "	Assignment
=, ≠	Tests for equality and inequality
	Bit-string concatenation
x ^y	A y-bit string formed by y copies of the single-bit value x
b#n	A constant value n in base b. For instance 10#100 represents the decimal value 100, 2#100 represents the binary value 100 (decimal 4), and 16#100 represents the hexadecimal value 100 (decimal 256). If the "b#" prefix is omitted, the default base is 10.
x _{y..z}	Selection of bits y through z of bit string x. Little-endian bit notation (rightmost bit is 0) is used. If y is less than z, this expression is an empty (zero length) bit string.
+,-	2's complement or floating point arithmetic: addition, subtraction
* , x	2's complement or floating point multiplication (both used for either)
div	2's complement integer division
mod	2's complement modulo
/	Floating point division
<	2's complement less-than comparison
>	2's complement greater-than comparison
≤	2's complement less-than or equal comparison

Item	Description
\geq	2's complement greater-than or equal comparison
NOR	Bitwise logical NOR
XOR	Bitwise logical XOR
AND	Bitwise logical AND
OR	Bitwise logical OR
GPRLEN	The length in bits (32 or 64) of the CPU general-purpose registers
GPR[x]	CPU general-purpose register x. The content of GPR[0] is always zero.
CPR[z,x,s]	Coprocessor unit z, general register x, select s
CCR[z,x]	Coprocessor unit z, control register x
COC[z]	Coprocessor unit z condition signal
Xlat[x]	Translation of the MIPS16 GPR number x into the corresponding 32-bit GPR number
BigEndianMem	Endian mode as configured at chip reset (0 = Little-Endian, 1 = Big-Endian). Specifies the endianness of the memory interface (see LoadMemory and StoreMemory pseudocode function descriptions), and the endianness of Kernel and Supervisor mode execution.
BigEndianCPU	The endianness for load and store instructions (0 = Little-Endian, 1 = Big-Endian). In User mode, this endianness may be switched by setting the RE bit in the Status register. Thus, BigEndianCPU may be computed as (BigEndianMem XOR ReverseEndian).
ReverseEndian	Signal to reverse the endianness of load and store instructions. This feature is available in User mode only, and is implemented by setting the RE bit of the Status register. Thus, ReverseEndian may be computed as (SRRE and User mode).
LLbit	Bit of virtual state used to specify operation for instructions that provide atomic read-modify-write. LLbit is set when a linked load occurs; it is tested and cleared by the conditional store. It is cleared, during other CPU operation, when a store to the location would no longer be atomic. In particular, it is cleared by exception return instructions.

NETLOGIC
CONFIDENTIAL



Table of Contents

Chapter 1 XLS™ Processor Family Introduction

1.1	Introduction	35
	Versatile Next Generation XLS™ Thread Processor Core	35
1.2	Key Features of the XLS™ Processor Family	38
	XLS616 and XLS608 Block Diagram	39
	XLS416 and XLS408 Block Diagram	40
	XLS408-Lite and XLS404-Lite Block Diagram	41
	XLS404 Block Diagram	42
	XLS208 and XLS204 Block Diagram	43
	XLS108 and XLS104 Architecture, Memory and I/O Diagram	44
	Low-Power Operation	45
1.3	Enhanced CPU Core Features	45
	CPU Registers	45
	Coprocessor 0 (COP0) Registers	46
	Coprocessor 2 (COP2) Registers	46
	Integer Unit	46
	MIPS64 Instruction Set Enhancements	46
	Extended MIPS64 Interrupt Architecture	47
	Message Passing Extensions	47
1.4	XLS Multi-Threading	47
	Scheduling Flexibility of the XLS Processor	48
1.5	Processor Memory Subsystem	52
	Level-1 Caches	53
	Level-2 Unified Cache	54
	Virtual MIPS Mode (Virtualization)	54
	DRAM Interface	54
1.6	XLS Processor Network Accelerators	55
	Networking Accelerator Architecture	55
1.7	Fast Messaging Network	57
	Fast Messaging Network Architecture	58
1.8	Security Engine	59
1.9	Programmable Interrupt Controller	60
1.10	Compression/Decompression Engine	60
1.11	Data Transport	61
1.12	Peripherals	61
1.13	Development Support	62
	Debug Capabilities	62

Chapter 2 XLS MIPS64 CPU Core

2.1	Introduction	63
------------	---------------------	-----------

Conformance	63
2.2 CPU Data Path	65
Integer Unit	65
CPU Registers	65
Load/Store Unit	65
Level-1 Instruction Cache	66
Level-1 Data Cache	66
DedicatedPlus Cache	66
Coprocessor 0 (COP0) Registers	66
Coprocessor 2 (COP2) Registers	66
2.3 Memory SubSystem	67
Memory Management Unit	67
2.4 CPU Instruction Path	68
Processor Pipeline	68
2.5 Instruction Timing	69
Threading Considerations	70
Load Timing	70
Pipeline Hazards	71
2.6 Code Compatibility	71
2.7 CPU Reset Control	71

Chapter 3 Virtual MIPS Mode

3.1 Introduction	73
3.2 Virtual MIPS Mode Operation in 32-bit Compatibility Mode	73
3.3 XLS-Specific Registers for Virtual MIPS Mode	75
Instruction-Side Register Settings	75
Data-Side Register Settings	75

Chapter 4 XLS Processor Core Registers

4.1 XLS Processor Core Registers	77
General-Purpose Registers	77
Special-Purpose Registers	77
4.2 Coprocessor 0 (COP0) Registers	78
Definition of Terms and Usage	78
Coprocessor 0 Register Summary	79
Coprocessor 0 Register Descriptions	81
4.3 Coprocessor 2 (COP2) Registers	127
Coprocessor 2 Register Summary	127
Coprocessor 2 Register Descriptions	128
4.4 XLS Processor Control Registers	135
MFCR / MTCR Instructions	135
XLS Processor Control Registers Summary	136
XLS Processor Control Registers Descriptions	137

Chapter 5 XLS CPU Instruction Set

5.1 Introduction	161
5.2 XLS Processor Instruction Set Overview	161

5.3 MIPS64 Instruction Set	161
CACHE - Perform Cache Operation	166
PREFETCH	170
DMFC2 — Doubleword Move from Coprocessor 2	172
DMTC2 — Doubleword Move to Coprocessor 2	173
MFC2 — Move Word From Coprocessor 2	174
MTC2 — Move Word to Coprocessor 2	175
MSG SND — Send Message Command	176
MSG LD — Load Message Command	177
MSG WAIT — Wait for Message in the Receive Queue	178
LDADDW — Load and Add Word	179
LDADDWU — Load and Add Word Unsigned	180
LDADD — Load and Add Double	181
SWAPW — Swap Word	182
SWAPWU — Swap Word Unsigned	183
SWAPD — Swap Double	184
DADDWC — DoubleWord Add with Carry	185
MFCR — Move From Control Register	186
MTCR — Move To Control Register	187

Chapter 6 XLS L1 Cache

6.1 Introduction	189
6.2 Capabilities and Functions	189
Key Capabilities Summary	189
Level-1 Instruction Cache	190
Level-1 Data Cache	190
ECC Protection	190
DedicatedPlus™ Cache	190
6.3 Programming Model	191
6.4 L1 Cache Control Registers	191
L1 Cache Control Registers Summary	191
L1 Cache Control Registers Descriptions	192

Chapter 7 XLS Unified L2 Cache

7.1 Introduction	205
7.2 Capabilities and Functions	205
7.3 Programming Notes	206
Using the L2 Cache Bank Configuration Registers	206
7.4 L2 Cache Control Registers	209
Background	209
L2 Cache Control Register Summary	210
L2 Cache Control Register Descriptions	211
L2 Cache Access Through JTAG	220

Chapter 8 Memory and I/O Access

8.1 Introduction	221
8.2 Elements of the XLS Internal Data Transport Architecture	221

Memory Distributed Interconnect (MDI)	221
I/O Distributed Interconnect (I/O DI)	221
Memory and I/O Distributed Interconnect Hub	222
Memory Bridge CD for DRAM Controller Pair CD	222
8.3 System Physical Memory Map	222
8.4 Devices Mapped in the System Bridge Controller	226
8.5 System Bridge Controller Registers	228
System Bridge Controller Read-Access Requirements	228
System Bridge Controller Registers Summary	229
8.6 DRAM Configuration Registers	231
DRAM_BAR[0-7]	235
DRAM_CHNAC_DTR[0-7]	236
DRAM_CHNBD_DTR[0-7]	237
DRAM_BRIDGE_CFG	238
8.7 Peripherals, I/O and I/O Memory Base Address Registers	241
XLS_IO_BAR	242
FLASH_BAR	242
8.8 Functional Blocks Without Separate Base Address Registers	243
8.9 Address Error Registers	244
DEVICE_MASK	245
AERR0_LOG1	247
AERR0_LOG2	248
AERR0_LOG3	249
AERR0_DEVSTAT	250
AERR1_LOG1	251
AERR1_LOG2	252
AERR1_LOG3	253
AERR1_DEVSTAT	254
AERR0_EN	255
AERR0_UPG	257
AERR0_CLEAR	258
AERR1_CLEAR	259
SBE_COUNTS	259
DBE_COUNTS	260
BITERR_INT_EN	261
8.10 System Credits and Counter Registers	262
SYS2IO_CREDITS	262
EVENT_CNT_CNTRL1	263
EVENT_CNR1	265
EVENT_CNT_CNTRL2	265
EVENT_CNR2	267
MISC FUNCTION CTL1	268
8.11 Scratch Registers	270
SCRATCH[0-3]	270
8.12 PCIe Base Address Registers	271
PCIE_CFG_BAR	271
PCIE_ECFG_BAR	271
PCIE_MEM_BAR	272
PCIE_IO_BAR	272
8.13 SRIO Base Address Registers	273
SRIO_MEM_BAR	273
SRIO_IO_BAR	273
8.14 Additional Device Mask Register	274
DEVICE_MASK2	274

Chapter 9 Direct Memory Access (DMA) Engine

9.1	Introduction	275
	Functionality	275
	Architecture and Theory of Operation	276
9.2	Programming Model	277
	Background Information for Initialization	277
	Reset Sequence	278
9.3	DMA Engine Message Formats	279
	Descriptor Formats	279
	Simultaneous Simple and Other DMA Message Types	286
9.4	DMA Engine Registers	287
	DMA Engine Register Summary	287
	DMA Engine Configuration and Debug Register Descriptions	288
	DMA Engine Credit Counter Registers	299
	DMA MsgBucketSize Registers	299

Chapter 10 Programmable Interrupt Controller

10.1	Introduction	301
	PIC Functionality Overview	302
	Interrupt Processing Details	303
	Interrupt Redirection Table (IRT)	304
	Global and Local Scheduling to Select the Processor Core	304
	Description of Interrupt Redirection Table	305
	Per-Thread Interrupt Registers	308
	Interrupts from External Sources	308
10.2	Programmable Interrupt Controller Timers	309
10.3	Programmable Interrupt Controller Registers	310
	Programmable Interrupt Controller Registers Summary and Descriptions	310

Chapter 11 DRAM Controllers and Interface

11.1	Introduction	313
11.2	Operating Capabilities	313
11.3	DRAM Subsystem Elements and Descriptions	314
	Terminology	317
	DRAM Device Types and Characteristics	318
	DDR2 DRAM Interface Signals	319
	DRAM Configurations Supported by XLS	320
11.4	Programming Model	323
	Memory Controller Configuration	323
	Memory Controller Initialization Sequence	325
	Understanding The DLLs	327
	Understanding the DRAM Memory Bridge/Memory Controller Interaction	334
	Understanding the Thermal Control Registers	335
	Understanding the User Command Sequence	335
	Understanding Read Data Capture	337
	Initialization Sequence	341
	Using DDR2 On-Die Termination (ODT)	342

DRAM Controller Modes	343
Programming Sequence	344
11.5 Memory Controller Register Descriptions	345
Memory Controller Register Summary	345
DDR Timing Parameter Registers	348
DDR Controller Configuration Registers	355
DDR DLL Control Registers	359
DDR Thermal Control Registers	374
DDR MRS Command Registers	378
DDR Reset Sequence Control Registers	382
DDR ECC Registers	386
DDR ODT Control Registers	389
Performance Measuring Registers	393

Chapter 12 Fast Messaging Network

12.1 Introduction	397
Fast Messaging Network Architecture	398
Example Usage of the FMN	398
Message Size	400
Stations, Entries, Receive Queues and Buckets	400
Credit-based Message Management	404
Message Precedence and Automatic Bandwidth Administration	407
12.2 FMN Registers for Non-Core Stations	409
Non-Core MsgBucketSize Register Format	409
XAUI/SGMII Non-Core MsgBucketSize Register IDs	409
Non-Core Credit Counter Register Format	411
Non-Core Credit Counter Register IDs	411
PCIE_MSG_TX_THRESHOLD	411
12.3 Interrupt Causes from Core Stations	412
Functional Interrupts	412
Debugging/Status Interrupts	413
12.4 Configuring the Messaging System	414
Setting Up Buckets	414
Credits	415
Sending Messages	417
Receive/Load Messages	418
Defeaturing and Debug	419

Chapter 13 Networking Accelerators

13.1 Introduction	421
13.2 Architecture	421
13.3 Theory of Operation	423
Packet Data Movement	423
Accessing the Networking Accelerator Registers	424
Message Flow and Use of Packet Descriptors	427
Packet Descriptor Types	431
FMN Messages	432
Packet Descriptor Definitions	433
Usage of Pointer-to-Data Type Tx Packet Descriptors	438

Using Tx P2D Descriptors to Modify Egress Packets	439
Tx Free Back Descriptor	440
Rx Free In Descriptor	441
13.4 Parser	442
Theory of Operation	442
Parser Registers	447
13.5 TCP Checksum Hardware Support	458
13.6 Packet Director	459
TCAM	462
Secondary Methods	464
13.7 Packet Distribution Engine	467
Bypass Packets Capability	468
Packet Director and Packet Distribution Engine Registers	470
13.8 Prepad	474
13.9 Programming Model	481
System Setup	481
Thread Processing	481
Software Reset	482
Interrupts	482
13.10 Networking Accelerator Registers	483
Register Summary	483
GMAC Registers for SGMII Interfaces in the Network Accelerators	487
SGMII SERDES Control Registers	496
DMA Credit Registers	498
DMA Spill Control Registers	500
Transmit Data FIFO Configuration	508
Split Mode Support	510
FIFO Watermark Registers	510
IEEE1588_PTP Source Registers	512
13.11 PDE Class Registers	513
PDE_CLASS0	513
PDE_CLASS1	514
PDE_CLASS2	514
PDE_CLASS3	515
Debug Registers	515

Chapter 14 Security Acceleration Engine

14.1 Introduction	517
Crypto Core	518
RSA Core	521
14.2 Theory of Operation	522
Security Acceleration Engine Flow	522
Security Block Data and Control Exchange	524
Fragmented Payload Support and Arbitrary Size Payloads	524
14.3 Security Descriptor Message Format	526
Control Descriptor Data Structure	528
Data Descriptor Data Structure	536
14.4 Security Free Out Descriptor Message Format	542
14.5 RSA Core	545
RSA / ECC Control Descriptor Message Format	545
RSA Data Formats	548
ECC Prime Data Formats	549

ECC Binary Data Formats	551
RSA / ECC Free Out Descriptor Message Format	553
14.6 Clock Gating	555
14.7 Security Acceleration Registers	556
Register Access Model	556
Register Summary	556
SEC_STATE0 Register	558
DMA_Credit Register	559
Config1 Register	559
Config2 Register	559
Config3 Register	559
14.8 Fragmentation and Offset	560
Rebuilding Packets from Fragments on 64-bit Boundaries	560
Implications of Fragmentation on AES	562
14.9 Programming Model	563
Basic Configuration	563
Optional Configuration Settings	565
Additional Considerations	566
14.10 Application Example	570

Chapter 15 Compression/Decompression Engine

15.1 Introduction	573
15.2 Capabilities and Functions	573
Key Capabilities Summary	573
15.3 Theory of Operation	574
Architecture	574
Modes of Operation	575
Compression/Decompression Engine Processing Flow	576
Interrupts	580
Errors and Events Handling for CDE in the System Architecture	580
15.4 Programming Model	581
System Setup for Startup and Initialization	581
Interface Setup for Startup and Initialization	582
Standard Message Interface (Two Buckets)	582
Standard I/O Ring Interface (Three Channel DMA)	582
Message Format	582
15.5 CDE Configuration Registers	588
Definition of Terms	588
Endianness for the Compression/Decompression Engine	588
Register Access	588
Compression/Decompression Engine Register Summary	588
CONTROL_REG_0	589
DMA_CREDITS	590
FREE_IN_SPILL_MEM_START_0	590
FREE_IN_SPILL_MEM_START_1	591
FREE_IN_SPILL_MEM_SIZE	591
FREE_IN_SPILL_MEM_BYTES	591
CRC_ADLER_SPILL_MBUF	592
SCRATCH_PAGE	592
INTERRUPT_VEC	593
INTERRUPT_MASK	594

FREE_DESC_THRESHOLD	594
DESC_FIFO_WORD_COUNTS	594
RESET_REG	595
ERROR_RESET_MASK	596
ADDRESS_OF_READ_ERROR_LIST0	597
ADDRESS_OF_READ_ERROR_LIST1	597

Chapter 16 SGMII and RGMII Interface

16.1 Introduction	599
Optional XAUI Interface	599
16.2 Capabilities and Functions	601
Key Capabilities Summary	601
16.3 Theory of Operation	602
Position of Interface Within the System Architecture	602
Overview of SGMII/RGMII Port GMAC Mode Operation	603
RGMII Operation	605
Overview of RGMII Mode FIFO Mode Operation	606
16.4 Programming Model	612
System Setup for Startup and Initialization	612
Interface Setup for Startup and Initialization	613
Interrupts	613
Statistics Monitoring	613
16.5 SGMII and RGMII Registers	614
Introduction	614
GMAC Register Descriptions	617
GMAC Statistics Registers Descriptions	639
PCS Register Descriptions	666

Chapter 17 XAUI Interface

17.1 Introduction	675
XAUI Interface Selection and Configuration	675
17.2 XAUI Overview	676
XAUI Interface Features	677
XAUI Architecture	677
17.3 Theory of Operation	680
Position of XAUI Interface Within the System Architecture	682
XAUI Block Diagram	683
17.4 Programming Model	684
System Setup for Startup and Initialization	684
Reset Handling	685
Statistics Monitoring	685
17.5 XAUI Registers	686
Introduction	686
Register Access	686
XAUI Interface Register Summary	687
Configuration Register 0	688
Configuration Register 1	689
Configuration Register 2	692
Configuration Register 3	693

MAC_ADDR0_LO	694
MAC_ADDR0_HI	694
Maximum Frame Length	695
Revision Level	695
MII Management Command Register	696
MII Management Field Register	697
MII Management Configuration Register	698
MIIM Link Fail Vector	698
MII Management Indicator Register	699
MAC_ADDR0_LO	700
MAC_ADDR0_HI	700
MAC_ADDR1_LO	700
MAC_ADDR1_HI	701
MAC_ADDR2_LO	702
MAC_ADDR2_HI	702
MAC_ADDR3_LO	703
MAC_ADDR3_HI	703
MAC_ADDR_MASK0_LO	703
MAC_ADDR_MASK0_HI	703
MAC_ADDR_MASK1_LO	704
MAC_ADDR_MASK1_HI	704
MAC_FILTER_CONFIG	704
HASH_TABLE_VECTOR	704
17.6 XAUI Statistics Registers Summary	705
Transmit and Receive Counters	707
Receive Counters	710
Transmit Counters	718
Carry and Carry Mask Registers	727
17.7 XAUI PCS Registers	732
XAUI PCS-Register Summary Table	732

Chapter 18 USB Interface

18.1 Introduction	737
18.2 Theory of Operation	738
Overall Operation	738
Device Controller Theory of Operation	740
Host Controller Theory of Operation	742
18.3 Global Programming Model	744
System Setup for Startup and Initialization	744
Clock and Reset	745
Errors and Events Handling for USB Interface in the System Architecture	745
18.4 Host Programming Model	746
18.5 Device Programming Model	749
Interface Setup	749
Core Initialization	749
Modes of Operation	750
Device Programming Model — Endpoint Initialization	751
Device Programming Model — Operational Model	753
Device Programming Model — Handling Babble Conditions	785
Device Programming Model -- Partial Power-Down and Clock Gating	785
Miscellaneous Topics	786

18.6 USB Interface Registers	789
Register Fields	789
Access to Registers	789
18.7 USB Interface General Registers	792
Accessing the USB Interface General Registers	792
USB Interface General Registers Summary	792
USB Interface General Registers Descriptions	793
USB_GEN_CTRL1	793
USB_GEN_CTRL2	793
USB_GEN_CTRL3	794
USB_IOBM_TIMER	794
USB_VBUS_TIMER	795
USB_BYTESWAP_EN	795
USB_COHERENT_MEM_BASE	795
USB_COHERENT_MEM_LIMIT	796
USB_L2ALLOC_MEM_BASE	796
USB_L2ALLOC_MEM_LIMIT	797
USB_READEX_MEM_BASE	797
USB_READEX_MEM_LIMIT	798
USB_PHY_STATUS	798
USB_INTERRUPT_STATUS	798
USB_INTERRUPT_ENABLE	799
18.8 Device Controller Registers	800
Accessing the USB Device Controller Registers	800
Device Controller Control and Status Registers Summary	801
Register Fields	803
USB Device Controller Data FIFO (DFIFO) Register Summary	804
18.9 Device Controller Register Descriptions	805
UDC Configuration A (UDC_GAHBCFG)	805
UDC Configuration B (UDC_GUSBCFG)	806
UDC Reset (UDC_GRSTCTL)	808
UDC Interrupt Register (UDC_GINTSTS)	811
UDC Interrupt Mask Register (UDC_GINTMSK)	814
UDC Receive Status Read/Pop Register (Read Only) (UDC_GRXSTSP)	815
UDC Receive FIFO Size Register (UDC_GRXFSIZ)	816
UDC Non-periodic Transmit FIFO Size Register (UDC_GNPTXFSIZ)	816
UDC User ID Register (UDC_GUID)	817
UDC ID Register (Read Only) (UDC_GSNPSID)	817
UDC User HW Config1 Register (Read Only) (UDC_GHWCFG1)	817
UDC User HW Config2 Register (Read Only) (UDC_GHWCFG2)	818
UDC User HW Config3 Register (Read Only) (UDC_GHWCFG3)	819
UDC User HW Config4 Register (Read Only) (UDC_GHWCFG4)	820
UDC Device IN Endpoint Transmit FIFO-n Size Register (UDC_DPTXFSIZn)	821
UDC Configuration C (UDC_DCFG)	822
UDC Control Register (UDC_DCTL)	823
UDC Status Register (Read Only) (UDC_DSTS)	825
UDC Device IN Endpoint Common Interrupt Mask Register (UDC_DIEPMSK)	826
UDC Device OUT Endpoint Common Interrupt Mask (UDC_DOEPMSK)	826
UDC Device All Endpoints Interrupt Register (UDC_DAINT)	827
UDC Device All Endpoints Interrupt Mask Register (UDC_DAINTMSK)	827
Device Threshold Control Register (UDC_DTKNQR3)	828
Device IN Endpoint FIFO Empty Interrupt Mask Register (UDC_DTKNQR4)	829
Device Control IN Endpoint 0 Control Register (UDC_DIEPCTLn)	829
Device Control OUT Endpoint 0 Control Register (UDC_DOEPCTL0)	831

Device Endpoint-n Control Register	832
Device Endpoint-n Interrupt Register (UDC_DIEPINTn)	835
Device Endpoint 0 Transfer Size (UDC_DIEPTSIZ0/ UDC_DOEPTSIZ0)	837
Device Endpoint-n Transfer Size Register (UDC_DIEPTSIZn)	838
Device Endpoint-n DMA Address Register (UDC_DIEPDMA _n)	839
Device IN Endpoint Transmit FIFO Status Register (UDC_DTXFSTS _n)	839
Power and Clock Gating Control Register (UDC_PCGCR)	840
18.10 Host Controller Registers	841
18.11 ECHI Operational Registers (EOR)	842
18.12 OHCI PCI Configuration, and Operational Register Maps	843

Chapter 19 PCI Express™ Interface

19.1 Introduction	845
PCIe Features	845
19.2 Overview of PCIe Interface Configurations	846
XLS6xx and XLS4xx PCIe Controllers	846
XLS408Lite and XLS404Lite PCIe Controllers	846
XLS2xx Controllers	846
XLS1xx Controllers	846
19.3 Configuring the PCIe Interface	847
Supported PCIe Configurations for XLS6xx and XLS4xx Devices	849
19.4 Theory of Operation	854
Accessing PCIe Controller's Configuration Space	854
Accessing PCIe Memory Space	856
Accessing PCIe I/O Space	857
PCIe Configuration Requests	857
19.5 DMA Transfers	858
XLS to PCIe Bus Transaction	858
PCIe to XLS Subsystem Memory	858
19.6 PCIe Interrupts	860
Interrupt Setup	860
Message Signaled Interrupts (MSI)	860
PCIe Invalid Access Interrupt	861
Configuring PCIe and CPU for PCIe Interrupts	862
Interface Setup for Startup and Initialization in Device Mode	862
Interface Setup for Startup and Initialization in Host Mode	862
19.7 Programming Model	862
Startup/Initialization	862
Flushing the Interface Buffers	862
Ensure Writes Complete Before Reads	863
19.8 Global PCI Express Interface Registers	864
Field Read/Write Access Codes	864
Register Address Offsets	864
Summary of XLS6xx, XLS4xx, XLS2xx & XLS1xx PCIE Global Registers	864
Summary of the XLS408Lite & XLS404Lite Global PCIE Registers	900
19.9 PCIe Configuration Registers: EP Mode	923
Register Space Layout	923
Register Maps	924
Accessing Configuration Registers	927
Register Default Reset Values	927
PCI-Compatible Configuration Header Register Details	928

PCI Power Management Capability Register Details	939
MSI Capability Register Details	941
PCI Express Capability Register Details	943
PCI Express Extended Capabilities Register Details	949
19.10 PCIe Configuration Registers: RC Mode	953
Register Space Layout	953
Register Maps	953
Accessing Configuration Registers	956
Register Default Values	957
PCI-Compatible Configuration Header Register Details	958
PCI Power Management Capability Register Details	968
MSI Capability Register Details	970
PCI Express Capability Register Details	971
PCI Express Extended Capabilities Register Details	977

Chapter 20 Serial-RapidIO Interface

20.1 Introduction	983
20.2 Overview of Serial-RapidIO (SRIO) Interface Capabilities	983
SRIO Selection and Configuration	984
20.3 SRIO Theory of Operation	985
Accessing SRIO Interface Configuration Registers	985
Accessing SRIO Configuration and Control Registers	986
Resetting the SRIO Controllers	986
20.4 Programming Model	987
Startup/Initialization	987
Functionality Overview	987
Source Processing	988
Destination Processing	989
Data Structures	990
20.5 Serial RapidIO (SRIO) Interface Registers	1003
Field Modifiability Codes	1003
SRIO_CTRL	1004
SRIO_PHY_CTRL0	1004
SRIO_PHY_CTRL1	1005
SRIO_PHY0_CTRL	1005
SRIO_PHY1_CTRL	1005
SRIO_PHY2_CTRL	1006
SRIO_PHY3_CTRL	1006
SRIO_COHERENT_MEM_BASE	1007
SRIO_COHERENT_MEM_LIMIT	1007
SRIO_REG_L2ALLOC_MEM_BASE	1007
SRIO_REG_L2ALLOC_MEM_LIMIT	1007
SRIO_REG_READEX_MEM_BASE	1007
SRIO_REG_READEX_MEM_LIMIT	1007
SRIO_REG_PHY_CR_CMD	1008
SRIO_REG_PHY_CR_WR_DATA	1008
SRIO_REG_PHY_CR_RESP	1008
SRIO_REG_PHY_CR_RD_DATA	1008
20.6 SRIO Programming Register Space	1009
DEVICE_IDENTITY_CAR	1010
DEVICE_INFORMATION_CAR	1010
SOURCE_OPERATIONS_CAR	1010

DESTINATION_OPERATION_CAR	1011
PE_LL_CSR	1011
LCS_UBA_CSR	1011
LCS_LBA_CSR	1012
BD_ID_CSR	1012
HBD_ID_CSR	1012
CP_TAG_CSR	1013
PM_BLK_HD0	1013
PL_TO_CSR	1013
PR_TO_CSR	1013
PG_CTRL_CSR	1014
P0_LMRQ_CSR	1014
P0_LMRS_CSR	1014
P0_LAID_CSR	1015
P0_EAS_CSR	1015
P0_CTRL_CSR	1016
20.7 SRI0 Configuration and Control Registers	1018
Interrupt Generation	1020
General Registers	1022
Physical Layer Registers	1027
RapidIO to Host Address Translation Registers	1029
Error Injection Registers	1030
Transaction Queues Registers	1032
Status Queue	1037
DOORBELL FIFO	1040
Port-write FIFO	1042
Free-Lists	1044
Mailbox Queues	1049
Logical/Transport Layer Error and Capture Registers	1056
Port 0 Error Reporting Registers	1062

Chapter 21 UARTs, I²C, and Peripherals Interface

21.1 Introduction	1067
21.2 UART Ports	1068
UART Port Signals	1068
21.3 I²C Ports	1069
I ² C Port Signals	1069
21.4 Programming Model	1070
Register Addressing for UARts and I ² C	1070
UART Port Programming	1071
I ² C Port Programming	1071
21.5 UART and I²C Registers	1076
UART and I ² C Register Summary	1076
UART Register Descriptions	1076
I ² C Host Controller Register Descriptions	1085

Chapter 22 Flash/PCMCIA Memory and the Peripherals Interface

22.1 Introduction	1091
22.2 Technology: NOR Flash vs. NAND Flash	1091

22.3 NAND Flash Basics	1092
22.4 Interfacing FLASH Memory to the XLS	1093
FLASH Memory on the Peripherals Input/Output Bus	1093
Flash Memory Devices and GPIO Port Signals	1096
Initial States for Flash ROM on the Peripherals I/O Bus	1098
XLS NAND Flash Interface	1100
PCMCIA Flash Memory Support	1103
22.5 Programming Model	1104
Register Addressing for Flash/PCMCIA	1104
22.6 Flash/PCMCIA Memory Registers	1105
22.7 Flash/PCMCIA Register Descriptions	1108
Chip-Select Area Registers	1108
Flash Memory Controller Registers	1113
NAND Flash Memory Control Registers	1116
Notes	1116
AC Timing for Flash Enhancements	1116

Chapter 23 GPIO and the Peripherals Interface

23.1 Introduction	1119
GPIO Port Signals	1119
General System Functions Supported by GPIO Pins	1120
Obtaining the Processor ID Value using the GPIO Register	1120
23.2 Programming Model	1122
Register Addressing for GPIO Interface	1122
Setup for Operation	1123
23.3 GPIO Registers	1123
General Purpose Input/Output Signal Registers	1125
DRAM Clock Rate Registers	1132
PLL Control and Status Registers	1135
Clock Divider Control Registers	1139
GPIO System Control Registers	1141

Chapter 24 IEEE 1588 Precision Time Protocol

24.1 Introduction	1159
Benefits of IEEE 1588 Version 2	1160
Boundary and Transparent Clocks	1161
XLS6xx and XLS4xx PTP Programming Registers	1162
IEEE-1588_PTP Registers	1163
1588_PTP Control in the Network Accelerator	1166

Chapter 25 XLS Debugging Features

25.1 Introduction	1167
25.2 EJTAG and Debug Testing	1167
25.3 Obtaining Processor ID and Revision ID via the IDCODE Instruction	1168
25.4 EJTAG Register Interface	1169
TAP_INST_REG, TAP Instruction Register	1169
IMPCODE	1171

Address Register	1172
Data Register	1174
TAP Control Register	1174
Debug Control Register (DCR)	1176
25.5 System Diagnostics	1177
25.6 Performance Monitoring	1178
Performance Monitoring of Processor Events	1178
Performance Monitoring of Memory Transactions	1178
25.7 Instruction Statistical Sampling	1179
25.8 Watchpoint Debugging	1179
25.9 Trace Buffer	1180
Trace Buffer Operation	1181
Programming the Trace Buffer	1181
Extracting Trace Buffer Entries	1182
25.10 Trace Buffer Registers	1182
Request Match Register 1	1182
Request Match Register #2	1183
Request Address Low Match Register	1183
Request Address High Match Register	1184
Control Register	1184
Initialization Register	1184
Access Register	1184
Read Data Registers (RDR) 0 – 3	1185
Write Data Register (WDR) 0 – 3	1186
Status Register	1186
Trace Buffer Driver Example	1186
25.11 Network Accelerator Debug Registers	1187
DebugCounter0_A	1187
DebugCounter0_B	1187
DebugEvent0_A	1188
DebugEvent0_B	1190
DebugCounter1_A	1190
DebugCounter1_B	1190
DebugEvent1_A	1191
DebugEvent1_B	1191
DebugCounter2_A	1192
DebugCounter2_B	1192
DebugEvent2_A	1192
DebugEvent2_B	1192
DebugDescWordCounts	1192
DebugDescWordCountsSel	1193
DebugRxDescInMAC	1194
DebugTxDescInMAC	1194
25.12 Debug Interrupt Overview	1195
Appendix A Multi-Threading Advantages.....	1197
Multi-Threading Improves Processor Performance	1197
Traditional Processor Design	1197
Single-Thread, Single-Issue Processors	1197
Multiple-Issue Processor Design	1198
The Multi-Threaded Single-Issue Approach	1199
Appendix B Revision History.....	1203



List of Figures

Chapter 1 XLS™ Processor Family Introduction

Figure 1-1.	XLS616 and XLS608 Block Diagram.....	39
Figure 1-2.	XLS416 and XLS408 Block Diagram.....	40
Figure 1-3.	Diagram: XLS408-Lite and XLS404-Lite Block Diagram	41
Figure 1-4.	Diagram: XLS404 Block Diagram	42
Figure 1-5.	Diagram: XLS208 and XLS204 Block Diagram	43
Figure 1-6.	Diagram: XLS108 and XLS104 Block Diagram	44
Figure 1-7.	XLS Core Attributes	45
Figure 1-8.	Round Robin (Fine-Grained) Scheduling.....	48
Figure 1-9.	Round Robin (Fine-Grained) Scheduling with One Stalled Thread	49
Figure 1-10.	Fixed Cycle Scheduling	49
Figure 1-11.	Priority Thread Scheduling	50
Figure 1-12.	Networking Accelerator Packet Flow (XLS6xx Example)	56
Figure 1-13.	Fast Messaging Network Stations Example	58
Figure 1-14.	2.5-Gigabit Security Acceleration Engine (Conceptual Diagram)	59
Figure 1-15.	Single 2.5 Gigabit Crypto Core	59
Figure 1-16.	Programmable Interrupt Controller Block Diagram.....	60

Chapter 2 XLS MIPS64 CPU Core

Figure 2-1.	Core Functional Block Diagram.....	64
Figure 2-2.	The XLS Processor MIPS64 CPU Core Pipelines.....	68

Chapter 3 Virtual MIPS Mode

Figure 3-1.	Virtual MIPS Mode.....	74
-------------	------------------------	----

Chapter 4 XLS Processor Core Registers

Figure 4-1.	Wired and Random Entries in TLB	87
-------------	---------------------------------------	----

Chapter 5 XLS CPU Instruction Set

Figure 5-1.	Usage of Address Fields to Select Index and Way.....	167
-------------	--	-----

Chapter 6 XLS L1 Cache

Chapter 7 XLS Unified L2 Cache

Chapter 8 Memory and I/O Access

Figure 8-1.	Distributed Interconnect in the XLS616	223
Figure 8-2.	The Memory and I/O Distributed Interconnect Stations In Detail.....	224

Figure 8-3.	The 1.0 TeraByte Physical Address Space on the MDI.....	225
Figure 8-4.	Example of Three DRAM Port MB Regions in the Address Space on the MDI.....	233
Figure 8-5.	DRAM Channel Address Mapping in the SBC.....	234
Figure 8-6.	Peripherals, I/O and I/O Memory Base Address Registers on the MDI	241
Figure 8-7.	Address Error Registers for Devices on the MDI.....	244
Figure 8-8.	General-Purpose Scratch Registers on the MDI.....	270
Chapter 9	Direct Memory Access (DMA) Engine	
Figure 9-1.	DMA Engine Key Elements.....	276
Chapter 10	Programmable Interrupt Controller	
Figure 10-1.	Programmable Interrupt Controller Inputs and Outputs	301
Figure 10-2.	Interrupt Processing Flow	303
Figure 10-3.	Per-Thread Registers for Exceptions	308
Figure 10-4.	Interrupts from External Sources and Their Mappings into IRT.....	309
Chapter 11	DRAM Controllers and Interface	
Figure 11-1.	DRAM System Elements, XLS616 and XLS608	314
Figure 11-2.	DRAM System Elements, XLS4xx-Lite and XLS4xx Devices.....	315
Figure 11-3.	DRAM System Elements, XLS2xx and XLS1xx Devices.....	316
Figure 11-4.	Example Connections for 1 x 32b DDR2 SDRAM Interface	321
Figure 11-5.	Example XLS2xx Connections for 1 x 72b Single-Rank DDR2 SDRAM Interface	322
Figure 11-6.	Example of Three DRAM Port MB Regions in the Address Space	324
Figure 11-7.	Clock, Data and Command Shifting with Signal Group DLLs	327
Figure 11-8.	Signal Group DLLs for the Channel-Pair AB DRAM Interface	329
Figure 11-9.	Functional Blocks and Registers in Each DLL for a Signal Group	330
Figure 11-10.	Example Read Data Functional Timing Diagrams	338
Chapter 12	Fast Messaging Network	
Figure 12-1.	Fast Messaging Network Stations – XLS616 Processor Example	398
Figure 12-2.	Fast Messaging Network Packet Processing Example.....	399
Figure 12-3.	Message Sizes.....	400
Figure 12-4.	Message Station	401
Figure 12-5.	Each Bucket uses 32 Receive Queue Entries by Default	403
Figure 12-6.	One Bucket Uses All 256 Receive Queue Entries	404
Figure 12-7.	Mixed Bucket Sizes	404
Figure 12-8.	Message Station Overview	407
Chapter 13	Networking Accelerators	
Figure 13-1.	Packet Flow in the XLS Processor.....	422
Figure 13-2.	Packet Data Movement on the Distributed Interconnects.....	424
Figure 13-3.	Network Accelerator Register Layout	425
Figure 13-4.	Memory Map for the Networking Accelerator Registers	426
Figure 13-5.	Networking Accelerator Connecting to the Fast Messaging Network	427
Figure 13-6.	Pointer-to-Data (P2D) Type Packet Descriptors	438
Figure 13-7.	Pointer-to-Pointer (P2P) Type Packet Descriptors	439
Figure 13-8.	Pointer-to-Data (P2D) Type Packet Descriptors	440
Figure 13-9.	Parser — Processing of Packet Header	442
Figure 13-10.	Parser — L3CTable Lookup	444

Figure 13-11.	Parser — L4CTable L4 Field Extraction	445
Figure 13-12.	Classification Vector	446
Figure 13-13.	Setting Checksum Result Bits	458
Figure 13-14.	Packet Director	459
Figure 13-15.	Selector Index Modifier Details	461
Figure 13-16.	Packet Distribution Engine	468
Figure 13-17.	RxPacket	488
Figure 13-18.	Transmit Data FIFO Configuration.....	508

Chapter 14 Security Acceleration Engine

Figure 14-1.	XLS Security Acceleration Engine	517
Figure 14-2.	Crypto Core Block Diagram	518
Figure 14-3.	RSA Core	521
Figure 14-4.	Security Engine Packet Flow - Example.....	523
Figure 14-5.	Security Messaging Block Diagram.....	524
Figure 14-6.	Cipher and HMAC Keys	530
Figure 14-7.	Cipher and HMAC Keys	533
Figure 14-8.	Typical IPsec ESP Packet	536
Figure 14-9.	512-bit boundary mode (SW padding) with 147-byte message	568
Figure 14-10.	512-bit boundary mode (SW padding) with 56-byte message	569
Figure 14-11.	8-bit boundary mode (HW padding)	569
Figure 14-12.	Tunneling an IP Packet	570
Figure 14-13.	In-Memory Representation	570
Figure 14-14.	Cipher Destination	571
Figure 14-15.	Authentication Destination	571

Chapter 15 Compression/Decompression Engine

Figure 15-1.	Compression/Decompression Engine Block Diagram.....	573
Figure 15-2.	CDEENG Architecture	575
Figure 15-3.	Interrupt Model for Compression/Decompression Engine.....	580

Chapter 16 SGMII and RGMII Interface

Figure 16-1.	High-Level View of GMAC Interface in XLS Architecture	600
Figure 16-2.	GMAC Architecture.....	602
Figure 16-3.	Internal Details of SGMII/RGMII Port in GMAC Mode.....	603
Figure 16-4.	High-Level View of XLS RGMII Port A.....	606
Figure 16-5.	Port RA FIFO Mode Block Diagram.....	607
Figure 16-6.	Bus Encoding and Timing	607
Figure 16-7.	Generation of Backpressure Based on Async FIFO and RxFIFO Occupancy	609
Figure 16-8.	State Diagram for Backpressure	610
Figure 16-9.	Async FIFO and Clocking Scheme in FIFO Mode — FIFO Mode (Async).....	611
Figure 16-10.	Async FIFO and Clocking Scheme in FIFO Mode — FIFO mode (sync)	611
Figure 16-11.	Accessing PCS Registers in SGMII.....	616
Figure 16-12.	MAC Filter Control Logic Controlled by MAC_FILTER_CONFIG Register.....	637

Chapter 17 XAUI Interface

Figure 17-1.	XAUI Lane	676
Figure 17-2.	XAUI Link.....	676
Figure 17-3.	XAUI Ports	678
Figure 17-4.	XLS616 and XLS608 Block Diagram.....	680

Figure 17-5.	XLS416 and XLS408 Block Diagram	681
Figure 17-6.	XAUI Interface Access to XLS Internal Architecture	682
Figure 17-7.	Internal Details of an XLS XAUI Port	683

Chapter 18 USB Interface

Figure 18-1.	XLS Processor USB Interface.....	737
Figure 18-2.	Interface as USB Device	738
Figure 18-3.	Interface as USB Host Controller.....	738
Figure 18-4.	Device Controller Interrupt Hierarchy.....	741
Figure 18-5.	USB Host Controller Main Modules - (XLS1xx devices use Port 0 only)	742
Figure 18-6.	Interrupt Model for USB Interface and System Architecture	745
Figure 18-7.	Processing a SETUP Packet	755
Figure 18-8.	DMA Mode Bulk OUT Transaction.....	764
Figure 18-9.	DMA Mode Bulk IN Transfer	776
Figure 18-10.	Bulk IN Stall	778
Figure 18-11.	Bulk IN DMA mode with Thresholding	780
Figure 18-12.	Two-Stage Control Transfer.....	784
Figure 18-13.	Location of the USB Interface Within the XLS Addressing Space	791
Figure 18-14.	USB Device Controller Address Block Map	800
Figure 18-15.	USB Host Controller Address Block Map	841

Chapter 19 PCI Express™ Interface

Figure 19-1.	Internal XLS6xx and XLS4xx PCIe 4x1 and 1x4 Topology	847
Figure 19-2.	Internal XLS408Lite and XLS404Lite PCIe Topology	848
Figure 19-3.	Internal XLS208 and XLS204 PCIe 4x1 Topology.....	849
Figure 19-4.	Internal XLS108 and XLS104 PCIe 2x1 Topology.....	849
Figure 19-5.	PCI Express Link Configurations, XLS616 and XLS4xx Devices	850
Figure 19-6.	PCI Express Link Configurations, XLS408Lite and XLS404Lite Devices	851
Figure 19-7.	PCI Express Link Configurations (4x1) for XLS2xx Devices.....	852
Figure 19-8.	PCI Express Link Configurations (2x1) for XLS1xx Devices.....	853
Figure 19-9.	EP Mode PCIe core Configuration Space Layout.....	923
Figure 19-10.	Example Base Address Register Configuration.....	937
Figure 19-11.	RC Mode PCIe core Configuration Space Layout	953
Figure 19-12.	Example Base Address Register Configuration.....	963

Chapter 20 Serial-RapidIO Interface

Figure 20-1.	Internal XLS6xx and XLS4xx SRIO 4x1 and 1x4 Topology	984
Figure 20-2.	Transaction Queues.....	991
Figure 20-3.	Input/Output Transaction Queue Entry (Little Endian)	992
Figure 20-4.	MESSAGE Transaction Queue Entry (Little Endian)	994
Figure 20-5.	DOORBELL Transaction Queue Entry Format (Little Endian)	995
Figure 20-6.	MAINTENANCE Transaction Queue Entry (Little Endian)	995
Figure 20-7.	Port-Write Transaction Queue Entry (Little Endian)	996
Figure 20-8.	Status Queue	997
Figure 20-9.	Status Queue Entry (Little Endian)	998
Figure 20-10.	Free-Lists	999
Figure 20-11.	Free-List Entry (Little Endian)	999
Figure 20-12.	Mailbox Queues (Limited to four) for Received MESSAGEs	1000
Figure 20-13.	Mailbox Queue Entry (Little Endian)	1001
Figure 20-14.	Payload Data Structure (Little Endian)	1002
Figure 20-15.	Packet Capture 0 CSR - Bits 0-15	1065

Chapter 21 UARTs, I²C, and Peripherals Interface

Figure 21-1.	The Peripherals Interface on the I/O Distributed Interconnect Ring	1067
Figure 21-2.	Memory Map of the UART/I ² C Portion of the Peripheral Interface	1070
Figure 21-3.	I ² C Bus Normal Transfer	1073
Figure 21-4.	Device Address Disabled	1073
Figure 21-5.	Address Disabled	1074
Figure 21-6.	I ² C Bus Address Only Transfer	1074
Figure 21-7.	I ² C Bus Address Only with No Device Address Transfer	1074
Figure 21-8.	Sequential Transfer	1075

Chapter 22 Flash/PCMCIA Memory and the Peripherals Interface

Figure 22-1.	NAND Flash Memory Device Block Diagram	1092
Figure 22-2.	Interfacing Flash Memory to the XLS	1094
Figure 22-3.	NAND Flash Connection to XLS (8-bit NAND Device)	1100
Figure 22-4.	NAND Flash Access Diagram (Read Operation Shown)	1101
Figure 22-5.	Memory Map of the Flash ROM Interface	1104
Figure 22-6.	Write Timing for COMMAND and Address Transfer	1117
Figure 22-7.	Read Timing for COMMAND and Address Transfer	1118

Chapter 23 GPIO and the Peripherals Interface

Figure 23-1.	GPIO Sub-Region in the Peripherals & I/O Configuration Region	1122
Figure 23-2.	Memory Map of the GPIO Interface	1122

Chapter 24 IEEE 1588 Precision Time Protocol

Figure 24-1.	PTP Controller Logic Diagram	1159
--------------	------------------------------------	------

Chapter 25 XLS Debugging Features

NETLOGIC
CONFIDENTIAL



List of Tables

Chapter 1	XLS™ Processor Family Introduction	
Table 1-1.	The XLS6xx, XLS4xx-Lite, XLS4xx, XLS2xx and XLS1xx Processor Families	36
Table 1-2.	Instruction Set Additions	46
Table 1-3.	Priority Thread Scheduling Example 1	50
Table 1-4.	Priority Thread Scheduling Example 2	51
Table 1-5.	Primary Cache Parameters	53
Chapter 2	XLS MIPS64 CPU Core	
Table 2-1.	Instruction Timing	70
Chapter 3	Virtual MIPS Mode	
Chapter 4	XLS Processor Core Registers	
Table 4-1.	Bit Field Definitions	78
Table 4-2.	Coprocessor 0 Register Summary.....	79
Table 4-1.	Performance Counter Event Types	119
Table 4-2.	Coprocessor 2 Register Summary.....	127
Table 4-3.	XLS Processor Control Registers Summary.....	136
Table 4-4.	Fields for Debug Accesses to ICU Arrays	143
Chapter 5	XLS CPU Instruction Set	
Table 5-1.	MIPS64 CPU Instruction Summary	161
Table 5-2.	XLS Processor MIPS64 Extensions	165
Table 5-3.	Usage of Effective Address	166
Table 5-4.	Encoding of Bits[17:16] of CACHE Instruction	167
Table 5-5.	Encoding of Bits [20:18] of the CACHE Instruction	168
Table 5-6.	Values of the hint Field for the PREF Instruction.....	170
Chapter 6	XLS L1 Cache	
Table 6-1.	Primary Cache Parameters	189
Table 6-2.	Field Modifiability Codes.....	191
Table 6-3.	L1 Cache Control Registers List.....	191
Chapter 7	XLS Unified L2 Cache	
Table 7-1.	Field Modifiability Codes.....	209
Table 7-2.	Address Format for Accessing L2 Cache Registers	209
Table 7-3.	L2 Cache Control Registers Summary and Addressing	210
Table 7-4.	Command Operation Field [3:0]	213

Table 7-5.	Performance Counter (Pbits) Event Selection	213
Chapter 8	Memory and I/O Access	
Table 8-1.	Peripheral and I/O Configuration Region of Memory	226
Table 8-2.	System Bridge Controller Register Summary	229
Table 8-3.	Key to Address Processing Symbols Used in Example.....	240
Table 8-4.	DRAM Example	240
Chapter 9	Direct Memory Access (DMA) Engine	
Table 9-1.	Message Formats	279
Table 9-2.	DMA Engine Register Summary	287
Table 9-3.	DMA Credit Counter Registers	299
Table 9-4.	DMA MsgBucketSize Registers	299
Chapter 10	Programmable Interrupt Controller	
Table 10-1.	Interrupt Redirection Table Entry	305
Table 10-2.	Trigger-Mode/Polarity Combinations	306
Table 10-3.	PIC Interrupt Redirection Table Map	307
Table 10-4.	List of PIC Registers	310
Chapter 11	DRAM Controllers and Interface	
Table 11-1.	Supported DDR2 DRAM Configurations.....	316
Table 11-2.	Comparison of Supported DRAM Types in XLS Processors	318
Table 11-3.	Memory Device Types and Channel Configurations.....	318
Table 11-4.	Generic DDR Interface Signal Descriptions for Ports MA - MD	319
Table 11-6.	Memory Controller Register Region Offsets	323
Table 11-7.	Configuration of DDR Signals with DLLs and Their Registers.....	331
Table 11-8.	DRAM Memory Controller Commands	336
Table 11-9.	Example DRAM Read Data Timing Parameters	339
Table 11-10.	Suggested Initialization Sequences	341
Table 11-11.	DDR2 Termination Values for Read/Write and Different Configurations	342
Table 11-12.	DDR Page Policies	343
Table 11-13.	Memory Controller Register Summary	345
Chapter 12	Fast Messaging Network	
Table 12-1.	Stations and Addressable Buckets on the Fast Messaging Network.....	402
Table 12-2.	XAUI/SGMIII MsgBucketSize Registers	409
Table 12-3.	Security Acceleration Engine MsgBucketSize Registers	410
Table 12-4.	DMA MsgBucketSize Registers	410
Table 12-5.	CDE MsgBucketSize Registers (XLS6xx, XLS4xx-Lite and XLS4xx devices).....	410
Table 12-6.	PCIe MsgBucketSize Registers	410
Table 12-7.	Credit Counter Registers	411
Chapter 13	Networking Accelerators	
Table 13-1.	Network Accelerator Sub-Regions within the Peripherals & I/O Configuration Region .	425
Table 13-2.	Packet Director Classification Modes	429
Table 13-3.	Rx Packet Descriptor Format	433
Table 13-4.	Rx Packet Descriptor Status Field Values for GMACs	434

Table 13-5.	Tx Packet Descriptor Pointer-to-Data (P2D) Format	435
Table 13-6.	Tx Packet Descriptor Pointer-to-Pointer (P2P) Format.....	437
Table 13-7.	Free Back Packet Descriptor Format.....	441
Table 13-8.	Free In Packet Descriptor Format.....	441
Table 13-9.	Parser Register Summary	447
Table 13-10.	Packet Director Classification Modes	460
Table 13-11.	TCAM Key and Mask Set-up Procedure.....	463
Table 13-12.	Parser Key Bit Match Results	464
Table 13-13.	CAM4x128Table Register Addresses.....	470
Table 13-14.	CAM4x128Key Register Addresses	471
Table 13-15.	CAM4x128Table Register Addresses.....	472
Table 13-16.	Prepad Format '00'	474
Table 13-17.	Prepad Format '01'	476
Table 13-18.	Prepad Format '10'	477
Table 13-19.	Prepad Format '11'	479
Table 13-20.	Networking Accelerator Register Summary.....	483

Chapter 14 Security Acceleration Engine

Table 14-1.	Crypto Core Raw Performance (at Processor Clock Speed).....	519
Table 14-2.	Raw ARC4 Performance in Stateless Case	520
Table 14-3.	ECC (Elliptic Curve Cryptography) Performance Characteristics.....	521
Table 14-4.	Control Descriptor.....	526
Table 14-5.	Control Descriptor Vector	526
Table 14-6.	Data Descriptor.....	527
Table 14-7.	Data Descriptor Vector	527
Table 14-8.	Cipher Instruction	528
Table 14-9.	Cipher Instruction Fields	528
Table 14-10.	Data Descriptor Structure	536
Table 14-11.	Segment Source Address Data Structure (srcLengthIVOffUseIVNext)	537
Table 14-12.	Cipher Destination (dstDataSettings)	539
Table 14-13.	Authentication Destination Address (authDstNonceLow)	540
Table 14-14.	Checksum Destination Address (ckSumDstNonceHiCFBMaskLLWMask)	540
Table 14-15.	Security Free Out Descriptor	542
Table 14-16.	Control Error Conditions	543
Table 14-17.	Data Error Conditions	543
Table 14-18.	RSA / ECC Control Descriptor	545
Table 14-19.	RSA / ECC Control Descriptor Vector	545
Table 14-20.	RSA / ECC Data Descriptor Vector	547
Table 14-23.	Security Engine Return Message	553
Table 14-24.	RSA / ECC Control Error Conditions	554
Table 14-25.	RSA Core Data Error Conditions	554
Table 14-26.	Security Acceleration Engine Registers Summary	556
Table 14-27.	Security to DMA Credit Register Format	564

Chapter 15 Compression/Decompression Engine

Table 15-1.	SRC_ADDR Descriptors List	585
Table 15-2.	Field Modifiability Codes	588
Table 15-3.	Compression/Decompression Engine Configuration Registers.....	588

Chapter 16 SGMII and RGMII Interface

Table 16-1.	States	607
-------------	--------------	-----

Table 16-2.	Field Modifiability Codes	614
Table 16-4.	GMAC Registers.....	617
Table 16-5.	Hash Table Vector Registers.....	638
Table 16-6.	GMAC Statistics Registers.....	639
Table 16-7.	PCS-Specific Registers.....	666
Table 16-8.	Jitter Pattern Select Encoding.....	673

Chapter 17 XAUI Interface

Table 17-1.	Field Code Description.....	686
Table 17-2.	XAUI Register Map	687
Table 17-3.	XAUI Statistics Registers	705
Table 17-4.	XAUI PCS-Specific Registers	732

Chapter 18 USB Interface

Table 18-1.	Host Mode Configurations	739
Table 18-2.	USB Interface Signals (Per Port) - (XLS1xx devices use Port 0 only)	739
Table 18-3.	Transmit FIFO RAM Allocation	786
Table 18-4.	Register Field Access	789
Table 18-5.	Physical Address Fields.....	790
Table 18-6.	Summary of USB Interface General Registers	792
Table 18-7.	Register Address Fields for Accessing USB Interface in Device Mode	800
Table 18-8.	USB Device Controller CSR Summary	801
Table 18-9.	Register Field Access	803
Table 18-11.	Minimum Duration for Soft Disconnect	824
Table 18-12.	Device IN Endpoint 0 Transfer Size Register: UDC_DIEPTSIZ0	837
Table 18-13.	Device OUT Endpoint 0 Transfer Size Register: DOEPTSIZ0	837
Table 18-14.	Register Address Fields for Accessing Host Mode USB General Register	841
Table 18-15.	Register Address Fields for Accessing Host Mode Controller Register.....	841

Chapter 19 PCI Express™ Interface

Table 19-1.	PCI Express Resources for Each XLS Device.....	846
Table 19-2.	PCI Express Link Configurations for XLS6xx and XLS4xx Devices	850
Table 19-3.	PCI Express Link Configurations, XLS408Lite and XLS404Lite	851
Table 19-4.	PCI Express Link Configurations, XLS2xx	852
Table 19-5.	PCI Express Link Configurations, XLS1xx	853
Table 19-6.	Physical Address Fields.....	855
Table 19-7.	Physical Address Fields.....	856
Table 19-8.	Physical Address Fields.....	857
Table 19-9.	DMA Message Fields.....	859
Table 19-10.	Message Out Fields	859
Table 19-11.	Field Read/Write Access Codes	864
Table 19-12.	Summary of the XLS Families PCI Express Interface Global Registers	864
Table 19-13.	Summary of XLS408Lite and XLS404Lite PCI Express Interface Global Registers.....	900
Table 19-14.	PCI Configuration Space Header - Type 0	924
Table 19-15.	Configuration Structure: Starting Addresses and Next Capability Pointers	925
Table 19-16.	Power Management Capability Structure	925
Table 19-17.	MSI Capability Structure	925
Table 19-18.	PCI Express Capability Structure.....	925
Table 19-19.	Advanced Error Reporting Capability Structure	926
Table 19-20.	Device Serial Number Capability Register.....	926
Table 19-21.	Configuration Register Bit-Field Types	927

Table 19-22.	PCI Configuration Space Header - Type 1	954
Table 19-23.	Configuration Structure: Starting Addresses and Next Capability Pointers.....	954
Table 19-24.	Power Management Capability Structure	954
Table 19-25.	MSI Capability Structure	955
Table 19-26.	PCI Express Capability Structure	955
Table 19-27.	Advanced Error Reporting Capability Structure.....	956
Table 19-28.	Device Serial Number Capability Register.....	956
Table 19-29.	Configuration Register Bit-Field Types.....	957

Chapter 20 Serial-RapidIO Interface

Table 20-1.	Pinstrap Selection of PCIe or SRIO for the XLS6xx and XLS4xx.....	985
Table 20-2.	SRIO Pinstrap Lane Configuration for the XLS6xx and XLS4xx Devices.....	985
Table 20-3.	Physical Address Fields for Interface Registers	986
Table 20-4.	Physical Address Fields for Configuration and Control Registers	986
Table 20-5.	XLS6xx and XLS4xx Supported SRIO Transactions.....	987
Table 20-6.	Supported PDU Sizes for Generated Requests	988
Table 20-7.	Supported Transaction Requests	992
Table 20-8.	Legal Transaction Sizes and Register Offset Address	996
Table 20-9.	Summary of XLS6xx and XLS4xx SRIO Interface Global Registers	1003
Table 20-10.	Field Modifiability Codes.....	1003
Table 20-12.	Summary of XLS6xx and XLS4xx SRIO Interface Global Registers	1009
Table 20-13.	Summary of XLS6xx and XLS4xx SRIO Implementation Defined Registers.....	1018
Table 20-14.	Logical Layer Response Timeout Pre-scale value	1025

Chapter 21 UARTs, I²C, and Peripherals Interface

Table 21-1.	UART Ports 1 and 2 Signal Descriptions.....	1068
Table 21-2.	I ² C Ports 1 and 2 Signal Descriptions	1069
Table 21-3.	UART/I ² C Sub-Regions in the Peripherals & I/O Configuration Region	1070
Table 21-4.	Summary of UART and I ² C Interface Registers	1076
Table 21-5.	Interrupt Identification Properties.....	1078

Chapter 22 Flash/PCMCIA Memory and the Peripherals Interface

Table 22-1.	External Device Connections to Peripheral IO Bus (Bus Functional Modes)	1095
Table 22-2.	Peripherals Input/Output Bus Port Signal Descriptions	1096
Table 22-3.	GPIO Port Signals and Flash Memory.....	1096
Table 22-4.	Initialization Options Sensed at Reset Time	1098
Table 22-5.	Flash ROM Registers in the Peripherals & I/O Configuration Region	1104
Table 22-6.	Summary of Flash/PCMCIA Memory Interface Registers	1105

Chapter 23 GPIO and the Peripherals Interface

Table 23-1.	GPIO Port and Flash Memory Signal Descriptions.....	1119
Table 23-2.	ProcessorID Value Locations in GPIO Register	1121
Table 23-3.	Summary of GPIO Interface Registers	1123
Table 23-4.	Typical DRAM AB PLL Ratio Values	1133
Table 23-5.	PLL Prescale [1:0] Decoding	1137
Table 23-6.	MPLL Divider Settings	1137
Table 23-7.	GPIO Reset Configuration Register	1141

Chapter 24 IEEE 1588 Precision Time Protocol

Table 24-1.	IEEE1588 PTP Register Summary	1162
-------------	-------------------------------------	------

Chapter 25 XLS Debugging Features

Table 25-1.	IDCODE Data Format.....	1168
Table 25-2.	XLS ProcessorID and RevisionID Values per Product	1168
Table 25-3.	Address Error Registers in System Bridge Controller.....	1177
Table 25-4.	Trace Buffer Register Space.....	1180
Table 25-5.	DebugDescWordCountsSel Register SEL Field.....	1193

Appendix A Multi-Threading Advantages**Appendix B Revision History**



Chapter 1 XLS™ Processor Family Introduction

1.1 Introduction

As communications solutions such as content-aware networking, multiservice Firewall / VPN / IDS systems, and new forms of storage management continue to converge, the need for flexible and cost-effective processing is rapidly increasing. Today's system designs, based on a growing mix of semi-programmable ASICs, NPUs, and coprocessors, have created a hardware and software environment that is increasingly complex, expensive and difficult to scale. The solution lies in high-performance general-purpose processing with intelligent system and functional integration that, until now, has been unavailable.

The XLS Processor Family, a value-based throughput-optimized processor, combines the power of a multiprocessing, multi-threaded XLR™ Core architecture with the simplicity of a leading edge, general-purpose MIPS64® engine, enabling wire speed, software-driven applications across multiple NetLogic-based™ platforms. The XLS Processor Family based on the same architecture as the industry leading XLR™ Processor family, extends the XLR Processor's benefits of multi-threading for optimized connected computing to the cost sensitive small enterprise, ROBO, SOHO and other system solutions while maintaining software compatibility with XLR-based solutions.

A programmable SuperSoC™ solution built for a C/C++ environment, the XLS Processor eliminates the need for microcoding or proprietary scripting enabling rapid application development and leveraging widely available, well understood development tools.

The XLS Processor's extensive set of integrated connectivity options as shown in [Table 1-1](#) simplifies design and lowers cost, power and PCB area. The XLS Processor incorporates multiple Autonomous Acceleration Engine® innovations. The Autonomous Network Acceleration Engine® functionality empowers the network interface. In addition, security acceleration is also a standard feature with up to 2.5 Gbps of DES/3DES, AES, ARC4, Kasumi, SHA, MD5, RSA, and ECC to support in-line IPSec, SSL, and other secure protocol processing. An autonomous compression engine (except for the XLS2xx and XLS1xx devices) provides the support to achieve the ultimate in performance of payload inspection for XLS-based anti-spam, anti-virus and similar applications.

As a true programmable solution, it eliminates the need for microcoding or proprietary scripting. In addition, its industry-standard media interfaces provide an extensive set of connectivity options. The XLS Processor Family is a cost-effective, single-chip solution providing a key building block for next-generation, scalable multiservice systems. The XLS Processor is offered as a complete family of five pin-compatible processors spanning performance and bandwidth to enable multiple applications with a single board design. [Table 1-1](#) lists the functional capabilities of the XLS Processor Family members.

1.1.1

Versatile Next Generation XLS™ Thread Processor Core

There are fundamental bottlenecks preventing today's new processors from delivering performance improvements meaningful to system designers. Current processor design approaches to overcome these barriers include deeper pipelines and superscalar (multi-issue) designs. However, these efforts are yielding diminishing returns. The XLS Processor Family overcomes these concerns with a set of innovative, MIPS-Based™ cores, which are

throughput-optimized and built for maximum flexibility and scalability. Each XLS™ core consists of four nCPU™ NetLogic virtual CPU devices. Each nCPU occupies one of the four tightly integrated and performance optimized hardware threads on the XLS core. Unlike single-threaded, multi-issue designs, this architecture is well suited to today's multi-threaded, network-oriented computing environment.

The flexibility of multi-threading in the XLS Processor™ family can be used in a variety of ways to accommodate specific application and software requirements. Consider the case in network oriented applications where there is a high sensitivity to memory latencies. In this environment, when one task must wait on memory, other tasks running on the nCPU in the same XLS core can continue processing additional data, thus mitigating latencies and dramatically improving overall throughput. Multi-threading may also be used in multiple combinations and be assigned to replace discrete co-processor functionality. In more compute-intensive applications, a single core can combine all processor threads into a single-threaded mode, thus applying all available clock cycles of a given core for a single focused task.

By providing a flexible, general-purpose architecture that can easily adapt to changing system design and application needs, the versatile XLS Thread Processor™ family enables NetLogic Scalable Processor Solutions™ products to deliver a broad set of performance and implementation options across a wide range of system offerings. The components of the nine (seven pin-compatible) XLS Processor Family members are shown in [Table 1-1](#).

Table 1-1. The XLS6xx, XLS4xx-Lite, XLS4xx, XLS2xx and XLS1xx Processor Families

Feature	XLS616	XLS608	XLS416	XLS408-Lite XLS408	XLS404-Lite XLS404	XLS208	XLS204	XLS108	XLS104
Threads/nCPUs	16	8	16	8	4	8	4	8	4
XLS Cores	4	2	4	2	1	2	1	2	1
L2 Cache	1MB	1MB	1MB	1MB	512 KB	512 KB	256 KB	512 KB	256 KB
Max. Operating Freq.	1.2 GHz	1.0 GHz	1.0 GHz	750 MHz	750 MHz				
Security Acceleration	2.5 Gbps	2.5 Gbps	2.5 Gbps	2.5 Gbps	1.25 Gbps	1.0 Gbps	1.0 Gbps	750 Mbps	750 Mbps
Compression Acceleration	2.5 Gbps	2.5 Gbps	2.5 Gbps	2.5 Gbps	1.25 Gbps	—	—	—	—
DDR2 SDRAM Channels	4 x 36b 2 x 72b	4 x 36b 2 x 72b	2 x 36b 1 x 72b	2 x 36b 1 x 72b	2 x 36b 1 x 72b	1 x 36b 1 x 72b	1 x 36b 1 x 72b	1 x 36b	1 x 36b
Ethernet SGMII Ports 10/100/1000 Mbps	8 ^a	4 ^b	4 ^b	3 ^c	3 ^c				
Ethernet RGMII Port 10/100/1000 Mbps	1 ^a	1 ^b	1 ^b	1 ^c	1 ^c				
10-Gbps XAUI Ports	2 ^d	2 ^d	2 ^d	1 ^e	—	—	—	—	—
USB Ports	2	2	2	2	2	2	2	1	1
PCI-Express Lanes	1x4 or 4x1	1x4 or 4x1	1x4 or 4x1	1x4 or 4x1 ^f	1x4 or 4x1 ^f	4 x 1	4 x 1	2 x 1	2 x 1
SRIO Ports	1x4 or 4x1 ^g	—	—	—	—				
BGA Package	957 pins	957 pins	845 pins	845 pins	845 pins	845 pins	845 pins	845 pins	845 pins

- a. The XLS6xx, XLS4xx-Lite and XLS4xx support eight Gigabit Ethernet ports (two groups of four) that can be allocated as either eight SGMII ports, or one RGMII port and seven SGMII ports.
- b. The XLS208 and XLS204 support four Gigabit Ethernet ports which collectively can be allocated as either four SGMII ports, or one RGMII port and three SGMII ports.
- c. The XLS108 and XLS104 support three Gigabit Ethernet ports which can be allocated as either three SGMII ports, or one RGMII port and two SGMII ports.
- d. The XLS6xx and XLS416 allow each group of four Ethernet ports to be optionally configured as a 10-Gbps 4-lane XAUI link, for a total of up to two 4-lane XAUI links. The XLS408-Lite and XLS404-Lite do not support XAUI.
- e. The XLS408 allows Ethernet Quad[4-7] to be optionally configured as the 10-Gbps 4-lane XAUI_1 link.

- f. The XLS408-Lite and XLS404-Lite support either 1x4 or 2x1 PCIe links.
- g. The XLS6xx and XLS4xx allow the PCIe ports to be optionally configured as a 4x1 or 1x4 SRIO ports; the XLS408-Lite and XLS404-Lite do not support the SRIO Interface.

NETLOGIC
CONFIDENTIAL

1.2

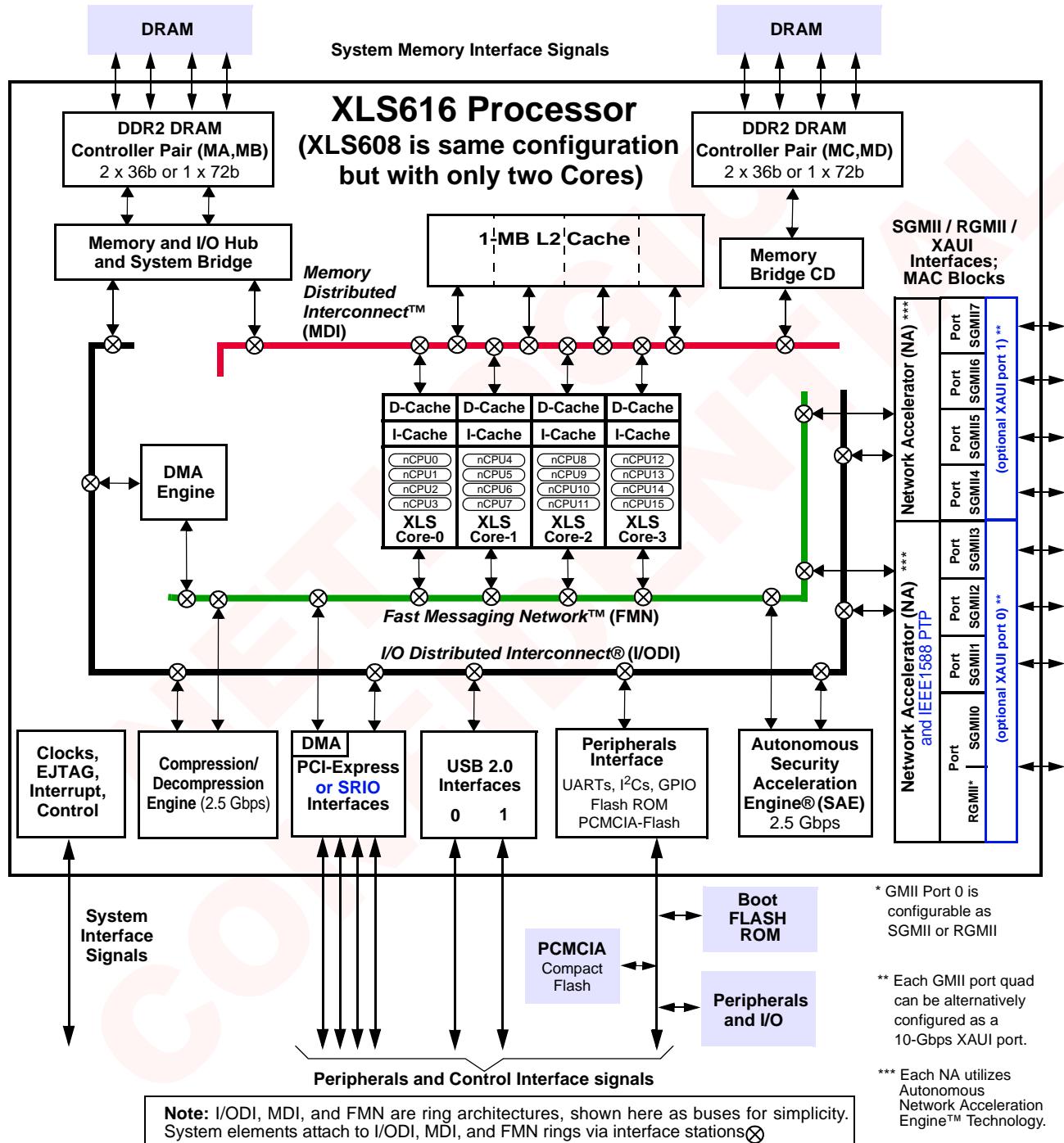
Key Features of the XLS™ Processor Family

Next Generation XLS Cores <ul style="list-style-type: none"> • 64-bit MIPS64 with enhancements • Up to 16 nCPUs - 4 nCPUs per core • 1, 2, or 4 cores--each core is 4-way multi-threaded • Scales to 1.2-GHz • Branch prediction • Auto-alignment of Load-Store addresses without exception generation • Flexible thread enable mode 	Extensive Networking Interfaces <ul style="list-style-type: none"> • Up to 8 Ethernet SGMII Interfaces • One RGMII Ethernet Interface <ul style="list-style-type: none"> – Optional FIFO mode operation instead of Ethernet MAC • One or two optional 4-lane 10-Gbps XAUI ports • 10/100/1000 MACs • Advanced networking hardware acceleration for all network interfaces
Cache Subsystem <ul style="list-style-type: none"> • Fully coherent, multi-level memory subsystem • L1 DedicatedPlus Cache™ • 32 KB ECC L1 data per XLS Core • 32 KB Parity L1 instruction per XLS Core • Up to 1 MB 4-way banked ECC L2 cache, operating at 1/2 core speed • All caches 8-way set associative 	High Performance Configurable Memory Controllers <ul style="list-style-type: none"> • Up to 800-MHz DDR2 DRAM with ECC • Up to four ECC-protected DRAM channels, configurable as four by 36 bits or two by 72 bits (see Table 1-1) • Four-channel DMA
High-Speed Distributed Interconnects <ul style="list-style-type: none"> • Connects all cores, caches and processing agents • Highly efficient on-chip interconnect provides high-speed internal connectivity and scalability • Fast Messaging Network™ for scalable communication btw key processing & I/O elements • High-speed Memory Distributed Interconnect™ network for fast data transfer • I/O Distributed Interconnect® network 	Integrated System Interfaces <ul style="list-style-type: none"> • PCI-Express® Controllers with 1x4 or 4x1 configuration (the XLS408-Lite/404-Lite support 1x4 or 2x1) • Optional 1x4 or 4x1 SRIO ports shared with PCIe port • Up to two USB 2.0 interfaces • PCMCIA interface • Bootable Flash memory interfaces (NAND or NOR) • Two I²C interfaces • Two 16550 UART interfaces • 32-bit GPIO interface • IEEE 1149.1 EJTAG and Memory BIST functionality
Networking Hardware Acceleration <ul style="list-style-type: none"> • Packet Distribution Engine™ for line rate processing • Flexible packet tagging, queuing and packet distribution management • Autonomous Network Acceleration Engine® technology • TCP checksum verification / generation 	Compression Engine (not in XLS2xx or XLS1xx) <ul style="list-style-type: none"> • Autonomous Operations (like Security Accel. Engine) • Up to 2.5 Gbps • ZLIB/Deflate/GZIP (RFC1950/1/2) • IPComp • 32-KB dictionary size • Fixed or Dynamic Huffman code
Leading Edge Security Acceleration Engine <ul style="list-style-type: none"> • Up to 2.5 Gbps of bulk encryption / decryption • Up to 2 high-speed crypto cores • DES / 3DES, AES (128, 192, 256), AES-GSM, ARC4, Kasumi-F8, F9 • MD5, SHA-1/256/384/512 (All HMAC) • RSA, ECC Exponentiation for SSL / IPsec • Random number generator • CRC 16/32 	General Purpose Programming <ul style="list-style-type: none"> • Virtual MIPS Mode isolates cores and enables virtualization of unmapped region • Supports both clustered and SMP modes • Fine and coarse grained scheduling modes / CPU • Parallel, Pipelined, & Hybrid processing modes • On-board debug support; performance monitoring • IEEE-1588 Programmable Timing Protocol support
Power Management <ul style="list-style-type: none"> • On-chip thermal sensor • Software-programmable clock throttling 	

1.2.1 XLS616 and XLS608 Block Diagram

The XLS616 is the highest-performance member of the XLS™ family of Communication Processor Solutions™ products. Figure 1-1 presents an example hardware block diagram for the quad-core XLS616 device. The XLS608 is identical, except it has only two Cores.

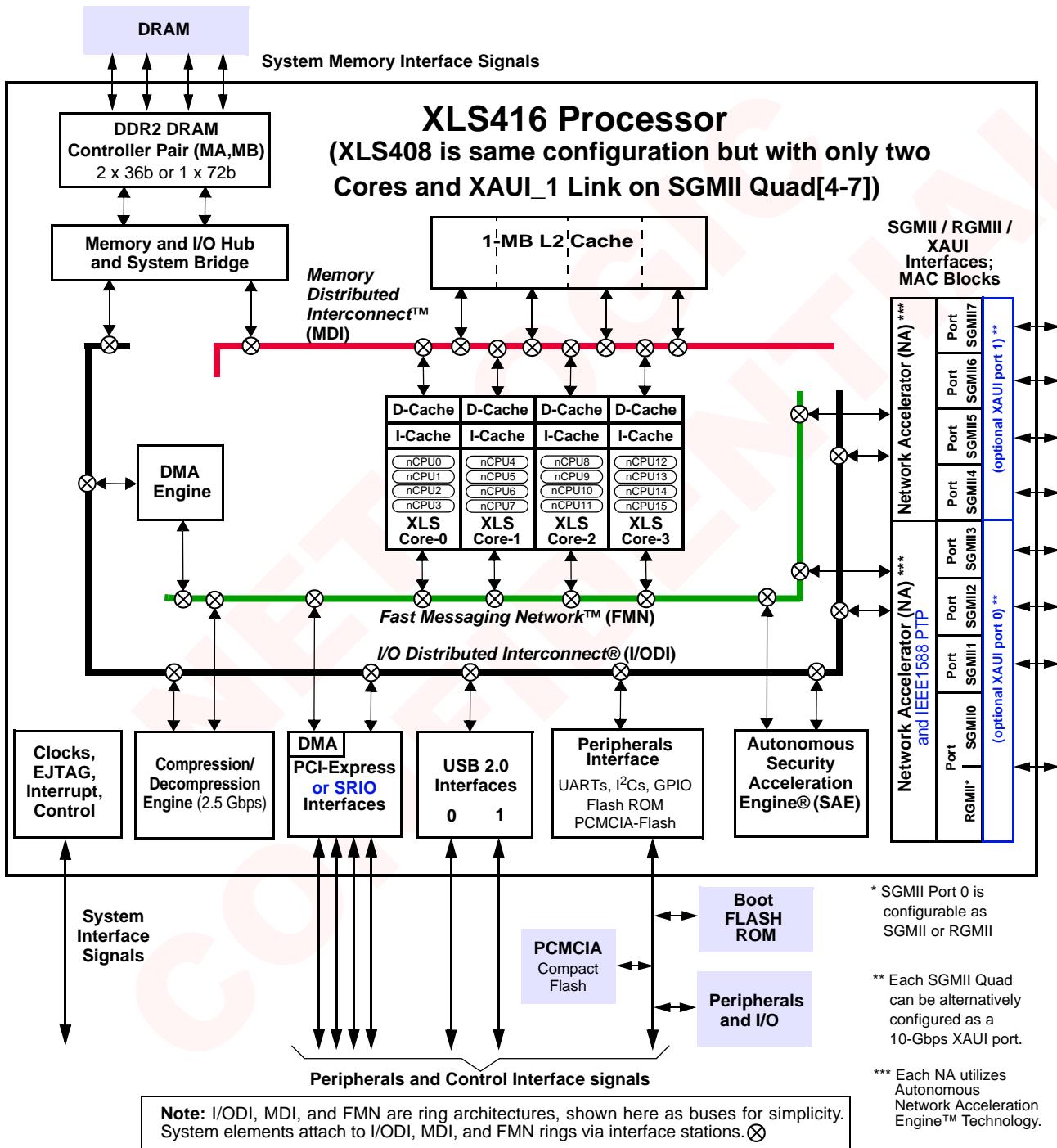
Figure 1-1. XLS616 and XLS608 Block Diagram



1.2.2 XLS416 and XLS408 Block Diagram

The XLS416 is identical to the XLS616 except for the number of DRAM channels supported. Figure 1-2 presents an example hardware block diagram for the quad-core XLS416 device. The XLS408 is the same as the XLS416, except it has only two cores and only one XAUI interface.

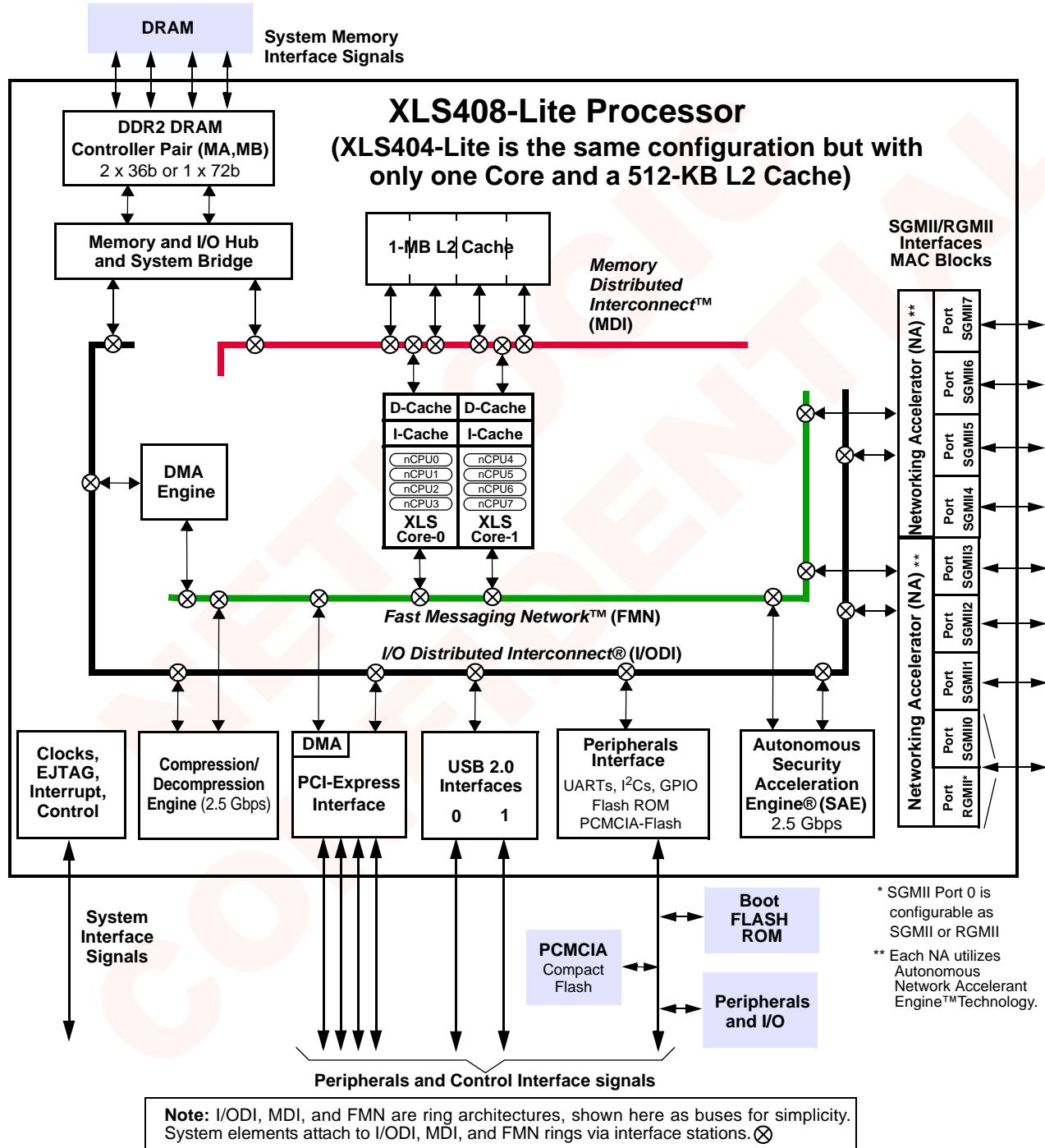
Figure 1-2. XLS416 and XLS408 Block Diagram



1.2.3 XLS408-Lite and XLS404-Lite Block Diagram

The dual-core XLS408-Lite is similar to the XLS408, but it has fewer I/O interfaces (no XAUI or SRIO interfaces), supports 2x1 or 1x4 PCIe and does not support IEEE-1588 PTP as shown in Figure 1-3. The XLS404-Lite is the same as the XLS408-Lite, except it has only one Core and a 512-KB L2 Cache.

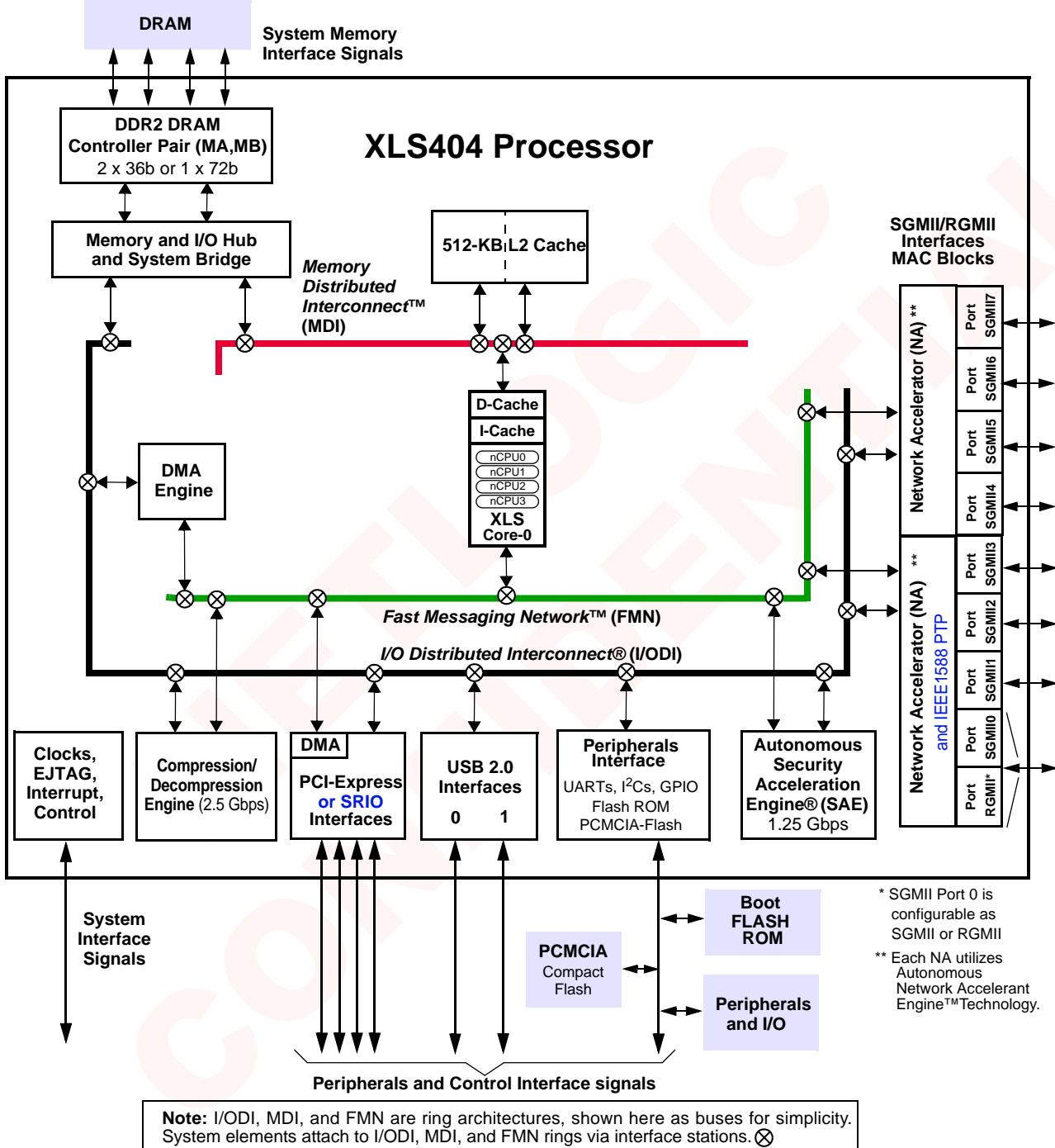
Figure 1-3. Diagram: XLS408-Lite and XLS404-Lite Block Diagram



1.2.4 XLS404 Block Diagram

The single-core XLS404 is similar to the XLS416 and XLS408, but it has one Core, no XAUI interface, and a 512-KB L2 Cache as shown in [Figure 1-4](#).

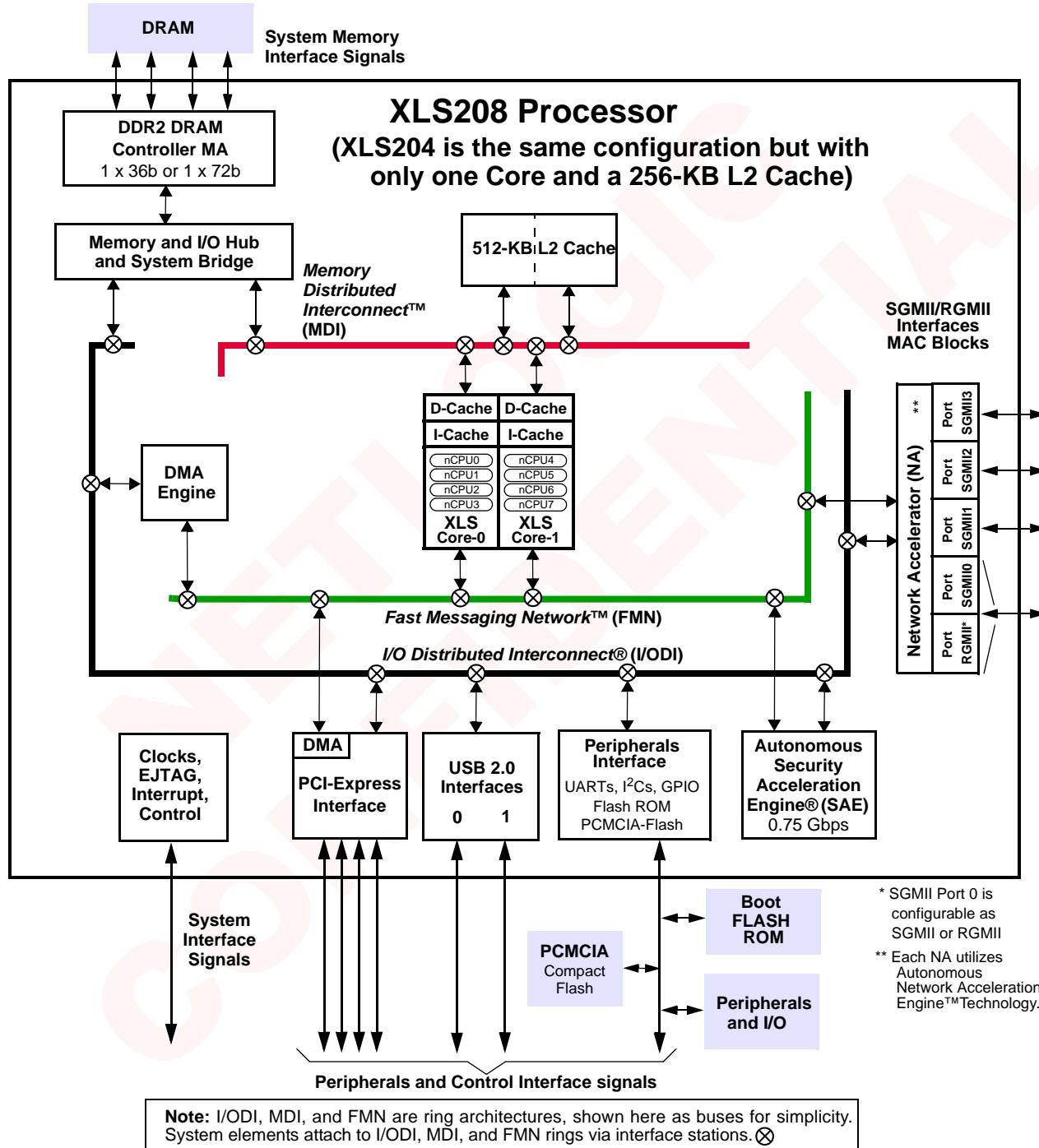
Figure 1-4. Diagram: XLS404 Block Diagram



1.2.5 XLS208 and XLS204 Block Diagram

The dual-core XLS208 is similar to the XLS408, but it has no Compression/Decompression Engine, only one SGMII/RGMII Quad, no SRIO Option, no IEEE1588 PTP, one memory 36-bit (or 72-bit) controller, and only 512-KB of L2 Cache as shown in [Figure 1-5](#). The XLS204 is the same as the XLS208, except it has only one Core and a 256-KB L2 Cache.

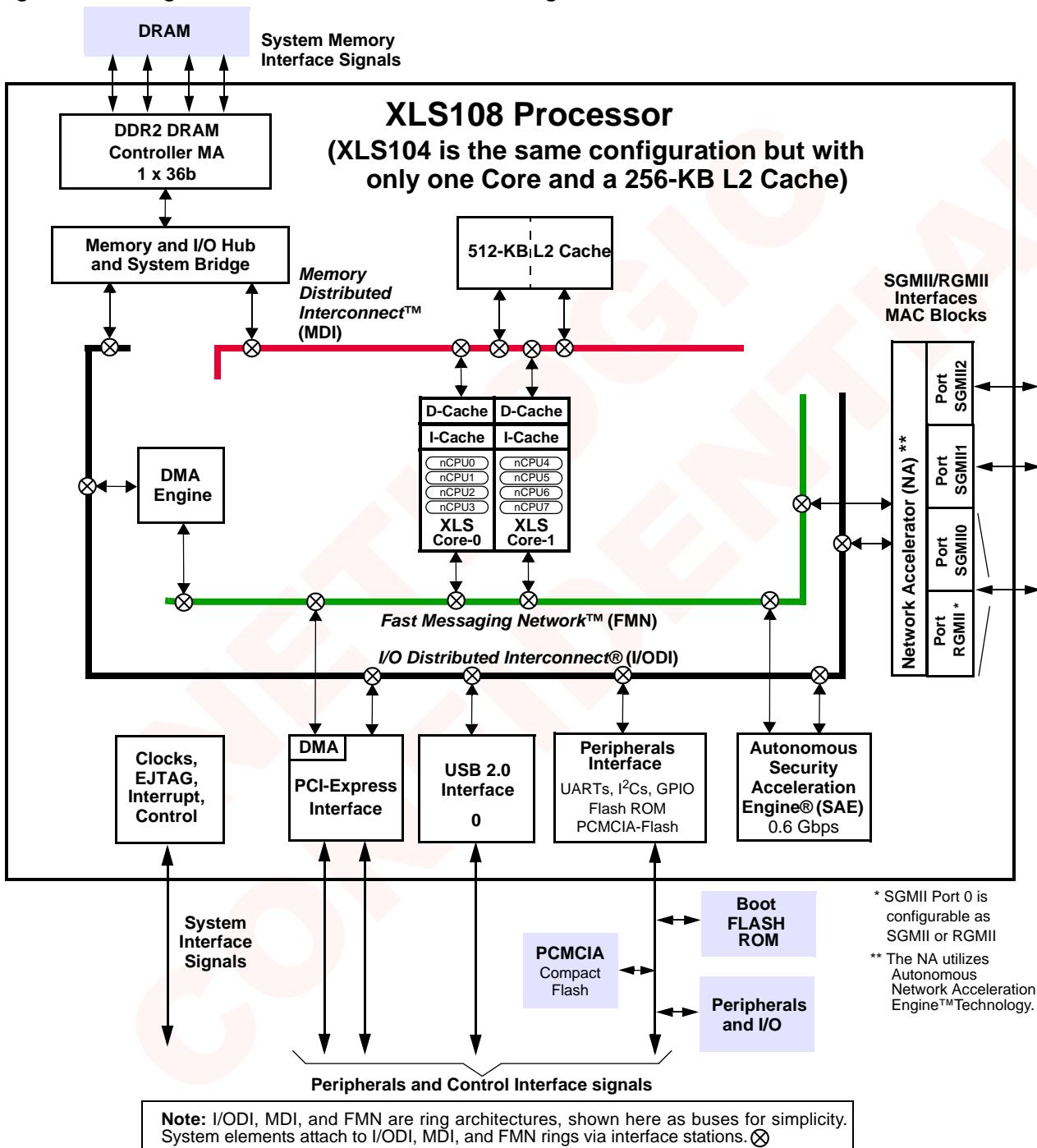
Figure 1-5. Diagram: XLS208 and XLS204 Block Diagram



1.2.6 XLS108 and XLS104 Architecture, Memory and I/O Diagram

The dual-core XLS108 is similar to the XLS208, but it has one 36-bit memory controller, three SGMII ports, one USB port, and one 2-Lane PCIe interface as shown in [Figure 1-6](#). The XLS104 is the same as the XLS108, except it has only one Core and a 256-KB L2 Cache.

Figure 1-6. Diagram: XLS108 and XLS104 Block Diagram



1.2.7 Low-Power Operation

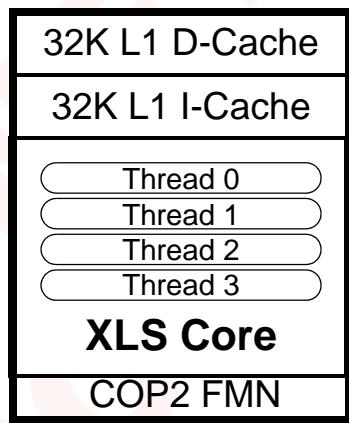
XLS Processors employ several techniques to deliver a low-power solution. The CPU core(s) run at the full core frequency. All other elements of the XLS device, such as the Memory Distributed Interconnect, IO Distributed Interconnect, Fast Messaging Network, L2-caches etc., operate at half the speed of the core clock. This approach results in delivering significant power savings without impacting the throughput of the XLS processor due to the large headroom built into the performance of all these components of the architecture. Additionally, XLS design uses extensive clock-gating techniques to deliver very low levels of overall power.

1.3 Enhanced CPU Core Features

The XLS architecture is based on a set of innovative cores enabling up to 16 threads in up to four identical 4-way multi-threaded XLS processor cores. Each core includes up to four tightly integrated hardware threads or nCPU™ NetLogic virtual CPU, which can be considered as virtual CPUs. Unlike single-threaded, multi-issue designs, this architecture is well suited to today's increasingly network-oriented computing environment.

The flexible nCPUs of the XLS Processor can be utilized in a variety of ways to increase execution efficiency and accommodate the specific needs of networking-centric applications. Consider the high sensitivity to memory latencies inherent to networking applications. In this environment, when one nCPU must wait on memory, the next nCPU in the core can immediately begin processing additional data, thus hiding latencies and dramatically improving overall throughput. XLS nCPUs can also be used in multiple combinations and be assigned to replace typical discrete coprocessor functionality. And in more compute-intensive applications, a single core can be programmed to dedicate all processing resources and available clock cycles to a single, focused task. The key attributes of XLS cores are shown in [Figure 1-7](#).

Figure 1-7. XLS Core Attributes



- 64-bit MIPS64 with XLS enhancements
- Up to four threads/nCPUs per core
- Up to 1.2 GHz operation
- Flexible thread-scheduling mode
- Branch Prediction – 2K-entry gshare branch history table
- Return Stack Buffer – 8-entry return stack buffer
- 64-Entry paired TLB's
- 32-KB Level-1 Instruction Cache per Core (with parity)
- 32-KB Level-1 Data Cache per Core (with ECC)
- EJTAG Debug Support
- Atomic Instructions
- Automatic Alignment of Load-Store addresses
- Extra scratch registers
- DedicatedPlus Cache

1.3.1 CPU Registers

The XLS cores are fully compliant with the MIPS64 ISA, release 1, which has a user-visible state consisting of 32 general-purpose registers, two special-purpose registers for integer multiplication and division, a program counter, and no condition code bits. All states are replicated per nCPU.

1.3.2**Coprocessor 0 (COP0) Registers**

The integrated system control coprocessor 0 (COP0) in the MIPS64 architecture is responsible for the virtual memory subsystem, the exception control system, and the diagnostics capability of the processor. On-chip system control registers are associated with COP0. COP0 registers and functions are accessed through the Coprocessor 0 instructions. These provide the path through which the virtual memory system's page mapping is examined and modified, exceptions are handled, and operating modes are controlled. Additional registers in this group are used to implement a real-time cycle counting facility, to aid in cache diagnostic testing, and to assist in data error detection.

1.3.3**Coprocessor 2 (COP2) Registers**

The coprocessor 2 (COP2) registers provide access to the Fast Messaging Network, an important element in the throughput-optimized XLS processor architecture. The COP2 registers and associated Coprocessor 2 instructions are also used to manage message-passing within the processor.

1.3.4**Integer Unit**

The integer unit includes 32 general-purpose 64-bit registers, a load/store architecture with single-cycle ALU operations (add, sub, logical, shift), and an autonomous multiply/divide unit. The autonomous multiply/divide unit optimizes multiply and multiply-accumulate operations, maximizing throughput while still using an area-efficient implementation. Additional resources include the HI/LO result registers for the two-operand integer multiply/divide operations, and the program counter. It includes two implementation-specific instructions that are useful in the embedded market; integer multiply accumulate (eliminating the need for a separate DSP engine) and a 3-operand integer multiply.

1.3.5**MIPS64 Instruction Set Enhancements**

NetLogic has enhanced the XLS CPU with a number of extensions to the MIPS64 instruction set to improve system and application-level performance. [Table 1-2](#) lists the new instructions.

Table 1-2. Instruction Set Additions

Instruction	Description	Usage
SWAP	Atomic Swap	A fast and efficient mechanism for implementing software locks.
LDADD	Atomic Load and Add	Used for updating shared counters or implementing advanced software-locking mechanisms. This instruction can offer large benefits in many applications that rely heavily on maintaining statistics counters.
DADDWC	Double Add With Carry	This instruction is very useful for TCP checksum generation and can double the performance of checksum generation loops.
MSGLD	Message Load	Allows the processor to pop a message received over the Fast Messaging Network (FMN). ^a
MSGSEND	Message Send	Allows the processor to send a message on the Fast Messaging Network. ^a
MSGWAIT	Message Wait	Allows the processor to wait for a message to be received over the Fast Messaging Network. ^a

a. See [Chapter 12, "Fast Messaging Network"](#).

1.3.6 Extended MIPS64 Interrupt Architecture

To enable fast interrupt handling, the system must be capable of quickly identifying and prioritizing pending interrupts. To this end, the XLS processor has extended the existing MIPS64R1 architecture by allowing up to 64 possible identifiable interrupt sources. Each device on the XLS processor can be assigned, via the Programmable Interrupt Controller (PIC), a unique interrupt ID ranging in values from 0 to 63. The processor implements a 64-bit extended interrupt request register (EIRR) to be used for interrupt delivery. Each bit in the EIRR represents one of the possible interrupt ID's. Based upon the bit location in the EIRR software can determine the source of the interrupt as well as its priority (assuming software assigned interrupt IDs in a way which reflects priority).

1.3.7 Message Passing Extensions

Message passing is a common software mechanism widely used for efficient inter-processor communication in multiprocessor systems. However, conventional architectures rely on using shared memory data structures and interrupts, to implement message passing which results in increased system bandwidth requirements, long latency software locks, and poor software response time, resulting in lower scalable performance.

The XLS processor implements a unique message passing facility using the coprocessor 2 space in the MIPS64 architecture. Processors, as well as devices, can form and pass messages across a dedicated Fast Messaging Network (FMN), facilitating efficient low-latency communication. For a more in-depth discussion of the message network, see [Chapter 12, "Fast Messaging Network"](#).

1.4 XLS Multi-Threading

Each XLS processor core contains a single issue pipeline that is 4-way multi-threaded. Multi-threading allows each processor core to actively maximize pipeline performance capitalizing not only on instruction-level parallelism (ILP) but also thread-level parallelism (TLP). The threads on each core have unique contexts with a fully replicated register set, interrupt and exception handlers. Any operating system or application running on a hardware thread sees a unique CPU that is logically identical to a physical CPU. In the context of the XLS, the hardware thread is also called an nCPU™ NetLogic virtual CPU for this reason.

Multi-threading allows the processor pipeline to be filled efficiently whenever another nCPU encounters a stall event. A pipeline stall can occur for many reasons, including:

- L1/L2 cache misses
- branch prediction failures
- TLB misses and refills
- Memory accesses

During such events, the thread scheduler automatically puts the nCPU into sleep and schedules the next available nCPU based on the scheduling configuration. The context switch is immediate without cycle loss and the sleeping nCPU becomes eligible to rerun when the dependency is satisfied.

The thread scheduler on each core is configurable using core registers. In general, coarse-grained scheduling allows an nCPU to execute continuously as long as there is no stall event. When a stall occurs, the nCPU is immediately switched out in favor of the next ready-to-run nCPU based on the scheduling principle. Fine-grained scheduling allows specific cycle counts to be allocated on a per-nCPU basis. The XLS core provides the following flexible scheduling schemes:

- Round robin
- Fixed cycle

- Priority thread

Round-robin is a fine-grained scheduling model where each nCPU is provided a cycle as long as it is ready to run. If an nCPU is not ready to run, the next nCPU in the circle queue is scheduled. The fixed-cycle scheduling provides an 8-bit counter per nCPU where a block of cycles is allocated to each nCPU. Priority thread scheduling allows a guaranteed minimum of cycles to be allowed for a given nCPU before which an nCPU cannot be rescheduled. Depending on the target application being run on a given core, a specific scheduling policy may be selected to optimize for maximize performance. (See [Appendix A, “Multi-Threading Advantages”](#))

1.4.1

Scheduling Flexibility of the XLS Processor

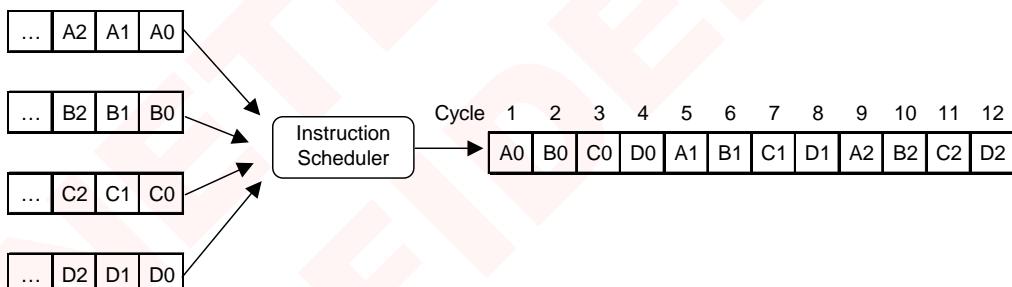
The threaded architecture provides the flexibility to schedule nCPU use to be tailored to the specific processing requirements of a given application. For example, more network-centric environments benefit from a balanced round-robin use of threads while a more compute-intensive application may require the use of all available cycles for a single nCPU. The XLS processor provides three highly flexible options for thread scheduling to maximize resource utilization: Round Robin, Fixed cycle and Priority Thread.

1.4.1.1

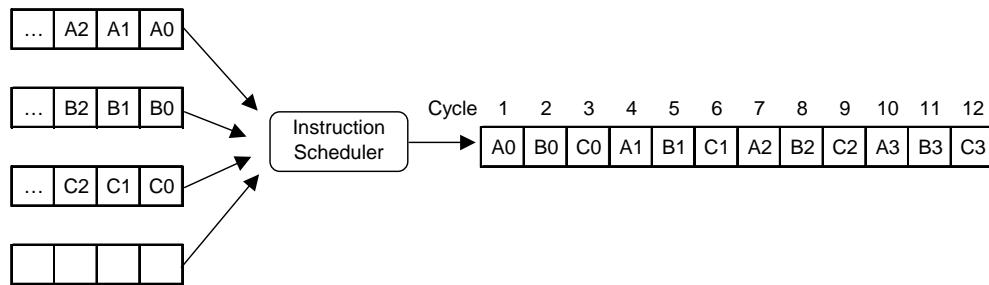
Round Robin (*Fine-Grained*) Scheduling

Round Robin is the default scheduling mode. Threads are scheduled on a cycle-by-cycle basis using a round robin algorithm. All threads are given equal access as long as an instruction is available to be issued (see [Figure 1-8](#)). This is also known as “fine-grained thread scheduling.”

Figure 1-8. Round Robin (*Fine-Grained*) Scheduling



If an nCPU is stalled and cannot be issued, the round robin logic skips over it, freeing up cycles for the remaining nCPUs (see [Figure 1-9](#)).

Figure 1-9. Round Robin (Fine-Grained) Scheduling with One Stalled Thread

This scheduling scheme also uses a Linear Feedback Shift Register (LFSR) to randomize the nCPU picked for scheduling every 4 cycles. The randomization ensures a process is not blocked when nCPUs are competing for resources.

1.4.1.2 Fixed Cycle (Coarse-Grained) Scheduling

Fixed cycle scheduling allows blocks of cycles to be allocated to each nCPU using 8-bit scheduling counters. As the threads are scheduled in blocks of cycles instead of single cycles, this is also known as “coarse-grained thread scheduling.”

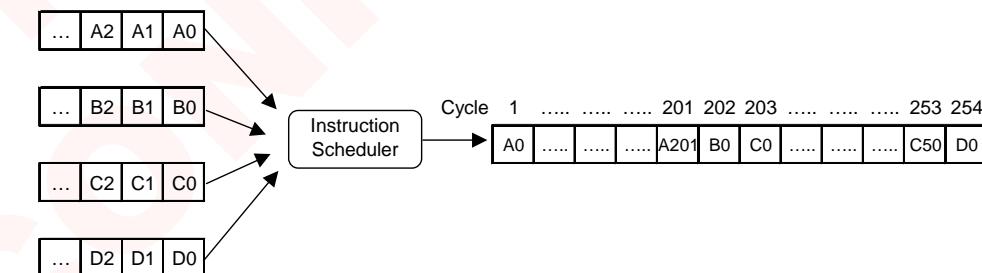
Consider an example in which the counters are programmed as follows:

Thread	D	C	B	A
Scheduling Counters	0	50	0	200
Cycles Scheduled To Thread	1	51	1	201

Each nCPU will be scheduled for the indicated number of cycles, unless it is sleeping. As shown in [Figure 1-10](#), nCPU A will get scheduled for 201 consecutive cycles unless sleeping. nCPU B will get scheduled for 1 cycle unless sleeping, and so on.

Figure 1-10. Fixed Cycle Scheduling

Fixed Cycle



Fixed cycle scheduling essentially utilizes the same balanced logic as in round robin scheduling, except that the round robin pointer is held at each nCPU for the block of cycles programmed in the scheduling counters. When the counter reaches zero, the pointer moves to the next nCPU. If an nCPU is sleeping while its block of time is active, the other nCPUs get scheduled using the round robin logic.

The counter values for Fixed Cycle scheduling logic are limited to the following configurations:

Thread	D	C	B	A
Scheduling Counters Configuration 1	0	M	0	N
Scheduling Counters Configuration 2	M	0	N	
Scheduling Counters Configuration 3	X	X	X	X

where: M and N can be different numbers

X can be any combination of 0 or N, with all N's the same

1.4.1.3 Priority Thread Scheduling

Priority thread scheduling uses scheduling counters to specify the minimum number of cycles before an nCPU/thread can be rescheduled. For example, based on the counters programmed as shown in Table 1-3:

Thread A would not be scheduled more than every 3rd cycle

Thread B would not be scheduled more frequently than every 4th cycle

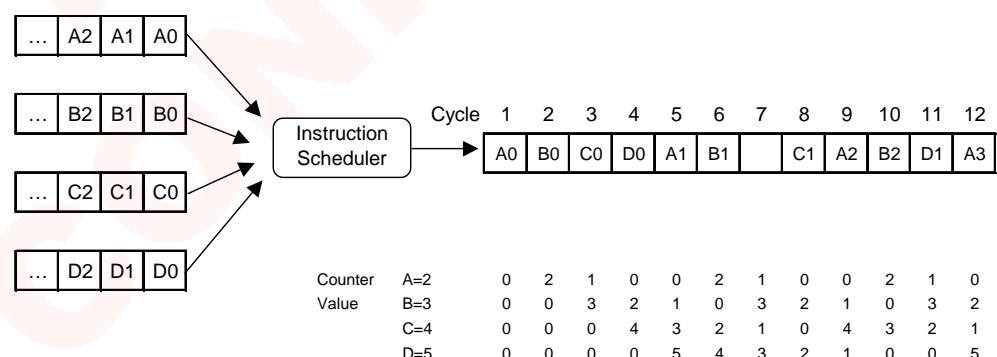
Thread C would not be scheduled more frequently than every 5th cycle

Thread D would not be scheduled more frequently than every 6th cycle

Table 1-3. Priority Thread Scheduling Example 1

Thread	D	C	B	A
Scheduling Counters	5	4	3	2

Figure 1-11. Priority Thread Scheduling



The same round robin logic is used for the instruction scheduling with the following restriction: While its counter is decrementing, that particular nCPU is not eligible to be scheduled. When the counter reaches zero, the nCPU is put back into the pool ready to be scheduled. This scheduling mode is a throttling or policing mechanism. It's important to note that even if all

other nCPUs are asleep, an nCPU will not be scheduled until its scheduling counter reaches zero, thus resulting in unused or wasted cycles in the pipeline. Zero (0) is a legal value for a counter, which implies no wait before an nCPU can be scheduled again. With the counters set to all zeroes (default), this scheme operates exactly the same as Round Robin scheduling. For example, if the counters are programmed as shown in [Table 1-4](#), nCPU/thread D will be scheduled no more frequently than 1 of every 10 cycles. The remaining nCPUs will be scheduled based on round robin scheduling.

Table 1-4. Priority Thread Scheduling Example 2

Thread	D	C	B	A
Scheduling Counters	10	0	0	0

1.5

Processor Memory Subsystem

The memory subsystem forms one of the most critical components of any highly integrated system on a chip. A poorly designed memory subsystem will lead to an unbalanced design resulting in memory and I/O bottlenecks, chokepoints throughout the system, and increasingly inconsistent performance as system load increases.

Advanced processors require a high performance memory subsystem in order to achieve scalable performance. In addition, a processor must be able to maintain a large number of outstanding transactions, reorder memory references, implement a non-blocking load/store architecture, have a scalable high bandwidth system and memory interconnect, and provide a flexible software model to enable applications to attain their potential performance levels. Without these capabilities, applications will exhibit poor performance scalability with increasing compute resources.

To ensure the best overall performance and throughput, the XLS processor implements a well-balanced memory subsystem suitable for both existing and future applications. Key attention has been paid to the use and implementation of write-back data caches and multi-channel DRAM interfaces to optimize bandwidth. The write-back data caches conserve critical memory interconnect bandwidth that would otherwise be wasted on unnecessary store traffic with a write-through scheme. In the XLS, the multi-channel DRAM interfaces improve overall bandwidth by providing more banks than a single channel approach.

Key features of the XLS processor memory subsystem include:

- 256 KB to 1 MB Level-2 cache used for caching processor and I/O traffic
- Dedicated Level-1 instruction cache per processor core
- Dedicated write-back Level-1 data cache per processor core
- Up to 4 outstanding cache line requests
- Up to 32 outstanding load requests
- Programmable memory ordering model-enabling software to configure the processor to support a range of ordering models, from strongly ordered to weakly ordered
- Memory Distributed Interconnect which scales with increasing clock frequency and compute resources
- Fully coherent memory subsystem, including both processor and I/O traffic
- Scalable Distributed Interconnect
- Configurable up to four channel DRAM interface supporting DDR2
- DRAM interfaces configurable as up to two 72-bit channel pairs or four 36-bit channels
 - XLS208 and XLS204 configurable as one 36-bit channel or one 72-bit channel
 - XLS108 and XLS104 configurable as one 36-bit channel
- 64-GB max addressable DRAM (16 GB per 36-bit channel, or 32 GB per 72-bit channel)

1.5.1 Level-1 Caches

Each XLS core implements both a 32-KB Level-1 Instruction cache and 32-KB Level-1 Data cache. The key attributes of the cache memories are noted in [Table 1-5](#).

Table 1-5. Primary Cache Parameters

Attribute	Instruction Cache	Data Cache
Size (Kbytes)	32	32
Associativity	8-way	8-way
Line size	32 bytes	32 bytes
Line Locking	Yes	Yes
Write policy	N/A	Write-back
Coherency	Software managed	Hardware managed – MOSI
Addressing scheme	Physical index/Physical Tag	Physical index/Physical Tag
Replacement policy	Pseudo-LRU	Pseudo-LRU
Error management on tags	Parity bit per tag	Parity bit per tag
Error management on data	Parity (2 bits per double word)	ECC – SECDED: Single Error Correction, Double Error Detection (8 bits per double word)
DedicatedPlus Cache	Partitions cache per nCPU (thread)	Partitions cache per nCPU (thread)

1.5.1.1 Level-1 Instruction Cache

Each XLS CPU contains a 32-KB, Level-1 8-way set-associative instruction cache with a 32-byte line size. The high degree of set associativity ensures the most efficient use of the processor threads/nCPUs by reducing cache-conflict misses. Both the cache instruction and cache tag arrays are protected by parity. When a parity error is detected, the instruction is discarded, the cache line is invalidated, and a new instruction fetch is executed. This process is accomplished without software intervention.

1.5.1.2 Level-1 Data Cache

Each XLS CPU contains a 32-KB, Level-1, 8-way set-associative data cache with a 32-byte line size. As with the instruction cache, the high degree of set associativity reduces cache-conflict misses.

1.5.1.3 Data Cache ECC Protection

The data cache array is protected by ECC (SECDED – Single Error Correction and Double Error Detection) for single-bit error correction without software intervention. All multiple-bit data errors are detected and then passed to the software via a cache error exception. The cache tag arrays are protected by parity.

1.5.1.4 DedicatedPlus™ Cache

The XLS supports a DedicatedPlus Cache mode for L1 Data and Instruction Caches. Using this mode, the software may partition the 32-KB I & D caches such that every nCPU (hardware thread) is guaranteed a minimum size. Cache coherency is maintained across the partitions created by software. All nCPUs are still able to read the code/data allocated by other nCPUs and hence benefit from the pre-fetching performed by the other nCPUs.

1.5.1.5**Unaligned Load/Stores**

The XLS processor implements a mode to allow the CPU to issue and complete unaligned load/store operations without generating an exception. This mode allows the CPU to issue and complete unaligned load/store operations without generating an exception. Completion of unaligned load/store operations will take longer compared to aligned operations as each unaligned reference will require hardware to complete two load/store operations. This is still faster than taking an exception and executing exception handler code for completion of the operation.

1.5.2**Level-2 Unified Cache**

The XLS processor contains up to 1 MB of Level-2 cache with 8-way set-associativity and a 32-byte line size. It uses a pseudo LRU algorithm for way selection and supports locking on a per line basis. The L2 multi-banked design allows for up to eight parallel accesses for any given cycle.

The data and tag arrays of the L2 cache are protected using ECC (SECDED). All single-bit errors are corrected and all double-bit errors are detected without software intervention. When uncorrectable errors are detected, they are passed to the software as a cache error exception.

The L2 cache is designed to be non-inclusive of the L1 caches allowing more data to be cached than would ordinarily be in an inclusive design.

1.5.3**Virtual MIPS Mode (Virtualization)**

The XLS facilitates virtualization using virtual MIPS mode, that provides a method of allowing multiple operating systems such as Linux, VxWorks and NetLogicOS to reside cooperatively within a single XLS memory system.

This unique operational mode enables a straightforward means of isolating multiple virtual CPUs and preventing memory contention and collisions in heterogeneous operating system environments. Virtual MIPS Mode increases the flexibility of the MIPS architecture by allowing virtualization of the unmapped regions. See [Chapter 3, “Virtual MIPS Mode”](#).

Virtualization or para-virtualization of the XLS can be achieved by implementing a Hypervisor using the virtual MIPS mode, which also allows multiple copies of the same operating system (e.g., Linux) to run simultaneously.

1.5.4**DRAM Interface**

The XLS processor memory controllers implement a number of advanced features designed to enable performance, reliability, and flexibility.

- Transaction Reordering support for higher performance and utilization
- Programmable closed page, open page and hybrid policies
- Hybrid page policy (closed page with open hints based on pending transactions)
- Nibble correcting advanced ECC code
- Supports Active Power Down and Self Refresh Power Down

The XLS processor provides several memory design options to the system designer to optimize for cost and performance based on the needs of the design. There are up to four independent memory controllers and up to four DRAM memory channels, which can be configured as 36-bits per channel.

The memory controller supports DDR2 DRAM up to an 800-MHz data rate (with registered DIMMs).

The DRAM memory controllers support several configurations for organizing the address space. The simplest approach is to allocate portions of the memory space to each channel. Another option is to interleave the data across the two or four memory channels. For example, under user control, data can be striped across all channels using a subset of the address bits.

1.6 XLS Processor Network Accelerators

The XLS Processor's architecture is designed to deliver high performance using an extended set of resources, which can share processing work. For conventional designs, as more and more processing elements are added, it becomes increasingly difficult to ensure efficient resource sharing. The result is ineffective workload distribution and throughput. Many of today's multiprocessors utilize a standard DMA ring in which all CPU resources will attempt to access a single data structure. This is extremely inefficient since it frequently requires the use of data locks, thus diverting an unusually large percentage of CPU time to simply access the data without completing any meaningful work. It's also not uncommon for a subset of processor resources to be dedicated simply to managing the workload distribution burden rather than be used for more productive output. In either case, throughput potential is wasted.

As a next-generation, multi-threaded multiprocessor with up to 16 fine grain threads operating in parallel, the XLS processor is best supported by an efficient means of packet management and distribution. NetLogic addressed this requirement by equipping the XLS processor with a highly optimized Networking Accelerator subsystem supporting higher-layer programmable parsing, packet direction management, checksum verification and load balanced packet distribution across the multiple threads. This hardware ensures the XLS processor operates at peak levels of performance and workload efficiency. See [Chapter 13, "Networking Accelerators"](#).

1.6.1 Networking Accelerator Architecture

Each XLS has from one to two Network Accelerators, as shown in the main architectural diagram [Figure 1-1](#), and [Figure 1-12](#) on the next page. Each Network Accelerator services a "quad" of up to four SGMII interfaces.

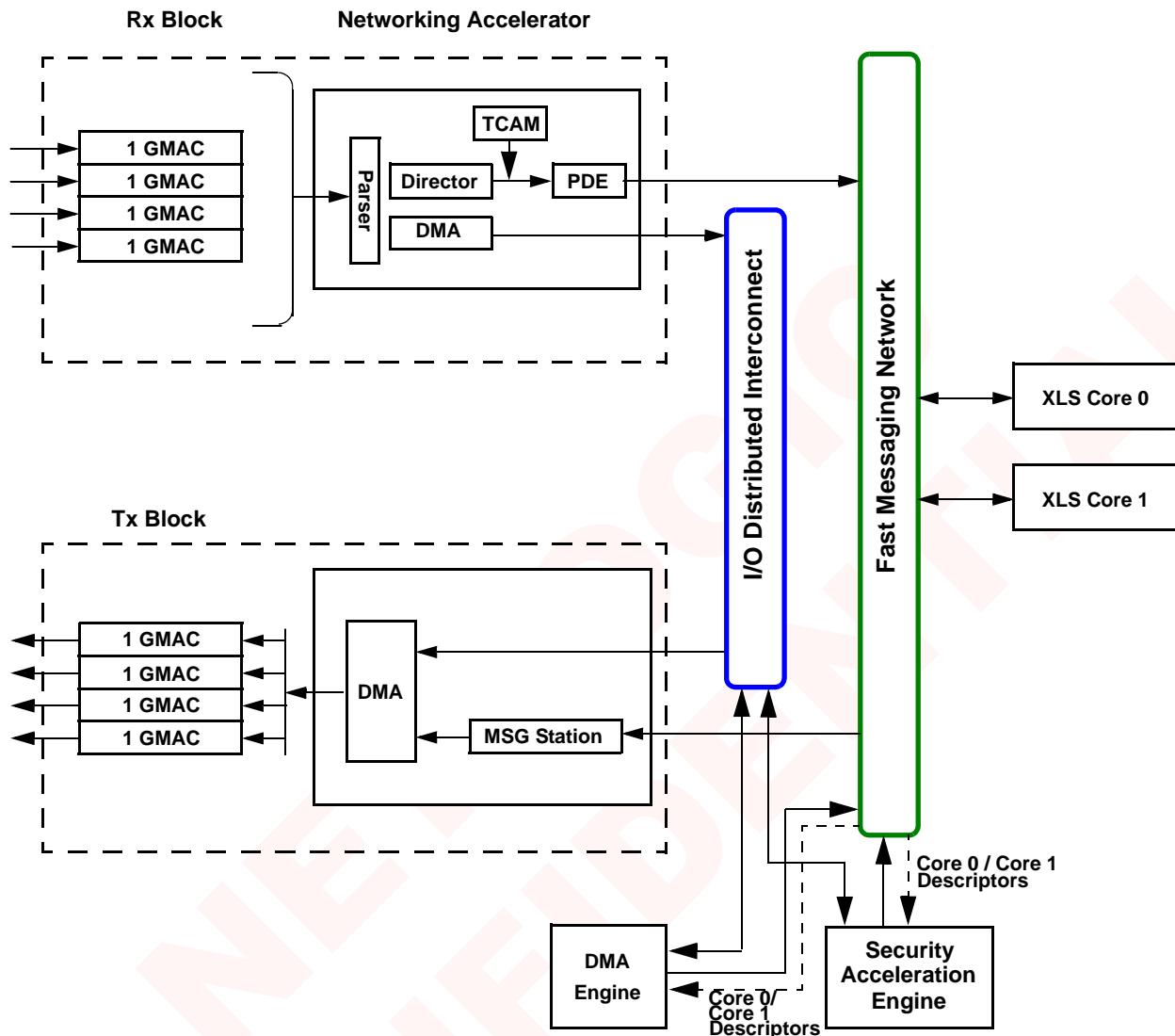
Overall, the accelerators support:

- Up to eight 1 Gbps MACs with CRC, filtering on incoming packets and pass-through optimizations

The key elements and features of each networking accelerator include:

- Packet Distribution Engine for software-controlled load-balanced distribution of incoming packets directly to all threads
- Programmable Packet Parser, capable of extracting up to a 128-bit key consisting of up to six fields
- Packet Director provides flexible packet tagging, queuing and distribution management
- IPv4 header, TCP, UDP checksum verification on incoming packets
- Automatic memory management — allocate or de-allocate buffers during packet-flow
- Per-port programmable 0-7 byte shift for storing incoming packets
- Transmit packets may start from any arbitrary offset within a descriptor.
- Accelerators' DMA controllers require no software intervention

[Figure 1-12](#) is a functional block diagram showing the Networking Accelerator with its key acceleration elements, the Parser, the Packet Director and Packet Distribution Engine (PDE).

Figure 1-12. Networking Accelerator Packet Flow (XLS6xx Example)

In the above diagram, the IO DI and FMN are transport rings. DMA carrying packet data connects to the I/O DI, and packet management information goes to and from the FMN.

For more information, please refer to [Chapter 13, “Networking Accelerators”](#).

1.7 Fast Messaging Network

Another performance-related feature of NetLogic's XLS architecture is its Fast Messaging Network (FMN). The FMN supports the thread-based architecture of the XLS processor, which is designed and tuned to deliver extremely efficient parallel processing performance. To ensure maximum utilization of its throughput-optimized capabilities, communications among key processing agents must be swift and lightweight. With the XLS processor handling up to 16 threads, up to eight 1-Gbps network interfaces, a compression/decompression engine (XLS6xx and XLS4xx-Lite/XLS4xx), and a security acceleration engine supporting up to 2.5-Gbps of bulk encryption, it is critical to have an intelligent and scalable interconnect designed to eliminate communications bottlenecks.

With this kind of processing power, typical shared bus architectures will get bogged down with excessive memory traffic and arbitration overhead. The XLS processor provides an innovative solution for optimized data communications among all key processing agents by implementing a unique high-performance, low-latency Fast Messaging Network between the processor cores, accelerated networking interfaces and Security Acceleration Engine.

The FMN provides extremely efficient management of packet data between functional blocks of the XLS processor. The FMN passes packet information without consuming interconnect bandwidth, and without using inter-processor communication spin-locks or semaphores. Once configured, the FMN automatically manages message transmission and administers transmission bandwidth with no software overhead.

The FMN is optimized to transmit messages describing packets or control messages. These messages can contain descriptive information about packets and pointers to packet data. The Distributed Interconnect (DI) carries packet data referenced by the messages. CPUs and I/O devices communicate about packets by transmitting messages to each other on the FMN. This optimizes packet throughput and eliminates bandwidth bottlenecks.

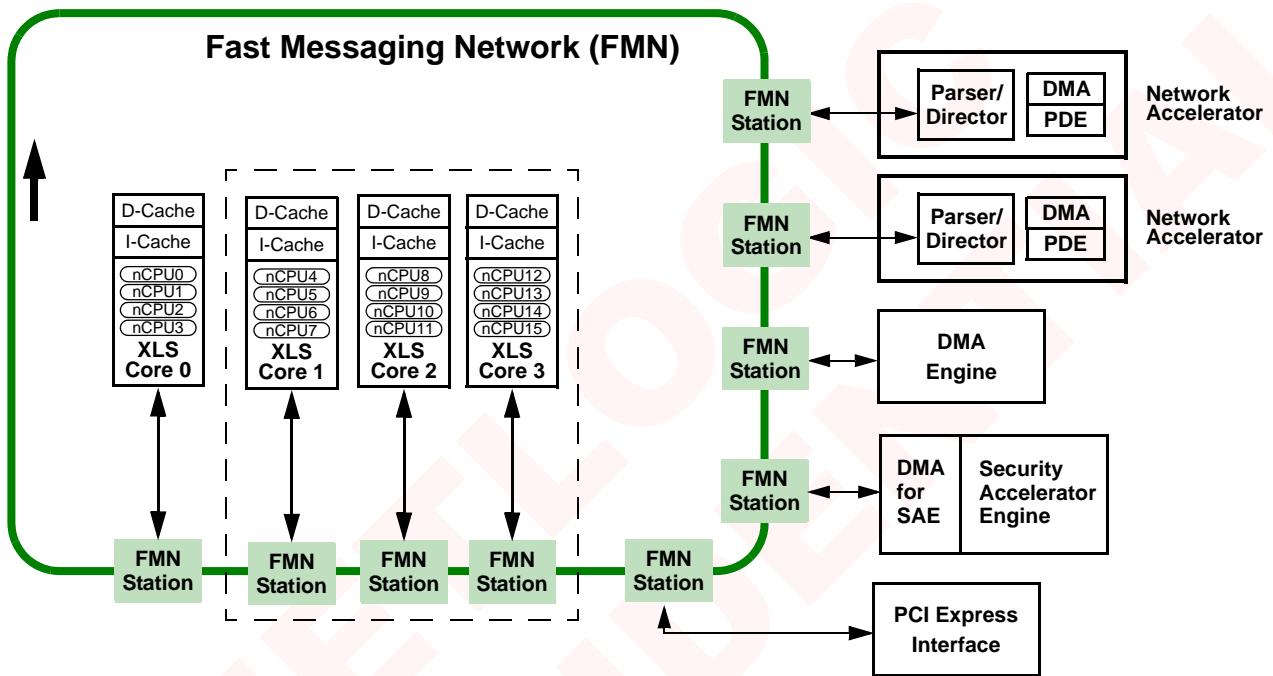
The FMN ensures fairness among its stations by using a credit-based, round-robin scheme for placing messages onto the FMN. This scheme guarantees that in-transit messages arrive quickly at their destination and ensures that stations are not starved from access to the FMN.

While the FMN is designed primarily for fast packet movement, it can be used for any other purpose by defining the syntax and semantics of the message container. The FMN is designed for point-to-point communications.

1.7.1 Fast Messaging Network Architecture

The FMN is a 32-bit-wide low-latency, unidirectional, ring-based message network running at half the core clock frequency. As shown in [Figure 1-13](#), the key functional blocks of the XLS processor are connected to the Fast Messaging Network. The connection points between the FMN and these functional blocks are called *stations*. The XLS cores, networking interfaces, Security Acceleration Engine (SAE), DMA Engine, PCI Express interface, and Compression/Decompression Engine (CDE) all have stations on the FMN. See [Figure 1-13](#).

Figure 1-13. Fast Messaging Network Stations Example



The 2nd core available in XLS6xx, XLS416, XLS408-Lite, XLS408, XLS208 and XLS108

The 3rd & 4th cores available in XLS616, XLS416 only.

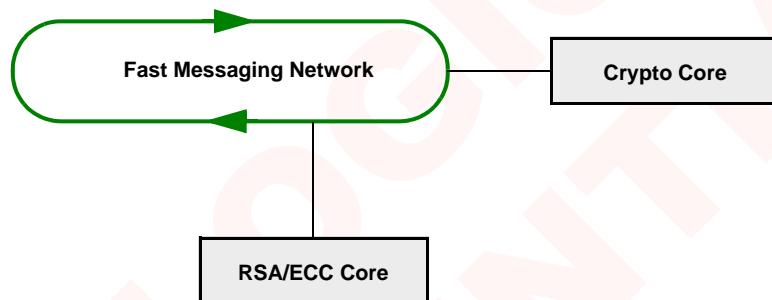
For more information, please see [Chapter 12, “Fast Messaging Network”](#).

1.8 Security Engine

The XLS Processor integrates a high performance Security Acceleration Engine (SAE) that enables up to 2.5Gbps of bulk cryptographic processing supporting industry standard security protocols such as IPsec and SSL. The SAE was designed to provide encryption/decryption and authentication-related hashing acceleration support for industry standard security algorithms. When combined with the protocol processing capabilities of the processor cores, the XLS Processor is capable of providing high performance flow through security processing.

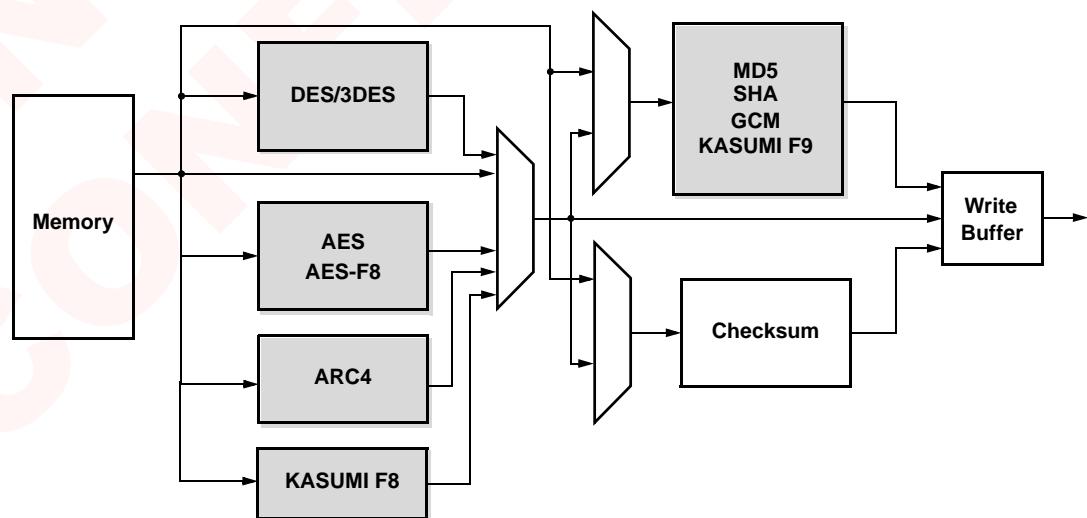
To provide greater flexibility for security implementations, the SAE is designed with a crypto cores and an RSA/ECC core, each of which operates independently, as shown conceptually in [Figure 1-14](#). The SAE supports DES, 3DES, AES, ARC4, and Kasumi F8 cryptography, SHA and MD5 authentication/hashing and CRC checksum and RSA exponential key generation.

Figure 1-14. 2.5-Gigabit Security Acceleration Engine (Conceptual Diagram)



[Figure 1-15](#) shows a block diagram of the crypto core, showing the cryptographic functions, hashing and CRC blocks.

Figure 1-15. Single 2.5 Gigabit Crypto Core



For more information, please see the [Security Acceleration Engine](#) chapter.

1.9

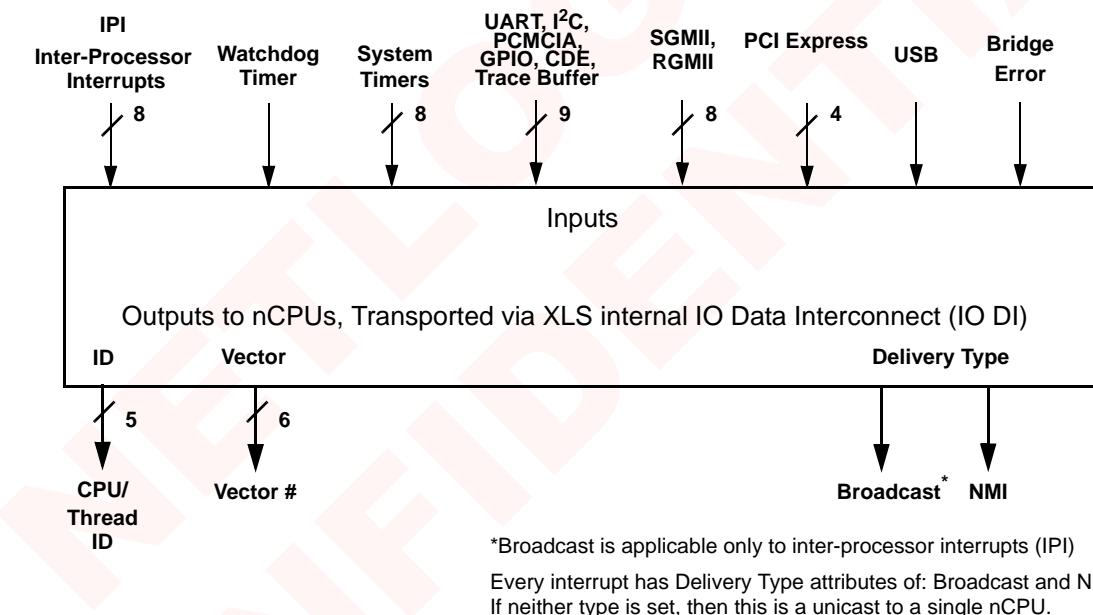
Programmable Interrupt Controller

The Programmable Interrupt Controller (PIC) receives all interrupt requests from internal sources such as timers, CDE, on-chip peripherals (Network Accelerators, GMAC, PCIe, USB, PCMCIA, etc.), as well as external sources. The PIC synchronizes, validates, prioritizes, schedules and delivers the interrupts to any one of the available nCPUs. See [Figure](#).

An Interrupt Redirection Table (IRT) determines the actions taken by the PIC for all interrupt requests. It specifies how the PIC will direct interrupt requests. The IRT provides a great deal of flexibility and efficiency for interrupt handling. For example, software may use it to direct all external interrupt requests to a particular nCPU or nCPUs; or, certain classes of interrupt request may be distributed among a pool of nCPUs using round-robin scheduling to achieve load-balanced interrupt servicing.

Threads/nCPUs can also signal other nCPUs synchronously by sending Inter-Processor Interrupts (IPI) via the PIC. For more information, see the [Programmable Interrupt Controller](#) chapter.

Figure 1-16. Programmable Interrupt Controller Block Diagram



1.10

Compression/Decompression Engine

XLS Processors (excluding XLS2xx devices) integrate a high performance Compression/Decompression Engine (CDE) providing autonomous operation and relieving the CPUs from the processing required for industry standard compression algorithms. Performance of up to 2.5 Gbps is achievable.

Capabilities include:

- ZLIB/Deflate/GZIP (per RFC1950/1/2)
- IPCOMP
- 32KB dictionary size
- Fixed or Dynamic Huffman code

1.11 Data Transport

As shown in [Figure 1-12](#), the XLS provides a high-speed internal data transport system, the Distributed Interconnect. This system has two rings: the **MDI** or Memory Distributed Interconnect, and the **I/ODI**, or I/O Distributed Interconnect. Each major XLS internal subsystem requiring data transport connects a ring via a DI station. See also [Figure 2-1](#).

The Memory and I/O Distributed Interconnect Hub connects the two rings. The System Bridge Controller within the Hub maintains transactions with the individual I/O devices on the I/O Distributed Interconnect Ring. Allocation of these transactions is made by a credit-based mechanism. See [Chapter 8, “Memory and I/O Access”](#) for details.

1.12 Peripherals

XLS family devices offer extensive networking interfaces:

- Up to eight 10/100/1000 SGMII ports
- One (optional) RGMII Ethernet port
 - FIFO mode operation on RGMII Ethernet MAC port
- Up to two optional 4x1 or 1x4 XAUI Interfaces
- Each interface is integrated with advanced Networking Acceleration
- PIC services all networking interfaces

Industry-standard high performance interfaces:

- PCIe 4x1, 1x4, 2x1
- SRIO optional 4x1 or 1x4

XLS processors also include integrated system Interfaces:

- Up to two USB 2.0 interfaces
- PCMCIA interface
- Flash memory interface
- Two I²C bus interfaces
- Two 16550 UART interfaces
- 32-bit GPIO interface
- IEEE 1149.1 EJTAG and Memory BIST functionality

1.13 Development Support

1.13.1 Debug Capabilities

1.13.1.1 EJTAG

The XLS processor family supports an enhanced industry-standard IEEE-1149.1 (EJTAG) boundary-scan test port to enable a full-featured development and debug environment. This interface is compliant with the MIPS EJTAG 2.6 specification and supports a variety of JTAG (EJTAG) in-circuit emulators. The key EJTAG capabilities include:

- Download / upload to EJTAG memory, Single step execution and breakpoints
- Master and slave modes
- JTAG probe can initiate transactions as a master
- Responds to CPU accesses to the EJTAG memory as a slave
- Probe and CPU access are controlled using Address, Data and Control Registers
- Debug Interrupts
- Supports the following instructions through the JTAG probe

```
IDCODE    ALL
IMPCODE  EJTAGBOOT
ADDRESS   NORMALBOOT
DATA      FASTDATA
CONTROL   BYPASS
```

1.13.1.2 Trace Buffer

The XLS processor family includes a special buffer to store address traces from the system Distributed Interconnect, which can store up to 256 address traces. The XLS processor provides a programmable trigger mechanism to start and stop the trace based on a filter. The filter is set of AND'ed match criteria, which includes the address, the transaction type, snoop result, interrupt raised, etc.

The trace buffer can operate in two modes: [1] Start trace on filter match and continue collecting all subsequent transactions until a specified number of transactions is reached, or [2] Collect only transactions that match the filter until a specified number of transactions is reached. Both the CPU and JTAG can configure the trace buffer and read the captured information. The trace buffer can be configured to generate an interrupt when the trace is complete. The trace buffer can also be used as a scratch buffer in a FIFO manner.



Chapter 2 XLS MIPS64 CPU Core

2.1

Introduction

The XLS processors each integrate up to four identical multi-threaded MIPS64 processor cores. The cores are designed to take advantage of the inherent packet-level parallelism exhibited by networking applications, while delivering competitive performance for general-purpose applications. The key attributes of each core include:

- MIPS64 ISA compliance
- 4-Way Multi-threaded
- Single In-order Instruction Issue
- Branch Prediction
- Return Stack Buffer
- 32 KB Level-1 Instruction Cache with parity
- 32 KB Level-1 Data Cache with ECC

The XLS processor core adds several features to the MIPS64 ISA and register structure. Extensions include coprocessor registers, memory reference system, and interrupt handling.

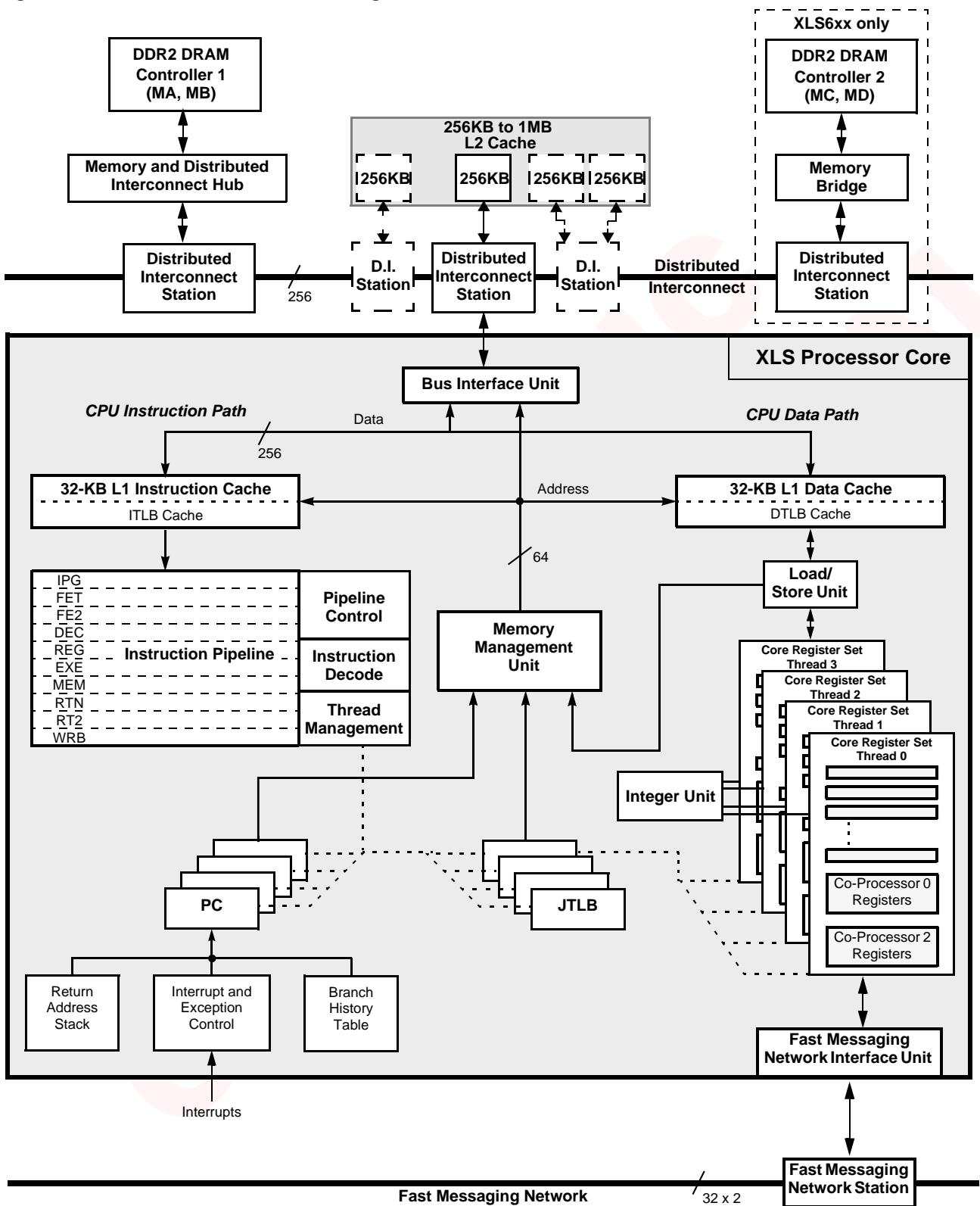
[Figure 2-1](#) shows a core's major function blocks and interconnection paths. The core is connected to other system elements through the Distributed Interconnect's two rings (for connection to non-core memories), and the Fast Messaging Network (for connection to Network Accelerators, etc.). For details of these interconnects, see [Chapter 8, "Memory and I/O Access"](#) and [Chapter 12, "Fast Messaging Network"](#).

2.1.1

Conformance

The processor cores in the XLS are fully conformant to the MIPS32 and MIPS64 specifications. Each processor core contains a single in-order issue pipeline and thus supports R1 of both ISAs. For specific details on architecture, instruction set, and privileged resource architecture, please consult the following specifications from MIPS Technologies, Inc.

1. MIPS32(R) Architecture for Programmers Volume I: Introduction to the MIPS32(R) architecture, version 2.50
2. MIPS32(R) Architecture for Programmers Volume II: The MIPS32(R) Instruction Set, version 2.50
3. MIPS32(R) Architecture for Programmers Volume III: The MIPS32(R) Privileged Resource Architecture, version 2.50
4. MIPS64(R) Architecture for Programmers Volume I: Introduction to the MIPS64(R) architecture, version 2.50
5. MIPS64(R) Architecture for Programmers Volume II: The MIPS64(R) Instruction Set, version 2.50
6. MIPS64(R) Architecture for Programmers Volume III: The MIPS64(R) Privileged Resource Architecture, version 2.50

Figure 2-1. Core Functional Block Diagram

2.2 CPU Data Path

The CPU data path consists of the Integer Unit with its two co-processors, associated general purpose register set, and the Load/Store Unit which connects the register set with the level-1 data cache. The level-1 data cache is the entry point for the data/instruction memory subsystem.

2.2.1 Integer Unit

The Integer Unit is fully compatible with all MIPS ISA in current use. It uses 32 general purpose 64-bit registers, a load/store architecture with single-cycle ALU operations (add, subtract, logical, and shift operations), and an autonomous multiply/divide unit. The multiply/divide unit optimizes multiply and multiply-accumulate operations, maximizing throughput while still achieving an area-efficient implementation. Additional resources include the HI/LO result registers for two-operand integer multiply and divide operations, and the program counter.

2.2.2 CPU Registers

The XLS processor core uses the MIPS64 ISA which has a user-visible state consisting of 32 general purpose registers, two special purpose registers for integer multiplication and division, scratch registers, a program counter, and no condition code bits.

2.2.3 Load/Store Unit

The Load/Store Unit creates virtual addresses for transfers between the register set and the memory subsystem, and effects data transfers.

2.2.3.1 Unaligned Load/Store Operation Core Support

The MIPS ISA requires that any load/store operation issued by the processor which is not naturally aligned must result in an address error exception. Following are the requirements for a load/store operation to be considered naturally aligned by the processor.

- 8-byte load/store: Address bits [2:0] == 000
- 4-byte load/store: Address bits [1:0] == 00
- 2-byte load/store: Address bit [0] == 0

All 1-byte load/store accesses are considered naturally aligned.

Any load/store operation which does not meet this criteria is considered unaligned and will cause the processor to raise the Address Error Exception. The exception handler software will either complete the original operation or result in aborting the program. Most modern compilers produce naturally aligned references with potential exception when dealing with packed structures.

Typically, the restriction pertaining to aligned references results in a faster and simpler hardware implementation for load/store operations.

The XLS processor implements a mode to allow the CPU to issue and complete unaligned load/store operations without generating an exception. Completion of unaligned load/store operations will take longer compared to aligned operations, as each unaligned reference will require hardware to complete two load/store operations, requiring up to 6 clocks (depending on the location of the data). This is still faster than taking an exception and executing exception-handler code for completion of the operation.

This mode is turned on by setting bit 14 in the L1D_CONFIG0 register as described in the following chapter.

Note that one bit is used for every set of 4 virtual CPUs. For example, setting bit 14 of L1D_CONFIG0 register in CPU0 will turn on the unaligned access support mode for nCPU0,

nCPU1, nCPU2, and nCPU3. In this example, nCPU4, nCPU5,...nCPU7 will continue to generate address error exception whenever an unaligned load/store operation is encountered.

2.2.3.2 Verifying Store Operation Completion

Regardless whether a store operation ultimately becomes a memory transfer, a PCI or PCI-X transfer, a GPIO transfer, or any other transfer, verifying completion is best accomplished by reading the same address back to verify contents.

2.2.4 Level-1 Instruction Cache

Each XLS CPU contains a 32-KB, Level-1 8-way set-associative instruction cache with a 32-byte line size. The high degree of set associativity ensures the most efficient use of the processor threads by reducing cache-conflict misses. Both the cache instruction and cache tag arrays are protected by parity. When a parity error is detected, the instruction is discarded, the cache line is invalidated, and a new instruction fetch is executed. This process is accomplished without software intervention.

2.2.5 Level-1 Data Cache

Each XLS processor core contains a 32-KB, Level-1, 8-way set-associative data cache with 32-byte line size. As with the instruction cache, the high degree of set associativity reduces cache-conflict misses.

The data portion of the cache array is protected by ECC (SECDED) for correction of single bit errors without software intervention. All multiple-bit errors are detected and then passed to the software via a cache error exception. The cache tag arrays are protected by parity. When a parity error is detected on an unmodified line, the data is discarded and the line is re-fetched from memory without software intervention.

2.2.6 DedicatedPlus Cache

XLS processors support a DedicatedPlus Cache mode for L1 Data and Instruction Caches. Using this mode, the software may partition the 32-KB I & D caches such that every nCPU (hardware thread) is guaranteed a minimum size. Cache coherency is maintained across the partitions created by software. All nCPUs are still able to read the code/data allocated by other nCPUs and hence benefit from the pre-fetching performed by the other nCPUs.

2.2.7 Coprocessor 0 (COP0) Registers

The integrated system control coprocessor (COP0) in the MIPS64 architecture is responsible for the virtual memory subsystem, the exception control system, and the diagnostics capability of the processor. Associated with COP0 are the on-chip system control registers. These provide the path through which the virtual memory system's page mapping is examined and modified, exceptions are handled, and operating modes are controlled. Additional registers in this group are used to implement a real-time cycle counting facility, to aid in cache diagnostic testing, and to assist in data error detection.

2.2.8 Coprocessor 2 (COP2) Registers

The COP2 co-processor registers provide access to the Fast Messaging Network. See Chapter 12, “Fast Messaging Network” for more information.

2.3 Memory SubSystem

2.3.1 Memory Management Unit

A full-featured Memory Management Unit (MMU) is included for each core processor. Each MMU uses on-chip translation look-aside buffers (TLB) to translate virtual addresses into physical addresses. Three modes of execution privilege are available to system software to provide a secure environment for user processes and provide configurability of address space depending on user applications. These execution modes are:

- User mode
- Supervisor mode
- Kernel mode

A Joint TLB (JTLB) is used for both instruction and data translations. It is fully associative, mapping 128 virtual pages to their corresponding physical addresses. It is organized as 64 pairs of even-odd entries, and maps a virtual address and address space identifier into the large, 64 GB physical address space.

Page size can be configured, on a per-entry basis to use page sizes in the range of 4 KB to 256 MB (in multiples of 4). The user has a choice of using the hardware supported random replacement algorithm or implementing their own algorithm in software. In addition, there is a mechanism to allow a number of mappings to be locked into the JTLB. There is also information contained in the JTLB that controls the cache coherency protocol for each page.

In addition to the JTLB there are a Data TLB (DTLB) and Instruction TLB (ITLB). The 32-entry ITLB and 32-entry DTLB both support a range of entries from 4 KB to 256 MB. They improve performance by allowing instruction address translation and data address translation to occur in parallel. This minimizes contention for the JTLB, eliminates the timing-critical path of translating through a large associative array, and saves power. Both are filled from the JTLB when a miss occurs. This operation is transparent to the user.

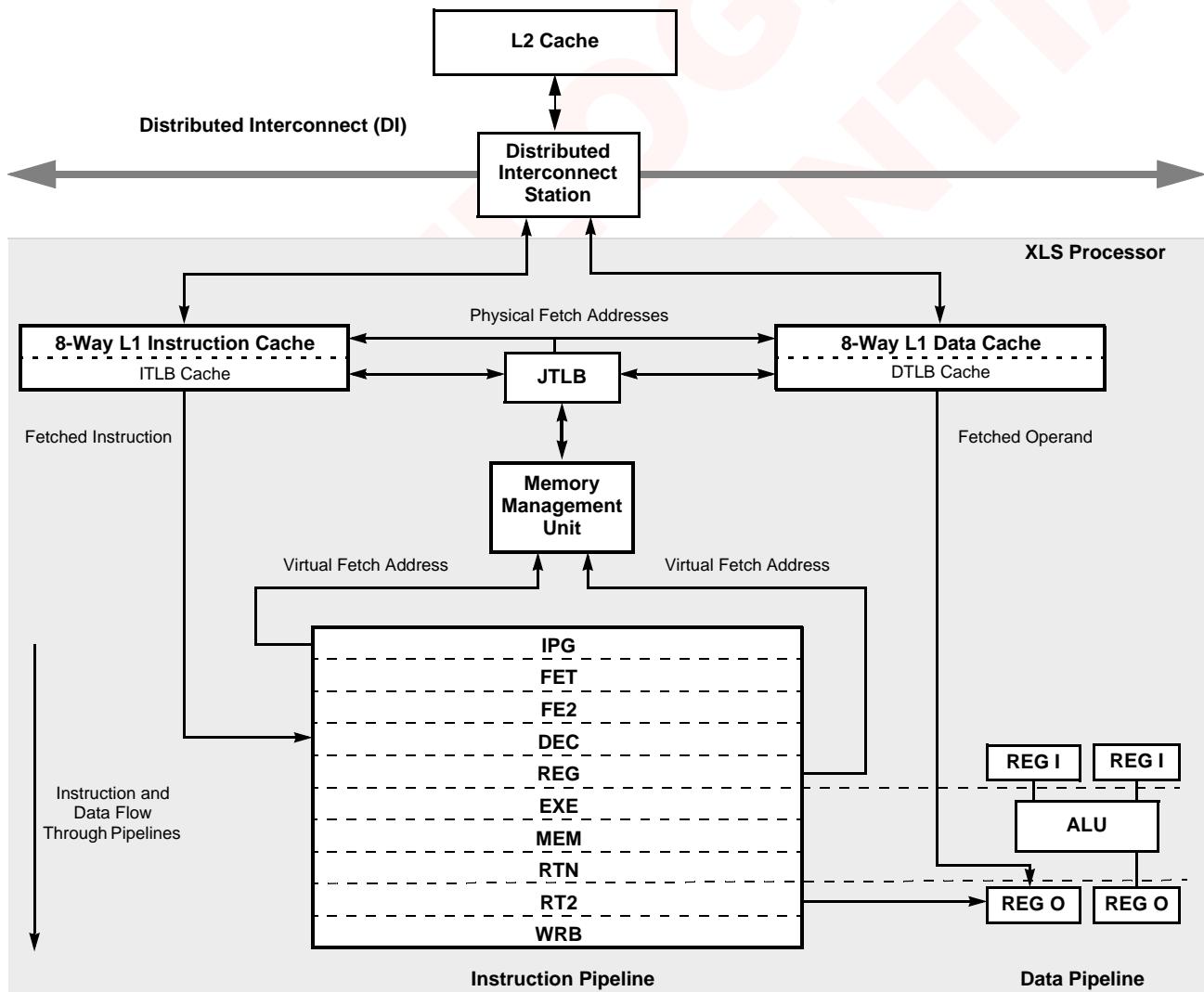
2.4 CPU Instruction Path

2.4.1 Processor Pipeline

The XLS processor core uses a 10-stage pipeline for integer operations, loads, stores, and other non floating-point operations as shown in [Figure 2-2](#). The high degree of pipelining allows the core frequency to scale with process enhancements. The pipeline issues one instruction every cycle with in-order execution of all issued instructions. A stall-on-use model is implemented to minimize the number of stalls in the pipeline. For example, a miss on a memory load will not cause the main pipeline to stall until the consumer of the load operation enters the REG (Register Read) stage.

The pipeline is designed to automatically handle up to four independent hardware contexts concurrently. The 4-way multi-threaded design enables a very high utilization of the processor resources, thus maximizing performance across the concurrently executing threads.

Figure 2-2. The XLS Processor MIPS64 CPU Core Pipelines



The following are the major functions for each one of the ten stages of the pipeline:

1. IPG — Instruction Program Generation

The IPG stage generates the next instruction program counter address.

2. FET — Instruction Fetch 1

The FET stage is the first cycle of the instruction fetch for the predicted path.

3. FT2 — Instruction Fetch 2

The FT2 stage fetches the instruction from the instruction cache.

4. DEC — Decode Buffer

The DEC stage receives the fetched instruction, decodes the addressing and executes branch and jump predictions. The buffer is a four deep replay register with bypass capability.

5. REG — Register File Read

The REG stage fetches operands from the registers or initiates a data cache memory fetch.

6. EXE — Execution

The EXE stage executes the instruction using register operands or generates the addresses for load instructions.

7. MEM — Memory Access

The MEM stage is the first cycle of the data cache access and resolves the branch prediction.

8. RET — Data Cache Fetch 1

The RET stage is the second cycle of the data cache access.

9. RT2 — Data Cache Fetch 2

The RT2 stage is the return cycle for the data cache load back into the registers.

10. WRB — Register Write Back

The WRB stage writes the results of the instruction execution back into the registers and retires the instruction.

2.5

Instruction Timing

The robust XLS pipeline always guarantees correctness of data for subsequent instructions in the stream. While most instructions are executed in a single cycle, some instructions as detailed below in this section, consume more than one cycle.

If the interceding cycles on a 'multi-cycle' instruction are optimally filled with other unrelated instructions, the performance of the pipeline may be improved. This can be achieved in two ways:

1. Allowing the compiler to automatically optimize the instruction stream by passing optimization (-On) flags. The machine description provided for the XLS pipeline will be used by the compiler to best optimize the instruction sequence.
2. Manually writing assembler functions or macros that are very performance sensitive. In this case, the instructions listed in [Table 2-1](#) can be used to interleave instructions that make use of unused issue slots in the pipeline.

Table 2-1. Instruction Timing

Instruction/Category	Cycles & Considerations
Load	4 cycles
MFC0 and MFC2	3 cycles
MFCR	6 cycles
MOVZ and MOVN	2 cycles
[D]CLO, [D]CLZ	2 cycles
SLT to non-branch instruction	2 cycles
Multiply: 32 bit	5 cycles (issue rate is 1 every 5 cycles)
Multiply: 64 bit	7 cycles (issue rate is 1 every 7 cycles)
Divide	8-72 cycles (dependent upon operands, issue rate is 1 every 8-72 cycles)
Shift result to re-use in shift amount field	2 cycles
Branch miss-predict	5 cycles
MTC0 [D]EPC followed by [D]ERET	5 cycles
MTC0 status followed by dependent instruction	6 cycles
MSGLD and MSGSEND followed by MFC2 status	5 cycles
Unaligned load that crosses 8-byte boundary	5 cycles
Unaligned store that crosses 8-byte boundary	2 cycles
All other instructions	1 cycle

2.5.1 Threading Considerations

The single-issue XLS pipeline is 4-way multi-threaded, but can also be run as a single- or dual-threaded core. This uniqueness must be considered when writing assembler routines and when running in an *n*-way threaded configuration.

Consider the scenario where the pipeline operates in the default 4-way threaded mode, with round-robin scheduling. A thread issuing a load has the load satisfied in 4 cycles, which is already optimal as the data becomes available when the thread is scheduled again.

In contrast, consider the same event occurring on a pipeline that is configured to run in single-threaded mode (i.e. only thread 0 is enabled). In this case, there are three delay cycles immediately after the load as the register fill does not occur until the fourth cycle. Compiling code with optimization turned on will automatically produce efficient code. Code written using assembler instructions can achieve the same benefit by introducing unrelated instructions in such empty slots.

The foregoing discussion applies not only to the load instruction, but also all instructions that consume more than one cycle as shown in [Table 2-1](#).

2.5.2 Load Timing

The XLS core implements a “stall on use” model; that is, a load that misses the L1 cache does not immediately cause a stall and put the thread to sleep. Instead a subsequent instruction performing a register access resulting from the load will cause a stall, if the data has not already been fetched into the register.

Cache lines that already reside in the L1 data cache take the least load time of 4 cycles. If data is prefetched (including stores), subsequent instructions of near proximity performing the load/store can be assured a hit in the L1.

2.5.3 Pipeline Hazards

The XLS pipeline is a single-issue, in-order execution pipe that provides full interlocking to prevent both structural and data hazards. Generally, only events that simultaneously involve data- and instruction-synchronization need special consideration to avoid a control hazard. Events such as interrupts and exceptions always cause an automatic pipeline flush.

Accesses to both coprocessors (CP0 and CP2) are also fully interlocked-i.e. an immediate read followed by a write to a co-processor register will always return a correct result to the read. The hazard is automatically prevented by entering the instruction in the replay queue until the resulting data is available.

With respect to control hazards, there exist instructions that change the operational state of the thread. The state change caused by an instruction becomes effective at the next instruction. For example, the MTC0 instruction changing the EXL or ERL bit is effective during the next instruction-i.e. the intervening cycles between the two instructions comprise an interrupt window where the state change desired cannot be considered committed.

TLB instructions are a special class that affect MMU state and behavior. Operations such as a random or indexed TLB write are not guaranteed to be immediately visible-i.e. the resulting or expected behavior is not immediate. It is advised that the pipeline always be flushed using an ERET instruction following a TLB operation that changes the MMU state.

2.6 Code Compatibility

The XLS processor shares the core microarchitecture with the XLR processor. Generally, any common code written for a MIPS32/MIPS64 processor and compiled with a standard MIPS compiler can be executed on the XLS or XLR processors.

Writing manual assembler code requires some careful consideration, especially with system registers. The following facts must be kept in mind while writing or porting code to run on the XLS/XLR processors.

1. The XLS/XLR processors do not feature a float-point co-processor (CP1) and hence do not support any floating point instructions. Soft-floating point emulation must be used where necessary.
2. The Fast Messaging Network is implemented using the MIPS coprocessor model via CP2. The instructions MSGSND, MSGLD and MSGWAIT are specific to the XLS/XLR processors.
3. Processor core registers have reserved or specific fields that may apply only to XLS or XLR, or a specific subset of processor models. Where necessary, an abstraction API should be used to differentiate model- or processor-specific configurations.

2.7 CPU Reset Control

Note that each CPU can be reset using the **GPIO_CPU_RST** register. See [Section 23.3.5.13 GPIO_CPU_RST](#).

NETLOGIC
CONFIDENTIAL



Chapter 3 Virtual MIPS Mode

3.1 Introduction

Virtual MIPS Mode provides a way of allowing multiple heterogeneous operating systems to reside cooperatively within a single XLS processor system.

This unique operational mode provides a straightforward means of isolating heterogeneous operating systems on multiple processor cores. Virtual MIPS Mode is designed to increase the flexibility of the MIPS architecture by allowing the unmapped regions of each operating system to be virtualized, while requiring only minimal changes to startup code in the affected operating systems.

3.2 Virtual MIPS Mode Operation in 32-bit Compatibility Mode

One of the central issues for designing a multiple heterogeneous operating system environment in a multi-core system is how to resolve any potential conflict between the operating systems over unmapped memory regions. Ordinarily, unmapped kseg0 and kseg1 regions are reserved for the operating system kernel. But where multiple heterogeneous operating systems must interoperate, unmapped memory areas used for kernel data structures must be mapped to prevent overlapping access by the different operating systems.

The solution is to allow one operating system to use kseg0 and kseg1 unmapped, but to virtualize these segments for all other co-resident operating systems through the TLB. The XLS processor family makes this possible by optionally allowing these segments to be virtualized through hardware register extensions.

Programming Virtual MIPS Mode for two heterogeneous operating systems on multiple nCPU™ NetLogic virtual CPUs (nCPUs) is as follows. The method can be extended arbitrarily. The first operating system uses kseg0 and kseg1 in unmapped mode, as usual. For each subsequent operating system in other nCPUs the following actions must be performed in the following order:

- Two TLB entries are allocated to each unmapped region (kseg0 and kseg1).
- The TLB entries should be set to the largest TLB size (256 MB).
- The TLB entries should be in the wired region to prevent unintentional removal from the TLB.
- The TLB entries can be set by either the XLS boot loader software or by the operating system.
- Cached / uncached properties for kseg 0 and 1 are set by TLB entries.
- Virtual MIPS Mode is set per thread on all but the first operating system.

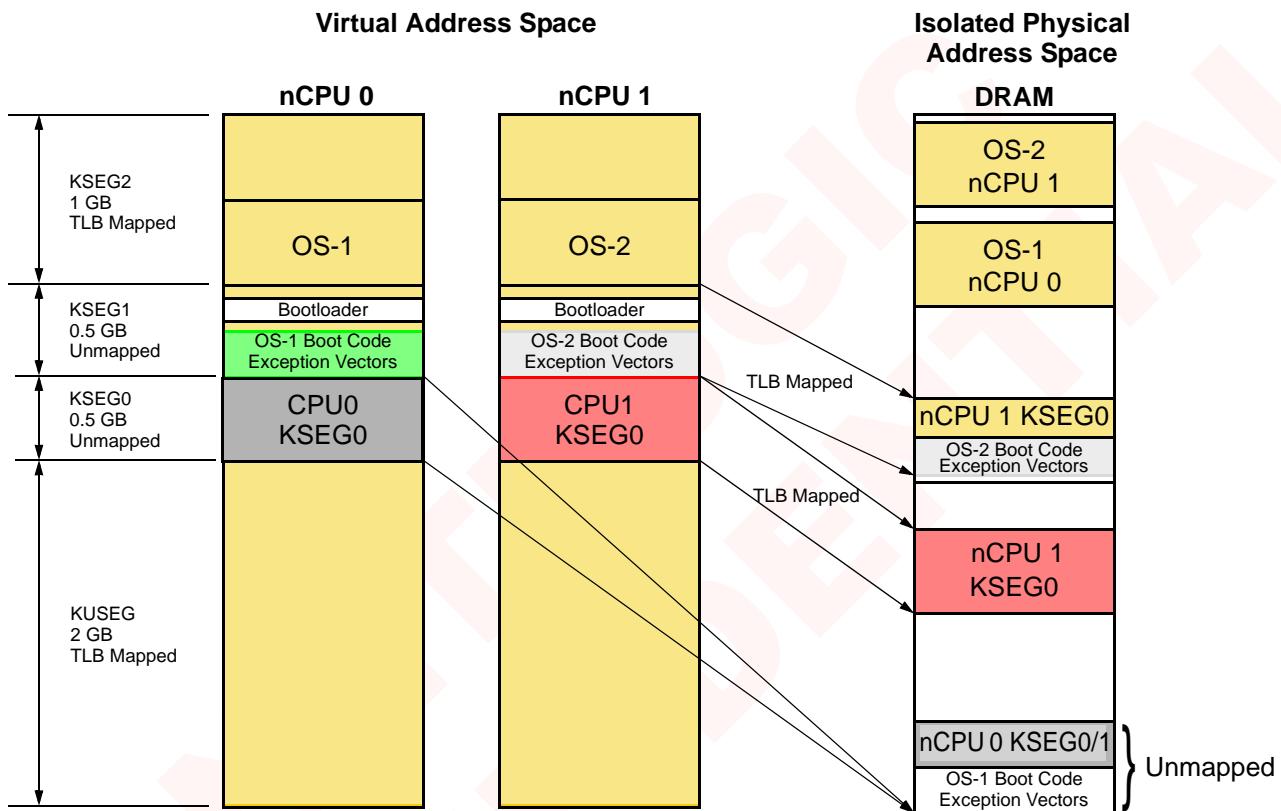
Note: Virtual MIPS Mode should be enabled only when the processor is in Kernel mode. If turned on in User or Supervisor mode, Virtual MIPS Mode has no effect.

As defined in the MIPS 64 specification, in User mode and Supervisor mode, an Address Error is returned when referencing kseg0 and kseg1.

Note: Startup code in the affected operating systems is primarily all that will need to be changed in order to utilize Virtual MIPS Mode.

Figure 3-1 shows an example of Virtual MIPS Mode for two operating systems. One nCPU runs OS-1 in standard MIPS-compliant mode. OS-2 uses the Virtual MIPS mode and uses the TLB entries to map the unmapped space of kseg0 and kseg1 to avoid conflict with OS-1.

Figure 3-1. Virtual MIPS Mode



3.3 XLS-Specific Registers for Virtual MIPS Mode

There are XLS-specific register settings to enable Virtual MIPS Mode for the Instruction side and Data side memory systems.

3.3.1 Instruction-Side Register Settings

The ICU_DEFFEATURE register contains two read-writable bit-fields for Virtual MIPS Mode, as follows:

- `force_mapped_mode` If set, I-side accesses to unmapped regions of the MIPS address map (not including kuseg with Status.ERL = 1) are handled as mapped accesses (i.e. they undergo translation in the IuTLB/mainTLB).
- `force_mapped_erl` If set, I-side accesses to kuseg with Status.ERL = 1 are handled as mapped accesses (i.e. they undergo translation in the IuTLB/mainTLB).

See [Section 4.4.3.7, “ICU_DEFFEATURE”](#) for details of this register.

Note, as defined in the MIPS64 architecture, kuseg dynamically becomes mapped/cached or unmapped/uncached depending on the ERL bit in the Status register. If ERL = 1, then it is unmapped/uncached, otherwise it is mapped/cached. This must be handled dynamically because kuseg may contain cache error exception handlers. On cache error, ERL is set, forcing kuseg to be unmapped/uncached. This prevents attempts to reference the cache that could possibly generate infinite exception loops.

If set, the `force_mapped_mode` bit affects all segments except kuseg when ERL = 1, turning all unmapped transactions to mapped transactions. The case for kuseg where ERL = 1 is governed by the `force_mapped_erl` bit. These two bits manage the instruction side of Virtual MIPS Mode.

3.3.2 Data-Side Register Settings

The L1D_CONFIG0 register contains two bits to manage the data side of Virtual MIPS Mode.

- `force_mapped_data` If set, D-side accesses to unmapped regions of the MIPS address map (not including kuseg with Status.ERL = 1) are handled as mapped accesses (i.e. they undergo translation in the DuTLB/mainTLB).
- `force_mapped_data_erl` If set, D-side accesses to kuseg with Status.ERL = 1 are handled as mapped accesses (i.e. they undergo translation in the DuTLB/mainTLB).

See [Section 4.4.3.19 on page 147](#) for details of this register.

Note: Correct operation is only guaranteed if the instruction side and data side settings are the same. Here is an example of correct settings:

```
force_mapped_mode = 1
force_mapped_erl = 0
force_mapped_data = 1
force_mapped_data_erl = 0.
```

NETLOGIC
CONFIDENTIAL



Chapter 4 XLS Processor Core Registers

4.1 XLS Processor Core Registers

The registers available in the XLS processor include:

- Thirty-two 64-bit MIPS64 general-purpose registers
- A pair of special-purpose registers
- Special-purpose program counter (PC) which is affected only indirectly by certain instructions (not an architecturally visible register)
- Coprocessor 0 registers
- Coprocessor 2 registers
- XLS-specific control registers

Note: A MIPS64 processor always produces a 64-bit result, even for those instructions which are architecturally defined to operate on 32 bits. Such instructions typically sign-extend their 32-bit result into 64 bits. In so doing, 32-bit programs work as expected, even though the registers are actually 64 bits wide rather than 32.

4.1.1 General-Purpose Registers

Two of the general-purpose registers have assigned functions:

- r0 is hard-wired to a value of zero, and can be used as the target register for any instruction whose result is to be discarded. r0 can also be used as a source when a zero value is needed.
- r31 is the destination register used by JAL, BLTZAL, BLTZALL, BGEZAL, and BGEZALL without being explicitly specified in the instruction word. Otherwise, r31 is available as a normal register.

The remaining registers are available for general-purpose use.

4.1.2 Special-Purpose Registers

A pair of special-purpose registers (HI and LO) hold the results of integer multiply, divide, and multiply-accumulate operations.

- During a multiply operation, the HI and LO registers store the product of integer multiply
- During a multiply-add or multiply-subtract operation, the HI and LO registers store the result of the integer multiply-add or multiply-subtract
- During a division, the HI and LO registers store the quotient (in LO) and remainder (in HI) of integer divide
- During a multiply-accumulate, the HI and LO registers store the accumulated result of the operation

4.2 Coprocessor 0 (COP0) Registers

The following sections define the Coprocessor 0 registers, showing the bit definitions and read/write requirements.

Structure of register descriptions in this chapter is:

- Register name
- Register index
- Register text description
- Bit fields diagram
- Bit fields descriptions

- Note:**
1. Fields identified as not implemented in an XLS, or which are marked **Reserved**, are set to zero.
 2. Reserved bits and words should not be modified by the user.
 3. Unless otherwise specified, these bits read as '0' and should be written as '0'.

4.2.1 Definition of Terms and Usage

Table 4-1. Bit Field Definitions

Field	Hardware Interpretation	Software Interpretation
R/W	A field in which all bits are readable and writable by software and, potentially, by hardware. Hardware updates of this field are visible by software read. Software updates of this field are visible by hardware read. If the Reset State of this field is "Undefined", either software or hardware must initialize the value before the first read. Otherwise it will return a predictable value.	
R	A field which is either static or is updated only by hardware. If the Reset State of this field is either '0' or 'Preset', hardware initializes this field to zero or to the appropriate state, respectively, on powerup. If the Reset State of this field is 'Undefined', hardware updates this field only under those conditions specified in the description of the field.	A field to which the value written by software is ignored by hardware. Software may write any value to this field without affecting hardware behavior. Software reads of this field return the last value updated by hardware. If the Reset State of this field is "Undefined," software reads of this field result in an UNPREDICTABLE value except after a hardware update done under the conditions specified in the description of the field.
0	A field which hardware does not update, and for which hardware can assume a zero value.	A field to which the value written by software must be zero. Software writes of non-zero values to this field may result in UNDEFINED behavior of the hardware. Software reads of this field return zero as long as all previous software writes are zero. If the Reset State of this field is "Undefined", software must write this field with zero before it is guaranteed to read as zero.
Reserved	Must be written as 0x0	Returns 0x0 on read

4.2.2 Coprocessor 0 Register Summary

The Coprocessor 0 (COP0) registers are summarized in [Table 4-2](#).

Table 4-2. Coprocessor 0 Register Summary

Register			Sel	Description	Unit
Num	Name	Bits			
0	Index	32	0	Index Entry into TLB Array	MMU
1	Random	32	0	Randomly generated index into TLB Array	MMU
2	EntryLo0	64	0	Low-order portion of TLB entry for even-numbered virtual pages	MMU
3	EntryLo1	64	0	Low-order portion of TLB entry for odd-numbered virtual pages	MMU
4	Context	64	0	Pointer to page table entry in memory	MMU
5	PageMask	32	0	Control for variable page size in TLB entries	MMU
6	Wired	32	0	Controls the number of fixed (wired) TLB entries	MMU
7	Reserved		All	Reserved	
8	BadVAddr	64	0	Reports the address for the most recent address-related exception	MMU
9	Count	32	0	Processor cycle count	IEU
9	EIRR	64	6	Extended Interrupt Request register	IEU
9	EIMR	64	7	Mask register for EIRR register.	IEU
10	EntryHi	64	0	High-order portion of the TLB entry	MMU
11	Compare	32	0	Timer interrupt control	IEU
12	Status	32	0	Processor status and control	IEU
13	Cause	32	0	Cause of last general exception	IEU
14	Exception Program Counter	64	0	Program counter at last exception	IEU
15	ProcessorID	32	0	Processor identification and revision	IEU
15	EBase	32	1	Exception Base	IEU
16	Config0	32	0	Configuration register 0	IEU
	Config1	32	1	Configuration register 1	IEU
	Config2	32	2	Configuration register 2	IEU
	Config3	32	3	Configuration register 3	IEU
	Config7	32	7	Configuration register 7	IEU
17	Reserved		All	Reserved	
18	WatchLo	64	0-n	Watchpoint address	MMU
19	WatchHi	32	0-n	Watchpoint control	MMU
20	XContext	64	0	Extended Addressing Page Table Context	MMU
21	Reserved		all	Reserved	
22	OS Scratch0 – Scratch7	64	0-7	Eight 64-bit scratch-pad registers	IEU
23	Debug	32	0	EJTAG Debug register	IEU
24	DEPC	32	0	Program counter at last EJTAG debug exception	IEU

Table 4-2. Coprocessor 0 Register Summary (continued)

Register			Sel	Description	Unit
Num	Name	Bits			
25	PerfCntrCtl0	32	0	Performance counter Control register 0	PRF
	PerfCntr0	32	1	Performance counter Counter register 0	PRF
	PerfCntrCtl1	32	2	Performance counter Control register 1	PRF
	PerfCntr1	32	3	Performance counter Counter register 1	PRF
26	ErrorCtl			Not implemented	
27	CacheError			Not implemented	
28	TagLo	64	0,2	Low-order portion of cache tag interface	LSU
28	DataLo			Not Implemented	
29	TagHi	64	0,2	High-order portion of cache tag interface	LSU
29	DataHi			Not Implemented	
30	ErrorEPC	64	0	Program counter at last error	IEU
31	DESAVE	64	0	EJTAG debug exception save register	IEU

4.2.3 Coprocessor 0 Register Descriptions

4.2.3.1 Index

COP0 Register: 0

This register contains the index used to access the TLB for TLBP, TLBR, and TLBWI instructions.

31	30	n	$n-1$	0
TLB_MATCH	Reserved			Index

Bits	Name	Description	R/W	Reset
31	TLB_MATCH	TLB match occurred Hardware writes this bit during execution of the TLBP instruction to indicate whether a TLB match occurred: 0: A match occurred, and the Index field contains the index of the matching entry. 1: Probe Failure. No match occurred and the Index field is UNPREDICTABLE.	R	Undefined
30: n	Reserved	Reserved = 0	0	0
$n-1:0$	Index	TLB index Software writes this field to provide the index to the TLB entry referenced by the TLBR and TLBWI instructions. Hardware writes this field with the index of the matching TLB entry during execution of the TLBP instruction. If the TLBP fails to find a match, the contents of this field are UNPREDICTABLE. $n=4$: Partitioned 4-thread mode $n=5$: Partitioned 2-thread mode $n=6$: One-thread mode: $n=6$ Global mode	R/W	Undefined

Note: The operation of the processor is UNDEFINED if a value greater than or equal to the number of TLB entries is written to the Index register.

4.2.3.2 Random

COP0 Register: 1

Used to index the TLB during a TLBWR instruction. The total number of TLB entries is 64.

31	n	$n-1$	0
Reserved			Random

Bits	Name	Description	R/W	Reset
31: n	Reserved	Reserved = 0	0	0
$n-1:0$	Random	TLB Random Index $n=4$: Partitioned 4-thread mode $n=5$: Partitioned 2-thread mode $n=6$: One-thread mode; $n=6$ Global mode	R	TLB Entries – 1

- The value of the Random register varies between an upper and lower bound as follows:
 - A lower bound is set by the number of TLB entries reserved for exclusive use by the operating system (the contents of the Wired register). The entry indexed by the Wired register is the first entry available to be written by a TLB Write Random operation.
 - An upper bound is set by the total number of TLB entries available to this thread minus 1.
- Within the required constraints of the upper and lower bounds, the manner in which the processor selects values for the Random register is set by the hardware.
- The processor initializes the Random register to the upper bound at Reset, and when the Wired register is written.

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.3.3 EntryLo0**COP0 Register: 2**

EntryLo0 and EntryLo1 registers act as the interface between the TLB and the TLBP, TLBR, TLBWI, and TLBWR instructions.

63	34	33	6	5	3	2	1	0
	Fill	PFN	C	D	V	G		

Bits	Name	Description	R/W	Reset
63:34	Fill	Fill Ignored on write; returns zero on read.	R	0
33:6	PFN	Page Frame Number Corresponds to bits [39:12] of the physical address.	R/W	Undefined
5:3	C (Coherency)	Coherency attribute of the page 000: Unimplemented — will cause Address Error exception 001: Unimplemented — will cause Address Error exception 010: Uncached 011: Cachable 100: Cachable 101: Cachable 110: Cachable 111: Unimplemented — will cause Address Error exception	R/W	Undefined
2	D (Dirty)	“Dirty” bit Indicates that the page is writable. 1: Stores to the page are permitted 0: Stores to the page cause a TLB Modified exception. Kernel software may use this bit to implement paging algorithms that require knowing which pages have been written. If this bit is always 0 when a page is initially mapped, the TLB Modified exception that results on any store to the page can be used to update kernel data structures that indicate that the page was actually written.	R/W	Undefined
1	V (Valid)	Valid bit Indicates that the TLB entry and thus the virtual page mapping are valid. 1: Accesses to the page are permitted 0: Accesses to the page cause a TLB Invalid exception	R/W	Undefined
0	G (Global)	Global bit On a TLB write, the logical AND of the G bits from both EntryLo0 and EntryLo1 becomes the G bit in the TLB entry. If the TLB entry G bit is a one, ASID comparisons are ignored during TLB matches. On a read from a TLB entry, the G bits of both EntryLo0 and EntryLo1 reflect the state of the TLB G bit.	R/W	Undefined

- The contents of the EntryLo0 and EntryLo1 registers are not defined after an address error exception and some fields may be modified by hardware during the address error exception sequence. Software writes of the EntryHi register (via MTC0 or DMTC0) do not cause the implicit update of address-related fields in the BadVAddr or Context registers.

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.3.4 EntryLo1

COP0 Register: 3

EntryLo0 and EntryLo1 act as the interface between the TLB and the TLBP, TLBR, TLBWI, and TLBWR instructions.

63	34	33	6	5	3	2	1	0
	Fill	PFN	C	D	V	G		

Bits	Name	Description	R/W	Reset
63:34	Fill	Fill Ignored on write; returns zero on read.	R	0
33:6	PFN	Page Frame Number Corresponds to bits [39:12] of the physical address.	R/W	Undefined
5:3	C (Coherency)	Coherency attribute of the page 000: Unimplemented — will cause Address Error exception 001: Unimplemented — will cause Address Error exception 010: Uncached 011: Cachable 100: Cachable 101: Cachable 110: Cachable 111: Unimplemented — will cause Address Error exception	R/W	Undefined
2	D (Dirty)	“Dirty” bit Indicates that the page is writable. 1: stores to the page are permitted 0: stores to the page cause a TLB Modified exception. Kernel software may use this bit to implement paging algorithms that require knowing which pages have been written. If this bit is always ‘0’ when a page is initially mapped, the TLB Modified exception that results on any store to the page can be used to update kernel data structures that indicate that the page was actually written.	R/W	Undefined
1	V (Valid)	Valid bit Indicates that the TLB entry, and thus the virtual page mapping are valid. 1: Accesses to the page are permitted 0: Accesses to the page cause a TLB Invalid exception	R/W	Undefined
0	G (Global)	Global bit. On a TLB write, the logical AND of the G bits from both EntryLo0 and EntryLo1 becomes the G bit in the TLB entry. If the TLB entry G bit is a one, ASID comparisons are ignored during TLB matches. On a read from a TLB entry, the G bits of both EntryLo0 and EntryLo1 reflect the state of the TLB G bit.	R/W	Undefined

- The contents of the EntryLo0 and EntryLo1 registers are not defined after an address error exception and some fields may be modified by hardware during the address error exception sequence. Software writes of the EntryHi register (via MTC0 or DMTC0) do not cause the implicit update of address-related fields in the BadVAddr or Context registers.

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.3.5 Context**COP0 Register: 4**

Contains a pointer to an entry in the page table entry (PTE) array which is an operating system data structure that stores virtual-to-physical translations.

63						32
PTEBase[40:9]						
31	23	22			4	3 0
PTEBase[8:0]	BadVPN2				<i>Reserved</i>	

Bits	Name	Description	R/W	Reset
63:23	PTEBase	This field is for use by the operating system and is normally written with a value that allows the operating system to use the Context Register as a pointer into the current PTE array in memory.	R/W	Undefined
22:4	BadVPN2	This field is written by hardware on a TLB exception. It contains bits VA[31:13] of the virtual address that caused the exception.	R	Undefined
3:0	Reserved	<i>Reserved = 0</i>	0	0

- During a TLB miss, the operating system loads the TLB with the missing translation from the PTE array. The Context register is primarily intended for use with the TLB Refill handler, but is also loaded by hardware on an XTLB Refill and may be used by software in that handler.
- The Context register duplicates some of the information provided in the BadVAddr register, but is organized in such a way that the operating system can directly reference a 16-byte structure in memory that describes the mapping.
- A TLB exception (TLB Refill, XTLB Refill, TLB Invalid, or TLB Modified) causes bits VA[31:13] of the virtual address to be written into the BadVPN2 field of the Context register. The PTEBase field is written and used by the operating system.
- The BadVPN2 field of the Context register is not defined after an address error exception and this field may be modified by hardware during the address error exception sequence.

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.3.6 PageMask**COP0 Register: 5**

Used for reading from and writing to the TLB. It holds a comparison mask that sets the variable page size for each TLB entry.

31	29	28	13	12	0
RSVD		Mask		Reserved	

Bits	Name	Description	R/W	Reset
31:29	Reserved	Reserved = 0	0	0
28:13	Mask	The Mask field is a bit mask in which a 1 bit indicates that the corresponding bit of the virtual address should not participate in the TLB match. 4KB = 0000000000000000 16KB = 0000000000000011 64KB = 0000000000001111 256KB = 0000000000111111 1MB = 0000000011111111 4MB = 0000001111111111 16MB = 0000111111111111 64MB = 0011111111111111 256MB = 1111111111111111	R/W	Undefined
12:0	Reserved	Reserved = 0	0	0

- Mask[28:13] are all implemented in this design.
- Software may determine which page sizes are supported by examining the value returned from a read of the PageMask register. If a pair of bits reads back as ones, the processor implements that page size.
- The operation of the processor is UNDEFINED if software loads the PageMask register with a value other than one of those listed in the table above.

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.3.7 Wired**COP0 Register: 6**

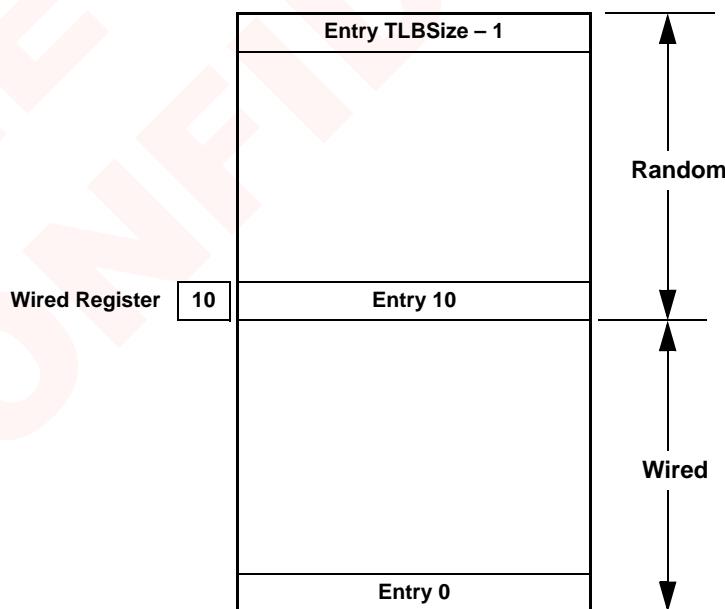
Specifies the boundary between the wired and random entries in the TLB.

31	n	$n-1$	0
Reserved			Wired

Bits	Name	Description	R/W	Reset
31: n	Reserved	Reserved = 0	0	0
$n-1:0$	Wired	Wired[$n-1:0$] TLB wired boundary $n=4$: Partitioned 4-thread mode $n=5$: Partitioned 2-thread mode $n=6$: One-thread mode $n=6$: Global mode	R/W	0

- The width of the Wired field is calculated in the same manner as that described for the index register. Wired entries are fixed, non-replaceable entries which are not overwritten by a TLBWR instruction. Wired entries can be overwritten by a TLBWI instruction. [Figure 4-1](#) shows the Wired and Random entries in the TLB.
- The Wired register is set to zero at Reset. Writing the Wired register causes the Random register to reset to its upper bound.
- The operation of the processor is UNDEFINED if a value greater than or equal to the number of TLB entries is written to the Wired register.

Figure 4-1. Wired and Random Entries in TLB



Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.3.8 BadVAddr**COP0 Register: 8**

Captures the most recent virtual address that caused one of the following exceptions:

- Address error (AdEL or AdES)
- TLB/XTLB Refill
- TLB Invalid (TBLI, TLBS)
- TLB Modified

63	BadVAddr[63:32]	32
31	BadVAddr[31:0]	0

Bits	Name	Description	R/W	Reset
63:0	BadVAddr	Bad virtual address	R	Undefined

- The BadVAddr register does not capture address information for cache or bus errors, or for Watch exceptions, since none is an addressing error.

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.3.9 Count**COP0 Register: 9, Select 0**

Acts as a timer.

31	Count	0
----	-------	---

Bits	Name	Description	R/W	Reset
31:0	Count	Interval counter	R/W	Undefined

- The Count Register is incremented every processor cycle. Wait instructions do not stop it.
- The Count register increments at a constant rate, whether or not an instruction is executed, retired, or any forward progress is made through the pipeline. The rate at which the counter increments is implementation-dependent, and is a function of the pipeline clock of the processor, not the issue width of the processor. The Count register can be written for functional or diagnostic purposes, including at reset or to synchronize processors.
- In Debug Mode, the CountDM bit in the Debug Register controls whether the counter is active. See [Section 4.2.4.28, “Debug”](#).

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.3.10 EIRR**COP0 Register: 9, Select 6**

Extended Interrupt Request register.

63	EIRR[63:32]	32
31	EIRR[31:0]	0

Bits	Name	Description	R/W	Reset
63:0	EIRR	<p>This is an extension of the Interrupt Pending bits IP7:IP0 in the CAUSE Register.</p> <ul style="list-style-type: none"> A read of the EIRR returns a 64 bit vector, each bit indicating an interrupt request pending. A write of the EIRR implements a bit-wise clear of EIRR. WRITE_FUNCTION: $EIRR = \sim EIRR_WRDATA \& EIRR$. (Does not apply to Bits 7, 1, 0. See following text). 	R/W	Undefined

- EIRR[7:0] == CAUSE[15:8]. These will always be the same.
- EIRR[1:0] are software interrupts, set by writing to CAUSE[9:8] or EIRR[1:0]. These bits are writable by software.
- EIRR[7]/CAUSE[15] is set when COUNT == COMPARE. It can be cleared by writing to the COMPARE register, but is not affected by writes to EIRR.
- EIRR[6:2] / CAUSE[14:10] can also be cleared by writing to EIRR but are not otherwise affected by writes to CAUSE.

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.3.11 EIMR**COP0 Register: 9, Select 7**

Extended Interrupt Mask register.

63	32
EIMR[63:32]	
31	0
EIMR[31:0]	

Bits	Name	Description	R/W	Reset
63:0	EIMR	<p>This is an extension of the Interrupt Mask bit IM7:IM0 in the STATUS register.</p> <ul style="list-style-type: none"> Contains a 64 bit bitwise mask for the EIRR. For compatibility purposes the lower mask bits are ORed together with the STATUS interrupt mask bits. <p>EFFECTIVE_MASK = EIMR[63:8] (EIMR[7:0] STATUS[15:8])</p> <ul style="list-style-type: none"> In our case, EIRM[7:0] and Status[15:8] are shadowed and will always be the same. Interrupt Vectors 0,1, and 7 are reserved, and if an interrupt is delivered to the CPU with these vectors, it will be ignored. 	R/W	Undefined

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.3.12 EntryHi

COP0 Register: 10

Contains the virtual address match information used for TLB read, write, and access operations.

63	62	61		13	12	9	8	7	0
R			VPN2		Reserved	MV_L		ASID	

Bits	Name	Description	R/W	Reset
63:62	R	Virtual memory region, corresponding to VA[63:62] 00: xuseg: user address region 01: xsseg: supervisor address region. 10: Reserved 11: xkseg: kernel address region This field is written by hardware on a TLB exception or on a TLB read, and is written by software before a TLB write.	R/W	Undefined
61:13	VPN2	VA[61:13] of the virtual address (virtual page number / 2). This field is written by hardware on a TLB exception or on a TLB read, and is written by software before a TLB write. The default width of this field implicitly limits the size of each virtual address space to 64 bits.	R/W	Undefined
12:9	Reserved	Reserved = 0	0	0
8	MV_L	Only readable and writable in Virtual MIPS mode. In virtual MIPS mode, this bit can be used as a “master valid” bit with TLBWI or TLBWR instructions to invalidate entries in the main TLB. 0: This is a valid TLB entry 1: This is not a valid TLB entry, i.e., the entry will never match on a main TLB lookup or for a TLBP instruction.	R/W (only in virtual MIPS mode)	0
7:0	ASID	Address space identifier. This field is written by hardware on a TLB read and by software to establish the current ASID value for TLB write and against which TLB references match each entry’s TLB ASID field.	R/W	Undefined

- A TLB exception (TLB Refill, XTLB Refill, TLB Invalid, or TLB Modified) causes the bits of the virtual address corresponding to the R and VPN2 fields to be written into the EntryHi register. The ASID field is written by software with the current address space identifier value and is used during the TLB comparison process to determine TLB match.
- Software may determine the size of the virtual address space implemented by a processor by writing all ones to the EntryHi register and then reading the value back.
- The VPN2 and R fields of the EntryHi register are not defined after an address error exception and these fields may be modified by hardware during the address error exception sequence. Software writes of the EntryHi register (via MTC0 or DMTC0) do not cause the implicit write of address-related fields in the BadVAddr, Context, or XContext registers.

4.2.3.13 Compare**COP0 Register: 11**

Acts in conjunction with the Count register to implement a timer and timer interrupt function.

31	0
Compare	

Bits	Name	Description	R/W	Reset
31:0	Compare	Interval count compare value	R/W	Undefined

- The Compare register maintains a stable value and does not change on its own.
- When the value of the Count register equals the value of the Compare register, an interrupt request is combined in an implementation-dependent way with hardware interrupt 5 to set interrupt bit IP[7] in the Cause register. This causes an interrupt as soon as the interrupt is enabled.
- For diagnostic purposes, the Compare register is a read/write register. In normal use however, the Compare register is write-only. Writing a value to the Compare register clears the timer interrupt.

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.3.14 Status**COP0 Register: 12**

Contains the operating mode, interrupt enabling, and the diagnostic states of the processor.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RSVD	CU2	RSVD	CU0	RP	FR	RE	MX	PX	BEV	TS	SR	NMI	Reserved	CRY	
15							8	7	6	5	4	3	2	1	0
								KX	SX	UX		KSU	ERL	EXL	IE

Bits	Name	Description	R/W	Reset
31	Reserved	Reserved = 0	0	0
30	CU2	Controls access to Coprocessor 2 0: Access denied 1: Access allowed	R/W	Undefined
29	Reserved	Reserved = 0	0	0
28	CU0	Controls access to Coprocessor 0 0: Access denied 1: Access allowed Coprocessor 0 is always usable when the processor is running in Kernel Mode or Debug Mode, independent of the state of the CU0 bit.	R/W	Undefined
27	RP	Enables reduced power mode on some implementations. The specific operation of this bit is implementation-dependent. In the XLS processor it performs no function. Setting this bit is ignored.	R/W	0
26	FR	Reserved = 0	R/W	0
25	RE	Reversed Endianness 0: user mode uses configured endianness 1: user mode uses reversed endianness	R/W	
24	MX	Reserved = 0		
23	PX	Enables access to 64-bit operations in User mode, without enabling 64-bit addressing: 0: 64-bit operations are <i>not</i> enabled in User Mode 1: 64-bit operations are enabled in User Mode	R/W	Undefined
22	BEV	Controls the location of exception vectors: 0: Normal 1: Bootstrap	R/W	1
21	TS	Indicates that the TLB has detected a match on multiple entries. When such a detection occurs, the processor initiates a machine check exception and sets this bit. It is implementation dependent whether this condition can be corrected by software. If the condition can be corrected, this bit should be cleared by software before resuming normal operation. Software should not write a 1 to this bit when its value is a 0, thereby causing a 0-to-1 transition. If such a transition is caused by software, it is UNPREDICTABLE whether hardware ignores the write, accepts the write with no side effects, or accepts the write and initiates a machine check exception.	R/W	0

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

Bits	Name	Description	R/W	Reset
20	SR	Indicates that the entry through the reset exception vector was due to a Soft Reset: 0: Not Soft Reset (NMI or RESET) 1: Soft Reset Software should not write a 1 to this bit when its value is a 0, thereby causing a 0-to-1 transition. If such a transition is caused by software, it is UNPREDICTABLE whether hardware ignores or accepts the write.	R/W	1 for Soft Reset 0 otherwise
19	NMI	Indicates that the entry through the reset exception vector was due to an NMI: 0: Not NMI (Soft Reset or RESET) 1: NMI Software should not write a 1 to this bit when its value is a 0, thereby causing a 0-to-1 transition. If such a transition is caused by software, it is UNPREDICTABLE whether hardware ignores or accepts the write.	R/W	1 for NMI 0 otherwise
18:17	Reserved	Reserved = 0		0 0
16	CRY	Carry bit for the DADDWC instruction	R/W	Undefined
15:8	IM	Interrupt Mask. Controls the enabling of each of the external, internal and software interrupts. Note, IM[7:0] is identical to EIMR[7:0]. 0: Interrupt request disabled 1: Interrupt request enabled	R/W	Undefined
7	KX	Enables Access to 64-bit Kernel Segments and use of the XTLB Refill Vector for references to Kernel Segments: 0: Access to 64-bit Kernel Segments is disabled; TLB Refill Vector is used for references to Kernel Segments 1: Access to 64-bit Kernel Segments is enabled; XTLB Refill Vector is used for references to Kernel Segments	R/W	Undefined
6	SX	If Supervisor Mode is implemented, this bit enables access to 64-bit Supervisor Segments and use of the XTLB Refill Vector for references to Supervisor Segments: 0: Access to 64-bit Supervisor Segments is disabled; TLB Refill Vector is used for references to Supervisor Segments 1: Access to 64-bit Supervisor Segments is enabled XTLB Refill Vector is used for references to Supervisor Segments	R/W	Undefined
5	UX	Enables Access to 64-bit User Segments, use of the XTLB Refill Vector for references to User Segments, and execution of instructions which perform 64-bit operations while the processor is operating in User Mode: 0: Access to 64-bit User Segments is disabled TLB Refill Vector is used for references to User Segments; execution of instructions which perform 64-bit operations is disallowed while the processor is running in User Mode 1: Access to 64-bit User Segments is enabled XTLB Refill Vector is used for references to User Segments; execution of instructions which perform 64-bit operations is allowed while the processor is running in User Mode	R/W	Undefined
4:3	KSU	The encoding of this field denotes the base operating mode of the processor. 00: Base mode is Kernel Mode 01: Base mode is Supervisor Mode 10: Base mode is User Mode 11: Reserved. The operation of the processor is UNDEFINED if this value is written to the KSU field Note: This field overlaps the UM and R0 fields, described in the text following this table.	R/W	Undefined

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

Bits	Name	Description	R/W	Reset
2	ERL	<p>Error Level Set by the processor when a RESET, Soft Reset, NMI or Cache Error exception are taken.</p> <ul style="list-style-type: none"> 0: Normal level 1: Exception level <p>When ERL = 1:</p> <ul style="list-style-type: none"> • The processor is running in kernel mode • Interrupts are disabled • The ERET instruction will use the return address held in ErrorEPC instead of EPC • The lower 2^{29} bytes of kuseg are treated as an unmapped and uncached region. This allows main memory to be accessed in the presence of cache errors. • The operation of the processor is UNDEFINED if the ERL = 1 while the processor is executing instructions from kuseg. 	R/W	1
1	EXL	<p>Exception Level Set by the processor when any exception other than Reset, Soft Reset, NMI or Cache Error exception are taken.</p> <ul style="list-style-type: none"> 0: Normal Level 1: Exception Level <p>When EXL = 1:</p> <ul style="list-style-type: none"> • The processor is running in Kernel Mode • Interrupts are disabled • TLB/XTLB Refill exceptions use the general exception vector instead of the TLB/XTLB Refill vectors • EPC and Cause_{BD} will not be updated if another exception is taken 	R/W	Undefined
0	IE	<p>Interrupt Enable: Acts as the master enable for software and hardware interrupts:</p> <ul style="list-style-type: none"> 0: Interrupts are disabled 1: Interrupts are enabled 	R/W	Undefined

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.3.15 Cause**COP0 Register: 13**

This register primarily describes the cause of the most recent exception.

31	30	29	28	27	24	23	22	21	16
BD	RSVD	CE	Reserved			IV	WP	Reserved	
15			10	9	8	7	6	2	1
			IP[7:2]		IP[1:0]	Reserved	ExcCode	Reserved	

Bits	Name	Description	R/W	Reset
31	BD	Indicates whether the last exception taken occurred in a branch delay slot: 0: Last exception taken is not in delay slot 1: Last exception taken is in delay slot The processor updates BD only if Status_EXL = 0 when the exception occurred.	R	Undefined
30	Reserved	Reserved	0	0
29:28	CE	Coprocessor unit number referenced when a Coprocessor Unusable exception is taken. This field is loaded by hardware on every exception, but is UNPREDICTABLE for all exceptions except for Coprocessor Unusable.	R	Undefined
27:24	Reserved	Reserved	0	0
23	IV	Indicates whether an interrupt exception uses the general exception vector or a special interrupt vector: 0: Use the general exception vector (0x180) 1: Use the special interrupt vector (0x200)	R/W	Undefined
22	WP	Indicates that a watch exception was deferred because Status_EXL or Status_ERL were 1 at the time the watch exception was detected. This bit both indicates that the watch exception was deferred, and causes the exception to be initiated once Status_EXL and Status_ERL are both 0. As such, software must clear this bit as part of the watch exception handler to prevent a watch exception loop. Software should not write a 1 to this bit when its value is a 0, thereby causing a 0-to-1 transition. If such a transition is caused by software, it is UNPREDICTABLE whether hardware ignores the write, accepts the write with no side effects, or accepts the write and initiates a watch exception once Status_EXL and Status_ERL are both zero.	R/W	Undefined
21:16	Reserved	Reserved	0	0
15:10	IP[7:2]	Indicates an external interrupt is pending when the bit is 1: Cause[15], IP[7] = Hardware Interrupt 5, timer or performance counter interrupt Cause[14], IP[6] = Hardware Interrupt 4 Cause[13], IP[5] = Hardware Interrupt 3 Cause[12], IP[4] = Hardware Interrupt 2 Cause[11], IP[3] = Hardware Interrupt 1 Cause[10], IP[2] = Hardware Interrupt 0 Note: this field is identical to EIRR[7:2].	R	Undefined

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

Bits	Name	Description				R/W	Reset
9:8	IP[1:0]	Controls the request for software interrupts: Cause[9], IP[1] = Software Interrupt 1 request Cause[8], IP[0] = Software Interrupt 0 request Note: this field is identical to EIRR[1:0].				R/W	Undefined
7	Reserved	Reserved				0	0
6:2	ExcCode	Exception Code:				R	Undefined
		Num	Hex	Mnemonic	Description		
		0	0x00	Int	Interrupt		
		1	0x01	Mod	TLB modification exception		
		2	0x02	TLBL	TLB Exceptions (load or instruction fetch)		
		3	0x03	TLBS	TLB Exception (store)		
		4	0x04	AdEL	Address error exception (load or instruction fetch)		
		5	0x05	AdES	Address error exception (store)		
		6	0x06	IBE	Bus error exception (instruction fetch)		
		7	0x07	DBE	Bus error exception (data reference: load or store)		
		8	0x08	Sys	Syscall exception		
		9	0x09	Bp	Breakpoint exception		
		10	0x0A	RI	Reserved instruction exception		
		11	0x0B	CpU	Coprocessor Unusable exception		
		12	0x0C	Ov	Arithmetic Overflow exception		
		13	0x0D	Tr	Trap exception		
		14	0x0E	–	Reserved		
		15	0x0F	FPE	Floating Point Exception		
		16-17	0x10 0x11	–	Available for independent use		
		18	0x12	C2E	Reserved for Coprocessor 2 exceptions		
		19-21	0x13 0x15	–	Reserved		
		22	0x16	MDMX	MDMX Unusable Exception.		
		23	0x17	WATCH	Reference to WatchHi/WatchLo address		
		24	0x18	MCheck	Machine Check		
		25-29	0x19 0x1D	–	Reserved		
		30	0x1E	CacheErr	Cache error. In normal mode, a cache error exception has a dedicated vector and the Cause register is not updated. If EJTAG is implemented and a cache error occurs while in Debug Mode, this code is used to indicate that re-entry to Debug Mode was caused by a cache error.		
		31	0x1F	–	Reserved		
1:0	Reserved	Reserved				0	0

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.3.16 Exception Program Counter**COP0 Register: 14**

This register contains the address at which processing resumes after an exception has been serviced. All bits of the EPC register are significant and writable.

63		32
EPC[63:32]		
31		0
EPC[31:0]		

Bits	Name	Description	R/W	Reset
63:0	EPC	Exception Program Counter	R/W	Undefined

- For synchronous (precise) exceptions, EPC contains either:
 - a. The virtual address of the instruction that was the direct cause of the exception, or
 - b. The virtual address of the immediately preceding branch or jump instruction, when the exception causing instruction is in a branch delay slot, and the Branch Delay bit in the Cause register is set.
- For asynchronous (imprecise) exceptions, EPC contains the address of the instruction at which to resume execution.
- The processor does not write to the EPC register when Status_EXL = 1

4.2.3.17 ProcessorID**COP0 Register: 15**

This register contains information identifying NetLogic Corporation, NetLogic options, processor identification and revision level of the processor.

31	24	23	16	15	8	7	0
Reserved			CompanyID		ProcessorID		Revision

Bits	Name	Description	R/W	Reset
31:24	NetLogic-Specific	The value in this field is specified by NetLogic Corporation.	R	Preset
23:16	CompanyID (NetLogic ID)	Identifies the device as a product of NetLogic Corporation. Software can distinguish a MIPS32 or MIPS64 processor from one implementing an earlier MIPS ISA by checking this field for zero. If it is non-zero the processor implements the MIPS32 or MIPS64 Architecture. This NetLogic ID is assigned by MIPS Technologies when a MIPS32 or MIPS64 license is acquired. The encoding of this field is: 0xC: NetLogic Corporation.	R	Preset to 0xC

Bits	Name	Description	R/W	Reset																								
15:8	ProcessorID	<p>Type of NetLogic MIPS-based Processor</p> <p>This field allows software to distinguish between various NetLogic MIPS-based processor implementations, and is qualified by the CompanyID field described above. The combination of the CompanyID and ProcessorID fields creates a unique number assigned to each processor implementation.</p> <p>The encoding of this field is:</p> <table border="1"> <thead> <tr> <th>Device</th><th>Processor ID</th></tr> </thead> <tbody> <tr><td>XLS616</td><td>0x40</td></tr> <tr><td>XLS608</td><td>0x4A</td></tr> <tr><td>XLS416</td><td>0x44</td></tr> <tr><td>XLS408-Lite</td><td>0x88</td></tr> <tr><td>XLS408</td><td>0x4E</td></tr> <tr><td>XLS404-Lite</td><td>0x8C</td></tr> <tr><td>XLS404</td><td>0x4F</td></tr> <tr><td>XLS208</td><td>0x8E</td></tr> <tr><td>XLS204</td><td>0x8F</td></tr> <tr><td>XLS108</td><td>0xCE</td></tr> <tr><td>XLS104</td><td>0xCF</td></tr> </tbody> </table>	Device	Processor ID	XLS616	0x40	XLS608	0x4A	XLS416	0x44	XLS408-Lite	0x88	XLS408	0x4E	XLS404-Lite	0x8C	XLS404	0x4F	XLS208	0x8E	XLS204	0x8F	XLS108	0xCE	XLS104	0xCF	R	-
Device	Processor ID																											
XLS616	0x40																											
XLS608	0x4A																											
XLS416	0x44																											
XLS408-Lite	0x88																											
XLS408	0x4E																											
XLS404-Lite	0x8C																											
XLS404	0x4F																											
XLS208	0x8E																											
XLS204	0x8F																											
XLS108	0xCE																											
XLS104	0xCF																											
7:0	Revision	<p>Specifies the revision number of the processor. This field allows software to distinguish between one revision and another of the same processor type. The values below indicate production silicon. Note that additional values may be read for pre-production silicon.</p> <table border="1"> <thead> <tr> <th>Stepping</th><th>Silicon Revision ID</th></tr> </thead> <tbody> <tr><td>A1</td><td>0x01</td></tr> <tr><td>B0</td><td>0x02</td></tr> <tr><td>B1</td><td>0x03</td></tr> </tbody> </table>	Stepping	Silicon Revision ID	A1	0x01	B0	0x02	B1	0x03	R	-																
Stepping	Silicon Revision ID																											
A1	0x01																											
B0	0x02																											
B1	0x03																											

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.3.18 EBase**COP0 Register: 15, Select 1**

Exception Base.

This is a Release 2 Register, supported by the XLS processor.

It is a 32-bit register which contains the Exception Base and identifies the CpuNum, the physical and virtual address.

31	30	29	12	11	10	9	5	4	2	1	0
Reserved		EBase		Reserved		CpuNum (0)		CpuNum		CpuNum	

Bits	Name	Description	R/W	Reset
31:30	Reserved	Reserved	R	2
29:12	EBase		R/W	0
11:10	Reserved	Reserved	R	0
9:5	CpuNum	Always 0	R	0
4:2	CpuNum	Physical CpuID	R	0 - 7
0:1	CpuNum	ThreadId / Virtual CpuID	R	0 - 3

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.3.19 Config0**COP0 Register: 16, Select 0**

Config0 specifies various configurations and capabilities.

Reset Value: 0x8000c083 (binary: 1000 0000 0000 0000 1100 0000 1000 0011)

All fields which are designated as R are preset and cannot be changed. The R/W fields (K0 only), can be modified.

31	30												16
M	IMPL												
15	14	13	12	10	9	7	6	4	3	2			0
BE	AT	AR		MT	Reserved		VI	K0					

Bits	Name	Description	R/W	Reset
31	M	Denotes that the Config1 register is implemented at a select field value of 1.	R	1
30:16	IMPL	Implementation-dependent	R	0
15	BE	Defines Endian mode presently running, Preset or Externally set 0: Little Endian 1: Big Endian	R	1
14:13	AT	Architecture Type definition: 00: MIPS32 01: MIPS64 with access to 32-bit compatibility segments 10: MIPS64 with access to all address segments 11: Reserved	R	2
12:10	AR	Architecture revision level 000: Revision 1 001-111: Reserved	R	0
9:7	MT	MMU Type: 000: None 001: Standard TLB 010: Standard BAT 011: Standard fixed mapping 100-111: Reserved	R	1
6:4	Reserved	Reserved	R	0
3	VI	Virtual Instruction cache (using both virtual indexing and virtual tags): 0: Instruction Cache is not virtual 1: Instruction Cache is virtual		0
2:0	K0	Kseg0 coherency algorithm: 000: Unimplemented — will cause Address Error exception 001: Unimplemented — will cause Address Error exception 010: Uncached 011: Cachable 100: Cachable 101: Cachable 110: Cachable 111: Unimplemented — will cause Address Error exception	R/W	3

- Most of the fields in the Config register are initialized by hardware during Reset, or are

constant. One field, K0, must be initialized by software in the reset handler.
 Portions of the preceding information reproduced under license from the MIPS® Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.3.20 Config1**COP0 Register: 16, Select 1**

Config1 is an adjunct to Config0 that encodes additional capabilities.

All fields are read-only.

Reset Value: 0x7e6733da (binary: 0111 1110 0110 0111 0011 0011 1101 1010)

31	30	25	24	22	21	19	18	16				
M	MMU Size – 1			IS		IL		IA				
15	13	12	10	9	7	6	5	4	3	2	1	0
DS	DL	DA	C2	MD	PC	WR	CA	EP	FP			

Table 4-4. Config1 Register

Bits	Name	Description	R/W	Reset
31	M	This bit is reserved; it indicates that a Config2 register is not present.	R	0
30:25	MMU Size–1	Number of entries in the TLB minus one. The values 0 through 63 in this field correspond to 1 to 64 TLB entries. The value zero is implied by Config_MT having a value of ‘none’.	R	See Note 1
24:22	IS	Icache sets per way: 000: 64 001: 128 010: 256 011: 512 100: 1024 101: 2048 110: 4096 111: Reserved	R	1
21:19	IL	Icache line size: 000: No Icache present 001: 4 bytes 010: 8 bytes 011: 16 bytes 100: 32 bytes 101: 64 bytes 110: 128 bytes 111: Reserved	R	0x4
18:16	IA	Icache associativity: 000: Direct mapped 001: 2-way 010: 3-way 011: 4-way 100: 5-way 101: 6-way 110: 7-way 111: 8-way	R	0x7

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

Table 4-4. Config1 Register (*continued*)

Bits	Name	Description	R/W	Reset
15:13	DS	Dcache sets per way: 000: 64 001: 128 010: 256 011: 512 100: 1024 101: 2048 110: 4096 111: Reserved	R	1
12:10	DL	Dcache line size: 000: No Dcache present 001: 4 bytes 010: 8 bytes 011: 16 bytes 100: 32 bytes 101: 64 bytes 110: 128 bytes 111: Reserved	R	0x4
9:7	DA	Dcache associativity: 000: Direct mapped 001: 2-way 010: 3-way 011: 4-way 100: 5-way 101: 6-way 110: 7-way 111: 8-way	R	0x7
6	C2	Coprocessor 2 implemented: 0: No Coprocessor 2 implemented 1: Coprocessor 2 is implemented This bit indicates not only that the processor contains support for Coprocessor 2, but that such a coprocessor is attached.	R	1
5	MD	MDMX AASE implemented: 0: No MDMX AASE implemented 1: MDMX AASE is implemented This bit indicates not only that the processor contains support for MDMX, but that such a processing element is attached.	R	0
4	PC	Performance Counter registers implemented: 0: No Performance Counter registers implemented 1: Performance Counter registers is implemented	R	1
3	WR	Watch registers implemented: 0: No Watch registers implemented 1: Watch registers is implemented	R	1
2	CA	Code compression (MIPS16) implemented: 0: No MIPS16 implemented 1: MIPS16 is implemented	R	0

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

Table 4-4. Config1 Register (*continued*)

Bits	Name	Description	R/W	Reset
1	EP	EJTAG implemented: 0: No EJTAG implemented 1: EJTAG is implemented	R	1
0	FP	Floating Point Unit (FPU) implemented: 0: No FPU implemented 1: FPU is implemented	R	0

1. The number of TLB are really determined at Boot time based on how many threads are active. There are a total of 64 entries, and they will be equally divided among the active threads.
2. The Icache and Dcache configuration parameters include encodings for the number of sets per way, the line size, and the associativity. The total cache size for a cache is therefore:

$$\text{Cache Size} = \text{Associativity} * \text{Line Size} * \text{Sets Per Way}$$
3. If the line size is zero, there is no cache implemented.

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.4.21 Config2**COP0 Register: 16, Select 2**

Encodes Level 2 and Level 3 cache configurations.

Note: Not implemented.

31	30	16		
M	TRS[30:16]			
15		0		
	TRS[15:0]			
Bits	Name	Description	R/W	Reset
31	M	Not implemented.	R	0
30:0	TRS	Not implemented.	R	0

4.2.4.22 Config3**COP0 Register: 16, Select 3**

Encodes additional capabilities.

Note: Not implemented.

31	30	16		
M	Reserved			
15		2 1 0		
	Reserved	SM TL		
Bits	Name	Description	R/W	Reset
31	M	Not implemented.	R	0
30:2	—	Not implemented	R	0
1	SM	Not implemented.	R	0
0	TL	Not implemented	R	0

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.4.23 Config7**COP0 Register: 16, Select 7**

Processor Identification.

This is a 32-bit read-only register which identifies the Processor and Thread Id. This same information is duplicated in the lower bits of the EBASE (COP0 15, Select 1) register.

31:7		6:4	3:2	1:0		
<i>Reserved</i>		CpuNum	<i>Reserved</i>	CpuNum		
Bits	Name	Description			R/W	Reset
31:7	<i>Reserved</i>	<i>Reserved</i>			R	0
6:4	CpuNum	Physical CpuID			R	0
3:2	<i>Reserved</i>	<i>Reserved</i>			R	0
1:0	CpuNum	ThreadID / Virtual CpuID			R	0

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.4.24 WatchLo**COP0 Register: 18, Select 0**

The WatchLo and WatchHi registers together provide the interface to a watchpoint debug facility which initiates a watch exception if an instruction or data access matches the address specified in the registers. As such, they duplicate some functions of the EJTAG debug solution (see [Section 25.2, “EJTAG and Debug Testing”](#)). Watch exceptions are only taken if the EXL and ERL bits are zero in the Status register. If either bit is a one, the WP bit is set in the Cause register, and the watch exception is deferred until both the EXL and ERL bits are zero.

This implementation provides one pair of WatchLo and WatchHi registers. Software may determine that WatchLo and WatchHi registers are implemented via the WR bit of the Config1 register. See the discussion of the M bit in the [WatchHi](#) register description.

The WatchLo register specifies the base virtual address and the type of reference (instruction fetch, load, store) to match. All three reference types are supported.¹

Data watch exceptions will be taken by the XLS Processor for PREFETCH instructions, as long as PREFETCH instructions are not disabled in the LSU configuration register. Data watch exceptions will be taken for the fetch and lock CACHE instructions, but not for other types.

63	3	2	1	0
VAddr	I	R	W	

Bits	Name	Description	R/W	Reset
63:3	VAddr	This field specifies the virtual address to match. Note that this is a doubleword address, since bits [2:0] are used to control the type of match, i.e., this is VA[63:3].	R/W	Undefined
2	I	Enable instruction fetch watch exceptions when IEU_DEFFEATURE [DBE] is 0 ^a 0: Watch exceptions are disabled for instruction fetches 1: Watch exceptions are enabled for instruction fetches that match the address	R/W	0
1	R	Enable load watch exceptions when IEU_DEFFEATURE [DBE] is 0 ¹ 0: Watch exceptions are disabled for loads 1: Watch exceptions are enabled for loads that match the address	R/W	0
0	W	Enable store watch exceptions when IEU_DEFFEATURE [DBE] is 0 ¹ 0: Watch exceptions are disabled for stores 1: Watch exceptions are enabled for stores that match the address	R/W	0

- a. The DBE bit in the [IEU_DEFFEATURE](#) register is set by the bootloader to 1 which causes the exception to be delivered to the JTAG vector instead of the Watchpoint. The [IEU_DEFFEATURE](#)[DBE] must be set to 0 to enable the Watchpoint debug facility.

Portions of the preceding information are reproduced under license from the MIPS™ Architecture for Programmers, Volume I, Copyright © MIPS Technologies, Inc. All rights reserved. ^aactually set.

4.2.4.25 WatchHi

COP0 Register: 19, Select 0

The WatchLo and WatchHi registers together provide the interface to a watchpoint debug facility which initiates a watch exception if an instruction or data access matches the address specified in the registers. As such, they duplicate some functions of the EJTAG debug solution (see [Section 25.2, “EJTAG and Debug Testing”](#)). Watch exceptions are only taken if the EXL and ERL bits are zero in the Status register. If either bit is a one, the WP bit is set in the Cause register, and the watch exception is deferred until both the EXL and ERL bits are zero.

The implementation provides one pair of WatchLo and WatchHi registers, referencing them via the select field of the MTC0/MFC0 and DMTC0/DMFC0 instruction. Software may determine that WatchLo and WatchHi registers are implemented via the WR bit of the Config1 register.

The WatchHi register contains information that qualifies the virtual address specified in the WatchLo register: an ASID, a G(lobal) bit, and an optional address mask. If the G bit is one, any virtual address reference that matches the specified address will cause a watch exception. If the G bit is zero, only those virtual address references for which the ASID value in the WatchHi register matches the ASID value in the EntryHi register cause a watch exception. The optional mask field provides address masking to qualify the address specified in WatchLo.

31	30	29	24	23	16	15	12	11	3	2	0
M	G	Reserved		ASID	Reserved	Mask			Reserved		

Bits	Name	Description	R/W	Reset
31	M	Another pair of WatchHi/WatchLo registers is implemented at a MTC0 or MFC0 select field value of $n+1$.	R	0
30	G	1: If this bit is set, any address that matches that specified in the WatchLo register will cause a watch exception. 0: If this bit is zero, the ASID field of the WatchHi register must match the ASID field of the EntryHi register to cause a watch exception.	R/W	Undefined
29:24	Reserved	Reserved	0	0
23:16	ASID	ASID value which is required to match that in the EntryHi register if the G bit is zero in the WatchHi register.	R/W	Undefined
15:12	Reserved	Reserved	0	0
11:3	Mask	Bit mask that qualifies the address in the WatchLo register. Any bit in this field that is set inhibits the corresponding address bit from participating in the address match. Software may determine how many mask bits are implemented by writing ‘1s’ to this field and then reading back the result.	R/W	Undefined
2:0	Reserved	Reserved	0	0

4.2.4.26 XContext**COP0 Register: 20**

The XContext register is a read/write register containing a pointer to an entry in the page table entry (PTE) array. This array is an operating system data structure that stores virtual-to-physical translations. During a TLB miss, the operating system loads the TLB with the missing translation from the PTE array. The XContext register is primarily intended for use with the XTLB Refill handler, but is also loaded by hardware on a TLB Refill. However, it is unlikely to be useful to software in the TLB Refill Handler. The XContext register duplicates some of the information provided in the BadVAddr register, but is organized in such a way that the operating system can directly reference a 16-byte structure in memory that describes the mapping.

A TLB exception (TLB Refill, XTLB Refill, TLB Invalid, or TLB Modified) causes bits [63:62] of the virtual address to be written into the R field and bits [61:13] of the virtual address to be written into the BadVPN2 field of the XContext register. The PTEBase field is written and used by the operating system.

The BadVPN2 field if the XContext register is not defined after an address error exception and this field may be modified by hardware during the address error exception sequence.

63	55	54	53	52	32
PTEBase	R			BadVPN2[48:28]	
31				4 3 0	
			BadVPN2[27:0]		Reserved

Bits	Name	Description	R/W	Reset
63:55	PTEBase	This field is for use by the operating system and is normally written with a value that allows the operating system to use the XContext Register as a pointer into the current PTE array in memory.	R/W	Undefined
54:53	R	The Region field contains bits 63:62 of the virtual address. 00: xuseg 01: xsseg: supervisor address region. 10: Reserved 11: xkseg	R	Undefined
52:4	BadVPN2	The Bad Virtual Page Number/2 field is written by hardware on a miss. It contains bits 61:13 of the virtual address that missed.	R	Undefined
3:0	Reserved	Reserved	0	0

A TLB exception (TLB Refill, XTLB Refill, TLB Invalid, or TLB Modified) causes bits [63:62] of the virtual address to be written into the R field and bits [61:13] of the virtual address to be written into the BadVPN2 field of the XContext register. The PTEBase field is written and used by the operating system.

The BadVPN2 field of the XContext register is not defined after an address error exception and this field may be modified by hardware during the address error exception sequence.

4.2.4.27 OS Scratch0 – Scratch7**COP0 Register: 22, Select 0-7**

OS Scratch Registers

These are eight (8) 64-bit registers which can be used as temporary storage.

63	0
OS_Scratch0	

Bits	Name	Description	R/W	Reset
63:0	OS_Scratch0	OS Scratch pad	R/W	Undefined

4.2.4.28 Debug**COP0 Register: 23**

Part of EJTAG specification.

See [Section 25.2, “EJTAG and Debug Testing”](#) for information on the use of this register.

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBD	DM	No DCR	LSNM	Doze	Halt	Count DM	IBus EP	MCheckP	CacheE	DBus EP	IEXI	DDB Slmpr	DDB Llmp	EJTAG VER[2:1]	
15	14			10	9	8	7		6	5	4	3	2	1	0
EJTAG VER[0]				DExcCode		NoSST	SSt		Reserved	DINT	DIB	DDBS	DDBL	DBp	DSS

Bits	Name	Description	R/W	Reset
31	DBD	Indicates whether the last debug exception or exception in Debug Mode occurred in a branch or jump delay slot: 0: Not in delay slot 1: In delay slot	R	Undefined
30	DM	Indicates that the processor is operating in Debug Mode: 0: Processor is operating in Non-Debug Mode 1: Processor is operating in Debug Mode	R	0
29	NoDCR	Indicates whether the dseg memory segment is present: 0: dseg is present 1: No dseg present	R	Preset 0
28	LSNM	Controls access of loads/stores between dseg and remaining memory when dseg is present: 0: Loads/stores in dseg address range go to dseg 1: Loads/stores in dseg address range go to system memory See bit 29, NoDCR.	R/W	0
27	Doze	Indicates that the processor was in a low-power mode when a debug exception occurred: 0: Processor not in low-power mode when debug exception occurred 1: Processor in low-power mode when debug exception occurred The Doze bit indicates Reduced Power (RP) and WAIT, and other implementation-dependent low-power modes.	R	0
26	Halt	Indicates that the internal processor system bus clock was stopped when the debug exception occurred: 0: Internal system bus clock running 1: Internal system bus clock stopped Halt indicates WAIT, and other implementation-dependent events that stop the system bus clock.	R	0
25	CountDM	Controls or indicates the Count register behavior in Debug Mode. Implementations can have fixed behavior, in which case this bit is read-only (R), or the implementation can allow this bit to control the behavior, in which case this bit is read/write (R/W). The reset value of this bit indicates the behavior after reset, and depends on the implementation. 0: Count register stopped in Debug Mode 1: Count register is running in Debug Mode	R/W	0

Portions of the preceding information reproduced under license from the MIPS EJTAG Specification, Doc. No. MD00047 rev 3.20. Copyright © MIPS Technologies Inc. All rights reserved

Bits	Name	Description	R/W	Reset
24	IBusEP	Not implemented. Indicates if a Bus Error exception is pending from an instruction fetch. Set when an instruction fetch bus error event occurs or a 1 is written to the bit by software. Cleared when a Bus Error exception on an instruction fetch is taken by the processor. If IBusEP is set when IEXI is cleared, a Bus Error exception on an instruction fetch is taken by the processor, and IBusEP is cleared. In Debug Mode, a Bus Error exception applies to a Debug Mode Bus Error exception.	R/W1	0
23	MCheckP	Not implemented. Indicates if a Machine Check exception is pending. Set when a machine check event occurs or a 1 is written to the bit by software. Cleared when a Machine Check exception is taken by the processor. If MCheckP is set when IEXI is cleared, a Machine Check exception is taken by the processor, and MCheckP is cleared. In Debug Mode, a Machine Check exception applies to a Debug Mode Machine Check exception.	R/W1	0
22	CacheE	Not implemented. Indicates if a Cache Error is pending. Set when a cache error event occurs or a 1 is written to the bit by software. Cleared when a Cache Error exception is taken by the processor. If CacheEP is set when IEXI is cleared, a Cache Error exception is taken by the processor, and CacheEP is cleared. In Debug Mode, a Cache Error exception applies to a Debug Mode Cache Error exception.	R/W1	0
21	DBusEP	Not implemented. Indicates if a Data Access Bus Error exception is pending. Set when a data access bus error event occurs or a 1 is written to the bit by software. Cleared when a Bus Error exception on data access is taken by the processor. If DBusEP is set when IEXI is cleared, a Bus Error exception on data access is taken by the processor, and DBusEP is cleared. In Debug Mode, a Bus Error exception applies to a Debug Mode Bus Error exception.	R/W1	0
20	IEXI	Not implemented. An Imprecise Error eXception Inhibit controls exceptions taken due to imprecise error indications. Set when the processor takes a debug exception or an exception in Debug Mode occurs. Cleared by execution of the DERET instruction. Otherwise modifiable by Debug Mode software. When IEXI is set, then the imprecise error exceptions from bus errors on instruction fetches or data accesses, cache errors, or machine checks are inhibited and deferred until the bit is cleared.	R/W	0
19	DDBSImp	Not implemented. Indicates that a Debug Data Break Store Imprecise exception due to a store was the cause of the debug exception, or that an imprecise data hardware break due to a store was indicated after another debug exception occurred. Cleared on exception in Debug Mode. 0: No match of an imprecise data hardware breakpoint on store 1: Match of imprecise data hardware breakpoint on store	R	0
18	DDBLImp	Not implemented. Indicates that a Debug Data Break Load Imprecise exception due to a load was the cause of the debug exception, or that an imprecise data hardware break due to a load was indicated after another debug exception occurred. Cleared on exception in Debug Mode. 0: No match of an imprecise data hardware breakpoint on load 1: Match of imprecise data hardware breakpoint on load	R	0

Portions of the preceding information are reproduced under license from the MIPS® JTAG Specification, Doc. No. MD00047 rev 3.20. Copyright © MIPS Technologies Inc. All rights reserved.

Bits	Name	Description	R/W	Reset
17:15	EJTAG VER	EJTAG Version	R	2
14:10	DExcCode	Indicates the cause of the latest exception in Debug Mode. The field is encoded as the ExcCode field in the Cause register for those exceptions that can occur in Debug Mode (the encoding is shown in MIPS32 and MIPS64 specifications), with addition of code 30 with the mnemonic CacheErr for cache errors. This value is undefined after a debug exception.	R	Undefined
9	NoSST	Single Step Feature Available	R	0
8	SSt	Controls whether single-step feature is enabled: 0: No enable of single-step feature 1: Single-step feature enabled	R/W	0
7:6	Reserved	Reserved	0	0
5	DINT	Indicates that a Debug Interrupt exception occurred. Cleared on exception in Debug Mode. 0: No Debug Interrupt exception 1: Debug Interrupt exception	R	Undefined
4	DIB	Indicates that a Debug Instruction Break exception occurred. Cleared on exception in Debug Mode. 0: No Debug Instruction Break exception 1: Debug Instruction Break exception	R	Undefined
3	DDBS	Indicates that a Debug Data Break Store exception occurred on a store due to a precise data hardware break. Cleared on exception in Debug Mode. 0: No Debug Data Break Store Exception 1: Debug Data Break Store Exception	R	Undefined
2	DDBL	Indicates that a Debug Data Break Load exception occurred on a load due to a precise data hardware break. Cleared on exception in Debug Mode. 0: No Debug Data Break Store Exception 1: Debug Data Break Store Exception	R	Undefined
1	DBp	Indicates that a Debug Breakpoint exception occurred. Cleared on exception in Debug Mode. 0: No Debug Breakpoint exception 1: Debug Breakpoint exception	R	Undefined
0	DSS	Indicates that a Debug Single Step exception occurred. Cleared on exception in Debug Mode. 0: No debug single-step exception 1: Debug single-step exception	R	Undefined

The Debug register contains the cause of the most recent debug exception and exception in Debug Mode. It also controls single stepping. This register indicates low-power and clock states on debug exceptions, debug resources, and other internal states.

Only the DM bit and the EJTAGver field are valid when read from the Debug register in Non-Debug Mode; the value of all other bits and fields is UNPREDICTABLE.

The following bits and fields are only updated on debug exceptions and/or exceptions in Debug Mode:

- DSS, DBp, DDBL, DDBS, DIB, DINT, DDBLImpr, and DDBSImpr are updated on both debug exceptions and on exceptions in Debug Modes
- DExcCode is updated on exceptions in Debug Mode, and is undefined after a debug exception

Portions of the preceding information reproduced under license from the MIPS EJTAG Specification, Doc. No. MD00047 rev 3.20. Copyright © MIPS Technologies Inc. All rights reserved

- Halt and Doze are updated on a debug exception, and are undefined after an exception in Debug Mode
- DBD is updated on both debug and on exceptions in Debug Modes

The SYNC instruction, followed by appropriate spacing, must be executed to ensure that the DDBLImpr, DDBSImpr, IBusEP, DBusEP, CacheEP, and MCheckP bits are fully updated. This instruction sequence must be used both in the beginning of the debug handler before pending imprecise errors are detected from Non-Debug Mode, and at the end of the debug handler before pending imprecise errors are detected from Debug Mode. The IEXI bit controls enable/disable of imprecise error exceptions.

**NETLOGIC
CONFIDENTIAL**

Portions of the preceding information reproduced under license from the MIPS EJTAG Specification, Doc. No. MD00047 rev 3.20. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.4.29 DEPC**COP0 Register: 24**

Debug Exception Program Counter, part of EJTAG spec.

See [Section 25.2, “EJTAG and Debug Testing”](#) for information on the use of this register.

63	0
DEPC	

Bits	Name	Description	R/W	Reset
63:0	DEPC	Debug Exception Program Counter	R/W	Undefined

In processors that implement the MIPS16 ASE, a read of the DEPC register (via MFC0 or DMFC0) returns the following value in the destination GPR:

```
GPR[rt] ← RestartPC63..1 || ISAMode
```

That is, the upper 63 bits of the restart PC are combined with the ISA Mode bit and written to the GPR.

Similarly, a write to the DEPC register (via MTC0 or DMTC0) takes the value from the GPR and distributes that value to the restart PC and the ISA Mode bit, as follows

```
RestartPC ← GPR[rt]63..1 || 0
ISAMode ← GPR[rt]0
```

That is, the upper 63 bits of the GPR are written to the upper 63 bits of the restart PC, and the lower bit of the restart PC is cleared. The ISA Mode bit is loaded from the lower bit of the GPR.

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.4.30 PerfCntrCtl0**COP0 Register: 25, Select 0**

Performance Counter Control Register 0

See [Section 25.6, “Performance Monitoring”](#) for information on the use of this register.

31	30		14	13	12	11	10	5	4	3	2	1	0
M		Reserved	G	TID	Event	IE	U	S	K	E			

Bits	Name	Description	R/W	Reset
31	M	Another pair of Performance Counter Control and Counter registers is implemented at MTC0 or MFC0 small select field value of $n+2$ and $n+3$.	R	1
30:14	Reserved	Reserved	R	0
13	G	If set, events from all Threads are counted.	R/W	0
12:11	TID	Identifies the Thread whose events should be counted.	R/W	0
10:5	Event	Selects the event to be counted by the corresponding Counter Register. The list of events is shown in Table 4-1 .	R/W	0
4	IE	Interrupt Enable: Enables the interrupt request when the corresponding counter overflows (when counter bit 31 goes to 1). Note that this bit simply enables the interrupt request. The actual interrupt is still gated by the normal interrupt masks and enable in the Status register. 0: Performance counter interrupt disabled 1: Performance counter interrupt enabled	R/W	0
3	U	Enables event counting in User Mode. 0: Disable event counting in User Mode 1: Enable event counting in User Mode	R/W	0
2	S	Enables event counting in Supervisor Mode. 0: Disable event counting in Supervisor Mode 1: Enable event counting in Supervisor Mode	R/W	0
1	K	Enables event counting in Kernel Mode. This bit enables event counting only when the EXL and ERL bits in the Status register are 0. 0: Disable event counting in Kernel Mode 1: Enable event counting in Kernel Mode	R/W	0
0	E	Enables event counting when the EXL bit in the Status register is 1 and the ERL bit in the Status register is 0. Counting is never enabled when the ERL bit in the Status register or the DM bit in the Debug register is 1. 0: Disable event counting while EXL = 1, ERL = 0 1: Enable event counting while EXL = 1, ERL = 0	R/W	0

Note: Unlike all other MIPS CP0 registers, the performance counter registers are NOT duplicated for each Thread, i.e., only one Thread may utilize these at a time.

Table 4-1. Performance Counter Event Types

#	Event Type
0	Instructions fetched
1	Instruction Cache misses
2	Instruction Cache Unit (ICU) Miss Queue Entry 0 allocated
3	ICU Miss Queue Entry 0 occupied
4	ICU Miss Queue merges
5	ICU Miss Queue full rejects
6	Instruction cache parity errors
7	Instruction fetch pipe bubbles requested by ICU
8	Instruction microTLB misses
9	Instruction or data microTLB miss requests rejected by MMU
10	A TLB instruction (TLBR, TLBWI, TLBWR, or TLBP) lost arbitration for processing in the MMU
11	Overlapping entries removed from main TLB
12	Count sleep cycles for each Thread
13	Active when an Instruction is issued out of the Replay Buffer
14	Active when a Valid instruction is sent out by the IFU
15	Active when a particular Thread needs to be sent back because there is no space in the Replay Buffer, i.e. it is Full.
16	Asserted when a branch instruction, which was predicted taken, is retired
17	Asserted when an instruction is retired
18	Asserted when any branch instruction is retired
19	Asserted when a taken branch instruction is retired
20	Asserted when a branch or a jump instruction retired
21	Asserted when a branch or jump instruction was mispredicted and required a flush
22	Asserted when a jump register instruction is retired
23	Asserted when a jump register instruction, which required a flush to correct a mispredicted target, is retired
24	Asserted when the PC/instruction which came down the pipe does not match what is architecturally expected. This can happen when the fetch unit sees a flush on a branch delay slot and is not able to track it correctly.
25	Asserted when an interrupt is issued
26	Asserted when a exception happens
27	Asserted each time a ReplayFlush happens in the Reg. Stage of the pipe
28	Asserted each time a ReplayFlush happened due to a LD/USE hazard
29	Asserted each time a ReplayFlush happened due to an ALU hazard
30	Asserted anytime a flush is generated in the WRB Stage
31	Prefetch instruction has retired
32	Store instruction has retired
33	Load instruction (including prefetches) has retired
34	A prefetch has hit the cache
35	A load has hit the cache
36	A Store has hit the cache

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright ©³⁷ MIPS Technologies Inc. All rights reserved.

Table 4-1. Performance Counter Event Types (continued)

#	Event Type
38	A memory reference (load or store) has hit the cache
39	A memory reference (load or store) has retired
40	The Read Miss Queue has issued an abort
41	The Write Queue has issued an abort
42	The Data cache controller has issued an abort
43	A Data Cache unit has issued a pipeline flush
44	Data microTLB misses
45	Victim
46	Data Cache front end stall
47	Snoop Upgrade failed
48	Cache to Cache transfer
49	Snoop hit
50	Snoop request
51	Miss queue entry allocation
52	Miss queue entry valid
53	Reserved - Never increments (no event)
54	Reserved - Never increments (no event)
55	Reserved - Never increments (no event)
56	Reserved - Never increments (no event)
57	Reserved - Never increments (no event)
58	Reserved - Never increments (no event)
59	Reserved - Never increments (no event)
60	Reserved - Never increments (no event)
61	Reserved - Never increments (no event)
62	Reserved - Never increments (no event)
63	Always increment (cycle count)

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.4.31 PerfCntr0**COP0 Register: 25, Select 1**

Performance Counter Register 0

See [Section 25.6, “Performance Monitoring”](#) for information on the use of this register.

31	0
EventCount	

Bits	Name	Description	R/W	Reset
31:0	EventCount	Increments once for each event that is enabled by the corresponding Control Register. When bit 31 = 1, an interrupt request is made if Control Register IE = 1. The interrupt is identified by the CAUSE Register, bit 15.	R/W	0

Note: The counter increments by a maximum of 1; this is the case even if multiple occurrences of the chosen event occur in the same cycle.

Note: Unlike all other MIPS CP0 registers, the performance counter registers are NOT duplicated for each nCPU, that is, only one nCPU may utilize these at a time.

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.4.32 PerfCntrCtl1**COP0 Register: 25 Select 2**

Performance Counter Control Register 1

See [Section 25.6, “Performance Monitoring”](#) for information on the use of this register.

31	30		14	13	12	11	10	5	4	3	2	1	0
M		Reserved	G	TID		Event	IE	U	S	K	E		

Bits	Name	Description	R/W	Reset
31	M	Another pair of Performance Counter Control and Counter registers is implemented at MTC0 or MFC0 small select field value of $n+2$ and $n+3$.	R	0
30:14	Reserved	Reserved	R	0
13	G	If set, events from all threads are counted.	R/W	0
12:11	TID	Identifies the thread whose events should be counted.	R/W	0
10:5	Event	Selects the event to be counted by the corresponding Counter Register. The list of events is shown in Table 4-1 .	R/W	0
4	IE	Interrupt Enable: Enables the interrupt request when the corresponding counter overflows (counter[31] = 1 ₂). Note that this bit simply enables the interrupt request. The actual interrupt is still gated by the normal interrupt masks and enable in the Status register. 0: Performance counter interrupt disabled 1: Performance counter interrupt enabled	R/W	0
3	U	Enables event counting in User Mode. 0: Disable event counting in User Mode 1: Enable event counting in User Mode	R/W	0
2	S	Enables event counting in Supervisor Mode. 0: Disable event counting in Supervisor Mode 1: Enable event counting in Supervisor Mode	R/W	0
1	K	Enables event counting in Kernel Mode. This bit enables event counting only when the EXL and ERL bits in the Status register are '0'. 0: Disable event counting in Kernel Mode 1: Enable event counting in Kernel Mode	R/W	0
0	E	Enables event counting when the EXL bit in the Status register is '1' and the ERL bit in the Status register is '0'. Counting is never enabled when the ERL bit in the Status register or the DM bit in the Debug register is '1'. 0: Disable event counting while EXL = '1', ERL = '0' 1: Enable event counting while EXL = '1', ERL = '0'	R/W	0

Note: Unlike all other MIPS CP0 registers, the performance counter registers are NOT duplicated for each nCPU, i.e., only one nCPU may utilize these at a time.

4.2.4.33 PerfCntr1**COP0 Register: 25 Select 3**

Performance Counter Register 1

See [Section 25.6, “Performance Monitoring”](#) for information on the use of this register.

31	0
EventCount	

Bits	Name	Description	R/W	Reset
31:0	EventCount	Increments once for each event that is enabled by the corresponding Control Register. When bit 31 = 1, an interrupt request is made if Control Register IE = 1.	R/W	0

Note: The counter increments by a maximum of 1; this is the case even if multiple occurrences of the chosen event occur in the same cycle.

Note: Unlike all other MIPS CP0 registers, the performance counter registers are NOT duplicated for each nCPU, i.e., only one nCPU may utilize these at a time.

4.2.4.34 ErrorCtl**COP0 Register: 26**

Note: Not implemented.

31	0
Reserved	

Bits	Name	Description	R/W	Reset
31:0	Reserved	Reserved = 0	R	0

4.2.4.35 CacheError**COP0 Register: 27**

Note: Not implemented.

31	0
Reserved	

Bits	Name	Description	R/W	Reset
31:0	Reserved	Reserved = 0	R	0

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.4.36 TagLo**COP0 Register: 28**

63		1	0
<i>Reserved</i>			V

Bits	Name	Description	R/W	Reset
63:1	<i>Reserved</i>	<i>Reserved</i>	R	0
0	V	Used by Index Store Tag CACHE Ops to invalidate an entry in the L1D, L1I, or L2 caches. The bit can be read, or can be written with 0.	R/W0	0

4.2.4.37 DataLo**COP0 Register: 28****Note:** Not implemented.

31		0
<i>Reserved</i>		

Bits	Name	Description	R/W	Reset
31:0	<i>Reserved</i>	<i>Reserved = 0</i>	R	0

4.2.4.38 TagHi**COP0 Register: 29**

63		1	0
<i>Reserved</i>			V

Bits	Name	Description	R/W	Reset
63:1	<i>Reserved</i>	<i>Reserved</i>	R	0
0	V	Used by Index Store Tag CACHE Ops to invalidate an entry in the L1D, L1I, or L2 caches. The bit can be read, or can be written with 0.	R/W0	0

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume III. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.4.39 DataHi**COP0 Register: 29****Note:** Not implemented.

31	0
<i>Reserved</i>	

Bits	Name	Description	R/W	Reset
31:0	<i>Reserved</i>	<i>Reserved = 0</i>	R	0

4.2.4.40 ErrorEPC**COP0 Register: 30**

Similar to EPC register, but is used on error exceptions.

63	0
ErrorEPC	

Bits	Name	Description	R/W	Reset
63:0	ErrorEPC	Error Exception Program Counter	R/W	Undefined

- ErrorEPC is also used to store the program counter on RESET, Soft Reset, Non-maskable Interrupt (NMI), and Cache Error exceptions.
- The ErrorEPC register contains the virtual address at which instruction processing can resume after servicing an error.
- ErrorEPC contains either:
 - The virtual address of the instruction that was the direct cause of the exception, or
 - The virtual address of the immediately preceding branch or jump instruction when the error causing instruction is in a branch delay slot.
- Unlike the EPC register, there is no corresponding branch delay slot indication for the ErrorEPC register.

Portions of the preceding information reproduced under license from the MIPS EJTAG Specification, Doc. No. MD00047 rev 3.20. Copyright © MIPS Technologies Inc. All rights reserved.

4.2.4.41 DESAVE**COP0 Register: 31**

Part of EJTAG spec.

63	0
DESAVE	

Bits	Name	Description	R/W	Reset
63	DESAVE	Debug Exception Save contents	R/W	Undefined

The debug exception handler uses this to save one of the GPRs, which is then used to save the rest of the context to a pre-determined memory area, for example, in the dmseg. This register allows the safe debugging of exception handlers and other types of code where the existence of a valid stack for context saving cannot be assumed.

4.3 Coprocessor 2 (COP2) Registers

To access each of the following registers, COP2 must be enabled.

Note that writes to these registers require a 5-cycle window before a subsequent read is performed. A read performed within the 5-cycle window will return stale data.

4.3.1 Coprocessor 2 Register Summary

The Coprocessor 2 (COP2) registers are summarized in [Table 4-2](#).

Table 4-2. Coprocessor 2 Register Summary

Register			Sel	Description	Context
Num	Name	Bits			
0	Transmit Buffer	64	0–3	Transmit Buffers 0–3 Messaging Ring Transmit Buffers	Local
1	Receiver Buffer	64	0–3	Receive Buffers 0–3 Receiver buffer	Local
2	MsgStatus	32	0	MsgStatus Status for the last transmit and receive messages	Local / Global
2	MsgStatus1	32	1	MsgStatus1 Additional status	Local / Global
3	MsgConfig	32	0	MsgConfig Configuration for message interrupts	Global
3	MsgConfig1	32	1	MsgConfig1 Additional configuration	Global
4	MsgBucketSize	32	0–7	MsgBucketSize Sizes for each bucket	Global
5–15	Reserved	32	0–7	Reserved	--
16–31	Credit Counters	32	0–7	Credit Counters 0–127 128 credit counters	Global

Each nCPU (thread) has its own COP2 register set. However, the Fast Messaging Network (FMN) station to which the registers are attached is a shared resource. As such, some of the state information in each nCPU's COP2 Register Set is read directly from the FMN station, lending a "global" or "partly global" context to such information.

For example, only one nCPU per core should access or program MsgBucketSize and the Credit Counters, as these map directly to the FMN station and should be considered global in context.

4.3.2 Coprocessor 2 Register Descriptions

4.3.2.1 Transmit Buffers 0–3

COP2 Reg 0, Select 0–3

Messaging Ring Transmit Buffers (for messages to be sent over the FMN).

Software sets up messages to be sent via the FMN in these registers, before issuing the Send Message (MsgSnd) command. Up to four messages can be sent contiguously using the Send Message command. If only one message element is required, then only one Select 0 need be preloaded before the Send Message command is executed.

All of these registers are Readable/Writable by software. However, software generally only writes to these registers.

63		0		
MRTB				
Bits	Name	Description	R/W	Reset
63:0	MRTB	Messaging Ring Transmit Buffer	R/W	—

4.3.2.2 Receive Buffers 0–3

COP2 Reg 1, Select 0–3

Messaging Ring Receive Buffers (these are per-nCPU buffers).

After a load message command is executed, the message at the head of the selected receive bucket is transferred to the Receive Buffers. (A Message Status Read is required to determine how many elements are in a message, and thus how many Receive Buffers are valid.)

These registers are only Readable by software.

63		0		
MRRB				
Bits	Name	Description	R/W	Reset
63:0	MRRB	Messaging Ring Receive Buffers.	R	—

4.3.2.3 MsgStatus**COP2 Reg 2, Select 0**

This register contains the status information about the last send and receive message. Each nCPU has its own MsgStatus register.

This register is read-only by software.

31	24	23	22	16	15	8	7	6	5	4	3	2	1	0
RFBE	Reserved	RMSID	RMSC	RMS	LEF	LPF	LMP	SCF	SPF	SMP				

Bits	Name	Description	R/W	Reset
31:24	RFBE	Receive FIFO Bucket [7–0] bit mask. The respective bucket's bit is zero if the bucket is not empty, and will be '1' if the bucket is empty.	R	0xFF
23	Reserved	Reserved	0	0
22:16	RMSID	Receive Message Source ID. Software reads this field to determine the ID of the sending Station. Note that the sending station ID will be the first bucket number for that station.	R	0
15:8	RMSC	Receive Message Software Code (only valid if the source is another nCPU).	R	0
7:6	RMS	Receive message size minus 1. Software reads this field to determine the number of valid receive message buffers.	R	0
5	LEF	Load Message Empty Fail: This bit, when set, indicates the Receive FIFO is empty. Software can use this bit to check the Receive FIFO for a message, after calling Load Message.	R	0
4	LPF	Load Message Pending Fail. Load message failed because another one of the same nCPU is pending. Software checks this bit after calling Load Message.	R	0
3	LMP	Load Message Pending: Load Message was accepted, but hasn't completed. Software checks this bit before calling Load Message or Send Message.	R	0
2	SCF	Send Message Credit Failed. No space in destination. Software checks this bit after calling Send Message.	R	0
1	SPF	Send Message Pending Fail. Send Message failed because another one of the same nCPU is pending. Software checks this bit before calling Send Message.	R	0
0	SMP	Send Message Pending. A SndMsg command was accepted but has not been transferred into the transmit FIFO. Software checks this bit before calling Send Message.	R	0

4.3.2.4**MsgStatus1****COP2 Reg 2, Select 1**

The MsgStatus1 register

The contents of this register are Read-only by software. Any write to this register clears all Error Conditions for all fields.

31	30	29	23	22	16	15	14	13	7	6	0
Reserved	C	CCFCME		SIDFCME	T	F		SIDE		DIDE	

Bits	Name	Description	R/W	Reset
31	Reserved	Reserved	0	0
30	C	Credit Overrun Error • 1 = Credit Overrun Error • 0 = no error	R	0
29:23	CCFCME	Credit Counter of Free Credit Message with Error	R	0
22:16	SIDFCME	Source ID of the Free Credit Message with Error	R	0
15	T	Invalid Target Error. • 1 = Invalid Target Error • 0 = no error	R	0
14	F	Receive Queue "Write When Full" Error • 1 = "Write When Receive Queue is Full" Error • 0 = no error	R	0
13:7	SIDE	Source ID of the incoming message with Error	R	0
6:0	DIDE	Destination ID of the incoming message with Error	R	0

4.3.2.5 MsgConfig

COP2 Reg 3, Select 0

The MsgConfig register contains configuration information about how to handle interrupts regarding FMN messages.

If enabled, a level interrupt is triggered when any Receive Bucket transitions from empty or low watermark to occupied. The interrupt condition continues to be true so long as the Receive Bucket is occupied.

For example, when any Receive Bucket transitions from empty to occupied, or when the threshold for the sum of all buckets is exceeded (assuming EIE = 1 and WIE = 1), an interrupt will be generated. The interrupt condition will continue as long as the Receive Bucket is occupied. Hence, if program flow branches to an ISR on this interrupt, and if the ISR does not result in the Receive Queue being emptied, then upon exiting this ISR, program flow will immediately branch back to it so long as the interrupt condition (Receive Queue non-empty) persists.

The Round-Robin Pointer is updated on the falling edge of the OR of all interrupt conditions.

Another interrupt can be delivered when the Receive Queue reaches a watermark level.

Note: There is only one of these registers per CPU, *NOT* per Thread. So any Thread writing to this register writes to the same register.

Note: If interrupts are used to process messages, then all interrupt handlers must support messages from all message sources and buckets. Polling and messaging will not alleviate this requirement.

This register is R/W by software.

31	24 23	22 21	16 15	12 11	8 7	2	1	0
WM	Reserved	IV	Reserved	ITM	Reserved	WIE	EIE	

Bits	Name	Description	R/W	Reset
31:24	WM	Watermark level; refers to the sum of all buckets.	R/W	0xf0
23:22	Reserved	Reserved. Set to 0	R	0
21:16	IV	Interrupt Vector. The interrupt vector delivered to the CPU.	R/W	0
15:12	Reserved	Reserved. Set to 0	R	0
11:8	ITM	Interrupt Thread Mask. Indicates which Threads on a core to deliver the interrupt in a round-robin scheduling scheme. 11: thread 3 10: thread 2 9: thread 1 8: thread 0	R/W	0xff
7:2	Reserved	Reserved. Set to 0	R	0

Bits	Name	Description	R/W	Reset
1	WIE	WaterMark Interrupt Enable. 0: not enabled (default) 1: enabled	R/W	0
0	EIE	Receive Queue Not Empty Enable. 0: not enabled (default) 1: enabled	R/W	0

4.3.2.6 MsgConfig1

COP2 Reg 3, Select 1

The MsgConfig1 controls the debug/de-featuring aspects of messaging

63	3	2	1	0
Reserved	T	C	M	

Bits	Name	Description	R/W	Reset
63:3	Reserved	Reserved. Set to 0	R	0
2	T	Trace Mode Enable. (Default is 0, NOT enabled.)	R/W	0
1	C	Credit Over-run Interrupt Enable. (Default is 0, NOT enabled.)	R/W	0
0	M	Messaging Errors Interrupt Enable. (Default is 0, NOT enabled.)	R/W	0

4.3.2.7 MsgBucketSize**COP2 Reg 4, Select 0–7**

These eight registers contain the sizes for Buckets 0 through 7. There is one register per Bucket. Each register is 64 bits wide, with bits 63:8 Reserved, and bits 7:0 specifying the size.

63		8	7	0
Reserved				SIZE n

Bits	Name	Description	R/W	Reset
63:8	Reserved	Reserved. Set to 0	R	0
7:0	SIZE n	Size of the n^{th} Bucket.	R/W	8

Size has the following requirement:

- Must be power of 2, or 0 (writing a value of 0, disables the bucket).
- Must be greater than or equal to 4, if not 0.

Based on the size specified for each bucket, the logic calculates the base address into the RcvQ FIFO for each bucket.

The Base address for each bucket is calculated as follows:

$$\begin{aligned} \text{Base}(0) &= 0 \\ \text{BaseAddr}(N) &= \text{BaseAddr}(N-1) + \text{Size}(N-1). \end{aligned}$$

The resulting Base Addresses have the following requirement.

$$-(\text{BaseAddr} \bmod \text{Size}) == 0$$

Thus, specifying `Size(0) = 16`, and `Size(1) = 4`, and `Size(2) = 16` is illegal, because this results in `BaseAddr(2) = 20 (0x14)`.

At Reset, each size is initialized to 32 for the CPUs (Total receive queue size per number of buckets = 256/8).

The total of all these registers should be less than or equal to the total Receive Queue size.

These registers should only be updated at initialization when there are no messages in the system.

The results of writing to these registers while there are messages in the system is unpredictable.

Note: There is only one of these registers per CPU, NOT per nCPU. So any nCPU writing to this register writes to the same register as any other nCPU on the same core.

4.3.2.8 Reserved**COP2 Reg 5, Select 0–7**

63	0
<i>Reserved</i>	

Bits	Name	Description	R/W	Reset
63:0	<i>Reserved</i>	<i>Reserved.</i> Set to 0		Undefined

4.3.2.9 Credit Counters 0–127**COP2 Reg 16–31, Select 0–7**

These are credit counters for the FMN destination station Receive Buckets.

There are a total of 128 credit counters for each FMN-addressable bucket.

The sum of credits for a particular destination bucket MUST NOT EXCEED the size of that bucket.

Note: There is only one of these registers per CPU, *NOT* per nCPU. So any nCPU on the same core writing to this register to initialize credits for a destination bucket will write to the same register.

7	0
CC	

Bits	Name	Description	R/W	Reset
7:0	CC	Credit count. When read, this register returns the current credit count for each destination bucket.	R/W	0

4.4 XLS Processor Control Registers

This section details these registers, which are important in XLS system setup and debugging.

4.4.1 MFCR / MTCR Instructions

MFCR (**M**ove **F**rom **C**ontrol **R**egister) and MTCR (**M**ove **T**o **C**ontrol **R**egister) instructions provide software access to XLS processor control registers. The MFCR instruction is used to move the contents of an XLS processor control register into a CPU General-Purpose Register (GPR). The MTCR instruction moves the contents of a GPR to an XLS control register.

XLS processor control registers are divided into blocks. Individual registers within each block are identified by a register offset.

The *rs* field of the MFCR/MTCR instructions contains the Block ID, to which is appended the register offset (within that block) of the targeted XLS processor control register. Bits [15:8] of the *rs* field specify the Block ID, and bits [7:0] specify the register's offset within the block.

For example, XLS Processor Control Register ICU_SAMPLING_SETUP is in block 1 (i.e., 00000001b), with a register offset of 7 (i.e., 00000111b). Thus, the address in the *rt* field of an MFCR or MTCR instruction targeting this register will be 0000 0001 0000 0111, or 0x107.

The Enable bit (bit 0) of this register can be set as follows:

```
#define ICU_SAMPLING_SETUP_IND 0x107
#define RS 0x7 // use GPR 7 for index
#define RT 0x8 // use GPR 8 for data
#define ICU_SAMPLING_SETUP_ENA 1
MOV RT, ICU_SAMPLING_SETUP_ENA
MOV RS, ICU_SAMPLING_SETUP_IND
MTCR RT, RS
```

The following sections describe the registers accessed through this mechanism

4.4.2

XLS Processor Control Registers Summary.**Table 4-3. XLS Processor Control Registers Summary**

Block ID	Local Offset within Block	[Block ID : Offset] = Register Address	Register Name
0	0	0x000	ThreadEn
0	1	0x001	Software Sleep
0	2	0x002	Scheduling
0	3	0x003	Scheduling Counters
0	4	0x004	Branch History Register Programmable Mask (BHRPM)
0	6	0x006	IFU_DEFECTURE
1	0	0x100	ICU_DEFECTURE
1	1	0x101	ICU_ERROR_LOGGING
1	2	0x102	ICU_DEBUG_ACCESS_ADDR
1	3	0x103	ICU_DEBUG_ACCESS_DATALO
1	4	0x104	ICU_DEBUG_ACCESS_DATAHI
1	5	0x105	ICU_SAMPLING_LFSR
1	6	0x106	ICU_SAMPLING_PC
1	7	0x107	ICU_SAMPLING_SETUP
1	8	0x108	ICU_SAMPLING_TIMER
1	9	0x109	ICU_SAMPLING_PC_UPPER
2	0	0x200	IEU_DEFECTURE
2	7	0x207	Target PC Register
3	0	0x300	L1D_CONFIG0
3	1	0x301	L1D_CONFIG1
3	2	0x302	L1D_CONFIG2
3	3	0x303	L1D_CONFIG3
3	4	0x304	L1D_CONFIG4
3	5	0x305	L1D_STATUS
3	6	0x306	L1D_DEFECTURE
3	7	0x307	L1D_DEBUG0
3	8	0x308	L1D_DEBUG1
3	9	0x309	L1D_CACHE_ERROR_LOG
3	A	0x30A	L1D_CACHE_ERROR_OVF_LO
3	B	0x30B	L1D_CACHE_INTERRUPT
4	0	0x400	MMU_SETUP
5	0	0x500	PRF_SMP_EVENT
5	1	0x501	PRF_SMP_RPLY_BUF

4.4.3 XLS Processor Control Registers Descriptions

4.4.3.1 ThreadEn

Block ID: 0 Offset: 0 Address: 0x000

All Threads can R/W all bits.

63:4	3:0
<i>Reserved</i>	ENA

Bits	Name	Description	R/W	Reset
63:4	Reserved	<i>Reserved</i>		
3:0	ENA	Enable bits for each Thread. Software can write this to enable/disable Threads. 0: 0 = Thread 0 disabled; 1 = Thread 0 enabled. 1: 0 = Thread 1 disabled; 1 = Thread 1 enabled 2: 0 = Thread 2 disabled; 1 = Thread 2 enabled 3: 0 = Thread 3 disabled; 1 = Thread 3 enabled	R/W	0xF

4.4.3.2 Software Sleep

Block ID: 0 Offset: 1 Address: 0x001

All Threads can R/W all bits.

63:4	3:0
<i>Reserved</i>	SLEEP

Bits	Name	Description	R/W	Reset
63:4	Reserved	<i>Reserved</i>		
3:0	SLEEP	Sleep bits for each Thread. For each bit set to 1, the corresponding Thread is put to sleep. When an interrupt is received for the sleeping Thread, that Thread is awakened and the corresponding bit is cleared. 0 : 0 = Thread 0 not asleep; 1 = Thread 0 asleep 1 : 0 = Thread 1 not asleep; 1 = Thread 1 asleep 2 : 0 = Thread 2 not asleep; 1 = Thread 2 asleep 3 : 0 = Thread 3 not asleep; 1 = Thread 3 asleep	R/W	0

4.4.3.3 Scheduling**Block ID: 0 Offset: 2 Address: 0x002***All Threads can R/W all bits.*

63:2	
Reserved	SCHED

Bits	Name	Description	R/W	Reset
63:2	Reserved	Reserved.		
1:0	SCHED	Scheduling policy. Bit 0 controls Coarse Grained Scheduling. Bit 1 controls Priority Based Scheduling. 00: Fine-grained (Round Robin) scheduling 01: Coarse-grained (fixed cycle) scheduling 10: Priority scheduling 11: Reserved (defaults to fine-grained scheduling)	R/W	0

4.4.3.4 Scheduling Counters**Block ID: 0 Offset: 3 Address: 0x003***All Threads can R/W all bits.*

63:32	31:24	23:16	15:8	7:0
Reserved	ISCT3	ISCT2	ISCT1	ISCT0

Bits	Name	Description	R/W	Reset
31:24	ISCT3	Instruction scheduling counter for Thread 3.	R/W	0
23:16	ISCT2	Instruction scheduling counter for Thread 2.	R/W	0
15:8	ISCT1	Instruction scheduling counter for Thread 1.	R/W	0
7:0	ISCT0	Instruction scheduling counter for Thread 0.	R/W	0

4.4.3.5 Branch History Register Programmable Mask (BHRPM)

Block ID: 0 **Offset: 4** **Address: 0x004**

All Threads can R/W all bits.

To help improve performance, each XLS CPU uses a branch prediction mechanism that maintains a 2K-entry Branch History Table (BHT) and which uses the internal Branch History Register. Users can tune the mechanism's performance approximately 1 to 2% using the BHRPM register value, which changes the way history is accessed in the Branch History Table.

The mechanism accesses the table by computing an index using the program counter, the BHRPM value, and the Branch History Register (BHR) contents. The BHRPM value controls how many bits of the BHR are used in the calculation. The code below shows how the index is calculated:

```
index[4:0] = pc[6:2] xor BHR[4:0];
index[10:5] = BHRPM[5:0] ? pc[12:7] xor BHR[10:5] : pc[12:7];
```

where

pc = program counter value

BHR[] = contents of the Branch History Register (not user-accessible)

BHRPM[] = mask value in the BHRPM Register. Masks must be one of the following values:

Bit					
5	4	3	2	1	0
0	0	0	0	0	0
0	0	0	0	0	1
0	0	0	0	1	1
0	0	0	1	1	1
0	0	1	1	1	1
0	1	1	1	1	1
1	1	1	1	1	1

63:6		5:0		
Reserved		BHRPM		
Bits	Name	Description	R/W	Reset
63:6	Reserved	Reserved		
5:0	BHRPM	Branch History Register Programmable Mask.	R/W	3

4.4.3.6 IFU_DEFFEATURE**Block ID: 0 Offset: 6 Address: 0x006***All Threads can R/W all bits.*

63:6	5	4:0						
<i>Reserved</i>		Low Power Disable IFU	<i>Reserved</i>					
Bits	Name	Description			R/W	Reset		
63:6	<i>Reserved</i>	<i>Reserved</i>						
5	LowPowerDisableIFU	Disable Clock Gating. 1: Disable clock gating 0: Enable clock gating			R/W	0		
4:0	<i>Reserved</i>	<i>Reserved</i>						

4.4.3.7 ICU_DEFFEATURE**Block ID: 1 Offset: 0 Address: 0x100**

63:20	19	18	17	16	15	14:7	6:3	2	1	0
Reserved'	Low Power Disable ICU	Partitioned Miss Queue	Partitioned IUTLB	Force Mapped ERL	Force Mapped Mode	W[7:0] Cache Way En	T[3:0] Parity Interrupt En	Parity Detect En	Spec Cache Miss En	Miss Queue Merge En

Bits	Name	Description	R/W	Reset
63:20	Reserved	Reserved		
19	LowPowerDisableICU	If 1, clock gating for the ICU is disabled, i.e. clocks are free running regardless of activity. 0: Enable clock gating 1: Disable clock gating	R/W	0
18	partitionedMissQueue	If set, the 8-entry ICU miss queue is equally partitioned based on MMU_SETUP.MMU_Thread_Mode. Each nCPU can then write only those entries within its own partition.	R/W	0
17	partitionedIUTLB	If set, the 32-entry IuTLB is equally partitioned based on MMU_SETUP.MMU_Thread_Mode. Each nCPU can then write only those entries within its own partition.	R/W	0
16	ForceMappedERL	If set, I-side accesses to kuseg with Status.ERL = 1 are handled as mapped accesses (i.e. they undergo translation in the IuTLB/mainTLB).	R/W	0
15	ForceMappedMode	If set, I-side accesses to unmapped regions of the MIPS address map (not including kuseg with Status.ERL = 1) are handled as mapped accesses (i.e. they undergo translation in the IuTLB/mainTLB).	R/W	0
14	w7CacheWayEn	Enables I-cache way 7.	R/W	0
13	w6CacheWayEn	Enables I-cache way 6.	R/W	0
12	w5CacheWayEn	Enables I-cache way 5.	R/W	0
11	w4CacheWayEn	Enables I-cache way 4.	R/W	0
10	w3CacheWayEn	Enables I-cache way 3.	R/W	0
9	w2CacheWayEn	Enables I-cache way 2.	R/W	0
8	w1CacheWayEn	Enables I-cache way 1.	R/W	0
7	w0CacheWayEn	Enables I-cache way 0.	R/W	0
6	t3ParityInterruptEn	Enables parity error interrupts for thread 3 I-cache accesses.	R/W	0
5	t2ParityInterruptEn	Enables parity error interrupts for thread 2 I-cache accesses.	R/W	0
4	t1ParityInterruptEn	Enables parity error interrupts for thread 1 I-cache accesses.	R/W	0
3	t0ParityInterruptEn	Enables parity error interrupts for thread 0 I-cache accesses.	R/W	0
2	ParityDetectEn	Enables I-cache parity error detection.	R/W	0
1	SpecCacheMissEn ^a	If set, I-cache misses result in an immediate L2-cache request, without waiting for older instructions from that nCPU to retire.	R/W	1
0	MissQueueMergeEn	Multiple I-cache misses to the same physical address will result in only one request to L2-cache.	R/W	1

- a. To avoid possible corruption of the instruction stream, the value of this bit should be changed using uncacheable instructions.

4.4.3.8 ICU_ERROR_LOGGING**Block ID: 1 Offset: 1 Address: 0x101**

63:23	22	21	20	19:12	11:5	4	3:0
Reserved	microtlb Multi Hit	bus Data Error	bus Address Error	Way	index	corrected	valid

Bits	Name	Description	R/W	Reset
63:23	Reserved	Reserved		
22	microtlbMultihit	The IuTLB experienced multiple hits while CAMing.	R/W	0
21	busDataError	The BIU signalled a bus data error to the ICU.	R/W	0
20	busAddressError	The Bus Interface Unit (BIU) signalled an address error to the ICU.	R/W	0
19:12	way	The I-cache way that experienced the parity error.	R/W	n/a
11: 5	index	The I-cache index that experienced the parity error.	R/W	n/a
4	corrected	The parity error has already been corrected (i.e. the I-cache line has been invalidated) by the h/w.	R/W	0
3:0	valid	The thread whose I-cache access experienced the parity error.	R/W	0

4.4.3.9 ICU_DEBUG_ACCESS_ADDR**Block ID: 1 Offset: 2 Address: 0x102**

Table 4-4 describes how to program the fields for debug accesses to the various ICU arrays.

63:18	17:15	14:12	11:4	3:2	1	0
Reserved	encoded Array Sel	encoded Way Sel	index	data Select	write Enable	trans Active

Bits	Name	Description	R/W	Reset
63:18	Reserved	Reserved		
17:15	encodedArraySel	See the table below.	R/W	n/a
14:12	encodedWaySel	See the table below.	R/W	n/a
11:4	index	See the table below.	R/W	n/a
3:2	dataSelect	See the table below.	R/W	n/a
1	writeEnable	Can be set by s/w to perform a ICU debug write.	R/W	0
0	transActive	Can be set by s/w to trigger the ICU debug access. Reset by hardware when the debug access is done.	R/W	0

Table 4-4. Fields for Debug Accesses to ICU Arrays

Array Name	Logical Size; [Physical Size]	Encoded_array_select	Encoded_way_select	Index	Data_Select	Data Used For Write	Read Data Location
Vld	128e_8b [64e_16b]	000	Picks write way	[11:5] pick entry	N/A	DataLo[0]	DataLo[7:0]
Tag	128e_8w_29b [128e_8w_29b]	001	Picks read and write way	[11:5] pick entry	N/A	DataLo[28:0]	DataLo[28:0]
Data	128e_8w_288b [256e_8w_2B_72b]	010	Picks read and write way	[11:4] pick entry	[3] picks bank for read and write	{DataHi[71:40], DataLo[39:0]}	{DataHi[71:40], DataLo[39:0]}
LRU/LCK	128e_15b [64e_30b]	011	Picks LCK array write way	[11:5] pick entry	N/A	DataLo[14:8](LRU), one of DataLo[7:0](LCK)	DataLo[14:8](LRU) DataLo[7:0](junk)
Miss Queue	8e_38b [8e_38b]	100	N/A	[6:4] pick entry	N/A	Miss Queue valid bit reset to 0.	DataLo[39:0]
Sleep Queue	4e_12b [4e_12b]	101		[5:4] pick entry	N/A	Sleep queue valid bit reset to 0.	DataHi[11:0]
uTLB_TAG	32e_80b [32e_80b]	110	N/A	[8:4] pick entry	N/A	None None	DataLo[39:0] DataHi[79:40]
uTLB_DATA	32e_44b [32e_44b]	111	N/A	[8:4] pick entry	N/A	None None	DataLo[39:0] DataHi[3:0]

4.4.3.10 ICU_DEBUG_ACCESS_DATALO

Block ID: 1 Offset: 3 Address: 0x103

		63:40	39:0
Reserved		data	

Bits	Name	Description	R/W	Reset
63:40	Reserved	Reserved		
39:0	data	Least-significant 40 bits of ICU debug access read or write data.	R/W	n/a

4.4.3.11 ICU_DEBUG_ACCESS_DATAHI

Block ID: 1 Offset: 4 Address: 0x104

		63:40	39:0
Reserved		data	

Bits	Name	Description	R/W	Reset
63:40	Reserved	Reserved		
39:0	data	Most-significant 40 bits of ICU debug access read or write data.	R/W	n/a

4.4.3.12 ICU_SAMPLING_LFSR

Block ID: 1 Offset: 5 Address: 0x105

See [Section 25.7, “Instruction Statistical Sampling”](#) for information on the use of this register.

		63:20	19:0
Reserved		lfsr_value	

Bits	Name	Description	R/W	Reset
63:20	Reserved	Reserved		
19:0	lfsr_value	LFSR used for the instruction statistical sampling performance monitoring feature.	RO	0xBA5AB

4.4.3.13 ICU_SAMPLING_PC

Block ID: 1 Offset: 6 Address: 0x106

See [Section 25.7, “Instruction Statistical Sampling”](#) for information on the use of this register.

		63:0
sample_pc		

Bits	Name	Description	R/W	Reset
63:0	sample_pc	PC of the instruction chosen for sampling. Only bits 39:0 can be written using this address (use REGID 0x0109 to write bits 63:40).	R/W	n/a

4.4.3.14 ICU_SAMPLING_SETUP

Block ID: 1 Offset: 7 Address: 0x107

See [Section 25.7, “Instruction Statistical Sampling”](#) for information on the use of this register.

63:16		15:14		13	12	11	10:9	8:1	0
Reserved		report Threadid	pc Mode	interrupt Enable	thread Global	encoded Threadid	Ifsr Mask	enable	
Bits Name Description R/W Reset									
63:16	Reserved	Reserved							
15:14	reportThreadid	The thread to which an interrupt should be signalled when the sample is done. Always the same as the thread which programmed this register.		Ronly	0				
13	pcMode	If set, the instruction selected for sampling should have the PC specified in ICU_SAMPLING_PC.		R/W	0				
12	interruptEnable	If set, an interrupt should be taken once the sample is complete. Note that the interrupt will be mapped to timer 7 and may be shared. Software must check the status of bit 0 (enable) to determine whether a COP0 compare or a statistical sampling event caused the interrupt.		R/W	0				
11	threadGlobal	If set, an instruction may be selected for sampling from any active thread.		R/W	0				
10:9	encodedThreadid	Identifies the thread from which an instruction should be selected for sampling.		R/W	0				
8:1	IfsrMask	If set, removes corresponding bit of sampling LFSR[17:10] from instruction selection criteria, thereby making instruction selection more frequent.		R/W	0xFF				
0	enable	Enable instruction statistical sampling performance monitoring feature. Reset by hardware to 0 once a sample has been captured.		R/W	0				

4.4.3.15 ICU_SAMPLING_TIMER

Block ID: 1 Offset: 8 Address: 0x108

63:10			9:0		
Reserved			timer_value		
Bits Name Description R/W Reset					
63:10	Reserved	Reserved.			
9:0	timer_value	Timeout timer used for the instruction statistical sampling performance monitoring feature.	RO	0	

4.4.3.16 ICU_SAMPLING_PC_UPPER

Block ID: 1 Offset: 9 Address: 0x109

See [Section 25.7, "Instruction Statistical Sampling"](#) for information on the use of this register.

63:24		23:0			
Reserved		PC_upper			
Bits	Name	Description		R/W	Reset
63:24	Reserved	Reserved.			
23:0	PC_upper	This address is used to write bits 63:40 of the ICU_SAMPLING_PC register.		WO	n/a

4.4.3.17 IEU_DEFEATURE

Block ID: 2 Offset: 0 Address: 0x200

Some capabilities of the IEU can be de-featured by writing to this register.

63:8	7	6	5	4	3	2:0
Reserved	DBE	Reserved	Low Power Disable Div	Low Power Disable Mult	Low Power Disable IEU	Reserved

Bits	Name	Description	R/W	Reset
63:8	Reserved	Reserved		
7	DBE	Debug Break Enable. When a watch exception is signalled, setting this bit causes a debug breakpoint exception to be reported instead of a Watch exception.	R/W	0
6	Reserved	Reserved.	R/W	0
5	LowPowerDisableDiv	Low-power disable for Div 1: Disable clock gating for divide 0: Enable clock gating for divide	R/W	0
4	LowPowerDisableMult	Disable Clock Gating for Mult 1: Disable clock gating for multiply 0: Enable clock gating for multiply	R/W	0
3	LowPowerDisableIEU	Disable Clock Gating for IEU (Does not include Mult and Div) 1: Disable clock gating 0: Enable clock gating	R/W	0
2:0	Reserved	Reserved		

4.4.3.18 Target PC Register

Block ID: 2 Offset: 7 Address: 0x207

This is the PC of the Branch or Jump Target Instruction.

63:0					
TargetPC					
Bits	Name	Description		R/W	Reset
63:0	TargetPC	TargetPC		RO	0

4.4.3.19 L1D_CONFIG0

Block ID: 3 Offset: 0 Address: 0x300

63:15	14	13	12	11:8	7	6	5	4	3	2	1	0
RSVD	Unalign AccEn	Low Power Disable L1D	Force Mapped Data Erl	Mem Ordering Mode	uTLB Partitioning	Force Mapped Data	Cache Error En	L2 Line Alloc En	Data ECC En	Snoop Tag Parity En	Tag Parity En	L1D_CEn

Bits	Name	Description	R/W	Reset
63:15	Reserved	Reserved.	R/W	
14	UnalignAccEn	Unaligned Access Enable. When enabled, LH, LHU, LW, LWU, LD, SH, SW, and SD instructions may access memory addresses along any byte boundary. 0: Memory accesses must be naturally-aligned. Unaligned accesses will result in an Address Error exception. 1: Memory accesses can be unaligned. Unaligned accesses will NOT result in an Address Error exception. When an unaligned store crosses an 8-byte boundary and an exception (e.g. -- TLB, interrupt, etc.) occurs on the part that crosses the boundary, then the first part will complete correctly, but an exception will be taken and the EPC will point to the store's PC. Note: Some operating systems may use Address Errors to pass control between processes (for example, when doing floating point emulation). If Unaligned Accesses are enabled, then the desired exception may not be taken. One example of such an operating system is Linux. In this case, the unaligned exception was created by changing an unaligned LW to an unaligned LL.	R/W	0
13	LowPowerDisableL1D	Low Power DisableL1D. Controls low power mode for a D cache in a core. Normally, low power mode is enabled. This bit allows disabling the mode for this core. 0: enable LP mode (default) 1: disable LP mode	R/W	0
12	ForceMappedDataErl	Virtual MIPS mode - ERL is virtualized 0: disable (default) 1: enable virtualizing of ERL	R/W	0

Bits	Name	Description	R/W	Reset															
11:8	MemOrderingMode	Memory ordering mode. Any combination of enabled modes is valid.	R/W	see left															
		<table border="1"> <thead> <tr> <th>Bit</th><th>Function</th><th>Value</th></tr> </thead> <tbody> <tr> <td>11</td><td>LD-LD ordering</td><td>0: disable (default) 1: enable</td></tr> <tr> <td>10</td><td>ST-LD ordering</td><td>0: disable (default) 1: enable</td></tr> <tr> <td>9</td><td>LD-ST ordering</td><td>0: disable (default) 1: enable</td></tr> <tr> <td>8</td><td>ST-ST ordering</td><td>0: disable 1: enable (default)</td></tr> </tbody> </table>	Bit	Function	Value	11	LD-LD ordering	0: disable (default) 1: enable	10	ST-LD ordering	0: disable (default) 1: enable	9	LD-ST ordering	0: disable (default) 1: enable	8	ST-ST ordering	0: disable 1: enable (default)		
Bit	Function	Value																	
11	LD-LD ordering	0: disable (default) 1: enable																	
10	ST-LD ordering	0: disable (default) 1: enable																	
9	LD-ST ordering	0: disable (default) 1: enable																	
8	ST-ST ordering	0: disable 1: enable (default)																	
7	uTLBPartitioning	uTLB partitioned by threads for this core. 0: disable (default) 1: enable	R/W	0															
6	ForceMappedData	Force-mapped data (Virtual MIPS mode; see chapter 3.) 0: disable (default) 1: enable	R/W	0															
5	CacheErrorEn	Cache error log and report. Enables logging and reporting of L1 D cache errors for this core 0: disable (default) 1: enable	R/W	0															
4	L2LineAllocEn	L2 line allocation. Enables use of line allocation in L2 D cache when servicing L1 miss request or line eviction. 0: disable 1: enable feature	R/W	0															
3	DataECCEn	Data ECC enable 0: disable (default) 1: enable	R/W	0															
2	SnoopTagParityEn	Snoop Tag parity enable 0: disable (default) 1: enable	R/W	0															
1	TagParityEn	Tag parity enable 0: disable (default) 1: enable	R/W	0															
0	L1D_CEn	L1 D cache enable 0: disable (default) 1: enable	R/W	0															

4.4.3.20 L1D_CONFIG1

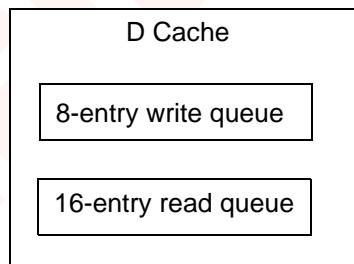
Block ID: 3 Offset: 1 Address: 0x301

63	48 47	32
	RDQ3	
31	16 15	0
	RDQ1	

Bits	Name	Description	R/W	Reset
63:48	RDQ3	Thread 3 RDQ mask 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0
47:32	RDQ2	Thread 2 RDQ mask 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0
31:16	RDQ1	Thread 1 RDQ mask 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0
15:0	RDQ0	Thread 0 RDQ mask 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0

RDQ Control

As shown below, the D cache has an 8 entry write queue and a 16 entry read queue. The bits of the fields in this register mask the use of the corresponding position in the Read Queue.



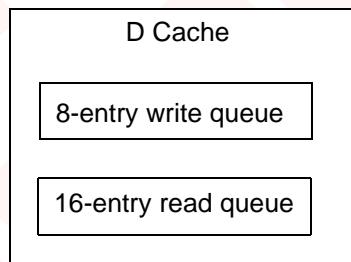
4.4.3.21 L1D_CONFIG2**Block ID: 3****Offset: 2****Address: 0x302**

31	24 23	16 15	8 7	0
WRQ3	WRQ2	WRQ1	WRQ0	

Bits	Name	Description	R/W	Reset
31:24	WRQ3	Thread 3 WDQ mask 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0
23:16	WRQ2	Thread 2 WDQ mask 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0
15:8	WRQ1	Thread 1 WDQ mask 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0
7:0	WRQ0	Thread 0 WDQ mask 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0

WRQ Control

As shown below, the D cache has an 8 entry write queue and a 16 entry read queue. The bits of these fields mask the use of the corresponding position in the Write Queue.



4.4.3.22 L1D_CONFIG3

Block ID: 3 Offset: 3 Address: 0x303

63	8 7	0
Reserved		SNOOPMASK

Bits	Name	Description	R/W	Reset
63:8	Reserved	Reserved		
7:0	SNOOPMASK	Snoop Mask. For a snoop, these bits mask use of a position in the D cache's Write queue. See diagram for L1_config2 WRQ control. 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0

4.4.3.23 L1D_CONFIG4

Block ID: 3 Offset: 4 Address: 0x304

63	4 3 2 1 0	
Reserved		F_ST_PE F_T_PE F_DBE F_SBE

Bits	Name	Description	R/W	Reset
63:4	Reserved	Reserved		
3	F_ST_PE	Force Snoop Tag Parity Error 0: no action 1: force snoop tag parity error.	R/W	0
2	F_T_PE	Force Tag Parity Error 0: no action 1: force tag parity error.	R/W	0
1	F_DBE	Force Double Bit Error 0: no action 1: force double bit error.	R/W	0
0	F_SBE	Force Single Bit Error 0: no action 1: force single bit error.	R/W	0

These control bits are for testing use, The user must manually clear them if they had previously set them.

4.4.3.24 L1D_STATUS

Block ID: 3 Offset: 5 Address: 0x305

63	6	5	2	1	0
	Reserved		CCEP	WRQE	RDQE

Bits	Name	Description	R/W	Reset
63:6	Reserved	Reserved		
5:2	CCEP	Cumulative Cache Error Pending 0: no activity 1: indicates activity in either the cache error log or the error ovl log. User must read status in those logs to find the source. Once user has cleared the log, this bit clears. Bits 5 - thread 3 4 - thread 2 3 - thread 1 2 - thread 0	R	0
1	WRQE	WRQ empty 0: not empty 1: empty	R	0
0	RDQE	RDQ empty 0: not empty 1: empty	R	0

4.4.3.25 L1D_DEFFEATURE

Block ID: 3 Offset: 6 Address: 0x306

Bit Range							
31		24	23	22	21 20		
13		12	0				
Reserved		DP	Rsv	FPA	WAYDIS	Reserved	
Bits		Description				Type	Reset
#	Name						
31:24	Reserved	Reserved				R/W	0
23	DP	Disable Prefetches 0: enable prefetches 1: disables prefetches				R/W	0
22	Reserved	Reserved (must write 0)				R/W	0
21	FPA	Flush on prefetch abort 0: disable flush 1: enable flush				R/W	0
20:13	WAYDIS	Way Disable 0: enable way 1: disable way Bits: 20: way 7 .. 13: way 0				R/W	0
12:0	Reserved	Reserved (must write 0)				R/W	0

4.4.3.26 L1D_DEBUG0

Block ID: 3 Offset: 7 Address: 0x307

63:28		27:0		
Reserved		tag_data		
Bits	Name	Description	Type	Reset
63:28	reserved	Reserved.		
27:0	tag_data	Tag data.	R/W	0

4.4.3.27 L1D_DEBUG1**Block ID: 3 Offset: 8 Address: 0x308**

31	30	24	23	21	20	18	17	15	14	12	11	9	8	6	5	3	2	0
Reserved	LRU		MOSI7	MOSI6	MOSI5	MOSI4	MOSI3	MOSI2	MOSI1	MOSI0								

Bits	Name	Description	R/W	Reset
31	Reserved	Reserved		
30:24	LRU	LRU Status of cache.	R	Undefined
23:21	MOSI7	Data for MOSI way 7. MOSI encoding: 0 : INVALID 1 : SHARED 2 : MODIFIED 3 : OWN {7-4} : Intermediate states not defined.	R	Undefined
20:18	MOSI6	MOSI way 6. See above	R	Undefined
17:15	MOSI5	MOSI way 5. See above	R	Undefined
14:12	MOSI4	MOSI way 4. See above	R	Undefined
11:9	MOSI3	MOSI way 3. See above	R	Undefined
8:6	MOSI2	MOSI way 2. See above	R	Undefined
5:3	MOSI1	MOSI way 1. See above	R	Undefined
2:0	MOSI0	MOSI way 0. See above	R	Undefined

These parameters are loaded upon execution of a cache op load tag instruction.

Note: This register is for NetLogic engineering debug purposes only.

4.4.3.28 L1D_CACHE_ERROR_LOG

Block ID: 3

Offset: 9

Address: 0x309

63	56	55	54	47	46	10	9	8	7	6	5	4	3	2	1	0
Reserved	L	WHV		ADR		D	A	P	B	S	T	E3	E2	E1	E0	

Cache error log.

Bits	Name	Description	R/W	Reset
63:56	Reserved	Reserved		
55	L	Dirty Line 0: write 0 to clear 1: dirty line	R/W	0
54:47	WHV	Way hit vector	R/W	0
46:10	ADR	The physical address associated with the error. Physical addresses are 40 bits wide (i.e., [39:0]). This field holds the word-aligned version of the error's physical address--that is, it holds bits [39:2] of the error's physical address.	R/W	0
9	D	Data Error 0: write 0 to clear 1: data error	R/W	0
8	A	Address Error 0: write 0 to clear 1: address error	R/W	0
7	P	Tag Parity Error 0: write 0 to clear 1: tag parity error	R/W	0
6	B	Double Bit Error 0: write 0 to clear 1: double bit error	R/W	0
5	S	Store Type 0: write 0 to clear 1: store type error	R/W	0
4	T	Load Type 0: write 0 to clear 1: load type error	R/W	0
3	E3	Cache Error Pending thread 3 0: write 0 to clear 1: cache error pending - thread 3	R/W	0
2	E2	Cache Error Pending thread 2 0: write 0 to clear 1: cache error pending - thread 2	R/W	0
1	E1	Cache Error Pending thread 1 0: write 0 to clear 1: cache error pending - thread 1	R/W	0
0	E0	Cache Error Pending thread 0 0: write 0 to clear 1: cache error pending - thread 0	R/W	0

4.4.3.29 L1D_CACHE_ERROR_OVF_LOG**Block ID: 3****Offset: 0x0A****Address: 0x30A**

31	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	DL	DE	AE	TPE	DBE	ST	LT	CEO_T3	CEO_T2	CEO_T1	CEO_T0	

Bits	Name	Description	R/W	Reset
31:11	Reserved	Reserved		
10	DL	Dirty Line 0: write 0 to clear 1: dirty line	R/W	0
9	DE	Data Error 0: write 0 to clear 1: data error	R/W	0
8	AE	Address Error 0: write 0 to clear 1: address error	R/W	0
7	TPE	Tag Parity Error 0: write 0 to clear 1: tag parity error	R/W	0
6	DBE	Double Bit Error 0: write 0 to clear 1: double bit error	R/W	0
5	ST	Store Type 0: write 0 to clear 1: store type error	R/W	0
4	LT	Load Type 0: write 0 to clear 1: load type error	R/W	0
3	CEO_T3	Cache Error Overflow thread 3 0: write 0 to clear 1: cache error overflow - thread 3	R/W	0
2	CEO_T2	Cache Error Overflow thread 2 0: write 0 to clear 1: cache error overflow - thread 2	R/W	0
1	CEO_T1	Cache Error Overflow thread 1 0: write 0 to clear 1: cache error overflow - thread 1	R/W	0
0	CEO_T0	Cache Error Overflow thread 0 0: write 0 to clear 1: cache error overflow - thread 0	R/W	0

4.4.3.30 L1_CACHE_INTERRUPT

Block ID: 3 Offset: 0x0B Address: 0x30B

63	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	CS	CI	MH	DP	SBE	CIP_T3	CIP_T2	CIP_T1	CIP_T0	CIM_T3	CIM_T2	CIM_T1	CIM_T0	

Bits	Name	Description	R/W	Reset
63:13	Reserved	Reserved		
12	CS	Cache set. See note.	R/W	0
11	CI	Cache index. See note.	R/W	0
10	MH	uTLB multi-hit 0: write 0 to clear 1: this was source of interrupt	R/W	0
9	DP	Data poisoned 0: write 0 to clear 1: this was source of interrupt	R/W	0
8	SBE	Single Bit Error 0: write 0 to clear 1: this was source of interrupt	R/W	0
7	CIP_T3	Cache Interrupt Pending thread 3 0: write 0 to clear 1: this was source of interrupt	R/W	0
6	CIP_T2	Cache Interrupt Pending thread 2 0: write 0 to clear 1: this was source of interrupt	R/W	0
5	CIP_T1	Cache Interrupt Pending thread 1 0: write 0 to clear 1: this was source of interrupt	R/W	0
4	CIP_T0	Cache Interrupt Pending thread 0 0: write 0 to clear 1: this was source of interrupt	R/W	0
3	CIM_T3	Cache Interrupt Enable thread 3 0: disables interrupt reporting for this thread 1: enables interrupt reporting for this thread	R/W	0
2	CIM_T2	Cache Interrupt Mask thread 2 0: disables interrupt reporting for this thread 1: enables interrupt reporting for this thread	R/W	0
1	CIM_T1	Cache Interrupt Mask thread 1 0: disables interrupt reporting for this thread 1: enables interrupt reporting for this thread	R/W	0
0	CIM_T0	Cache Interrupt Mask thread 0 0: disables interrupt reporting for this thread 1: enables interrupt reporting for this thread	R/W	0

This register identifies causes of an interrupt to the PIC that is cache-related.

If a uTLB multi-hit or data poisoned or single bit error occurred, then the cache set and cache index are logged here. Upon a cache interrupt, the programmer must read the EIRR register and the appropriate Px bit here, then clear the active Px bit

4.4.3.31 MMU_SETUP**Block ID: 4 Offset: 0 Address: 0x400**

63:4	3	2:1	0		
Bits	Name	Description		R/W	Reset
63:4	Reserved	<i>Reserved.</i>			
3	LowPowerDisable MMU	If 1, clock gating for the MMU is disabled, i.e. MMU clocks are free-running regardless of activity. 0: Enable clock gating 1: Disable clock gating		R/W	0
2:1	MMUThreadMode	MMU Thread Mode 00: Single thread active; mainTLB, RAS(in IFU) partitioned accordingly 01: Single thread active; mainTLB, RAS(in IFU) partitioned accordingly 10: 2 threads active; mainTLB, RAS(in IFU) partitioned accordingly 11: 4 threads active; mainTLB, RAS(in IFU) partitioned accordingly		R/W	0
0	MMUGlobalMode	MMU Global Mode 0: Partitioned mainTLB, i.e. for writes, entries are equally partitioned between threads based on MMU_Thread_mode setting. For translation, each thread only CAM's its own entries. Partitioned IuTLB/DuTLB, i.e. all IuTLB and DuTLB entries available to all threads for writes, but for translation, each thread only CAM's its own entries. 1: All mainTLB entries available to all threads for writes, all mainTLB entries CAM'ed by all threads during translation, all IuTLB/DuTLB entries available to all threads for writes, all IuTLB/DuTLB entries CAM'ed by all threads during translation.		R/W	0

4.4.3.32 PRF_SMP_EVENT

Block ID: 5 Offset: 0 Address: 0x500

See [Section 25.7, “Instruction Statistical Sampling”](#) for information on the use of this register.

	63:16	15:11	10	9	8	7	6	5	4	3	2	1	0
RSVD		iter Cnt	instr Excep	instr Retired	jump Target Mispredict	br Resolved Not Taken	br Resolved Taken	brPred Not Taken	br Pred Taken	data Cache Miss	data Utlb Miss	instr Cache Miss	instr Utlb Miss

Bits	Name	Description	R/W	Reset
63:16	Reserved	Reserved.		
15:11	iterCnt	Number of times the sampled instruction passed the MEM stage before retirement.	R/W	0
10	instrExcep	Sampled instruction took an exception.	R/W	0
9	instrRetired	Sampled instruction retired.	R/W	0
8	jumpTargetMispredict	Sampled instruction suffered a jump-target mispredict.	R/W	0
7	brResolvedNotTaken	Sampled instruction was a branch that resolved not-taken.	R/W	0
6	brResolvedTaken	Sampled instruction was a branch that resolved taken.	R/W	0
5	brPredNotTaken	Sampled instruction was a predicted-not-taken branch.	R/W	0
4	brPredTaken	Sampled instruction was a predicted-taken branch.	R/W	0
3	dataCacheMiss	Sampled instruction took an D-cache miss.	R/W	0
2	dataUtlbMiss	Sampled instruction took an DuTLB miss.	R/W	0
1	instrCacheMiss	Sampled instruction took an I-cache miss.	R/W	0
0	instrUtlbMiss	Sampled instruction took an IuTLB miss.	R/W	0

4.4.3.33 PRF_SMP_RPLY_BUF

Block ID: 5 Offset: 1 Address: 0x501

63:7	6:3	2:1	0
Reserved	pipetag	thread_id	entry_valid

Bits	Name	Description	R/W	Reset
63:7	Reserved	Reserved.		
6:3	pipetag	Pipetag of sampled instruction that is in the replay buffer.	RO	n/a
2:1	thread_id	Thread-id of sampled instruction that is in the replay buffer.	RO	n/a
0	entry_valid	Sampled instruction is in the replay buffer.	RO	0

NETLOGIC
CONFIDENTIAL



Chapter 5 XLS CPU Instruction Set

5.1 Introduction

The XLS Processor is a highly integrated multi-core microprocessor implementing a superset of the MIPS64 Instruction Set Architecture (ISA). Each CPU instruction consists of a single 32-bit word aligned on a word boundary. There are three instruction formats:

- Immediate (I-type)
- Jump (J-type)
- Register (R-type)

Additional coprocessor CP2 instructions have been added to support the special data movement requirements of very high-throughput processors.

5.2 XLS Processor Instruction Set Overview

The instructions available in the XLS Processor include:

- [MIPS64 CPU Instruction Summary](#)
- [XLS Processor MIPS64 Extensions](#) (page 165)

Note: The Floating Point registers (CP1) are not implemented in this design

5.3 MIPS64 Instruction Set

The detailed descriptions of the MIPS64 CPU and CP2 instructions are available in:

"MIPS64(r) Architecture For Programmers Volume II: The MIPS64(r) Instruction Set",
Document Number: MD00087, Revision 2.50, July 1, 2005

The detailed descriptions of the special XLS Processor CP2 instructions are given in the following sections.

The MIPS64 instruction set is given in [Table 5-1](#).

Table 5-1. MIPS64 CPU Instruction Summary

Mnemonic	Instruction
CPU Arithmetic Instructions	
ADD	Add Word
ADDI	Add Immediate Word
ADDIU	Add Immediate Unsigned Word
ADDU	Add Unsigned Word
CLO	Count Leading Ones in Word
CLZ	Count Leading Zeros in Word
DADD	Doubleword Add
DADDI	Doubleword Add immediate

Table 5-1. MIPS64 CPU Instruction Summary (continued)

Mnemonic	Instruction
DADDIU	Doubleword Add Immediate Unsigned
DADDU	Doubleword Add Unsigned
DCLO	Count Leading Ones in Doubleword
DCLZ	Count Leading Zeros in Doubleword
DDIV	Doubleword Divide
DDIVU	Doubleword Divide Unsigned
DIV	Divide Word
DIVU	Divide Unsigned Word
DMULT	Doubleword Multiply
DMULTU	Doubleword Multiply Unsigned
DSUB	Doubleword Subtract
DSUBU	Doubleword Subtract Unsigned
MADD	Multiply and Add Word to Hi, Lo
MADDU	Multiply and Add Unsigned Word to Hi, Lo
MSUB	Multiply and Subtract Word to Hi, Lo
MSUBU	Multiply and Subtract Unsigned Word to Hi, Lo
MUL	Multiply Word to GPR
MULT	Multiply Word
MULTU	Multiply Unsigned Word
SLT	Set on Less Than
SLTI	Set on Less Than Immediate
SLTIU	Set on Less Than Immediate Unsigned
SLTU	Set on Less Than Unsigned
SUB	Subtract Word
SUBU	Subtract Unsigned Word
CPU Branch and Jump Instructions	
B	Unconditional Branch
BAL	Branch and Link
BEQ	Branch on Equal
BGEZ	Branch on Greater Than or Equal to Zero
BGEZAL	Branch on Greater Than or Equal to Zero and Link
BGTZ	Branch on Greater Than Zero
BLEZ	Branch on Less Than or Equal to Zero
BLTZ	Branch on Less Than Zero
BLTZAL	Branch on Less Than Zero and Link
BNE	Branch on Not Equal
J	Jump
JAL	Jump and Link
JALR	Jump and Link Register
JR	Jump Register
CPU Instruction Control Instructions	
NOP	No Operation

Table 5-1. MIPS64 CPU Instruction Summary (continued)

Mnemonic	Instruction
SSNOP	Superscalar No Operation
CPU Load, Store, and Memory Control Instructions	
LB	Load Byte
LBU	Load Byte Unsigned
LD	Load Doubleword
LDL	Load Doubleword Left
LDR	Load Doubleword Right
LH	Load Halfword
LHU	Load Halfword Unsigned
LL	Load Linked Word
LLD	Load Linked Doubleword
LW	Load Word
LWL	Load Word Left
LWR	Load Word Right
LWU	Load Word Unsigned
PREF	Prefetch
SB	Store Byte
SC	Store Conditional Word
SCD	Store Conditional Doubleword
SD	Store Doubleword
SDL	Store Doubleword Left
SDR	Store Doubleword Right
SH	Store Halfword
SW	Store Word
SWL	Store Word Left
SWR	Store Word Right
SYNC	Synchronize Shared Memory
CPU Logical Instructions	
AND	And
ANDI	And Immediate
LUI	Load Upper Immediate
NOR	Not Or
OR	Or
ORI	Or Immediate
XOR	Exclusive Or
XORI	Exclusive Or Immediate
CPU Move Instructions	
MFHI	Move From HI Register
MFLO	Move From LO Register
MOVN	Move Conditional on Not Zero
MOVZ	Move Conditional on Zero
MTHI	Move To HI Register

Table 5-1. MIPS64 CPU Instruction Summary (continued)

Mnemonic	Instruction
MTLO	Move To LO Register
CPU Shift Instructions	
DSLL	Doubleword Shift Left Logical
DSLL32	Doubleword Shift Left Logical Plus 32
DSLLV	Doubleword Shift Left Logical Variable
DSRA	Doubleword Shift Right Arithmetic
DSRA32	Doubleword Shift Right Arithmetic Plus 32
DSRAV	Doubleword Shift Right Arithmetic Variable
DSRL	Doubleword Shift Right Logical
DSRL32	Doubleword Shift Right Logical Plus 32
DSRLV	Doubleword Shift Right Logical Variable
SLL	Shift Word Left Logical
SLLV	Shift Word Left Logical Variable
SRA	Shift Word Right Arithmetic
SRAV	Shift Word Right Arithmetic Variable
SRL	Shift Word Right Logical
SRLV	Shift Word Right Logical Variable
CPU Trap Instructions	
BREAK	Breakpoint
SYSCALL	System Call
TEQ	Trap if Equal
TEQI	Trap if Equal Immediate
TGE	Trap if Greater or Equal
TGEI	Trap if Greater of Equal Immediate
TGEIU	Trap if Greater or Equal Immediate Unsigned
TGEU	Trap if Greater or Equal Unsigned
TLT	Trap if Less Than
TLTI	Trap if Less Than Immediate
TLTIU	Trap if Less Than Immediate Unsigned
TLTU	Trap if Less Than Unsigned
TNE	Trap if Not Equal
TNEI	Trap if Not Equal Immediate
Obsolete CPU Branch Instructions	
BEQL	Branch on Equal Likely
BGEZALL	Branch on Greater Than or Equal to Zero and Link Likely
BGEZL	Branch on Greater Than or Equal to Zero Likely
BGTZL	Branch on Greater Than Zero Likely
BLEZL	Branch on Less Than or Equal to Zero Likely
BLTZALL	Branch on Less Than Zero and Link Likely
BLTZL	Branch on Less Than Zero Likely
BNEL	Branch on Not Equal Likely

Table 5-1. MIPS64 CPU Instruction Summary (continued)

Mnemonic	Instruction
Privileged Instructions	
CACHE	Perform Cache Operation
DMFC0	Doubleword Move from Coprocessor 0
DMTC0	Doubleword Move to Coprocessor 0
ERET	Exception Return
MFC0	Move from Coprocessor 0
MTC0	Move to Coprocessor 0
TLBP	Probe TLB for Matching Entry
TLBR	Read Indexed TLB Entry
TLBWI	Write Indexed TLB Entry
TLBWR	Write Random TLB Entry
WAIT	Enter Standby Mode
CP2 Move Instructions	
DMFC2	Doubleword Move from Coprocessor 2
DMTC2	Doubleword Move to Coprocessor 2
MFC2	Move Word from Coprocessor 2
MTC2	Move Word to Coprocessor 2
EJTAG Instructions	
DERET	Debug Exception Return
SDBBP	Software Debug Breakpoint
CP2 Messaging Instructions	
MSG SND	Send Message Command
MSG LD	Load Message Command
MSG WAIT	Wait for message in the Receive Q

Table 5-2. XLS Processor MIPS64 Extensions

Mnemonic	Instruction
LDADDW	Load and Add Word
LDADDWU	Load and Add Word Unsigned
LDADDD	Load and Add Double
SWAPW	Swap Word
SWAPWU	Swap Word Unsigned
SWAPD	Swap Double
DADDWC	DoubleWord Add with Carry
MFCR	Move From Control Register
MTCR	Move To Control Register

5.3.1 CACHE - Perform Cache Operation

31	26	25	21	20	16	15	0
CACHE 101111		base 5		op 5		offset 16	

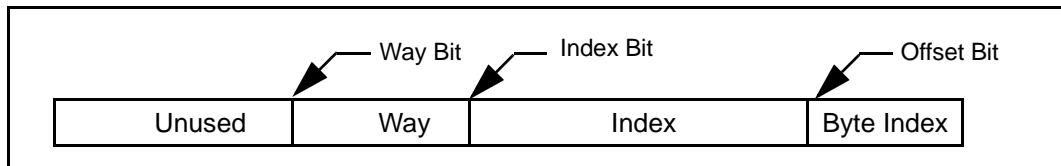
Format: CACHE op, offset(base) MIPS32

Purpose: To perform the cache operation specified by op.

Description: The 16-bit offset is sign-extended and added to the contents of the base register to form an effective address. The effective address is used in one of the following ways, based on the operation to be performed and the type of cache, as described in the following table.

Table 5-3. Usage of Effective Address

Operation Requires an:	Type of Cache	Usage of Effective Address
Address	Virtual	The effective address is used to address the cache. An address translation may or may not be performed on the effective address (with the possibility that a TLB Refill or TLB Invalid exception might occur)
Address	Physical	The effective address is translated by the MMU to a physical address. The physical address is then used to address the cache
Index	N/A	<p>The effective address is translated by the MMU to a physical address. It is implementation dependent whether the effective address or the translated physical address is used to index the cache. As such, a kseg0 address should always be used for cache operations that require an index. See the Programming Notes section below.</p> <p>Assuming that the total cache size in bytes is CS, the associativity is A, and the number of bytes per tag is BPT, the following calculations give the fields of the address which specify the way and the index:</p> <ul style="list-style-type: none"> • OffsetBit Log2(BPT) • IndexBit Log2(CS / A) • WayBit IndexBit + Ceiling(Log2(A)) • Way AddrWayBit-1..IndexBit • Index AddrIndexBit-1..OffsetBit <p>For a direct-mapped cache, the Way calculation is ignored and the Index value fully specifies the cache tag. This is shown symbolically in the figure below.</p>

Figure 5-1. Usage of Address Fields to Select Index and Way

A TLB Refill and TLB Invalid (both with cause code equal TLBL) exception can occur on any operation. For index operations (where the address is used to index the cache but need not match the cache tag) software should use unmapped addresses to avoid TLB exceptions. This instruction never causes TLB Modified exceptions nor TLB Refill exceptions with a cause code of TLBS.

The effective address may be an arbitrarily-aligned by address. The CACHE instruction never causes an Address Error Exception due to a non-aligned address.

A Cache Error exception may occur as a by-product of some operations performed by this instruction. For example, if a Writeback operation detects a cache or bus error during the processing of the operation, that error is reported via a Cache Error exception. Similarly, a Bus Error Exception may occur if a bus operation invoked by this instruction is terminated in an error. However, cache error exceptions must not be triggered by an Index Load Tag or Index Store tag operation, as these operations are used for initialization and diagnostic purposes.

An Address Error Exception (with cause code equal AdEL) may occur if the effective address references a portion of the kernel address space which would normally result in such an exception. It is implementation dependent whether such an exception does occur.

It is implementation dependent whether a data watch is triggered by a cache instruction whose address matches the Watch register address match conditions.

Bits [17:16] of the instruction specify the cache on which to perform the operation, as follows:

Table 5-4. Encoding of Bits[17:16] of CACHE Instruction

Code	Name	Cache
0b00	1	Primary Instruction
0b01	D	Primary Data or Unified Primary

Bits [20:18] of the instruction specify the operation to perform. To provide software with a consistent base of cache operations, certain encodings must be supported on all processors. The remaining encodings are recommended.

Table 5-5. Encoding of Bits [20:18] of the CACHE Instruction

Code	Caches	Name	Effective Address Operand Type	Operation
0b000	I	Index Invalidate	Index	Set the state of the cache block at the specified index to invalid. This required encoding may be used by software to invalidate the entire instruction cache by stepping through all valid indices.
	D	Index Writeback Invalidate / Index Invalidate	Index	For a write-back cache: If the state of the cache block at the specified index is valid and dirty, write the block back to the memory address specified by the cache tag. After that operation is completed, set the state of the cache block to invalid. If the block is valid but not dirty, set the state of the block to invalid. For a write-through cache: Set the state of the cache block at the specified index to invalid. This required encoding may be used by software to invalidate the entire data cache by stepping through all valid indices. Note that Index Store Tag should be used to initialize the cache at powerup.
0b001		Reserved		Reserved
0b010	All	Index Store Tag	Index	Write the tag for the cache block at the specified index from the <i>TagLo</i> and <i>TagHi</i> Coprocessor 0 registers. This operation must not cause a Cache Error Exception. This required encoding may be used by software to initialize the entire instruction or data caches by stepping through all valid indices. Doing so requires that the <i>TagLo</i> and <i>TagHi</i> registers associated with the cache be initialized first.
0b011		Reserved		Reserved
0b100	I, D	Hit Invalidate	Address	If the cache block contains the specified address, set the state of the cache block to invalid. This required encoding may be used by software to invalidate a range of addresses from the instruction cache by stepping through the address range by the line size of the cache
0b101	D	Hit Writeback Invalidate / Hit Invalidate	Address	For a write-back cache: If the cache block contains the specified address and it is valid and dirty, write the contents back to memory. After that operation is completed, set the state of the cache block to invalid. If the block is valid but not dirty, set the state of the block to invalid. For a write-through cache: If the cache block contains the specified address, set the state of the cache block to invalid. This required encoding may be used by software to invalidate a range of addresses from the data cache by stepping through the address range by the line size of the cache.

Table 5-5. Encoding of Bits [20:18] of the CACHE Instruction (continued)

Code	Caches	Name	Effective Address Operand Type	Operation
0b110		Reserved		
0b111	I, D	Fetch and Lock	Address	<p>If the cache does not contain the specified address, fill it from memory, performing a writeback if required, and set the state to valid and locked. If the cache already contains the specified address, set the state to locked. In set-associative or fully-associative caches, the way selected on a fill from memory is implementation dependent. The lock state may be cleared by executing an Index Invalidate, Index Writeback Invalidate, Hit Invalidate, or Hit Writeback Invalidate operation to the locked line, or via an Index Store Tag operation to the line that clears the lock bit. Note that clearing the lock state via Index Store Tag is dependent on the implementation-dependent cache tag and cache line organization, and that Index and Index Writeback Invalidate operations are dependent on cache line organization. Only Hit and Hit Writeback Invalidate operations are generally portable across implementations.</p> <p>It is implementation dependent whether a locked line is displaced as the result of an external invalidate or intervention that hits on the locked line. Software must not depend on the locked line remaining in the cache if an external invalidate or intervention would invalidate the line if it were not locked.</p> <p>It is implementation dependent whether a Fetch and Lock operation affects more than one line. For example, more than one line around the referenced address may be fetched and locked. It is recommended that only the single line containing the referenced address be affected.</p>

5.3.2 PREFETCH

31	26	25	21	20	16	15	0
PREF 110011 6		base 5		hint 5		offset 16	

Format: PREF hint, offset(base)

MIPS32

Purpose: To move data between memory and cache

Description: Prefetch memory(GPR[base] + offset)

PREF adds the 16-bit signed *offset* to the contents of GPR *base* to form an effective byte address. The *hint* field supplies information about the way that the data is expected to be used.

PREF enables the processor to take some action, typically causing data to be moved to or from the cache, to improve program performance. The action taken for a specific PREF instruction is both system and context dependent. Any action, including doing nothing, is permitted as long as it does not change architecturally visible state or alter the meaning of a program. Implementations are expected either to do nothing, or to take an action that increases the performance of the program. The PrepareForStore function is unique in that it may modify the architecturally visible state.

PREF does not cause addressing-related exceptions, including TLB exceptions. If the address specified would cause an addressing exception, the exception condition is ignored and no data movement occurs. However even if no data is moved, some action that is not architecturally visible, such as writeback of a dirty cache line, can take place.

Cache Error exception is reported if such an error is detected as a by product of the action taken by the PREF instruction.

PREF neither generates a memory operation nor modifies the state of a cache line for a location with an *uncached* memory access type, whether this type is specified by the address segment (e.g., kseg1), the programmed coherency attribute of a segment (e.g., the use of the K0, KU, or K23 fields in the *Config* register), or the per-page coherency attribute provided by the TLB.

If PREF results in a memory operation, the memory access type and coherency attribute used for the operation are determined by the memory access type and coherency attribute of the effective address, just as it would be if the memory operation had been caused by a load or store to the effective address.

For a cached location, the expected and useful action for the processor is to prefetch a block of data that includes the effective address. The size of the block and the level of the memory hierarchy it is fetched into are implementation specific.

Table 5-6. Values of the hint Field for the PREF Instruction

Value	Name	Data Use and Desired Prefetch Action
0 or 4 or 6	load	Use: Prefetched data is expected to be read (not modified). Action: Fetch data as if for a load.
1 or 5 or 7	store	Use: Prefetched data is expected to be stored or modified. Action: Fetch data as if for a store.
2 or 3	Reserved	Reserved for future use
8 to 24	Reserved	Reserved for future use

Table 5-6. Values of the hint Field for the PREF Instruction (continued)

Value	Name	Data Use and Desired Prefetch Action
25	writeback_invalidate (also known as “nudge”)	Use: Data is no longer expected to be used. Action: For a writeback cache, schedule a writeback of any dirty data. At the completion of the writeback, mark the state of any cache lines written back as invalid. If the cache line is not dirty, its implementation dependent whether the state of the cache line is marked invalid or left unchanged. If the cache line is locked, no action is taken.
26 to 29	Implementation Dependent	Unassigned by the Architecture - available for implementation-dependent use.
30	PrepareForStore	Use: Prepare the cache for writing an entire line, without the overhead involved in filling the line from memory. Action: If the reference hits in the cache, no action is taken. If the reference misses in the cache, the following actions are taken: <ul style="list-style-type: none"> • A line is selected for replacement, and • Any valid and dirty victim line is written back to memory, and • The entire line is filled with zero data, and • The state of the line is marked as valid and dirty. Programming Note: Although the cache line is filled with zero data on a cache miss, software must not assume that this action, in and of itself, can be used as a fast bzero-type function (a function used to define a number of zero bytes).
31	Implementation Dependent	Unassigned by the Architecture - available for implementation-dependent use.

Restrictions: None**Operation:**

```

vAddr    GPR[base] + sign_extend(offset)
(pAddr, CCA)  AddressTranslation(vAddr, DATA, LOAD)
Prefetch(CCA, pAddr, vAddr, DATA, hint)

```

Exceptions: Bus Error, Cache Error

Prefetch does not take any TLB-related or address-related exceptions under any circumstances.

Programming Notes:

Prefetch cannot move data to or from a mapped location unless the translation for that location is present in the TLB. Locations in memory pages that have not been accessed recently may not have translations in the TLB, so prefetch may not be effective for such locations.

Prefetch does not cause addressing exceptions. A prefetch may be used using an address pointer before the validity of the pointer is determined without worrying about an addressing exception.

It is implementation dependent whether a Bus Error or Cache Error exception is reported if such an error is detected as a by-product of the action taken by the PREF instruction. Typically, this only occurs in systems which have high-reliability requirements.

Prefetch operations have no effect on cache lines that were previously locked with the CACHE instruction.

Hint field encodings whose function is described as “streamed” or “retained” convey usage intent from software to hardware. Software should not assume that hardware will always prefetch data in an optimal way. If data is to be truly retained, software should use the Cache instruction to lock data into the cache.

5.3.3 DMFC2 — Doubleword Move from Coprocessor 2

31	26	25	21	20	16	15	11	10	3	0
COP2 b010010	DMF b00001		rt		rd		b00000000		sel	
6	5	5	5	5			8			

Format: DMFC2 rt, rd **MIPS64**
DMFC2 rt, rd, sel **MIPS64**

Purpose: To move a doubleword from a Coprocessor 2 register to a GPR.
Description: $\text{GPR}[\text{rt}] \leftarrow \text{CP2CPR}[2, \text{rd}, \text{sel}]$
The contents of the Coprocessor 2 register denoted by the rd field is loaded into GPR rt . The interpretation of the rd field is left entirely to the Coprocessor 2 implementation and is not specified by the architecture.

Restrictions: The results are **UNPREDICTABLE** if rd specifies a Coprocessor 2 register that does not exist, or if the Coprocessor 2 register specified by rd and sel is a 32-bit register.

Operation: $\text{datadoubleword} \leftarrow \text{CP2CPR}[2, \text{rd}, \text{sel}]$
 $\text{GPR}[\text{rt}] \leftarrow \text{datadoubleword}$

Exceptions: Coprocessor Unusable
Reserved Instruction

Portions of the preceding information reproduced under license from the MIPS64™ Architecture for Programmers Volume II. Copyright © MIPS Technologies Inc. All rights reserved.

5.3.4 DMTC2 — Doubleword Move to Coprocessor 2

31	26	25	21	20	16	15	11	10	3	0
COP2 b010000	DMT b00101		rt		rd		b00000000		sel	
6	5	5	5	5	8					

Format:	DMTC2 rt, rd	MIPS64
	DMTC2 rt, rd, sel	MIPS64

Purpose: To move a doubleword from a GPR to a Coprocessor 2 register.

Description: $CPR[2, \text{rd}, \text{sel}] \leftarrow GPR[\text{rt}]$

The contents GPR rt are loaded into the Coprocessor 2 register denoted by the rd field. The interpretation of the rd field is left entirely to the Coprocessor 2 implementation and is not specified by the architecture.

Restrictions: The results are **UNPREDICTABLE** if `rd` specifies a Coprocessor 2 register that does not exist, or if the Coprocessor 2 register specified by `rd` and `sel` is a 32-bit register.

Operation:

```
datadoubleword ← GPR[rt]  
CP2CPR[0,rd,sel] ← datadoubleword
```

Exceptions: Coprocessor Unusable

Reserved Instruction

5.3.5 MFC2 — Move Word From Coprocessor 2

31	26	25	21	20	16	15	11	10	3	0
COP2 b010010	MF b00000		rt		rd		b00000000		sel	
6	5	5	5	5			8			

Format: MFC2 rt, rd **MIPS32**
MFC2 rt, rd, sel **MIPS32**

Purpose: To copy a word from a COP2 general register to a GPR.
Description: $\text{GPR}[\text{rt}] \leftarrow \text{CP2CPR}[0, \text{rd}, \text{sel}]$

The contents of the Coprocessor 2 register denoted by the `rd` field are sign-extended and placed into general register `rt`. The interpretation of the `rd` field is left entirely to the Coprocessor 2 implementation and is not specified by the architecture.

Restrictions: The results are **UNPREDICTABLE** if `rd` and `sel` specify a Coprocessor 2 register that does not exist.

Operation:

```
data ← CPR[0,rd,sel]31..0
GPR[rt] ← sign extend(data)
```

Exceptions: Coprocessor Unusable

5.3.6 MTC2 — Move Word to Coprocessor 2

31	26	25	21	20	16	15	11	10	3	0
COP2 b010010	MT b00100		rt		rd		b00000000		sel	

Format:	MTC2 rt, rd	MIPS32
	MTC2 rt, rd, sel	MIPS32

Purpose: To copy a word from a GPR to a COP2 general register.

Description: CP2CPR[0,rd,sel] \leftarrow GPR[rt]

The low word in GPR `rt` is placed into the low word of Coprocessor 2 general register denoted by the `rd` field. If Coprocessor 2 general registers are 64 bits wide, bits[63:32] of the register denoted by the `rd` field become undefined. The interpretation of the `rd` field is left entirely to the Coprocessor 2 implementation and is not specified by the architecture.

Restrictions: The results are **UNPREDICTABLE** if *rd* specifies a Coprocessor 2 register that does not exist.

Operation: $\text{data} \leftarrow \text{data}_{31..0}$
 $\text{CP2CPR}[0, \text{rd}, \text{sel}] \leftarrow \text{data}$

Exceptions:

Coprocessor Unusable

Reserved Instruction

5.3.7 **MSG SND** — Send Message Command

31	26	25	24	21	20	16	15	6	5	0
COP2 010010	1		0 0000		rt		0 00 0000 0000		MSG SND 000001	
6	1	4		5		10		6		

Format: MSGSND rt

Purpose: To send a message

MIPS64

Purpose: To send a message

Description: send message \rightarrow messaging block

Dest_Id \leftarrow rt[6:0]

Dest SW Code \leftarrow rt[15:8]

```
SizeMinus1 ← rt[17:16]
```

On receiving the Send Msg Command, two conditions are checked:

1. No Send Message Pending for the same thread.
 2. The Destination Receive FIFO has enough space.

If the conditions are met, the command is accepted and the Pending bit is set in the MsgStatus Register. Otherwise it fails, and the relevant bits are set in the MsgStatus Register. On being accepted, the messaging block will try to schedule the transfer based on space being available in the Transmit FIFO, and Round Robin scheduling across other threads.

On the completion of the MsgSnd operation, the Pending Bit in the MsgStatus (COP2) register will be updated.

Restrictions: COP2 must be enabled.

It is recommended that a sync instruction be executed before MSGSND is sent. This ensures that data stored by the CPU sending the message becomes visible to the recipient of the message.

Exceptions: Coprocessor Unusable, Reserved Instruction, TLB Refill, TLB Invalid, TLB Modified, Address Error, Watch

5.3.8 MSGLD — Load Message Command

31	26	25	24	21	20	16	15	6	5	0
COP2 010010	1		0 0000		rt		0 00 0000 0000		MSGLD 000010	

6 1 4 5 10 6

Format: MSGLD rt

MIPS64

Purpose: To load a message

Description: messaging_block \leftarrow load_message;

bucket \leftarrow rt[2:0];

Bits 2:0 specify Bucket 0–7. GPR rt[7:0] specifies the particular bucket, 00000xxx. A value of 11111111 specifies highest priority (0–7) non-empty bucket.

On receiving the Load Message Command, the following conditions are checked:

1. No Load Message Pending for the same thread
2. Receive FIFO Bucket not empty.

If the conditions are met, the command is accepted. Otherwise it fails. On being accepted, the messaging block will try to schedule the transfer based on a Round Robin scheduling across other threads.

Note that by the time the Load Message is scheduled, the Receive FIFO may have been emptied by previously scheduled loads, and the Load Message can still fail.

On the completion of the transfer from the appropriate Bucket of the receive FIFO to the Receive Buffer, the MsgStatus (COP2) register will be updated.

Restrictions: COP2 must be enabled.

Exceptions: Coprocessor Unusable, Reserved Instruction, TLB Refill, TLB Invalid, TLB Modified, Address Error, Watch

5.3.9 MSGWAIT — Wait for Message in the Receive Queue

31	26	25	24	21	20	16	15	6	5	0
COP2 010010	1		0 0000		rt		0 00 0000 0000		MSWAIT 000011	6

6 1 4 5 10 6

Format: MSGWAIT rt

MIPS64

Purpose: To wait for Receive Queue Bucket to be not empty.

Description: On executing this instruction, the thread goes to sleep. The lower 8-bit vector in GPR rt specifies the buckets for which the thread should wait. If the 8-bit vector is all zeros, the MSGWAIT instruction will behave like a regular WAIT instruction.

The sleep state will be cleared when any Bucket specified by the 8-bit vector of the Messaging Block Receive Queue is not empty. Thus, if this instruction is executed when the Receive Queue Bucket is not empty, nothing will happen.

All threads in a CPU waiting on a message for that particular bucket will be woken up.

Restrictions. COP2 must be enabled.

Exceptions: Coprocessor Unusable, Reserved Instruction, TLB Refill, TLB Invalid, TLB Modified, Address Error, Watch

The XLS MSGWAIT instruction may incorrectly use a stale bucket number when executed after instructions that incur a delay prior to the loading of the desired bucket number. The data dependency between the load instruction and the MsgWait instruction should hold off the execution of MsgWait instruction until the most recent bucket number becomes available in Register Rx due to the preceding load operation to the same register Rx. However, the XLS Processor under some cases may not honor the data dependency correctly. To guarantee completion of the proper register load prior to the MsgWait instruction execution, an extra step is added to the Load - MsgWait sequence:

- Load, RegX, offset(RegY) -> Loading bucket number into register RegX from memory address at offset of RegY
- Addu RegX, RegX, 0 -> Ensure that register RegX contains the desired bucket number before MsgWait instruction is executed
- MsgWait, RegX -> Waiting on the bucket number contained in register RegX

5.3.10 LDADDW — Load and Add Word

31	26	25	21	20	16	15	6	5	0
SPECIAL2 011100		base		rt		0 00 0000 0000		LDADDW 010000	

6 5 5 10 6

Format: LDADDW rt, base **MIPS32**

Purpose: To load and then add a word in GPR rt to a value in memory.

Description: $v \leftarrow rt; rt \leftarrow \text{memory}[base]; \text{memory}[base] \leftarrow \text{memory}[base] + v;$

The value of GPR rt is saved in v. The contents of the 32-bit word at the memory location specified by the aligned effective address are fetched and replace the value of GPR rt. The contents of GPR base forms the effective address (there is no offset). The signed word value of the effective address is replaced with the value of itself plus the signed value v.

Restrictions: The effective address must be naturally-aligned. If any of the 2 least-significant bits of the address is non-zero, an Address Error exception occurs.

If either GPR rt or memory[base] does not contain sign-extended 32-bit values (bits [63:31] equal), then the result of the operation is UNPREDICTABLE.

Operation:

```

ADD_VALUE = GPR[RT];
GPR[RT]   = MEM[BASE];                                // No Base + Offset
MEM[BASE] = MEM[BASE] + ADD_VALUE; // signed addition

```

Exceptions: TLB Refill, TLB Invalid, Bus Error, Address Error, Watch, TLB Modified.

5.3.11 LDADDWU — Load and Add Word Unsigned

31	26	25	21	20	16	15	6	5	0
SPECIAL2 011100	base		rt		0 00 0000 0000		LDADDWU 010001		
6	5		5		10		6		

Format: LDADDWU rt, base

MIPS64

Purpose: To load and then add an unsigned word in GPR rt to an unsigned value in memory.

Description: $v \leftarrow rt; rt \leftarrow \text{memory}[base]; \text{memory}[base] \leftarrow \text{memory}[base] + v;$

The value of GPR rt is saved in v. The contents of the 32-bit word at the memory location specified by the aligned effective address are fetched and replace the value of GPR rt. The contents of GPR base forms the effective address (there is no offset). The unsigned word value of the effective address is replaced with the value of itself plus unsigned word v.

Restrictions: The effective address must be naturally-aligned. If any of the 2 least-significant bits of the address is non-zero, an Address Error exception occurs.

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

```

ADD_VALUE = GPR[RT];
GPR[RT]   = MEM[BASE];           // No Base + Offset
MEM[BASE] = MEM[BASE] + ADD_VALUE; // unsigned addition

```

5.3.12 LDADDD — Load and Add Double

31	26	25	21	20	16	15	6	5	0
SPECIAL2 011100		base		rt		0 00 0000 0000		LDADDD 010010	

6 5 5 10 6

Format: LDADDDW rt, base **MIPS64**

Purpose: To load and then add a doubleword in GPR rt to a value in memory.

Description: $v \leftarrow rt; rt \leftarrow \text{memory}[base]; \text{memory}[base] \leftarrow \text{memory}[base] + v;$

The value of GPR rt is saved in v. The contents of the 64-bit double-word at the memory location specified by the aligned effective address are fetched and replace the value of GPR rt. The contents of GPR base forms the effective address (there is no offset). The doubleword signed value of the effective address is replaced with the value of itself plus signed doubleword v to form a 64-bit result.

If the addition results in 64-bit 2's complement arithmetic overflow, then the destination is not modified and an Integer Overflow exception occurs. If the addition does not overflow, the 64-bit result is placed into memory[base].

Restrictions: The effective address must be naturally-aligned. If any of the 3 least-significant bits of the address is non-zero, an Address Error exception occurs.

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

```

ADD_VALUE = GPR[RT];
GPR[RT]   = MEM[BASE];                                // No Base + Offset
MEM[BASE] = MEM[BASE] + ADD_VALUE; // signed doubleword addition

```

Exceptions: TLB Refill, TLB Invalid, TLB Modified, Bus Error, Address Error, Reserved Instruction, Watch.

5.3.13 SWAPW — Swap Word

31	26	25	21	20	16	15	6	5	0
SPECIAL2 011100	base		rt		0 00 0000 0000		SWAPW 010100		
6	5		5		10		6		

Format: SWAPW rt, base

MIPS32

Purpose: To swap a signed word value in GPR rt with a signed word value in memory.

Description: $v \leftarrow rt; rt \leftarrow \text{memory}[base]; \text{memory}[base] \leftarrow v;$

The value of GPR rt is saved in v. The contents of the 32-bit signed word at the memory location specified by the aligned effective address are fetched and replace the value of GPR rt. The contents of GPR base forms the effective address (there is no offset). The signed word value of the effective address is replaced with the signed word value of v.

Restrictions: The effective address must be naturally-aligned. If any of the 2 least-significant bits of the address is non-zero, an Address Error exception occurs.

Operation:

```
SWAP_VALUE = GPR[RT];
GPR[RT]    = MEM[BASE];
MEM[BASE]   = SWAP_VALUE;
```

Exceptions: TLB Refill, TLB Invalid, Bus Error, Address Error, Watch, TLB Modified.

5.3.14 SWAPWU — Swap Word Unsigned

31	26	25	21	20	16	15		6	5	0
SPECIAL2 011100		base		rt			0 00 0000 0000		SWAPWU 010101	

6 5 5 10 6

Format: SWAPWU rt, base **MIPS64**

Purpose: To swap an unsigned word value in GPR rt with an unsigned word value in memory.

Description: $v \leftarrow rt; rt \leftarrow \text{memory}[base]; \text{memory}[base] \leftarrow v;$

The value of GPR rt is saved in v. The contents of the 32-bit unsigned word at the memory location specified by the aligned effective address are fetched and replace the value of GPR rt. The contents of GPR base forms the effective address (there is no offset). The unsigned word value of the effective address is replaced with the unsigned word value of v.

Restrictions: The effective address must be naturally-aligned. If any of the 2 least-significant bits of the address is non-zero, an Address Error exception occurs.

A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

```
SWAP_VALUE = GPR[RT];
GPR[RT]    = MEM[BASE];
MEM[BASE]  = SWAP_VALUE;
```

Exceptions: TLB Refill, TLB Invalid, Bus Error, Address Error, Reserved Instruction, Watch, TLB Modified.

5.3.15 SWAPD — Swap Double

31	26	25	21	20	16	15	6	5	0
SPECIAL2 011100	base		rt		0 00 0000 0000		SWAPD 010110		
6	5		5		10		6		

Format: SWAPD rt, base

MIPS64

Purpose: To swap a signed doubleword value in GPR *rt* with a signed doubleword value in memory.

Description: $v \leftarrow rt; rt \leftarrow \text{memory}[base]; \text{memory}[base] \leftarrow v;$

The value of GPR *rt* is saved in *v*. The contents of the 64-bit signed doubleword at the memory location specified by the aligned effective address are fetched and replace the value of GPR *rt*. The contents of GPR *base* forms the effective address (there is no offset). The signed doubleword value of the effective address is replaced with the signed doubleword value of *v*.

Restrictions: The effective address must be naturally aligned. If any of the 3 least-significant bits of the address is non-zero, an Address Error exception occurs. A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation:

```
SWAP_VALUE = GPR[RT];
GPR[RT]    = MEM[BASE];
MEM[BASE]   = SWAP_VALUE;
```

Exceptions: TLB Refill, TLB Invalid, Bus Error, Address Error, Reserved Instruction, Watch, TLB Modified.

5.3.16 DADDWC — DoubleWord Add with Carry

31	26	25	21	20	16	15	11	10	6	5	0
SPECIAL2 011100		rs		rt		rd		0 00000		DADDWC 111000	

6 5 5 5 5 6

Format: DADDWC rd, rs, rt **MIPS64**

Purpose: To add 64-bit integers. No trap if overflow occurs.

Description: $\{ \text{cryout}, \text{rd} \} \leftarrow \text{rs} + \text{rt} + \text{Status}[16]; \text{Status}[16] \leftarrow \text{cryout};$

This is a double unsigned add (i.e. no overflow exception), where a previously saved carry is used as carry-in, and the newly generated carry-out is saved. The carry flag is stored in Status[16].

The 64-bit double-word value in GPR rt is added to the 64-bit value in GPR rs to produce a 64-bit result. Even if the addition results in 64-bit 2's complement arithmetic overflow, the destination register GPR rd is modified, and no Integer Overflow exception occurs.

Restrictions: A Reserved Instruction Exception is signaled if access to 64-bit operations is not enabled.

Operation: $\{ \text{CARRY_OUT}, \text{GPR[RD]} \} = \text{GPR[RS]} + \text{GPR[RT]} + \text{STATUS[16]};$
 $\text{STATUS[16]} = \text{CARRY_OUT};$

Exceptions: Reserved Instruction.

Programming Notes: Even though the carry is stored in the status register, this is really a user instruction, so software can only indirectly affect the carry bit in User Mode. When COP0 is usable, software can write to this bit by writing to the Status Register.

5.3.17 MFCR — Move From Control Register

31	26	25	21	20	16	15	6	5	0
SPECIAL2 011100	rs		rt			0 0000000000	MFCR 011000		
6	5		5			10	6		

Format: MFCR rt, rs

MIPS64

Purpose: To move an XLS processor control register into a GPR.

Description: $rt \leftarrow CR[[rs]]$

This instruction moves up to 64 bits from the internal XLS control register indexed by the contents of GPR rs into GPR rt. If the control register is less than 64 bits, the result will be zero-extended.

Restrictions: This instruction will cause a Reserved Instruction Exception if COP0 is unusable.

Operation: $RT \leftarrow CR[[RS]]$

Exceptions: Reserved Instruction Exception

5.3.18 MTCR — Move To Control Register

31	26	25	21	20	16	15	6	5	0
SPECIAL2 011100		rs		rt		0 0000000000		MTCR 011001	

6 5 5 10 6

Format: MTCR rt, rs **MIPS64**

Purpose: To move a GPR to an XLS control register.

Description: CR[[RS]] ← RT

This instruction moves up to 64 bits from GPR rt to the internal XLS control register indexed by the contents of GPR rs. If the control register is less than 64 bits, the result will be truncated.

Restrictions: This instruction will cause a Reserved Instruction Exception if COP0 is unusable.

Operation: CR[[RS]] ← RT

Exceptions: Reserved Instruction Exception

NETLOGIC
CONFIDENTIAL



Chapter 6 XLS L1 Cache

6.1 Introduction

Each XLS CPU core contains both a 32-KB Level-1 instruction cache and 32-KB Level-1 data cache.

This chapter provides the following information:

- Overview of L1 cache capabilities and functions
- Programming model
- Control register summary and descriptions

6.2 Capabilities and Functions

Instruction cache and data cache key attributes are listed in [Table 6-1](#).

6.2.1 Key Capabilities Summary

Table 6-1. Primary Cache Parameters

Attribute	Instruction Cache	Data Cache
Size (KBytes)	32	32
Associativity	8-way	8-way
Line size	32 bytes	32 bytes
Line Locking	Yes	Yes
Write policy	N/A	Write-back
Coherency	Software managed	Hardware managed - MOSI
Addressing scheme	Physical index/Physical Tag	Physical index/Physical Tag
Replacement policy	Pseudo-LRU	Pseudo-LRU
Error management on tags	Parity bit per tag	Parity bit per tag
Error management on data	Parity 2 bits per double word	ECC – SECDED Single Error Correction, Double Error Detection - 8 bit per double word

6.2.2 Level-1 Instruction Cache

Each CPU core contains a 32-KB, Level-1 8-way set-associative instruction cache with 32-byte line size. The high degree of set associativity ensures the most efficient use of the processor threads by reducing cache-conflict misses. Both the cache instruction and cache tag arrays are protected by parity. When a parity error is detected, the instruction is discarded, the cache line is invalidated, and a new instruction fetch is executed. This process is accomplished without software intervention.

6.2.3 Level-1 Data Cache

Each CPU core contains a 32-KB, Level-1, 8-way set-associative data cache with 32-byte line size. As with the instruction cache, the high degree of set associativity reduces cache-conflict misses.

6.2.4 ECC Protection

The data cache array is protected by ECC (SECDED – Single Error Correction and Double Error Detection) for correction of single bit errors without software intervention. All multiple-bit errors are detected and then passed to the software via a cache error exception. The cache tag arrays are protected by parity.

6.2.5 DedicatedPlus™ Cache

DedicatedPlus Cache mode allows the L1 cache to be partitioned among the nCPU™ NetLogic virtual CPUs. The partitioning of the cache is accomplished by reserving specific ways for specific nCPUs. For example, software may allocate the available cache as follows.

- nCPU0: ways 0 & 1
- nCPU1: ways 2 & 3
- nCPU2: ways 4 & 5
- nCPU3: ways 6 & 7

Now, whenever nCPU0 needs to replace a cacheline, only ways 0 & 1 will be used. Similarly, all cacheline replacements caused by nCPU3 will use ways 6 & 7. However, all reads from nCPU0/1/2/3 will look up all the 8 ways of the I & D L1-caches, thus, all nCPUs will benefit from the pre-fetching performed by other nCPUs. The DedicatedPlus Cache mode is utilized by writing registers [L1I_WAY_ENABLE](#) and [L1D_WAY_DISABLE](#) registers as described later in this chapter.

6.3 Programming Model

There is one set of cache control registers per CPU core, as well as related cache control functions performed by CPU registers such as the ICU_DEFFEATURE register and the EIRR register.

6.4 L1 Cache Control Registers

This section discusses the L1 cache control registers. The following terms are used in the register descriptions to describe field modifiability type:

Table 6-2. Field Modifiability Codes

Type Code	Meaning	Description
R/O	Read Only	The field contains a static value and cannot be written.
R/W	Read/Write	The field can be modified by software. For single-bit fields, except as noted, 0 disables the related function. 1 enables the related function
R/C	Read/Clear	The field can be read to give status information. Bits may be cleared by writing a 1 to them; writes of 0 are ignored.

A core's L1 cache controller registers are accessed as core registers. They are 64-bit registers (although some bits may be unused). Individual registers are identified according to their hardware blocks, and within each block by their register offsets. Bits [15:8] of the register address specify the Block ID, and bits [7:0] specify the register's offset within the block.

For example, L1 Cache Control Register L1D_CONFIG1 is in block 3 (i.e., 00000011b), with an offset of 1 (i.e., 00000001b). Thus, the register address is 0000 0011 0000 0001, or 0x301.

6.4.1 L1 Cache Control Registers Summary

Table 6-3. L1 Cache Control Registers List

Block ID	Local Offset within Block	[Block ID : Offset] = Register Address	Register Name
1	A	0x10A	L1_WAY_ENABLE
3	0	0x300	L1D_CONFIG0
3	1	0x301	L1D_CONFIG1
3	2	0x302	L1D_CONFIG2
3	3	0x303	L1D_CONFIG3
3	4	0x304	L1D_CONFIG4
3	5	0x305	L1D_STATUS
3	6	0x306	L1D_DEFFEATURE
3	7	0x307	L1D_DEBUG0
3	8	0x308	L1D_DEBUG1
3	9	0x309	L1D_CACHE_ERROR_LOG
3	A	0x30A	L1D_CACHE_ERROR_OVF_LOG
3	B	0x30B	L1D_CACHE_INTERRUPT
3	C	0x30C	L1D_WAY_DISABLE

6.4.2 L1 Cache Control Registers Descriptions

6.4.2.1 L1I_WAY_ENABLE

Block ID: 1 Offset: A Address: 0x10A

		63:32	31:24	23:16	15:8	7:0	
Bits	Name	Description				R/W	Reset
63:32	RESERVED	RESERVED				RO	
31:24	nCPU3_IW_EN	nCPU3 I-Cache Way Enable: The msb of this field (bit 31) corresponds to I-Cache Way 7; the lsb of this field (bit 24) corresponds to I-Cache Way 0. Other bits correspond to their respective I-Cache Way. For each bit of this field, 0 = disable use of the respective way for nCPU3 1 = enable use of the respective way for nCPU3 (default)				RW	0xFF
23:16	nCPU2_IW_EN	nCPU2 I-Cache Way Enable: The msb of this field (bit 23) corresponds to I-Cache Way 7; the lsb of this field (bit 16) corresponds to I-Cache Way 0. Other bits correspond to their respective I-Cache Way. For each bit of this field, 0 = disable use of the respective way for nCPU2 1 = enable use of the respective way for nCPU2 (default)				RW	0xFF
15:8	nCPU1_IW_EN	nCPU1 I-Cache Way Enable: The msb of this field (bit 15) corresponds to I-Cache Way 7; the lsb of this field (bit 8) corresponds to I-Cache Way 0. Other bits correspond to their respective I-Cache Way. For each bit of this field, 0 = disable use of the respective way for nCPU1 1 = enable use of the respective way for nCPU1 (default)				RW	0xFF
7:0	nCPU0_IW_EN	nCPU0 I-Cache Way Enable: The msb of this field (bit 7) corresponds to I-Cache Way 7; the lsb of this field (bit 0) corresponds to I-Cache Way 0. Other bits correspond to their respective I-Cache Way. For each bit of this field, 0 = disable use of the respective way for nCPU0 1 = enable use of the respective way for nCPU0 (default)				RW	0xFF

Writing a 0 to a particular bit disables the use of the particular cache way for an nCPU. For example, a value of 0xC0300C03 would enable two unique cache ways per nCPU creating a partitioned cache mode where each nCPU has 8KB of 2-way I-cache.

6.4.2.2 L1D_CONFIG0**Block ID: 3 Offset: 0 Address: 0x300**

63:15	14	13	12	11:8	7	6	5	4	3	2	1	0
RSVD	Unalign AccEn	Low Power Disable L1D	Force Mapped Data Erl	Mem Ordering Mode	uTLB Partition ing	Force Mapped Data	Cache Error En	L2 Line Alloc En	Data ECC En	Snoop Tag Parity En	Tag Parity En	L1D Cache En

Bits	Name	Description	R/W	Reset															
63:15	RESERVED	RESERVED.	R/W																
14	UnalignAccEn	<p>Unaligned Access Enable. When enabled, LH, LHU, LW, LWU, LD, SH, SW, and SD instructions may access memory addresses along any byte boundary.</p> <p>0: Memory accesses must be naturally-aligned. Unaligned accesses will result in an Address Error exception.</p> <p>1: Memory accesses can be unaligned. Unaligned accesses will NOT result in an Address Error exception.</p> <p>When an unaligned store crosses an 8-byte boundary and an exception (e.g. -- TLB, interrupt, etc.) occurs on the part that crosses the boundary, then the first part will complete correctly, but an exception will be taken and the EPC will point to the store's PC.</p> <p>Note: Some operating systems may use Address Errors to pass control between processes (for example, when doing floating point emulation). If Unaligned Accesses are enabled, then the desired exception may not be taken. One example of such an operating system is Linux. In this case, the unaligned exception was created by changing an unaligned LW to an unaligned LL.</p>	R/W	0															
13	LowPowerDisableL1D	<p>Low Power Disable. Controls low power mode for a D cache in a core.</p> <p>Normally, low power mode is enabled. This bit allows disabling the mode for this core.</p> <p>0: enable LP mode (default) 1: disable LP mode</p>	R/W	0															
12	ForceMappedDataErl	<p>Virtual MIPS mode - ERL is virtualized</p> <p>0: disable (default) 1: enable virtualizing of ERL</p>	R/W	0															
11:8	MemOrderingMode	<p>Memory ordering mode. Any combination of enabled modes is valid.</p> <table border="1"> <thead> <tr> <th>Bit</th><th>Function</th><th>Value</th></tr> </thead> <tbody> <tr> <td>11</td><td>LD-LD ordering</td><td>0: disable (default) 1: enable</td></tr> <tr> <td>10</td><td>ST-LD ordering</td><td>0: disable (default) 1: enable</td></tr> <tr> <td>9</td><td>LD-ST ordering</td><td>0: disable (default) 1: enable</td></tr> <tr> <td>8</td><td>ST-ST ordering</td><td>0: disable 1: enable (default)</td></tr> </tbody> </table>	Bit	Function	Value	11	LD-LD ordering	0: disable (default) 1: enable	10	ST-LD ordering	0: disable (default) 1: enable	9	LD-ST ordering	0: disable (default) 1: enable	8	ST-ST ordering	0: disable 1: enable (default)	R/W	See Desc
Bit	Function	Value																	
11	LD-LD ordering	0: disable (default) 1: enable																	
10	ST-LD ordering	0: disable (default) 1: enable																	
9	LD-ST ordering	0: disable (default) 1: enable																	
8	ST-ST ordering	0: disable 1: enable (default)																	

Bits	Name	Description	R/W	Reset
7	uTLBPartitioning	uTLB partitioned by threads for this core. 0: disable (default) 1: enable	R/W	0
6	ForceMappedData	Force-mapped data (Virtual MIPS mode; see chapter 3.) 0: disable (default) 1: enable	R/W	0
5	CacheErrorEn	Cache error log and report. Enables logging and reporting of L1 D cache errors for this core 0: disable (default) 1: enable	R/W	0
4	L2LineAllocEn	L2 line allocation. Enables use of line allocation in L2 D cache when servicing L1 miss request or line eviction. 0: disable 1: enable feature	R/W	0
3	DataECCEn	Data ECC enable 0: disable (default) 1: enable	R/W	0
2	SnoopTagParityEn	Snoop Tag parity enable 0: disable (default) 1: enable	R/W	0
1	TagParityEn	Tag parity enable 0: disable (default) 1: enable	R/W	0
0	L1D_CEn	L1 D cache enable 0: disable (default) 1: enable	R/W	0

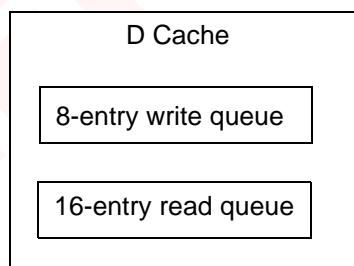
6.4.2.3 L1D_CONFIG1**Block ID: 3 Offset: 1 Address: 0x301**

63	48 47	32
	RDQ3	
31	16 15	0
	RDQ1	

Bits	Name	Description	R/W	Reset
63:48	RDQ3	Thread 3 RDQ mask 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0
47:32	RDQ2	Thread 2 RDQ mask 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0
31:16	RDQ1	Thread 1 RDQ mask 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0
15:0	RDQ0	Thread 0 RDQ mask 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0

RDQ Control

As shown below, the D cache has an 8 entry write queue and a 16 entry read queue. The bits of the fields in this register mask the use of the corresponding position in the Read Queue.



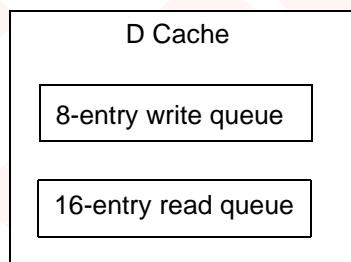
6.4.2.4 L1D_CONFIG2**Block ID: 3 Offset: 2 Address: 0x302**

31	24 23	16 15	8 7	0
	WRQ3	WRQ2	WRQ1	WRQ0

Bits	Name	Description	R/W	Reset
31:24	WRQ3	Thread 3 WDQ mask 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0
23:16	WRQ2	Thread 2 WDQ mask 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0
15:8	WRQ1	Thread 1 WDQ mask 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0
7:0	WRQ0	Thread 0 WDQ mask 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0

WRQ Control

As shown below, the D cache has an 8 entry write queue and a 16 entry read queue. The bits of these fields mask the use of the corresponding position in the Write Queue.



6.4.2.5 L1D_CONFIG3

Block ID: 3 Offset: 3 Address: 0x303

63	8 7	0
RESERVED		SNOOPMASK

Bits	Name	Description	R/W	Reset
63:8	RESERVED	RESERVED		
7:0	SNOOPMASK	Snoop Mask. For a snoop, these bits mask use of a position in the D cache's Write queue. 0: enables use of the position in the queue (default) 1: disables use of the position in the queue.	R/W	0

6.4.2.6 L1D_CONFIG4

Block ID: 3 Offset: 4 Address: 0x304

63	4	3	2	1	0
RESERVED		F_ST_PE	F_T_PE	F_DBE	F_SBE

Bits	Name	Description	R/W	Reset
63:4	RESERVED	RESERVED		
3	F_ST_PE	Force Snoop Tag Parity Error 0: no action 1: force snoop tag parity error.	R/W	0
2	F_T_PE	Force Tag Parity Error 0: no action 1: force tag parity error.	R/W	0
1	F_DBE	Force Double Bit Error 0: no action 1: force double bit error.	R/W	0
0	F_SBE	Force Single Bit Error 0: no action 1: force single bit error.	R/W	0

These control bits are for testing use. The user must manually clear them if they had previously set them.

6.4.2.7 L1D_STATUS

Block ID: 3 Offset: 5 Address: 0x305

63	6	5	2	1	0
RESERVED		CCEP	WRQE	RDQE	

Bits	Name	Description	R/W	Reset
63:6	RESERVED	RESERVED		
5:2	CCEP	Cumulative Cache Error Pending 0: no activity 1: indicates activity in either the cache error log or the error ovl log. User must read status in those logs to find the source. Once user has cleared the log, this bit clears. Bits 5 - thread 3 4 - thread 2 3 - thread 1 2 - thread 0	R	0
1	WRQE	WRQ empty 0: not empty 1: empty	R	0
0	RDQE	RDQ empty 0: not empty 1: empty	R	0

6.4.2.8 L1D_DEFFEATURE

Block ID: 3 Offset: 6 Address: 0x306

31	24	23	22	21	20	13	12	0
RESERVED	DP	Rsv	FPA	WAYDIS		RESERVED		

Bits	Name	Description	R/W	Reset
31:24	RESERVED	RESERVED	R/W	0
23	DP	Disable Prefetches 0: enable prefetches 1: disables prefetches	R/W	0
22	RESERVED	RESERVED (must write 0)	R/W	0
21	FPA	Flush on prefetch abort 0: disable flush 1: enable flush	R/W	0
20:13	WAYDIS	Way Disable 0: enable way 1: disable way Bits: 20: way 7 19: way 6 18: way 5 17: way 4 16: way 3 15: way 2 14: way 1 13: way 0	R/W	0
12:0	RESERVED	RESERVED (must write 0)	R/W	0

6.4.2.9 L1D_DEBUG0

Block ID: 3 Offset: 7 Address: 0x307

63	28 27	0
RESERVED		TAGDATA

Bits	Name	Description	R/W	Reset
63:28	RESERVED	RESERVED		
27:0	TAGDATA	Tag data that is loaded upon execution of a cache op load tag instruction.	R/W	-

6.4.2.10 L1D_DEBUG1

Block ID: 3 Offset: 8 Address: 0x308

31	30	24	23	21	20	18	17	15	14	12	11	9	8	6	5	3	2	0
RESERVED	LRU	MOSI7	MOSI6	MOSI5	MOSI4	MOSI3	MOSI2	MOSI1	MOSI0									

Bits	Name	Description	R/W	Reset
31	RESERVED	RESERVED		
30:24	LRU	LRU Status of cache.	R	Undefined
23:21	MOSI7	Data for MOSI way 7. MOSI encoding: 0x0 : INVALID 0x1 : SHARED 0x2 : MODIFIED 0x3 : OWN {0x7-0x4} : Intermediate states not defined.	R	Undefined
20:18	MOSI6	MOSI way 6. See above	R	Undefined
17:15	MOSI5	MOSI way 5. See above	R	Undefined
14:12	MOSI4	MOSI way 4. See above	R	Undefined
11:9	MOSI3	MOSI way 3. See above	R	Undefined
8:6	MOSI2	MOSI way 2. See above	R	Undefined
5:3	MOSI1	MOSI way 1. See above	R	Undefined
2:0	MOSI0	MOSI way 0. See above	R	Undefined

These parameters are loaded upon execution of a cache op load tag instruction

6.4.2.11 L1D_CACHE_ERROR_LOG

Block ID: 3 Offset: 9 Address: 0x309

63	56	55	54	47	46	10	9	8	7	6	5	4	3	2	1	0
RESERVED	L	WHV		ADR		D	A	P	B	S	T	E3	E2	E1	E0	

Cache error log.

Bits	Name	Description	R/W	Reset
63:56	RESERVED	RESERVED		
55	L	Dirty Line 0: write 0 to clear 1: dirty line	R/W	0
54:47	WHV	Way hit vector	R/W	0
46:10	ADR	Bits [39:3] of the physical address associated with the error	R/W	0
9	D	Data Error 0: write 0 to clear 1: data error	R/W	0
8	A	Address Error 0: write 0 to clear 1: address error	R/W	0
7	P	Tag Parity Error 0: write 0 to clear 1: tag parity error	R/W	0
6	B	Double Bit Error 0: write 0 to clear 1: double bit error	R/W	0
5	S	Store Type 0: write 0 to clear 1: store type error	R/W	0
4	T	Load Type 0: write 0 to clear 1: load type error	R/W	0
3	E3	Cache Error Pending thread 3 0: write 0 to clear 1: cache error pending - thread 3	R/W	0
2	E2	Cache Error Pending thread 2 0: write 0 to clear 1: cache error pending - thread 2	R/W	0
1	E1	Cache Error Pending thread 1 0: write 0 to clear 1: cache error pending - thread 1	R/W	0
0	E0	Cache Error Pending thread 0 0: write 0 to clear 1: cache error pending - thread 0	R/W	0

6.4.2.12 L1D_CACHE_ERROR_OVF_LOGBlock ID: **Offset: A** Address: **0x30A**

31	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	DL	DE	AE	TPE	DBE	ST	LT	CEO_T3	CEO_T2	CEO_T1	CEO_T0	

Bits	Name	Description	R/W	Reset
31:11	RESERVED	RESERVED		
10	DL	Dirty Line 0: write 0 to clear 1: dirty line	R/W	0
9	DE	Data Error 0: write 0 to clear 1: data error	R/W	0
8	AE	Address Error 0: write 0 to clear 1: address error	R/W	0
7	TPE	Tag Parity Error 0: write 0 to clear 1: tag parity error	R/W	0
6	DBE	Double Bit Error 0: write 0 to clear 1: double bit error	R/W	0
5	ST	Store Type 0: write 0 to clear 1: store type error	R/W	0
4	LT	Load Type 0: write 0 to clear 1: load type error	R/W	0
3	CEO_T3	Cache Error Overflow thread 3 0: write 0 to clear 1: cache error overflow - thread 3	R/W	0
2	CEO_T2	Cache Error Overflow thread 2 0: write 0 to clear 1: cache error overflow - thread 2	R/W	0
1	CEO_T1	Cache Error Overflow thread 1 0: write 0 to clear 1: cache error overflow - thread 1	R/W	0
0	CEO_T0	Cache Error Overflow thread 0 0: write 0 to clear 1: cache error overflow - thread 0	R/W	0

6.4.2.13 L1_CACHE_INTERRUPT

This register identifies cache-related causes of an interrupt to the PIC.

Note: If a uTLB multi-hit or data poisoned or single bit error occurred, then the cache set and cache index are logged here. Upon a cache interrupt, the programmer must read the EIRR register and the appropriate Px bit here, then clear the active Px bit.

Block ID: 3 Offset: B Address: 0x30B

63	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED	CS	CI	MH	DP	SBE	CIP_T3	CIP_T2	CIP_T1	CIP_T0	CIM_T3	CIM_T2	CIM_T1	CIM_T0	

Bits	Name	Description	R/W	Reset
63:26	Reserved	Reserved		0
25:18	CS	Cache way (only valid if SBE bit is set)	R/W	0
17:11	CI	Cache index (only valid if SBE bit is set)	R/W	0
10	MH	uTLB multi-hit 0: write 0 to clear 1: this was source of interrupt	R/W	0
9	DP	Data poisoned -- 0: write 0 to clear 1: this may be source of interrupt ^a	R/W	0
8	SBE	Single Bit Error 0: write 0 to clear 1: this was source of interrupt	R/W	0
7	CIP_T3	Cache Interrupt Pending thread 3 0: write 0 to clear 1: this was source of interrupt	R/W	0
6	CIP_T2	Cache Interrupt Pending thread 2 0: write 0 to clear 1: this was source of interrupt	R/W	0
5	CIP_T1	Cache Interrupt Pending thread 1 0: write 0 to clear 1: this was source of interrupt	R/W	0
4	CIP_T0	Cache Interrupt Pending thread 0 0: write 0 to clear 1: this was source of interrupt	R/W	0
3	CIM_T3	Cache Interrupt Enable thread 3 0: disables interrupt reporting for this thread 1: enables interrupt reporting for this thread	R/W	0
2	CIM_T2	Cache Interrupt Mask thread 2 0: disables interrupt reporting for this thread 1: enables interrupt reporting for this thread	R/W	0
1	CIM_T1	Cache Interrupt Mask thread 1 0: disables interrupt reporting for this thread 1: enables interrupt reporting for this thread	R/W	0
0	CIM_T0	Cache Interrupt Mask thread 0 0: disables interrupt reporting for this thread 1: enables interrupt reporting for this thread	R/W	0

- a. When a data poisoning occurs due to a store (no load return) the thread which receives the interrupt is random.
 All cache-error reporting is unaffected by this....the only result is the reduced visibility on data poisoning cases.
 To recover the data poisoning source history, use one of the following:

1. When all 4 threads of a CPU Core are enabled, a thread receiving a cache error interrupt can send a message to the appropriate thread the cache error logging registers.
2. If less than 4 threads are being used, then cache error interrupts can be disabled so that they are not delivered to a non-existent thread.

6.4.2.14 L1D_WAY_DISABLE

Block ID: 3 Offset: C Address: 0x30C

63:32		31:24		23:16		15:8		7:0					
RESERVED		nCPU3_DW_DIS		nCPU2_DW_DIS		nCPU1_DW_DIS		nCPU0_DW_DIS					
Bits	Name	Description						R/W	Reset				
63:32	RESERVED	RESERVED						RO					
31:24	nCPU3_DW_DIS	nCPU3 D-Cache Way Disable: The msb of this field (bit 31) corresponds to D-Cache Way 7; the lsb of this field (bit 24) corresponds to D-Cache Way 0. Other bits correspond to their respective D-Cache Way. For each bit of this field, 0: enable use of the respective way for nCPU3 (default) 1: disable use of the respective way for nCPU3						RW	0x00				
23:16	nCPU2_DW_DIS	nCPU2 D-Cache Way Disable: The msb of this field (bit 23) corresponds to D-Cache Way 7; the lsb of this field (bit 16) corresponds to D-Cache Way 0. Other bits correspond to their respective D-Cache Way. For each bit of this field, 0: enable use of the respective way for nCPU2 (default) 1: disable use of the respective way for nCPU2						RW	0x00				
15:8	nCPU1_DW_DIS	nCPU1 D-Cache Way Disable: The msb of this field (bit 15) corresponds to D-Cache Way 7; the lsb of this field (bit 8) corresponds to D-Cache Way 0. Other bits correspond to their respective D-Cache Way. For each bit of this field, 0: enable use of the respective way for nCPU1 (default) 1: disable use of the respective way for nCPU1						RW	0x00				
7:0	nCPU0_DW_DIS	nCPU0 D-Cache Way Disable: The msb of this field (bit 7) corresponds to D-Cache Way 7; the lsb of this field (bit 0) corresponds to D-Cache Way 0. Other bits correspond to their respective D-Cache Way. For each bit of this field, 0: enable use of the respective way for nCPU0 (default) 1: disable use of the respective way for nCPU0							RW	0x00			

Writing a '1' to a particular bit disables the use of the particular cache way for an nCPU. For example, a value of 0x3FCFF3FC would enable two unique cache ways per nCPU creating a partitioned cache mode where each nCPU has 8KB of 2-way D-cache.



Chapter 7 XLS Unified L2 Cache

7.1 Introduction

The XLS Processor incorporates a unified second level cache for its MIPS64 CPU cores. Depending on the XLS family model, the cache is either 1MB, 512KB, or 256KB.

- In models with a 1MB cache, the on-chip unified write-back cache is 4-banked and 8-way set-associative with a 32-byte line size. Logically, it appears to the processors as a 4-banked array.
- In models with a 512KB cache, the on-chip unified write-back cache is 2-banked and 8-way set-associative with a 32-byte line size. Logically, it appears to the processors as a 2-banked array.
- In models with a 256KB cache, the on-chip unified write-back cache is 1-banked and 8-way set-associative with a 32-byte line size. Logically, it appears to the processors as a single-banked array.

This chapter provides the following information about the cache and controller:

- Overview of capabilities
- Theory of operation
- Programming notes
- L2 cache controller registers
 - Master list of registers
 - Register descriptions

Note: In this chapter, 'group of registers' means a collection of registers. The phrase 'set of registers' was not used here to avoid confusion with the use of the word 'set' in the cache context.

7.2 Capabilities and Functions

The Level-2 cache uses Pseudo Least Recently Used (Pseudo LRU, or PLRU) replacement policy for way selection and supports locking on a per line basis. The L2's design is multi-banked and allows for up to 8 parallel (simultaneous) accesses at any given cycle.

The data and tag arrays of the L2 cache are protected using ECC (SECDED). All single bit errors are corrected and double error detected without software intervention. When uncorrectable errors are detected, they are passed to the software as a cache error exception.

The L2 cache is designed to be non-inclusive of the L1 caches allowing more data to be cached than would ordinarily be in an inclusive design.

7.3 Programming Notes

7.3.1 Using the L2 Cache Bank Configuration Registers

Each of the banks in the L2 cache has its own group of configuration registers.

- Models with 256KB L2 cache have one bank and hence one group of registers
- Models with 512KB cache have two banks and two groups
- Models with 1MB L2 cache have four banks and hence four groups

The 40-bit physical address is broken up differently for different XLS models.

- Models with 256KB L2 cache:
 - address bits [39:15] are used to specify the tag
 - address bits [14:5] are used to specify the set
 - no address bits are used to specify the bank, as there is always one bank
- Models with 512KB L2 cache:
 - address bits [39:15] are used to specify the tag
 - address bits [15:6] are used to specify the set
 - address bit [5] used to specify the bank
- Models with 1MB L2 cache:
 - address bits [39:15] are used to specify the tag
 - address bits [16:7] are used to specify the set
 - address bits [6:5] are used to specify the bank

This section provides general guidelines for using L2 bank configuration registers. The rules are:

1. All groups of configuration registers always must be configured.
2. All banks need to be configured correctly.
3. All accesses must be uncacheable.
4. Use bits 7:5 of address to specify the bank (when needed)
5. All registers except for CMD register are used simply by writing and reading to them.
6. L2 CMD register usage:

L2 CMD requires parameters as shown in [Table 7-3](#). Writes to the L2 CMD register trigger the operation requested using the CMD OP field and will be applied to the SET and WAY specified in the L2 CMD register.

- Return data will be read out of appropriate register.
- For writes, data must be loaded into the appropriate tag/data registers before enabling write commands to tag/data rams.
- The L2 CMD Valid Array field states whether or not the particular set/way is valid or not (encoding has three bits: lock, dirty, clean). If the dirty or clean bit is set, then the data in the L2 is valid.
- The L2 CMD Tag Ram field contains the Tag of the address (bits [39:15]).
- The L2 CMD Data Ram field contains the 256 bits of data.

7. The following pseudocode illustrates how to read L2 Valid Array, TagRam, and DataRam:

```

    :
For each bank ($bank)
{
  For each set ($set)
  {
    For each way ($way)
    {
      Write L2_CMD register (with L2_READ_VALID_ARRAY opcode, and $set)
      Read L2_TAG/VLD_ARRAY register
      Write L2_CMD register (with L2_READ_TAG_RAM opcode, $set, and $way)
      Read L2_TAG/VLD_ARRAY register
      Write L2_CMD register (with L2_READ_DATA_RAM opcode, $set, and $way)
      Read L2_Data_Register_0
      Read L2_Data_Register_1
      Read L2_Data_Register_2
      Read L2_Data_Register_3
      Read L2_Data_Register_4
      Read L2_Data_Register_5
      Read L2_Data_Register_6
      Read L2_Data_Register_7
    }
  }
}
}

```

Locking of Ways in L2

- For correct operation, only 7 of the 8 ways for a particular set can be locked.
- Locking all 8 ways will lead to undefined behavior.
- There are two methods for locking a particular address in the L2:
 - Method 1:
 - calculate tag -> address bits [39:15]
 - calculate set -> address bits [17:8]
 - calculate bank -> address bits [7:5]
 - calculate ECC for tag
 - calculate ECC for data
 - read valid bits for that particular set and bank.
 - for each valid way for that set/bank, read the tag.
 - if the tag you wish to lock is in a valid way, you can lock that way,
 - otherwise pick an invalid way for the locking. (it is illegal to have the same tag in two VALID ways of a particular bank/set.)
 - write the data for that bank/set/way
 - write the tag for that bank/set/way
 - write the valid array for that bank/set/way with the lock bit set and either dirty/clean state.
 - Method 2:
 - calculate tag -> address bits [39:15]
 - calculate set -> address bits [17:8]
 - calculate bank -> address bits [7:5]
 - calculate ECC for tag
 - read valid bits for that particular set and bank.
 - for each valid way for that set/bank, read the tag.

- if the tag you wish to lock is in a valid way, you can lock that way,
- otherwise pick an invalid way for the locking. (it is illegal to have the same tag in two VALID ways of a particular bank/set.)
- write the tag for that bank/set/way
- write the valid array for that bank/set/way with the lock bit set and a zero for dirty/clean state [i.e., use value b100 for lock/dirty/clean]
- do a store to that address with correct data
- flush L1 cache for that address
- line will become locked/valid in L2

NETLOGIC
CONFIDENTIAL

7.4 L2 Cache Control Registers

This section discusses the L2 cache control registers. As mentioned previously, there is one group per bank and thus one group for 256KB caches, two groups for 512KB caches and four groups for 1MB caches. All banks (that is, groups) must be configured for correct operation.

7.4.1 Background

7.4.1.1 Definition of Terms

The following terms are used in the register descriptions to describe field modifiability type:

Table 7-1. Field Modifiability Codes

Type Code	Meaning	Description
R/O	Read Only	The field contains a static value and cannot be written.
R/W	Read/Write	The field can be modified by software. Note that some fields, where indicated, are modifiable only indirectly, by writing to a shadow register then performing a soft reset. Except as noted, for single-bit fields, 1 enables the related function, 0 disables it.
R/C	Read/Clear	The field can be read to give status information. Bits may be cleared by writing a 1 to them; writes of 0 are ignored.

7.4.1.2 Register Access

Address access to the L2 cache controller's registers is controlled by the System Bridge Controller, as described in Chapter 8.

L2 cache control registers are accessed using the combination of individual register offset, plus L2 cache controller base address, plus base address for all XLS_IO devices as described in Chapter 8. In addition, selection of the registers for a particular bank requires defining the group (IDH and IDL values) and bank number. See [Table 7-3](#) for addressing.

Bits 39:2 select the register to be addressed.

Table 7-2. Address Format for Accessing L2 Cache Registers

Address Bits		Description
Bit#	Name	
39:20	Base	XLS I/O base - this base address value is programmable in the XLS System Bridge Controller, and sets base address for all devices. (The L2 cache is accessed as a device.)
19:12	Config	Address of L2 configuration space, which is at fixed offset of 0x1 B000 from the start of XLS I/O base
11:8	IDH	L2 cache control register group ID. This value selects the register group to access for the selected bank.
7:5	Bank	Bank number. Required to define which bank the register group is for. 000: bank 0 010: bank 2 001: bank 1 011: bank 3
4:2	IDL	L2 cache control register number (IDL) in Table 7-3 . This value selects a specific register within the group for the selected bank
1:0	Reserved	set to 0

7.4.2 L2 Cache Control Register Summary

The following table summarizes registers and bank addressing for each bank in the L2 cache. Note that correct operation requires that all banks are configured.

- For an XLS with a 1MB L2 cache: only 4 banks must be configured. Bit[7] (upper bit of the BANK# field) will always be a zero.
- For an XLS with a 512KB L2 cache: only 2 banks must be configured. Bits[7:6] (upper two bits of the BANK# field) will always be zero.
- For an XLS with a 256KB L2 cache: 1 bank must be configured.

Caution: When the L2 cache processes more than one configuration register write or read at nearly the same time (that comes from different nCPU™ NetLogic virtual CPUs), it is possible for one of the requests to fail to complete, which can lead to a processor hang. This potential problem can be avoided by using only one nCPU to perform configuration register accesses. If this is not possible, a lock or barrier should be used to ensure that only one nCPU accesses the L2 configuration registers at any given time.

Table 7-3. L2 Cache Control Registers Summary and Addressing

Address				Register Name
11:8	7:5	4:2	1:0	(All groups of registers for L2 cache banks use the same names)
GROUP (IDH)	BANK#	REG (IDL)		
0000	000 to 011	000	00	L2 Feature Control Register
0000	000 to 011	001	00	L2 CMD Register
0000	000 to 011	010	00	L2 Disable WLB Entries
0000	000 to 011	011	00	L2 Disable DOB Entries
0000	000 to 011	100	00	Reserved
0000	000 to 011	101	00	L2 Tag/Vld Array Register
0000	000 to 011	110	00	L2 Data ECC Register
0000	000 to 011	111	00	L2 XLS IO Shadow Register
0001	000 to 011	000	00	L2 Data Register 7 Bits [255:224]
0001	000 to 011	001	00	L2 Data Register 6 Bits [223:192]
0001	000 to 011	010	00	L2 Data Register 5 Bits [191:160]
0001	000 to 011	011	00	L2 Data Register 4 Bits [159:128]
0001	000 to 011	100	00	L2 Data Register 3 Bits [127:96]
0001	000 to 011	101	00	L2 Data Register 2 Bits [95:64]
0001	000 to 011	110	00	L2 Data Register 1 Bits [63:32]
0001	000 to 011	111	00	L2 Data Register 0 Bits [31:0]
0010	000 to 011	000	00	L2 Correctable ECC Log Register
0010	000 to 011	001	00	L2 Uncorrectable ECC Log Reg
0010	000 to 011	010	00	L2 Performance Counter 0
0010	000 to 011	011	00	L2 Performance Counter 1
0010	000 to 011	100	00	L2 Disable Ways Register 0
0010	000 to 011	101	00	L2 Disable Ways Register 1
0010	000 to 011	110	00	Reserved
0010	000 to 011	111	00	Reserved

Example: address field 11:0 for L2 Data ECC register for bank 3 is:

group IDH 0000 bank 011 reg IDL 110 00, or 000001111000.

Example: address field 11:0 for L2 Data register 1 for bank 7 is:

group IDH 0001 bank 111reg IDL 110 00, or 00011111000.

7.4.3 L2 Cache Control Register Descriptions

7.4.3.1 L2 Feature Control Register

This register in each bank's group contains various control bits for L2 configuration and defeaturing for that bank.

31:24	23:12	11	10	9	8	7	6	5	4	3	2	1	0
GDW	Reserved	LPDL2	E	F	G	H	J	K	L	M	N	O	P

Bits	Name	Description	R/W	Reset
31:24	GDW	Global Disable Ways [7, 6, 5, 4, 3, 2, 1, 0] 0: enable way 1: disables the way. 31: way 7 30: way 6 29: way 5 28: way 4 27: way 3 26: way 2 25: way 1 24: way 0 Note: For correct operation, never disable all 8 ways.	R/W	0
23:12	Reserved	Reserved	-	-
11	LowPowerDisableL2	Disable clock for low-power operation of L2 cache. 0: Enable clock gating 1: Disable clock gating	R/W	
10	E	Bank Enable Bit 0: disable bank 1: enable bank	R/W	0
9	F	Allow bank to drop fills when WLBFULL 0: disable 1: enable	R/W	0
8	G	Inject Data Ram Double Bit Error 0: do not inject 1: inject error	R/W	0
7	H	Inject Data Ram Single Bit Error 0: do not inject 1: inject error	R/W	0
6	J	Inject Tag Ram Double Bit Error 0: do not inject 1: inject error	R/W	0
5	K	Inject Tag Ram Single Bit Error 0: do not inject 1: inject error	R/W	0
4	L	Inject Valid Array Parity Error 0: do not inject 1: inject error	R/W	0

Bits	Name	Description	R/W	Reset
3	M	Disable Data Poisoning for the bank 0: enable 1: disable	R/W	0
2	N	Disable Data RAM ECC 0: enable 1: disable	R/W	0
1	O	Disable Tag RAM ECC 0: enable 1: disable	R/W	0
0	P	Disable Valid Array Parity 0: enable 1: disable	R/W	0

7.4.3.2 L2 CMD Register

This special configuration register is used to access the L2 arrays for this bank, and other debug features. After setting up supporting registers, a write to this register will trigger the programmed operation. Note that the use of the data in bit fields 31:22 depends on which CMD is being executed.

31:28	27:22	21:19	18:16	15:4	3:0
Pbits[31:28]	not used	Way	Vbits	Reserved	CMD
Set[31:22]					

Bits	Name	Description	R/W	Reset																		
31:28	Pbits	Performance Counter Select Bits. Use of this 4-bit field depends on the CMD value. See Table 7-4 and Table 7-5 .	W	0																		
31:22	Set	Set Bits - address bits [17:8] Use of this field depends on the CMD value. See Table 7-4	W	0																		
21:19	Way	Way Bits [3-bit encoded value] <table border="1"> <tr> <th>21:19</th> <th>Way</th> </tr> <tr> <td>000</td> <td>0</td> </tr> <tr> <td>001</td> <td>1</td> </tr> <tr> <td>010</td> <td>2</td> </tr> <tr> <td>011</td> <td>3</td> </tr> <tr> <td>100</td> <td>4</td> </tr> <tr> <td>101</td> <td>5</td> </tr> <tr> <td>110</td> <td>6</td> </tr> <tr> <td>111</td> <td>7</td> </tr> </table>	21:19	Way	000	0	001	1	010	2	011	3	100	4	101	5	110	6	111	7	W	0
21:19	Way																					
000	0																					
001	1																					
010	2																					
011	3																					
100	4																					
101	5																					
110	6																					
111	7																					

Bits	Name	Description	R/W	Reset																																				
18:16	Vbits	Valid Bits [Lock, Dirty, Clean] <table border="1" style="margin-left: 20px;"> <thead> <tr> <th>L 18</th> <th>D 17</th> <th>C 16</th> <th>meaning</th> </tr> </thead> <tbody> <tr><td>0</td><td>0</td><td>0</td><td>invalid entry</td></tr> <tr><td>0</td><td>0</td><td>1</td><td>clean line</td></tr> <tr><td>0</td><td>1</td><td>0</td><td>dirty line</td></tr> <tr><td>0</td><td>1</td><td>1</td><td>illegal value</td></tr> <tr><td>1</td><td>0</td><td>0</td><td>locked and invalid</td></tr> <tr><td>1</td><td>0</td><td>1</td><td>locked and clean</td></tr> <tr><td>1</td><td>1</td><td>0</td><td>locked and dirty</td></tr> <tr><td>1</td><td>1</td><td>1</td><td>illegal</td></tr> </tbody> </table>	L 18	D 17	C 16	meaning	0	0	0	invalid entry	0	0	1	clean line	0	1	0	dirty line	0	1	1	illegal value	1	0	0	locked and invalid	1	0	1	locked and clean	1	1	0	locked and dirty	1	1	1	illegal	W	0
L 18	D 17	C 16	meaning																																					
0	0	0	invalid entry																																					
0	0	1	clean line																																					
0	1	0	dirty line																																					
0	1	1	illegal value																																					
1	0	0	locked and invalid																																					
1	0	1	locked and clean																																					
1	1	0	locked and dirty																																					
1	1	1	illegal																																					
15:4	Reserved	Reserved	—	—																																				
3:0	CMD	Command operation field. See Table 7-4 .	W	0																																				

Table 7-4. Command Operation Field [3:0]

CMD	Operation	Parameters required
0000	L2 Read Valid Array	[SET]
0001	L2 Write Valid Array	[SET, WAY, VBITS]
0010	L2 Read Tag Ram	[SET, WAY]
0011	L2 Write Tag Ram	[SET, WAY]
0100	L2 Read Data Ram	[SET, WAY]
0101	L2 Write Data Ram	[SET, WAY]
0110	L2 Performance Counter 0 Select	[PBits]
0111	L2 Performance Counter 1 Select	[PBits]
1000	L2 Evict Line From L2	[SET, WAY]
1001	L2 Release UCError Log Lock	0
1010	Reserved	0
1011 to 1110	Reserved	0
1111	Reserved	0

Table 7-5. Performance Counter (Pbits) Event Selection

Value	Pbits Event
0000	L2 Valid Ops
0001	L2 CPU Ops
0010	L2 IO Ops
0011	L2 Hit

Table 7-5. Performance Counter (Pbits) Event Selection (continued)

Value	Pbits Event
0100	L2 Hit Shared
0101	L2 Hit Modified
0110	L2 Read Ex Hits
0111	L2 Read Upg Hits
1000	L2 Write Hits
1001	L2 Write Allocated (can be cancelled)
1010	L2 Evict Allocated (can be cancelled)
1011	L2 Retry
1100	L2 Guard Retry
1101	L2 WLB Full Retry
1110	L2 DOB Full Retry
1111	L2 Access FIFO Full Retry

7.4.3.3**L2 Disable WLB Entries**

The L2 Disable Write Line Buffer register is used to partition the 16 buffers available for data being written to the L2 cache. Note: for correct operation do not disable all entries.

		31:16	15:0		
		L2 Disable WLB Entries for Writes	L2 Disable WLB Entries for Fills		
Bits	Name	Description		R/W	Reset
31:16		L2 Disable WLB Entries for Writes 0: enable (default) 1: disable		R/W	0
15:0		L2 Disable WLB Entries for Fills 0: enable (default) 1: disable		R/W	0

7.4.3.4**L2 Disable DOB Entries**

The L2 Disable Data Out Buffer register is used to partition the 16 buffers available for data being sent from the L2 cache. Note: for correct operation do not disable all entries.

		31:28	27:16	15:12	11:0	
		Reserved	L2 Disable DOB Entries for Evicts	Reserved	L2 Disable DOB Entries for Read Transfers	
Bits	Name	Description			R/W	Reset
31:28	Reserved	Reserved			R	0
27:16	DisDOBforEvicts	L2 Disable DOB Entries for Evicts 0: enable (default) 1: disable			R/W	0

Bits	Name	Description	R/W	Reset
15:12	Reserved	Reserved	R	0
11:0	DisDOBforRdTrans	L2 Disable DOB Entries for Read Transfers 0: enable (default) 1: disable	R/W	0

7.4.3.5 L2 Tag/Vld Array Register

Case 1: CMD 0000: This register returns VLD Array for CMD OP 0000: L2 Read Valid Array

31	30:24	23:21	20:18	17:15	14:12	11:9	8:6	5:3	2:0
P	LRU	W7VB	W6VB	W5VB	W4VB	W3VB	W2VB	W1VB	W0VB

Bits	Name	Description	R/W	Reset
31	P	Parity Bit	R	0
30:24	LRU	LRU Bits	R	0
23:21	W7VB	Way 7 VBits	R	0
20:18	W6VB	Way 6 VBits	R	0
17:15	W5VB	Way 5 VBits	R	0
14:12	W4VB	Way 4 VBits	R	0
11:9	W3VB	Way 3 VBits	R	0
8:6	W2VB	Way 2 VBits	R	0
5:3	W1VB	Way 1 VBits	R	0
2:0	W0VB	Way 0 VBits	R	0

Case 2: CMD 0010: This register returns tag for CMD OP 0010: L2 Read Tag RAM

31:7	6:0
Address Bits[39:15]	Tag ECC

Bits	Name	Description	R/W	Reset
31:7	Address Bits	Address bits [39:15]	R	0
6:0	Tag ECC	Tag ECC bits	R	0

Case 3: CMD 0011: This register is used to Write Tag for CMD OP 0011: L2 Write Tag RAM

31:7	6:0
Address Bits[39:15]	Tag ECC

Bits	Name	Description	R/W	Reset
31:7	Address Bits	Address bits [39:15]	W	0
6:0	Tag ECC	Tag ECC bits	W	0

7.4.3.6 L2 Data ECC Register

Case 1: CMD 0100: This register returns Data ECC for CMD OP 0100: L2 Read Data Ram

31:24	23:15	14:8	7:0
DECC255_192	DECC191_128	DECC127_64	DECC63_0

Bits	Name	Description	R/W	Reset
31:24	DECC255_192	Data ECC for Data [255:192]	R	0
23:15	DECC191_128	Data ECC for Data [191:128]	R	0
14:8	DECC127_64	Data ECC for Data [127:64]	R	0
7:0	DECC63_0	Data ECC for Data [63:0]	R	0

Case 2: CMD 0101: This register is used to write Data ECC for CMD OP 0101: L2 Write Data Ram

31:24	23:15	14:8	7:0
DECC255_192	DECC191_128	DECC127_64	DECC63_0

Bits	Name	Description	R/W	Reset
31:24	DECC255_192	Data ECC for Data [255:192]	W	0
23:15	DECC191_128	Data ECC for Data [191:128]	W	0
14:8	DECC127_64	Data ECC for Data [127:64]	W	0
7:0	DECC63_0	Data ECC for Data [63:0]	W	0

7.4.3.7 L2 XLS IO Shadow Register

Contains shadow copy of System Bridge controller XLS IO Bar Base Register values.

31:12	11:0
IOBarBase	IOBarRegNum

Bits	Name	Description	R/W	Reset
31:12	IOBarBase	IO Bar Base Value	R	0x000FF
11:0	IOBarRegNum	IO Bar Register Number	R	0x064

7.4.3.8 L2 Data Register 7

Case 1: CMD 0100. This register returns data for CMD OP 0100: L2 Read Data RAM

Case 2: CMD 0101: This register is used to write data for CMD OP 0101: L2 Write Data RAM.

31:0
Data [255:224]

Bits	Name	Description	R/W	Reset
31:0	Data	Data [255:224]	R/W	0

7.4.3.9 L2 Data Register 6

Case 1: CMD 0100: This register returns data for CMD OP 0100: L2 Read Data RAM.

Case 2: CMD 0101: This register is used to write data for CMD OP 0101: L2 Write Data RAM.

		31:0
Data [223:192]		

Bits	Name	Description	R/W	Reset
31:0	Data	Data [223:192]	R/W	0

7.4.3.10 L2 Data Register 5

Case 1: CMD 0100: This register returns data for CMD OP 0100: L2 Read Data RAM.

Case 2: CMD 0101: This register is used to write data for CMD OP 0101: L2 Write Data RAM.

		31:0
Data [191:160]		

Bits	Name	Description	R/W	Reset
31:0	Data	Data [191:160]	R/W	0

7.4.3.11 L2 Data Register 4

Case 1: CMD 0100: This register returns data for CMD OP 0100: L2 Read Data RAM.

Case 2: CMD 0101: This register is used to write data for CMD OP 0101: L2 Write Data RAM.

		31:0
Data [159:128]		

Bits	Name	Description	R/W	Reset
31:0	Data	Data [159:128]	R/W	0

7.4.3.12 L2 Data Register 3

Case 1: CMD 0100: This register returns data for CMD OP 0100: L2 Read Data RAM.

Case 2: CMD 0101: This register is used to write data for CMD OP 0101: L2 Write Data RAM.

		31:0
Data [127:96]		

Bits	Name	Description	R/W	Reset
31:0	Data	Data [127:96]	R/W	0

7.4.3.13 L2 Data Register 2

Case 1: CMD 0100: This register returns data for CMD OP 0100: L2 Read Data RAM.

Case 2: CMD 0101: This register is used to write data for CMD OP 0101: L2 Write Data RAM.

		31:0
Data [95:64]		
Bits	Name	Description
31:0	Data	Data [95:64]

7.4.3.14 L2 Data Register 1

Case 1: CMD 0100: This register returns data for CMD OP 0100: L2 Read Data RAM.

Case 2: CMD 0101: This register is used to write data for CMD OP 0101: L2 Write Data RAM.

		31:0
Data [63:32]		
Bits	Name	Description
31:0	Data	Data [63:32]

7.4.3.15 L2 Data Register 0

Case 1: CMD 0100: This register returns data for CMD OP 0100: L2 Read Data RAM.

Case 2: CMD 0101: This register is used to write data for CMD OP 0101: L2 Write Data RAM.

		31:0
Data [31:0]		
Bits	Name	Description
31:0	Data	Data [31:0]

7.4.3.16 L2 Correctable ECC Log Register

The most recent single bit Tag/Data error for this bank will be logged here.

31:22	21:14	13:0
SET	WAY	Reserved
Bits	Name	Description
31:22	SET	Set bits (Address Bits [17:8])
21:14	WAY	Way bits [7, 6, 5, 4, 3, 2, 1, 0]
13:0	Reserved	Reserved

7.4.3.17 L2 Uncorrectable ECC Log Reg

The first double bit Tag/Data error or VLD array parity error for this bank will be logged here and locked until the lock is released for this register.

Use CMD OP 1001 to release lock.

31:22	21:14	13:0
SET	WAY	Reserved

Bits	Name	Description	R/W	Reset
31:22	SET	Set bits (Address Bits [17:8])	R	0
21:14	WAY	Way bits [7, 6, 5, 4, 3, 2, 1, 0]	R	0
13:0	Reserved	Reserved	-	-

7.4.3.18 L2 Performance Counter 0

Use CMD OP: 0110 to select which event to count.

Read this register to get the count.

Writing this register initializes the counter to a new value.

31:0
CTR

Bits	Name	Description	R/W	Reset
31:0	CTR	Counter Results	R/W	0

7.4.3.19 L2 Performance Counter 1

Use CMD OP: 0111 to select which event to count.

Read this register to get the count.

Writing this register initializes the counter to a new value.

31:0
CTR

Bits	Name	Description	R/W	Reset
31:0	CTR	Counter Results	R/W	0

7.4.3.20 L2 Disable Ways Register 0

This register specifies which ways of the L2 for this bank are used for allocation purposes based on which CPU is requesting the cache line.

Note: for correct operation never disable all eight ways.

31:24	23:16	15:8	7:0
Reserved	Reserved	L2CPU1WDB	L2CPU0WDB

Bits	Name	Description	R/W	Reset
31:24	Reserved	Reserved	R/W	0
23:16	Reserved	Reserved	R/W	0
15:8	L2CPU1WDB	L2 CPU 1 Way Disable Bits	R/W	0
7:0	L2CPU0WDB	L2 CPU 0 Way Disable Bits	R/W	0

7.4.3.21 L2 Disable Ways Register 2

This register specifies which ways of the L2 for this bank are used for allocation purposes based on which I/O agent is requesting the cache line.

Note: for correct operation, never disable all eight ways.

31:24	23:16	15:8	7:0
L2IOGMAC	Reserved	Reserved	L2IOOTHER

Bits	Name	Description	R/W	Reset
31:24	L2IOGMAC	L2 IO GMAC Way Disable Bits	R/W	0
23:16	Reserved	Reserved	R/W	0
15:8	Reserved	Reserved	R/W	0
7:0	L2IOOTHER	L2 IO Other Way Disable Bits	R/W	0

7.4.4 L2 Cache Access Through JTAG

The L2 cache may be accessed through JTAG. The control capability for this is in the System Bridge Controller. See [Section 8.10 System Credits and Counter Registers](#), register 59. Setting register 59, bit[0] to a value of 1 allows JTAG access to the L2 cache registers.



Chapter 8 Memory and I/O Access

8.1 Introduction

Internal data transport in all XLS processors is managed by a fast Data Interconnect (DI) subsystem comprised of a high speed hub and two ring-topology interconnects, one for memory and one for I/O. Modules in the XLS attach to the rings via individual connection stations. The DI operates together with the DMA Engine, the Network Accelerator, and the Security Accelerator to ensure effective use of the XLS MIPS64 CPU cores.

The DI subsystem, shown in [Figure 8-1](#), has these key elements:

- The **Memory Distributed Interconnect (MDI)**, a high-speed unidirectional transport ring handling the L2 cache and main memory
- The **I/O Distributed Interconnect (I/O DI)**, a high-speed unidirectional transport ring handling all peripherals
- The **Memory and I/O Distributed Interconnect Hub (DI Hub)**, which connects the MDI and I/O DI with the DRAM Controller Pair AB, and which contains the **System Bridge Controller** that manages memory addressing

Additionally, the DMA Engine, the Network Accelerators and the Security Accelerator connect both to the I/O DI and the Fast Messaging Network, and integrate data transport activity with the operation of the DI Hub.

System elements connect to the rings transparently to the user, through high-speed stations. The XLS rings exhibit high bandwidth and operate at half the core clock frequency.

All non-core memories and I/O are mapped into a terabyte of physical address space (40-bits) by their appropriate bridges on the Memory Distributed Interconnect. All I/O bridges and I/O peripheral controllers are controlled in effect by the System Bridge Controller (SBC), which operates in the same memory-mapped I/O address space.

8.2 Elements of the XLS Internal Data Transport Architecture

[Figure 8-1](#) shows the overall architecture. The key elements are described below.

8.2.1 Memory Distributed Interconnect (MDI)

The MDI is a high-speed unidirectional transport ring handling the L2 cache and main memory.

8.2.2 I/O Distributed Interconnect (I/O DI)

The I/O DI is a high-speed unidirectional transport ring handling all peripherals. Credits may be allocated for data transport priorities for various peripherals to ensure adequate bandwidth.

8.2.3

Memory and I/O Distributed Interconnect Hub

The Memory and I/O Distributed Interconnect Hub (DI Hub) provides the bridges for the DRAM Controller Pair AB for channels A and B, as well as the System Bridge Controller for the I/O Distributed Interconnect (I/O DI) with its peripherals as shown in [Figure 8-2](#). There is a separate bridge for the DRAM Controller Pair CD for channels C and D.

The hub is responsible for managing memory requests and the efficient transfer of data between L1 Data Cache, L2 Unified Cache, and the DDR2 Controller Pair AB for up to 72 bits of DRAM on ports MA and MB. For data that is not available in either L1 or L2 cache or in any of the DRAMs, a request is sent to the hub I/O portion to access the external network for the data.

The primary function of the hub I/O portion is to steer output data traffic through the I/O DI to the appropriate I/O devices (such as SGMII/RGMII, PCIe, USB, UART, I²C, etc.). The I/O hub identifies the target device using the I/O address map programmed in the System Bridge Controller (SBC). Software is responsible for selecting and programming the address ranges for every device before any reference can be made.

8.2.4

Memory Bridge CD for DRAM Controller Pair CD

[Figure 8-2](#) shows the Memory Bridge CD. This bridge manages memory transactions and data transfer between the L1 Data Cache or L2 Unified Cache, and the DDR2 Controller Pair CD for the up to 72-bits of channels C and D on Ports MC and MD.

8.3

System Physical Memory Map

The 40-bit physical addresses on the MDI define a 1-Terabyte address space as shown in [Figure 8-3](#). Regions are not drawn to scale.

The upper right-hand side of the figure shows the major re-locatable regions in the 1-TB space: DRAM, the Peripheral & I/O Memory Map and the Peripheral & I/O Configuration regions.

The left side of [Figure 8-3](#) shows an expansion of the Peripheral & I/O Configuration region with the physical addresses after reset. The base for the entire region can be changed after reset using its own Base Address Register (BAR).

The lower right side of the figure shows an expansion of the System Bridge Controller region of memory and its sixty-four registers. These registers set the locations of the various regions for subsystem registers within the full 1-TB address space as shown by the connecting arrows.

Note that the control/status registers located in the Peripheral & I/O Configuration region for each peripheral device are distinct from System Bridge Controller registers. The control/status registers in the Peripheral & I/O Configuration region govern the operation of the I/O and peripheral controllers directly.

System Bridge Controller (SBC) registers, in general, control the bridges between actual memory controllers and I/O peripherals. They control the mapping of the memory or peripheral in the physical address space as connected on the MDI. The SBC can be thought of as a single controller for all bridges, but note that the Bridge CD Controller is distinct with its own DI Station on the MDI ring as shown in [Figure 8-2](#).

Figure 8-1. Distributed Interconnect in the XLS616

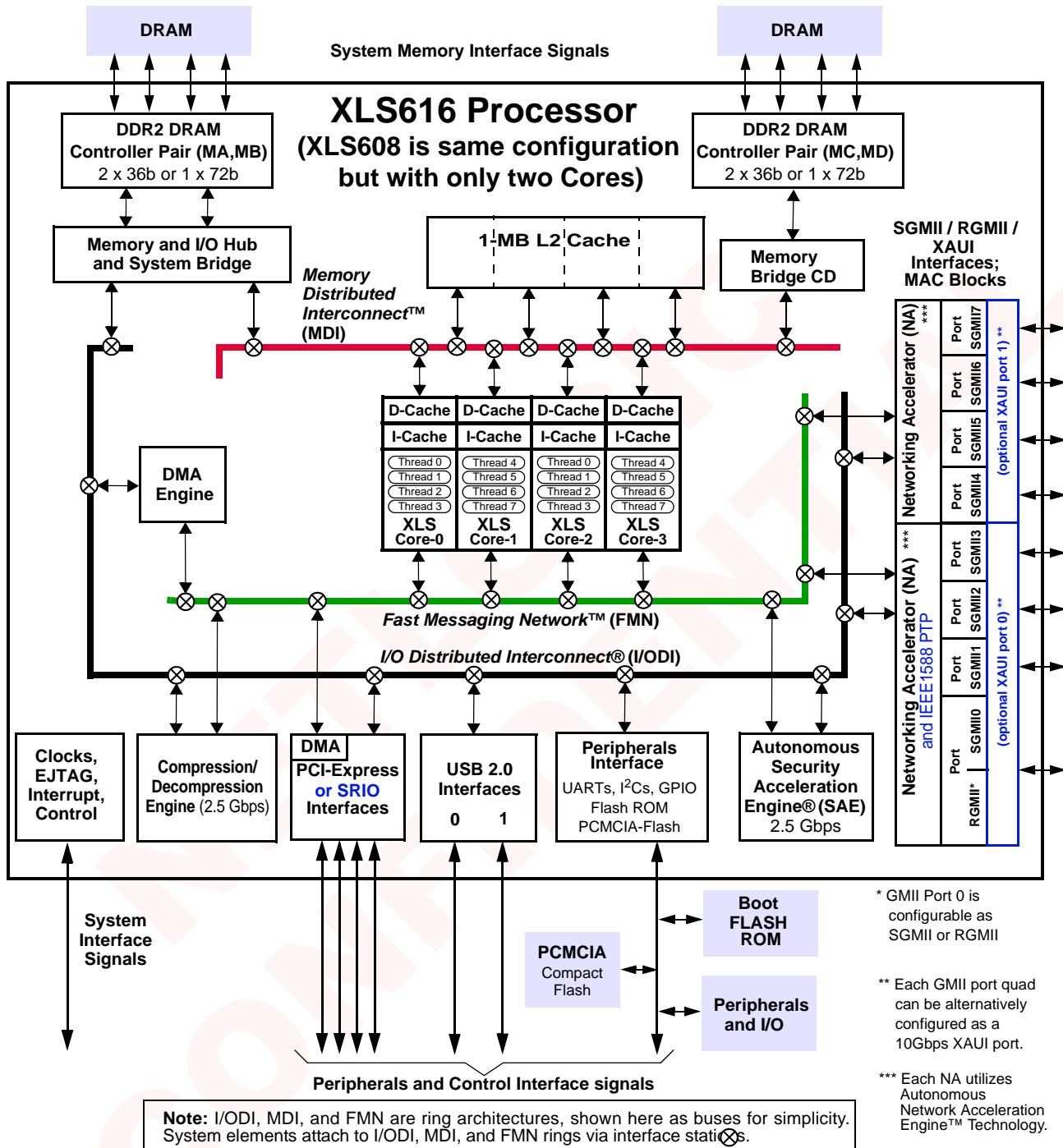


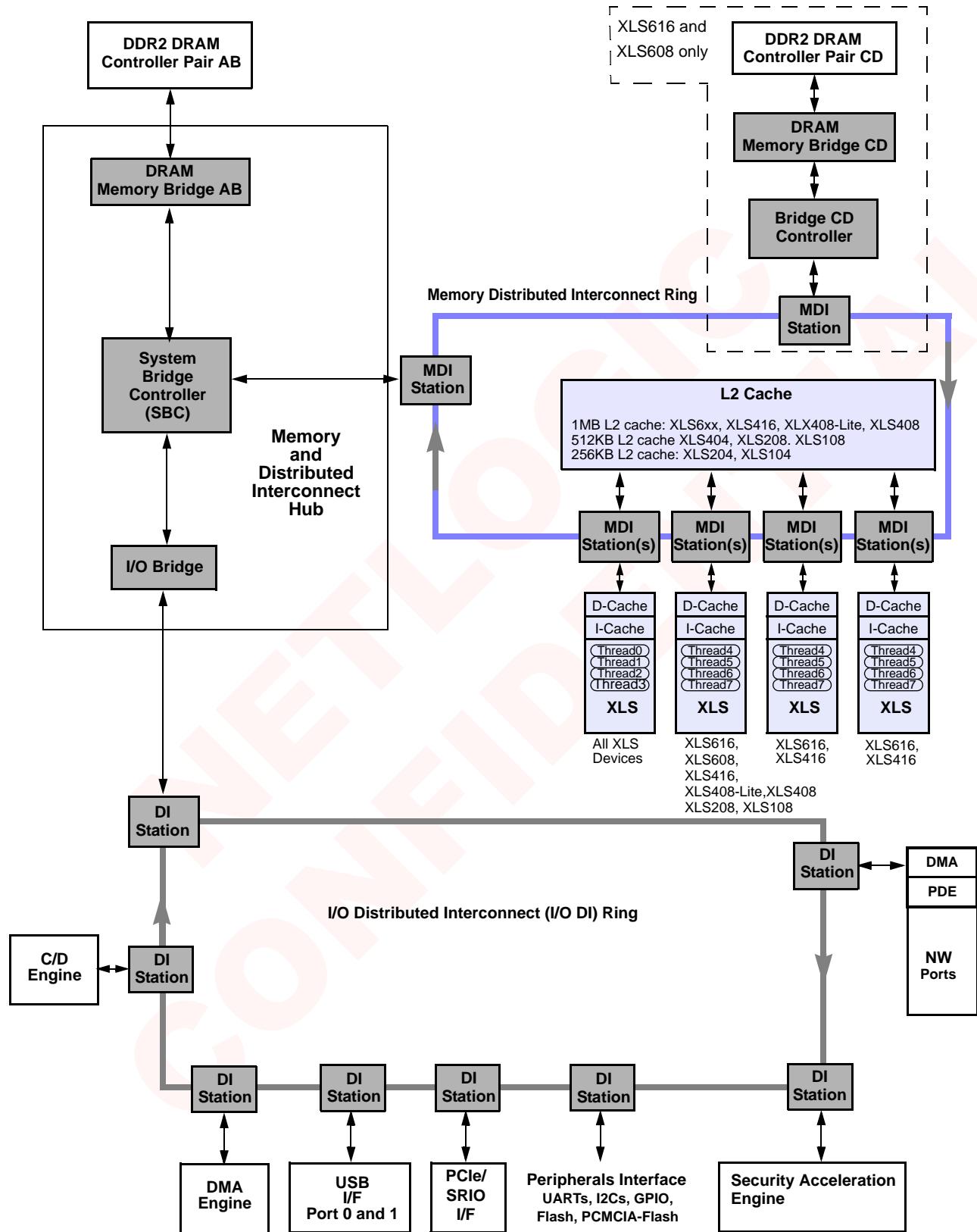
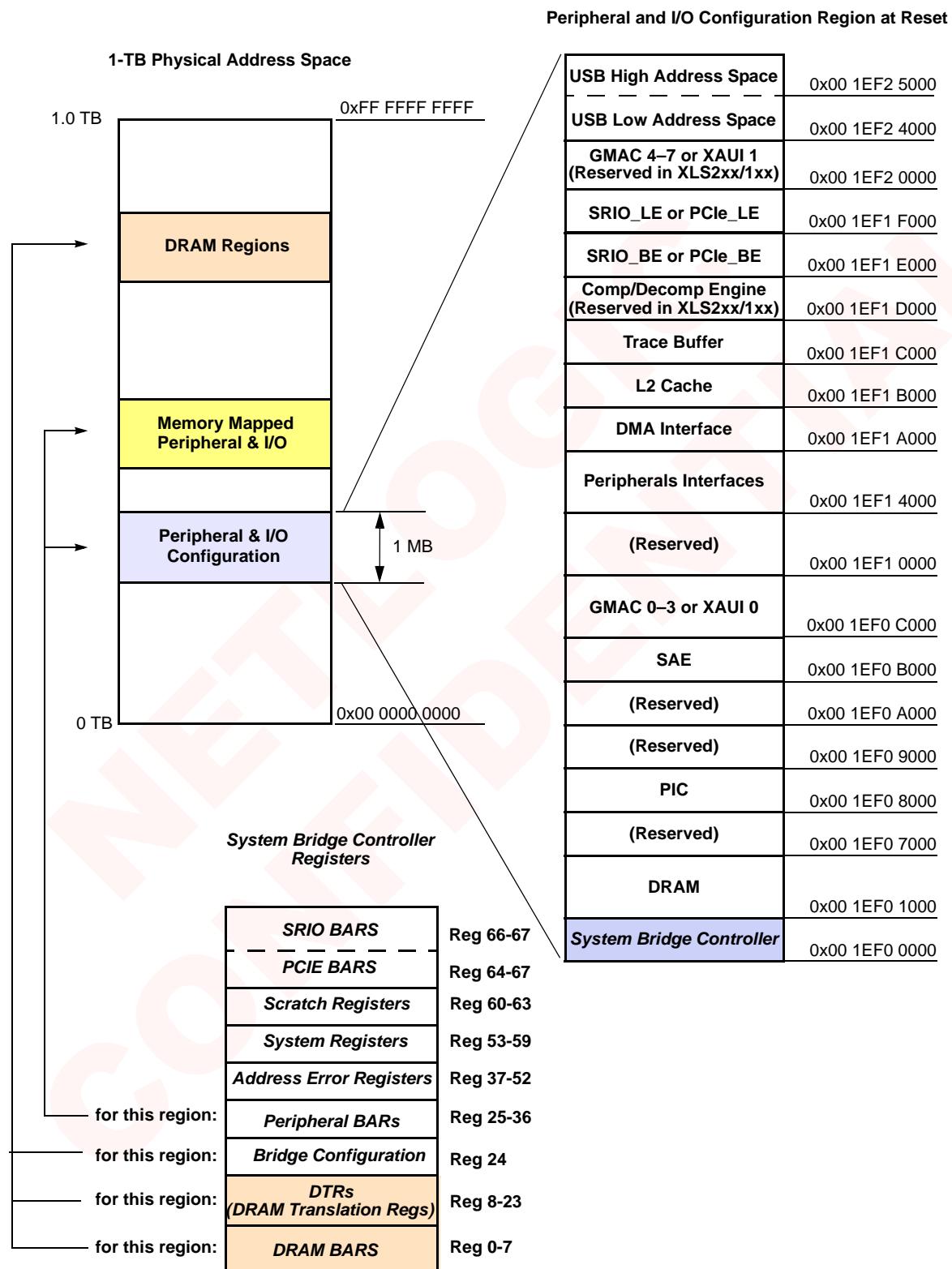
Figure 8-2. The Memory and I/O Distributed Interconnect Stations In Detail

Figure 8-3. The 1.0 TeraByte Physical Address Space on the MDI

8.4 Devices Mapped in the System Bridge Controller

The Peripheral and I/O Configuration Space is mapped into a one-MB memory region that is divided into 256 regions of 4 KB each, as shown in [Figure 8-3](#). Base addresses for memory and I/O devices can be configured by programming corresponding registers. [Table 8-1](#) shows the default addresses at reset time. The registers within each device are listed in the corresponding chapter for that device. Address mapping for all devices is done in the System Bridge Controller (SBC) in the Memory and Distributed Interconnect Hub. Note that the System Bridge Controller's base address is also programmable.

Table 8-1. Peripheral and I/O Configuration Region of Memory

Peripheral	Device Name	Sub-Region Offset	Sub-Region Offset Address at Reset
Chapter 8, “Memory and I/O Access”, Section 8.5, “System Bridge Controller Registers”			
System Bridge Controller	XLS_IO_DEV_BRIDGE	0x0 0000	0x00 1EF0 0000
Chapter 11, “DRAM Controllers and Interface”			
DDR1/DDR2 DRAM_A Channel, Port MA	XLS_IO_DEV_DDR_CHNA	0x0 1000	0x00 1EF0 1000
DDR1/DDR2 DRAM_B Channel, Port MB	XLS_IO_DEV_DDR_CHNB	0x0 2000	0x00 1EF0 2000
DDR1/DDR2 DRAM_C Channel, Port MC	XLS_IO_DEV_DDR_CHNC	0x0 3000	0x00 1EF0 3000
DDR1/DDR2 DRAM_D Channel, Port MD	XLS_IO_DEV_DDR_CHND	0x0 4000	0x00 1EF0 4000
Chapter 10, “Programmable Interrupt Controller”			
Programmable Interrupt Controller	XLS_IO_DEV_PIC	0x0 8000	0x00 1EF0 8000
Chapter 14, “Security Acceleration Engine”			
Security Acceleration Engine	XLS_IO_DEV_SAE	0x0 B000	0x00 1EF0 B000
Chapter 16, “SGMII and RGMII Interface” (When XAUI Interface_0 is not used)			
SGMII-Interface_0, Port SGMII0	XLS_IO_DEV_GMAC_0	0x0 C000	0x00 1EF0 C000
SGMII-Interface_1, Port SGMII1	XLS_IO_DEV_GMAC_1	0x0 D000	0x00 1EF0 D000
SGMII-Interface_2, Port SGMII2	XLS_IO_DEV_GMAC_2	0x0 E000	0x00 1EF0 E000
SGMII-Interface_3, Port SGMII3	XLS_IO_DEV_GMAC_3	0x0 F000	0x00 1EF0 F000
Chapter 17, “XAUI Interface” (When SGMII Interface_0/1/2/3 is not used)			
XAUI Interface_0	XLS_IO_DEV_XAUI_0	0x0 C000	0x00 1EF0 C000
Chapter 21, “UARTs, I²C, and Peripherals Interface”			
UART_1	XLS_IO_DEV_UART_1	0x1 4000	0x00 1EF1 4000
UART_2	XLS_IO_DEV_UART_2	0x1 5000	0x00 1EF1 5000
I ² C_1	XLS_IO_DEV_I2C_1	0x1 6000	0x00 1EF1 6000
I ² C_2	XLS_IO_DEV_I2C_2	0x1 7000	0x00 1EF1 7000
Chapter 23, “GPIO and the Peripherals Interface”			
General Purpose I/O (GPIO)	XLS_IO_DEV_GPIO	0x1 8000	0x00 1EF1 8000
Chapter 22, “Flash/PCMCIA Memory and the Peripherals Interface”			
Flash ROM	XLS_IO_DEV_FLASH	0x1 9000	0x00 1EF1 9000
Chapter 9, “Direct Memory Access (DMA) Engine”			
DMA	XLS_IO_DEV_DMA	0x1 A000	0x00 1EF1 A000

Table 8-1. Peripheral and I/O Configuration Region of Memory (continued)

Peripheral	Device Name	Sub-Region Offset	Sub-Region Offset Address at Reset
Chapter 7, “XLS Unified L2 Cache”			
L2 Cache	XLS_IO_DEV_L2	0x1 B000	0x00 1EF1 B000
Chapter 25, “XLS Debugging Features”, Section 24.8, “Trace Buffer”			
Trace Buffer	XLS_IO_DEV_TB	0x1 C000	0x00 1EF1 C000
Chapter 15, “Compression/Decompression Engine”			
Compression/Decompression	XLS_IO_DEV_CMP	0x1 D000	0x00 1EF1 D000
Chapter 19, “PCI Express™ Interface” (When SRIO interface is not used)			
PCI-Express_BE	XLS_IO_DEV_PCIE_BE	0x1 E000	0x00 1EF1 E000
PCI-Express_LE	XLS_IO_DEV_PCIE_LE	0x1 F000	0x00 1EF1 F000
Chapter 20, “Serial-RapidIO Interface” (When PCIe interface is not used)			
SRIO_BE	XLS_IO_DEV_SRIO_BE	0x1 E000	0x00 1EF1 E000
SRIO_LE	XLS_IO_DEV_SRIO_LE	0x1 F000	0x00 1EF1 F000
Chapter 16, “SGMII and RGMII Interface” (When XAUI Interface_1 is not used)			
SGMII-Interface_4, Port SGMII4	XLS_IO_DEV_GMAC_4	0x2 0000	0x00 1EF2 0000
SGMII-Interface_5, Port SGMII5	XLS_IO_DEV_GMAC_5	0x2 1000	0x00 1EF2 1000
SGMII-Interface_6, Port SGMII6	XLS_IO_DEV_GMAC_6	0x2 2000	0x00 1EF2 2000
SGMII-Interface_7, Port SGMII7	XLS_IO_DEV_GMAC_7	0x2 3000	0x00 1EF2 3000
Chapter 17, “XAUI Interface” (When SGMII Interface_4/5/6/7 is not used)			
XAUI Interface_1	XLS_IO_DEV_XAUI_1	0x2 0000	0x00 1EF2 0000
Chapter 18, “USB Interface”			
USB Interface Low Address Space	XLS_IO_DEV_USB_A	0x2 4000	0x00 1EF2 4000
USB Interface High Address Space	XLS_IO_DEV_USB_B	0x2 5000	0x00 1EF2 5000

8.5

System Bridge Controller Registers

Figure 8-3 shows the SBC Configuration region. SBC register “I/O Base Address” contains the base address for the SBC region. At power-on, this register contains physical address 0x1EF0 0000 (virtual address 0xFFFF FFFF BEF0 0000).

For convenience,

- A Region is assigned an address offset relative to “I/O Base Address.”
- A register within a Region is assigned an address offset relative to the Region address.
- Address offsets are expressed in bytes.

The physical byte address for a register for a particular XLS device is formed as follows:

SBC Register Physical Address			
Address bits	39:20	19:12	11:0
Source	from register “IO Base Address”	Region Address Offset	Register Address Offset

8.5.1

System Bridge Controller Read-Access Requirements

To ensure execution of intended register read operations and prevent potential CPU hangs, simultaneous (time-overlap) read access of System Bridge registers and DRAM/TB/PIC control registers must be avoided. Software must ensure BOTH of the following:

1. Only one nCPU may access these registers at any given time. This can be accomplished by establishing a semaphore to grant or deny System Bridge register read access. Any nCPU (thread) requiring access to these registers must first acquire the capability (via the semaphore), after which it may issue read commands.
2. If any given nCPU intends to issue a System Bridge register read access followed immediately by a read of any of the DRAM/TB/PIC control registers, care must be taken that the System Bridge register read is complete before any DRAM/TB/PIC control register reads are issued.

Likewise, if an nCPU intends to issue a DRAM/TB/PIC control register read access followed immediately by a read of a System Bridge register, care must be taken that the DRAM/TB/PIC control register read is complete before the System Bridge register read is issued.

In each case the completion of the initial read must be ensured prior to allowing the second read. This can be accomplished by using the contents of the register first read (for example by adding zero to the value) before allowing the second access.

8.5.2 System Bridge Controller Registers Summary

Table 8-2 shows the register set for the System Bridge Controller.

Table 8-2. System Bridge Controller Register Summary

Reg. ID (Hex)	Addr. Offset (Hex)	Register Name	Description	R/W	Reset Value
Section 8.6, “DRAM Configuration Registers”					
DRAM Base Address Registers - Section 8.6.1, “DRAM_BAR[0-7]”					
0x000 to 0x007	0x000 to 0x01C	see DRAM_BAR[0-7]	DRAM Base Address Registers 0 to 7	R/W	0x0000 0000
Memory Translation Channels A & C Registers - Section 8.6.2, “DRAM_CHNAC_DTR[0-7]”					
0x008	0x020	see DRAM_CHNAC_DTR[0-7]	DRAM Translation Register 0 for Channels A & C	R/W	0x0000 0000
0x009	0x024	see DRAM_CHNAC_DTR[0-7]	DRAM Translation Register 1 for Channels A & C	R/W	0x0000 0000
0x00A	0x028	see DRAM_CHNAC_DTR[0-7]	DRAM Translation Register 2 for Channels A & C	R/W	0x0000 0000
0x00B	0x02C	see DRAM_CHNAC_DTR[0-7]	DRAM Translation Register 3 for Channels A & C	R/W	0x0000 0000
0x00C	0x030	see DRAM_CHNAC_DTR[0-7]	DRAM Translation Register 4 for Channels A & C	R/W	0x0000 0000
0x00D	0x034	see DRAM_CHNAC_DTR[0-7]	DRAM Translation Register 5 for Channels A & C	R/W	0x0000 0000
0x00E	0x038	see DRAM_CHNAC_DTR[0-7]	DRAM Translation Register 6 for Channels A & C	R/W	0x0000 0000
0x00F	0x03C	see DRAM_CHNAC_DTR[0-7]	DRAM Translation Register 7 for Channels A & C	R/W	0x0000 0000
Memory Translation Channels B & D Registers - Section 8.6.3, “DRAM_CHNBD_DTR[0-7]”					
0x010	0x040	see DRAM_CHNBD_DTR[0-7]	DRAM Translation Register 0 for Channels B & D	R/W	0x0000 0000
0x011	0x044	see DRAM_CHNBD_DTR[0-7]	DRAM Translation Register 1 for Channels B & D	R/W	0x0000 0000
0x012	0x048	see DRAM_CHNBD_DTR[0-7]	DRAM Translation Register 2 for Channels B & D	R/W	0x0000 0000
0x013	0x04C	see DRAM_CHNBD_DTR[0-7]	DRAM Translation Register 3 for Channels B & D	R/W	0x0000 0000
0x014	0x050	see DRAM_CHNBD_DTR[0-7]	DRAM Translation Register 4 for Channels B & D	R/W	0x0000 0000
0x015	0x054	see DRAM_CHNBD_DTR[0-7]	DRAM Translation Register 5 for Channels B & D	R/W	0x0000 0000
0x016	0x058	see DRAM_CHNBD_DTR[0-7]	DRAM Translation Register 6 for Channels B & D	R/W	0x0000 0000
0x017	0x05C	see DRAM_CHNBD_DTR[0-7]	DRAM Translation Register 7 for Channels B & D	R/W	0x0000 0000
Memory Bridge Configuration Register - Section 8.6.4, “DRAM_BRIDGE_CFG”					
0x018	0x060	DRAM_BRIDGE_CFG	DRAM Ports Configuration	R/W	See Definition.
Section 8.7, “Peripherals, I/O and I/O Memory Base Address Registers”					
0x019	0x064	XLS_IO_BAR	I/O Configuration Base Address Register	R/W	See Definition
0x01A	0x068	FLASH_BAR	Flash Memory Base Address Register	R/W	See Definition
0x01B to 0x024	0x06C to 0x090	Reserved	Reserved	R/W	0x0000 0000
Section 8.9, “Address Error Registers” and see Chapter 25, “XLS Debugging Features”					
0x025	0x094	Address Error Device Mask	Available device register	R/W	See Definition.
0x026	0x098	AERR0_LOG1	Address Error Interrupt Information log#1	R/W	0x0000 0000
0x027	0x09C	AERR0_LOG2	Address-Low Error Interrupt log#2	R/W	0x0000 0000
0x028	0x0A0	AERR0_LOG3	Address-High Error Interrupt log#3	R/W	0x0000 0000
0x029	0x0A4	AERR0_DEVSTAT	Address Error Source/Cause	R/W	0x0000 0000
0x02A	0x0A8	AERR1_LOG1	Address Error Information log#1	R/W	0x0000 0000

Table 8-2. System Bridge Controller Register Summary (continued)

Reg. ID (Hex)	Addr. Offset (Hex)	Register Name	Description	R/W	Reset Value
0x02B	0x0AC	AERR1_LOG2	Address-Low Error log#2	R/W	0x0000 0000
0x02C	0x0B0	AERR1_LOG3	Address-High Error log#3	R/W	0x0000 0000
0x02D	0x0B4	AERR1_DEVSTAT	Address Error NMI Source	R/W	0x0000 0000
0x02E	0x0B8	AERR0_EN	Address Error Source-Mask Enables	R/W	0x0000 0000
0x02F	0x0BC	AERR0_UPG	Address Error Priority NMI Source Upgrade	R/W	0x0000 0000
0x030	0x0C0	AERR0_CLEAR	Address Error Clear	R/W	0x0000 0000
0x031	0x0C4	AERR1_CLEAR	Address Error NMI Clear	R/W	0x0000 0000
0x032	0x0C8	SBE_COUNTS	Single-bit error counters	R/W	0x0000 0000
0x033	0x0CC	DBE_COUNTS	Double-bit error counters	R/W	0x0000 0000
0x034	0x0D0	BITERR_INT_EN	Enable bit error detection interrupts	R/W	0x0000 0000
Section 8.10, "System Credits and Counter Registers"					
0x035	0x0D4	SYS2IO_CREDITS	System to I/O credits for I/O DI	R/W	See Definition
0x036	0x0D8	EVENT_CNT_CNTRL1	Event Counter 1 event selection	R/W	0x0000 0000
0x037	0x0DC	EVENT_CNTR1	Counter 1	R/W	0x0000 0000
0x038	0x0E0	EVENT_CNT_CNTRL2	Event Counter 2 event selection	R/W	0x0000 0000
0x039	0x0E4	EVENT_CNTR2	Counter 2	R/W	0x0000 0000
0x03A	0x0E8	Reserved	Reserved	-	-
0x03B	0x0EC	MISC FUNCTION CTL1	Controls JTAG access to L2 cache for debug use; controls ordering modes for transactions with certain devices; sets SBC's PCIe Addressing translation mode	R/W	0x0000 0000
Section 8.11.1, "SCRATCH[0-3]"					
0x03C to 0x03F	0x0F0 to 0x0FC	SCRATCH[0-3]	General purpose scratch registers 0 to 3	R/W	0x0000 0000
Section 8.12, "PCIe Base Address Registers" when SRIO Interface is not used					
0x040	0x100	PCIE_CFG_BAR	PCIe configuration Base Address Register	R/W	0x0000 0000
0x041	0x104	PCIE_ECFG_BAR	PCIe Extended Configuration Base Address Register	R/W	0x0000 0000
0x042	0x108	PCIE_MEM_BAR	PCIe Memory Base Address Register	R/W	0x0000 0000
0x043	0x10C	PCIE_IO_BAR	PCIe I/O Base Address Register	R/W	0x0000 0000
Section 8.13, "SRIO Base Address Registers" when PCIe Interface is not used					
0x042	0x108	SRIO_MEM_BAR	SRIO Memory Base Address Register	R/W	0x0000 0000
0x043	0x10C	SRIO_IO_BAR	SRIO I/O Base Address Register	R/W	0x0000 0000
Section 8.14, "Additional Device Mask Register"					
0x044	0x100	DEVICE_MASK2	Companion to available device register, covering additional devices	R/W	See Definition

8.6 DRAM Configuration Registers

One of the key functions performed by the bridge is processing DRAM requests. This includes:

- Distributing incoming DRAM memory read or write transactions to the four DRAM memory controllers for processing, and
- Address translation between the physical address space and DRAM address space.

The parameters used to perform these functions are user programmable. Twenty-five control registers are provided for this purpose: BRIDGE_CFG, DRAM_BAR[0-7], DRAM_CHNAC_DTR[0-7], and DRAM_CHNBD_DTR[0-7]. The fields within these registers are described in subsequent sections.

The twenty-five [DRAM_BAR\[0-7\]](#), [DRAM_CHNAC_DTR\[0-7\]](#), [DRAM_CHNBD_DTR\[0-7\]](#) and [DRAM_BRIDGE_CFG](#) registers shown in the lower corner of [Figure 8-4](#) configure the mapping of the DRAM channels into up to eight separate regions in the physical address space. DRAM may be partitioned into eight separate regions. Each region may be assigned a range in the physical address space. This range is defined as a contiguous address block starting from a programmable base address. The size of a region may be between 16 MB and 32 GB, with the condition that their address ranges must not overlap. Physically, access to the DRAM is gained through four channels, referred to as Channel A, Channel B, Channel C and Channel D. There is a set of DRAM registers common to Channels A and C, and another set common to Channels B and D. [Figure 8-4](#) illustrates an example with three DRAM regions.

When processing DRAM transactions, there are a few steps involved. [Figure 8-5](#) illustrates the sequence of steps required during a Read/Write transaction on a specified DRAM location.

1. Matching Physical Address with a Region Base address:
 - a. The DRAM location's physical address specified in the transaction is used to determine the corresponding DRAM Region.
 - b. Each DRAM Region Base Address register defines "Base Address" and "Address Mask" bit fields, which specify the starting address and the sizes of the region.
 - c. The specified Physical Address bits [35:24], masked out with "Address Mask" field, are compared against the non-masked bits of the "Base Address" field. If a match is detected and the region is enabled, a hit is declared, indicating that the specified address resides in this region. However, if a hit is detected on multiple regions, the transaction is declared faulty. Once a hit is detected, the matching bits in the physical address are zeroed, leaving bits [36:24] as the only significant field.
2. A DRAM memory controller is selected to process this transaction:
 - a. If the Interleaving-Scheme field of the Base Address register that was hit is 0, 1, 2, or 3, the destination controller is evident.
 - b. Otherwise, the Interleave-Scheme field of the "Bridge Configuration" register is used to extract a 1-bit or 2-bit wide portion of the transaction's physical address. The binary value of this portion is then used in conjunction with the "Interleaving Scheme" field to select a destination controller. Since the interleave bits are removed in this case, the remaining significant portion of the physical address is between 35 and 22 bits wide.
3. Determining the region partition:

The corresponding Address Translation register is determined. The "Divisions" field determines the number of divisions of the region. The "Position" field determines the bit offset of the address bit to be manipulated. "Divisions" and "Position" fields are used to determine the "Partition" field, which is then inserted in the physical address. The significant portion of the physical address is now bits [35:22].

4. The physical address is 32-byte-aligned: Since DRAM transactions are aligned with 32-byte boundaries, address bits [4:0] are cleared. The significant portion of the physical address is now bits [31:17].
5. The effective DRAM address obtained is then passed to the selected DRAM controller: Within the memory controller, the DRAM address will be further partitioned into a bank address, a rank address, a row address, and a column address. See ["DDR_PARAMS_3" on page 350.](#) for details.

Note: The DRAM controller accepts a maximum address width of 29 bits. Consequently, certain value combinations in the "Base Address" and "DRAM Address Translation" registers are illegal.

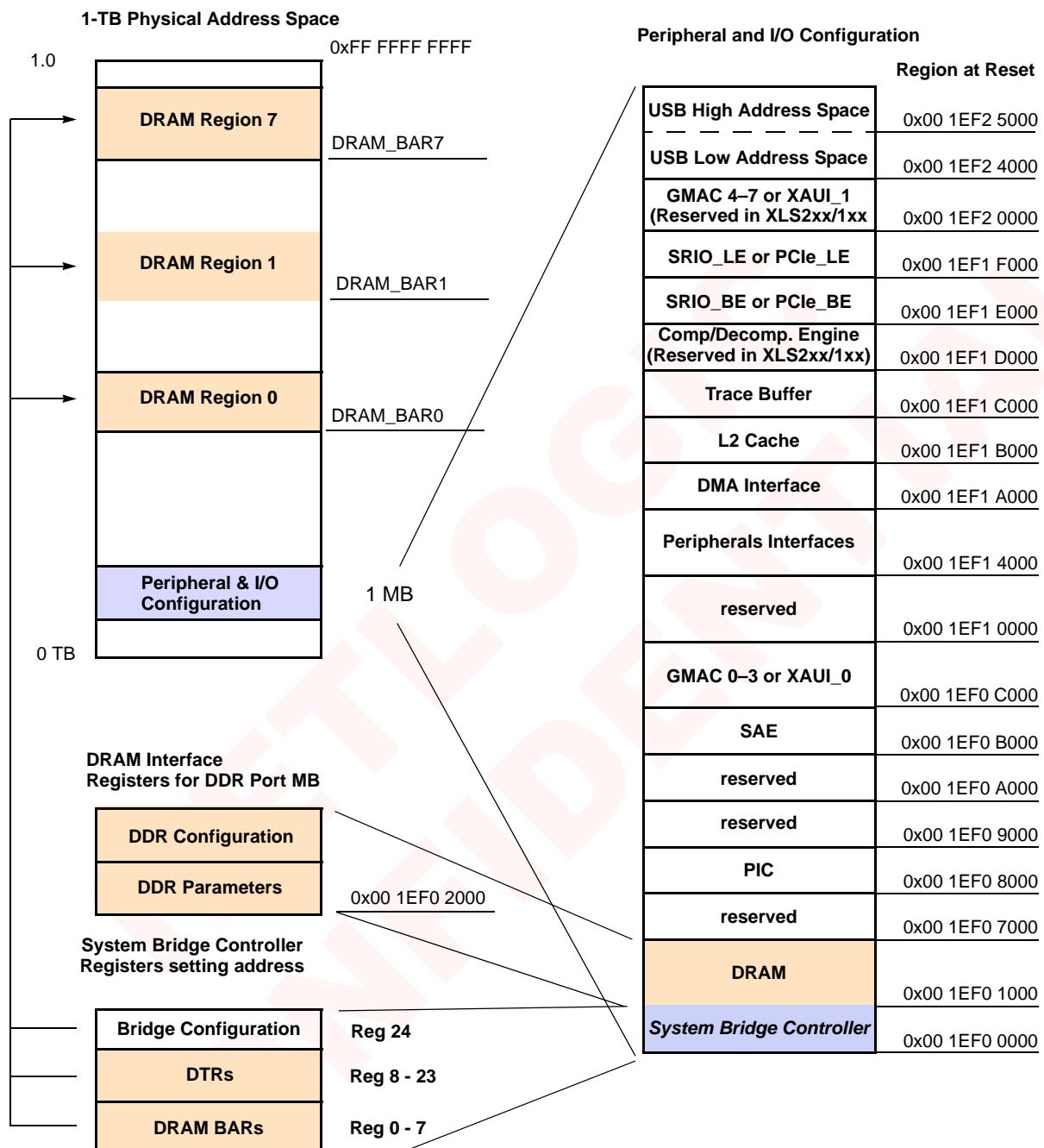
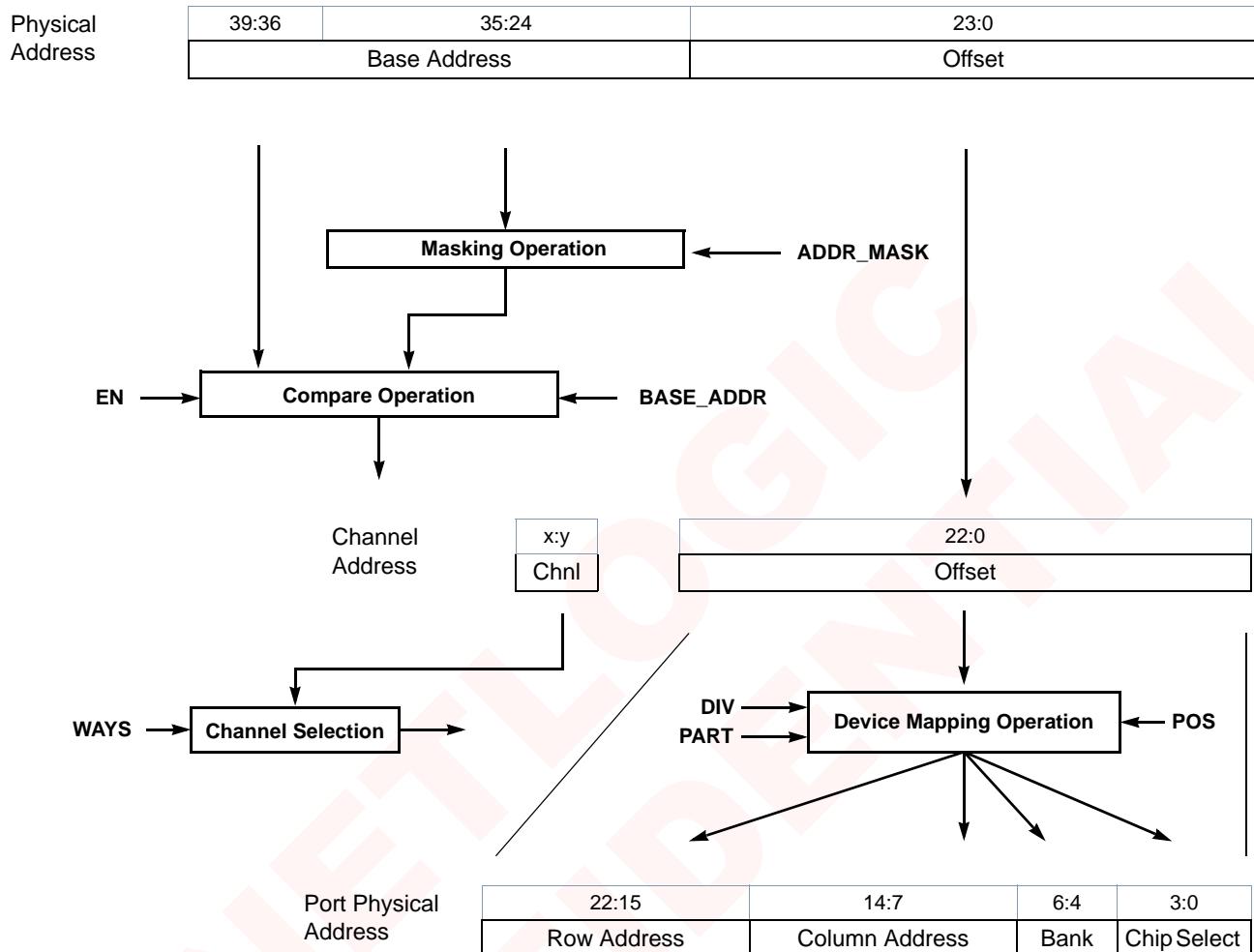
Figure 8-4. Example of Three DRAM Port MB Regions in the Address Space on the MDI

Figure 8-5. DRAM Channel Address Mapping in the SBC

8.6.1 DRAM_BAR[0-7]

Register ID: 0x000 – 0x007
Address Offset: 0x000 – 0x01C

DRAM Region Base Address Registers

These eight registers are used to define up to eight DRAM Regions. Each register defines an aligned address range within the 40-bit physical address space. This address range is mapped to the DRAM address space. Each Region Base Address register contains the following fields: start address, size, alignment and interleaving.

Note that DRAM Region Base Address registers must be programmed such that there is no overlap between the defined regions. The minimum size is 16MB and maximum size is 32GB.

[Figure 8-4](#) is an example of three DRAM regions mapped in the 1-TByte address space.

The reset value for all the DRAM BARs is 0x0000 0000.

31:16		15:4	3:1	0	
BASE_ADDR		ADDR_MASK	INTERLEAVE_MODE	STATUS	
Bits	Field Name	Field Description		R/W	Reset
31:16	BASE_ADDR	Defines bits [39:24] of the base physical address of the region.		R/W	0x0000
15:4	ADDR_MASK	Mask bits to increase the size of the region. The mask maps to bits [35:24] of the base physical address. When a mask bit is set to '1', the corresponding bit of the BAR's base physical address is excluded from the same bit of the transaction's physical address. 0x000: 16-MB region 0x001: 32-MB region 0x003: 64-MB region 0x007: 128-MB region 0x00F: 256-MB region 0x01F: 512-MB region 0x03F: 1-GB region 0x07F: 2-GB region 0x0FF: 4-GB region 0x1FF: 8-GB region 0x3FF: 16-GB region 0x7FF: 32-GB region (Only legal if interleaved across 2 or 4 channels) 0xFFFF: 64-GB region (Only legal if interleaved across 4 channels)		R/W	0x000
3:1	INTERLEAVE_MODE	Interleaving scheme for this region. 0x0: Channel A only 0x1: Channel B only 0x2: Channel C only 0x3: Channel D only 0x4: Channel A and Channel C 0x5: Channel B and Channel D 0x6,0x7: Channels A, B, C, and D *Note: For XLS2xx and XLS1xx devices, these bits are READ ONLY bits, with values b000		R/W RO*	b000
0	STATUS	Enable address matching using this BAR 0: Region disabled 1: Region enabled		R/W	b0

8.6.2 DRAM_CHNAC_DTR[0-7]

Register ID: 0x008 – 0x00F
Address Offset: 0x020 – 0x03C

DRAM Region Channels AC Address Translation Registers

There are eight DRAM Region Address Translation registers for Channels A and C. Each register defines parameters to translate a physical address into DRAM address used at the device level. The reset value is 0x0000 0000.

31:14		13:12	11:10	9:8	7:6	5:0	
Reserved		PARTITION	Reserved	DIVISIONS	Reserved	POS	
Bits	Field Name	Field Description				R/W	Reset
31:14	Reserved	Reserved Reset value = b0000 0000 0000 0000 00				RO	see Field Description
13:12	PARTITION	DRAM device partition. Value to be inserted into the physical address bit or bit-pair being manipulated. 0: b0 or b00 (1Q) 1: b0 or b01 (2Q) 2: b1 or b10 (3Q) 3: b1 or b11 (4Q)				R/W	b00
11:10	Reserved	Reserved				RO	b00
9:8	DIVISIONS	Number of physical address bits to be manipulated. (DRAM device divisions) 00: None (whole DRAM) 01: One bit (half DRAM) 10: Two bits (quarter DRAM) 11: Two bits (same as '10')				R/W	b00
7:6	Reserved	Reserved				RO	b00
5:0	POSITION	Position of physical address bit to be manipulated. Bit position specified should account for any interleave bits that were removed from the physical address. If two bits are to be manipulated, this field specifies the position of the less significant bit.				R/W	b00 0000

8.6.3 DRAM_CHNBD_DTR[0-7]

Register ID: 0x010 – 0x017

Address Offset: 0x040 – 0x05C

DRAM Region Channels BD Address Translation registers (not applicable to XLS1xx)

These eight DRAM translation registers specify the address manipulations (if any) to be performed for the regions defined in the eight corresponding BARs, for memory requests destined for channels B or D. The reset value is 0x0000 0000.

31:14		13:12	11:10	9:8	7:6	5:0	
Reserved		PARTITION	Reserved	DIVISIONS	Reserved	POS	
Bits	Field Name	Field Description				R/W	Reset
31:14	Reserved	Reserved Reset value = b0000 0000 0000 0000 00				RO	see Field Description
13:12	PARTITION	Region partition. Value to be inserted into the physical address bit or bit-pair for translation. 00: b0 or b00 (1Q) 01: b0 or b01 (2Q) 10: b1 or b10 (3Q) 11: b1 or b11 (4Q)				R/W	b00
11:10	Reserved	Reserved				RO	b00
9:8	DIVISIONS	Number of physical address bits to be manipulated. (DRAM device divisions) 00: None (whole DRAM) 01: One bit (half DRAM) 10: Two bits (quarter DRAM) 11: Two bits (same as '10')				R/W	b00
7:6	Reserved	Reserved				RO	b00
5:0	POSITION	Position of physical address bit to be manipulated. Bit position specified should account for any interleave bits that were removed from the physical address. If two bits are to be manipulated, this field specifies the position of the less significant bit.				R/W	b00 0000

8.6.4 DRAM_BRIDGE_CFG

Register ID: 0x018
Address Offset: 0x060

This register in the System Bridge Controller (SBC) configures the DRAM bridges for both Memory Controller Pairs AB and CD.

31:13	12	11	10:8	7:5	4	3:2	1:0
RSVD	CHANNEL_MODE	RSVD	INTERLEAVE_MODE	RSVD	BUS_MODE	RSVD	DRAM_MODE

Bits	Field Name	Field Description	R/W	Reset
31:13	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0	RO	see Field Description
12	CHANNEL_MODE	Allows interleaving across channels 0 & 1 when only channels 0 & 1 exist. 0: available channels = channels 0, 1, 2, 3 (default) 1: available channels = channels 0, 1 * Note: For XLS2xx and XLS1xx devices, since there is only one DRAM controller, this bit is a READ ONLY bit with a value of b0.	R/W RO*	0
11	Reserved	Reserved	RO	b0
10:8	INTERLEAVE_MODE	Defines the interleave bits, i.e. the physical address bits that are used to determine the DRAM controller to which the memory request should be sent. 0: physical addr bits [6:5] if 4-way interleaving, physical addr bit [5] if 2-way interleaving. 1: physical addr bits [7:6] if 4-way interleaving, physical addr bit [6] if 2-way interleaving. 2: physical addr bits[8:7] if 4-way interleaving, physical addr bit [7] if 2-way interleaving. 3: physical addr bits [9:8] if 4-way interleaving, physical addr bit [8] if 2-way interleaving. 4, 5, 6, 7: physical addr bits [10:9] if 4-way interleaving, physical addr bit [9] if 2-way interleaving. * Note: For XLS2xx and XLS1xx devices, since there is only one DRAM controller, these bits are READ ONLY bits with value b000.	R/W RO*	b000
7:5	Reserved	Reserved	RO	b000
4	BUS_MODE	Defines Channel configuration. 0: Four 32- or 36-bit channels 1: Two 64- or 72-bit channel-pairs * Note: For XLS2xx and XLS1xx devices, this bit should always be programmed as '1'. Do not clear this bit to '0'.	R/W*	b1
3:2	Reserved	Reserved	RO	b00
1:0	DRAM_MODE	DRAM Controller mode. 00: do not use this setting for normal operation. 01: Sets controller for DDR2 operation. 10: do not use this setting for normal operation. 11: do not use this setting for normal operation.	R/W	b01

8.6.4.1 Example of DRAM Address Processing

The following example assumes a system with 2GB of DRAM memory, configured as two 72-bit channel pairs. The following register contents are assumed:

```
DRAM_BAR0 0x000000F9  
DRAM_BAR1 0x001F0009  
DRAM_BAR2 0x002001F9  
DRAM_BAR3 0x004003F9  
DRAM_BAR4 0x00800079  
DRAM_BAR5 0x00880039  
DRAM_BAR6 0x008C0019  
DRAM_BAR7 0x008E0009  
  
DRAM_CHNAC_DTR0 0x0000211B  
DRAM_CHNAC_DTR1 0x00000017  
DRAM_CHNAC_DTR2 0x0000211C  
DRAM_CHNAC_DTR3 0x0000211D  
DRAM_CHNAC_DTR4 0x0000211A  
DRAM_CHNAC_DTR5 0x00002119  
DRAM_CHNAC_DTR6 0x00002118  
DRAM_CHNAC_DTR7 0x00002117  
  
DRAM_BRIDGE_CFG 0x00000011
```

(continued on next page)

Table 8-3. Key to Address Processing Symbols Used in Example

Symbol	Meaning
-	Removed during physical address truncation.
I	Removed due to interleaving.
x	May have any value.
0	Forced to 0 during BAR match or by address manipulation.
1	Forced to 1 during address manipulation.

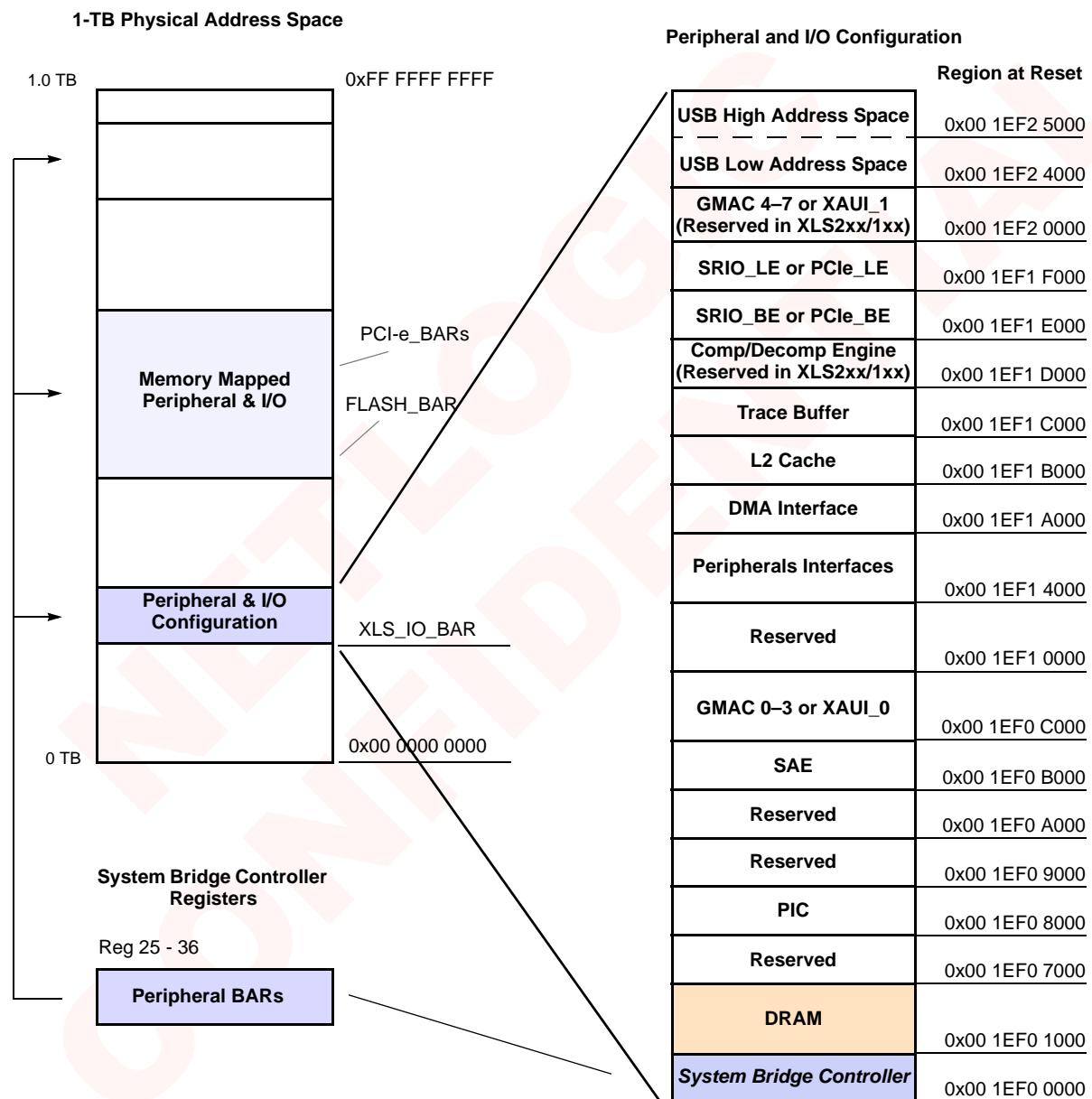
Table 8-4. DRAM Example

BAR base	BAR size	Physical Address Map	DRAM_CHNAC_DTR	Physical address processing Sent to DRAM controller:
				<- bit 34 through -->bit 6
				3333333333 2222222222 1111111111 9876543210 9876543210 9876543210 9876543210
0000	256M	0x00_0000_0000 0x00_0FFF_FFFF	PART=2 DIV=1 POS=27	0000000000 01xxxxxxxx xxxxxxxxxx xxxxI----
Not mapped		0x00_1000_0000 0x00_1EFF_FFFF		
001F	16M	0x00_1F00_0000 0x00_1FFF_FFFF	PART=0 DIV=1 POS=23	0000000000 000000xxxx xxxxxxxxxx xxxxI----
0020	512M	0x00_2000_0000 0x00_3FFF_FFFF	PART=2 DIV=1 POS=28	0000000000 1xxxxxxxxx xxxxxxxxxx xxxxI----
0040	1G	0x00_4000_0000 0x00_7FFF_FFFF	PART=2 DIV=1 POS=29	0000000001 xxxxxxxxxx xxxxxxxxxx xxxxI----
0080	128M	0x00_8000_0000 0x00_87FF_FFFF	PART=2 DIV=1 POS=26	0000000000 001xxxxxxxx xxxxxxxxxx xxxxI----
0088	64M	0x00_8800_0000 0x00_8BFF_FFFF	PART=2 DIV=1 POS=25	0000000000 0001xxxxxxxx xxxxxxxxxx xxxxI----
008C	32M	0x00_8C00_0000 0x00_8DFF_FFFF	PART=2 DIV=1 POS=24	0000000000 00001xxxxx xxxxxxxxxx xxxxI----
008E	16M	0x00_8E00_0000 0x00_8EFF_FFFF	PART=2 DIV=1 POS=23	0000000000 000001xxxx xxxxxxxxxx xxxxI----

8.7 Peripherals, I/O and I/O Memory Base Address Registers

These configuration registers set the base addresses for the Peripheral & I/O Configuration region, external peripheral, memory and I/O interfaces in the Memory Mapped Peripheral & I/O region of the physical address space as shown in [Figure 8-6](#).

Figure 8-6. Peripherals, I/O and I/O Memory Base Address Registers on the MDI



8.7.1 XLS_IO_BAR

Register ID: 0x019
Address Offset: 0x064

This Base Address Register determines the start address of the 1-megabyte Peripheral & I/O Configuration region. This register is permanently enabled, and at startup it contains a predefined base address 0x00 1EF0 0000.

31:12		11:2	1	0
BASE_ADDR		Reserved		OV EN
Bits	Field Name	Field Description	R/W	Reset
31:12	BASE_ADDR	Maps physical address bits 39:20	R/W	0x001E F
11:2	Reserved	Reserved Reset value = b0000 0000 00	RO	See Field Descr.
1	OV	The OVERLAY bit allows two BARs to map to the same address. If there is a conflict or collision between the IO_BAR and the DRAM_BAR, the IO_BAR takes precedence. 0: (default) 1: Overlapping address matches of the XLS_IO_BAR and any other BAR are ignored, and the XLS_IO_BAR match has precedence.	R/W	b0
0	EN	Enables the I/O BAR. Read-only. 1: (default) Always enabled.	RO	b1

8.7.2 FLASH_BAR

Register ID: 0x01A
Address Offset: 0x068

This Base Address Register determines the start address and the size of the 16 MB to 16 GB Flash Memory region in the Memory Mapped Peripheral & I/O region. Always enabled so that at reset it always starts at 0x00 1F00 0000.

Note that all unmasked address bits defined in the BASE_ADDR field are set to zero in the Flash memory address space.

31:16		15:5	4:1	0
BASE_ADDR		ADDR_MASK	Reserved	EN
Bits	Field Name	Field Description	R/W	Reset
31:16	BASE_ADDR	Maps physical address bits 39:24	R/W	0x001F
15:5	ADDR_MASK	Mask bits for physical address bits 34:24. All unmasked address bits are set to zero in the Flash memory address space Reset value = b0000 0000 000	R/W	See Field Desc
4:1	Reserved	Reserved	RO	b0 000
0	EN	Enables the Flash BAR. Enabled at reset. 1: Enabled (default)	R/W	b1

8.8 Functional Blocks Without Separate Base Address Registers

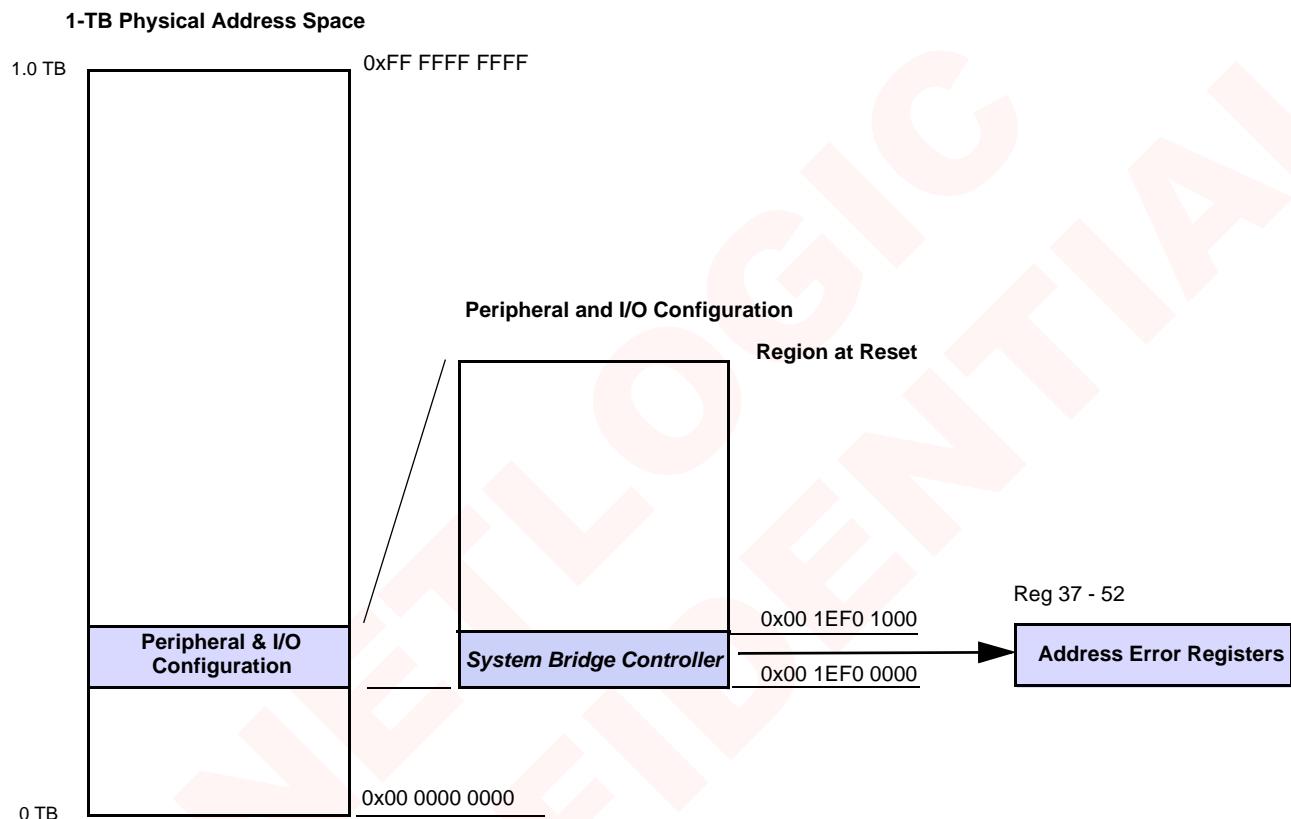
The following functional blocks do not have their own separate Base Address Registers. Instead, they lie at fixed offsets from the start of the Peripherals & I/O base; the location of this base is adjustable using XLS_IO_BAR.

- Programmable Interrupt Controller (PIC)
- Security Acceleration Engine (SAE)
- GMAC 0 through 7 – The GMACs reside in two separate “quads.” Quad0 contains GMAC0 through GMAC3, and Quad1 contains GMAC4 through GMAC7 (not supported in XLS2xx)
 - The XLS2xx devices only use GMAC0 through GMAC3
 - The XLS1xx devices only use GMAC0 and GMAC1
 - The Optional XAUI ports, when configured, use the same offsets as the GMAC quads. XAUI_0 uses the same offset as Quad0, and XAUI_1 uses the same offset as Quad1
- DMA Engine
- L2 Cache
- Compression/Decompression Engine (CDE)
- Trace Buffer

8.9 Address Error Registers

These sixteen 32-bit registers control and indicate address error conditions caused by devices around the MDI. The System Bridge Controller monitors transactions across the MDI and upon detecting fault conditions, it captures relevant data associated to the offending device. These registers reside in the System Bridge Controller region of the MDI address space as shown in Figure 8-7.

Figure 8-7. Address Error Registers for Devices on the MDI



The System Bridge Controller identifies the following error conditions:

- Invalid Address – The offending transaction provided a physical address that does not belong to any of the regions defined by the Base Address Registers.
- Multiple address match – The offending transaction provided a physical address that belongs to multiple regions defined by the Base Address Registers.
- Illegal Cached Access – The offending transaction attempted a cached access to an un-cached space.
- Disabled Device Access – The offending transaction attempted to access disabled device.
- L1 Cache Tag Error – The offending transaction encountered uncorrectable bit error in an L1 Cache tag address.
- L2 Cache Tag Error – The offending transaction encountered uncorrectable bit error in an L2 Cache tag

The error device source register (AERR_DEVID) is a map that enables error detection on transactions from desired devices.

Once an error is detected, data associated with the error is captured on the corresponding set of error registers. There are two sets of error registers: Address Error Set 0 (AERR0) and Address Error Set 1 (AERR1). Address Error Set 0 is always enabled to capture error data. Address Error Set 1 must be explicitly enabled in the Address Error Priority Upgrade register.

8.9.1 DEVICE_MASK

Register ID: 0x025

Address Offset: 0x094

Address Error Device Mask

This 32-bit Read/Write register contains a bit map representing all devices on the MDI. This register is used to activate/deactivate address error detection on the desired devices.

Some bits such as the PIC field are determined by hardwired connections while others are maintained by software. Values of some bits at reset depend upon specific XLS Family Members.

31	30	29	28	27	26	25	24
GMAC_7	GMAC_6	GMAC_5	GMAC_4	L2	DMA	FLASH	GPIO
23	22	21	20	19	18	17	16
I2C_2	I2C_1	UART_2	UART_1	Reserved	Reserved	Reserved	Reserved
15	14	13	12	11	10	9	8
GMAC_3	GMAC_2	GMAC_1	GMAC_0	SAE	Reserved	Reserved	PIC
7	6	5	4	3	2	1	0
Reserved	Reserved	Reserved	DDR_CHND	DDR_CHNC	DDR_CHNB	DDR_CHNA	BRIDGE

Bits	Field Name	Field Description	R/W	Reset
31	GMAC_7	SGMII7 Error Detection Mode. Controlled by SW (set to '0' in XAUI_1 port) 0: Error Detection Deactivated 1: Error Detection Activated (Note: This bit is Reserved and READ ONLY in XLS2xx and XLS1xx devices.)	R/W RO*	-
30	GMAC_6	SGMII6 Error Detection Mode. Controlled by SW (set to '0' in XAUI_1 port) 0: Error Detection Deactivated 1: Error Detection Activated (Note: This bit is Reserved and READ ONLY in XLS2xx and XLS1xx devices.)	R/W RO*	-
29	GMAC_5	SGMII5 Error Detection Mode. Controlled by SW (set to '0' in XAUI_1 port) 0: Error Detection Deactivated 1: Error Detection Activated (Note: This bit is Reserved and READ ONLY in XLS2xx and XLS1xx devices.)	R/W RO*	-
28	GMAC_4 or XAUI_1	SGMII4 Error Detection Mode. Controlled by SW (sets up Error-detection mode in XAUI_1 port) 0: Error Detection Deactivated 1: Error Detection Activated (Note: This bit is Reserved and READ ONLY in XLS2xx and XLS1xx devices.)	R/W RO*	-
27	L2	L2 Cache Error Detection Activated. Read-only and always set to b1	RO	b1
26	DMA	DMA Engine Error Detection Mode. Read-only and always set to b1	RO	b1
25	FLASH	Flash Memory Error Detection Mode. Set/cleared by software. 0: Error Detection DeActivated 1: Error Detection Activated	R/W	-
24	GPIO	General Purpose I/O Error Detection Mode. Set/cleared by software. 0: Error Detection DeActivated 1: Error Detection Activated	R/W	-

Bits	Field Name	Field Description	R/W	Reset
23	IIC_2	I ² C Port 2 Error Detection Mode. Set/cleared by software. 0: Error Detection DeActivated 1: Error Detection Activated	R/W	-
22	IIC_1	I ² C Port 1 Error Detection Mode. Set/cleared by software. 0: Error Detection DeActivated 1: Error Detection Activated	R/W	-
21	UART_2	UART Port 2 Error Detection Mode. Set/cleared by software. 0: Error Detection DeActivated 1: Error Detection Activated	R/W	-
20	UART_1	UART Port 1 Error Detection Mode. Set/cleared by software. 0: Error Detection DeActivated 1: Error Detection Activated	R/W	-
19:16	Reserved	Reserved	RO	-
15	GMAC_3	SGMII3 Error Detection Mode. Set/cleared by SW (set to '0' in XAUI_0 port) 0: Error Detection DeActivated 1: Error Detection Activated (Note: This bit is Reserved and READ ONLY in XLS1xx devices.)	R/W RO*	-
14	GMAC_2	SGMII2 Error Detection Mode. Set/cleared by SW (set to '0' in XAUI_0 port) 0: Error Detection DeActivated 1: Error Detection Activated (Note: This bit is Reserved and READ ONLY in XLS1xx devices.)	R/W RO*	-
13	GMAC_1	SGMII1 Error Detection Mode. Set/cleared by SW (set to '0' in XAUI_0 port) 0: Error Detection DeActivated 1: Error Detection Activated	R/W	-
12	GMAC_0 or XAUI_0	SGMII0 Error Detection Mode. Set/cleared by SW (sets up Error-detection mode in XAUI_0 port) 0: Error Detection DeActivated 1: Error Detection Activated	R/W	-
11	SAE	SAE Error Detection Mode. Set/cleared by software if present, always cleared if not present. 0: Error Detection DeActivated 1: Error Detection Activated	R/W	-
10:9	Reserved	Reserved	R/W	-
8	PIC	Programmable Interrupt Controller active. Read-only and always set. 1: Error Detection always Activated	RO	b1
7:5	Reserved	Reserved	RO	-
4	DDR_CHND	DDR Channel D Error Detection Mode Determined by DRAM_MODE and BUS_MODE fields of BRIDGE_CFG register 0: Error Detection DeActivated (DRAM_MODE=1) 1: Error Detection Activated (DRAM_MODE=0)	RO	b0
3	DDR_CHNC	DDR Channel C Error Detection Mode Determined by DRAM_MODE and BUS_MODE fields of BRIDGE_CFG register 0: Error Detection DeActivated (DRAM_MODE=1) 1: Error Detection Activated (DRAM_MODE=0)	RO	b0
2	DDR_CHNB	DDR Channel B Error Detection Mode Determined by DRAM_MODE and BUS_MODE fields of BRIDGE_CFG register 1: Error Detection Activated (DRAM_MODE=1/0)	RO	b1
1	DDR_CHNA	DDR Channel A Error Detection Mode Determined by DRAM_MODE and BUS_MODE fields of BRIDGE_CFG register 1: Error Detection Activated (DRAM_MODE=1/0)	RO	b1
0	BRIDGE	System Bridge active. Read-only and always set. 1: Error Detection always Activated	RO	b1

8.9.2 AERR0_LOG1

Register ID: 0x026
 Address Offset: 0x098

Address Error Set 0 Log 1

When the System Bridge Controller detects an address error, this register captures data associated with the offending transaction.

- Command executed by the transaction
- Device ID that generated the transaction
- Element which detected the error
- Error Detection indication

31:19	18:16	15:13	12:8	6:4	3:1	0
Reserved	AERR_CMD	Reserved	AERR_DEVID	AERR_STAT	Reserved	AERR_VALID

Bits	Field Name	Field Description	R/W	Reset
31:19	Reserved	Reserved Reset value = b0000 0000 0000 0	RO	See Field Descr.
18:16	AERR_CMD	Command field of offending memory request 0x0: Idle 0x1: Write 0x2: read 0x3: read exclusive 0x4: Upgrade 0x5: Invalidate 0x6: undefined 0x7: Write error	R/W	b000
15:13	Reserved	Reserved	RO	b000

Bits	Field Name	Field Description	R/W	Reset
12:8	AERR_DEVID	Device ID of offending memory request [0-1]: Device = CPU[n] [2-7]: Device = Reserved [8-11]: Device = L2[n] [12-15]: Device = Reserved 16: Device = Reserved 17: Device = Reserved 18: Device = Reserved 19: Device = any of GMAC Quad 0 (GMACs 0 to 3) or XAUI_0 port 20: Device = SAE 21: Device = GPIO 22: Device = Reserved 23: Device = Reserved 24: Device = XLS System DMA Engine 25: Device = CDE (Comp/Decomp. Engine) (<i>Rsvd</i> in XLS2xx and XLS1xx) 26: Device = PCIe 27: Device = USB 28: Device = any of GMAC Quad 1 (GMACs 4 to 7) or XAUI_1 port (<i>Reserved</i> in XLS2xx and XLS1xx) [29-31]: Device = Reserved	RO	b0 0000
7	Reserved	Reserved	RO	b0
6:4	AERR_STAT	Indicates which device detected the address error: bit 6 = System Bridge Controller bit 5 = L1 tag bit 4 = L2UC tag	R/W	b000
3:1	Reserved	Reserved	RO	b000
0	AERR_VALID	Address error indicator status 0: Not Detected 1: Detected	R/W	b0

8.9.3 AERR0_LOG2

Register ID: 0x027

Address Offset: 0x09C

Address Error Set 0 Log 2

When the System Bridge Controller detects an address error, this register captures the low order bits of the physical address provided by the offending transaction. The reset value is 0x0000 0000.

31:0
AERR_ADDRLO

Bits	Field Name	Field Description	R/W	Reset
31:0	AERR_ADDRLO	Physical address low-order bits [36:5] of offending address	R/W	0x0000 0000

8.9.4 AERR0_LOG3

Register ID: 0x028
 Address Offset: 0x0A0

Address Error Set 0 Log 3

When the System Bridge Controller detects an address error, this register captures the high order bits of the physical address provided by the offending transaction. The reset value is 0x0000 0000

		31:3	2:0	
		Reserved	AERR_ADDRHI	
Bits	Field Name	Field Description	R/W	Reset
31:3	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0000 0000 0	RO	See Field Desc.
2:0	AERR_ADDRHI	Physical address high-order bits [39:37] of offending address	R/W	b000

8.9.5 AERR0_DEVSTAT

Register ID: 0x029
Address Offset: 0x0A4

Address Error Set 0 Interrupt Status

When the System Bridge Controller detects an address error, it may raise an interrupt according to the settings in the Address Error Interrupt Enable register. When the device involved in the offending transaction has interrupt enabled, an interrupt is generated and the corresponding interrupt status bit in this register is set. Otherwise, an interrupt is not generated and the corresponding interrupt status bit in this register is not set.

The reset value is 0x0000 0000.

31:21	20	19	18	17	16	15:14	13	12	11	10:9	8	7:2	1:0
RSVD	GMAC_1	USB	PCIE	CDE	DMA	RSV	GPIO	SAE	GMAC_0	RSVD	L2	RSVD	CPU n

Bits	Field Name	Field Description	R/W	Reset
31:21	Reserved	Reserved	RO	0x0000
20	GMAC_1_AERR or XAUI_1_AERR	Quad 1 GMAC Address Error Interrupt status (GMACs 4 to 7, and optional XAUI_1 port) 0: inactive 1: active Note: This bit is Reserved in the XLS2xx and XLS1xx devices.	R/W	b0
19	USB_AERR	USB Address Error Interrupt status 0: inactive 1: active	R/W	b0
18	PCIE_AERR	PCIe Address Error Interrupt status	R/W	b0
17	CDE_AERR	CDE Address Error Interrupt status 0: inactive 1: active Note: This bit is Reserved in the XLS2xx and XLS1xx devices.	R/W	b0
16	DMA_AERR	DMA Engine Address Interrupt status (not the SAE DMA or CDE DMA)	R/W	b0
15:14	Reserved	Reserved	R/W	b0
13	GPIO	GPIO Address Error Interrupt status	R/W	b0
12	SAE	SAE Address Error Interrupt status	R/W	b0
11	GMAC_0_AERR or XAUI_1_AERR	Quad 0 GMAC Address Error Interrupt status (GMACs 0 to 3, and optional XAUI_0 port) 0: inactive 1: active	R/W	b0
10:9	Reserved	Reserved	R/W	b0
8	L2	AERR interrupt by L2 device (any bank)	R/W	b0
7:2	Reserved	Reserved	R/W	b0
1	CPU1_AERR	CPU 1 Address Error Interrupt status	R/W	b0
0	CPU0_AERR	CPU 0 Address Error Interrupt status	R/W	b0

8.9.6 AERR1_LOG1

Register ID: 0x02A
Address Offset: 0x0A8

Address Error Set 1 Log 1

Address Error registers Set 1 are used to capture address error events designated to have a higher priority level. When the System Bridge Controller detects an address error, this register captures data associated with the offending transaction.

- Command executed by the transaction
- Device ID that generated the transaction
- Element which detected the error
- Error Detection indication

31:19	18:16	15:13	12:8	6:4	3:1	0
Reserved	AERR_CMD	Reserved	AERR_DEVID	AERR_STAT	Reserved	AERR_VALID
Field Description						
31:19	Reserved	Reserved Reset value = b0000 0000 0000 0			RO	See Field Descr.
18:16	AERR_CMD	Command field of offending memory request 0x0: Idle 0x1: Write 0x2: read 0x3: read exclusive 0x4: Upgrade 0x5: Invalidate 0x6: undefined 0x7: Write error			R/W	b000
15:13	Reserved	Reserved			RO	b000

Bits	Field Name	Field Description	R/W	Reset
12:8	AERR_DEVID	Device ID of offending memory request [0-1]: Device = CPU[n] [2-7]: Device = Reserved [8-11]: Device = L2[n] [12-15]: Device = Reserved 16: Device = Reserved 17: Device = Reserved 18: Device = Reserved 19: Device = any of GMAC Quad 0 (GMACs 0 to 3) or XAUI_0 port 20: Device = SAE 21: Device = GPIO 22: Device = Reserved 23: Device = Reserved 24: Device = XLS System DMA Engine 25: Device = CDE (Comp/Decomp. Engine) (RSVD in XLS2xx and XLS1xx devices) 26: Device = PCIe 27: Device = USB 28: Device = any of GMAC Quad 1 (GMACs 4 to 7) or XAUI_1 port (RSVD in XLS2xx and XLS1xx devices) [29-31]: Device = Reserved	RO	b0 0000
7	Reserved	Reserved	RO	b0
6:4	AERR_STAT	Indicates which device detected the address error: bit 6 = System Bridge Controller bit 5 = L1 tag bit 4 = L2UC tag	R/W	b000
3:1	Reserved	Reserved	RO	b000
0	AERR_VALID	Set on detection of address error 0: Not Detected 1: Detected	R/W	b0

8.9.7 AERR1_LOG2

Register ID: 0x02B

Address Offset: 0x0AC

Address Error Set 1 Log 2

When the System Bridge Controller detects an address error, this register captures the low-order bits of the physical address provided by the offending transaction. The reset value is 0x0000 0000

31:0
AERR_ADDRLO

Bits	Field Name	Field Description	R/W	Reset
31:0	AERR_ADDRLO	Physical address low-order bits [36:5] of offending address	R/W	0x0000 0000

8.9.8 AERR1_LOG3

Register ID: 0x02C
 Address Offset: 0x0B0

Address Error Set 1 Log 3

When the System Bridge Controller detects an address error, this register captures the high order bits of the physical address provided by the offending transaction. The reset value is 0x0000 0000.

		31:3	2:0	
		Reserved	AERR_ADDRHI	
Bits	Field Name	Field Description	R/W	Reset
31:3	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0000 0000 0	RO	see Field Desc
2:0	AERR_ADDRHI	Physical address high-order bits [39:37] of offending address	R/W	b000

8.9.9 AERR1_DEVSTAT

Register ID: 0x02D
Address Offset: 0x0B4

Address Error Set 1 Interrupt Status

When the System Bridge Controller detects an address error, it may raise an interrupt according to the settings in the Address Error Interrupt Enable register. When the device involved in the offending transaction has its interrupt enabled, an interrupt is generated and the corresponding interrupt status bit in this register is set. Otherwise, an interrupt is not generated and the corresponding interrupt status bit in this register is not set.

The reset value is 0x0000 0000

31:21	20	19	18	17	16	15:14	13	12	11	10:9	8	7:2	1:0
RSVD	GMAC_1	USB	PCIE	CDE	DMA	RSVD	GPIO	SAE	GMAC_0	RSVD	L2	RSVD	CPU n

Bits	Field Name	Field Description	R/W	Reset
31:21	Reserved	Reserved	RO	0x0000
20	GMAC_1_AERR or XAUI_1_AERR	Quad 1 GMAC Address-Error Interrupt status (GMACs 4 to 7, and optional XAUI_1 port) 0: inactive 1: active Note: This bit is Reserved in the XLS2xx and XLS1xx devices.	R/W	b0
19	USB_AERR	USB Address-Error Interrupt status 0: inactive 1: active	R/W	b0
18	PCIE_AERR	PCIE Address-Error Interrupt status 0: inactive 1: active	R/W	b0
17	CDE_AERR	CDE Address-Error Interrupt status 0: inactive 1: active Note: This bit is Reserved in the XLS2xx and XLS1xx devices.	R/W	b0
16	DMA_AERR	DMA Address-Error Interrupt status (not the SAE DMA or CDE DMA) 0: inactive 1: active	R/W	b0
15	Reserved	Reserved	R/W	b0
14	Reserved	Reserved	R/W	b0
13	GPIO_AERR	USB Address-Error Interrupt status 0: inactive 1: active	R/W	b0
12	SAE_AERR	Security Acceleration Engine Address-Error Interrupt status 0: inactive 1: active	R/W	b0
11	GMAC_0_AERR or XAUI_0_AERR	Quad 0 GMAC Address-Error Interrupt status (GMACs 0 to 3, and optional XAUI_0 port) 0: inactive 1: active	R/W	b0
10:9	Reserved	Reserved	R/W	b0

Bits	Field Name	Field Description	R/W	Reset
8	L2_AERR	L2 Cache Address-Error Interrupt status (any bank) 0: inactive 1: active	R/W	b0
7:2	Reserved	Reserved	R/W	b0
1	CPU1_AERR	CPU1 Address-Error Interrupt status 0: inactive 1: active	R/W	b0
0	CPU0_AERR	CPU0 Address-Error Interrupt status 0: inactive 1: active	R/W	b0

8.9.10 AERR0_EN

Register ID: 0x02E

Address Offset: 0x0B8

Address-Error Interrupt Enable register

When the System Bridge Controller detects Address Errors, it uses this register to enable/disable the generation of interrupts. The reset value is 0x0000 0000

31:21	20	19	18	17	16	15
RSV	GMAC1_AERR0_EN	USB_AERR0_EN	PCIE_AERR0_EN	CDE_AERR0_EN	DMA_AERR0_EN	RSV

14	13	12	11	10	9	8	7:2	1	0
RSV	GIO_AERR0_EN	SAE_AERR0_EN	GMAC0_AERR0_EN	RSV	RSV	L2_AERR0_EN	RSV	CPU1_AERR0_EN	CPU0_AERR0_EN

Bits	Field Name	Field Description	R/W	Reset
31:21	Reserved	Reserved Reset value = b0000 0000 0000 000	RO	See Field Desc
20	GMAC1_AERR0_EN or XAUI_1_AERR0_EN	Quad 1 GMAC Address-Error Interrupt enable (GMACs 4 to 7, and optional XAUI_1 port) 0: disabled 1: enable Note: This bit is Reserved in the XLS2xx and XLS1xx devices.	R/W	
19	USB_AERR0_EN	USB Address-Error Interrupt enable 0: disabled 1: enable	R/W	b0
18	PCIE_AERR0_EN	PCIE Address-Error Interrupt enable 0: disabled 1: enable	R/W	b0

Bits	Field Name	Field Description	R/W	Reset
17	CDE_AERR0_EN	Compression/Decompression Engine Address-Error Interrupt enable 0: disabled 1: enable Note: This bit is <i>Reserved</i> in the XLS2xx and XLS1xx devices.	R/W	b0
16	DMA_AERR0_EN	DMA Engine Address-Error Interrupt enable 0: disabled 1: enable	R/W	b0
15:14	<i>Reserved</i>	<i>Reserved</i>	RO	b00
13	GIO_AERR0_EN	GPIO Address-Error Interrupt enable 0: disabled 1: enable	R/W	b0
12	SAE_AERR0_EN	Enable Security Acceleration Engine Address-Error Interrupt enable 0: disabled 1: enable	R/W	b0
11	GMAC0_AERR0_EN or XAUI_0_AERR0_EN	Quad 0 GMAC Address-Error Interrupt enable (GMACs 0 to 3, and optional XAUI_0 port) 0: disabled 1: enable	R/W	b0
10:9	<i>Reserved</i>	<i>Reserved</i>	RO	b00
8	L2_AERR0_EN	L2 Cache Address-Error Interrupt enable (GMACs 0 to 3) 0: disabled 1: enable	R/W	b0
7:2	<i>Reserved</i>	<i>Reserved</i>	R/W	b0000 00
1	CPU1_AERR0_EN	CPU1 Address-Error Interrupt enable 0: disabled 1: enable	R/W	b0
0	CPU0_AERR0_EN	CPU0 Address-Error Interrupt enable 0: disabled 1: enable	R/W	b0

8.9.11 AERR0_UPG

Register ID: 0x02F
Address Offset: 0x0BC

Address-Error Set 0 Upgrade

This register may be configured to allow certain AERR0 interrupts to be upgraded to AERR1. The corresponding bits in the Address-Error Interrupt enabled register (AERR0_EN) must also be set. Correspondingly, the PIC IRT must also be programmed to trigger interrupts to a desired CPU. The reset value is 0x0000 0000.

31:21		20	19	18		17		16		15
RSV	GMAC1_AERR1_UPG	USB_AERR1_UPG	PCIE_AERR1_UPG	CDE_AERR1_UPG	DMA_AERR1_UPG	RSV				
14	13	12	11	10	9	8	7:2	1	0	
RSV	GIO_AERR1_UPG	SAE_AERR1_UPG	GMAC0_AERR1_UPG	RSV	RSV	L2_AERR1_UPG	RSV	CPU1_AERR1_UPG	CPU0_AERR1_UPG	
Bits	Field Name		Field Description						R/W	Reset
31:21	Reserved		Reserved Reset value = b0000 0000 0000 0000						RO	See Field Desc
20	GMAC1_AERR1_UPG or XAUI1_AERR1_UPG		Quad 1 GMAC Address-Error Interrupt upgrade enable (GMACs 4 to 7, or XAUI_1 port) 0: disable 1: enable Note: This bit is Reserved in the XLS2xx and XLS1xx devices.						R/W	b0
19	USB_AERR1_UPG		USB Address-Error Interrupt upgrade enable 0: disable 1: enable						R/W	b0
18	PCIE_AERR1_UPG		PCIe Address-Error Interrupt upgrade enable 0: disable 1: enable						R/W	b0
17	CDE_AERR1_UPG		Compression/Decompression Engine Address-Error Interrupt upgrade enable 0: disable 1: enable Note: This bit is Reserved in the XLS2xx and XLS1xx devices.						R/W	b0
16	DMA_AERR1_UPG		DMA Engine Address-Error Interrupt upgrade enable 0: disable 1: enable						R/W	b0
15:14	Reserved		Reserved						RO	b00
13	GIO_AERR1_UPG		GPIO Address-Error Interrupt upgrade enable 0: disable 1: enable						R/W	b0
12	SAE_AERR1_UPG		Security Acceleration Engine Address-Error Interrupt upgrade enable 0: disable 1: enable						R/W	b0

Bits	Field Name	Field Description	R/W	Reset
11	GMAC0_AERR1_UPG or XAUI0_AERR1_UP	Quad 0 GMAC Address-Error Interrupt upgrade enable (GMACs 0 to 3, or XAUI_0 port) 0: disable 1: enable	R/W	b0
10:9	Reserved	Reserved	RO	b00
8	L2_AERR1_UPG	L2 Cache Address-Error Interrupt upgrade enable 0: disable 1: enable	R/W	b0
7:2	Reserved	Reserved	R/W	b0000 00
1	CPU1_AERR1_UPG	CPU1 Address-Error Interrupt upgrade enable 0: disable 1: enable	R/W	b0
0	CPU0_AERR1_UPG	CPU0 Address-Error Interrupt upgrade enable 0: disable 1: enable	R/W	b0

8.9.12 AERR0_CLEAR

Register ID: 0x030

Address Offset: 0x0C0

Address Error Set 0 Clear

This Read/Write register is used to clear AERR0 interrupts. Write any value to bit[0] to clear. The reset value is 0x0000 0000.

31:1		0
	Reserved	CLEAR

Bits	Field Name	Field Description	R/W	Reset
31:1	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0000 0000 0000	RO	See Field Desc
0	CLEAR	Writing '0' or '1' to this bit clears the AERR0_DEVSTAT, AERR0_LOG1, AERR0_LOG2 and AERR0_LOG3 registers.	R/W	b0

8.9.13 AERR1_CLEAR

Register ID: 0x031
Address Offset: 0x0C4

Address Error Set 1 Clear

This Read/Write register is used to clear AERR1 interrupts. Write any value to bit[0] to clear AERR1 interrupts. The reset value is 0x0000 0000.

31:1	0
<i>Reserved</i>	CLEAR

Bits	Field Name	Field Description	R/W	Reset
31:1	Reserved	<i>Reserved</i> Reset value = b0000 0000 0000 0000 0000 0000 0000 000	RO	See Field Desc
0	CLEAR	Writing '0' or '1' to this bit clears the AERR1_DEVSTAT, AERR1_LOG1, AERR1_LOG2 and AERR1_LOG3 registers.	R/W	b0

SBE_COUNTS

Register ID: 0x032
Address Offset: 0x0C8

Single-Bit Error Counts

Single-bit errors may occur upon transactions on the L1 Cache and L2 Cache. When single-bit errors are detected, this register is used to keep counts of the number of occurrences.

In addition, if any of the counts becomes non-zero and the corresponding bit is set in the Bit Error Interrupt register (BITERR_INT_EN), the System Bridge Controller raises an interrupt.

The reset value is 0x0000 0000.

31	30	29:24	23	22	21:16
Reserved	L2DATA_SBE_OVF	L2DATA_SBE	Reserved	L2TAG_SBE_OVF	L2TAG_SBE

15	14	13:8	7	6	5:0
Reserved	L1DATA_SBE_OVF	L1DATA_SBE	Reserved	L1TAG_ERR_OVF	L1TAG_ERR

Bits	Field Name	Field Description	R/W	Reset
31	Reserved	<i>Reserved</i>	RO	b0
30	L2DATA_SBE_OVF	L2 Cache Single Bit Data Error Count overflow	RO	b0
29:24	L2DATA_SBE	L2 Cache Single Bit Data Error Count	RO	b00 0000
23	Reserved	<i>Reserved</i>	RO	b0
22	L2TAG_SBE_OVF	L2 Cache Single Bit Tag Error Count overflow	RO	b0
21:16	L2TAG_SBE	L2 Cache Single Bit Tag Error Count	RO	b00 0000
15	Reserved	<i>Reserved</i>	RO	b0
14	L1DATA_SBE_OVF	L1 Cache Single Bit Data Error Count overflow	RO	b0
13:8	L1DATA_SBE	L1 Cache Single Bit Data Error Count	RO	b00 0000
7	Reserved	<i>Reserved</i>	RO	b0

Bits	Field Name	Field Description	R/W	Reset
6	L1TAG_ERR_OVF	L1Cache Single Bit Tag Error Count overflow	RO	b0
5:0	L1TAG_ERR	L1Cache Single Bit Tag Error Count	RO	b00 0000

8.9.15 DBE_COUNTS

Register ID: 0x033
 Address Offset: 0x0CC

Double-Bit Error Counts

Double-bit errors may occur upon transactions on the L1 Cache and L2 Cache. When double-bit errors are detected, this register is used to keep a count of the number of occurrences.

In addition, when any of the counts in this register becomes non-zero and the corresponding bit is set in the Bit Error Interrupt register (BITERR_INT_EN), the System Bridge Controller raises an interrupt. The reset value is 0x0000 0000.

31	30	29:24	23	22	21:16
<i>Reserved</i>	L2DATA_DBE_OVF	L2DATA_DBE	<i>Reserved</i>	L2TAG_DBE_OVF	L2TAG_DBE
15	14	13:8		7:0	
<i>Reserved</i>	L1DATA_DBE_OVF	L1DATA_DBE		<i>Reserved</i>	

Bits	Field Name	Field Description	R/W	Reset
31	<i>Reserved</i>	<i>Reserved</i>	RO	b0
30	L2DATA_DBE_OVF	L2 Cache Double-Bit Data Error Count overflow	RO	b0
29:24	L2DATA_DBE	L2 Cache Double-Bit Data Error Count	RO	b00 0000
23	<i>Reserved</i>	<i>Reserved</i>	RO	b0
22	L2TAG_DBE_OVF	L2 Cache Double-Bit Tag Error Count overflow	RO	b0
21:16	L2TAG_DBE	L2 Cache Double-Bit Tag Error Count	RO	b00 0000
15	<i>Reserved</i>	<i>Reserved</i>	RO	b0
14	L1DATA_DBE_OVF	L1 Cache Double-Bit Data Error Count overflow	RO	b0
13:8	L1DATA_DBE	L1 Cache Double-Bit Data Error Count	RO	b00 0000
7:0	<i>Reserved</i>	<i>Reserved</i>	RO	0x00

8.9.16 BITERR_INT_EN

Register ID: 0x034
Address Offset: 0x0D0

Bit Error Interrupt Enable

Single and Double-bit errors may occur upon transactions on the L1 Cache and L2 Cache. When single/double-bit errors are detected, the System Bridge Controller uses this register to enable/disable raising interrupts. The reset value is 0x0000 0000.

31:8	7	6	5	4	3	2	1	0	
Reserved	L2DATA_DBE_INT_EN	L2TAG_DBE_INT_EN	L1DATA_DBE_INT_EN	RSVD	L2DATA_SBE_INT_EN	L2TAG_SBE_INT_EN	L1DATA_SBE_INT_EN	L1TAG_ERR_INT_EN	
Bits	Field Name		Field Description					R/W	Reset
31:8	Reserved		Reserved					RO	0x0000 0000
7	L2DATA_DBE_INT_EN		L2 Cache Double-Bit Data Error Interrupt enable 0: disable 1: enable					R/W	b0
6	L2TAG_DBE_INT_EN		L2 Cache Double-Bit Tag Error Interrupt enable 0: disable 1: enable					R/W	b0
5	L1DATA_DBE_INT_EN		L1 Cache Double-Bit Data Error Interrupt enable 0: disable 1: enable					R/W	b0
4	Reserved		Reserved					RO	b0
3	L2DATA_SBE_INT_EN		L2 Cache Single-Bit Data Error Interrupt enable 0: disable 1: enable					R/W	b0
2	L2TAG_SBE_INT_EN		L2 Cache Single-Bit Tag Error Interrupt enable 0: disable 1: enable					R/W	b0
1	L1DATA_SBE_INT_EN		L1 Cache Single-Bit Data Error Interrupt enable 0: disable 1: enable					R/W	b0
0	L1TAG_ERR_INT_EN		L1 Cache Single-Bit Tag Error Interrupt enable 0: disable 1: enable					R/W	b0

8.10 System Credits and Counter Registers

The System Bridge Controller, through the I/O Bridge of [Table 8-2](#), can maintain a maximum of 32 transactions with the individual I/O devices on the I/O Distributed Interconnect Ring. Allocation of these transactions is made by assigning credits to I/O devices, which are divided into three groups, PCI-e, GIO, and all other I/O devices.

This section describes the registers used for assigning credits and monitoring the number of I/O transactions carried out through the System Bridge.

8.10.1 SYS2IO_CREDITS

Register ID: 0x035

Address Offset: 0x0D4

System Bridge I/O Transaction Credits register

This register specifies the number of I/O transaction credits assigned to the corresponding I/O device group. This number represents the maximum number of transactions pending execution that the IO devices may have through the System Bridge at any given time. A transaction refers to a memory transaction from the Memory Distributed Interconnect ring to the I/O Distributed Interconnect ring.

The sum of credits assigned to each I/O group must not exceed 32, which is the maximum number of outstanding transactions that the System Bridge can maintain.

31:29	28:24	23:21	20:16	15:13	12:8	7:5	4:0
Reserved	OTHER_CREDITS	Reserved	GIO_CREDITS	Reserved	Reserved	Reserved	PCIE_CREDITS

Bits	Field Name	Field Description	R/W	Reset
31:29	Reserved	Reserved	RO	b000
28:24	OTHER_CREDITS	I/O group credits for the SGMII/RGMII (optional XAUI), SAE, and DMA Engine	R/W	b0 1000
23:21	Reserved	Reserved	RO	b000
20:16	GIO_CREDITS	GPIO group credits and memory (Flash, PCMCIA)	R/W	b0 0100
15:13	Reserved	Reserved	RO	b000
12:8	Reserved	Reserved	RO	b000
7:5	Reserved	Reserved	RO	b000
4:0	PCIE_CREDITS	PCIe group credits	R/W	b1 0100

8.10.2 EVENT_CNT_CNTRL1

Register ID: 0x036
Address Offset: 0x0D8

System Bridge Event Count Control 1 register

This register is used to activate/deactivate the counting of the specified events through the System Bridge. One or multiple source of events may be specified to be included in the count. When activated, the Event Counter 1 keeps count of the occurrence of all events specified. That is, [EVENT_CNTRL1](#) register will increment each any of the events specified occurs. A hit refers to a transaction completed successfully. A retry refers to a transaction repeated because it failed previously. The reset value is 0x0000 0000. Set a bit to '1' to enable the function.

31:26	27	26	25	24:23	22	20	19	18	17	16	
RSVD	OTHER_RTY_EN	PCIE_RTY_EN	GIO_RTY_EN	RSVD	LCL_RTY_EN	BRDG_RTY_EN	DRAMD_RTY_EN	DRAMC_RTY_EN	DRAMB_RTY_EN	DRAMA_RTY_EN	
15:12	11	10	9	8:7	6	5	4	3	2	1	
RSVD	OTHER_HIT_EN	PCIE_HIT_EN	GIO_HIT_EN	RSVD	LCL_HIT_EN	BRDG_HIT_EN	RSVD	DRAMD_HIT_EN	DRAMC_HIT_EN	DRAMB_HIT_EN	
Bits	Field Name			Field Description						R/W	Reset
31:28	<i>Reserved</i>			<i>Reserved</i>						RO	b0000 00
27	OTHER_RETRY_EN			Other I/O group transaction retry enable (any SGMAC, SAE, DMA, C/DE, USB) 0: disable (default) 1: enable						R/W	b0
26	PCIE_RETRY_EN			PCIe transaction retry enable 0: disable (default) 1: enable						R/W	b0
25	GIO_RTY_EN			GPIO group transaction retry enable 0: disable (default) 1: enable						R/W	b0
24:23	<i>Reserved</i>			<i>Reserved</i>						RO	b00
22	LCL_RTY_EN			Local register transaction retry enable 0: disable (default) 1: enable						R/W	b0
21	BRDG_RTY_EN			Bridge transaction retry enable 0: disable (default) 1: enable						R/W	b0
20	<i>Reserved</i>			<i>Reserved</i>						RO	b0
19	DRAMD_RTY_EN			DRAM Channel D transaction retry enable 0: disable (default) 1: enable						R/W	b0
18	DRAMC_RTY_EN			DRAM Channel C transaction retry enable 0: disable (default) 1: enable						R/W	b0

Bits	Field Name	Field Description	R/W	Reset
17	DRAMB_RTY_EN	DRAM Channel B transaction retry enable 0: disable (default) 1: enable	R/W	b0
16	DRAMA_RTY_EN	DRAM Channel A transaction retry enable 0: disable (default) 1: enable	R/W	b0
15:12	Reserved	Reserved	RO	b0000
11	OTHER_HIT_EN	Other I/O group transaction hit enable (any SGMAC, SAE, DMA, C/DE, USB) 0: disable (default) 1: enable	R/W	b0
10	PCIE_HIT_EN	PCIe transaction hit enable 0: disable (default) 1: enable	R/W	b0
9	GIO_HIT_EN	GPIO transaction hit enable 0: disable (default) 1: enable	R/W	b0
8:7	Reserved	Reserved	RO	b00
6	LCL_HIT_EN	Local register transaction hit enable 0: disable (default) 1: enable	R/W	b0
5	BRDG_HIT_EN	Bridge Register transaction hit enable 0: disable (default) 1: enable	R/W	b0
4	Reserved	Reserved	RO	b0
3	DRAMD_HIT_EN	DRAM Channel D transaction hit enable 0: disable (default) 1: enable	R/W	b0
2	DRAMC_HIT_EN	DRAM Channel C transaction hit enable 0: disable (default) 1: enable	R/W	b0
1	DRAMB_HIT_EN	DRAM Channel B transaction hit enable 0: disable (default) 1: enable	R/W	b0
0	DRAMA_HIT_EN	DRAM Channel A transaction hit enable 0: disable (default) 1: enable	R/W	b0

8.10.3 EVENT_CNTR1

Register ID: 0x037
 Address Offset: 0x0DC

System Bridge Event Counter 1 register

This Read Only register keeps count of the occurrence of events enabled in the System Bridge Event Counter Control 1 ([EVENT_CNT_CNTRL1](#)) register. When activated, this register increments each time any of the specified events occurs. The reset value is 0x0000 0000.

		31:0		
EVENT_COUNT				
Bits	Field Name	Field Description	R/W	Reset
31:0	EVENT_COUNT	32-bit Event Counter 1 – Count occurrence of all events specified in EVENT_CNT_CNTRL1	RO	0x0000 0000

8.10.4 EVENT_CNT_CNTRL2

Register ID: 0x038
 Address Offset: 0x0E0

System Bridge Event Counter Control 2 register

This Read/Write register is used to activate/deactivate the counting of the specified events through the System Bridge. One or multiple source of events may be specified to be included in the count. When activated, the Event Counter 2 ([EVENT_CNTR2](#)) keeps count of the occurrence of all events specified. That is, EVENT_CNTR2 register increments when any of the events specified occurs.

A hit refers to a transaction completed successfully. A retry refers to a transaction that is repeated because it failed previously. The reset value is 0x0000 0000.

31:26	27	26	25	24:23	22	20	19	18	17	16
RSVD	OTHER_RTY_EN	PCIE_RTY_EN	GIO_RTY_EN	RSVD	LCL_RTY_EN	BRDG_RTY_EN	DRAMD_RTY_EN	DRAMC_RTY_EN	DRAMB_RTY_EN	DRAMA_RTY_EN
15:12	11	10	9	8:7	6	5	4	3	2	1
RSVD	OTHER_HIT_EN	PCIE_HIT_EN	GIO_HIT_EN	RSVD	LCL_HIT_EN	BRDG_HIT_EN	RSVD	DRAMD_HIT_EN	DRAMC_HIT_EN	DRAMB_HIT_EN
										0

EVENT_CNT_CNTRL2 Register

Bits	Field Name	Field Description	R/W	Reset
31:28	<i>Reserved</i>	<i>Reserved</i>	RO	b0000 00
27	OTHER_RETRY_EN	OTHER I/O transaction retry enable (any SGMAC, SAE, DMA, C/DE, USB) 0: disable (default) 1: enable	R/W	b0
26	PCIE_RETRY_EN	PCIe transaction retry enable 0: disable (default) 1: enable	R/W	b0
25	GIO_RTY_EN	GPIO transaction retry enable 0: disable (default) 1: enable	R/W	b0
24:23	<i>Reserved</i>	<i>Reserved</i>	RO	b0
22	LCL_RTY_EN	Local register transaction retry enable 0: disable (default) 1: enable	R/W	b0
21	BRDG_RTY_EN	Bridge transaction retry enable 0: disable (default) 1: enable	R/W	b0
20	<i>Reserved</i>	<i>Reserved</i>	RO	b0
19	DRAMD_RTY_EN	DRAM Channel D transaction retry enable 0: disable (default) 1: enable	R/W	b0
18	DRAMC_RTY_EN	DRAM Channel C transaction retry enable 0: disable (default) 1: enable	R/W	b0
17	DRAMB_RTY_EN	DRAM Channel B transaction retry enable 0: disable (default) 1: enable	R/W	b0
16	DRAMA_RTY_EN	DRAM Channel A transaction retry enable 0: disable (default) 1: enable	R/W	b0
15:12	<i>Reserved</i>	<i>Reserved</i>	RO	b0000
11	OTHER_HIT_EN	OTHER I/O transaction hit enable (any SGMAC, SAE, DMA, C/DE, USB) 0: disable (default) 1: enable	R/W	b0
10	PCIE_HIT_EN	PCIe transaction hit enable 0: disable (default) 1: enable	R/W	b0
9	GIO_HIT_EN	GPIO transaction hit enable 0: disable (default) 1: enable	R/W	b0
8:7	<i>Reserved</i>	<i>Reserved</i>	RO	b0
6	LCL_HIT_EN	Local register hit enable	R/W	b0
5	BRDG_HIT_EN	Bridge Register hit enable	R/W	b0
4	<i>Reserved</i>	<i>Reserved</i>	RO	b0

EVENT_CNT_CNTRL2 Register(*continued*)

Bits	Field Name	Field Description	R/W	Reset
3	DRAMD_HIT_EN	DRAM Channel D transaction hit enable 0: disable (default) 1: enable	R/W	b0
2	DRAMC_HIT_EN	DRAM Channel C transaction hit enable 0: disable (default) 1: enable	R/W	b0
1	DRAMB_HIT_EN	DRAM Channel B transaction hit enable 0: disable (default) 1: enable	R/W	b0
0	DRAMA_HIT_EN	DRAM Channel A transaction hit enable 0: disable (default) 1: enable	R/W	b0

8.10.5 EVENT_CTR2

Register ID: 0x039

Address Offset: 0x0E4

System Bridge Event Counter 2 register

This Read-Only register keeps count of the occurrence of events enabled in the System Bridge Event Counter Control 2 register ([EVENT_CNT_CNTRL2](#)). When enabled, this register is incremented each time any of the specified events occurs. The reset value is 0x0000 0000.

31:0	EVENT_COUNT

Bits	Field Name	Field Description	R/W	Reset
31:0	EVENT_COUNT	32-bit Event Counter 2 – Counts occurrence of all events specified in System Bridge EVENT_CNT_CNTRL2	RO	0x0000 0000

8.10.6 MISC FUNCTION CTL1

Register ID: 0x03A
 Address Offset: 0x0EC

Miscellaneous Function Control register

This register is used to activate/deactivate special functions including control of access to L2 Cache via JTAG, ordering modes for transaction with certain devices, and System Bridge Controller handling of PCIe addressing.

31:5	4	3	2	1	0
RSVD	DMA_ORDERING_MODE	USB_ORDERING_MODE	PCIE_ORDERING_MODE	PCIE_ADDRESSING_PASSTHRU	CACHEABLE_DISABLE

Bits	Field Name	Field Description	R/W	Reset
31:6	Reserved	Reserved. Returns 0 on read	R	0
5	SRIO_ORDERING_MODE	SRIO Transaction Ordering Mode 0: The System Bridge Controller enforces write-ordering-only on all memory transactions from the SRIO interface. This means that a SRIO write transaction blocks all subsequent SRIO memory transactions until the write transaction completes. 1: The System Bridge Controller enforces strong-ordering on all memory transactions from the SRIO interface. This means that a SRIO memory transaction blocks all subsequent SRIO memory transactions until the first transaction is completed.	R/W	0
4	DMA_ORDERING_MODE	DMA Engine Transaction Ordering Mode 0: The System Bridge Controller enforces write-ordering-only on all non-coherent memory transactions from the DMA Engine. This means that a non-coherent write transaction blocks all subsequent DMA Engine memory transactions until the write transaction completes 1: The System Bridge Controller enforces strong-ordering on all non-coherent memory transactions from the DMA Engine. This means that a non-coherent DMA memory transaction blocks all subsequent DMA memory transactions until the first transaction is completed.	R/W	0
3	USB_ORDERING_MODE	USB Transaction Ordering Mode 0: The System Bridge Controller enforces write-ordering-only on all memory transactions from the USB interface. This means that a write transaction blocks all subsequent USB memory transactions until the write transaction completes. 1: The bridge enforces strong-ordering on all memory transactions from the USB. This means that a USB memory transaction blocks all subsequent USB memory transactions until the first transaction is completed.	R/W	0

Bits	Field Name	Field Description	R/W	Reset
2	PCIE_ORDERING_MODE	<p>PCIe Transaction Ordering Mode</p> <p>0: The System Bridge Controller enforces write-ordering-only on all memory transactions from the PCIe interface. This means that a PCIe write transaction blocks all subsequent PCIe memory transactions until the write transaction completes.</p> <p>1: The System Bridge Controller enforces strong-ordering on all memory transactions from the PCIe interface. This means that a PCIe memory transaction blocks all subsequent PCIe memory transactions until the first transaction is completed.</p>	R/W	0
1	PCIE_ADDRESSING_MODE	<p>PCI Addressing Mode</p> <p>0: The System Bridge Controller passes addresses for PCIE MEM and PCIe I/O transactions without modification.</p> <p>1: The System Bridge Controller modifies PCIe memory addresses and PCIe I/O transactions by zeroing out the portions of the base address that match the base address specified in the corresponding BAR.</p>	R/W	0
0	CACHEABLE_DISABLE	<p>L2 Cache access through JTAG port</p> <p>0: A request is un-cacheable if its cacheable bit is cleared which causes all caches to ignore the request.</p> <p>1: All requests from I/O devices to the MDI and I/O DI rings are un-cacheable unless the I/O device sets the coherent bit. If the cacheable bit is set, then the caches snoop the request. By default, the cacheable attribute bit is always set and the L2-allocate bit is used to control whether or not the data is pulled into the L2 Cache.</p> <p>NOTE: This bit should be set only when L2 access through JTAG is required, that is, in debug situations and not normal operation. Enabling this option forces all MDI transactions from the I/O DI to be un-cacheable if the coherent attribute is disabled. L2 configuration registers may only be accessed by un-cacheable transactions. All I/O DI to MDI transactions are affected. Transactions from other I/O DI agents (DMA, PCIe, etc.) are made un-cacheable, which is why this bit should only be set for debugging purposes.</p> <p>See also Chapter 25, "XLS Debugging Features"</p>	R/W	0

8.11 Scratch Registers

The System Bridge Controller provides four registers that may be used for any purpose as shown in Figure 8-8 below.

8.11.1 SCRATCH[0-3]

Register ID: 0x03C – 0x03F

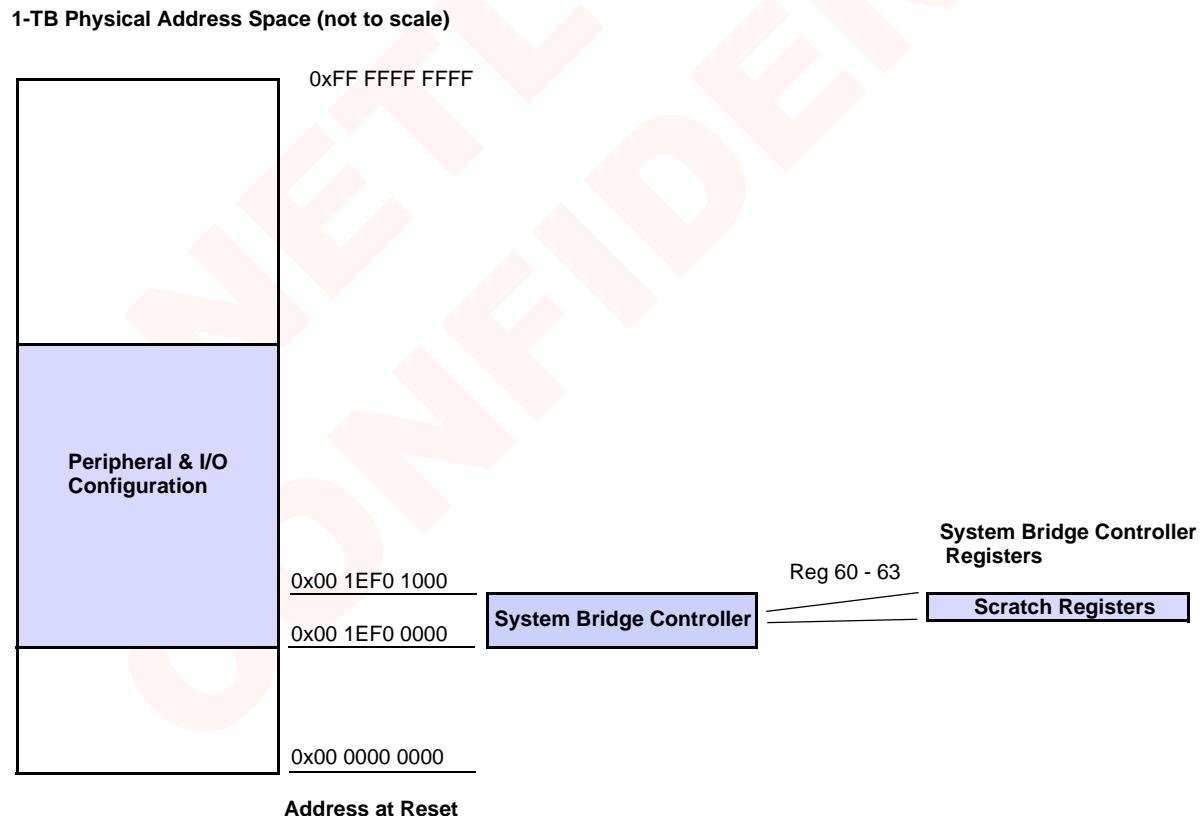
Address Offset: 0x0F0 – 0x0FC

System Bridge Scratch registers 0 - 3

These four 32-bit Read/Write registers are general-purpose (scratch) registers. Reset values are 0x0000 0000.

		31:0			
		SCRATCH			
Bits	Field Name	Field Description	R/W	Reset	
31:0	SCRATCH	General-purpose Scratch registers	R/W	0x0000 0000	

Figure 8-8. General-Purpose Scratch Registers on the MDI



8.12 PCIe Base Address Registers

8.12.1 PCIE_CFG_BAR

Register ID: 0x040

Address Offset: 0x100

PCIe Configuration Base-Address Register

This register specifies the starting physical address of the PCIe configuration region within the Memory Mapped Peripheral & I/O space. The reset value is 0x0000 0000.

31:17	PCIE_CFG_BAR	16:1	0
		Reserved	PCIE_CFG_BAR_EN

Bits	Field Name	Field Description	R/W	Reset
31:17	PCIE_CFG_BAR	PCIe starting physical address bits 39:25 Reset value = b0000 0000 0000 0000	R/W	See Field Desc
16:1	Reserved	Reserved Reset value = b0 0000 0000 0000 000	RO	See Field Desc
0	PCIE_CFG_BAR_EN	PCIe Configuration Base-Address register enable 0: Disable 1: Enable	R/W	b0

8.12.2 PCIE_ECFG_BAR

Register ID: 0x041

Address Offset: 0x104

PCIe Extended-Configuration Base-Address register

This register specifies the starting physical address of the PCIe Extended-Configuration region within the Memory Mapped Peripheral & I/O space. The reset value is 0x0000 0000.

31:21	PCIE_ECFG_BAR	20:1	0
		Reserved	PCIE_ECFG_BAR_EN

Bits	Field Name	Field Description	R/W	Reset
31:21	PCIE_ECFG_BAR	PCIe Extended-Configuration Base Physical Address bits 39:29	R/W	0
20:1	Reserved	Reserved	RO	0
0	PCIE_ECFG_BAR_EN	PCIe Extended-Configuration Base-Address register enable 0: Disable 1: Enable	R/W	0

8.12.3 PCIE_MEM_BAR

Register ID: 0x042
 Address Offset: 0x108

PCIe Memory Base-Address register (when SRIO is not used, see 8.13.1)

This register specifies the starting address of the PCIe Memory region within the Memory Mapped Peripheral & I/O space. The reset value is 0x0000 0000.

31:16	15:1	0		
PCIE_MEM_BASE_ADDR	ADDR_MASK	PCIE_MEM_BAR_EN		
Bits	Field Name	Field Description	R/W	Reset
31:16	PCIE_MEM_BASE_ADDR	PCIe Memory Physical Base-Address bits 39:24	R/W	0
15:1	ADDR_MASK	PCIe Memory Physical Base Address Mask bits 38:24 Reset value = b0000 0000 0000 0000 000	R/W	
0	PCIE_MEM_BAR_EN	PCIe Memory Base-Address register enable 0: Disable 1: Enable	R/W	0

8.12.4 PCIE_IO_BAR

Register ID: 0x043
 Address Offset: 0x10C

PCIe I/O Base-Address register (when SRIO is not used, see 8.13.2)

This register specifies the starting address of the PCIe I/O region within the Memory Mapped Peripheral & I/O space. The reset value is 0x0000 0000.

31:18	17:7	6:1	0	
PCIE_IO_BASE_ADDR	Reserved	ADDR_MASK	PCIE_IO_BAR_EN	
Bits	Field Name	Field Description	R/W	Reset
31:18	PCIE_IO_BASE_ADDR	PCIe I/O Physical Base Address bits 39:26	R/W	0
17:7	Reserved	Reserved	RO	0
6:1	ADDR_MASK	PCIe I/O Physical Address mask bits 31:26	R/W	0
0	PCIE_IO_BAR_EN	PCIe I/O Base-Address register enable 0: Disable 1: Enable	R/W	0

8.13 SRI0 Base Address Registers

8.13.1 SRI0_MEM_BAR

Register ID: 0x042

Address Offset: 0x108

SRI0 Memory Base-Address register (when PCIe is not used, see 8.12.3)

This register specifies the starting address of the SRI0 Memory region within the Memory Mapped Peripheral & I/O space. The reset value is 0x0000 0000.

31:16	15:1	0
SRI0_MEM_BASE_ADDR	ADDR_MASK	SRI0_MEM_BAR_EN

Bits	Field Name	Field Description	R/W	Reset
31:16	SRI0_MEM_BASE_ADDR	SRI0 Memory Physical Base-Address bits 39:24	R/W	0
15:1	ADDR_MASK	SRI0 Memory Physical Base Address Mask bits 38:24 Reset value = b0000 0000 0000 000	R/W	0
0	SRI0_MEM_BAR_EN	SRI0 Memory Base-Address register enable 0: Disable 1: Enable	R/W	0

8.13.2 SRI0_IO_BAR

Register ID: 0x043

Address Offset: 0x10C

SRI0 I/O Base-Address register (when PCIe is not used, see 8.12.4)

This register specifies the starting address of the SRI0 I/O region within the Memory Mapped Peripheral & I/O space. The reset value is 0x0000 0000.

31:18	17:7	6:1	0
SRI0_IO_BASE_ADDR	Reserved	ADDR_MASK	SRI0_IO_BAR_EN

Bits	Field Name	Field Description	R/W	Reset
31:18	SRI0_IO_BASE_ADDR	SRI0 I/O Physical Base Address bits 39:26	R/W	0
17:7	Reserved	Reserved	RO	0
6:1	ADDR_MASK	SRI0 I/O Physical Address mask bits 31:26	R/W	0
0	SRI0_IO_BAR_EN	SRI0 I/O Base-Address register enable 0: Disable 1: Enable	R/W	0

8.14 Additional Device Mask Register

8.14.1 DEVICE_MASK2

Register ID: 0x044

Address Offset: 0x110

Device Mask 2 register

This register represents active devices in addition to the devices specified by the Active Device register (See “[DEVICE_MASK](#)” on page 245.) It is a mask representing in real-time additional active devices in an XLS Processor Family member that are not covered by DEVICE_MASK. Some bits such as the PIC field are determined by hardwired connections while others are maintained by software.

Reset values depend upon the specific XLS Family Member.

31:3		2	1	0		
<i>Reserved</i>		USB	PCIE	CDE		
Bits	Field Name	Field Description			R/W	Reset
31:3	<i>Reserved</i>	<i>Reserved</i>			R/W	0
2	USB	USB device access enable/status 0: inactive 1: active			R/W	-
1	PCIE	PCIe device access enable/status 0: inactive 1: active			R/W	-
0	CDE	Compression/Decompression Engine device access enable/status 0: inactive 1: active <i>*Note:</i> This bit is Reserved and Read-Only in the XLS2xx and XLS1xx devices, with a value of b0.			R/W RO*	-



Chapter 9 Direct Memory Access (DMA) Engine

9.1 Introduction

This chapter provides the following information:

- Overview of DMA engine capabilities
- Architecture and theory of operation
 - module internal architecture
 - operational description
 - operating modes/states/access model
- Programming model
 - startup/initialization
- Message types for DMA Engine
- Registers
 - master list of registers for this interface
 - register descriptions

Recommended additional reading: [Section 13.10.4 DMA Credit Registers](#) in Chapter 13, “Networking Accelerators”.

9.1.1 Functionality

XLS Processors contain an independent 4-channel DMA Engine that handles data transfer to and from any internal memory-mapped device; device regions are identified in Chapter 8. The position of the Engine within the XLS architecture is shown in the system block diagram in [Figure 1-1](#), and in [Figure 9-1](#). The Engine provides DMA functionality separate from the DMA functionality which is embedded in each Network Accelerator, Compression Engine, PCIe, USB, and Security Accelerator modules.

The Engine supports these key features:

- Four DMA channels that allow threads to run four transfer operations in parallel
- Checksum calculation
- Calculating CRC with 32-bit programmable polynomial
- CRC register may be initialized with arbitrary polynomial to allow computing CRC over multiple chunks of data
- Engine provides positive acknowledgement via a FMN reply message to the initiating thread when transfer is complete
- Streaming mode support available on two channels

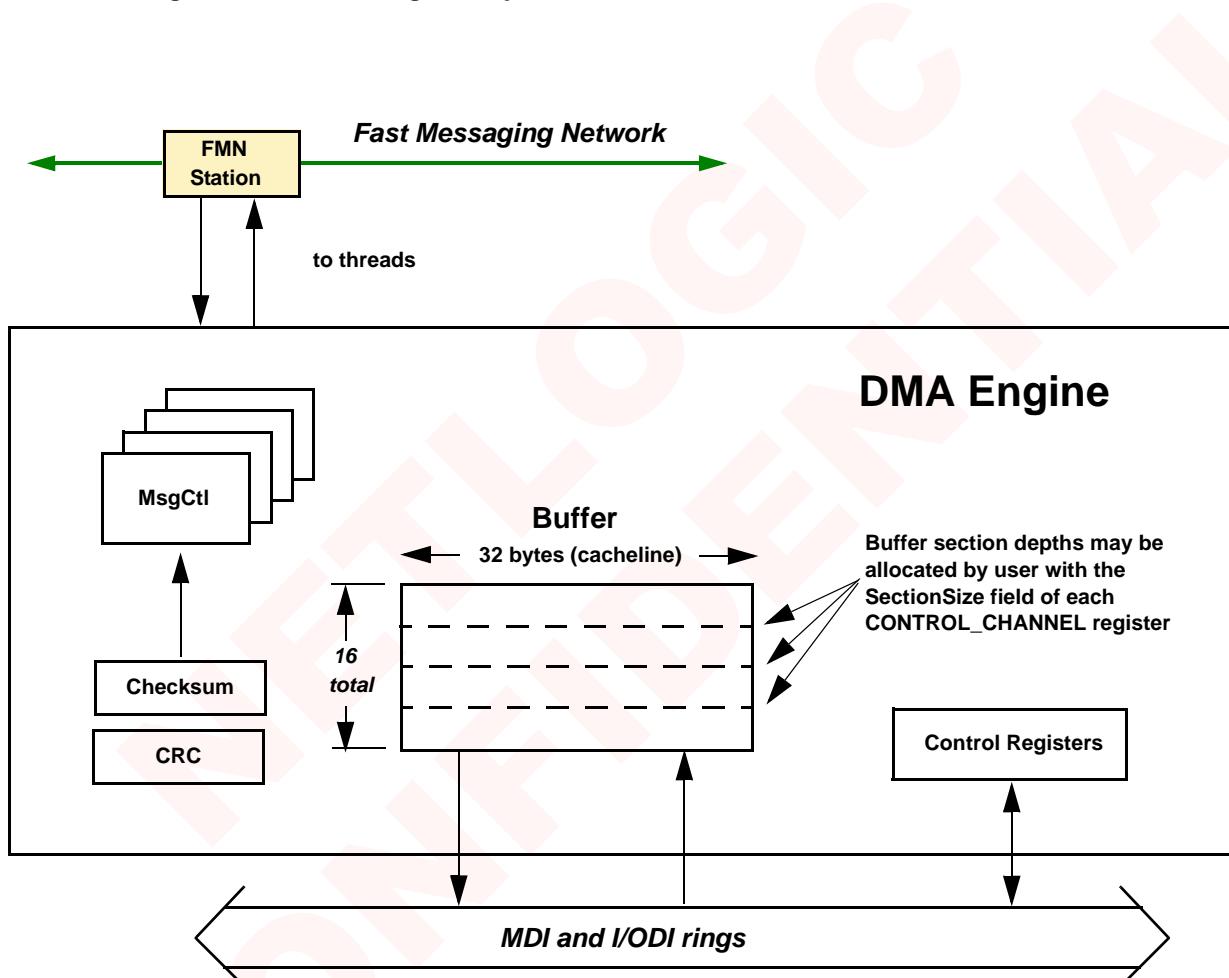
Maximum transfer length is 1,048,575 bytes (1 MB – 1 byte) per descriptor. Larger transfers can be achieved using multiple descriptors.

9.1.2 Architecture and Theory of Operation

The DMA Engine internal high-level architecture is shown in [Figure 9-1](#). The Engine is controlled by messages sent to it over the FMN; message types are described later in this chapter. The Engine then manages the transfer performed over the MDI and I/O DI rings with memory or I/O device sources and targets. When the transfer finishes or in the event of error, the Engine sends a message back to the originator.

The DMA Engine's buffer has a width of 32 bytes and a fixed total depth of 16. The buffer may be partitioned among channels up to the maximum depth of 16.

Figure 9-1. DMA Engine Key Elements



9.2 Programming Model

9.2.1 Background Information for Initialization

The DMA Engine must first be initialized before it can be used. This involves setting up the CHANNEL_CONTROL and CRC_POLY registers for all four channels.

9.2.1.1 Buffer Allocation

The DMA Engine uses an internal 16-deep holding buffer that has to be partitioned among the 4 channels.

The SectionsSize field in the Control Register is used to define the amount of buffer space that each channel gets. Size value for any channel can be from 0 to 16; the total buffer space that can be allocated among the four channels is 16.

The RMaxCr defines the number of outstanding read requests (ring credits) that can be on the MDI and I/ODI. This value should not be greater than the SectionSize. The Max value is 4. Recommended value for this register is 4.

The WMaxCr defines the number of write requests that can be queued to the MDOI and I/ODI. The Max value is 4. Recommended value for this register is 4.

9.2.1.2 Message Types and Related Setup

Once the DMA Engine is initialized, it is commanded by sending messages to it. The Engine handles four basic types of messages.

1. SIMPLE_TRANSFER.

Simple transfer is used for copying a block of data from one location in memory to another (including any memory mapped devices).

2. COMPUTE_CRC

DMA Engine is given the start and number of bytes and CRC is computed over it.

3. COMPUTE_CHECKSUM

DMA Engine is given the start and number of bytes and CHECKSUM is computed over it.

4. COMPUTE_CRC_AND_CHECKSUM

DMA Engine computes both CRC and CHECKSUM over the data.

These types are described in detail in [Section 9.3, “DMA Engine Message Formats”](#).

Some of the message types require prior setup, as described in the following section.

9.2.1.3 CRCs and Checksums

The DMA Engine can compute the CRC and Checksum over a given number of bytes in memory. For this the CRC_POLYNOMIAL field must be set in the CRC_POLY register during initialization. The InformSource bit in the message must be set for the CRC / Checksum to be returned to the nCPU. The InitCRC or the InitCSUM bit is also important to set based on the protocol. For example, if you want to compute Ethernet CRC, the InitCRC in the message must be set to all ones and the CrcBitOrd in the CONTROL register must be set. If CRC or Checksum has to be computed over data that is scattered in memory, then multiple messages can be sent to the DMA Engine.

- In the case where these messages are sent to the DMA Engine without any other message getting interleaved, then UseOldCRC bit in the message should be set in all messages excluding the first one.

- If it is not possible to guarantee non-interleaving messages, then the nCPU must wait for the partial CRC to return from the DMA engine and put this into the initCRC field of the next message that it sends to the DMA Engine.

9.2.1.4

Message Error Handling

In cases where the DMA Engine receives a message with bad format or zero length, then the return message has the MsgFormatError bit set. A length zero message error is sent back to the bucket programmed in the MsgSource field of the Incoming Message. A message format error is sent to the bucket programmed in the ErrMsgDstID field on the CONTROL register. The DEBUG_STATE register latches this for the current transfer.

Caution: Software should not send zero-sized checksum/CRC requests to the DMA engine as part of a linked list; the incremental checksum/CRC returned in this case will be inconsistent.

9.2.1.5

Address/Data Error Handling

During data transfer, if the DMA Engine receives a bus error from the System Bridge Controller, then an Error Message is sent to the bucket programmed in the MsgSource field of the Incoming Message. A bus error from the SBC can be caused by a bad address (Address for which the BAR was not set up in the bridge), or if data had an ECC error. The DEBUG_STATE register latches this for the current transfer.

9.2.2

Reset Sequence

This applies for any single channel or all channels

Perform the following.

1. Set AbortXfer and PopAllMessages fields in the CONTROL_CHANNEL register.
2. Pop all messages that may have been returned from the DMA.
3. Wait for 100000 core-clock cycles, and make sure no new messages have been returned by the DMA.
4. If no new messages are returned then set ClearTags.
5. Clear all these bits.

9.3 DMA Engine Message Formats

The DMA Engine responds to four types of messages sent to it by an originator via the FMN. [Table 9-1](#) identifies these types. Their formats are described in the following sections.

The DMA Engine performs memory block transfers in response to the Simple Transfer message type. A return message is sent back to the requestor via the FMN.

The DMA can also perform CRC and CSUM calculations without data transfer in response to the CRC and CSUM message types, respectively. A field of the return message sent back to the requestor via the FMN contains the computed CRC or CSUM.

CRC and CSUM can be calculated concurrently in response to the CRC+CSUM message type. Additionally, the SimpleXferAlso field of this message can invoke data transfer as well.

Table 9-1. Message Formats

Messages Supported	Number of Descriptor Entries	Message Type
Message Type — Simple Transfer	2 Entries	0
Message Type — CRC	2 Entries	1
Message Type — CSUM	2 Entries	2
Message Type — CRC + CSUM	3 Entries	3

9.3.1 Descriptor Formats

9.3.1.1 Message Type — Simple Transfer

Note: Do not use Simple Transfer messages with other Message Types; instead use the CSUM Message type with CRC or CRC + CSUM messages.

First Word							
63	62:61	60	59:40	39:0			
FWF	MessageType	Reserved	Length	SourceAddress			
Bit Name Description							
63	FWF	First Word Flag. Must be 1 for first word in message					
62:61	MessageType	Message type = 0 for simple transfer. See Table 9-1 .					
60	Reserved	Reserved Must be = 0					
59:40	Length	Transfer length in bytes. Maximum transfer length is 1,048,575 bytes					
39:0	SourceAddress	Source byte address					

Second Word										
63	62	61	60	59	58	57:48	47	46	45:40	39:0
FWF	Wr Coherent	Rd Coherent	WrL2 Allocate	RSVD	Inform Source	Transaction ID	RdL2 Allocate	Rd Exclusive	Msg Source	Destination Address
63	FWF	First Word Flag. Must be 0 if this is not first word in message.								
62	WrStrongOrdering	Write Strong Ordering Control. 0: Do not strongly order writes 1: When set, writes are strongly ordered, and the bridge does a read-modify-write for unaligned addresses.								
61	RdStrongOrdering	Read Strong Ordering Control 0: Do not strongly order reads 1: When set, reads are strongly ordered								
60	WrL2Allocate	L2 Cache Write Control 0: For writes not going to L2 cache 1: Set to cause writes to go to L2 cache								
59	Reserved	Reserved Must be = 0								
58	InformSource	Message Response Control 0: Do not send message back 1: Send message back to MsgSource when transaction completes								
57:48	TransactionID	Software programmable field that is copied into return message if InformSource is set								
47	RdL2Allocate	L2 Cache Read Control 0: Do not store reads in cache 1: When set, if the data is not present in the L2 Cache, then data is stored in L2 Cache during read.								
46	RdPurge	L2 Cache Read and Purge 0: Read operation will not affect the cached data (the data will not be purged from the cache after the read operation). 1: When set, if the data is present in the cache, then after the read, the data is discarded from the cache without being written back to external memory. Warning: Use extreme caution when setting this bit. This bit does not need to be set for normal operation. Only set it when it is certain that this will be the last operation to be performed by the DTE on the associated data. If an nCPU attempts to read the data after the DTE read, the data it fetches may be stale.								
45:40	MsgSource	Bucket ID to receive return message if enabled by InformSource. See Fast Messaging network chapter discussion of Bucket ID.								
39: 0	DestinationAddress	Destination byte address								

9.3.1.2 Message Type — CRC

First Word														
63	62:61	60	59:40						39:0					
FWF	Message Type	Reserved	Length	SourceAddress										
Bit	Name		Description											
63	FWF		First Word Flag. Must be 1 for first word in message											
62:61	MessageType		Message type. = 1 for CRC. See Table 9-1 .											
60	Reserved		Reserved Must be = 0											
59:40	Length		CRC calculation length in bytes											
39:0	SourceAddress		Source byte address											
Second Word														
63	62	61	60	59	58	57:48	47	46	45:40	39:32	31:0			
FWF	Wr Strong Ordering	Rd Strong Ordering	L2 Allocate	Use Old CRC	Inform Source	Transaction ID	RdL2 Allocate	Rd Purge	Msg Source	RSVD	Init CRC			
Bit	Name		Description											
63	FWF		First Word Flag. Must be 0 if this is not first word in message.											
62	WrStrongOrdering		Write Strong Ordering Control. 0: Do not strongly order writes 1: When set, writes are strongly ordered, and the bridge does a read-modify-write for unaligned addresses.											
61	RdStrongOrdering		Read Strong Ordering Control 0: Do not strongly order reads 1: When set, reads are strongly ordered											
60	WrL2Allocate		L2 Cache Write Control 0: Use this setting for writes not going to L2 cache 1: Set to cause writes to go to L2 cache											
59	UseOldCRC		CRC Control 0: use InitCRC field value to initialize CRC. 1: For multiple segments, set this bit on all but the first calculation											
58	InformSource		Message Response Control 0: Do not send message back 1: Send message back to MsgSource when transaction completes											
57:48	TransactionID		Software programmable field that is copied into return message if InformSource is set											
47	RdL2Allocate		L2 Cache Read Control 0: Do not store reads in cache 1: When set, if the data is not present in the L2 Cache, then data is stored in L2 Cache during read.											

Bit	Name	Description
46	RdPurge	L2 Cache Read and Purge 0: Read operation will not affect the cached data (the data will not be purged from the cache after the read operation). 1: When set, if the data is present in the cache, then after the read, the data is discarded from the cache without being written back to external memory. Warning: Use extreme caution when setting this bit. This bit does not need to be set for normal operation. Only set it when it is certain that this will be the last operation to be performed by the DTE on the associated data. If an nCPU attempts to read the data after the DTE read, the data it fetches may be stale.
45:40	MsgSource	Bucket ID to receive return message if enabled by InformSource.
39:32	Reserved	<i>Reserved</i> Must be = 0
31: 0	InitCRC	Initial CRC value

9.3.1.3 Message Type — CSUM

First Word											
Bit		Name		Description							
63	FWF	Message Type		First Word Flag. Must be 1 for first word in message							
62:61	MessageType		Message type. = 2 for CSUM. See Table 9-1 .								
60	Reserved		<i>Reserved</i> Must be = 0								
59:40	Length		CSUM calculation length in bytes								
39:0	SourceAddress		Source byte address								
Second Word											
63	62	61	60	59	58	57:48	47	46	45:40	39:16	15:0
FWF	Wr Strong Ordering	Rd Strong Ordering	L2 Allocate	Use Old CRC	Inform Source	Transaction ID	RdL2 Allocate	Rd Purge	Msg Source	RSVD	Init CSUM
Bit											
63	FWF		First Word Flag. Must be 0 if this is not first word in message.								
62	WrStrongOrdering		Write Strong Ordering Control. 0: do not strongly order writes 1: When set, writes are strongly ordered, and the bridge does a read-modify-write for unaligned addresses.								
61	RdStrongOrdering		Read Strong Ordering Control 0: do not strongly order reads 1: When set, reads are strongly ordered								
60	WrL2Allocate		L2 Cache Write Control 0: Use this setting for writes not going to L2 cache 1: Set to cause writes to go to L2 cache								
59	Reserved		<i>Reserved</i> Must be = 0								
58	InformSource		Message Response Control 0: Do not send message back 1: Send message back to MsgSource when transaction completes								
57:48	TransactionID		Software programmable field that is copied into return message if InformSource is set								
47	RdL2Allocate		L2 Cache Read Control 0: Do not store reads in cache 1: When set, if the data is not present in the L2 Cache, then data is stored in L2 Cache during read.								

Bit	Name	Description
46	RdPurge	L2 Cache Read and Purge 0: Read operation will not affect the cached data (the data will not be purged from the cache after the read operation). 1: When set, if the data is present in the cache, then after the read, the data is discarded from the cache without being written back to external memory. Warning: Use extreme caution when setting this bit. This bit does not need to be set for normal operation. Only set it when it is certain that this will be the last operation to be performed by the DTE on the associated data. If an nCPU attempts to read the data after the DTE read, the data it fetches may be stale.
45:40	MsgSource	Bucket ID to receive return message if enabled by InformSource.
39:16	Reserved	Reserved Must be = 0
15:0	InitCSUM	Initial CSUM value

9.3.1.4 Message Type — CRC + CSUM

First Word				
	63	62:61	60	59:40
FWF	Message Type (3)	SimpleXferAlso	Length	Source Address

Bit	Name	Description
63	FWF	First Word Flag. Must be 1 for first word in message
62:61	MessageType	Message type. = 3 for CRC+CSUM. See Table 9-1 .
60	SimpleXferAlso	Also perform DMA transfer 0: Perform only CRC and CSUM calculation 1: Also perform DMA operation
59:40	Length	Transfer length in bytes
39:0	SourceAddress	Source byte address

Second Word											
	63	62	61	60	59	58	57:48	47	46	45:40	39:0
FWF	Wr Strong Ordering	Rd Strong Ordering	WrL2 Allocate	Use Old CRC	Inform Source	Transaction ID	RdL2 Allocate	Rd Purge	Msg Source	Reserved	

Bit	Name	Description
63	FWF	First Word Flag. Must be 0 if this is not first word in message.
62	WrStrongOrdering	Write Strong Ordering Control. 0: Do not strongly order writes 1: When set, writes are strongly ordered, and the bridge does a read-modify-write for unaligned addresses.

Bit	Name	Description
61	RdStrongOrdering	Read Strong Ordering Control 0: Do not strongly order reads 1: When set, reads are strongly ordered
60	WrL2Allocate	L2 Cache Write Control 0: use this setting for writes not going to L2 cache 1: set to cause writes to go to L2 cache
59	UseOldCRC	CRC Control 0: use InitCRC and InitCSUM field values 1: For multiple segments, set this bit on all but the first calculation
58	InformSource	Message Response Control 0: Do not send message back 1: Send message back to MsgSource when transaction completes
57:48	TransactionID	Software programmable field that is copied into return message if InformSource is set
47	RdL2Allocate	L2 Cache Read Control 0: Do not store reads in cache 1: When set, if the data is not present in the L2 Cache, then data is stored in L2 Cache during read.
46	RdPurge	L2 Cache Read and Purge 0: Read operation will not affect the cached data (the data will not be purged from the cache after the read operation). 1: When set, if the data is present in the cache, then after the read, the data is discarded from the cache without being written back to external memory. Warning: Use extreme caution when setting this bit. This bit does not need to be set for normal operation. Only set it when it is certain that this will be the last operation to be performed by the DTE on the associated data. If an nCPU attempts to read the data after the DTE read, the data it fetches may be stale.
45:40	MsgSource	Bucket ID to receive return message if enabled by InformSource.
39: 0	Reserved	<i>Reserved</i> Must be = 0

Third Word			
63	62:48	47:32	31:0
FWF	Reserved	InitCSUM	InitCRC

Bit	Name	Description
63	FWF	First Word Flag. Must be 0 if this is not first word in message.
62:48	Reserved	<i>Reserved</i> Must be = 0
47:32	InitCSUM	value to use to initialize CSUM
31:0	InitCRC	value to use to initialize CRC

9.3.1.5 Return Message Format

	63:62	61	60	59:58	57:48	47:32	31:0
ReturnCode	Msg Format Error	Bus Error	Channel ID	Transaction ID	Partial CSUM		Partial CRC
Bit	Name		Description				
63:62	ReturnCode		return message sends value of 3 in this field to the originator				
61	MsgFormatError		A format error was encountered 0: no error found in original message 1: error found in original message				
60	BusError		A bus error was encountered, such as double-bit ECC error on read, or misprogramming of bridge 0: no error 1: bus error detected				
59:58	ChannelID		DMA Channel ID of respondent (0 to 3)				
57:48	TransactionID		Transaction ID from source message				
47:32	PartialCSUM		Running CSUM result				
31:0	PartialCRC		Running CRC result				

9.3.2 Simultaneous Simple and Other DMA Message Types

In XLS processors, if DMA transfers of the Simple Transfer type occur simultaneously with other types (CRC, CSUM, or CRC+CSUM), then the DMA engine may behave unpredictably.

Other types of DMA messages can be used simultaneously. But instead of using the Simple Transfer message type with the others, substitute the CSUM type described in Section 9.3.1.3 above, with the following modifications:

- Set bit 60 in the first word (Note that this bit is marked as reserved and is usually left at zero).
- Write the destination address into bits 39:0 of the second word in place of the initial CSUM value.
- Note that the message type in bits 62:61 is left as CSUM (0x2).

9.4 DMA Engine Registers

The registers controlling the DMA engine reside in a memory region identified in Chapter 8, “Memory and I/O Access”

Structure of register descriptions in this chapter is:

- Register name
- Register IDs are in hexadecimal
- Register address offsets are on 4-byte boundaries
- Register text description
- Bit fields diagram
- Bit fields descriptions

Note:

1. Functions identified as not implemented in an XLS, or which are marked *Reserved*, are set to zero.
2. Reserved bits and words should not be modified by the user.
3. Unless otherwise specified, these bits read as ‘0’ and should be written as ‘0’

9.4.1 DMA Engine Register Summary.

Table 9-2. DMA Engine Register Summary

Function	Name	Register ID	Addr Offset
Debug	DEBUG_STATE0 (Channel0) DEBUG_STATE1 (Channel1) DEBUG_STATE2 (Channel2) DEBUG_STATE3 (Channel3)	0x0 0x1 0x2 0x3	0x0 0x4 0x8 0xC
Cyclic Redundancy Check Polynomial	CRC_POLY_CHANNEL0 CRC_POLY_CHANNEL1 CRC_POLY_CHANNEL2 CRC_POLY_CHANNEL3	0x4 0x5 0x6 0x7	0x10 0x14 0x18 0x1C
Channel Control	CONTROL_CHANNEL0 CONTROL_CHANNEL1 CONTROL_CHANNEL2 CONTROL_CHANNEL3	0x8 0x9 0xA 0xB	0x20 0x24 0x28 0x2C
Message Received Counters	MessageCountChannel0 MessageCountChannel1 MessageCountChannel2 MessageCountChannel3	0xC 0xD 0xE 0xF	0x30 0x34 0x38 0x3C
Message Sent Counters	MessageSentCountChannel0 MessageSentCountChannel1 MessageSentCountChannel2 MessageSentCountChannel3	0x10 0x11 0x12 0x13	0x40 0x44 0x48 0x4C
Bad Message Counters	BadMessageCountChannel0 BadMessageCountChannel1 BadMessageCountChannel2 BadMessageCountChannel3	0x14 0x15 0x16 0x17	0x40 0x44 0x48 0x4C

9.4.2 DMA Engine Configuration and Debug Register Descriptions

9.4.2.1 DEBUG_STATE0 (Channel0)

Register ID: 0x0

Address Offset: 0x0

		31:19	18	17	16:7	6:4	3:0	
		Reserved	MsgErr	BusErr	CurrMsgId	rRWState	rState	
Bit	Name	Description					R/W	Reset
31:19	Reserved	<i>Reserved</i>						R 0
18	MsgErr	Message Error on Current Transaction 0: no error found 1: error detected						R/W 0
17	BusErr	BusError on Current Transaction 0: no error found 1: error detected						R/W 0
16:7	CurrMsgId	MsgID of Current Transaction						R/W 0
6:4	rRWState	Internal Debug States (Should be Idle at the End) NetLogic factory use						R/W 0
3:0	rState	Internal Debug States (Should be Idle at the End) NetLogic factory use.						R/W 0

9.4.2.2 DEBUG_STATE1 (Channel1)

Register ID: 0x1

Address Offset: 0x4

		31:19	18	17	16:7	6:4	3:0	
		Reserved	MsgErr	BusErr	CurrMsgId	rRWState	rState	
Bit	Name	Description					R/W	Reset
31:19	Reserved	<i>Reserved</i>						R 0
18	MsgErr	Message Error on Current Transaction 0: no error found 1: error detected						R/W 0
17	BusErr	BusError on Current Transaction 0: no error found 1: error detected						R/W 0
16:7	CurrMsgId	MsgID of Current Transaction						R/W 0
6:4	rRWState	Internal Debug States (Should be Idle at the End) NetLogic factory use						R/W 0
3:0	rState	Internal Debug States (Should be Idle at the End) NetLogic factory use.						R/W 0

9.4.2.3 DEBUG_STATE2 (Channel2)**Register ID:** 0x2**Address Offset:** 0x8

31:19	18	17	16:7	6:4	3:0
Reserved	MsgErr	BusErr	CurrMsgId	rRWState	rState

Bit	Name	Description	R/W	Reset
31:19	Reserved	Reserved	R	0
18	MsgErr	Message Error on Current Transaction 0: no error found 1: error detected	R/W	0
17	BusErr	BusError on Current Transaction 0: no error found 1: error detected	R/W	0
16:7	CurrMsgId	MsgID of Current Transaction	R/W	0
6:4	rRWState	Internal Debug States (Should be Idle at the End) NetLogic factory use	R/W	0
3:0	rState	Internal Debug States (Should be Idle at the End) NetLogic factory use.	R/W	0

9.4.2.4 DEBUG_STATE3 (Channel3)**Register ID:** 0x3**Address Offset:** 0xC

31:19	18	17	16:7	6:4	3:0
Reserved	MsgErr	BusErr	CurrMsgId	rRWState	rState

Bit	Name	Description	R/W	Reset
31:19	Reserved	Reserved	R	0
18	MsgErr	Message Error on Current Transaction 0: no error found 1: error detected	R/W	0
17	BusErr	BusError on Current Transaction 0: no error found 1: error detected	R/W	0
16:7	CurrMsgId	MsgID of Current Transaction	R/W	0
6:4	rRWState	Internal Debug States (Should be Idle at the End) NetLogic factory use	R/W	0
3:0	rState	Internal Debug States (Should be Idle at the End) NetLogic factory use.	R/W	0

9.4.2.5 CRC_POLY_CHANNEL0**Register ID:** 0x4**Address Offset:** 0x10

31:0				
Polynomial				
Bit	Name	Description	R/W	Reset
31:0	Polynomial	Polynomial for CRC function	R/W	0

9.4.2.6 CRC_POLY_CHANNEL1**Register ID:** 0x5**Address Offset:** 0x14

31:0				
Polynomial				
Bit	Name	Description	R/W	Reset
31:0	Polynomial	Polynomial for CRC function	R/W	0

9.4.2.7 CRC_POLY_CHANNEL2**Register ID:** 0x6**Address Offset:** 0x18

31:0				
Polynomial				
Bit	Name	Description	R/W	Reset
31:0	Polynomial	Polynomial for CRC function	R/W	0

9.4.2.8 CRC_POLY_CHANNEL3**Register ID:** 0x7**Address Offset:** 0x1C

31:0				
Polynomial				
Bit	Name	Description	R/W	Reset
31:0	Polynomial	Polynomial for CRC function	R/W	0

9.4.2.9 CONTROL_CHANNEL0

Register ID: 0x8
Address Offset: 0x20

31:28	25	24	23:17	16:12	11	10:8	7:5	4	3	2	1:0
RSVD	Pop All Msgs	Report Error	Err Msg DstId	Section Size	Clear Tags	R MaxCr	W MaxCr	En	Abort Xfer	Crc Bit Ord	Crc Width

Bit	Name	Description	R/W	Reset
31:26	Reserved	Reserved	R	0
25	PopAllMsgs	Pop all messages for channel 0: no action performed 1: set this to pop messages. Do this when resetting DMA Engine. See Section 9.2.2 Reset Sequence .	R/W	0
24	ReportError	Report message format errors to the Bucket ID specified by ErrMsgDstID 0: do not report 1: report errors	R/W	0
23:17	ErrMsgDstID	Bucket ID of destination to which engine should report format errors. See FMN chapter description for information on Bucket ID.	R/W	0
16:12	SectionSize	16-deep data FIFO is carved for 4 channels, in units of cacheline	R/W	0
11	ClearTags	Reset tag vector Tags are used internally to confirm write visibility. This bit should not be set by the user in normal operation. In case a channel must be reset, then this bit must be set, along with the PopAllMsgs bit.	R/W	0
10:8	RMaxCr	Maximum MDI or I/O DI Credits for read FIFO (Maximum value is 4)	R/W	0
7:5	WMaxCr	Maximum MDI or I/O DI Credits for write FIFO (Maximum value is 4)	R/W	0
4	En	Channel enable 0: disable channel 1: enable channel	R/W	0
3	AbortXfer	Abort current transfer 0: no action 1: abort transfer. Use this when resetting DMA Engine. See Section 9.2.2 Reset Sequence .	R/W	0
2	CrcBitOrd	CRC bit order 0: High to low 1: Low to high	R/W	0
1:0	CrcWidth	CRC width 1X: 32 01: 16 00: 8	R/W	0

9.4.2.10 CONTROL_CHANNEL1Register ID: **0x9**Address Offset: **0x24**

31:26 25 24 23:17 16:12 11 10:8 7:5 4 3 2 1:0												
Reserved		Pop All Msgs	Report Error	Err Msg DstID	Section Size	Clear Tags	RMaxCr	WMax Cr	En	Abort Xfer	Crc Bit Ord	Crc Width
31:26	Reserved	Reserved										R 0
25	PopAllMsgs	Pop all messages for channel 0: no action performed 1: set this to pop messages. Do this when resetting DMA Engine. See Section 9.2.2 Reset Sequence .										R/W 0
24	ReportError	Report message format errors to the Bucket ID specified by ErrMsgDstID 0: do not report 1: report errors										R/W 0
23:17	ErrMsgDstID	Bucket ID of destination to which engine should report format errors. See FMN chapter description for information on Bucket ID.										R/W 0
16:12	SectionSize	16-deep data FIFO is carved for 4 channels, in units of cacheline										R/W 0
11	ClearTags	Reset tag vector Tags are used internally to confirm write visibility. This bit should not be set by the user in normal operation. In case a channel need to be reset, then this bit must be set, along with the PopAllMsgs bit.										R/W 0
10:8	RMaxCr	Maximum MDI or I/O DI Credits for read FIFO (Maximum value is 4)										R/W 0
7:5	WMaxCr	Maximum MDI or I/O DI Credits for write FIFO (Maximum value is 4)										R/W 0
4	En	Channel enable 0: disable channel 1: enable channel										R/W 0
3	AbortXfer	Abort current transfer 0: no action 1: abort transfer. Use this when resetting DMA Engine. See Section 9.2.2 Reset Sequence .										R/W 0
2	CrcBitOrd	CRC bit order 0: High to low 1: Low to high										R/W 0
1:0	CrcWidth	CRC width 1X: 32 01: 16 00: 8										R/W 0

9.4.2.11 CONTROL_CHANNEL2

Register ID: 0xA
Address Offset: 0x28

		31:26	25	24	23:17	16:12	11	10:8	7:5	4	3	2	1:0
<i>Reserved</i>		Pop All Msgs	Report Error	Err Msg DstId	Section Size	Clear Tags	R MaxCr	W MaxCr	En	Abort Xfer	Crc Bit Ord	Crc Width	
Bit	Name	Description											
31:26	<i>Reserved</i>	<i>Reserved</i>											
25	PopAllMsgs	Pop all messages for channel 0: no action performed 1: set this to pop messages. Do this when resetting DMA Engine. See Section 9.2.2 Reset Sequence .											
24	ReportError	Report message format errors to the Bucket ID specified by ErrMsgDstID 0: do not report 1: report errors											
23:17	ErrMsgDstID	Bucket ID of destination to which engine should report format errors. See FMN chapter description for information on Bucket ID.											
16:12	SectionSize	16-deep data FIFO is carved for 4 channels, in units of cacheline											
11	ClearTags	Reset tag vector Tags are used internally to confirm write visibility. This bit should not be set by the user in normal operation. In case a channel need to be reset, then this bit must be set, along with the PopAllMsgs bit.											
10:8	RMaxCr	Maximum MDI or I/O DI Credits for read FIFO (Maximum value is 4)											
7:5	WMaxCr	Maximum MDI or I/O DI Credits for write FIFO (Maximum value is 4)											
4	En	Channel enable 0: disable channel 1: enable channel											
3	AbortXfer	Abort current transfer 0: no action 1: abort transfer. Use this when resetting DMA Engine. See Section 9.2.2 Reset Sequence .											
2	CrcBitOrd	CRC bit order 0: High to low 1: Low to high											
1:0	CrcWidth	CRC width 1X: 32 01: 16 00: 8											

9.4.2.12 CONTROL_CHANNEL3

Register ID: 0xB
Address Offset: 0x2C

31:26 25 24 23:17 16:12 11 10:8 7:5 4 3 2 1:0												
Reserved	Pop All Msgs	Report Error	Err Msg DstId	Section Size	Clear Tags	R MaxCr	W MaxCr	En	Abort Xfer	Crc Bit Ord	Crc Width	

Bit	Name	Description	R/W	Reset
31:26	Reserved	Reserved	R	0
25	PopAllMsgs	Pop all messages for channel 0: no action performed 1: set this to pop messages. Do this when resetting DMA Engine. See Section 9.2.2 Reset Sequence .	R/W	0
24	ReportError	Report message format errors to the Bucket ID specified by ErrMsgDstID 0: do not report 1: report errors	R/W	0
23:17	ErrMsgDstID	Bucket ID of destination to which engine should report format errors. See FMN chapter description for information on Bucket ID.	R/W	0
16:12	SectionSize	16-deep data FIFO is carved for 4 channels, in units of cacheline	R/W	0
11	ClearTags	Reset tag vector Tags are used internally to confirm write visibility. This bit should not be set by the user in normal operation. In case a channel need to be reset, then this bit must be set, along with the PopAllMsgs bit.	R/W	0
10:8	RMaxCr	Maximum MDI or I/O DI Credits for read FIFO (Maximum value is 4)	R/W	0
7:5	WMaxCr	Maximum MDI or I/O DI Credits for write FIFO (Maximum value is 4)	R/W	0
4	En	Channel enable 0: disable channel 1: enable channel	R/W	0
3	AbortXfer	Abort current transfer 0: no action 1: abort transfer. Use this when resetting DMA Engine. See Section 9.2.2 Reset Sequence .	R/W	0
2	CrcBitOrd	CRC bit order 0: High to low 1: Low to high	R/W	0
1:0	CrcWidth	CRC width 1X: 32 01: 16 00: 8	R/W	0

9.4.2.13 MessageCountChannel0

Register ID: 0xC

Address Offset: 0x30

Read-only

NRxMsg

Bit	Name	Description	R/W	Reset
31:0	NRxMsg	Number of messages received on this channel.	R	0

9.4.2.14 MessageCountChannel1

Register ID: 0xD

Address Offset: 0x34

Read-only

NRxMsg

Bit	Name	Description	R/W	Reset
31:0	NRxMsg	Number of messages received on this channel.	R	0

9.4.2.15 MessageCountChannel2

Register ID: 0xE

Address Offset: 0x38

Read-only

NRxMsg

Bit	Name	Description	R/W	Reset
31:0	NRxMsg	Number of messages received on this channel.	R	0

9.4.2.16 MessageCountChannel3

Register ID: 0xF
Address Offset: 0x3C

Read-only

		31:0		
		NRxMsg		
Bit	Name	Description	R/W	Reset
31:0	NRxMsg	Number of messages received on this channel.	R	0

9.4.2.17 MessageSentCountChannel0

Register ID: 0x10
Address Offset: 0x40

Read-only

		31:0		
		NTxMsg		
Bit	Name	Description	R/W	Reset
31:0	NTxMsg	Number of messages sent on this channel.	R	0

9.4.2.18 MessageSentCountChannel1

Register ID: 0x11
Address Offset: 0x44

Read-only

		31:0		
		NTxMsg		
Bit	Name	Description	R/W	Reset
31:0	NTxMsg	Number of messages sent on this channel.	R	0

9.4.2.19 MessageSentCountChannel2

Register ID: 0x12

Address Offset: 0x48

Read-only

31:0

Bit	Name	Description	R/W	Reset
31:0	NTxMsg	Number of messages sent on this channel.	R	0

9.4.2.20 MessageSentCountChannel3

Register ID: 0x13

Address Offset: 0x4C

Read-only

NTxMsg

Bit	Name	Description	R/W	Reset
31:0	NTxMsg	Number of messages sent on this channel.	R	0

9.4.2.21 BadMessageCountChannel0

Register ID: 0x14

Address Offset: 0x40

Read-only

NBadMsg

Bit	Name	Description	R/W	Reset
31:0	NBadMsg	Number of bad messages on this channel.	R	0

9.4.2.22 BadMessageCountChannel1**Register ID:** 0x15**Address Offset:** 0x44

Read-only

		31:0		
		NBadMsg		
Bit	Name	Description	R/W	Reset
31:0	NBadMsg	Number of bad messages on this channel.	R	0

9.4.2.23 BadMessageCountChannel2**Register ID:** 0x16**Address Offset:** 0x48

Read-only

		31:0		
		NBadMsg		
Bit	Name	Description	R/W	Reset
31:0	NBadMsg	Number of bad messages on this channel.	R	0

9.4.2.24 BadMessageCountChannel3**Register ID:** 0x17**Address Offset:** 0x4C

Read-only

		31:0		
		NBadMsg		
Bit	Name	Description	R/W	Reset
31:0	NBadMsg	Number of bad messages on this channel.	R	0

9.4.3 DMA Engine Credit Counter Registers

See Fast Messaging Network chapter for discussion of credits.

Register ID Base Address: **0x380 – 0x3FF**

There are 128 Credit Counter Registers (one for each bucket in XLS).

Table 9-3. DMA Credit Counter Registers

Register ID	Address Offset	Description	Reset
0x380 - 0x3FF	0xE00 - 0xFFC	128 Credit Counter Registers	0

9.4.4 DMA MsgBucketSize Registers

Register ID Base Address: **0x320 – 0x323**

Table 9-4. DMA MsgBucketSize Registers

Reg ID	Address Offset	Description	Reset
0x320	0xC80	DMA Channel 0	0x40
0x321	0xC84	DMA Channel 1	0x40
0x322	0xC88	DMA Channel 2	0x40
0x323	0xC8C	DMA Channel 3	0x40

NETLOGIC
CONFIDENTIAL



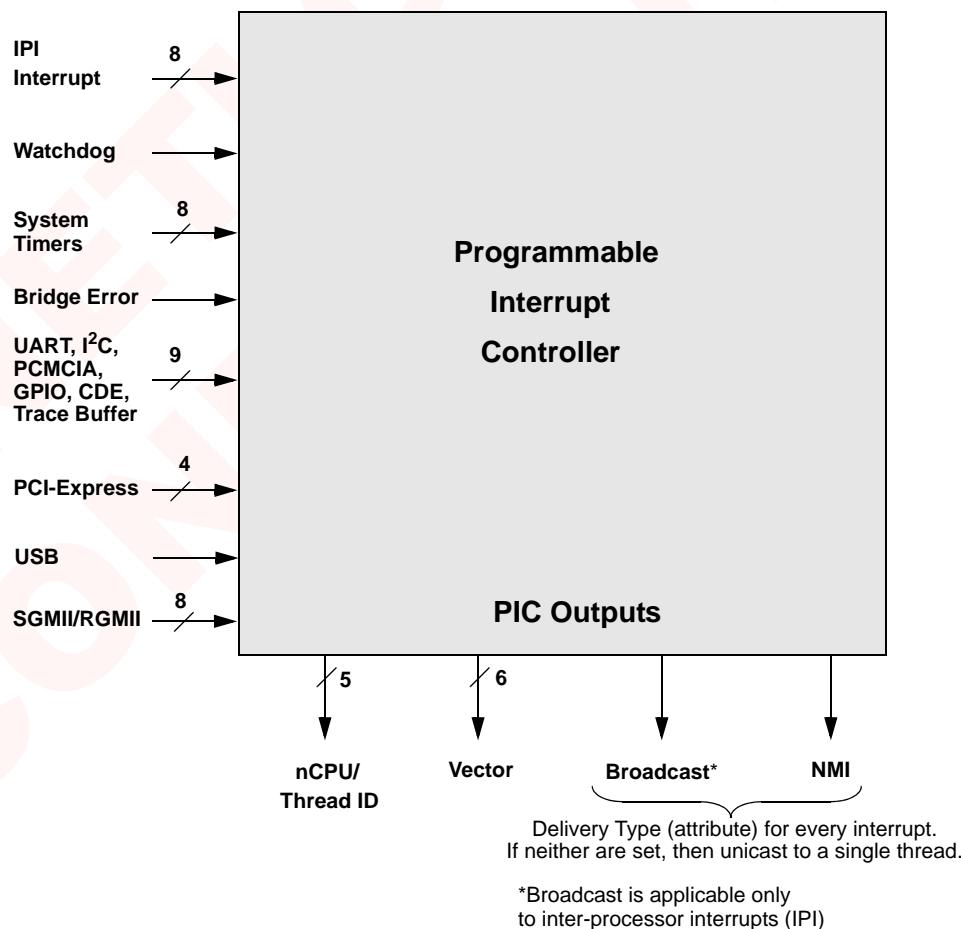
Chapter 10 Programmable Interrupt Controller

10.1 Introduction

The Programmable Interrupt Controller (PIC) receives all interrupt requests from internal sources such as timers, on-chip peripherals (GMACs, USB, PCI-E, PCMCIA, etc.), as well as external sources. The PIC synchronizes them, validates them, prioritizes them, then schedules and delivers them to threads executing in the processor cores. [Figure 10-1](#) shows a simplified view of the PIC.

Figure 10-1. Programmable Interrupt Controller Inputs and Outputs

PIC Inputs



10.1.1 PIC Functionality Overview

The core of the PIC is the Interrupt Redirection Table (IRT), shown in [Section 10.1.3](#), which determines the actions taken by the PIC for all interrupt requests. It determines the electrical characteristics of valid interrupt signals, and specifies how the PIC will direct interrupt requests to the available threads.

The IRT provides a great deal of flexibility and efficiency in interrupt handling. For example, software may use it to direct all external interrupt requests to a particular nCPU or f nCPUs; or, certain classes of interrupt request may be distributed among a pool of nCPUs using round-robin scheduling to achieve load-balanced interrupt servicing. nCPUs can also signal other nCPUs synchronously by sending inter-processor interrupts (IPI) via the PIC.

Different vector numbers potentially can be assigned to different sources, so the recipient can determine the source without polling the hardware.

When multiple recipients are programmed to receive a particular event, the PIC scheduler can be programmed to use a round-robin scheme for event delivery. For example, suppose threads 0, 4, and 7 are programmed to receive interrupts from a particular source. The PIC scheduler delivers the first interrupt request to nCPU0, the next one to nCPU4, the next one to nCPU7, and then returns to nCPU0 for the next interrupt request. Consequently, recipient threads experience a balanced load. Round-robin scheduling can be performed per interrupt source (local round-robin scheduling) or across all interrupt sources (global round-robin scheduling).

Every nCPU has its own 64-bit interrupt register. Whenever an nCPU is interrupted, the 6-bit vector number is decoded to set the corresponding bit in the interrupt register.

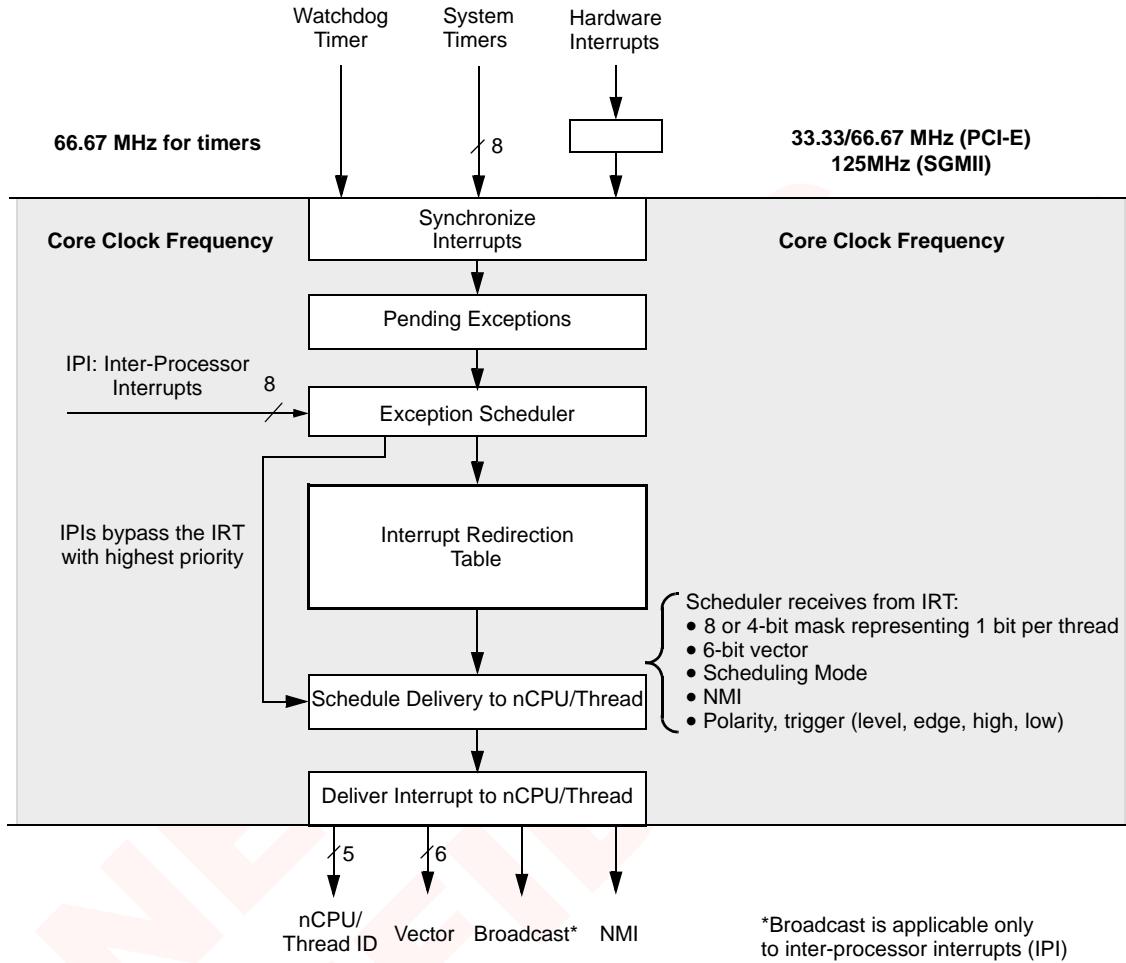
The PIC allows threads to interrupt other threads, called inter-processor interrupts (IPI) by performing a store operation to the PIC address space. The value stored specifies the interrupt vector and the target nCPU used by the PIC for this inter-nCPU interrupt. Software can use normal conventions to identify inter-processor interrupts; for example, a vector range may be Reserved for this purpose.

The PIC also has watch-dog and general-purpose system timers which can be configured to deliver interrupts to the nCPU.

10.1.2 Interrupt Processing Details

Figure 10-2 shows functional details of the PIC processing flow. Descriptions follow.

Figure 10-2. Interrupt Processing Flow



To deliver an interrupt from an asynchronous interrupt source, the PIC performs the following steps, as shown in Figure 10-2.

1. It first synchronizes incoming interrupt requests to the PIC core clock domain. Synchronization is required because interrupt requests must be transferred from the source clock domain to the core clock domain.
2. Each new interrupt request is qualified by examining the corresponding entry in the Interrupt Redirection Table (IRT) to determine if the interrupt condition is valid. The polarity (positive- or negative-logic) and trigger type (level, or edge) of the interrupt signal are conditioned by entries in the IRT. A Valid Entry bit is checked to see if the interrupt request is enabled. If the interrupt request is valid, the interrupt is marked Pending.
3. If multiple interrupt requests are pending at the same time, the PIC schedules them for processing one-by-one using Round-Robin (RR) scheduling. This scheduling step determines which interrupt request the PIC will take up next.
4. After an interrupt request has been scheduled, its IRT entry is accessed again to get the remaining attributes of the interrupt.

5. Now a 2nd Round-Robin scheduling operation takes place to decide where to send the interrupt request. The PIC determines the receiving nCPU by examining the IRT's Thread Mask field to determine candidate threads. The PIC determines whether local or global round-robin scheduling is enabled for this source. For local round-robin scheduling, a pointer local to this interrupt source is incremented with wrap-around to point to the next recipient nCPU in the Thread Mask. For global round-robin scheduling, a pointer global to all interrupt sources is incremented with wrap-around to determine the next recipient nCPU.
- Inter-processor interrupts (IPI) can be unicast or broadcast. (Hardware interrupts cannot be broadcast.)
6. The PIC then sends the interrupt to the receiving nCPU (unicast) or nCPUs (broadcast). The information sent includes the selected thread ID, the interrupt vector, and whether the interrupt request is non-maskable (NMI). The interrupt is then marked Delivered.

The interrupt request must be acknowledged by the receiving core before the PIC will process an interrupt request from that source again. The receiving nCPU acknowledges the interrupt request by writing to the Interrupt Acknowledge Register (IAR). This is bit-mask of all interrupt sources, and thus one or more interrupts can be acknowledged with one write to the IAR. Acknowledgment clears the delivered and pending state for the interrupt.

10.1.3

Interrupt Redirection Table (IRT)

To process interrupts, the PIC's Interrupt Redirection Table (IRT) must be initialized for each interrupt source that is to be serviced. Interrupting devices must also be initialized. For example, the watchdog and system timers must have their counter values programmed, and the timers must be enabled.

The 32-entry IRT describes how the PIC should respond to each source of interrupt requests. Each entry in the IRT includes the following fields:

- Valid Bit: determines whether requests will be accepted from this interrupt source
- Trigger Type: determines whether the interrupt signal is level- or edge-triggered
- Polarity: determines whether the interrupt signal uses positive or negative logic
- Delivery Type: the interrupt request sent is non-maskable
- Interrupt Vector: this 6-bit value determines which of the 64 possible interrupt priorities are associated with this interrupt request; it is used in conjunction with the EIRR register to determine the source of an interrupt (excluding NMIs).
- Thread Mask: this 8, 4, or 2-bit mask (depending on processor model) determines which threads can receive the interrupt
- Schedule: whether a round-robin pointer local to the interrupt source or the global round-robin pointer is used for the 2nd level scheduling to select the recipient.

The PIC also supports inter-processor interrupts (IPI). IPIs can also be delivered from one processor to another by performing a write transaction to the IPI register in the PIC. IPIs have higher priority than other interrupts. IPIs can also be broadcast to every single enabled Thread. (It is recommended that the processor sending the IPI precede it with a sync instruction immediately before the IPI is issued. This ensures that data stored by the source processor becomes visible to the target processor before the target processor receives the interrupt.)

10.1.4

Global and Local Scheduling to Select the Processor Core

The second level of round-robin scheduling (Step 6 above) can be directed by a local round-robin pointer or a global round-robin pointer. This choice is determined by the Scheduling Mode bit in the IRT.

In local mode, each interrupt source maintains its own (local) round-robin pointer which is updated when an interrupt from that source is processed. This allows the processing of a particular interrupt source to be equally distributed to each nCPU specified in the Thread Mask.

All interrupt sources not marked for local scheduling use a single global round-robin pointer that distributes interrupts to all eligible threads.

10.1.5 Description of Interrupt Redirection Table

The 32-entry Interrupt Redirection Table (IRT) can be used to map external interrupts to a priority level, specify the core or nCPU to receive the interrupt, and specify the trigger mechanism (edge, level, rising, falling). The IRT structure differs slightly depending on processor model. [Table 10-1](#) shows the fields of a single IRT entry.

The IRT Table is indexed by the Interrupt Source/Number (see [Table 10-3](#)). Note that IPIs do not use the IRT (see [Figure 10-2](#)).

Table 10-1. Interrupt Redirection Table Entry

63 62 61 60:40 39 38 37:32 31:8								7:0			
Vld	Trg	P	Reserved	NMI	GL	IntVec	Reserved	ThreadMask			
63	Vld	Valid. 0: Invalid IRT entry 1: Valid IRT entry									
62	Trg	Trigger. See Table 10-2 . 0: Edge triggered 1: Level triggered									
61	P	Polarity. See Table 10-2 . If Trigger = 0: 0: Rising edge 1: Falling edge Else if Trigger = 1: 0: Level High 1: Level Low									
60:40	Reserved	Reserved. Write value is discarded, reads back as zero.									
39	NMI	Maskable/non-maskable 0: Maskable (default) 1: Non-maskable.									
38	GL	Global/Local 0: Global — Interrupt delivery uses the Global round-robin scheduler 1: Local — Uses the Local round-robin scheduler dedicated for this interrupt									
37:32	IntVec	The Interrupt Vector gets mapped to a bit position in the nCPU's EIRR. It can be used by software to identify interrupt source and assign priorities. (Note that NMIs are the exception to this rule; the source of an NMI is not recorded in the EIRR register.)									

Bit #	Field Name	Description	R/W	Reset
31:16	Reserved	Reserved. Software should write 0 to this field.	R/W	0
15:0	ThreadMask	Mask of threads enabled to receive the interrupt. Structure of this mask depends on processor model as follows: XLS616 and XLS416: 16 threads using 16-bit mask on bits[15:0] XLS608, XLS408-Lite, XLS408, XLS208 and XLS108: 8 threads using 8-bit mask on bits [7:0] XLS404-Lite, XLS404, XLS204 and XLS104: 4 threads, using 4-bit mask as follows: Bits[5:4]: highest 2 bits of mask Bits[3:2]: Reserved, write 0 to these bits Bits[1:0]: lowest 2 bits of mask	R/W	0

The Trigger-Mode/Polarity combinations are shown in [Table 10-2](#).

Table 10-2. Trigger-Mode/Polarity Combinations

Trigger	Polarity	Interrupt Type
0	0	ER = Edge-Triggered (Rising Edge)
0	1	EF = Edge-Triggered (Falling Edge)
1	0	LH = Level-Triggered (High Level)
1	1	LL = Level-Triggered (Low Level)

Table 10-3 shows the IRT interrupt source map.

Table 10-3. PIC Interrupt Redirection Table Map

Entry	Device Families	Name	Interrupt Type	Interrupt
0	XLS6xx, XLS4xx-Lite, XLS4xx, XLS2xx, XLS1xx	PIC_IRT_ENTRY_WATCHDOG	LH	Watchdog
1:8		PIC_IRT_ENTRY_TIMERS	LH	Timers
9		PIC_IRT_ENTRY_UART0	LH	UART0
10		PIC_IRT_ENTRY_UART1	LH	UART1
11		PIC_IRT_ENTRY_I2C0	LH	I ² C0
12		PIC_IRT_ENTRY_I2C1	LH	I ² C1
13		PIC_IRT_ENTRY_PCMCIA	LH	PCMCIA
14		PIC_IRT_ENTRY_GPIO_A	LH	GPIO_A
15		PIC_IRT_ENTRY_CDE	LH	Compression/Decompression Engine <i>(Reserved in XLS2xx and XLS1xx)</i>
16		PIC_IRT_ENTRY_BRIDGE_TB	LH	System Bridge Controller TB
17		PIC_IRT_ENTRY_GMAC0	LH	GMAC0 (Port SGMII0 or Port RGMII)
18		PIC_IRT_ENTRY_GMAC1	LH	GMAC1
19		PIC_IRT_ENTRY_GMAC2	LH	GMAC2
20		PIC_IRT_ENTRY_GMAC3	LH	GMAC3
21		PIC_IRT_ENTRY_GMAC4 / <i>Reserved</i>	LH	GMAC4 (<i>Reserved in XLS2xx and XLS1xx</i>)
22		PIC_IRT_ENTRY_GMAC5 / <i>Reserved</i>	LH	GMAC5 (<i>Reserved in XLS2xx and XLS1xx</i>)
23		PIC_IRT_ENTRY_GMAC6 / PIC_IRT_ENTRY_PCIE_LINK2	LH	GMAC6 (this entry is "PCIe Link 2 interrupts" in XLS2xx) - (<i>Reserved in XLS1xx</i>)
24		PIC_IRT_ENTRY_GMAC7 / PIC_IRT_ENTRY_PCIE_LINK3	LH	GMAC7 (this entry is "PCIe Link 3 interrupts" in XLS2xx) - (<i>Reserved in XLS1xx</i>)
25		PIC_IRT_ENTRY_BRIDGE_ERR	LH	Bridge ERR, any of: AERR, BEER, or AERR_NMI
26	XLS6xx/4xx	PIC_IRT_ENTRY_PCIE_LINK0 ^a	LH	PCIE_LINK0 (<i>or SRIO Controller 0 INT</i>)
	XLS4xx-Lite, XLS2xx/1xx	PIC_IRT_ENTRY_PCIE_LINK0	LH	PCIE_LINK0
27	XLS6xx/4xx	PIC_IRT_ENTRY_PCIE_LINK1 ^a	LH	PCIE_LINK1 (<i>or SRIO Controller 1 INT</i>)
	XLS4xx-Lite, XLS2xx/1xx	PIC_IRT_ENTRY_PCIE_LINK1	LH	PCIE_LINK1
28	XLS6xx/4xx	PIC_IRT_ENTRY_PCIE_LINK2 ^a	LH	PCIE_LINK2 (<i>or SRIO Controller 2 INT</i>)
	XLS4xx-Lite, XLS2xx/1xx	<i>Reserved</i>	LH	<i>Reserved</i>
29	XLS6xx/4xx	PIC_IRT_ENTRY_PCIE_LINK3 ^a	LH	PCIE_LINK3 (<i>or SRIO Controller 3 INT</i>)
	XLS4xx-Lite, XLS2xx/1xx	PIC_IRT_ENTRY_PCIE_FATAL	LH	PCIE Fatal Error
30		PIC_IRT_ENTRY_GPIO_B	LH	GPIO_B
31		PIC_IRT_ENTRY_USB	LH	USB

a. In PCIe-Mode, PCIE_LINKn INT or Fatal INT
In SRIO-Mode, SRIO_LINKn INT

10.1.6

Per-Thread Interrupt Registers

Each nCPU contains two 64-bit registers for capturing and masking interrupts:

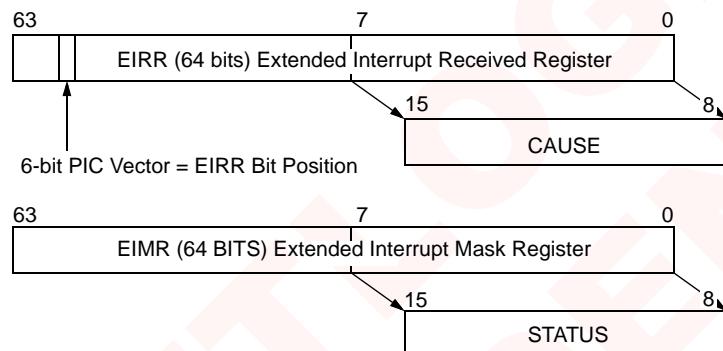
- **EIRR** (Extended Interrupt Received Register)
- **EIMR** (Extended Interrupt Mask Register).
 - Interrupt Vectors 0, 1, and 7 are reserved, and if an interrupt is delivered to the nCPU with these vectors, it will be ignored.

Since MIPS64 specifies 8-bit chunks in the STATUS and CAUSE registers for interrupt handling, the lower 8 bits of the EIRR and EIMR are shadowed to create the MIPS64 CAUSE and STATUS registers. See [Figure 10-3](#).

The lower 8 bits of the EIRR are shadowed with bits [15:8] of the MIPS64 CAUSE Register. This register is specified by MIPS to contain the interrupts which are pending at any given time.

The lower 8-bits of the EIMR are shadowed with bits [15:8] of the MIPS64 STATUS Register. This register is defined by MIPS and contains the Interrupt Mask Bits.

Figure 10-3. Per-Thread Registers for Exceptions



10.1.7

Interrupts from External Sources

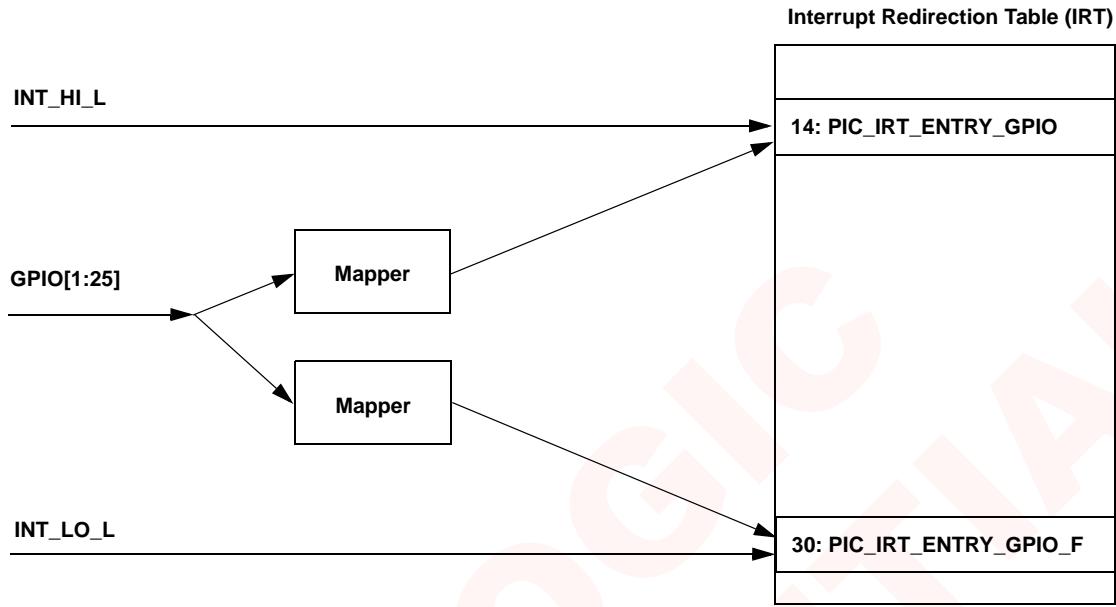
XLS Processors contain 25 General Purpose I/O (GPIO) pins which can be used for external hardware interrupts. Total number of available pins depends on system design. If the application requires PCMCIA, the lower 10 pins of the GPIO can be configured as PCMCIA. Eleven of the GPIO signals are used for Parity, CPU ID, and other Flash interface signals. This leaves the upper 4 bits of the GPIO for external interrupts or other general-purpose applications.

The XLS can accept external interrupts from two sources:

1. The INT_HI_L and INT_LO_L pins
2. GPIO pins, mapped into the IRT.

This is shown in [Figure 10-4](#).

Note: The terms HI and LO in the pin names INT_HI_L and INT_LO_L do not have any special level-related significance. They are just names equivalent to A and B.

Figure 10-4. Interrupts from External Sources and Their Mappings into IRT

10.1.7.1 INT_HI_L and INT_LO_L

These pins are not part of the GPIO and are dedicated for interrupt use. These pins are bi-directional and can be used by the XLS to trigger interrupts in the device this pin is connected to, or alternatively the XLS could be interrupted by an external device on this pin. An active-low on this pin generates an interrupt. For usage of these pins and related programming, see discussion of **GPIO_INT_MAP** and **GPIO_EXT_INT** registers in “GPIO and the Peripherals Interface” chapter.

Note that **GPIO_INT_STATUS[25]** indicates a **INT_HI_L** interrupt and **GPIO_INT_STATUS[26]** indicates **INT_LO_L** interrupt.

10.2 Programmable Interrupt Controller Timers

The Programmable Interrupt Controller (PIC) also provides a 64-bit WatchDog timer and eight 64-bit general-purpose System Timers, all running at 66.67 MHz. Each of these timers has an entry in the IRT Table and can be configured just like any other interrupts.

The WatchDog timer requires configuration before it can be used. The WatchDog Max Value register contains the value of the counter that will be reloaded every time it starts counting.

The Heartbeat Register maintains a record of which threads have acknowledged with a heartbeat. The Heartbeat Mask register quantifies which threads need to acknowledge. If all the required Heartbeats are not received before the Timer reaches 0, an interrupt will be generated.

The general-purpose System Timers must also have their Max Value registers set and must be enabled before they can be used.

These registers are listed in [Table 10-4](#).

10.3 Programmable Interrupt Controller Registers

Because the register set is uncomplicated, this chapter presents the registers in concise tabular form rather than the format used in the other chapters. [Table 10-4](#) shows the address map for PIC registers. Address offsets are from the base address of the XLS_IO_DEV_PIC, which is shown in [Table 8-1](#). Unlike other system devices, the PIC does not have a Base Address Register for relocation, but merely relocates along with the entire SBC if the SBC is moved.

The 64-bit registers such as timers and IRT are addressed in 32-bit chunks, as indicated. The LSB of all timers is a time unit of 1/66.67 MHz.

Register IDs are presented in hexadecimal.

Register address offsets are four times the hexadecimal ID values.

10.3.1 Programmable Interrupt Controller Registers Summary and Descriptions

Table 10-4. List of PIC Registers

Register			Description	R/W	Reset																								
Reg ID	Name	Width																											
0x0	Control	32	Control: <table border="1"> <thead> <tr> <th>Bits</th><th>Name</th><th>R/W</th><th>Reset</th></tr> </thead> <tbody> <tr> <td>0</td><td>WatchDogEn</td><td>R/W</td><td>0</td></tr> <tr> <td>2:1</td><td>Generate NMI after the specified number of WatchDog Interrupts.</td><td>R/W</td><td>'11'</td></tr> <tr> <td>7:3</td><td>Reserved</td><td>RO</td><td>0</td></tr> <tr> <td>15:8</td><td>Timer Enables 7 through 0</td><td>R/W</td><td>0</td></tr> <tr> <td>31:16</td><td>Reserved</td><td>RO</td><td>0</td></tr> </tbody> </table>	Bits	Name	R/W	Reset	0	WatchDogEn	R/W	0	2:1	Generate NMI after the specified number of WatchDog Interrupts.	R/W	'11'	7:3	Reserved	RO	0	15:8	Timer Enables 7 through 0	R/W	0	31:16	Reserved	RO	0	0x0000 0006	
Bits	Name	R/W	Reset																										
0	WatchDogEn	R/W	0																										
2:1	Generate NMI after the specified number of WatchDog Interrupts.	R/W	'11'																										
7:3	Reserved	RO	0																										
15:8	Timer Enables 7 through 0	R/W	0																										
31:16	Reserved	RO	0																										
0x4	IPIBase	32	IPI Base [5:0]: Vector Number 7: Broadcast 8: NMI [31:16]: CPU ID: – [31:23]: Reserved – [22:20]: Physical CPU ID – [19:18]: Reserved – [17:16]: Thread ID	R/W	–																								
0x6	InterruptAck	32	Interrupt Ack 32 bits, one for each interrupt	R/W	0																								
0x8	WatchDog.MaxValue0	32	WatchDog Max Value — Chunk 0 — Least-significant Word	R/W	0																								
0x9	WatchDog.MaxValue1	32	WatchDog Max Value — Chunk 1 — Most-significant Word	R/W	0																								
0xA	WatchDog.Mask0	32	WatchDog Mask — Chunk0	R/W	0																								
0xB	WatchDog.Mask1	32	WatchDog Mask — Chunk1	R/W	0																								
0xC	WatchDog.HeartBeat0	32	WatchDog HeartBeat — Chunk0	R/W	0																								
0xD	WatchDog.HeartBeat1	32	WatchDog HeartBeat — Chunk1	R/W	0																								
0x40	IRTEntryC0_0	32	IRT Entry 0 — Chunk 0	[31:0]: Interrupt Delivery nCPU Mask	R/W 0																								
...	IRTEntryC0_1..N	32	IRTEntry 1..30 — Chunk 0		R/W 0																								
0x5F	IRTEntryC0_31	32	IRT Entry 31 — Chunk 0		R/W 0																								

Table 10-4. List of PIC Registers (continued)

Register			Description	R/W	Reset
Reg ID	Name	Width			
0x80	IRTEEntryC1_0	32	IRT Entry 0 — Chunk 1 [37:32]: Interrupt Vector 38: Scheduling - 0 = Global,1 = Local 39: NMI	R/W	0
...	IRTEEntryC1_N	32	IRTEntry 1..30 – Chunk 1 61: Polarity: 0 = High, 1 = Low 62: Trigger: 0 = edge, 1 = level 63: Valid		0
0x9F	IRTEEntryC1_31	32	IRT Entry 31 — Chunk 1		0
0x100	SysTmrMaxValC0_0	32	System Timer MaxValue 0 — Chunk 0	R/W	0
...	SysTmrMaxValC0_N	32	...	R/W	0
0x107	SysTmrMaxValC0_7	32	System Timer MaxValue 7 — Chunk 0	R/W	0
0x110	SysTmrMaxValC1_7	32	System Timer MaxValue 0 — Chunk 1	R/W	0
...	SysTmrMaxValC1_N	32	...	R/W	0
0x117	SysTmrMaxValC1_7	32	System Timer MaxValue 7 — Chunk 1	R/W	0
0x120	SysTmrC0_0	32	System Timer 0 — Chunk 0	R/W	0
...	SysTmrC0_N	32	...	R/W	0
0x127	SysTmrC0_7	32	System Timer 7 — Chunk 0	R/W	0
0x130	SysTmrC1_0	32	System Timer 0 — Chunk 1	R/W	0
...	SysTmrC1_N	32	...	R/W	0
0x137	SysTmrC1_7	32	System Timer 7 — Chunk 1	R/W	0

NETLOGIC
CONFIDENTIAL



Chapter 11 DRAM Controllers and Interface

11.1 Introduction

This chapter describes the DRAM controller and device configurations, operation and programming model for the XLS Processor Family. The XLS Processor Family provides the designer with many options for balancing total XLS Processor system cost and performance. This chapter, together with [Chapter 8, “Memory and I/O Access”](#), provides information to help make such system-level decisions and provides details for the system programmer to create supporting operational code for the external DRAM.

Note that it is beyond the scope of this document to describe in detail the features and operations of the DDR2 memory protocol. It is assumed that the reader of this chapter is already familiar with these protocols, as described in the JEDEC DDR2 specifications.

Chapter contents:

- Operating Capabilities
- DRAM Subsystem Elements - identifies main modules, describes them at a high level, and introduces terminology
- DRAM Device Types and Channel Configurations - discusses DRAM types for use with the controller
- DDR2 DRAM Interface Signals
- External DRAM Configurations - identifies and illustrates external circuitry
- Programming Model
- Memory Controller Register Descriptions

11.2 Operating Capabilities

The DRAM memory controllers can interface with either DIMMs or stand-alone DRAM chips soldered onto a motherboard. Each memory channel is able to support up to two registered or two unbuffered DIMMs. (The XLS2xx series supports two buffered or registered DIMMs, and the XLS1xx series supports only one buffered or registered DIMM)

The four-banked XLS Processors are capable of supporting up to 64 GB of DRAM memory (up to 16 GB per channel). Peak theoretical data bandwidth is 12.8 GB/second (DDR2 operating at 800 bits/second).

With a few exceptions, the XLS Processor DRAM memory controllers support the full feature sets of the DDR2 memory protocols. The exceptions are:

- x 4 DRAM chips are not supported
- The DM (data mask) pins are not used. Partial-line-writes are performed by doing data-merging within the memory controller.
- The optional RDQS feature (i.e. separate DQS for reads and writes) is not supported.
- The OCD (Off-Chip Driver) impedance adjustment feature is not supported, since most DRAM chip manufacturers do not support it either.

11.3 DRAM Subsystem Elements and Descriptions

The DRAM system's major elements are shown in [Figure 11-1](#). The operation and programming of the two Memory Bridges and Bridge Controllers are covered in [Chapter 8, "Memory and I/O Access"](#).

Figure 11-1. DRAM System Elements, XLS616 and XLS608

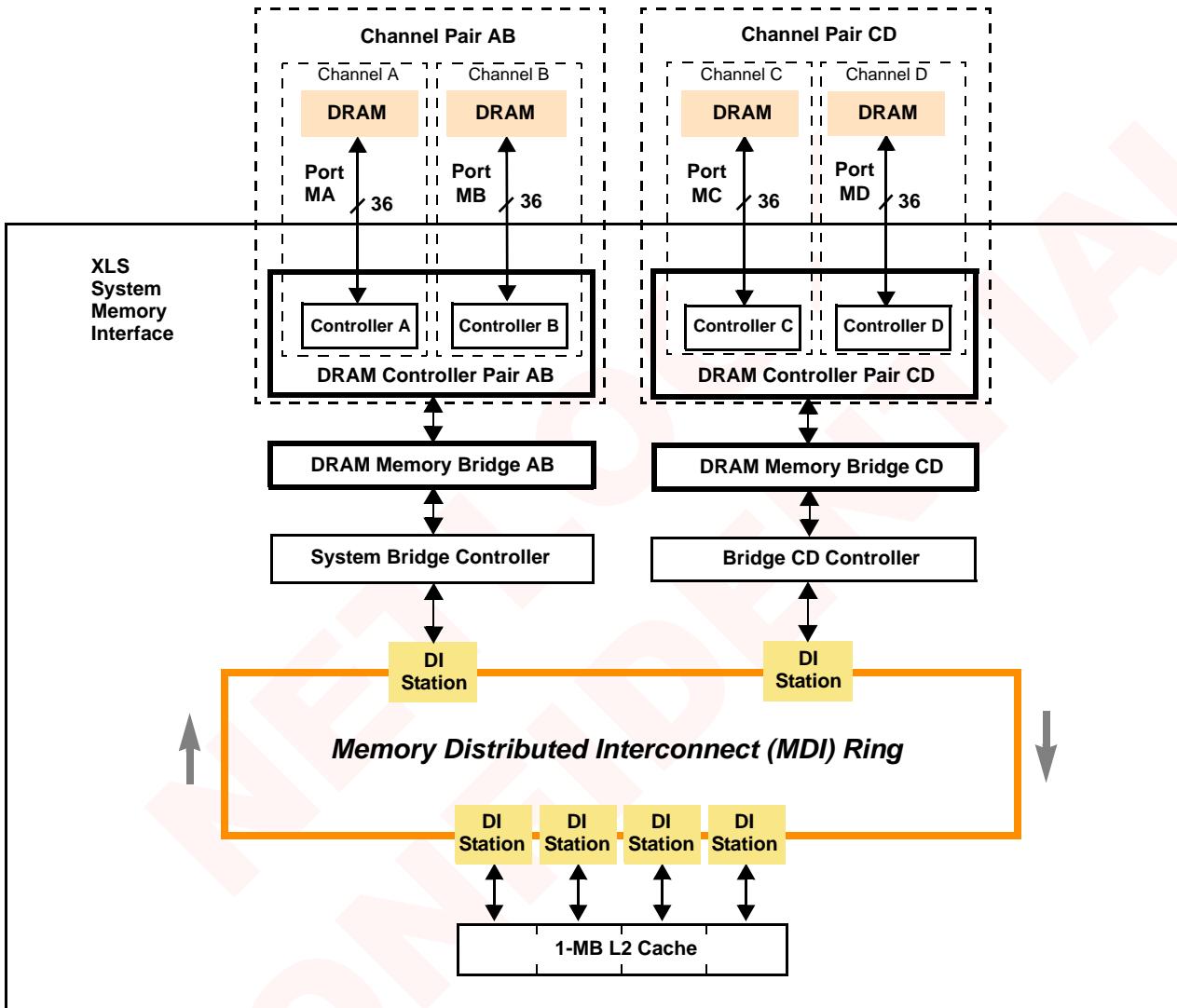


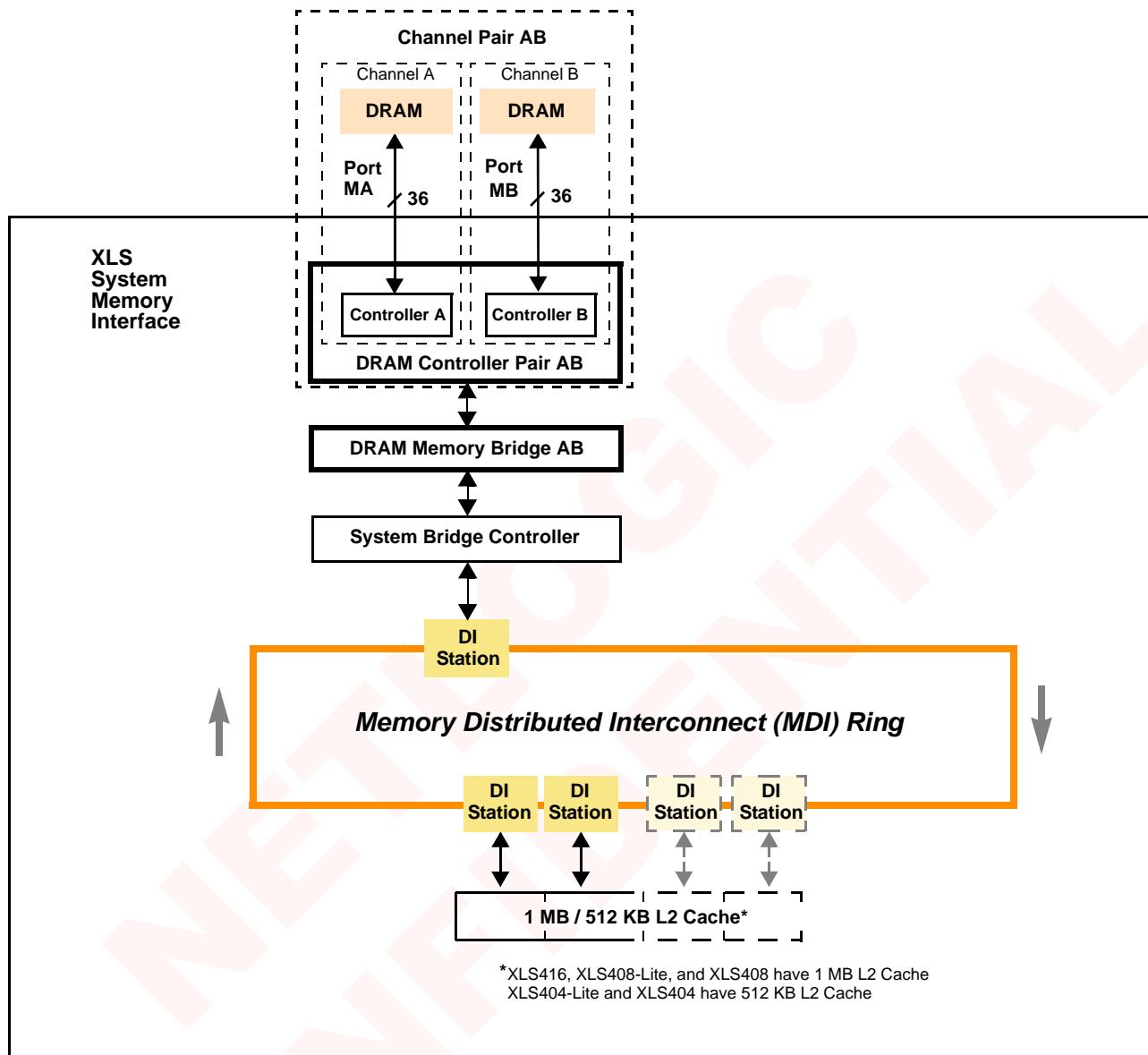
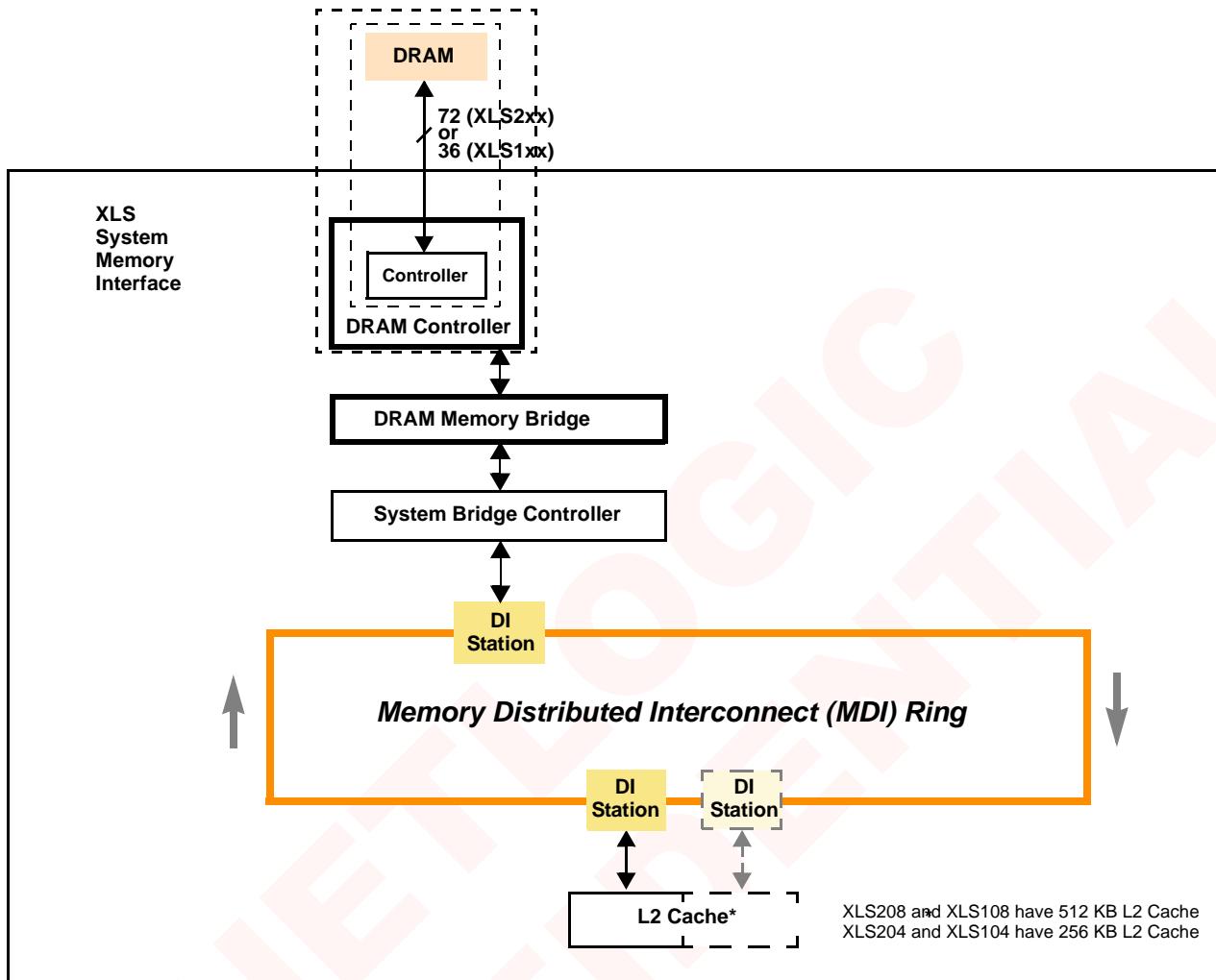
Figure 11-2. DRAM System Elements, XLS4xx-Lite and XLS4xx Devices

Figure 11-3. DRAM System Elements, XLS2xx and XLS1xx Devices

The number of DDR2 DRAM Controllers in an XLS device varies by device model number. For example, in the XLS6xx there are four independent DRAM controllers (A through D). The XLS4xx-Lite and XLS4xx provides two DRAM Controllers, and the XLS2xx and XLS1xx-class of devices contain a single DRAM Controller. [Table 11-1](#) Shows the number of controllers and supported DDR2 DRAM modes for the XLS device types.

Table 11-1. Supported DDR2 DRAM Configurations

XLX Device	Number of Controllers	Supported DDR2 DRAM Modes				
		1x36	2x36	4x36	1x72	2x72
XLS1xx	1	yes	no	no	no	no
XLS2xx	1	yes	no	no	yes	no
XLS4xx-Lite and XLS4xx	2	yes	yes	no	yes	no
XLS6xx	4	yes	yes	yes	yes	yes

Each controller supports a memory data channel of 32 bits with an additional 4 check bits for parity or ECC. Controllers A through D have 36-bit channel physical interfaces identified as

Port MA through MD. In all devices except the XLS1xx devices, 1x72b mode is supported. In all devices including the XLS2xx and XLS1xx devices, 1x36b mode is supported.

The Synchronous DRAM controllers can operate individually, or (where more than one DRAM controller exists) in combination as Channel Pairs (AB or CD) of 72 bits. The first external memory Channel Pair consists of DRAM Controller Pair AB (i.e., Controllers A and B). Similarly, DRAM Controllers C and D can operate either individually or in combination on Channel Pair CD.

Controllers connect to the MDI ring through a DRAM Memory Bridge and a Bridge Controller.

11.3.1 Terminology

This section defines the terminology used in this chapter.

Port: A group of logically-related signals on the XLS processor DRAM System Memory Interface. The four Ports are identified as MA, MB, MC, or MD. All Port signal names share a common prefix, for example, all MA Port signal names begin with DDRA_.

Controller: A standalone entity with all the necessary circuitry to control a DDR2 memory Channel. The XLS memory controllers are identified as A, B, C, or D, corresponding to the port with which they are connected.

Channel: A logical grouping of a Controller, the corresponding Port, and the DRAM memory attached to that Port. For example, Channel A consists of Controller A, Port MA, and the DRAM memory attached to Port MA.

Controller Pair: The four DRAM Controllers on the XLS6xx are physically organized as two Controller Pairs; this is reflected in configuration options and register programming. Each Controller Pair interfaces with an XLS DRAM Bridge Controller. Controller Pair AB consists of Controllers A and B, and Controller Pair CD consists of Controllers C and D. For programming purposes, the term “Controller Pair” refers to the grouping of two controllers, but does not necessarily imply that those two controllers are configured as a single 72-bit Channel Pair.

Channel-Pair Mode: Each Controller-pair may be configured in one of two ways:

1. The two constituent controllers operate completely independently, each controlling a 36-bit (32 data bits + 4 ECC bits) DRAM Channel.
2. One of the controllers is Master, controlling a 72-bit (64 data bits + 8 ECC bits) DRAM Channel Pair, utilizing two Ports. The other controller is Slave and is completely unused. This is referred to as Channel-pair Mode. Within Controller-pair AB, Controller A becomes the Master and B becomes the Slave. Within Controller-pair CD, Controller C becomes the Master, and D becomes the Slave. (Note that for XLS2xx products which provide only a single DRAM controller, 72-bit mode is still supported, since the controller is able to use some of the DDRB signals to support the wide mode.) – (Note however that the XLS1xx series does not support the 72-bit interface)

Bank: Within each DDR2 DRAM device, the memory array is subdivided into 4 or 8 banks. The individual bank is selected using the DDR[A/B/C/D]_BA Bank Address signals.

Rank: A unit within any banking organization that is implemented outside of a DRAM chip. For example, a number of DRAM chips may be grouped in multiple ranks. Chips within a rank share a common CS Chip Select signal, i.e. any issued command is simultaneously applied to all the chips within the rank. Different ranks act independently from one another. A typical DDR2 DIMM contains 1 or 2 ranks. Each XLS DRAM memory controller supports up to four ranks.

Memory Bridge: A module that interfaces between the DRAM memory controllers and the MDI ring. Some of its functions include address translation and memory transaction distribution.

11.3.2 DRAM Device Types and Characteristics

An XLS Processor supports DRAM device types as shown in [Table 11-2](#) below.

Table 11-2. Comparison of Supported DRAM Types in XLS Processors

Parameter	DDR2	Units
Bit Transfer Rate	400/533/666/800	Mbps
Device Clock Frequency	200/266/333/400	MHz
Prefetch	4	Bits
Burst Length	4/8	Cycles

Note: The actual controller clock frequency can be programmed in multiples of 33 MHz, from 100 MHz to 400 MHz, that is, 100 MHz, 133, 166, 200, and so on.

Operating configuration can be as separate 36-bit Channels or as a 72-bit Channel Pair.

[Table 11-3](#) identifies all possible combinations of supported device types and channel configurations. It also illustrates the configurations and terminology.

Table 11-3. Memory Device Types and Channel Configurations

DRAM Controller Pair AB (72-bit)		DRAM Controller Pair CD (72-bit)	
Channel Pair AB (Ports MA & MB)		Channel Pair CD (Ports MC & MD)	
Controller A (36-bit)	Controller B (36-bit)	Controller C (36-bit)	Controller D (36-bit)
Channel A (Port MA)	Channel B (Port MB)	Channel C (Port MC)	Channel D (Port MD)
DDR2	DDR2	DDR2	DDR2
	DDR2		DDR2
DDR2	DDR2	DDR2	
DDR2		DDR2	

11.3.3 DDR2 DRAM Interface Signals

Each memory port has the same set of signals. The following table shows the generic signal names, that is, names independent of port name. To form a name specific to a port, adjust the name depending on controller ID (A, B, C, or D).

Examples:

For Port MB and generic signal DDR_CK[1]P, the specific name for that port signal would be DDRB_CK1P.

For Port MC and generic signal DDR_DQS[2]N, the specific name for that port signal would be DDRC_DQS2N.

Table 11-4. Generic DDR Interface Signal Descriptions for Ports MA - MD

Generic Signal Name	Number of Signals	I/O	Description
DDR_CK[2:0]P, N	6	Out	Differential Clocks. Three parallel outputs.
DDR_CB[3:0]	4	I/O	DDR Check Bits
DDR_NC	4	I/O	NC
DDR_DQ[31:0]	32	I/O	Data bus
DDR_DQS[4:0]P, N	10	I/O	Differential Data Strobe per byte
DDR_NC	6	I/O	NC / DK Write strobe per 9-bit output
DDR_BA[2:0]	3	Out	Bank Address
DDR_CS[3:0]_L	4	Out	Chip Select
DDR_MA[15:0]	16	Out	Memory Address
DDR_RAS_L	1	Out	RAS
DDR_CAS_L	1	Out	CAS
DDR_WE_L	1	Out	Write Enable
DDR_CKE	1	Out	Clock Enable
DDR_ODT[1:0]	2	Out	On-Die Termination

11.3.4 DRAM Configurations Supported by XLS

Table 11-6 lists examples of DRAM configurations supported by the XLS Processor family. The memory can be mounted directly on a system motherboard or on DIMMs of either the unbuffered or the registered type.

Table 11-5. Representative Examples of Supported DRAM Configurations

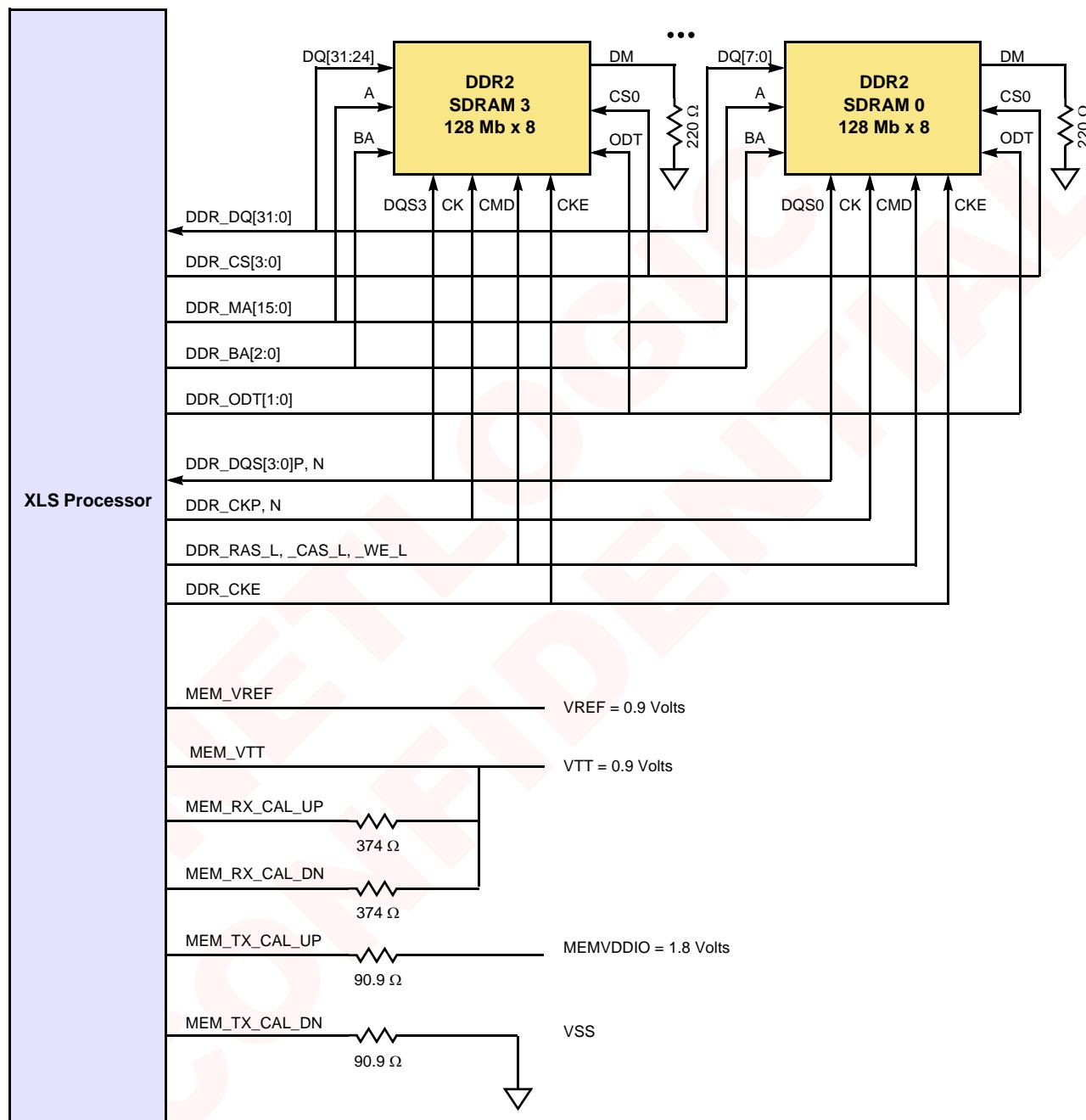
Mem. Type	System Total ^a	Configuration	Ports	Port Capacity ^a	Channel Size (CH. Pair Size)	Total Ranks	Device Size (Density)	Devices per Rank	Rank Data Capacity ^a
DDR2 Discrete ICs ^b	64 MB	One single 32-bit channel, no ECC	1	64 MB	32b (NA)	1	16 M x 16b (256 Mb)	2	64 MB
	2 GB	Two 72-bit channel pairs, with ECC	4 ^c	512 MB	NA (72b)	4	64 M x 8b (512 Mb)	9	512 MB
	8 GB ^d	Four single 36-bit channels, with ECC	4	2 GB	36b (NA)	16	128 M x 8b (1Gb)	5	512 MB
Mem. Type	System Total ^a	Configuration	Ports	Port Capacity ^a	Channel Size (CH. Pair Size)	Total Ranks	Device Size (Density)	Devices per Rank	Single DIMM Data Capacity ^a
DDR2 DIMMs ^e	128 MB	One 64-bit channel pair, with one single-rank DIMM, no ECC	2 ^c	64 MB	NA (64b)	1	16 M x 16b (256 Mb)	4	128 MB
	2 GB	Two 72-bit channel pairs, each with two dual-rank DIMMs, with ECC	4 ^c	512 MB	NA (72b)	8	32 M x 8b (256 Mb)	9	512 MB
	8 GB ^d	Two 72-bit channel pairs, each with two dual-rank DIMMs, with ECC	4 ^c	2 GB	NA (72b)	8	128 M x 8b (1Gb)	9	2 GB

- a. Although data capacities are shown in bytes, some configurations include a ninth bit, for parity or ECC. In that case, data capacity is shown in 9-bit symbols.
- b. Device densities shown vary from the popular 256 Mb to the newer 1Gb.
- c. A 72-bit port is referred to as having a port count of 2. For example, a single 72-bit interface making use of the DDRA port and some signals from the DDRB port is listed here as two ports.
- d. System data capacities shown vary from 32 MB to 8 GB, which are typical capacities. The XLS Processors support up to 64 GB.
- e. Each channel can support up to two DIMMs.

11.3.4.1 DDR2 SDRAM Example Circuit

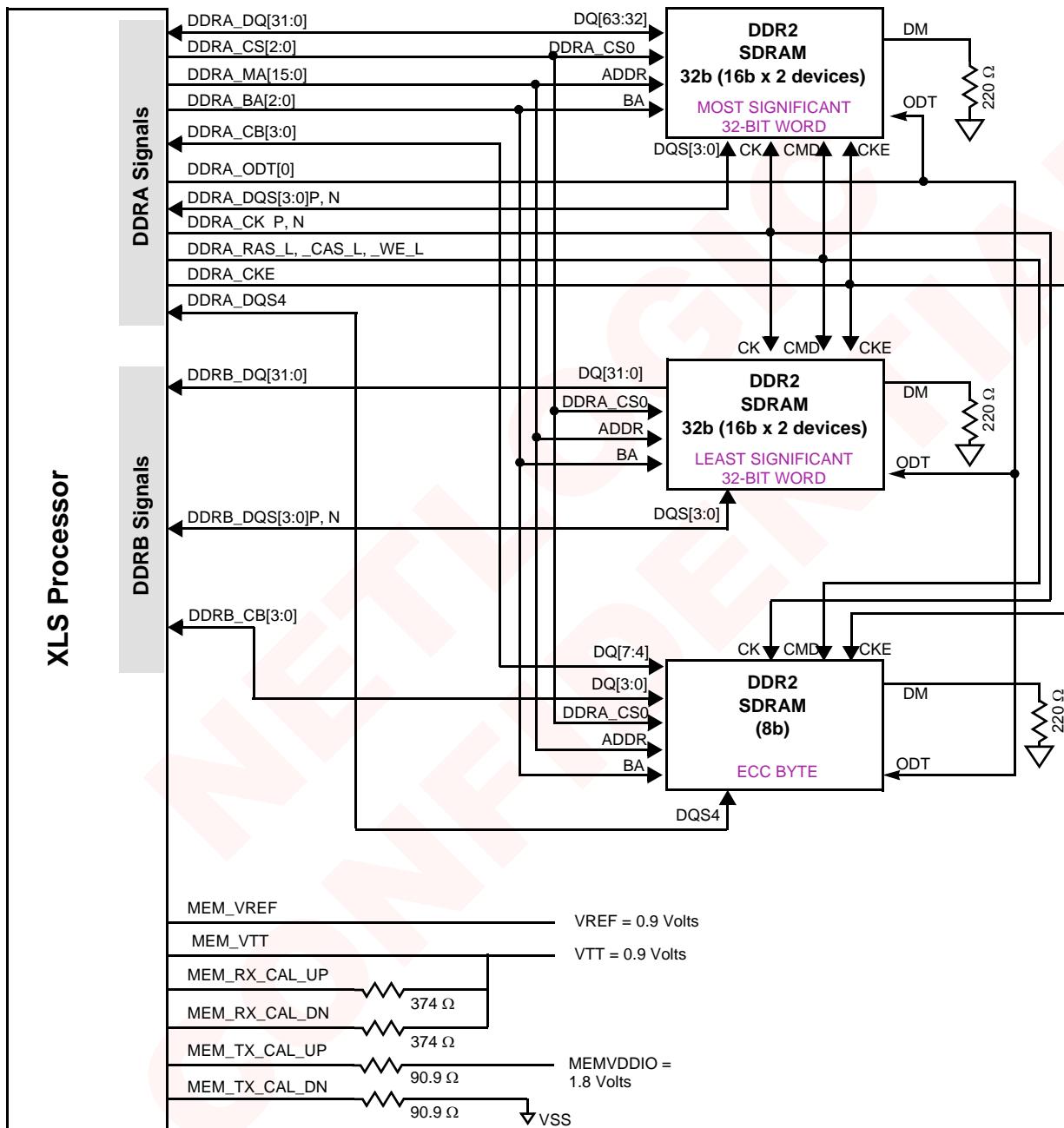
Figure 11-4. shows a typical interconnection example of a single XLS Processor DRAM port connected to four external 1 Gbit DDR2 SDRAMs in a 36-bit configuration.

Figure 11-4. Example Connections for 1 x 32b DDR2 SDRAM Interface



A typical example of a single XLS Processor DRAM channel (using two DRAM ports) connected to DDR2 SDRAMs in a 72-bit configuration is shown in [Figure 11-5](#).

Figure 11-5. Example XLS2xx Connections for 1 x 72b Single-Rank DDR2 SDRAM Interface



11.4 Programming Model

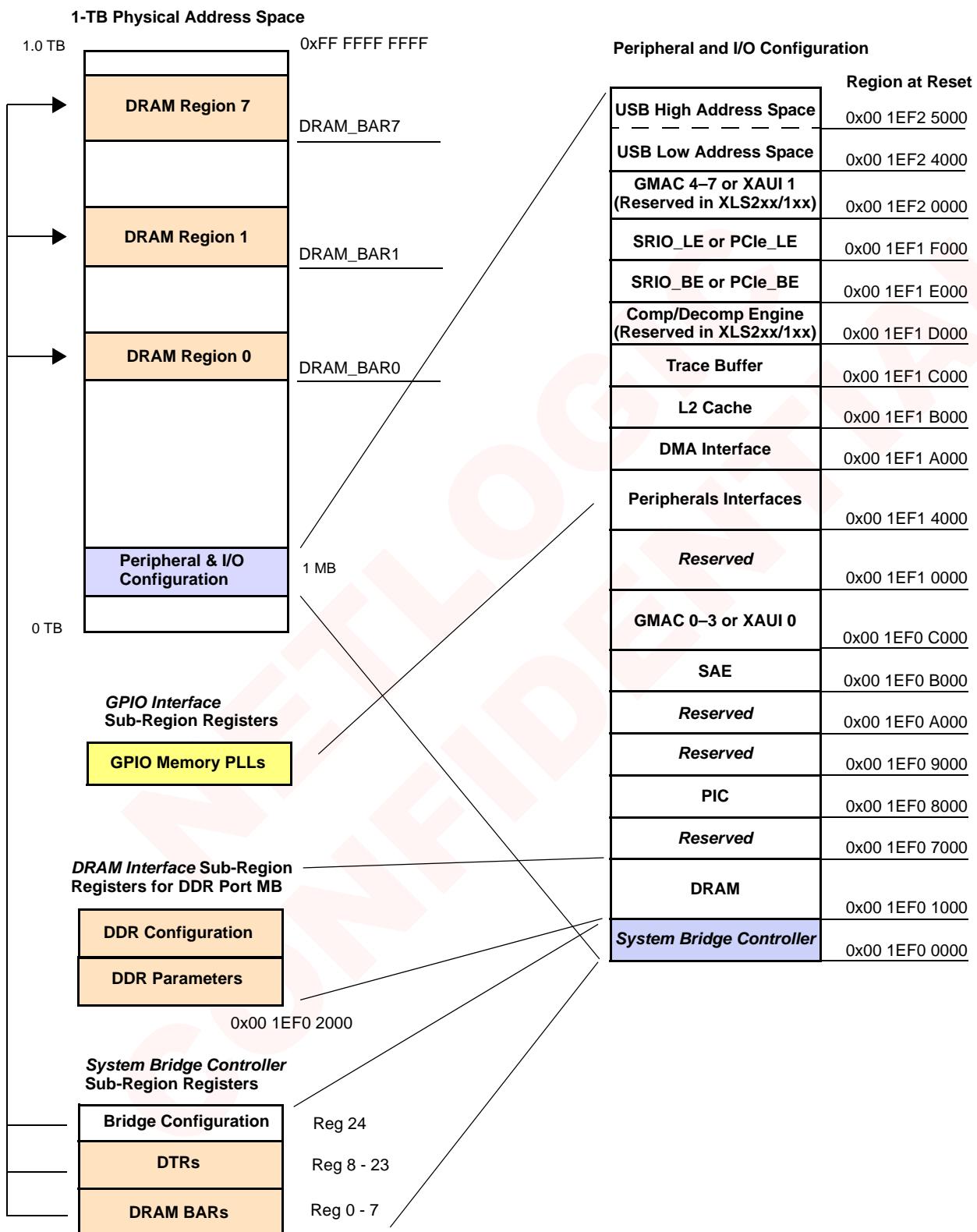
11.4.1 Memory Controller Configuration

The user must program each memory controller and its corresponding memory bridge shown in [Figure 11-1](#) to support the memory chip type, configuration, and data width that has been chosen for the external system memory.

The system memory map in [Figure 11-6](#) shows the three register spaces that govern DRAM operation. The Memory Bridge registers are covered in [Chapter 8, “Memory and I/O Access”](#). The DRAM clock rates are set in the GPIO register space as described in [Chapter 23, “GPIO and the Peripherals Interface”](#). [Table 11-6](#) below gives the starting addresses of the four register sets covered in this Chapter for the four DDR2 channels. The programming parameter registers described in the following sections all fall within these four sub-regions.

Table 11-6. Memory Controller Register Region Offsets

Memory Controller Function	Controller Name	Region Offset	Region Offset Address at Reset
DDR2 SDRAM_A Channel, Port MA. For a single channel A or the master channel A in channel-pair AB.	XLS_IO_DEV_DDR_CHNA	0x0 1000	0x00 1EF0 1000
DDR2 SDRAM_B Channel, Port MB. For a single channel B or the slave channel B in channel-pair AB. (Note: This DRAM Channel is supported by the XLS6xx, XLS4xx-Lite and XLS4xx devices.)	XLS_IO_DEV_DDR_CHNB	0x0 2000	0x00 1EF0 2000
DDR2 DRAM_C Channel, Port MC. For a single channel C or the master channel C in channel-pair CD. (Note: This SDRAM Channel is supported by XLS6xx devices only.)	XLS_IO_DEV_DDR_CHNC	0x0 3000	0x00 1EF0 3000
DDR2 DRAM_D Channel, Port MD. For a single channel D or the slave channel D in channel-pair CD. (Note: This SDRAM Channel is supported by XLS6xx devices only.)	XLS_IO_DEV_DDR_CHND	0x0 4000	0x00 1EF0 4000

Figure 11-6. Example of Three DRAM Port MB Regions in the Address Space

11.4.2 Memory Controller Initialization Sequence

This section describes the XLS DRAM memory controller initialization sequence as a guide to coders.

The initialization sequence consists of six states. State 1 is entered during a power-on reset, or other XLS chip-wide reset, regardless of prior state. Subsequently, states 2 and 3 follow one another in sequence. State 4 is entered when the user writes to the RST_START bit within the DDR_CFG_REG_RESET_TIMERS register, regardless of prior state. State 5 and 6 follow in sequence from state 4.

State 1:

Entry into this state is triggered by assertion of the reset signal going to each of the four memory controllers, by logic external to the DRAM memory controllers. The logic could be internal or external to the XLS processor. One example is the XLS processor chip-wide reset sequence, which triggers the DRAM memory controllers to enter this state.

During this state, the memory controllers are not supplied with a clock, so the output pins could have any value. DDR*_CK*P/N pins will not be toggling at this time.

State 2:

Upon assertion of the XLS chip PowerGood input pin, and locking of the XLS DRAM PLL's, this state is entered from state 1. The memory controllers are now being clocked, but are still in reset. All controller configuration and status registers are reset to their default values. Writes to any of these registers will be ignored during this phase. Both the DLL calibration logic and the read data capture logic are disabled. All internal state of each controller is reset, i.e. all information about any previously pending transactions is lost.

During this time:

- the DDR*_CK*P/N pins will be toggling
- the DDR*_CKE pins will be 0
- the DDR*_ODT pins will be 0
- the DDR*_CS0_L pins will be 0
- the DDR*_CS1_L pins will be 0
- the DDR*_CS2_L pins will be 1
- the DDR*_CS3_L pins will be 1
- the DDR*_DQ* and DDR*_DQS*P/N pins are not driven by the XLS
- the DDR*_MA* and DDR*_BA* address pins are not driven by the XLS

State 3:

Upon de-assertion of the XLS-internal reset signal going to the memory controllers, this state is entered from state 2. At this point the memory controllers are being clocked, they have been reset, and they are awaiting activation. It is during this time that the user may program the memory controller registers. If not programmed, the registers maintain their reset values. Entry into this state does trigger a DLL calibration. Once it completes, the CYC_MEAS_CNT, NXT_DLY_CNT, and DLY_CNT registers are updated.

During this time, the pins will maintain the same values as in State 2, with the exception of the DDR*_CS_L pins. These pins will be valid 0 or 1. Their actual value will depend on the values programmed by the user into DDR_ADDR_PARAMS_1 and DDR_ADDR_PARAMS_2.

State 4:

Whenever the user sets the RST_START bit within the DDR_CFG_REG_RESET_TIMERS register, the corresponding memory controllers enter this state, regardless of prior state. The memory controllers will stay in this state for a length of time determined by 16 times the number of clock cycles programmed into the RST_LEN field of the DDR_CFG_REG_RESET_TIMERS

register. Because it may not be assumed that the prior state was state 3, during this state a reset signal internal to the memory controllers is asserted. This signal has many of the same effects as states (1), (2), and (3), with a few exceptions: other XLS chip logic is not simultaneously reset, no DRAM memory controller configuration registers are reset, DRAM port address and command pins are fully enabled, and no DLL calibration is triggered.

While in this state, the observed pin behavior will be identical to that observed during state (3).

State 5: Once the time period specified in state (4) expires, the reset signal internal to the memory controllers is de-asserted, and this state is entered from state (4). The DRAM chip initialization sequence is issued. The commands come from registers DDR_CFG_REG_RESET_CMD0 to DDR_CFG_REG_RESET_CMD6. The number of commands issued is specified by the RST_SEQ_LEN field of the DDR_CFG_REG_RESET_TIMERS register.

While in this state:

- the DDR*_CK*P/N pins will be toggling
- the DDR*_CKE pins will transition from 0 to 1
- the DDR*_CS*_L pins will be toggling, based on the commands being issued
- the DDR*_DQ* and DDR*_DQS*P/N pins are now enabled, allowing WRITE commands to be issued when needed
- the read data capture logic is enabled, enabling READ commands to complete correctly.
- From this point forward, user writes to the DDR_PARAMS_1 to DDR_PARAMS_7 registers are staged - that is, the values written to these addresses are stored in temporary buffers, but do not update the actual configuration registers except under specific circumstances (issuance of a user-command sequence). The same is true of the NXT_DLY_CNT registers - user writes to these addresses are only propagated to the corresponding DLY_CNT registers when the next DLL calibration occurs.

State 6: Once the DDR2 device initialization sequence has completed, the memory controllers are fully operational. This state will be exited only if the reset signal to the memory controllers is asserted (in which case state 1 is entered), or if the RST_START bit within the DDR_CFG_REG_RESET_TIMERS register is written (in which case state 4 is entered).

During this time:

- the DDR*_CK*P/N pins will be toggling
- the DDR*_CKE pins will be logic 1
- the DDR*_CS*_L pins will be 0 when no transactions are in progress. When transactions are in progress, the selected rank will have its CS_L = 0, and the other ranks will have CS_L=1
- the DDR*_DQ* pins will toggle when transactions are in progress, and be high-impedance otherwise
- the DDR*_DQS* pins will toggle when transactions are in progress, and be high-impedance otherwise
- the DDR*_ODT* pins will toggle when transactions are in progress, and be 0 otherwise
- the DDR*_BA* and DDR*_MA* address pins will toggle as needed
- the DDR*_RAS_L, DDR*_CAS_L, and DDR*_WE_L command pins will toggle as needed

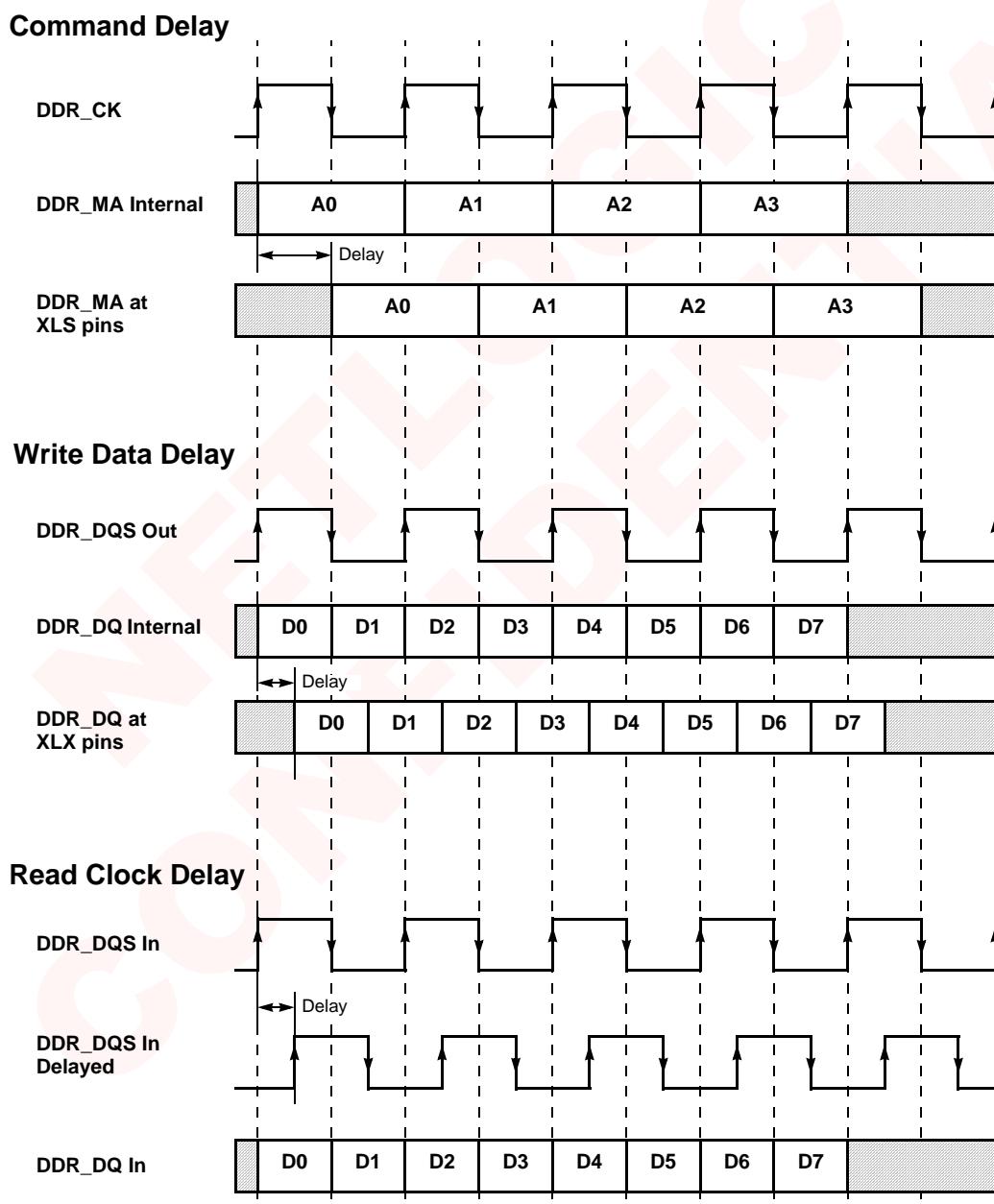
To summarize, recommended operating procedure on power-up would be to allow states 1, 2, and 3 to complete, program all desired DRAM memory controller and bridge configuration registers, manually trigger a DLL re-calibration, and then trigger states 4, 5 and 6. Recommended operating procedure for a “warm” reset would be to program all desired DRAM memory controller and bridge configuration registers, manually trigger a DLL re-calibration, and then trigger states 4, 5, and 6.

11.4.3 Understanding The DLLs

11.4.3.1 DLL Configurations

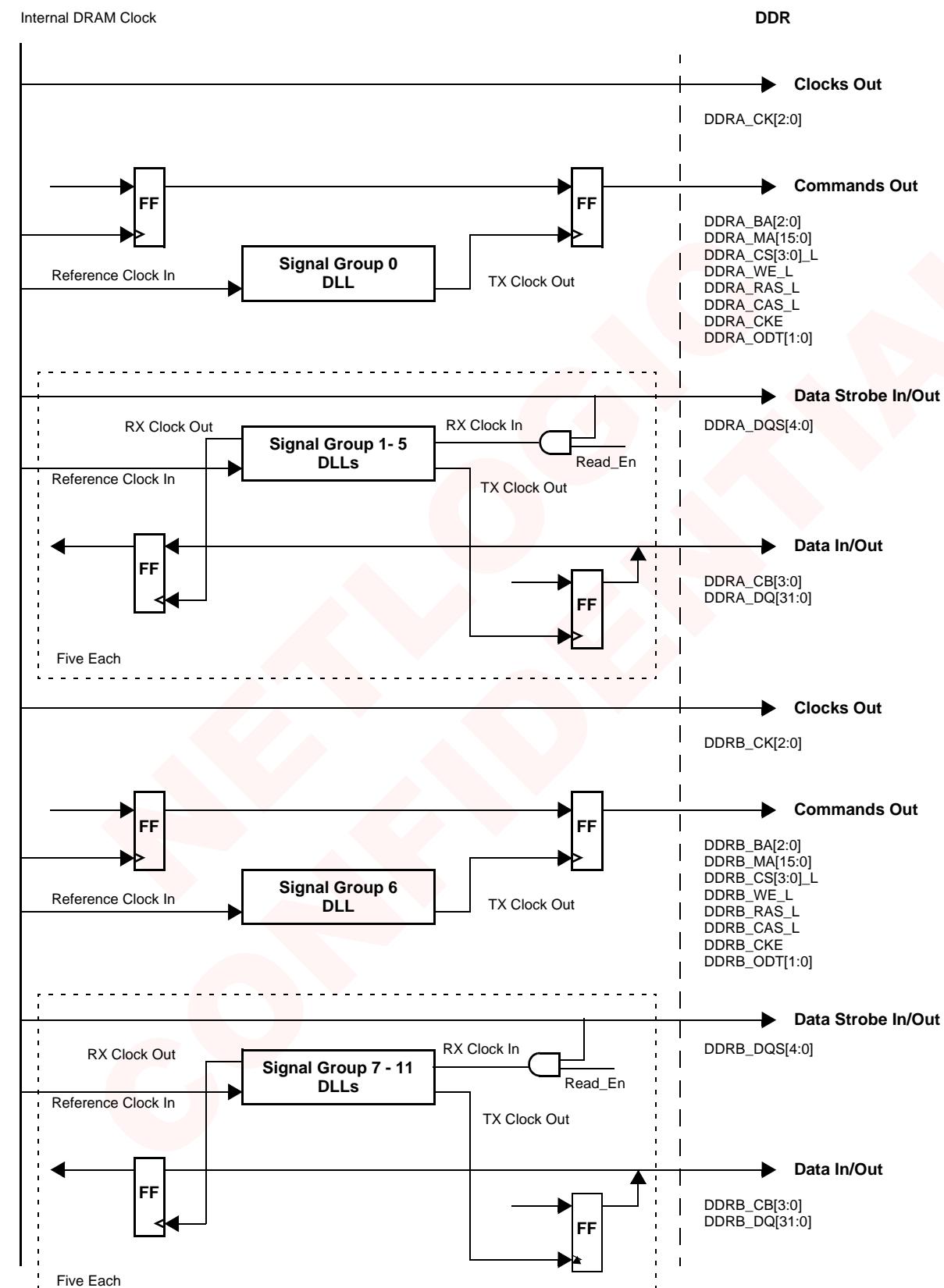
The DDR2 memory protocols require various signals in [Table 11-4](#) to be phase-shifted by 90 or 180 degrees. For example, command signals are shifted by 180 degrees to be centered around the rising edge of a clock. Double-Data-Rate (DDR) Read and Write data signals are shifted by 90 degrees to be centered around their corresponding rising or falling DQS edge. These three examples are illustrated in [Figure 11-7](#).

Figure 11-7. Clock, Data and Command Shifting with Signal Group DLLs



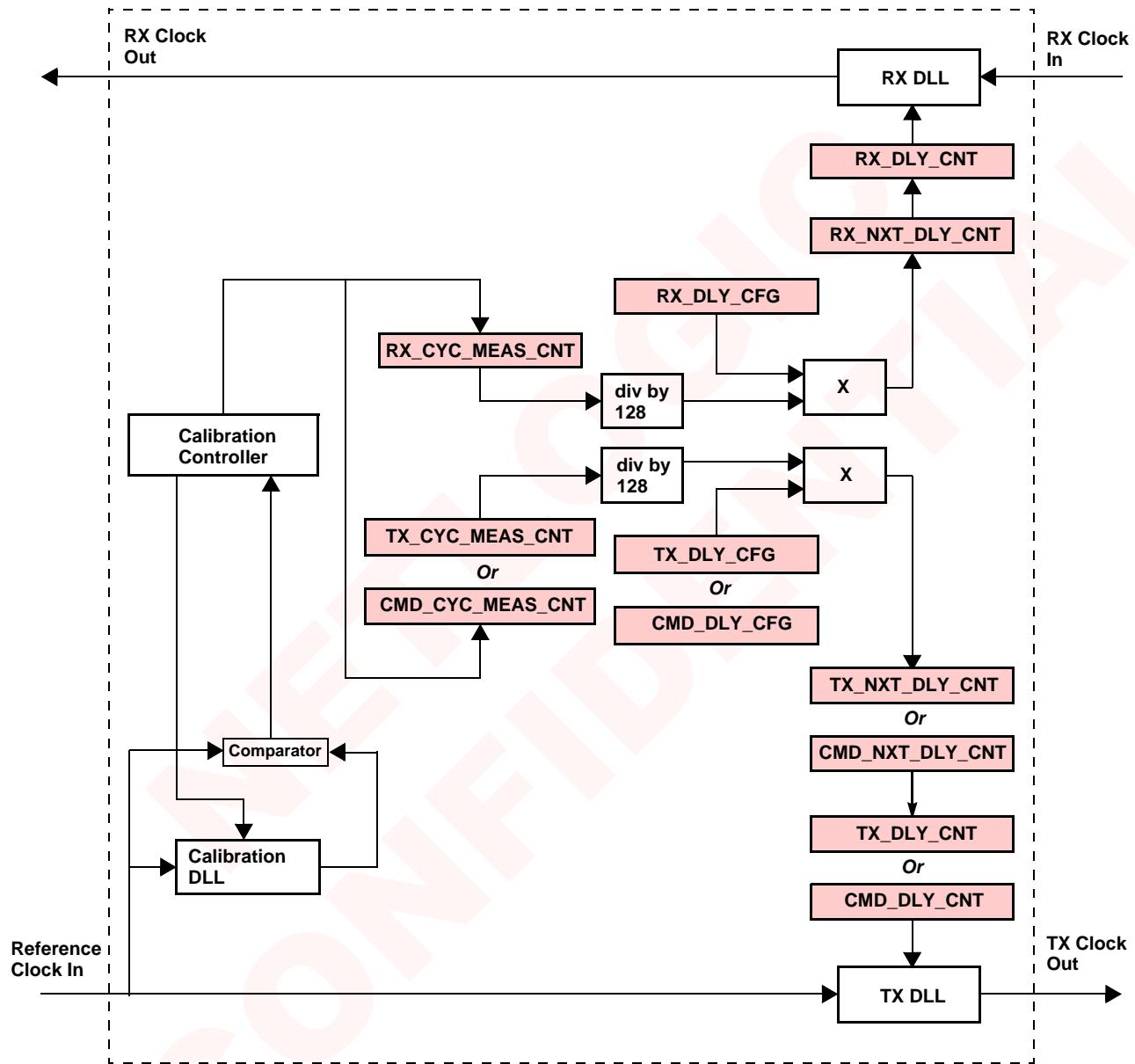
The DRAM memory controllers implement this phase-shifting using programmable delay lines for a Delay Lock Loop (DLL)-like function. There are thirty-four of these DLLs per controller-pair (AB or CD), organized into twelve DLL signal groups (0-11) per channel-pair as shown in [Figure 11-8](#) for the Channel-Pair AB. The corresponding signals for each group are given. Note that the ten data groups are eight for each data byte and two for the check bits.

NETLOGIC
CONFIDENTIAL

Figure 11-8. Signal Group DLLs for the Channel-Pair AB DRAM Interface

Each DLL signal group consists of a calibration DLL, a transmit DLL and an optional receive DLL with a total of four or eight control registers per DLL as shown in [Figure 11-9](#).

Figure 11-9. Functional Blocks and Registers in Each DLL for a Signal Group



[Table 11-7](#) shows the correspondence between the DRAM data and command signals, the DLL clocks and the DLL control registers for all twelve signal groups and both channel-pairs.

Table 11-7. Configuration of DDR Signals with DLLs and Their Registers

Signals	Related Clock Signal	DLL Calibration Control Registers	DLL Transmit Control Registers	DLL Receive Control Registers
Ports MA and MC				
Signal Group 0		DDR_CMD_CYC_MEAS_CNT	DDR_CMD_DLY_CFG DDR_CMD_NXT_DLY_CNT DDR_CMD_DLY_CNT	-
DDR(A/C)_ODT[1:0] DDR(A/C)_MA[15:0] DDR(A/C)_BA[2:0] DDR(A/C)_CS[3:0]_L DDR(A/C)_RAS_L DDR(A/C)_CAS_L DDR(A/C)_WE_L DDR(A/C)_CKE	DDR(A/C)_CK[2:0]P, N			
Signal Group 1		DDR_DQS0_RX_CYC_MEAS_CNT DDR_DQS0_TX_CYC_MEAS_CNT These two should always be the same value	DDR_DQS0_TX_DLY_CFG DDR_DQS0_TX_NXT_DLY_CNT DDR_DQS0_TX_DLY_CNT	DDR_DQS0_RX_DLY_CFG DDR_DQS0_RX_NXT_DLY_CNT DDR_DQS0_RX_DLY_CNT
Signal Group 2		DDR_DQS1_RX_CYC_MEAS_CNT DDR_DQS1_TX_CYC_MEAS_CNT These two should always be the same value	DDR_DQS1_TX_DLY_CFG DDR_DQS1_TX_NXT_DLY_CNT DDR_DQS1_TX_DLY_CNT	DDR_DQS1_RX_DLY_CFG DDR_DQS1_RX_NXT_DLY_CNT DDR_DQS1_RX_DLY_CNT
Signal Group 3		DDR_DQS2_RX_CYC_MEAS_CNT DDR_DQS2_TX_CYC_MEAS_CNT These two should always be the same value	DDR_DQS2_TX_DLY_CFG DDR_DQS2_TX_NXT_DLY_CNT DDR_DQS2_TX_DLY_CNT	DDR_DQS2_RX_DLY_CFG DDR_DQS2_RX_NXT_DLY_CNT DDR_DQS2_RX_DLY_CNT
Signal Group 4		DDR_DQS3_RX_CYC_MEAS_CNT DDR_DQS3_TX_CYC_MEAS_CNT These two should always be the same value	DDR_DQS3_TX_DLY_CFG DDR_DQS3_TX_NXT_DLY_CNT DDR_DQS3_TX_DLY_CNT	DDR_DQS3_RX_DLY_CFG DDR_DQS3_RX_NXT_DLY_CNT DDR_DQS3_RX_DLY_CNT
Signal Group 5		DDR_DQS8_RX_CYC_MEAS_CNT DDR_DQS8_TX_CYC_MEAS_CNT These two should always be the same value	DDR_DQS8_TX_DLY_CFG DDR_DQS8_TX_NXT_DLY_CNT DDR_DQS8_TX_DLY_CNT	DDR_DQS8_RX_DLY_CFG DDR_DQS8_RX_NXT_DLY_CNT DDR_DQS8_RX_DLY_CNT

Table 11-7. Configuration of DDR Signals with DLLs and Their Registers (continued)

Signals	Related Clock Signal	DLL Calibration Control Registers	DLL Transmit Control Registers	DLL Receive Control Registers
Ports MB and MD				
Signal Group 6	DDR(B/D)_CK[2:0]P, N	DDR_SLV_CMD_CYC_MEAS_CNT	DDR_SLV_CMD_DLY_CFG DDR_SLV_CMD_NXT_DLY_CNT DDR_SLV_CMD_DLY_CNT	-
DDR(B/D)_ODT[1:0] DDR(B/D)_MA[15:0] DDR(B/D)_BA[2:0] DDR(B/D)_CS[3:0]_L DDR(B/D)_RAS_L DDR(B/D)_CAS_L DDR(B/D)_WE_L DDR(B/D)_CKE				
Signal Group 7	DDR(B/D)_DQ[7:0]	DDR_DQS4_RX_CYC_MEAS_CNT DDR_DQS4_TX_CYC_MEAS_CNT These two should always be the same value	DDR_DQS4_TX_DLY_CFG DDR_DQS4_TX_NXT_DLY_CNT DDR_DQS4_TX_DLY_CNT	DDR_DQS4_RX_DLY_CFG DDR_DQS4_RX_NXT_DLY_CNT DDR_DQS4_RX_DLY_CNT
Signal Group 8	DDR(B/D)_DQ[15:8]	DDR_DQS5_RX_CYC_MEAS_CNT DDR_DQS5_TX_CYC_MEAS_CNT These two should always be the same value	DDR_DQS5_TX_DLY_CFG DDR_DQS5_TX_NXT_DLY_CNT DDR_DQS5_TX_DLY_CNT	DDR_DQS5_RX_DLY_CFG DDR_DQS5_RX_NXT_DLY_CNT DDR_DQS5_RX_DLY_CNT
Signal Group 9	DDR(B/D)_DQ[23:16]	DDR_DQS6_RX_CYC_MEAS_CNT DDR_DQS6_TX_CYC_MEAS_CNT These two should always be the same value	DDR_DQS6_TX_DLY_CFG DDR_DQS6_TX_NXT_DLY_CNT DDR_DQS6_TX_DLY_CNT	DDR_DQS6_RX_DLY_CFG DDR_DQS6_RX_NXT_DLY_CNT DDR_DQS6_RX_DLY_CNT
Signal Group 10	DDR(B/D)_DQ[31:24]	DDR_DQS7_RX_CYC_MEAS_CNT DDR_DQS7_TX_CYC_MEAS_CNT These two should always be the same value	DDR_DQS7_TX_DLY_CFG DDR_DQS7_TX_NXT_DLY_CNT DDR_DQS7_TX_DLY_CNT	DDR_DQS7_RX_DLY_CFG DDR_DQS7_RX_NXT_DLY_CNT DDR_DQS7_RX_DLY_CNT
Signal Group 11	DDR(B/D)_CB[3:0]	DDR_DQS9_RX_CYC_MEAS_CNT DDR_DQS9_TX_CYC_MEAS_CNT These two should always be the same value	DDR_DQS9_TX_DLY_CFG DDR_DQS9_TX_NXT_DLY_CNT DDR_DQS9_TX_DLY_CNT	In Single-Channel mode: DDR_DQS9_RX_DLY_CFG DDR_DQS9_RX_NXT_DLY_CNT DDR_DQS9_RX_DLY_CNT In Channel-Pair mode: DDR_DQS8_RX_DLY_CFG DDR_DQS8_RX_NXT_DLY_CNT DDR_DQS8_RX_DLY_CNT

11.4.3.2 Calibration of the DLLs

To account for voltage and temperature variations over time, the DLLs will re-calibrate periodically. This re-calibration is triggered either automatically or under software control. In the first step of this process, the calibration DLLs in each DLL group are used to measure the length of one clock cycle (or half a clock cycle) in terms of the number of DLL delay elements.

Once the measurement is complete, the result is stored in the read-only CYC_MEAS_CNT register within each DLL group. See [Figure 11-9](#) again. The stored value consists of three fields. **ROW_CYC_MEAS_CNT** is a 5-bit row delay value, **COL_CYC_MEAS_CNT** is a 3-bit column delay value, and **FINE_CYC_MEAS_CNT** is a 3-bit fine delay stage value. The number of delay elements corresponding to the length of the clock cycle (or half-clock cycle) is given by the following equation:

$$\text{Number Of Delay Elements} = 8(\text{ROW_CYC_MEAS_CNT}) + \text{COL_CYC_MEAS_CNT} + (\text{FINE_CYC_MEAS_CNT})/8$$

This corresponds to the physical organization of the DLLs, which consist of 256 standard delay elements organized in 32 rows and 8 columns, as well as 12 fine-delay elements that can be used for finer-granularity tuning. For example, a CYC_MEAS_CNT value of 0x11A represents 35 and 2/8 delay element delays. The time delay corresponding to one DLL delay element varies with temperature, voltage, and technology, but it is typically in the range of 30 to 90 picoseconds.

Once the clock cycle has been measured, if self-calibration mode is enabled by the appropriate bit (bits RX_NXT_DLY_CNT_UPDATE_EN, TX_NXT_DLY_CNT_UPDATE_EN, or CMD_NXT_DLY_CNT_UPDATE_EN in the DDR_CLK_CAL_PARAMS register), a new value is written into the NXT_DLY_CNT register of each DLL group. The new value is given by the following equation.

$$\text{NXT_DLY_CNT} = (\text{CYC_MEAS_CNT})(\text{DLY_CFG})/128$$

This value is encoded in the same format as in the CYC_MEAS_CNT register. Finally, some time later when the next REFRESH command is issued, the value in the NXT_DLY_CNT register is transferred to the corresponding current DLY_CNT register in the same DLL group. The DLY_CNT register feeds directly to the control inputs of the RX and TX DLLs within that group, so that within a few cycles the DLL phase-shift delays reflect the new calibration values.

11.4.3.3 Programming the DLL Control Registers

The **DDR_GLB_PARAMS**, **DDR_CLK_CAL_PARAMS** and **DDR_CLK_CAL_REQ** configuration registers contain most of the parameters that control the DLL calibration process, such as the duration of the calibration process, how frequently a calibration occurs, whether a full or half cycle is measured, etc. See the descriptions of these registers for more details.

The DLY_CFG register is intended to function as a scale factor for CYC_MEAS_CNT. For a 90 degree phase shift, the DLY_CFG register should be programmed with the value 0x20, and for a 180 degree phase shift, it should be programmed with the value 0x40. See [Figure 11-7](#) again. Although these are the default reset values of these registers, the user may want to modify them for the following reasons.

- Although 90 degrees and 180 degrees are the nominal phase-shift values for the data and command bits respectively, these phase-shift values assume ideal behavior on the part of the DRAM devices. To account for asymmetries and permitted pulse-width variations specified for most DDR2 devices, a slightly lower phase shift value is preferable, particularly at higher operating frequencies.
- Non-ideal electrical behavior can be compensated for by modifying the DLL phase shift amounts. For example, the DDR2 commands are ideally centered around the rising edge of clock. A 180 degree DLL phase shift will achieve this. However, depending on the

number of DRAM chips or DIMMs attached to a channel, the DRAM command output pins may experience considerably heavier electrical loading than the DRAM clock output pins, thus requiring more time to transition. This may be compensated for by reducing the command DLL phase shift amount, thereby allowing extra time for the command pins to meet the required setup time to the rising edge of the clock.

Note that the DLLs have a minimum phase-shift value of three delay elements, hence even if a DLY_CFG register is programmed with the value zero, the relevant signals will be shifted 90 to 270 picoseconds with respect to each other.

Although the NXT_DLY_CNT registers are also writable, it is strongly suggested that they never be programmed. The recommended mode of operation is to programming only the following registers, using the settings shown below.

- Set the RX_NXT_DLY_CNT_UPDATE_EN, TX_NXT_DLY_CNT_UPDATE_EN, and CMD_NXT_DLY_CNT_UPDATE_EN bits in the DDR_CLK_CAL_PARAMS register, thus enabling updating of NXT_DLY_CNT registers during calibration.
- Program DDR_CLK_CAL_PARAMS.CLK_MAX_DLY_STG_MOD with the value 10, i.e all bits of the CYC_MEAS_CNT registers are updated during calibration.
- Program DDR_CLK_CAL_PARAMS.CLK_FN_CAL with the value 0, i.e. DLL fine delay range is 0/8 to 7/8 of one main delay element.
- Program DDR_CLK_CAL_PARAMS.CLK_MSR_HLF with the value 0, i.e. a full clock cycle is measured during calibration.
- Program DDR_CLK_CAL_PARAMS.CLK_TRG_OFF_REF by setting it, thus enabling automatic triggering of calibrations 128 cycles before a REFRESH command is issued.
- Program DDR_DQS[0-9]_TX_DLY_CFG with an appropriate value
- Program DDR_DQS[0-9]_RX_DLY_CFG with an appropriate value.
- Program DDR_CMD_DLY_CFG and DDR_SLV_CMD_DLY_CFG with an appropriate value.

The registers should be programmed once at the beginning, prior to activating the controller. Once the controller is activated, prior to reading or writing any of the 88 DLL control registers, automatic calibrations should be disabled by setting

[DDR_CLK_CAL_PARAMS.CLK_TRG_OFF_REF](#) to 0. Failure to do so may result in disruption of an in-process automatic calibration, temporarily causing incorrect operation due to garbled DLL settings. Once the register Read or Write is complete, automatic triggering of calibrations should be enabled once again.

11.4.4

Understanding the DRAM Memory Bridge/Memory Controller Interaction

Accurately programming the DRAM Memory Bridge is essential to correct operation of the DRAM memory system. The Bridge programming determines how memory accesses are distributed to the Memory Controllers. For example, each memory channel or channel-pair may be allocated its own exclusive portion of the physical address space. Alternatively, requests to a particular address region may be interleaved between 2 or 4 memory channels, at programmable granularity levels.

The Bridge also performs address mapping. Based on the configuration and interleaving settings programmed in the Bridge Configuration Registers, DRAM BARs and DTRs, the 40-bit physical address of a transaction is translated into a 29-bit address and sent to the appropriate DRAM controller. For details on programming the Bridge registers, see [Chapter 8, "Memory and I/O Access"](#)

11.4.5 Understanding the Thermal Control Registers

The two DRAM memory controller channel-pairs each provide four programmable thermal control registers. These registers allow monitoring and, if necessary, manually overriding XLS DRAM port pad driver strengths and pad receiver impedances.

This adds extra flexibility when operating under marginal electrical conditions. Like other DRAM memory controller registers, the thermal control registers for each channel-pair must be accessed through their own address offset (0x1000 for channel-pair A/B, 0x3000 for channel-pair C/D). Note, however, that unlike other registers, these registers are NOT completely independent. In other words, the thermal control registers from one channel-pair DO influence signals in the other channel-pair. For this reason, it is recommended that the following restrictions be observed while programming these registers:

1. The value programmed into the writable bits of the channel-pair AB DDR_TX0_THERM_CTRL register should equal the value programmed into the writable bits of the channel-pair CD DDR_RX0_THERM_CTRL register.
2. The value programmed into the writable bits of the channel-pair AB DDR_RX1_THERM_CTRL register should equal the value programmed into the writable bits of the channel-pair CD DDR_RX1_THERM_CTRL register.
3. The value programmed into the writable bits of the channel-pair AB DDR_RX0_THERM_CTRL register should equal the value programmed into the writable bits of the channel-pair CD DDR_RX0_THERM_CTRL register.
4. The value programmed into the writable bits of the channel-pair AB DDR_RX1_THERM_CTRL register should equal the value programmed into the writable bits of the channel-pair CD DDR_RX1_THERM_CTRL register.

There are two fields in these registers that must be programmed.

First, the TERM_SEL field of the DDR_RX0_THERM_CTRL register controls the termination resistance applied to the XLS DRAM data bus input pads.

Second, the DATA_DRV_IMP and ADDR_DRV_IMP bits in the DDR_TX0_THERM_CTRL register must be programmed to the desired driver impedance strengths.

Under normal operation, the XLS processor automatically calibrates the pad driver strengths to maintain a constant drive strength and impedance in the face of transient temperature and voltage variations. Therefore, it is recommended that the remaining fields in the thermal control registers be left at their default values. However, if required, a program may disable the calibration process and manually specify a driver strength. This may be accomplished by clearing the EN_CNT field of these registers. The driver strength/input impedance can be specified as an integer value between 0 and 63 in the PRESET_VALUE field of these registers.

If the PRESET_P field and/or the PRESET_N field are set in these registers, then the programmed value is applied to the pull-up drive strength and the pull-down drive strength, respectively. The read-only PCODE field and NCODE field can be used to confirm that the desired values have been set. Again, the recommended mode of operation is to enable the automatic drive-strength calibration.

See register descriptions for DDR2_TX0_THERM_CTRL, DDR2_TX1_THERM_CTRL, DDR2_RX0_THERM_CTRL, and DDR2_RX1_THERM_CTRL register for details.

11.4.6 Understanding the User Command Sequence

Each XLS DRAM memory controller allows the user to issue a command sequence on the DRAM command bus at any time. This feature may be useful primarily for testing or debug purposes. The instruction sequence may be up to four commands long. The commands to be issued should be programmed into the RESET_CMD fields (bits 5:0 and bits 21:16) of the

[DDR_CFG_REG_RESET_CMD0](#) and [DDR_CFG_REG_RESET_CMD1](#) registers. The RESET_CMD_CNT fields of these registers (bits 15:6 and 31:22) specify the number of cycles each command should be issued. Note that these counts are repetition counts, i.e. a value of zero means the command is issued for one cycle, a value of one means the command is issued for two cycles, etc. The command sequence is triggered when the user Writes to the [CMD_SEQ_RQD](#) field (bit 7) of the [DDR_CFG_REG_RESET_TIMERS](#) register. Once the command sequence has been issued, this bit will be cleared.

The commands and their encodings are specified in [Table 11-8](#). Note that these same encodings are also used when programming the DRAM chip initialization command sequence in registers DDR_CFG_REG_RESET_CMD[0-6].

Table 11-8. DRAM Memory Controller Commands

Encoding	Internal Command Name	DRAM Bus Command Issued	Notes
0x01	ENABLE_DLL	MRS	Clears all bits of the Extended Mode Register
0x02	SELF_REF_ENT	REFRESH	Enter self-refresh mode by un-asserting CKE signal
0x03	DEFAULT_OCD	MRS	Set bits [9:7] of the Extended Mode Register. Uses DDR_CFG_REG_EMRS1 register contents, but forces bits [9:7] to 111.
0x0E	PWR_DN_ENT	NOP	Enter power-down mode by un-asserting CKE signal
0x20	MRS	MRS	Write to Mode Register. Uses DDR_CFG_REG_MRS register contents
0x21	EMRS1	MRS	Write to Extended Mode Register. Uses DDR_CFG_REG_EMRS1 register contents.
0x22	EMRS2	MRS	Write to Extended Mode Register 2. Uses DDR_CFG_REG_EMRS2 register contents.
0x23	EMRS3	MRS	Write to Extended Mode Register 3. Uses DDR_CFG_REG_EMRS3 register contents
0x24	PRECHARGE	PRECHARGE	
0x25	PRECHARGE_ALL	PRECHARGE_ALL	
0x26	ACTIVE	ACTIVE	
0x27	RESET_DLL	MRS	Uses DDR_CFG_REG_RDMRS register contents to set bit [8] of Mode Register.
0x28	WRITE	WRITE	
0x29	WRITE_PRE	WRITE_PRE	
0x2A	READ	READ	
0x2B	READ_PRE	READ_PRE	
0x2C	WRITE_DM	WRITE	Data masking not supported, hence converted to Write
0x2D	WRITE_PRE_DM	WRITE_PRE	Data masking not supported, hence converted to Write Precharge
0x2E	NOP	NOP	
0x2F	PWR_DN_EX	NOP	Exit power-down mode by asserting CKE signal
0x30	DESELECT	DESELECT	
0x32	AUTO_REF	AUTO_REF	
0x3E	SELF_REF_EX	NOP	Exit self-refresh mode by asserting CKE signal

11.4.7 Understanding Read Data Capture

Capturing the Read data returned from the DRAM memory devices involves three steps.

1. Gating the DQS strobes
2. Phase shifting the DQS strobes to capture the data
3. Moving the Read data back into the controller clock domain

These are explained in sequence in this section.

11.4.7.1 Gating the DQS Strobes

To prevent erroneous capturing of Read data, incoming Read DQS strobes are logically AND'ed with an internally-generated READ_EN signal prior to being fed to the phase-shift DLL's as shown in [Figure 11-8](#). Since the READ_EN signal is asserted only when Read data is expected, the Read-data capture flip-flops are not affected by the toggling of the DQS strobes during Write transactions or by noise-related transitions when the strobes are undriven in the high-impedance state.

Each DQS strobe has a corresponding READ_EN signal; there is one per byte-lane. Each READ_EN signal is a pulse whose timing and duration is determined by the value that has been programmed into the following register fields.

```
DDR_PARAMS_3.TRL
DDR_DQS[0-9]_RX_DL_Y_CFG.GEN_HALF_EARLY
DDR_DQS[0-9]_RX_DL_Y_CFG.GEN_CYC_NOMINAL
DDR_DQS[0-9]_RX_DL_Y_CFG.GEN_HALF_LATE
DDR_DQS[0-9]_RX_DL_Y_CFG.GEN_CYC_LATE
```

For correct operation, the generated READ_EN pulse must satisfy all of the following requirements.

1. It must transition from low to high during the Read preamble of the DQS strobe
2. It must be high during the first rising edge of the Read DQS strobe
3. It must remain high during the first falling edge of the Read DQS strobe
4. It must have transitioned Low by the time the DQS strobe post-amble is over

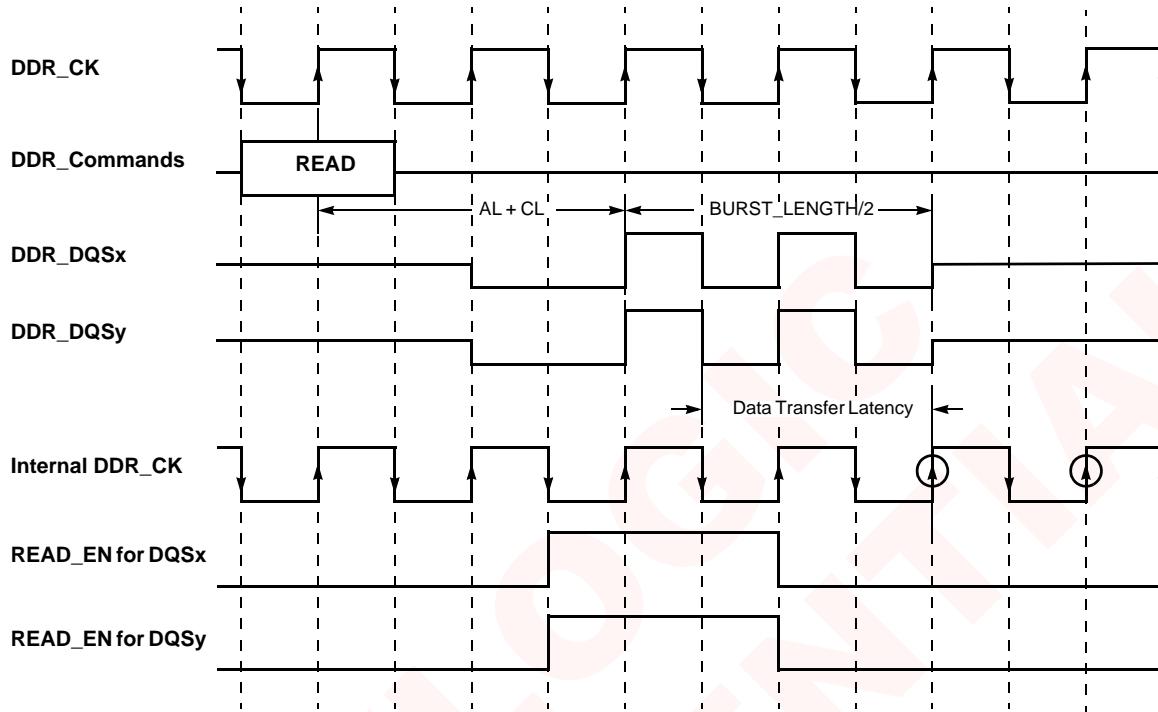
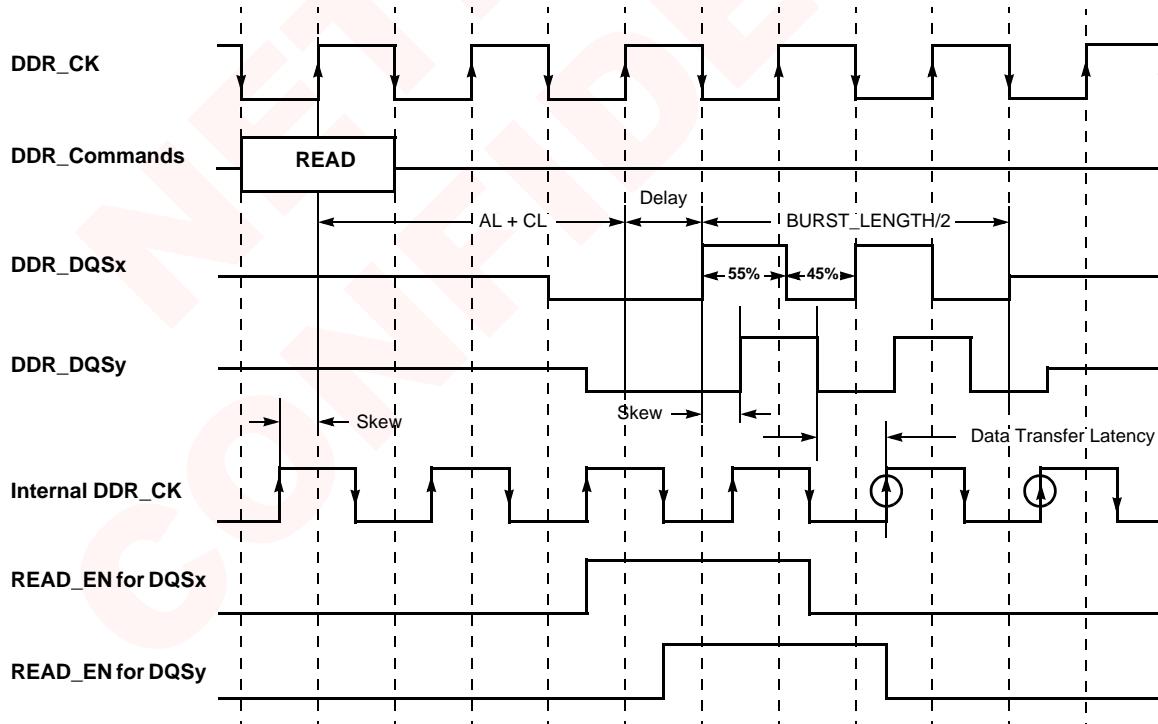
Note that if the READ_EN pulse is correctly asserted during the first rising and first falling edge of the Read DQS strobe, it need not remain asserted during subsequent rising and falling edges. An internal edge counter will automatically stretch the READ_EN pulse to the appropriate duration, based on the value programmed in the [DDR_PARAMS_2.TBRL-1](#) register field.

[Figure 11-10](#) shows some example timing waveforms. The first set of waveforms shows the ideal case, i.e. ignoring board and DRAM device delays. The first low-to-high transition of all the Read DQS strobes occurs AL + CL (i.e., additive latency and CAS latency) cycles after the READ command is issued on the DRAM command bus. The Read preamble begins approximately 1 cycle before this, the first falling edge occurs 0.5 cycles after this, and the DQS strobe post-amble will be completed [[BURST_LENGTH/2](#)] cycles after this.

The second set of waveforms shows several types of actual behavior, due to the board, package, and DRAM device delays involved.

1. Actual latency of the Read operation is larger than ideal
2. Different DQS Read strobes are skewed with respect to each other
3. Skew between internal and DDR command bus clock domains
4. DQS duty-cycle variation

Note how the position of the READ_EN pulses has been adjusted to compensate for this actual board behavior.

Figure 11-10. Example Read Data Functional Timing Diagrams**Ideal Case****Actual Case**

Each READ_EN pulse length is adjustable between 1 cycle and 2.5 cycles in 0.5 cycle increments, and the position is adjustable in 0.5 cycle increments. Each of the GEN_HALF_EARLY, GEN_CYC_NOMINAL, GEN_HALF_LATE, and GEN_CYC_LATE bits generates a 1-cycle wide pulse, and if more than one of these bits are asserted, the generated pulse is the logical OR of the individual pulses. Table 11-9 shows an example.

Table 11-9. Example DRAM Read Data Timing Parameters

DDR_PARAMS_3	DDR_DQS[0-9]_RX_DLY_CFG [10:7]				READ_EN Pulse (Cycles after READ Command)	
	.GEN_CYC_LATE [10]	.GEN_HALF_LATE [9]	.GEN_CYC_NOMINAL [8]	.GEN_HALF_EARLY [7]	Start	End
.TRL [24:20]						
-	0	0	0	0	None	None
n	0	0	0	1	n - 0.5	n + 0.5
n	0	0	1	1	n - 0.5	n + 1.0
n	0	0	1	0	n	n + 1.0
n	0	1	1	0	n	n + 1.5
n	0	1	0	0	n + 0.5	n + 1.5
n	1	1	0	0	n + 0.5	n + 2.0
n	1	0	0	0	n + 1.0	n + 2.0
n + 1	0	0	0	1	n + 0.5	n + 1.5
n + 1	0	0	1	1	n + 0.5	n + 2.0
n + 1	0	0	1	0	n + 1.0	n + 2.0
n + 1	0	1	1	0	n + 1.0	n + 2.5

The READ_EN waveforms shown in the ideal case portion of Figure 11-10 correspond to the following programmed values.

DDR_PARAMS_3.TRL = 0x2

DDR_DQSx_RX_DLY_CFG[10:7] = 0x3

DDR_DQSy_RX_DLY_CFG[10:7] = 0x3

The READ_EN waveforms shown in the actual case portion of Figure 11-10 correspond to the following programmed values.

DDR_PARAMS_3.TRL = 0x2

DDR_DQSx_RX_DLY_CFG[10:7] = 0x6

DDR_DQSy_RX_DLY_CFG[10:7] = 0xC

To summarize, the recommended operating procedure is to use a software routine to sweep the appropriate configuration register settings, starting with the lowest possible value (DDR_PARAMS_3.TRL = ADD_LAT + CAS_LATENCY, DDR_DQS[0-9]_RX_DLY_CFG = 0x3), and going to the highest possible expected value (e.g. DDR_PARAMS_3.TRL = ADD_LAT + CAS_LATENCY + 2, DDR_DQS[0-9]_RX_DLY_CFG[10:7] = 0x8). When a configuration is found that successfully gates all the data bus Read DQS strobes, these settings can be saved and used.

11.4.7.2 Capturing the Data

Once the gated version of the Read DQS strobe is available, it is passed through a DLL and phase-shifted. The output is then used to clock the XLS Read-data capture flip-flops. The goal of the phase shifting is to position the DQS rising or falling edge as close to the center of the data valid window as possible, thereby yielding maximum setup and hold margins. As explained in [Section 11.4.3.3, “Programming the DLL Control Registers”](#), the phase-shift amount is adjustable in increments of 1/128 of a clock cycle by programming `DDR_DQS[0-9]_RX_DLY_CFG.DLY_CFG`. For Reads, a 90 degree phase shift (i.e. `DDR_DQS[0-9]_RX_DLY_CFG.DLY_CFG = 0x20`) is the default value. Once actual skews are factored in, the optimal value may be slightly larger or smaller. The recommended operating procedure is to use a software routine to sweep the range of possible values during system bring-up and determine the optimal one.

11.4.7.3 Moving Captured Data into Controller Clock Domain

Once the Read data has been captured using the DQS strobe, it must be moved into the DRAM memory controller internal clock domain. In [Figure 11-10](#), in both the ideal case and actual case portions, there are two rising edges of the internal memory controller clock that are indicated by circles. At the time indicated by one of these circles, the received Read data corresponding to the first rising edge and first falling edge of the DQS strobes is assumed to be captured and stable (i.e. ready for re-synchronization to the memory controller internal clock domain). The first (left) circle applies if `DDR_PARAMS_4.3CYC_EN` is 0, otherwise the second (right) circle applies.

The recommended operating procedure for this bit during system bring-up is to begin with it set, because this will work in all cases. Once all the other parameter values have been established, the user should attempt operation with this bit cleared. Leaving this bit cleared reduces overall Read transaction latency and therefore improves performance, but this may not always be possible, due to the internal XLS delays involved. For example, in the ideal case section of [Figure 11-10](#), `DDR_PARAMS_4.3CYC_EN = 0` is achievable because the captured Read data is allocated 1.5 cycles of data transfer latency. However, in the actual case section of [Figure 11-10](#), the late arrival of DDR_DQSy means that using the first circled edge would leave less than 0.5 cycle for data transfer latency. Hence `DDR_PARAMS_4.3CYC_EN = 1` is probably required in this case, depending on the operating frequency of the memory interface.

11.4.8 Initialization Sequence

Table 11-10 below shows the suggested initialization sequence to program for DDR2 memory protocol. Note that the command counts are repeat counts, i.e. a value of n means the command is issued n+1 times.

Table 11-10. Suggested Initialization Sequences

Register	Field	DDR2 Value
DDR_CFG_REG_RESET_TIMERS	RST_STRT	1
	RST_LEN	1
	RST_SEQ_LEN	0xD
DDR_CFG_REG_RESET_CMD0	RESET_CMD_0	0x2F (PWR_DN_EXIT)
	RESET_CMD_CNT_0-1	0x0
	RESET_CMD_1	0x2E (NOP)
	RESET_CMD_CNT_1-1	0xA0
DDR_CFG_REG_RESET_CMD1	RESET_CMD_2	0x25 (PRECHARGE_ALL)
	RESET_CMD_CNT_2-1	0x0
	RESET_CMD_3	0x22 (EMRS2)
	RESET_CMD_CNT_3-1	0x0
DDR_CFG_REG_RESET_CMD2	RESET_CMD_4	0x23 (EMRS3)
	RESET_CMD_CNT_4-1	0x0
	RESET_CMD_5	0x01 (ENABLE_DLL)
	RESET_CMD_CNT_5-1	0x0
DDR_CFG_REG_RESET_CMD3	RESET_CMD_6	0x27 (RESET_DLL)
	RESET_CMD_CNT_6-1	0x0
	RESET_CMD_7	0x25 (PRECHARGE_ALL)
	RESET_CMD_CNT_7-1	0x0
DDR_CFG_REG_RESET_CMD4	RESET_CMD_8	0x32 (AUTO_REFRESH)
	RESET_CMD_CNT_8-1	0x2
	RESET_CMD_9	0x20 (MRS)
	RESET_CMD_CNT_9-1	0x0
DDR_CFG_REG_RESET_CMD5	RESET_CMD_10	0x2E (NOP)
	RESET_CMD_CNT_10-1	0xC8
	RESET_CMD_11	0x03 (DEFAULT_OCD)
	RESET_CMD_CNT_11-1	0x0
DDR_CFG_REG_RESET_CMD6	RESET_CMD_12	0x21 (EMRS1)
	RESET_CMD_CNT_12-1	0x0
	RESET_CMD_13	0x2E (NOP)
	RESET_CMD_CNT_13-1	0x0

11.4.9 Using DDR2 On-Die Termination (ODT)

This section summarizes some of the information needed to correctly program the DRAM memory controllers while using the DDR2 ODT feature.

Table 11-11, extracted from technical note TN-47-01 ("DDR2 Design Guide for Two-DIMM Systems") published by Micron Technology, shows the termination resistance values to be used in various memory system configurations.

Table 11-11. DDR2 Termination Values for Read/Write and Different Configurations

Configuration	Operation	Termination		
		At Controller	On Module 1	On Module 2
Single slot populated	Write to Slot 1	None	150 ohm	Empty
	Write to Slot 2	None	Empty	150 ohm
Two slots populated	Write to Slot 1	None	None	75 ohm
	Write to Slot 2	None	75 ohm	None
Single slot populated	Read from Slot 1	75 ohm	None	Empty
	Read from Slot 2	75 ohm	Empty	None
Two slots populated	Read from Slot 1	150 ohm	None	75 ohm
	Read from Slot 2	150 ohm	75 ohm	None

The relevant memory controller register fields are listed below.

1. The [DDR_PARAMS_4.ODT_EN](#) bit should be set to enable assertion of the ODT signals to the DRAM chip.
2. The [DDR_PARAMS_4.ODT_WR_CMD_ISS_DLY](#) field determines the number of cycles between a WRITE command being issued and the assertion of the ODT signals.
3. The [DDR_PARAMS_4.ODT_CMD_ISS_DLY](#) field determines the number of cycles between a READ command being issued and the assertion of the ODT signals.
4. The [DDR_PARAMS_4.TBOL](#) field determines the number of cycles the ODT signals remain asserted.
5. The [DDR_PARAMS_4.C_T_EN](#) bit enables controller-side termination.
6. The [DDR_PARAMS_4.C_T_A_EN](#) bit should be cleared.
7. The [DDR_RX0_THERM_CTRL.TERM_SEL](#) field can be used to select the appropriate controller-side termination resistance values.
8. The [DDR_CFG_REG_EMRS1.RTT_MSB](#) and [DDR_CFG_REG_EMRS1.RTT_LSB](#) fields select between 75 and 150 ohm DRAM chip-side termination resistance values.
9. The [DDR_CFG_REG_WRITE_ODT_MASK](#) register determines which ODT signals are asserted during *writes* to specific ranks.
10. The [DDR_CFG_REG_READ_ODT_MASK](#) register determines which ODT signals are asserted during *reads* from specific ranks.
11. The [DDR_GLB_PARAMS.ODT2_MUX_EN_L](#) bit, if cleared, converts the DDR_MA_14 and DDR_MA15 address outputs into a third and fourth ODT bits per port.
Recommended operating procedure is to leave this bit set.

11.4.10 DRAM Controller Modes

The XLS DRAM memory controllers support out-of-order issuing of memory transactions to optimize bus bandwidth. There are a number of mode bits associated with transaction scheduling and page-policy which are programmable. These bits are described below.

When transactions are first received in the memory controller, they are stored in queues based on the bank and rank they will be accessing. Arbitration logic determines when these stored transactions are issued. If the [DDR_GLB_PARAMS.DIS_BP](#) bit is cleared, a received transaction may bypass the queuing and arbitration logic and be immediately issued if all the queues are empty.

Read-to-Write and Write-to-Read bus turnarounds require a relatively large number of idle cycles. The memory controllers attempt to reduce the number of such turnarounds by grouping Reads and Writes into bursts. The [DDR_BNK_CTL.BWC-1](#) and [DDR_BNK_CTL.BRC-1](#) register fields determine the maximum number of transactions in a burst. If set, the [DDR_GLB_PARAMS.ORWR](#) bit enables early change of priority for banks to which only Write transactions are pending.

Note that recommended operating procedure is to leave these transaction-scheduling related mode bits at their default reset values.

In the DDR2 memory protocols, the memory is grouped into pages, typically 1 KB or 2 KB in size. Before accessing a page, it must be opened using an ACTIVE command. It may then be read from or written to. Before another page in the same bank can be accessed, the first page must be closed by a PRECHARGE command. The [DDR_GLB_PARAMS.OPP](#) and [DDR_GLB_PARAMS.CPO](#) bits determine the page policy that is followed by the DRAM memory controller, as shown in [Table 11-12](#):

Table 11-12. DDR Page Policies

DDR_GLB_PARAMS.OPP	DDR_GLB_PARAMS.CPO	Policy Behavior
0	0	Pages are always closed, i.e. a PRECHARGE command is always issued after a READ or WRITE.
0	1	Pages are always closed, unless there is a pending transaction to the same page.
1	-	Pages are left open as long as possible.

The optimal page policy for a particular software application is dependent on the memory access behavior of that application. Therefore, recommended operating procedure for these page-policy mode bits is to leave them at their default reset values, unless significant performance gain is exhibited by changing the page policy.

11.4.11 Programming Sequence

Programming the memory controllers involves correctly programming the relevant DRAM controller configuration registers, the DRAM-specific registers within the appropriate bridge, as well as the clock control registers within the GPIO block. The order of programming is significant.

If operating with single channels, each of the four memory controllers must be programmed and activated. If operating with channel pairs, only the two master memory controllers must be programmed and activated.

The programming described below is usually performed while in state 3 of the DRAM controller initialization sequence, as described in [Section 11.4.2, “Memory Controller Initialization Sequence”](#).

11.4.11.1 DDR2 Programming Sequence

1. Program the appropriate registers in the GPIO register space with the desired DRAM clock frequency. Poll to ensure the PLL has re-locked at the new frequency.
2. Program the BRIDGE_CFG register in the bridge with the desired memory protocol and channel configuration. This enables access to the appropriate set of DRAM memory controller configuration registers.
3. Program the [DDR_GLB_PARAMS](#) register. Do not activate ECC at this time.
4. Program the strobe configuration registers: [DDR_CMD_DLY_CFG](#), [DDR_SLV_CMD_DLY_CFG](#), [DDR_DQS\[0-9\]_RX_DLY_CFG](#) and [DDR_DQS\[0-9\]_TX_DLY_CFG](#).
5. Trigger a DLL calibration by writing to the [DDR_CLK_CAL_REQ](#) register.
6. Program the [DDR_CFG_REG_WRITE_ODT_MASK](#) and [DDR_CFG_REG_READ_ODT_MASK](#) registers.
7. Program the [DDR_ADDR_PARAMS_1](#) and [DDR_ADDR_PARAMS_2](#) registers.
8. Program the [DDR_PARAMS_1](#) to [DDR_PARAMS_7](#) registers.
9. Program the [DDR_CFG_REG_MRS](#) and [DDR_CFG_REG_RDMRS](#) registers.
10. Program the [DDR_CFG_REG_EMRS1](#) register.
11. Clear all the poison mask registers: [DDR_CFG_REG_ECC_POISON_MASK](#), [DDR_CFG_REG_POISON_MASK_0](#), [DDR_CFG_REG_POISON_MASK_1](#), [DDR_CFG_REG_POISON_MASK_2](#) and [DDR_CFG_REG_POISON_MASK_3](#).
12. Program all four thermal control registers: [DDR_TX0_THERM_CTRL](#), [DDR_TX1_THERM_CTRL](#), [DDR_RX0_THERM_CTRL](#) and [DDR_RX1_THERM_CTRL](#).
13. Program the reset sequence registers: [DDR_CFG_REG_RESET_CMD0](#) through [DDR_CFG_REG_RESET_CMD6](#).
14. Program the [DDR_CFG_REG_RESET_TIMERS](#) register, remembering to trigger entry into state 4 of the DRAM controller initialization sequence, as described in [Section 11.4.2, “Memory Controller Initialization Sequence”](#).
15. Program and enable the DRAM_BAR[0-7] and DRAM_CHN[AC/BD]_DTR[0-7] registers in the memory bridge.
16. Scrub the entire DRAM memory region by setting all data values to zero.
17. Enable ECC by writing to the [ECC_REPORT_EN](#) and [ECC_DETECT_EN](#) bits of the [DDR_GLB_PARAMS\[11 & 5\]](#) register.

11.5 Memory Controller Register Descriptions

11.5.1 Memory Controller Register Summary

[Table 11-13](#) is a summary of all of the registers for each of the channels A-D. The individual register offsets all apply within the four sub-regions listed in [Table 11-6](#) depending upon the memory type and channel. Address offsets are on four-byte boundaries, and are computed as 4 times the Register ID.

Note that the DRAM clock rates are set in the GPIO register space as described in [Chapter 23, “GPIO and the Peripherals Interface”](#). They are in [Section 23.3.2, “DRAM Clock Rate Registers”](#) and [Section 23.3.3, “PLL Control and Status Registers”](#).

Table 11-13. Memory Controller Register Summary

Register ID	Register Name	Description	R/W	Reset Value
Section 11.5.2, “DDR Timing Parameter Registers”				
0x0	DDR_PARAMS_1	DDR Timing Parameters	R/W	0x11B0 8C22
0x1	DDR_PARAMS_2	DDR Timing Parameters	R/W	0x0031 ADE0
0x2	DDR_PARAMS_3	DDR Timing Parameters	R/W	0x0072 A1A6
0x3	DDR_PARAMS_4	DDR Timing Parameters	R/W	0x0614 1885
0x4	DDR_PARAMS_5	DDR Timing Parameters	R/W	0x0010 0EC8
0x1A	DDR_PARAMS_6	DDR Timing Parameters	R/W	0x2C44 B64C
0x29	DDR_PARAMS_7	DDR Timing Parameters	R/W	0x3DEF 2504
Section 11.5.3, “DDR Controller Configuration Registers”				
0x5	DDR_BNK_CTL	DDR Bank Read/Write modes	R/W	0x0000 0067
0x6	DDR_ADDR_PARAMS_1	DDR Address Parameters 1	R/W	0x04E9 C026
0x1B	DDR_ADDR_PARAMS_2	DDR Address Parameters 2	R/W	0x0003 900A
0x7	DDR_GLB_PARAMS	DDR Global Parameters	R/W	0x0000 015A
Section 11.5.4, “DDR DLL Control Registers”				
0x8	DDR_CLK_CAL_PARAMS	DDR Clock Calibration Parameters	R/W	0x0000 7856
0x9	DDR_CLK_CAL_REQ	DDR Clock Calibration Request	W	0x0000 0000
0x100 - 0x124 by 4	DDR_DQS[0-9]_RX_NXT_DLY_CNT	DDR Receive Next Delay Count Registers	R/W	0x0000 0000
0X140 - 0X164 by 4	DDR_DQS[0-9]_TX_NXT_DLY_CNT	DDR Transmit Next Delay Count Registers	R/W	0x0000 0000
0x13C	DDR_CMD_NXT_DLY_CNT	DDR Command Next Delay Count	R/W	0x0000 0000
0x138	DDR_SLV_CMD_NXT_DLY_CNT	DDR Slave Command Next Delay Count	R/W	0x0000 0000
0x101 - 0x125 by 4	DDR_DQS[0-9]_RX_DLY_CNT	DDR Receive Current Delay Count Registers	RO	0x0000 0000
0X141 - 0X165 by 4	DDR_DQS[0-9]_TX_DLY_CNT	DDR Transmit Current Delay Count Registers	RO	0x0000 0000
0x13D	DDR_CMD_DLY_CNT	DDR Command Current Delay Count	RO	0x0000 0000
0x139	DDR_SLV_CMD_DLY_CNT	DDR Slave Command Current Delay Count	RO	0x0000 0000
0x102 - 0x126 by 4	DDR_DQS[0-9]_RX_CYC_MEAS_CNT	DDR Receive Cycle Measurement Count Registers	RO	0x0000 0000
0X142 - 0X166 by 4	DDR_DQS[0-9]_TX_CYC_MEAS_CNT	DDR Transmit Cycle Measurement Count Registers	RO	0x0000 0000
0x13E	DDR_CMD_CYC_MEAS_CNT	DDR Command Cycle Measurement Count	RO	0x0000 0000

Table 11-13. Memory Controller Register Summary (continued)

Register ID	Register Name	Description	R/W	Reset Value
0x13A	DDR_SLV_CMD_CYC_MEAS_CNT	DDR Slave Command Cycle Measurement Count	RO	0x0000 0000
0x103 - 0x127 by 4	DDR_DQS[0-9]_RX_DLY_CFG	DDR Receive Delay Configuration Registers	R/W	0x0000 0020
0X143 - 0X167 by 4	DDR_DQS[0-9]_TX_DLY_CFG	DDR Transmit Delay Configuration Registers	R/W	0x0000 0020
0x13F	DDR_CMD_DLY_CFG	DDR Command Delay Configuration	R/W	0x0000 0040
0x13B	DDR_SLV_CMD_DLY_CFG	DDR Slave Command Delay Configuration	R/W	0x0000 0040
Section 11.5.5, "DDR Thermal Control Registers"				
0x130	DDR_RX0_THERM_CTRL	DDR Receive 0 Thermal Control	R	0x0000 0001
0x134			W	
0x131	DDR_RX1_THERM_CTRL	DDR Receive 1 Thermal Control	R	0x0000 0001
0x135			W	
0x132	DDR_TX0_THERM_CTRL	DDR Transmit 0 Thermal Control	R	0x0000 0201
0x136			W	
0x133	DDR_TX1_THERM_CTRL	DDR Transmit 1 Thermal Control	R	0x0000 0001
0x137			W	
Section 11.5.6, "DDR MRS Command Registers"				
0x14	DDR_CFG_REG_MRS	DDR Configuration Register for Mode Register Set	R/W	0x0000 0643
0x15	DDR_CFG_REG_RDMRS	DDR Configuration Register for MRS with DLL reset	R/W	0x0000 0743
0x16	DDR_CFG_REG_EMRS1	DDR Configuration Register 1 for Extended Mode Register Set	R/W	0x0000 0018
0x17	DDR_CFG_REG_EMRS2	DDR Configuration Register 2 for Extended Mode Register Set	R/W	0x0000 0000
0x18	DDR_CFG_REG_EMRS3	DDR Configuration Register 3 for Extended Mode Register Set	R/W	0x0000 0000
Section 11.5.7, "DDR Reset Sequence Control Registers"				
0x19	DDR_CFG_REG_RESET_TIMERS	DDR Configuration Register for Reset Timers	R/W	0x0000 080D
0x1C	DDR_CFG_REG_RESET_CMD0	DDR Configuration Register 0 for Reset Command	R/W	0x1BAE 002F
0x1D	DDR_CFG_REG_RESET_CMD1	DDR Configuration Register 1 for Reset Command	R/W	0x0022 0025
0x1E	DDR_CFG_REG_RESET_CMD2	DDR Configuration Register 2 for Reset Command	R/W	0x0001 0023
0x1F	DDR_CFG_REG_RESET_CMD3	DDR Configuration Register 3 for Reset Command	R/W	0x0025 0027
0x20	DDR_CFG_REG_RESET_CMD4	DDR Configuration Register 4 for Reset Command	R/W	0x0020 00B2
0x21	DDR_CFG_REG_RESET_CMD5	DDR Configuration Register 5 for Reset Command	R/W	0x0003 282E
0x22	DDR_CFG_REG_RESET_CMD6	DDR Configuration Register 6 for Reset Command	R/W	0x002E 0021
Section 11.5.8, "DDR ECC Registers"				
0x2C	DDR_CFG_REG_RMW_ECC_LOG_LOW	DDR read-modify-write ECC error address logging register	R/W	0x0000 0000

Table 11-13. Memory Controller Register Summary (continued)

Register ID	Register Name	Description	R/W	Reset Value
0x23	DDR_CFG_REG_ECC_LOG_HI	DDR ECC error logging register	R/W	0x0000 0000
0x24	DDR_CFG_REG_ECC_POISON_MASK	DDR Configuration Register for ECC Poison Mask	R/W	0x0001 1001
0x25	DDR_CFG_REG_POISON_MASK_0	DDR Configuration Register 0 for Poison Mask	R/W	0x0000 0000
0x26	DDR_CFG_REG_POISON_MASK_1	DDR Configuration Register 1 for Poison Mask	R/W	0x0000 0000
0x27	DDR_CFG_REG_POISON_MASK_2	DDR Configuration Register 2 for Poison Mask	R/W	0x0000 0000
0x28	DDR_CFG_REG_POISON_MASK_3	DDR Configuration Register 3 for Poison Mask	R/W	0x0000 0000
Section 11.5.9, “DDR ODT Control Registers”				
0x2A	DDR_CFG_REG_WRITE_ODT_MASK	DDR Configuration Register for Write ODT	R/W	0x0000 0000
0x2B	DDR_CFG_REG_READ_ODT_MASK	DDR Configuration Register for Read ODT	R/W	0x0000 0000
0x30	DDR_DYN_DATA_PAD_CTRL	DDR Dynamic Data Pad Control	R/W	0x6031 6041
0x42	Reserved		R/W	0x0292 C008
0x43	Reserved		R/W	0x0000 0004
0x44	Reserved		R/W	0x0000 0003
0x45	Reserved		R/W	0x0000 0003
Section 11.5.9.3, “DDR_DYN_DATA_PAD_CTRL”				
0x2D	DDR_COUNTER_CONF	Performance Count Configuration	R/W	0x75C0 EB80
0x2E	DDR_PRF_COUNTER0	Performance Counter 0 value	R/W	0x0000 0000
0x2F	DDR_PRF_COUNTER1	Performance Counter 1 value	R/W	0x0000 0000

11.5.2 DDR Timing Parameter Registers

The first seven Parameter Registers are for DDR timing parameters. Timing parameters are in units of number of memory clock cycles.

For these seven registers, `DDR_PARAMS_[1-7]`, writes to the register may not take effect immediately. New values become active upon the following conditions.

1. During State 4 of the memory controller initialization sequence, and
2. By completion of a user command sequence (which is triggered by setting `DDR_CFG_REG_RESET_TIMERS.CMD_SEQ_RQD`).

The following keys (terms) describe the notation used in the Formula column of the register descriptions.

t_{xxx}: This term is used to describe the numerical value of the DRAM timing parameter with the same name after conversion into clock cycles by dividing with the operating frequency value. For example, if a DRAM data sheet describes t_{RRD} as 10 ns, and the chosen operating frequency is 266 MHz, the value of t_{RRD} in the equations below would be $10/3.75 = 3$ clock cycles. Note that when rounding, always round up, except where explicitly specified otherwise.

AL: Additive Latency as described in JEDEC DDR2 specification. The value of the additive latency is chosen by the user, in units of clock cycles.

CL: CAS Latency as described in JEDEC DDR2 specifications. The value of the CAS latency is determined by the type of DRAM being used. Units are clock cycles.

BL: Burst Length as described in JEDEC DDR2 specifications. If operating with single channels, BL must be 8, and if operating with channel pairs, burst length must be four.

RL: Read Latency, as defined in JEDEC DDR2 specification. Defined as AL + CL in DDR2 mode.

WL: Write Latency, as defined in JEDEC DDR/DDR2 specification. Defined as AL + CL - 1 in DDR2 mode.

t_{reg}: This term has the value of zero clock cycles if using discrete DRAM chips or unbuffered DIMMs, and equals one clock cycle if using registered DIMMs.

2*t_{board}: This term represents the delay from the XLS processor pin to the memory device pin plus the delay from the memory device pin back to the XLS pin, in units of clock cycles. This accounts for the package delay, board trace delay, and DIMM trace delay, among others. It is assumed that the values for the command bus, data strobes, and data bus are comparable.

11.5.2.1 DDR_PARAMS_1

This Read/Write register determines six DDR timing parameters. The initial value at reset is 0x11B0 8C22.

Register ID: 0x0
Address Offset: 0x000

31:29		28:26	25:20	19:15	14:10	9:5	4:0
Reserved	TRRDPW	TRFC-1	TMRD-1	TRP-1	TCCD-1	TRRD-1	

Bits	Field Name	Field Description	Formula	R/W	Reset
31:29	Reserved	Reserved	-	RO	b000
28:26	TRRDPW	Number of ACTIVE commands allowed Per t _{RRD} Window. Program as 4 for both 4 and 8 bank devices	-	R/W	b1 00
25:20	TRFC-1	REFRESH latency minus 1	t _{RF} C-1	R/W	b01 1011
19:15	TMRD-1	(E)MRS to any command Delay minus 1	t _M RD-1	R/W	b0000 1
14:10	TRP-1	PRECHARGE to ACTIVE delay minus 1	t _P R-1	R/W	b000 11
9:5	TCCD-1	CAS to CAS Delay minus 1	t _{CC} D-1	R/W	b00 001
4:0	TRRD-1	ACTIVE to ACTIVE Delay (different bank) minus 1	t _{RR} D-1	R/W	b0 0010

11.5.2.2 DDR_PARAMS_2

This Read/Write register determines five additional DDR timing parameters. The initial value at reset is 0x0031 ADE0.

Register ID: 0x1
Address Offset: 0x004

31:25		24:20	19:15	14:10	9:5	4:0
Reserved		TBWL-1	TBRL-1	TRAS-1	TRC-1	TRCD-AL-1

Bits	Field Name	Field Description	Formula	R/W	Reset
31:25	Reserved	Reserved	-	RO	b0000 000
24:20	TBWL-1	Burst Write Latency minus 1	BL/2 - 1	R/W	b0 0011
19:15	TBRL-1	Burst Read Latency minus 1	BL/2 - 1	R/W	b0001 1
14:10	TRAS-1	ACTIVE to PRECHARGE delay minus 1	t _{RAS(min)} -1	R/W	b010 11
9:5	TRC-1	ACTIVE to ACTIVE delay (same bank) minus 1	t _R C-1	R/W	b01 111
4:0	TRCD-AL-1	ACTIVE to READ or WRITE Delay minus Additive Latency minus 1.	t _{RCD-AL} -1	R/W	b0 0000

11.5.2.3 DDR_PARAMS_3

This Read/Write register determines five additional DDR timing parameters. The initial value at reset is 0x0072 A1A6.

Register ID: **0x2**

Address Offset: **0x008**

31:25		24:20	19:15	14:10	9:5	4:0	
Reserved		TRL	TRTW-1	TWTR-1	TWTP-1	TRTP-1	
Bits	Field Name	Field Description		Formula		R/W	Reset
31:25	Reserved	Reserved		-		RO	b0000 000
24:20	TRL	Read Latency		AL + CL + t_{reg} + 2*t _{board}		R/W	b0 0111
19:15	TRTW-1	Read to Write Turnaround latency minus 1 Note: t_{reg} has no effect on this parameter provided WL has been increased by t_{reg}		For DDR2: BL/2 + 1 + 2*t _{board}		R/W	b0010 1
14:10	TWTR-1	Write to Read Turnaround latency minus 1		For DDR2: CL + BL/2 + t_{wtr} - 2		R/W	b010 00
9:5	TWTP-1	WRITE to PRECHARGE delay minus 1. When using registered DIMMs, even though WL increases by 1, this parameter need not increase.		For DDR2: AL + CL + BL/2 + t_{wr} - 2		R/W	b01 101
4:0	TRTP-1	READ to PRECHARGE delay minus 1		Largest of: a) AL + BL/2 - 1 b) AL + BL/2 + t_{RTP} - 3 c) [TRL] + BL/2 - [TRP-1] - 3		R/W	b0 0110

11.5.2.4 DDR_PARAMS_4

This Read/Write register determines nine additional DDR timing parameters. The value at reset is 0x0614 1885.

Register ID: 0x3
Address Offset: 0x00C

31	30	29	28	27:25	24:20	19:16	15:11	10	9:5	4:0
Rsv	C_T_EN	C_T_A_EN	3CYC_EN	Rsv	ODT_WR_CMD_ISS_DLY	TBOL	ODT_CMD_ISS_DLY	ODT_EN	TRPA-1	TWL-1

Bits	Field Name	Field Description	Formula	R/W	Reset
31	Reserved	Reserved	-	RO	b0
30	C_T_EN	Controller Termination Enable. In XLS 100 and 200 series, this field is ignored if DYN_TERM_EN is set in the DDR_DYN_DATA_PAD_CTRL register, and termination properties are set as shown in Table 11-15 1: enable	-	R/W	b0
29	C_T_A_EN	Controller Termination Always Enabled 1: enable termination always	-	R/W	b0
28	3CYC_EN	Enable 3 Cycles for returning READ data instead of two. See description in Section 11.4.7.3, "Moving Captured Data into Controller Clock Domain" . 1: enable	-	R/W	b0
27:25	Reserved	Reserved.	-	RO	b011
24:20	ODT_WR_CMD_ISS_DLY	ODT WRITE Command Issue Delay. Controller-provided ODT WRITE command issue delay is $2 + [TWL-1] - ODT_WR_CMD_ISS_DLY$. Illegal condition: if programmed such that [TWL-1] is < ODT_WR_CMD_ISS_DLY , no ODT command is issued.	$t_{AOND} + 2 + t_{reg}$	R/W	b0 0001
19:16	TBOL	Burst ODT Latency	$BL/2 + 1$	R/W	b0100
15:11	ODT_CMD_ISS_DLY	ODT Command Issue Delay. Controller-provided ODT command issue delay for READ operations is $2 + DDR_PARAMS_3.TRL - ODT_CMD_ISS_DLY$. Illegal condition: if programmed such that ODT_CMD_ISS_DLY is > DDR_PARAMS_3.TRL , no ODT command is issued.	$t_{AOND} + 3 + t_{reg} + 2*t_{board}$	R/W	b0001 1
10	ODT_EN	ODT Enable. Set this bit if operating in DDR2 mode. 0: do not enable 1: enable ODT for DDR2		R/W	b0
9:5	TRPA-1	PRECHARGE_ALL delay minus 1	t_{RP}	R/W	b00 100
4:0	TWL-1	Write Latency minus 1	$AL + CL - 2 + t_{reg}$	R/W	b0 0101

11.5.2.5 DDR_PARAMS_5

This Read/Write register determines three additional DDR timing parameters. The value at reset is 0x0010 0EC8.

Register ID: 0x4

Address Offset: 0x010

31:28	27:25	24:9	8:0
Reserved	TRBC-1	TREF-1	TXSC-1

Bits	Field Name	Field Description	Formula	R/W	Reset
31:28	Reserved	Reserved	-	RO	b0000
27:25	TRBC-1	Number of REFRESH commands issued for every refresh, minus 1	-	R/W	b000
24:9	TREF-1	Refresh interval minus 1	$t_{REFI}-1$	R/W	b0 0001 0000 0000 111
8:0	TXSC-1	Self Refresh Exit Count minus 1	$t_{XSRD}-1$	R/W	b0 1100 1000

11.5.2.6 DDR_PARAMS_6

This Read/Write register determines five additional DDR timing parameters. The value at reset is 0x2C44 B64C.

Register ID: 0x1A

Address Offset: 0x068

31	30:26	25:21	20:18	17:15	14:0
Reserved	TRTPD-1	TCKE-1	TXARD-1	TXP-1	TRASMAX-1

Bits	Field Name	Field Description	Formula	R/W	Reset
31	Reserved	Reserved	-	RO	b0
30:26	TRTPD-1	READ to Power Down entry latency minus 1	$AL + CL + BL/2$	R/W	b010 11
25:21	TCKE-1	CKE minimum pulse duration in cycles minus 1	$t_{CKE}-1$	R/W	b00 010
20:18	TXARD-1	Exit active power down to a command Delay minus 1	largest of: $t_{XP}-1$, $t_{XARD}-1$, $t_{XARDS}-1$	R/W	b0 01
17:15	TXP-1	Exit Precharge power down to a command delay minus 1	largest of: $t_{XP}-1$, $t_{XPRD}-1$	R/W	b00 1
14:0	TRASMAX-1	$t_{RAS(max)}$ minus 1. Program as 20 cycles less than $t_{RAS(max)}$, in order to make sure the controller can issue the command in time.	$t_{RAS(max)}-1$	R/W	b011 0110 0100 1100

11.5.2.7 DDR_PARAMS_7

This Read/Write register determines an additional DDR timing parameter, as well as two alternative timing parameters that only apply while switching ranks. The value at reset is 0x3DEF 2504.

Register ID: 0x29
Address Offset: 0x0A4

		30:15	14:10	9:5	4:0	
rsv	ALT_PARAM_RANK_MASK	TRRDWINDOW-1	ALT_TWTR-1	ALT_TBRL-1		
Bits	Field Name	Field Description		Formula	R/W	Reset
31	Reserved	Reserved		-	RO	b0
30:15	ALT_PARAM_RANK_MASK[30:15]	A 4 x 4 rank-switch mask matrix to determine if the alternative timing parameters apply		-	R/W	0x7BDE
30	ALT_PARAM_RANK_MASK[30]	1: If set, timing between a Read from rank 3 and a Read from rank 3 is at least ALT_TBRL rather than TBRL and timing between a Write to rank 3 and a Read from rank 3 is at least ALT_TWTR rather than TWTR.		-	R/W	b0
29	ALT_PARAM_RANK_MASK[29]	1: If set, timing between a Read from rank 3 and a Read from rank 2 is at least ALT_TBRL rather than TBRL and timing between a Write to rank 3 and a Read from rank 2 is at least ALT_TWTR rather than TWTR.		-	R/W	b1
28	ALT_PARAM_RANK_MASK[28]	1: If set, timing between a Read from rank 3 and a Read from rank 1 is at least ALT_TBRL rather than TBRL and timing between a Write to rank 3 and a Read from rank 1 is at least ALT_TWTR rather than TWTR.		-	R/W	b1
27	ALT_PARAM_RANK_MASK[27]	1: If set, timing between a Read from rank 3 and a Read from rank 0 is at least ALT_TBRL rather than TBRL and timing between a Write to rank 3 and a Read from rank 0 is at least ALT_TWTR rather than TWTR.		-	R/W	b1
26	ALT_PARAM_RANK_MASK[26]	1: If set, timing between a Read from rank 2 and a Read from rank 3 is at least ALT_TBRL rather than TBRL and timing between a Write to rank 2 and a Read from rank 3 is at least ALT_TWTR rather than TWTR.		-	R/W	b1
25	ALT_PARAM_RANK_MASK[25]	1: If set, timing between a Read from rank 2 and a Read from rank 2 is at least ALT_TBRL rather than TBRL and timing between a Write to rank 2 and a Read from rank 2 is at least ALT_TWTR rather than TWTR.		-	R/W	b0
24	ALT_PARAM_RANK_MASK[24]	1: If set, timing between a Read from rank 2 and a Read from rank 1 is at least ALT_TBRL rather than TBRL and timing between a Write to rank 2 and a Read from rank 1 is at least ALT_TWTR rather than TWTR.		-	R/W	b1
23	ALT_PARAM_RANK_MASK[23]	1: If set, timing between a Read from rank 2 and a Read from rank 0 is at least ALT_TBRL rather than TBRL and timing between a Write to rank 2 and a Read from rank 0 is at least ALT_TWTR rather than TWTR.		-	R/W	b1
22	ALT_PARAM_RANK_MASK[22]	1: If set, timing between a Read from rank 1 and a Read from rank 3 is at least ALT_TBRL rather than TBRL and timing between a Write to rank 1 and a Read from rank 3 is at least ALT_TWTR rather than TWTR.		-	R/W	b1

Bits	Field Name	Field Description	Formula	R/W	Reset
21	ALT_PARAM_RANK _MASK[21]	1: If set, timing between a Read from rank 1 and a Read from rank 2 is at least ALT_TBRL rather than TBRL and timing between a Write to rank 1 and a Read from rank 2 is at least ALT_TWTR rather than TWTR.	-	R/W	b1
20	ALT_PARAM_RANK _MASK[20]	1: If set, timing between a Read from rank 1 and a Read from rank 1 is at least ALT_TBRL rather than TBRL and timing between a Write to rank 1 and a Read from rank 1 is at least ALT_TWTR rather than TWTR.	-	R/W	b0
19	ALT_PARAM_RANK _MASK[19]	1: If set, timing between a Read from rank 1 and a Read from rank 0 is at least ALT_TBRL rather than TBRL and timing between a Write to rank 1 and a Read from rank 0 is at least ALT_TWTR rather than TWTR.	-	R/W	b1
18	ALT_PARAM_RANK _MASK[18]	1: If set, timing between a Read from rank 0 and a Read from rank 3 is at least ALT_TBRL rather than TBRL and timing between a Write to rank 0 and a Read from rank 3 is at least ALT_TWTR rather than TWTR.	-	R/W	b1
17	ALT_PARAM_RANK _MASK[17]	1: If set, timing between a Read from rank 0 and a Read from rank 2 is at least ALT_TBRL rather than TBRL and timing between a Write to rank 0 and a Read from rank 2 is at least ALT_TWTR rather than TWTR.	-	R/W	b1
16	ALT_PARAM_RANK _MASK[16]	1: If set, timing between a Read from rank 0 and a Read from rank 1 is at least ALT_TBRL rather than TBRL and timing between a Write to rank 0 and a Read from rank 1 is at least ALT_TWTR rather than TWTR.	-	R/W	b1
15	ALT_PARAM_RANK _MASK[15]	1: If set, timing between a Read from rank 0 and a Read from rank 0 is at least ALT_TBRL rather than TBRL and timing between a Write to rank 0 and a Read from rank 0 is at least ALT_TWTR rather than TWTR.	-	R/W	b0
14:10	TRRDWINDOW-1	Eight-bank devices may not issue more than 4 ACTIVE commands in a window which is shorter than $4*t_{RRD} + 2$	$4*t_{RRD} + 1$	R/W	b010 01
9:5	ALT_TWTR-1	Alternate Write-to-Read Turnaround latency minus 1	$CL + BL/2 + t_{WTR} - 2$	R/W	b01 000
4:0	ALT_TBRL-1	Alternate Burst Read Latency minus 1	BL/2	R/W	b0 0100

11.5.3 DDR Controller Configuration Registers

11.5.3.1 DDR_BNK_CTL

This Read/Write register determines the two DDR read/write transition issue priority mode parameters. The value at reset is 0x0000 0067.

Register ID: 0x5

Address Offset: 0x014

31:10	9:5	4:0
<i>Reserved</i>	BWC-1	BRC-1

Bits	Field Name	Field Description	R/W	Reset
31:10	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 00	RO	See Field Desc
9:5	BWC-1	Burst Write Count minus 1. The number of Writes issued before switching to Read priority mode.	R/W	b00 011
4:0	BRC-1	Burst Read Count minus 1. The number of Reads issued before switching to Write priority mode.	R/W	b0 0111

11.5.3.2 DDR_ADDR_PARAMS_1

This Read/Write register determines DDR Addressing Parameters. The value at reset is 0x04E9 C026.

Register ID: 0x6

Address Offset: 0x018

31:30	29:25	24:21	20:16	15:12	11:7	6:4	3:2	1:0
rsv	BLCA	NCAB	BLRA	NRAB-1	BLBA	Reserved	NRKB	NBAB

Bits	Field Name	Field Description	R/W	Reset
31:30	Reserved	Reserved	RO	b00
29:25	BLCA	Bit Location of lsb of Column Address field. Legal values = 0 - 24.	R/W	b00 010
24:21	NCAB	Number of Column Address Bits. XLS Processor uses 32-byte cache lines; therefore, if not channel-pair mode, where BL = 8, program as (# of DRAM column address bits - 3), but in channel-pair mode, where BL = 4, program as (# of DRAM column address bits - 2). Legal values = 5 - 10.	R/W	b0 111
20:16	BLRA	Bit Location of lsb of Row Address field. Legal values = 0 - 17.	R/W	b0 1001
15:12	NRAB-1	Number of Row Address Bits minus one. Legal values = 11 - 15.	R/W	b1100
11:7	BLBA	Bit Location of lsb of Bank Address bits. Legal values = 0 - 27.	R/W	b0000 0
6:4	Reserved	Reserved	R/W	b010
3:2	NRKB	Number of Rank Address Bits. Legal values = 0, 1, 2.	R/W	b01
1:0	NBAB	Number of Bank Address Bits. Legal values = 2, 3.	R/W	b10

11.5.3.3 DDR_ADDR_PARAMS_2

This Read/Write register determines DDR Addressing Parameters. The value at reset is 0x0003 900A.

Register ID: **0x1B**

Address Offset: **0x06C**

31:19	18	17:16	15:14	13:12	11:10	9	8:4	3:0
RSV	USE_RANK_TO_CS_MAP	CS3_RANKID	CS2_RANKID	CS1_RANKID	CS0_RANKID	USE_RANK_LOCATION	BLRK	BLAP

Bits	Field Name	Field Description	R/W	Reset
31:19	Reserved	Reserved Reset value = b0000 0000 0000 0	RO	See Field Desc
18	USE_RANK_TO_CS_MAP	0: If cleared, the default mapping is assumed: DDR[A/B/C/D]_CS0_L: rank0 DDR[A/B/C/D]_CS1_L: rank1 DDR[A/B/C/D]_CS2_L: rank2 DDR[A/B/C/D]_CS3_L: rank3 1: If set, the Rank To Chip-Select Mapping is determined by DDR_ADDR_PARAMS_2[17:10].	R/W	b0
17:16	CS3_RANKID	Logical rank corresponding to DDR[A/B/C/D]_CS3_L.	R/W	b11
15:14	CS2_RANKID	Logical rank corresponding to DDR[A/B/C/D]_CS2_L.	R/W	b10
13:12	CS1_RANKID	Logical rank corresponding to DDR[A/B/C/D]_CS1_L.	R/W	b01
11:10	CS0_RANKID	Logical rank corresponding to DDR[A/B/C/D]_CS0_L.	R/W	b00
9	USE_RANK_LOCATION	0: If cleared, the rank address bits are assumed to be contiguous with the MSB of the bank address bits, i.e one of the following formats is assumed: bb, bbb, rbb, rbbb, rrbb, or rrrbb. 1: If set, the Location of the Rank address bits is specified by the BLRK field.	R/W	b0
8:4	BLRK	Bit location of lsb of rank address field	R/W	b0 0000
3:0	BLAP	Auto Precharge Bit Location	R/W	b1010

Address example

Assume that a DRAM chip data sheet listed the following:

Column Address: A[9:0]

Row Address: A[12:0]

Bank Address: BA[1:0]

Auto Precharge bit location: A[10]

Assume that the system configuration is as follows:

Ranks: 2

Data Bus: 64-bits wide

Burst Length: 4

Then the following register values could be programmed:

DDR_ADDR_PARAMS_1 = 0x070B C036

DDR_ADDR_PARAMS_2 = 0x0000 000A

The 29-bit cache-line address sent from the bridge to the memory controller would be interpreted as follows:

28:24	23:11	10:3	2	1:0
UN	RW	CL	RK	BK

Note that since this is a cache-line address, and the XLS Processor cache lines are 32-bytes wide, the actual amount of memory being addressed is 512 MB.

The following would be observed on the MA - MD Port signals:

	DDR_MA[15:0]					DDR_BA[2:0]	
	15:13	12:11	10	9:2	1:0	2	1:0
ACTIVE Command Issued	0	RW				0	BK
READ or WRITE Command Issued	0	AP	CL	0	0	0	BK

where:

UN = Unused

RW = Row Address

CL = Column Address

RK = Rank Address

BK = Bank Address

AP = Auto Precharge bit

11.5.3.4 DDR_GLB_PARAMS

This Read/Write register determines DDR global mode parameters. The value at reset is 0x0000 015A.

Register ID: 0x7

Address Offset: 0x01C

31:12		11	10	9	8			
RSVD	ECC_REPORT_EN	RSVD	DIS_BP	DIS_BP	ODT2_MUX_EN_L			
7	6	5	4	3	2	1	0	
Reserved	CPO	ECC_DET_EN	OPP	Reserved	SES	ORWR	CPM	

Bits	Field Name	Field Description	R/W	Reset
31:12	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000	RO	See Field Desc
11	ECC_REPORT_EN	ECC error reporting enable 0: Disable 1: Enable	R/W	b0
10	Reserved	Reserved	RO	b0
9	DIS_BP	Disable Bypass of empty bank transaction queues 0: bypass is enabled 1: disable bypass	R/W	b0
8	ODT2_MUX_EN_L	If 0: DDR_CFG_REG_[WRITE/READ]_ODT_MASK[17, 12, 7 or 2] is output instead of addr[14] on DDR[A/B/C/D]_MA[14]. Also, DDR_CFG_REG_[WRITE/READ]_ODT_MASK[18, 13, 8 or 3] is output instead of addr[15] on DDR[A/B/C/D]_MA[15].	R/W	b1
7	Reserved	Reserved. Always write 0	R/W	b0
6	CPO	Closed Page Open. 0: Always closes page after READ or WRITE in closed-page mode 1: Allow pages to be left open in closed-page mode if there is a dependent operation	R/W	b1
5	ECC_DETECT_EN	ECC Error Detect Enable. 0: Disable 1: Enable	R/W	b0
4	OPP	Page Policy. 0: Closed Page Policy 1: Open Page Policy	R/W	b1
3	Reserved	Reserved Always program to 1	RO	b1
2	SES	Single-Ended Strobe. 0: Differential double-ended strobe 1: Single-ended strobe	R/W	b0

Bits	Field Name	Field Description	R/W	Reset
1	ORWR	0: do not perform early change. 1: Optimize Rank Write to Read, early change from Read to Write priority on bank with only Write transaction	R/W	b1
0	CPM	Channel Pair Mode. For XLS 100 and 200-series devices, this field affects dynamic receive and dynamic termination properties as shown in the DDR_DYN_DATA_PAD_CTRL register description. 0: Controller operates independently as a 36-bit channel 1: Controller pair operates as 72-bit channel-pair of master & slave	R/W	b0

11.5.4 DDR DLL Control Registers

11.5.4.1 DDR_CLK_CAL_PARAMS

This Read/Write register determines DDR Clock Calibration Parameters. The value at reset is 0x0000 7856.

Register ID: 0x8

Address Offset: 0x020

31:15	14	13	12	11
RSVD	CMD_NXT_DLY_CNT_UPDATE_EN	TX_NXT_DLY_CNT_UPDATE_EN	RX_NXT_DLY_CNT_UPDATE_EN	CLK_TRG_OFF_REF
10	9:7	6:3	2:0	
RSVD	CLK_FN_CAL	CLK_MAX_DLY_STG_MOD	Reserved	

Bits	Field Name	Field Description	R/W	Reset
31:15	Reserved	Reserved Reset value = b0000 0000 0000 0000 0	RO	See Field Desc
14	CMD_NXT_DLY_CNT_UPDATE_EN	Enables update of DDR_[SLV_]CMD_NXT_DLY_CNT registers after completion of a DLL calibration. 0: Disable 1: Enable (default)	R/W	b1
13	TX_NXT_DLY_CNT_UPDATE_EN	Enables update of DDR_DQS[0 - 9]_TX_NXT_DLY_CNT registers after completion of a DLL calibration. 0: Disable 1: Enable (default)	R/W	b1
12	RX_NXT_DLY_CNT_UPDATE_EN	Enables update of DDR_DQS[0 - 9]_RX_NXT_DLY_CNT registers after completion of a DLL calibration. 0: Disable 1: Enable (default)	R/W	b1
11	CLK_TRG_OFF_REF	Enable automatic DLL calibration triggering 128 cycles before the next REFRESH command is issued.	R/W	b1
10	Reserved	Reserved Always program this bit to 0.	RO	b0

Bits	Field Name	Field Description	R/W	Reset
9:7	CLK_FN_CAL	Limits DLL Fine delay range during Calibration: 0, 6, 7: 0/8 to 7/8 of one delay element delay 1: 1/8 to 8/8 of one delay element delay 2: 2/8 to 9/8 of one delay element delay 3: 3/8 to 10/8 of one delay element delay 4: 4/8 to 11/8 of one delay element delay 5: 5/8 to 12/8 of one delay element delay	R/W	b000
6:3	CLK_MAX_DLY_STG_MOD	During clock calibration, bits CLK_MAX_DLY_STG_MOD to 0 of CYC_MEAS_CNT registers are updated, and the rest hold their reset value of 0.	R/W	b101 0
2:0	Reserved	Reserved	RO	b110

11.5.4.2 DDR_CLK_CAL_REQ

This Write Only register requests a DDR Clock re-Calibration sequence. The value after reset is 0x0000 0000 if read.

Register ID: 0x9

Address Offset: 0x024

31:0
DDR_CLK_CAL_REQ

Bits	Field Name	Field Description	R/W	Reset
31:0	DDR_CLK_CAL_REQ	Doing a Write of any value to this address triggers all controller DLL's to re-calibrate. *_DLY_CFG registers are read, and *_CYC_MEAS_CNT and *_NXT_DLY_CNT registers are updated.	WO	0x0000 0000

11.5.4.3 DDR_DQS[0-9]_RX_NXT_DLY_CNT

These ten Read/Write registers determine the next DDR receive byte strobe delay count parameters. The value at reset is 0x0000 0000.

Register IDs: 0x100-0x124 by 4

Register Offsets 0x400 – 0x490 by 0x10

Note: Register offsets depend on the single channel or channel pair being programmed as shown:

Register Name	Register Offset					
	Channel				Channel Pair	
	A	B	C	D	AB	CD
DDR_DQS0_RX_NXT_DLY_CNT	0x400	-	0x400	-	0x400	0x400
DDR_DQS1_RX_NXT_DLY_CNT	0x410	-	0x410	-	0x410	0x410
DDR_DQS2_RX_NXT_DLY_CNT	0x420	-	0x420	-	0x420	0x420
DDR_DQS3_RX_NXT_DLY_CNT	0x430	-	0x430	-	0x430	0x430
DDR_DQS4_RX_NXT_DLY_CNT	-	0x400	-	0x400	0x440	0x440
DDR_DQS5_RX_NXT_DLY_CNT	-	0x410	-	0x410	0x450	0x450
DDR_DQS6_RX_NXT_DLY_CNT	-	0x420	-	0x420	0x460	0x460
DDR_DQS7_RX_NXT_DLY_CNT	-	0x430	-	0x430	0x470	0x470
DDR_DQS8_RX_NXT_DLY_CNT	0x480	-	0x480	-	0x480	0x480
DDR_DQS9_RX_NXT_DLY_CNT	-	R: 0x470 W: 0x480	-	R: 0x470 W: 0x480	0x490	0x490

Register Description

31:11	10:6	5:3	2:0
Reserved	NXT_ROW_DLY_CNT	NXT_COL_DLY_CNT	NXT_FINE_DLY_CNT

Bits	Field Name	Field Description	R/W	Reset
31:11	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0	RO	See Field Desc
10:6	NXT_ROW_DLY_CNT	The DQS RX DLLs will be programmed with this Row Delay Count value during the next update.	R/W	b000 00
5:3	NXT_COL_DLY_CNT	The DQS RX DLLs will be programmed with this Column Delay Count value during the next update.	R/W	b00 0
2:0	NXT_FINE_DLY_CNT	The DQS RX DLLs will be programmed with this Fine Delay Count value during the next update.	R/W	b000

11.5.4.4 DDR_DQS[0-9]_TX_NXT_DLY_CNT

These ten Read/Write registers determine the next DDR transmit byte strobe delay count parameters. The value at reset is 0x0000 0000.

Register IDs: 0x140-0x164 by 4

Register Offsets 0x500 – 0x590 by 0x10

Register offsets depend on the single channel or channel pair being programmed as shown:

Register Name	Register Offset					
	Channel				Channel Pair	
	A	B	C	D	AB	CD
DDR_DQS0_TX_NXT_DLY_CNT	0x500	-	0x500	-	0x500	0x500
DDR_DQS1_TX_NXT_DLY_CNT	0x510	-	0x510	-	0x510	0x510
DDR_DQS2_TX_NXT_DLY_CNT	0x520	-	0x520	-	0x520	0x520
DDR_DQS3_TX_NXT_DLY_CNT	0x530	-	0x530	-	0x530	0x530
DDR_DQS4_TX_NXT_DLY_CNT	-	0x500	-	0x500	0x540	0x540
DDR_DQS5_TX_NXT_DLY_CNT	-	0x510	-	0x510	0x550	0x550
DDR_DQS6_TX_NXT_DLY_CNT	-	0x520	-	0x520	0x560	0x560
DDR_DQS7_TX_NXT_DLY_CNT	-	0x530	-	0x530	0x570	0x570
DDR_DQS8_TX_NXT_DLY_CNT	0x580	-	0x580	-	0x580	0x580
DDR_DQS9_TX_NXT_DLY_CNT	-	R: 0x570 W: 0x580	-	R: 0x570 W: 0x580	0x590	0x590

31:11	10:6	5:3	2:0
Reserved	NXT_ROW_DLY_CNT	NXT_COL_DLY_CNT	NXT_FINE_DLY_CNT

Bits	Field Name	Field Description	R/W	Reset
31:11	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0	RO	See Field Desc
10:6	NXT_ROW_DLY_CNT	The DQS TX DLLs will be programmed with this Row Delay Count value during the next update.	R/W	b000 00
5:3	NXT_COL_DLY_CNT	The DQS TX DLLs will be programmed with this Column Delay Count value during the next update.	R/W	b00 0
2:0	NXT_FINE_DLY_CNT	The DQS TX DLLs will be programmed with this Fine Delay Count value during the next update.	R/W	b000

11.5.4.5 DDR_CMD_NXT_DLY_CNT

This Read/Write register determines the next DDR command delay count parameters. The value at reset is 0x0000 0000.

Register ID: 0x13C
Address Offset: 0x4F0

31:11	10:6	5:3	2:0
Reserved	NXT_ROW_DLY_CNT	NXT_COL_DLY_CNT	NXT_FINE_DLY_CNT

Bits	Field Name	Field Description	R/W	Reset
31:11	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0	RO	See Field Desc
10:6	NXT_ROW_DLY_CNT	The CMD DLL will be programmed with this row Delay value during the next update.	R/W	b000 00
5:3	NXT_COL_DLY_CNT	The CMD DLL will be programmed with this column Delay value during the next update.	R/W	b00 0
2:0	NXT_FINE_DLY_CNT	The CMD DLL will be programmed with this fine Delay value during the next update.	R/W	b000

11.5.4.6 DDR_SLV_CMD_NXT_DLY_CNT

This Read/Write register determines the next DDR slave command delay count parameters. This register is used for Channel Pairs only. The value at reset is 0x0000 0000.

Register ID: 0x138
Address Offset: 0x4E0

31:11	10:6	5:3	2:0
Reserved	NXT_ROW_DLY_CNT	NXT_COL_DLY_CNT	NXT_FINE_DLY_CNT

Bits	Field Name	Field Description	R/W	Reset
31:11	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0	RO	See Field Desc
10:6	NXT_ROW_DLY_CNT	The SLV CMD DLL will be programmed with this row Delay value during the next update.	R/W	b000 00
5:3	NXT_COL_DLY_CNT	The SLV CMD DLL will be programmed with this column Delay value during the next update.	R/W	b00 0
2:0	NXT_FINE_DLY_CNT	The SLV CMD DLL will be programmed with this fine Delay value during the next update.	R/W	b000

11.5.4.7 DDR_DQS[0-9]_RX_DLY_CNT

These ten read-only registers determine the current DDR receive byte strobe delay count parameters. The value at reset is 0x0000 0000.

Register IDs: 0x101-0x125 by 4

Register Offsets 0x404 – 0x494 by 0x10

Register offsets depend on the single channel or channel pair being programmed as shown:

Register Name	Register Offset					
	Channel				Channel Pair	
	A	B	C	D	AB	CD
DDR_DQS0_RX_DLY_CNT	0x404	-	0x404	-	0x404	0x404
DDR_DQS1_RX_DLY_CNT	0x414	-	0x414	-	0x414	0x414
DDR_DQS2_RX_DLY_CNT	0x424	-	0x424	-	0x424	0x424
DDR_DQS3_RX_DLY_CNT	0x434	-	0x434	-	0x434	0x434
DDR_DQS4_RX_DLY_CNT	-	0x404	-	0x404	0x444	0x444
DDR_DQS5_RX_DLY_CNT	-	0x414	-	0x414	0x454	0x454
DDR_DQS6_RX_DLY_CNT	-	0x424	-	0x424	0x464	0x464
DDR_DQS7_RX_DLY_CNT	-	0x434	-	0x434	0x474	0x474
DDR_DQS8_RX_DLY_CNT	0x484	-	0x484	-	0x484	0x484
DDR_DQS9_RX_DLY_CNT	-	R: 0x474 W: 0x484	-	R: 0x474 W: 0x484	0x494	0x494

All of these registers are periodically updated from the corresponding **DDR_DQS[0-9]_RX_NXT_DLY_CNT** register:

1. During states 3 and 4 of the DRAM controller initialization sequence (see [Section 11.4.2, “Memory Controller Initialization Sequence”](#)), and
2. In DDR2 mode whenever a REFRESH command is issued.

31:11		10:6	5:3	2:0
Reserved		ROW_DLY_CNT	COL_DLY_CNT	FINE_DLY_CNT
Bits	Field Name	Field Description		R/W
31:11	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0		RO See Field Desc
10:6	ROW_DLY_CNT	The DQS RX DLLs are currently programmed with this row Delay value		RO b000 00
5:3	COL_DLY_CNT	The DQS RX DLLs are currently programmed with this column Delay value		RO b00 0
2:0	FINE_DLY_CNT	The DQS RX DLLs are currently programmed with this fine Delay value		RO b000

11.5.4.8 DDR_DQS[0-9]_TX_DLY_CNT

These ten read-only registers determine the current DDR transmit byte strobe delay count parameters. The value at reset is 0x0000 0000.

Register IDs: 0x141-0x165 by 4

Register Offsets 0x504 – 0x594 by 0x10

Register offsets depend on the single channel or channel pair being programmed as shown:

Register Name	Register Offset					
	Channel				Channel Pair	
	A	B	C	D	AB	CD
DDR_DQS0_TX_DLY_CNT	0x504	-	0x504	-	0x504	0x504
DDR_DQS1_TX_DLY_CNT	0x514	-	0x514	-	0x514	0x514
DDR_DQS2_TX_DLY_CNT	0x524	-	0x524	-	0x524	0x524
DDR_DQS3_TX_DLY_CNT	0x534	-	0x534	-	0x534	0x534
DDR_DQS4_TX_DLY_CNT	-	0x504	-	0x504	0x544	0x544
DDR_DQS5_TX_DLY_CNT	-	0x514	-	0x514	0x554	0x554
DDR_DQS6_TX_DLY_CNT	-	0x524	-	0x524	0x564	0x564
DDR_DQS7_TX_DLY_CNT	-	0x534	-	0x534	0x574	0x574
DDR_DQS8_TX_DLY_CNT	0x584	-	0x584	-	0x584	0x584
DDR_DQS9_TX_DLY_CNT	-	R: 0x574 W: 0x584	-	R: 0x574 W: 0x584	0x594	0x594

All of these registers are periodically updated from the corresponding **DDR_DQS[0-9]_TX_NXT_DLY_CNT** register:

1. During states 3 and 4 of the DRAM controller initialization sequence (see [Section 11.4.2, “Memory Controller Initialization Sequence”](#)), and
2. In DDR2 mode whenever a REFRESH command is issued.

31:11		10:6	5:3	2:0		
Reserved		ROW_DLY_CNT	COL_DLY_CNT	FINE_DLY_CNT		
Bits	Field Name	Field Description			R/W	Reset
31:11	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0			RO	See Field Desc
10:6	ROW_DLY_CNT	The DQS TX DLLs are currently programmed with this row Delay value			RO	b000 00
5:3	COL_DLY_CNT	The DQS TX DLLs are currently programmed with this column Delay value			RO	b00 0
2:0	FINE_DLY_CNT	The DQS TX DLLs are currently programmed with this fine Delay value			RO	b000

11.5.4.9 DDR_CMD_DLY_CNT

This read-only register determines the current DDR command delay count parameters. The value at reset is 0x0000 0000.

Register ID: 0x13D

Address Offset: 0x4F4

This register is periodically updated from the corresponding [DDR_CMD_NXT_DLY_CNT](#) register:

1. During states 3 and 4 of the DRAM controller initialization sequence (see [Section 11.4.2, "Memory Controller Initialization Sequence"](#)), and
2. In DDR2 mode whenever a REFRESH command is issued.

31:11		10:6	5:3	2:0
<i>Reserved</i>		ROW_DLY_CNT	COL_DLY_CNT	FINE_DLY_CNT
Bits Field Name Field Description R/W Reset				
31:11	<i>Reserved</i>	<i>Reserved</i> Reset value = b0000 0000 0000 0000 0000 0	RO	See Field Desc
10:6	ROW_DLY_CNT	The CMD DLL is currently programmed with this row Delay value	RO	b000 00
5:3	COL_DLY_CNT	The CMD DLL is currently programmed with this column Delay value	RO	b00 0
2:0	FINE_DLY_CNT	The CMD DLL is currently programmed with this fine Delay value	RO	b000

11.5.4.10 DDR_SLV_CMD_DLY_CNT

This read-only register determines the current DDR slave command delay count parameters. The value at reset is 0x0000 0000. This register is used for Channel Pairs only.

Register ID: **0x139**

Address Offset: **0x4E4**

This register is periodically updated from the corresponding [DDR_SLV_CMD_NXT_DLY_CNT](#) register:

1. During states 3 and 4 of the DRAM controller initialization sequence (see [Section 11.4.2, “Memory Controller Initialization Sequence”](#)), and
2. In DDR2 mode whenever a REFRESH command is issued.

31:11		10:6	5:3	2:0		
Reserved		ROW_DLY_CNT	COL_DLY_CNT	FINE_DLY_CNT		
Bits	Field Name	Field Description			R/W	Reset
31:11	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0			RO	See Field Desc
10:6	ROW_DLY_CNT	The SLV CMD DLL is currently programmed with this row Delay value			RO	b000 00
5:3	COL_DLY_CNT	The SLV CMD DLL is currently programmed with this column Delay value			RO	b00 0
2:0	FINE_DLY_CNT	The SLV CMD DLL is currently programmed with this fine Delay value			RO	b000

11.5.4.11 DDR_DQS[0-9]_RX_CYC_MEAS_CNT

These ten read-only registers measure the current DDR clock cycle width, as measured by the receive calibration DLLs. The value at reset is 0x0000 0000.

Register IDs: 0x102-0x126 by 4

Register Offsets 0x408 – 0x498 by 0x10

Register offsets depend on the single channel or channel pair being programmed as shown:

Register Name	Register Offset					
	Channel				Channel Pair	
	A	B	C	D	AB	CD
DDR_DQS0_RX_CYC_MEAS_CNT	0x408	-	0x408	-	0x408	0x408
DDR_DQS1_RX_CYC_MEAS_CNT	0x418	-	0x418	-	0x418	0x418
DDR_DQS2_RX_CYC_MEAS_CNT	0x428	-	0x428	-	0x428	0x428
DDR_DQS3_RX_CYC_MEAS_CNT	0x438	-	0x438	-	0x438	0x438
DDR_DQS4_RX_CYC_MEAS_CNT	-	0x408	-	0x408	0x448	0x448
DDR_DQS5_RX_CYC_MEAS_CNT	-	0x418	-	0x418	0x458	0x458
DDR_DQS6_RX_CYC_MEAS_CNT	-	0x428	-	0x428	0x468	0x468
DDR_DQS7_RX_CYC_MEAS_CNT	-	0x438	-	0x438	0x478	0x478
DDR_DQS8_RX_CYC_MEAS_CNT	0x488	-	0x488	-	0x488	0x488
DDR_DQS9_RX_CYC_MEAS_CNT	-	R: 0x478 W: 0x488	-	R: 0x478 W: 0x488	0x498	0x498

31:11	10:6	5:3	2:0
Reserved	ROW_CYC_MEAS_CNT	COL_CYC_MEAS_CNT	FINE_CYC_MEAS_CNT

Bits	Field Name	Field Description	R/W	Reset
31:11	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0	RO	See Field Desc
10:6	ROW_CYC_MEAS_CNT	Row delay value of the DQS RX calibration DLL's result	RO	b000 00
5:3	COL_CYC_MEAS_CNT	Column delay value of the DQS RX calibration DLL's result	RO	b00 0
2:0	FINE_CYC_MEAS_CNT	Fine delay value of the DQS RX calibration DLL's result	RO	b000

11.5.4.12 DDR_DQS[0-9]_TX_CYC_MEAS_CNT

These ten read-only registers measure the current DDR clock cycle width, as measured by the transmit calibration DLLs. The value at reset is 0x0000 0000.

Register IDs: 0x142-0x166 by 4

Register Offsets 0x508 – 0x598 by 0x10

Register offsets depend on the single channel or channel pair being programmed as shown:

Register Name	Register Offset					
	Channel				Channel Pair	
	A	B	C	D	AB	CD
DDR_DQS0_TX_CYC_MEAS_CNT	0x508	-	0x508	-	0x508	0x508
DDR_DQS1_TX_CYC_MEAS_CNT	0x518	-	0x518	-	0x518	0x518
DDR_DQS2_TX_CYC_MEAS_CNT	0x528	-	0x528	-	0x528	0x528
DDR_DQS3_TX_CYC_MEAS_CNT	0x538	-	0x538	-	0x538	0x538
DDR_DQS4_TX_CYC_MEAS_CNT	-	0x508	-	0x508	0x548	0x548
DDR_DQS5_TX_CYC_MEAS_CNT	-	0x518	-	0x518	0x558	0x558
DDR_DQS6_TX_CYC_MEAS_CNT	-	0x528	-	0x528	0x568	0x568
DDR_DQS7_TX_CYC_MEAS_CNT	-	0x538	-	0x538	0x578	0x578
DDR_DQS8_TX_CYC_MEAS_CNT	0x588	-	0x588	-	0x588	0x588
DDR_DQS9_TX_CYC_MEAS_CNT	-	R: 0x578 W: 0x588	-	R: 0x578 W: 0x588	0x598	0x598

31:11	10:6	5:3	2:0
Reserved	ROW_CYC_MEAS_CNT	COL_CYC_MEAS_CNT	FINE_CYC_MEAS_CNT

Bits	Field Name	Field Description	R/W	Reset
31:11	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0	RO	See Field Desc
10:6	ROW_CYC_MEAS_CNT	Row delay value of the DQS TX calibration DLL's result	RO	b000 00
5:3	COL_CYC_MEAS_CNT	Column delay value of the DQS TX calibration DLL's result	RO	b00 0
2:0	FINE_CYC_MEAS_CNT	Fine delay value of the DQS TX calibration DLL's result	RO	b000

11.5.4.13 DDR_CMD_CYC_MEAS_CNT

This read-only register determines the current clock cycle width, as measured by the command calibration DLL. The value at reset is 0x0000 0000.

Register ID: **0x13E**

Address Offset: **0x4F8**

31:11	10:6	5:3	2:0
Reserved	ROW_CYC_MEAS_CNT	COL_CYC_MEAS_CNT	FINE_CYC_MEAS_CNT

Bits	Field Name	Field Description	R/W	Reset
31:11	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0	RO	See Field Desc
10:6	ROW_CYC_MEAS_CNT	Row delay value of the CMD calibration DLL's result	RO	b000 00
5:3	COL_CYC_MEAS_CNT	Column delay value of the CMD calibration DLL's result	RO	b00 0
2:0	FINE_CYC_MEAS_CNT	Fine delay value of the CMD calibration DLL's result	RO	b000

11.5.4.14 DDR_SLV_CMD_CYC_MEAS_CNT

This read-only register determines the current clock cycle width, as measured by the slave controller's command calibration DLL. The value at reset is 0x0000 0000. This register is used for Channel Pairs only.

Register ID: **0x13A**

Address Offset: **0x4E8**

31:11	10:6	5:3	2:0
Reserved	ROW_CYC_MEAS_CNT	COL_CYC_MEAS_CNT	FINE_CYC_MEAS_CNT

Bits	Field Name	Field Description	R/W	Reset
31:11	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0	RO	See Field Desc
10:6	ROW_CYC_MEAS_CNT	Row delay value of the SLV CMD calibration DLL's result	RO	b000 00
5:3	COL_CYC_MEAS_CNT	Column delay value of the SLV CMD calibration DLL's result	RO	b00 0
2:0	FINE_CYC_MEAS_CNT	Fine delay value of the SLV CMD calibration DLL's result	RO	b000

11.5.4.15 DDR_DQS[0-9]_RX_DLY_CFG

These ten Read/Write registers configure the DDR receive byte strobe phase-shift and strobe gating parameters. The value at reset is 0x0000 0020.

Register IDs: 0x103-0x127 by 4

Register Offsets 0x40C – 0x49C

Register offsets depend on the single channel or channel pair being programmed as shown:

Register Name	Register Offset					
	Channel				Channel Pair	
	A	B	C	D	AB	CD
DDR_DQS0_RX_DLY_CFG	0x40C	-	0x40C	-	0x40C	0x40C
DDR_DQS1_RX_DLY_CFG	0x41C	-	0x41C	-	0x41C	0x41C
DDR_DQS2_RX_DLY_CFG	0x42C	-	0x42C	-	0x42C	0x42C
DDR_DQS3_RX_DLY_CFG	0x43C	-	0x43C	-	0x43C	0x43C
DDR_DQS4_RX_DLY_CFG	-	0x40C	-	0x40C	0x44C	0x44C
DDR_DQS5_RX_DLY_CFG	-	0x41C	-	0x41C	0x45C	0x45C
DDR_DQS6_RX_DLY_CFG	-	0x42C	-	0x42C	0x46C	0x46C
DDR_DQS7_RX_DLY_CFG	-	0x43C	-	0x43C	0x47C	0x47C
DDR_DQS8_RX_DLY_CFG	0x48C	-	0x48C	-	0x48C	0x48C
DDR_DQS9_RX_DLY_CFG	-	R: 0x47C W: 0x48C	-	R: 0x47C W: 0x48C	0x49C	0x49C

31:11	10	9	8	7	6:0
Reserved	GEN_CYC_LATE	GEN_HALF_LATE	GEN_CYC_NOMINAL	GEN_HALF_EARLY	DLY_CFG

Bits	Field Name	Field Description	R/W	Reset
31:11	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0	RO	See Field Desc
10	GEN_CYC_LATE	Generate a one-cycle-wide Read_En pulse that starts at TRL + 1 and ends at TRL + 2 where TRL is the value of DDR_PARAMS_3.TRL	R/W	b0
9	GEN_HALF_LATE	Generate a one-cycle-wide Read_En pulse that starts at TRL + 0.5 and ends at TRL + 1.5	R/W	b0
8	GEN_CYC_NOMINAL	Generate a one-cycle-wide Read_En pulse that starts at TRL and ends at TRL + 1	R/W	b0
7	GEN_HALF_EARLY	Generate a one-cycle-wide Read_En pulse that starts at TRL - 0.5 and ends at TRL + 0.5	R/W	b0
6:0	DLY_CFG	DQS_RX_NXT_DLY_CNT = DQS_RX_CYC_MEAS_CNT multiplied by (DLY_CFG/128)	R/W	b010 0000

11.5.4.16 DDR_DQS[0-9]_TX_DLY_CFG

These ten Read/Write registers configure the DDR transmit byte strobe phase-shift parameters. The value at reset is 0x0000 0020.

Register IDs: 0x143-0x167 by 4

Register Offsets 0x50C – 0x59C

Register offsets depend on the single channel or channel pair being programmed as shown:

Register Name	Register Offset					
	Channel				Channel Pair	
	A	B	C	D	AB	CD
DDR_DQS0_TX_DLY_CFG	0x50C	-	0x50C	-	0x50C	0x50C
DDR_DQS1_TX_DLY_CFG	0x51C	-	0x51C	-	0x51C	0x51C
DDR_DQS2_TX_DLY_CFG	0x52C	-	0x52C	-	0x52C	0x52C
DDR_DQS3_TX_DLY_CFG	0x53C	-	0x53C	-	0x53C	0x53C
DDR_DQS4_TX_DLY_CFG	-	0x50C	-	0x50C	0x54C	0x54C
DDR_DQS5_TX_DLY_CFG	-	0x51C	-	0x51C	0x55C	0x55C
DDR_DQS6_TX_DLY_CFG	-	0x52C	-	0x52C	0x56C	0x56C
DDR_DQS7_TX_DLY_CFG	-	0x53C	-	0x53C	0x57C	0x57C
DDR_DQS8_TX_DLY_CFG	0x58C	-	0x58C	-	0x58C	0x58C
DDR_DQS9_TX_DLY_CFG	-	R: 0x57C W: 0x58C	-	R: 0x57C W: 0x58C	0x59C	0x59C

31:7	6:0
Reserved	DLY_CFG

Bits	Field Name	Field Description	R/W	Reset
31:7	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0000 0	RO	See Field Desc
6:0	DLY_CFG	DQS_TX_NXT_DLY_CNT = DQS_TX_CYC_MEAS_CNT multiplied by (DLY_CFG/128)	R/W	b010 0000

11.5.4.17 DDR_CMD_DLY_CFG

This Read/Write register determines the DDR command delay phase-shift parameters. The value at reset is 0x0000 0040.

Register ID: 0x13F
Address Offset: 0x4FC

31:7	Reserved	6:0
<i>Reserved</i>		DLY_CFG
Bits	Field Name	Field Description
31:7	Reserved	<i>Reserved</i> Reset value = b0000 0000 0000 0000 0000 0000 0
6:0	DLY_CFG	CMD_NXT_DLY_CNT = CMD_CYC_MEAS_CNT multiplied by (DLY_CFG/128)

11.5.4.18 DDR_SLV_CMD_DLY_CFG

This Read/Write register determines the DDR slave command delay phase-shift parameters. The value at reset is 0x0000 0040. This register is used for Channel Pairs only.

Register ID: 0x13B
Address Offset: 0x4EC

31:7	Reserved	6:0
<i>Reserved</i>		DLY_CFG
Bits	Field Name	Field Description
31:7	Reserved	<i>Reserved</i> Reset value = b0000 0000 0000 0000 0000 0000 0
6:0	DLY_CFG	SLV_CMD_NXT_DLY_CNT = SLV_CMD_CYC_MEAS_CNT multiplied by (DLY_CFG/128)

11.5.5 DDR Thermal Control Registers

11.5.5.1 DDR_RX0_THERM_CTRL

This Read/Write register is common for the two channels within a Channel Pair controller. The value at reset is 0x0000 0001.

See Section 11.4.5 before programming this register.

Read Register ID:	0x130	Write Register ID:	0x134
Read Address Offset:	0x4C0	Write Address Offset:	0x4D0

31:27	26:21	20:15	14:12	11:10	9	8	7	6:1	0
RSV	NCODE	PCODE	RSV	TERM_SEL	RSV	PRESET_P	PRESET_N	PRESET_VALUE	EN_CNT

Bits	Field Name	Field Description	R/W	Reset
31:27	Reserved	Reserved	RO	b00000
26:21	NCODE	Encoded pull-down finger control strength value	RO	b000000
20:15	PCODE	Encoded pull-up finger control strength value	RO	b000000
14:12	Reserved	Reserved	RO	b000
11:10	TERM_SEL	Termination resistance: 00: not valid selection. User must reprogram from the reset value. 01: 75 ohm 10: 150 ohm 11: 50 ohm	R/W	b00
9	Reserved	Reserved	RO	b0
8	PRESET_P	Enables writing of PRESET_VALUE to the PCODE field	R/W	b0
7	PRESET_N	Enables writing of PRESET_VALUE to the NCODE field	R/W	b0
6:1	PRESET_VALUE	Write data for the PCODE and NCODE fields	R/W	b000000
0	EN_CNT	Enables the autocalibration process to update the PCODE/NCODE fields	R/W	b1

11.5.5.2 DDR_RX1_THERM_CTRL

This Read/Write register is common for the two channels within a channel pair controller. The value at reset is 0x0000 0001.

See Section 11.4.5 before programming this register.

Read Register ID: 0x131 **Write Register ID:** 0x135

Read Address Offset: 0x4C4 **Write Address Offset:** 0x4D4

31:27	26:21	20:15	14:9	8	7	6:1	0
RSV	NCODE	PCODE	Reserved	PRESET_P	PRESET_N	PRESET_VALUE	EN_CNT

Bits	Field Name	Field Description	R/W	Reset
31:27	Reserved	Reserved	RO	b00000
26:21	NCODE	Encoded pull-down finger control strength value	RO	b000000
20:15	PCODE	Encoded pull-up finger control strength value	RO	b000000
14:9	Reserved	Reserved	RO	b000000
8	PRESET_P	Enables writing of PRESET_VALUE to the PCODE field	R/W	b0
7	PRESET_N	Enables writing of PRESET_VALUE to the NCODE field	R/W	b0
6:1	PRESET_VALUE	Write data for the PCODE and NCODE fields	R/W	b000000
0	EN_CNT	Enables the autocalibration process to update the PCODE/NCODE fields	R/W	b1

11.5.5.3 DDR_TX0_THERM_CTRL

This Read/Write register is common for the two channels within a channel pair controller. The value at reset is 0x0000 0201.

See Section 11.4.5 before programming this register.

Read Register ID: 0x132 **Write Register ID:** 0x136

Read Address Offset: 0x4C8 **Write Address Offset:** 0x4D8

31:27	26:21	20:15	14	13	12	11:9	8	7	6:1	0
RSV	NCODE	PCODE	RSV	ADDR_DRV_IMP	DATA_DRV_IMP	RSV	PRESET_P	PRESET_N	PRESET_VALUE	EN_CNT

Bits	Field Name	Field Description	R/W	Reset
31:27	Reserved	Reserved	RO	b000000
26:21	NCODE	Encoded pull-down finger control strength value	RO	b0000000
20:15	PCODE	Encoded pull-up finger control strength value	RO	b000000
14	Reserved	Reserved	RO	b0
13:	ADDR_DRV_IMP	Controls address bus driver impedance. 0: 18 ohms 1: 36 ohms	R/W	b0
12	DATA_DRV_IMP	Controls data bus driver impedance. 0: 18 ohms 1: 36 ohms	R/W	b0
11:9	Reserved	Reserved	RO	b001
8	PRESET_P	Enables writing of PRESET_VALUE to the PCODE field.	R/W	b0
7	PRESET_N	Enables writing of PRESET_VALUE to the NCODE field.	R/W	b0
6:1	PRESET_VALUE	Write data for the PCODE and NCODE fields	RO	b000000
0	EN_CNT	Enables the autocalibration process to update the PCODE/NCODE fields	R/W	b1

11.5.5.4 DDR_TX1_THERM_CTRL

This Read/Write register is common for the two channels within a channel pair controller. The value at reset is 0x0000 0001.

See Section 11.4.5 before programming this register.

Read Register ID: 0x133 **Write Register ID:** 0x137

Read Address Offset: 0x4CC **Write Address Offset:** 0x4DC

31:27	26:21	20:15	14:9	8	7	6:1	0
Reserved	NCODE	PCODE	Reserved	PRESET_P	PRESET_N	PRESET_VALUE	EN_CNT

Bits	Field Name	Field Description	R/W	Reset
31:27	Reserved	Reserved	RO	b0000 0
26:21	NCODE	Encoded pull-down finger control strength value	RO	b000 000
20:15	PCODE	Encoded pull-up finger control strength value	RO	b0 0000 0
14:9	Reserved	Reserved	RO	b000 000
8	PRESET_P	Enables writing of PRESET_VALUE to the PCODE field	R/W	b0
7	PRESET_N	Enables writing of PRESET_VALUE to the NCODE field	R/W	b0
6:1	PRESET_VALUE	Write data for the PCODE and NCODE fields	R/W	b000 000
0	EN_CNT	Enables the autocalibration process to update the PCODE/NCODE fields	R/W	b1

11.5.6 DDR MRS Command Registers

11.5.6.1 DDR_CFG_REG_MRS

This register contains data that is delivered to the DRAM chip when an MRS command is issued. The value at reset is 0x0000 0643.

Register ID: 0x14

Address Offset: 0x050

31:13	12	11:9	8	7	6:4	3	2:0
Reserved	PWRDN_EXIT_MODE	WRITE_REC	DLL_RESET	TEST_MODE	CAS_LATENCY	BURST_TYPE	BURST_LENGTH

Bits	Field Name	Field Description	Formula	R/W	Reset
31:13	Reserved	Reserved Reset value = b0000 0000 0000 0000 000		RO	See Field Desc
12	PWRDN_EXIT_MODE	Powerdown Exit Mode: 0: Fast exit 1: Slow exit	For DDR2: chosen exit mode	R/W	b0
11:9	WRITE_REC	Write Recovery: 001: 2 cycles 010: 3 cycles 011: 4 cycles 100: 5 cycles 101: 6 cycles	For DDR2: t_{wr}	R/W	b011
8	DLL_RESET	DLL Reset 1: perform reset		R/W	b0
7	TEST_MODE	DRAM chip Test Mode enable: 0: Disabled 1: Enabled		R/W	b0
6:4	CAS_LATENCY	010: CL = 2 011: CL = 3 100: CL = 4 101: CL = 5 110: CL = 6 111: CL = 7		R/W	b100
3	BURST_TYPE	0: Sequential 1: Interleaved		R/W	b0
2:0	BURST_LENGTH	010: BL = 4 011: BL = 8		R/W	b011

11.5.6.2 DDR_CFG_REG_RDMRS

This register should be programmed with the same values as DDR_CFG_REG_MRS, except that bit [8] should be set. The value at reset is 0x0000 0743.

Register ID: **0x15**

Address Offset: **0x054**

31:13	12	11:9	8	7	6:4	3	2:0
Reserved	PWRDN_EXIT_MODE	WRITE_REC	DLL_RESET	TEST_MODE	CAS_LATENCY	BURST_TYPE	BURST_LENGTH

Bits	Field Name	Field Description	R/W	Reset
31:13	Reserved	Reserved Reset value = b0000 0000 0000 0000 000	RO	See Field Desc
12	PWRDN_EXIT_MODE	Powerdown Exit Mode: 0: Fast exit 1: Slow exit	R/W	b0
11:9	WRITE_REC	Write Recovery: 001: 2 cycles 010: 3 cycles 011: 4 cycles 100: 5 cycles 101: 6 cycles	R/W	b011
8	DLL_RESET	DLL Reset. 1: perform reset	R/W	b1
7	TEST_MODE	DRAM chip Test Mode enable: 0: Disabled 1: Enabled	R/W	b0
6:4	CAS_LATENCY	010: CL = 2 011: CL = 3 100: CL = 4 101: CL = 5 110: CL = 6 111: CL = 7	R/W	b100
3	BURST_TYPE	0: Sequential 1: Interleaved	R/W	b0
2:0	BURST_LENGTH	010: BL = 4 011: BL = 8	R/W	b011

11.5.6.3 DDR_CFG_REG_EMRS1

This register contains data that is delivered to the DRAM chip when an EMRS command is issued. The value at reset is 0x0000 0018.

Register ID: **0x16**

Address Offset: **0x058**

31:13	12	11	10	9:7	6	5:3	2	1	0
Reserved	OUTPUT_EN	RDQS_EN	DQS#_EN	OCD_OP	RTT_MSB	ADD_LAT	RTT_LSB	OUTPUT_DRIVE	DLL_EN

Bits	Field Name	Field Description	R/W	Reset
31:13	Reserved	Reserved Reset value = b0000 0000 0000 0000 000	RO	See Field Desc
12	OUTPUT_EN	Output Enable. 0: Enabled 1: Disabled	R/W	b0
11	RDQS_EN	RDQS Enable. 0: Disabled 1: Enabled	R/W	b0
10	DQS#_EN	DQS# Enable. 0: Enabled 1: Disabled	R/W	b0
9:7	OCD_OP	OCD Operation. 000: OCD Exit 111: OCD Default	R/W	b00 0
6	RTT_MSB	R _{TT} Value. Occupies bits [6 & 2]. DDR_CFG_REG_EMRS1[6],[2] = 00: R _{TT} Disabled 01: 75 Ohm 10: 150 Ohm 11: <i>illegal</i>	R/W	b0
5:3	ADD_LAT	Additive Latency. 0: AL = 0 1: AL = 1 2: AL = 2 3: AL = 3 4: AL = 4	R/W	b01 1
2	RTT_LSB	See RTT_MSB, bit[6] description	R/W	b0
1	OUTPUT_DRIVE	Output Drive strength. 0:100% 1: 50%	R/W	b0
0	DLL_EN	DLL: 0: Enabled 1: Disabled	R/W	b0

11.5.6.4 DDR_CFG_REG_EMRS2

This register contains data that is delivered to the DRAM chip when an EMRS2 command is issued. The value at reset is 0x0000 0000.

Register ID: **0x17**

Address Offset: **0x05C**

31:16	15:0
Reserved	EMRS2_REG

Bits	Field Name	Field Description	R/W	Reset
31:16	Reserved	Reserved	RO	0x0000
15:0	EMRS2_REG	Extended Mode Register 2 values	R/W	0x0000

11.5.6.5 DDR_CFG_REG_EMRS3

This register contains data that is delivered to the DRAM chip when an EMRS3 command is issued. The value at reset is 0x0000 0000.

Register ID: **0x18**

Address Offset: **0x060**

31:16	15:0
Reserved	EMRS3_REG

Bits	Field Name	Field Description	R/W	Reset
31:16	Reserved	Reserved	RO	0x0000
15:0	EMRS3_REG	Extended Mode Register 3 values	R/W	0x0000

11.5.7 DDR Reset Sequence Control Registers

11.5.7.1 DDR_CFG_REG_RESET_TIMERS

This Read/Write register configures the DRAM chip initialization sequence and the user specified command sequence. The value at reset is 0x0000 080D.

Register ID: 0x19

Address Offset: 0x064

31:22	21	20:8	7	6:4	3:0
Reserved	RST_STRT	RST_LEN	CMD_SEQ_RQD	CMD_SEQ_LEN	RST_SEQ_LEN-1

Bits	Field Name	Field Description	R/W	Reset
31:22	Reserved	Reserved	RO	b0000 0000 00
21	RST_STRT	Activate controller and Start Reset sequence	WO	b0
20:8	RST_LEN	Internal Reset signal will be asserted for 16 times this number of clock cycles.	R/W	b0 0000 0000 1000
7	CMD_SEQ_RQD	Begin the user specified Command Sequence. Will be cleared when command sequence is issued.	R/W	b0
6:4	CMD_SEQ_LEN	Length of the User specified Command Sequence. Can be 0,1,2,3 or 4 instructions long. This uses the values in the following registers: DDR_CFG_REG_RESET_CMD0 and DDR_CFG_REG_RESET_CMD1 .	R/W	b000
3:0	RST_SEQ_LEN-1	Number of commands to be issued for the DRAM chip initialization sequence, minus one.	R/W	0xD

11.5.7.2 DDR_CFG_REG_RESET_CMD0

This Read/Write register configures the DRAM chip initialization sequence and the user-specified command sequence. The value at reset is 0x1BAE 002F.

Register ID: 0x1C

Address Offset: 0x070

31:22	21:16	15:6	5:0
RESET_CMD_CNT_1-1	RESET_CMD_1	RESET_CMD_CNT_0-1	RESET_CMD_0

Bits	Field Name	Field Description	R/W	Reset
31:22	RESET_CMD_CNT_1-1	2nd Command Count minus 1	R/W	b0001 1011 10
21:16	RESET_CMD_1	2nd Command in reset sequence. Reset value = NOP command.	R/W	b10 1110
15:6	RESET_CMD_CNT_0-1	1st Command Count minus 1	R/W	b0000 0000 00
5:0	RESET_CMD_0	1st Command in reset sequence. Reset value = PWR_DN_EX command.	R/W	b10 1111

11.5.7.3 DDR_CFG_REG_RESET_CMD1

This Read/Write register configures the DRAM chip initialization sequence and the user-specified command sequence. The value at reset is 0x0022 0025.

Register ID: **0x1D**

Address Offset: **0x074**

31:22	21:16	15:6	5:0
RESET_CMD_CNT_3-1	RESET_CMD_3	RESET_CMD_CNT_2-1	RESET_CMD_2

Bits	Field Name	Field Description	R/W	Reset
31:22	RESET_CMD_CNT_3-1	4th Command Count minus 1	R/W	b0000 0000 00
21:16	RESET_CMD_3	4th Command in reset sequence. Reset value = EMRS2 command.	R/W	b10 0010
15:6	RESET_CMD_CNT_2-1	3rd Command Count minus 1	R/W	b0000 0000 00
5:0	RESET_CMD_2	3rd Command in reset sequence. Reset value = PRECHG_ALL command.	R/W	b10 0101

11.5.7.4 DDR_CFG_REG_RESET_CMD2

This Read/Write register configures the DRAM chip initialization sequence. The value at reset is 0x0001 0023.

Register ID: **0x1E**

Address Offset: **0x078**

31:22	21:16	15:6	5:0
RESET_CMD_CNT_5-1	RESET_CMD_5	RESET_CMD_CNT_4-1	RESET_CMD_4

Bits	Field Name	Field Description	R/W	Reset
31:22	RESET_CMD_CNT_5-1	6th Command Count minus 1	R/W	b0000 0000 00
21:16	RESET_CMD_5	6th Command in reset sequence. Reset value = ENABLE_DLL command.	R/W	b00 0001
15:6	RESET_CMD_CNT_4-1	5th Command Count minus 1	R/W	b0000 0000 00
5:0	RESET_CMD_4	5th Command in reset sequence. Reset value = EMRS3 command.	R/W	b10 0011

11.5.7.5 DDR_CFG_REG_RESET_CMD3

This Read/Write register configures the DRAM chip initialization sequence. The value at reset is 0x0025 0027.

Register ID: **0x1F**

Address Offset: **0x07C**

31:22	21:16	15:6	5:0
RESET_CMD_CNT_7-1	RESET_CMD_7	RESET_CMD_CNT_6-1	RESET_CMD_6

Bits	Field Name	Field Description	R/W	Reset
31:22	RESET_CMD_CNT_7-1	8th Command Count minus 1	R/W	b0000 0000 00
21:16	RESET_CMD_7	8th Command in reset sequence. Reset value = PRECHG_ALL command.	R/W	10 0101
15:6	RESET_CMD_CNT_6-1	7th Command Count minus 1	R/W	b0000 0000 00
5:0	RESET_CMD_6	7th Command in reset sequence. Reset value = RESET_DLL command.	R/W	10 0111

11.5.7.6 DDR_CFG_REG_RESET_CMD4

This Read/Write register configures the DRAM chip initialization sequence. The value at reset is 0x0020 00B2.

Register ID: **0x20**

Address Offset: **0x080**

31:22	21:16	15:6	5:0
RESET_CMD_CNT_9-1	RESET_CMD_9	RESET_CMD_CNT_8-1	RESET_CMD_8

Bits	Field Name	Field Description	R/W	Reset
31:22	RESET_CMD_CNT_9-1	10th Command Count minus 1	R/W	b0000 0000 00
21:16	RESET_CMD_9	10th Command in reset sequence. Reset value = MRS command.	R/W	b10 0000
15:6	RESET_CMD_CNT_8-1	9th Command Count minus 1	R/W	b0000 0000 10
5:0	RESET_CMD_8	9th Command in reset sequence. Reset value = AUTO_REF command.	R/W	11 0010

11.5.7.7 DDR_CFG_REG_RESET_CMD5

This Read/Write register configures the DRAM chip initialization sequence. The value at reset is 0x0003 282E.

Register ID: **0x21**

Address Offset: **0x084**

31:22	21:16	15:6	5:0
RESET_CMD_CNT_11-1	RESET_CMD_11	RESET_CMD_CNT_10-1	RESET_CMD_10

Bits	Field Name	Field Description	R/W	Reset
31:22	RESET_CMD_CNT_11-1	12th Command Count minus 1	R/W	b0000 0000 00
21:16	RESET_CMD_11	12th Command in reset sequence. Reset value = DFLT_OCD command.	R/W	b00 0011
15:6	RESET_CMD_CNT_10-1	11th Command Count minus 1	R/W	b0010 1000 00
5:0	RESET_CMD_10	11th Command in reset sequence. Reset value = NOP command.	R/W	b10 1110

11.5.7.8 DDR_CFG_REG_RESET_CMD6

This Read/Write register configures the DRAM chip initialization sequence. The value at reset is 0x002E 0021.

Register ID: **0x22**

Address Offset: **0x088**

31:22	21:16	15:6	5:0
RESET_CMD_CNT_13-1	RESET_CMD_13	RESET_CMD_CNT_12-1	RESET_CMD_12

Bits	Field Name	Field Description	R/W	Reset
31:22	RESET_CMD_CNT_13-1	14th Command Count minus 1	R/W	b0000 0000 00
21:16	RESET_CMD_13	14th Command in reset sequence. Reset value = NOP command.	R/W	b10 1110
15:6	RESET_CMD_CNT_12-1	13th Command Count minus 1	R/W	b0000 0000 00
5:0	RESET_CMD_12	13th Command in reset sequence. Reset value = EMRS1 command.	R/W	b10 0001

11.5.8 DDR ECC Registers

11.5.8.1 DDR_CFG_REG_RMW_ECC_LOG_LOW

This Read/Write configuration register contains Read-Modify-Write ECC error addresses. The value at reset is 0x0000 0000.

The contents of this register are valid only
if register DDR_CFG_REG_ECC_LOG_HI bit field ECC_ON_RMW =1

Register ID: 0x2C

Address Offset: 0x0B0

31	30:26	25:10	9:0
RSV	ECC_LOG_BANK	ECC_LOG_ROW	ECC_LOG_COL

Bits	Field Name	Field Description	R/W	Reset
31	Reserved	Reserved	RO	b0
30:26	ECC_LOG_BANK	Logged Bank address	R/W	b000 00
25:10	ECC_LOG_ROW	Logged Row address	R/W	b00 0000 0000 0000 00
9:0	ECC_LOG_COL	Logged Column address	R/W	b00 0000 0000

11.5.8.2 DDR_CFG_REG_ECC_LOG_HI

This Read/Write configuration register logs ECC errors. The value at reset is 0x0000 0000.

Register ID: 0x23

Address Offset: 0x08C

31:4	3	2	1	0
Reserved	ECC_ON_RMW	ECC_LOG_OVFL	ECC_LOG_C	ECC_LOG_U

Bits	Field Name	Field Description	R/W	Reset
31:4	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0000	RO	See Field Desc
3	ECC_ON_RMW	ECC error occurred on RMW (read-modify-write) operation. If set, contents of DDR_CFG_REG_RMW_ECC_LOG_LOW are valid	R/W	b0
2	ECC_LOG_OVFL	Overflow Log when more than one ECC error was logged	R/W	b0
1	ECC_LOG_C	Logged Correctable error	R/W	b0
0	ECC_LOG_U	Logged Uncorrectable error	R/W	b0

11.5.8.3 DDR_CFG_REG_ECC_POISON_MASK

This Read/Write configuration register controls ECC poisoned data. The value at reset is 0x0001 1001.

Register ID: 0x24

Address Offset: 0x090

31:19	18	17:16	15:0
Reserved	ECC_PSN_ALL	ECC_PSN_CHNK	ECC_PSN_MSK

Bits	Field Name	Field Description	R/W	Reset
31:19	Reserved	Reserved	RO	b0000 0000 0000 0
18	ECC_PSN_ALL	Poison All data	R/W	b0
17:16	ECC_PSN_CHNK	Poison Chunk select	R/W	b01
15:0	ECC_PSN_MSK	Poison Mask for ECC bits	R/W	0x1001

11.5.8.4 DDR_CFG_REG_POISON_MASK_0

This Read/Write configuration register controls ECC poisoned data. The value at reset is 0x0000 0000.

Register ID: 0x25

Address Offset: 0x094

31:0
PSN_MSK_31_0

Bits	Field Name	Field Description	R/W	Reset
31:0	PSN_MSK_31_0	Poison Mask for data bits 31:0	R/W	0x0000 0000

11.5.8.5 DDR_CFG_REG_POISON_MASK_1

This Read/Write configuration register controls ECC poisoned data. The value at reset is 0x0000 0000.

Register ID: 0x26

Address Offset: 0x098

31:0
PSN_MSK_63_32

Bits	Field Name	Field Description	R/W	Reset
31:0	PSN_MSK_63_32	Poison Mask for data bits 63:32	R/W	0x0000 0000

11.5.8.6 DDR_CFG_REG_POISON_MASK_2

This Read/Write configuration register controls ECC poisoned data. The value at reset is 0x0000 0000.

Register ID: 0x27

Address Offset: 0x09C

31:0				
PSN_MSK_95_64				
Bits	Field Name	Field Description	R/W	Reset
31:0	PSN_MSK_95_64	Poison Mask for data bits 95:64	R/W	0x0000 0000

11.5.8.7 DDR_CFG_REG_POISON_MASK_3

This Read/Write configuration register controls ECC poisoned data. The value at reset is 0x0000 0000.

Register ID: 0x28

Address Offset: 0x0A0

31:0				
PSN_MSK_127_96				
Bits	Field Name	Field Description	R/W	Reset
31:0	PSN_MSK_127_96	Poison Mask for data bits 127:96	R/W	0x0000 0000

11.5.9 DDR ODT Control Registers

11.5.9.1 DDR_CFG_REG_WRITE_ODT_MASK

This Read/Write register configures the Write ODT mask for DDR2 only. The value at reset is 0x0000 0000.

Register ID: 0x2A

Address Offset: 0x0A8

31:19	18:15	14	13:10	9	8:5	4	3:0
Reserved	RANK3_ODT_BITS	rsv	RANK2_ODT_BITS	rsv	RANK1_ODT_BITS	rsv	RANK0_ODT_BITS

Bits	Field Name	Field Description	R/W	Reset
31:19	Reserved	Reserved Reset value = b0000 0000 0000 0	RO	See Field Desc
18	RANK3_ODT_BITS	Applied to DDR(B/D)_ODT1 during Writes to Rank 3.	R/W	b0
17		Applied to DDR(B/D)_ODT0 during Writes to Rank 3.	R/W	b0
16		Applied to DDR(A/C)_ODT1 during Writes to Rank 3.	R/W	b0
15		Applied to DDR(A/C)_ODT0 during Writes to Rank 3.	R/W	b0
14	Reserved	Reserved	RO	b0
13	RANK2_ODT_BITS	Applied to DDR(B/D)_ODT1 during Writes to Rank 2.	R/W	b0
12		Applied to DDR(B/D)_ODT0 during Writes to Rank 2.	R/W	b0
11		Applied to DDR(A/C)_ODT1 during Writes to Rank 2.	R/W	b0
10		Applied to DDR(A/C)_ODT0 during Writes to Rank 2.	R/W	b0
9	Reserved	Reserved	RO	b0
8	RANK1_ODT_BITS	Applied to DDR(B/D)_ODT1 during Writes to Rank 1.	R/W	b0
7		Applied to DDR(B/D)_ODT0 during Writes to Rank 1.	R/W	b0
6		Applied to DDR(A/C)_ODT1 during Writes to Rank 1.	R/W	b0
5		Applied to DDR(A/C)_ODT0 during Writes to Rank 1.	R/W	b0
4	Reserved	Reserved	RO	b0
3	RANK0_ODT_BITS	Applied to DDR(B/D)_ODT1 during Writes to Rank 0.	R/W	b0
2		Applied to DDR(B/D)_ODT0 during Writes to Rank 0.	R/W	b0
1		Applied to DDR(A/C)_ODT1 during Writes to Rank 0.	R/W	b0
0		Applied to DDR(A/C)_ODT0 during Writes to Rank 0.	R/W	b0

11.5.9.2 DDR_CFG_REG_READ_ODT_MASK

This Read/Write register configures the Read ODT mask for DDR2 only. The value at reset is 0x0000 0000.

Register ID: 0x2B

Address Offset: 0x0AC

Register Description

31:19	18:15	14	13:10	9	8:5	4	3:0
Reserved	RANK3_ODT_BITS	rsv	RANK2_ODT_BITS	rsv	RANK1_ODT_BITS	rsv	RANK0_ODT_BITS

Bits	Field Name	Field Description	R/W	Reset
31:19	Reserved	Reserved Reset value = b0000 0000 0000 0	RO	See Field Desc
18	RANK3_ODT_BITS	Applied to DDR(B/D)_ODT1 during Reads from Rank 3.	R/W	b0
17		Applied to DDR(B/D)_ODT0 during Reads from Rank 3.	R/W	b0
16		Applied to DDR(A/C)_ODT1 during Reads from Rank 3.	R/W	b0
15		Applied to DDR(A/C)_ODT0 during Reads from Rank 3.	R/W	b0
14	Reserved	Reserved	RO	b0
13	RANK2_ODT_BITS	Applied to DDR(B/D)_ODT1 during Reads from Rank 2.	R/W	b0
12		Applied to DDR(B/D)_ODT0 during Reads from Rank 2.	R/W	b0
11		Applied to DDR(A/C)_ODT1 during Reads from Rank 2.	R/W	b0
10		Applied to DDR(A/C)_ODT0 during Reads from Rank 2.	R/W	b0
9	Reserved	Reserved	RO	b0
8	RANK1_ODT_BITS	Applied to DDR(B/D)_ODT1 during Reads from Rank 1.	R/W	b0
7		Applied to DDR(B/D)_ODT0 during Reads from Rank 1.	R/W	b0
6		Applied to DDR(A/C)_ODT1 during Reads from Rank 1.	R/W	b0
5		Applied to DDR(A/C)_ODT0 during Reads from Rank 1.	R/W	b0
4	Reserved	Reserved	RO	b0
3	RANK0_ODT_BITS	Applied to DDR(B/D)_ODT1 during Reads from Rank 0.	R/W	b0
2		Applied to DDR(B/D)_ODT0 during Reads from Rank 0.	R/W	b0
1		Applied to DDR(A/C)_ODT1 during Reads from Rank 0.	R/W	b0
0		Applied to DDR(A/C)_ODT0 during Reads from Rank 0.	R/W	b0

11.5.9.3 DDR_DYN_DATA_PAD_CTRL

This read/write register configures the controller-side termination and receiver timing. The value at reset is 0x6031 6041. This register is only present in the XLS 100 and 200 series devices.

Register ID: 0x30

Address Offset: 0x0C0

Register Description

31:28	27:25	24:20	19:17	16	15:12	11:9	8:4	3:1	0
RCV_DURATION	Res	RCV_TRIG	Res	DYN_RCV_EN	TERM_DURATION	Res	TERM_TRIG	Res	DYN_TERM_EN
Bits Field Name Field Description R/W Reset									
31:28	RCV_DURATION		Number of clock cycles that receiver circuitry should remain enabled. Recommendation: program as BL/2 + 5						R/W 6
27:25	Reserved		Reserved						RO 0
24:20	RCV_TRIG		Receive Trigger. Number of clock cycles prior to first rising edge of earliest-possible read DQS strobe that receiver circuitry should be switched on. Legal values are 4 to DDR_PARAMS_3.TRL . Recommendation: always program as 4.						R/W 3
19:17	Reserved		Reserved						RO 0
16	DYN_RCV_EN		Used with the CPM field in the DDR_GLB_PARAMS register to select the algorithm that activates XLS-side receiver circuitry for the channel or channel-pair's DQ, CB, and DQS pads. See Table 11-14 below for behavior.						R/W 1
15:12	TERM_DURATION		Number of clock cycles that termination should remain enabled. Recommendation: program as BL/2 + 5.						R/W 6
11:9	Reserved		Reserved						RO 0
8:4	TERM_TRIG		Termination Trigger. Number of clock cycles prior to first rising edge of earliest possible read DQS strobe where termination should be switched on. Legal values are 5 to DDR_PARAMS_3.TRL . Recommendation: always program as 5.						R/W 4
3:1	Reserved		Reserved						RO 0
0	DYN_TERM_EN		Dynamic Termination Enable. Used with DDR_PARAMS_4.C_T_EN and DDR_GLB_PARAMS.CPM fields to select the algorithm that activates XLS-side termination for the channel or channel-pair's DQ, CB, and DQS pads. See Table 11-15 below for termination behavior.						R/W 1

Table 11-14. Dynamic Receive Enable Options

DYN_RCV_EN	DDR_GLB_PARAMS.CPM	DDRA_DQ[31:0]	DDRA_CB[3:0]	DDRA_DQS[4:0] P/N	DDRB_DQ[31:0]	DDRB_CB[3:0]	DDRB_DQS[3:0] P/N
0	x	On	On	On	On	On	On
1	1	On only when read data/strobe expected (timing governed by other fields in the DDR_DYN_DATA_PAD_CTRL register).					
1	0	On only when read data/strobe expected		Off		Off	

Table 11-15. Dynamic Termination Enable Options

DYN_TERM_EN	DDR_PARAMS_4.C_T_EN	DDR_GLB_PARAMS.CPM	DDRA_DQ[31:0]	DDRA_CB[3:0]	DDRA_DQS[4:0] P/N	DDRB_DQ[31:0]	DDRB_CB[3:0]	DDRB_DQS[3:0] P/N
0	0	x	Off	Off	Off	Off	Off	Off
0	1	x			On when data/strobe not being driven			
1	x	1			On only when read data/strobe expected (timing governed by other fields in the DDR_DYN_DATA_PAD_CTRL register).			
1	x	0			On only when read data/strobe expected		Off	Off

11.5.10 Performance Measuring Registers

The XLS DRAM Controller Unit has a built in performance/utilization measuring capability which allows users to probe the interface efficiency and load. In order to profile the interface's performance, the user can monitor DRAM transactions (read, write, active, etc.) during a fixed period of time, compared against the interface clock. The DRAM interface has added latency for accessing the memory cells (such as command time, precharge, etc.), implementation latency, and other factors that limit utilization. For the XLS environment, a 50% rule of thumb is used for high memory utilization (such as excessive memory accesses).

Use the following method to measure the performance:

1. The free runner counter should be armed with the DRAM clock, that is counting every clock cycle.
2. Arm the specific event to count such as active, read, write, etc.
3. Enable the counters. For accurate measurements, make sure the enable operation takes place simultaneously for both counters. Using an additional counter (either internal COP0 timers or XLS peripheral timers), make a measurement window of 1sec, then disable the counters.

The interface utilization will be a product of event_count/free_count as follows:

$$\text{utilization} = \frac{\text{event_count}}{\text{free_count}} \times 100\%$$

11.5.10.1 DDR_COUNTER_CONF

This Read/Write register configures the Performance Counters. The value at reset is 0x75C0EB80.

Register ID: 0x2D
Address Offset: 0x0B4

31	30:27	26:21	20	19:16
rsv	CNT1_BUS_CMD	CNT1_CMD	CNT1_EN	CNT1_COND
	15:12	11:6	5	4
	CNT0_BUS_CMD	CNT0_CMD	Reserved	CNT0_COND

Bits	Field Name	Field Description	R/W	Reset
31	Reserved	Reserved	RO	b0
30:27	CNT1_BUS_CMD	Bus Command to match for Counter 1. Same encodings as CNT0_BUS_CMD; see [15:12].	R/W	b111 0
26:21	CNT1_CMD	Command to match for Counter 1. Same encodings as CNT0_CMD; see [11:6].	R/W	b101 110
20	CNT1_EN	Enable Counter 1 1: enable	R/W	b0
19:16	CNT1_COND	Condition for Counter 1 increment. Same encoding as CNT0_COND; see [3:0].	R/W	0x0

Bits	Field Name	Field Description	R/W	Reset
15:12	CNT0_BUS_CMD	Bus Command sent to DRAM device to match for Counter 0: 0x0: MRS 0x2: AUTO_REF 0x4: PRECHARGE 0x5: PRECHARGE_ALL 0x6: ACTIVE 0x8: WRITE 0x9: WRITE_PRE 0xA: READ 0xB: READ_PRE 0xE: NOP	R/W	0xE
11:6	CNT0_CMD	Internal controller command to match for Counter 0. 0x01: ENABLE_DLL 0x02: SELF_REF_ENT 0x03: DEFAULT_OCD 0x0E: PWR_DN_ENT 0x20: MRS 0x21: EMRS1 0x22: EMRS2 0x23: EMRS3 0x24: PRECHARGE 0x25: PRECHARGE_ALL 0x26: ACTIVE 0x27: RESET_DLL 0x28: WRITE 0x29: WRITE_PRE 0x2A: READ 0x2B: READ_PRE 0x2C: WRITE_DM 0x2D: WRITE_PRE_DM 0x2E: NOP 0x2F: PWR_DN_EX 0x30: DESEL 0x32: AUTO_REF 0x3E: SELF_REF_EX	R/W	b1011 10
5	Reserved	Reserved	R/W	b0
4	CNT0_EN	Enable Counter 0 1: enable	R/W	b0
3:0	CNT0_COND	Condition for Counter 0 increment. 0x0: Never increment 0x1: Always increment 0x2: Bank queue empty 0x3: Read enable 0x4: Write enable 0x5: Read enable or Write enable 0x8: Command match 0x9: Command mismatch 0xA: Bus command match 0xB: Bus command mismatch	R/W	0x0

11.5.10.2 DDR_PRF_COUNTER0

This Read/Write register provides the Performance Counter 0 value. The value at reset is 0x0000 0000.

Register ID: 0x2E
Address Offset: 0x0B8

31:0	
COUNTER0_VALUE	

Bits	Field Name	Field Description	R/W	Reset
31:0	COUNTER0_VALUE	Performance Counter 0	R/W	0x0000 0000

11.5.10.3 DDR_PRF_COUNTER1

This Read/Write register provides the Performance Counter 1 value. The value at reset is 0x0000 0000.

Register ID: 0x2F
Address Offset: 0x0BC

31:0	
COUNTER1_VALUE	

Bits	Field Name	Field Description	R/W	Reset
31:0	COUNTER1_VALUE	Performance Counter 1	R/W	0x0000 0000

NETLOGIC
CONFIDENTIAL



Chapter 12 Fast Messaging Network

12.1 Introduction

The Fast Messaging Network (FMN) is a high-performance, low-latency messaging network that connects key system elements. It connects:

- the CPU cores
- Network Accelerators
- DMA Engine
- Security Accelerator Engine
- Compression/Decompression Engine
- PCIe

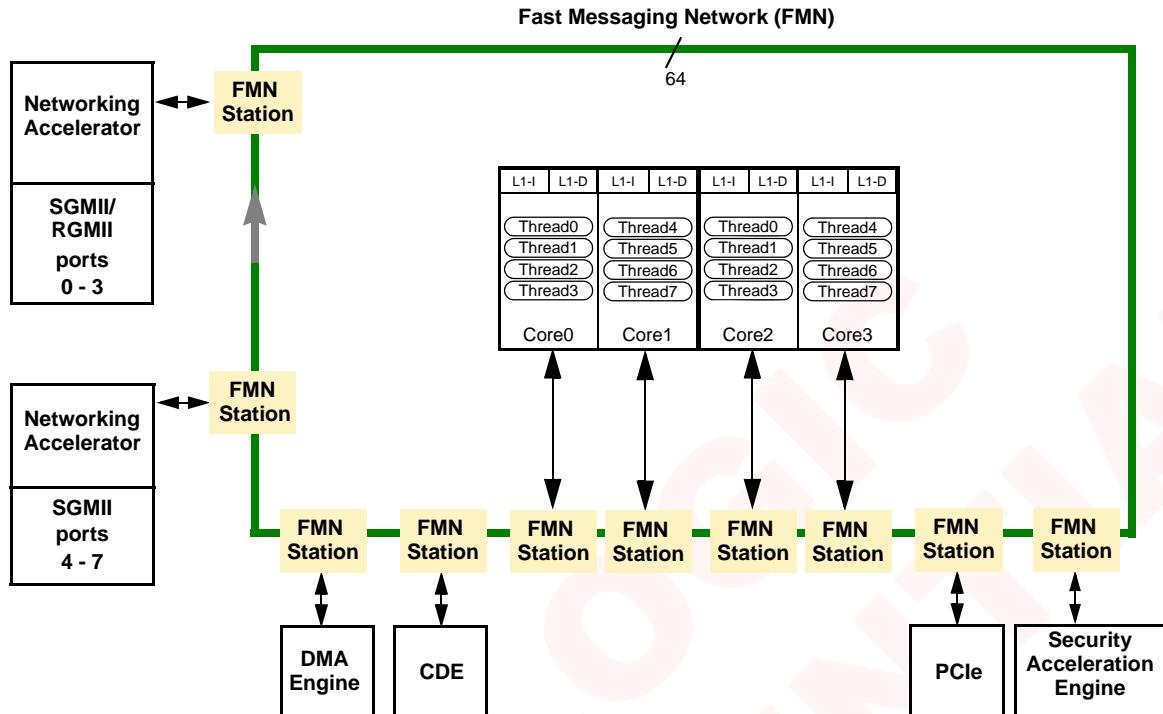
Network interfaces and software communicate about packets via messages sent on the FMN. Once configured, the FMN automatically manages message bandwidth with no software overhead.

The FMN passes packet information without consuming interconnect bandwidth, and without using inter-processor communication, spin-locks, or semaphores. Messages about packets contain *pointers* to packets rather than the packet data itself. Packet data referenced by such messages are concurrently transferred to memory via the Distributed Memory Interconnect.

The FMN ensures fairness among its stations by using a credit-based, round-robin scheme for placing messages onto the FMN. This scheme guarantees that in-transit messages arrive quickly at their destination and ensures that stations are not starved from access to the FMN.

Additional information:

See also: [Section 4.3.2.5 MsgConfig](#)

Figure 12-1. Fast Messaging Network Stations – XLS616 Processor Example

While the FMN is designed primarily for fast packet movement, it can be used for any other purpose by defining the syntax and semantics of the message container. The FMN is designed for point-to-point communications.

The FMN controls message traffic among all senders and receivers to ensure that message recipients are not over-subscribed. It does so using a credit-based scheme for placing messages on the FMN. This scheme guarantees that in-transit messages arrive quickly at their destination, and ensures that none are starved for access to the FMN.

The FMN is designed for point-to-point communications. The network interfaces represent packet data in FMN messages using Packet Descriptor format that they all support. On the other hand, the FMN is a general-purpose message-passing system, and supports messages of arbitrary semantics among Cores.

12.1.1

Fast Messaging Network Architecture

The FMN is a 64-bit wide, low-latency, unidirectional, ring-based message network running at 1/2 core clock frequency. As shown in [Figure 12-1](#), the key functional blocks of the XLS Processor are connected to the Fast Messaging Network. The connection points between the FMN and these functional blocks are called *Stations*. The Cores, networking interfaces, Security Acceleration Engine and the DMA Engine all have Stations on the FMN. The Stations on the FMN are shown in [Figure 12-1](#).

12.1.2

Example Usage of the FMN

The Networking Accelerators use the FMN to communicate information about packets. [Figure 12-2](#) shows an example of how the FMN can be used to process and forward a packet through the XLS processor. A packet is first received via a network interface such as an SGMII or

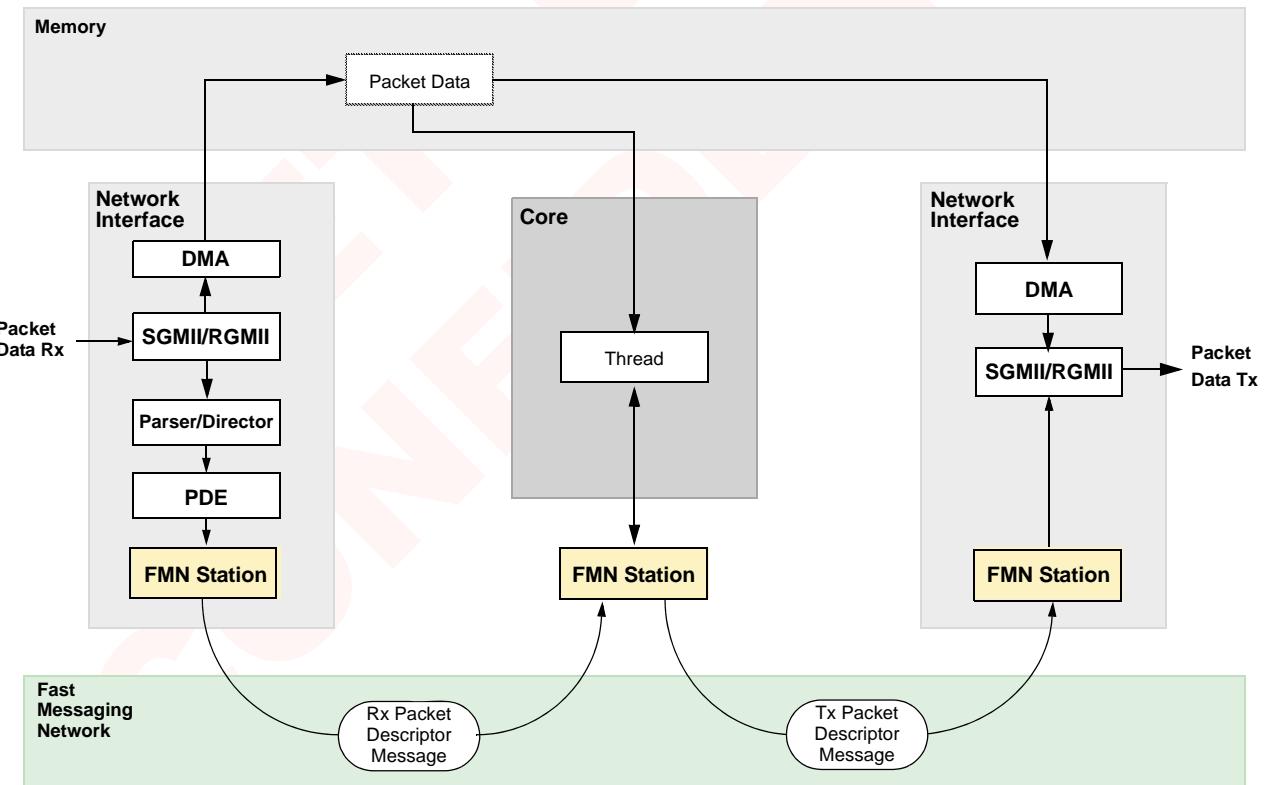
RGMII interface. Packet data is stored in memory as it arrives. Simultaneously, the interface parses and classifies the arriving packet so as to determine which Thread should receive a message about the packet's arrival. When classification and storage are complete, the network interface sends a message in Rx Packet Descriptor format via the FMN to a designated Station on the FMN. Upon receipt of the message, the Station signals the appropriate Thread to respond to the message. Core Stations queue messages and Threads. They may selectively read them by either polling, using `MsgWait`, or interrupt notification.

Messages about packet data are sent using a predefined Packet Descriptor message format. FMN messages in Packet Descriptor format include a collection of pointers to packet data that have been stored in memory. Other fields in the Packet Descriptors provide information about the originator, and other statistics. See [Section 13.3.6, "Packet Descriptor Definitions"](#) for a full description.

The Thread designated to receive the Packet Descriptor message can examine the packet data via the pointers in the message. The result of the classification performed by the network interface may also be included with the packet data.

If the Thread determines that the packet is to be forwarded, it makes any required modifications, creates a message in Tx Packet Descriptor format containing the packet data pointers, and sends the message to a Bucket assigned to the transmitting network interface. The network interface receiving the Tx Packet Descriptor message then serializes the message on its output port.

Figure 12-2. Fast Messaging Network Packet Processing Example



Though the networking interfaces are designed to send and receive messages in Packet Descriptor format, it is important to note that message semantics on the FMN are not fixed. Threads can send and receive messages among themselves that have arbitrary format and significance.

12.1.3

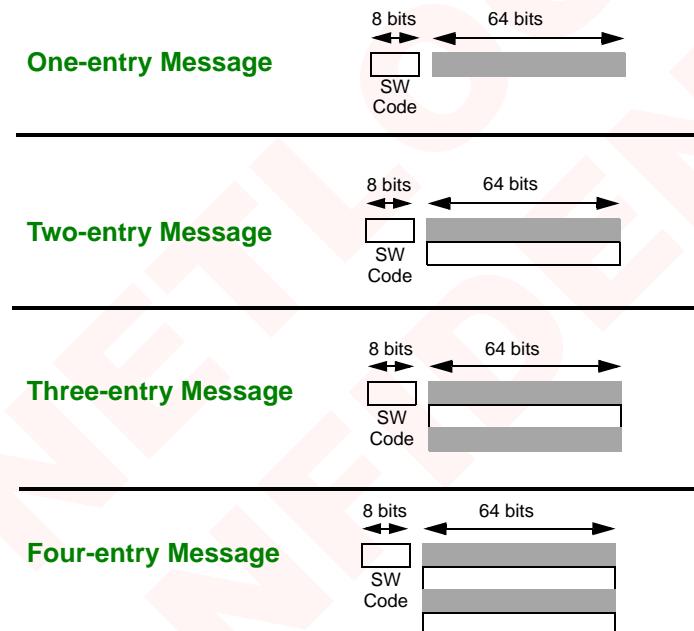
Message Size

As shown in [Figure 12-3](#), message size ranges from one to four 64-bit quantities. Each 64-bit quantity occupies one Receive Queue Entry. The FMN transmits one 64-bit Entry between each Station on every four clocks. The FMN guarantees that each message regardless of size is transmitted atomically.

The 8-bit Software Code field is an additional user-definable field that can be sent with a message between Cores only.

Message semantics are not fixed by the FMN architecture, however, networking blocks use a predefined Packet Descriptor message format for packet movement and processing. Packet Descriptors contain pointers to packet data captured by a network interface. The network interface stores packet data in global memory via the Distributed Interconnect, and notifies the recipient Thread with a message in Packet Descriptor format. Messages in Packet Descriptor format range in size from 1 to 16 Entries. The Packet Descriptor message formats are described in [Section 13.3.6, “Packet Descriptor Definitions”](#).

Figure 12-3. Message Sizes



12.1.4

Stations, Entries, Receive Queues and Buckets

As shown in [Figure 12-1](#), there are FMN Stations for all Cores, network interfaces, DMA Engine, Compression/Decompression Engine, the Security Acceleration Engine, and PCI Express interface. Each Station contains logic to send and receive messages on the FMN.

As shown in [Figure 12-4](#), each Core Station contains a Receive Queue that holds messages directed to that Station by the FMN. Each Receive Queue contains 256 Entries. Each Entry is a 64-bit quantity.

To provide messaging flexibility, each Station's Receive Queue Entries are partitioned into programmable-sized Buckets. Each Bucket has a unique ID, so any Station can send messages to any Bucket. These Buckets are like addressable post boxes in a post office.

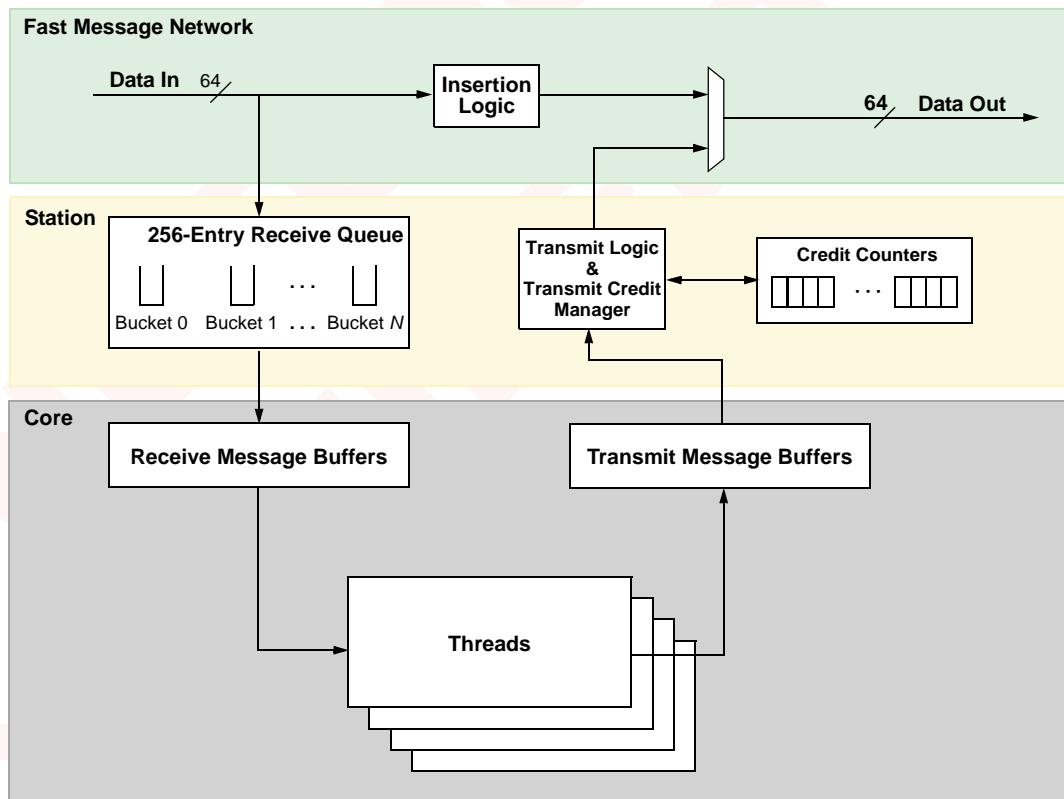
The XLS Processor provides flexible ways for Threads to be notified of the arrival of messages in Buckets. A particular Thread can be notified, or a set of Threads can be notified in round-robin fashion. Any Thread can monitor any Bucket, or multiple Buckets.

The depth of Buckets in each Station are configurable:

- Software may allocate all Buckets in a Station to have the same number of Receive Queue Entries, or
- One Bucket may use all Receive Queue Entries in a Station, or
- Buckets in a Station may be of varying sizes. (See [Section 12.4.1, “Setting Up Buckets”](#) for rules governing Bucket sizes.)

The FMN architecture specifies the number of Buckets in each Station, as shown in [Table 12-1](#). The number of Buckets actually used is programmable, up to the maximum available for each Station. Unused Buckets are disabled by assigning a size of zero. For example, Core0 Station has eight Buckets. If all but one Bucket had a size of 0, the remaining Bucket could use all 256 Entries of that Station’s Receive Queue. The sum of the Bucket sizes for each Station must not exceed the number of its Receive Queue Entries. So if Core0’s Bucket sizes were all non-zero, their sum would have to be less than or equal to 256.

Figure 12-4. Message Station



[Table 12-1](#) shows the Bucket IDs assigned for each Station. The right-most column of [Table 12-1](#) shows the default Bucket sizes of all Stations after reset. At reset, Buckets are allocated to contain the same number of Receive Queue Entries.

Table 12-1. Stations and Addressable Buckets on the Fast Messaging Network

Station	Number of Stations	FMN Station IDs (decimal)	Maximum Buckets per Station	Default Bucket Size (Number of Entries per Bucket)	Description
Core 0	1	0-7	8	32	Each Core contains a Station with 8 Buckets.
Core 1	1	8-15	8	32	Each Core contains a Station with 8 Buckets.
Core 2	1	16-23	8	32	Each Core contains a Station with 8 Buckets.
Core 3	1	24-31	8	32	Each Core contains a Station with 8 Buckets.
SGMII / RGMII or XAUI	2	GMAC Quad_1 or XAUI_1 80 through 87 GMAC Quad_0 or XAUI_0 96 through 103;	4+2	32	Each of the eight GMACs can have one Bucket each, plus two Buckets for Free Buffer Pointers. This can be used, for example, in Virtual MIPS mode, where each of two CPUs can have their own setup for Buffer pointers. (Station IDs 80 through 87 are Reserved in XLS2xx) (Station IDs 80 through 87, and 102 through 103 are Reserved in XLS1xx devices.)
DMA	1	ch 0: 104 ch 1: 105 ch 2: 106 ch 3: 107	4	64	The DMA Station supports partitioning of DMA channels. For example, a given core can have its own DMA channel.
Compression/Decompression Engine	1	CMP0: 108 CMP1: 109	4	128	(These Station IDs are Reserved in XLS2xx and XLS1xx devices.)
PCI Express	1	XLS408-Lite/ 404-Lite PCIE TX0: 116 PCIE RX0: 117 PCIE TX1: 118 PCIE RX1: 119	4	64	XLS408-Lite and XLS404-Lite devices have four message buckets (two Tx, two Rx) to support two individual by-1 PCIe ports TX0, RX0 are used to support a single by-4 PCIe port. (These Station IDs are Reserved in XLS6xx, XLS4xx, XLS2xx and XLS1xx devices.)
	1	XLS6xx, XLS4xx, XLS208/204, XLS108/104: PCIE TX0: 64 PCIE RX0: 65 PCIE TX1: 66 PCIE RX1: 67 PCIE TX2: 68 PCIE RX2: 69 PCIE TX3: 70 PCIE RX3: 71	8	32	XLS616/608, XLS416/408/404, and XLS208/204 devices have eight message buckets (four Tx, four Rx) to support four individual by-1 PCIe ports. XLS108 and XLS104 devices have four message buckets (two Tx, two Rx) to support two by-1 PCIe ports. TX0, RX0 are used to support a single by-4 PCIe port for the XLS6xx and XLS4xx devices.
Security Acceleration Engine	1	SAE0: 120 SAE1: 121	2	128	One security core and one RSA core can have one Bucket each.
Reserved ^a		122-127			Reserved

a. Messages to these Buckets are invalid but will not trigger the invalid Bucket ID condition. They are unused, but technically are part of the Security Acceleration Engine.

Note: The messaging functions within the Cores are accessed through the COP2 space; thus, access to messaging operations requires COP2 access to be enabled.

Particular Bucket IDs in non-Core Stations are dedicated for particular functions. Details of these Buckets can be found in the chapters covering each interface.

12.1.4.1

Allocating Receive Queue Entries to Buckets

The sizes of Buckets allocated in each FMN Station should reflect the expected flow of messages in the system. The principal resource to be allocated is Bucket size. Buckets are disabled by setting their size to zero. Buckets are FIFOs that queue multiple messages. Therefore, the size of active Buckets should be adjusted to match the worst-case processing latency in the Threads. The aggregate size of all Buckets in a Station must not exceed the total number of Receive Queue Entries for that Station.

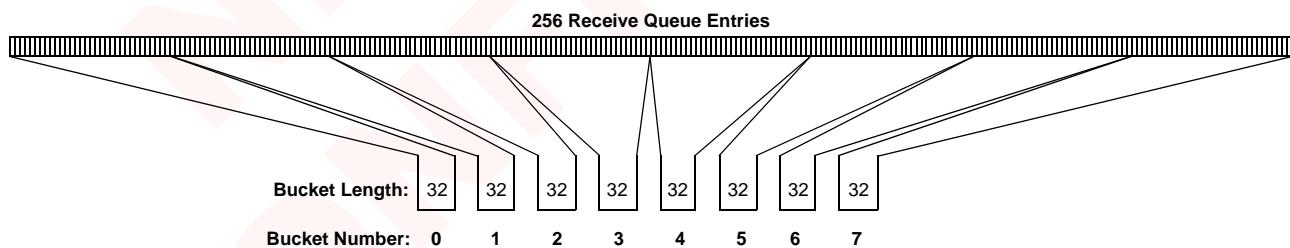
The Receive Queue Entries in Cores are allocated to Buckets by writing to the `MsgBucketSize` COP2 register. See [Section 4.3.2.7 on page 133](#). Non-Core Stations are configured through memory write operations to their internal registers. For details about I/O blocks, address space and registers, please see [Section 12.2, “FMN Registers for Non-Core Stations”](#). Also review the chapters discussing the particular blocks. This section will focus on setting up Core Buckets.

Each Core contains one `MsgBucketSize` register used to configure Bucket sizes. The `MsgBucketSize` register contains eight 8-bit fields, each specifying the size of a Bucket in the Core. The base address for each Bucket is calculated by hardware from the sizes written to this register. (Restrictions on the sizes of the buckets are described in [Section 12.4.1 on page 414](#).)

Note: Setup of the `MsgBucketSize` registers should only take place when there are no messages on the FMN, for instance, during system startup.

Following reset, the default Bucket sizes are as shown in [Table 12-1](#). All Buckets in all Cores are set to a length of 32 Receive Queue Entries each. [Figure 12-5](#) shows a way to think about this organization.

Figure 12-5. Each Bucket uses 32 Receive Queue Entries by Default

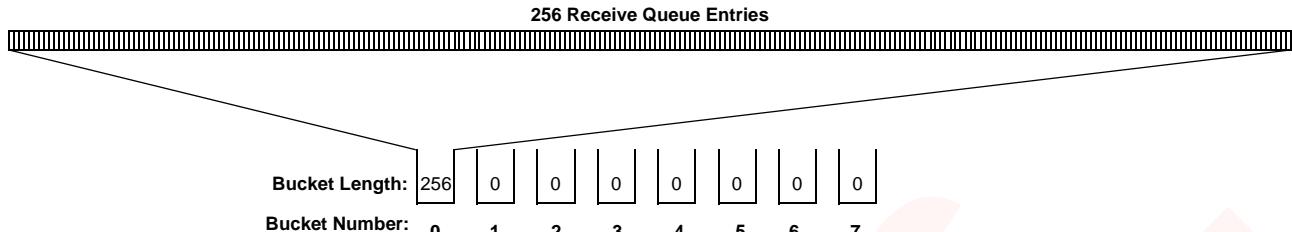


For example, suppose the software architecture assigns one Thread to receive and process messages from one Bucket, and that the other Threads have non-message-related tasks.¹ In this case, all 256 Receive Queue Entries in that Thread's FMN Station can be devoted to that one Bucket. (See [Figure 12-6](#).) The Thread is programmed to be notified when messages arrive at this Bucket. This provides the maximum possible latency for the Thread to process packets. If Bucket0 in each Core Station is assigned to receive messages, then according to [Table 12-1](#), the Bucket IDs that must be provided to the network interfaces as message

1. In fact, Threads can handle multiple Buckets, and one Thread could receive messages from all 8 Buckets of its Station, if desired. This is just a example for illustration.

destinations are: 0, 8, 16, 24, 32, 40, 48 and 54. These Bucket IDs would be programmed into the Packet Director of the network interfaces that are the source of the packet messages.²

Figure 12-6. One Bucket Uses All 256 Receive Queue Entries

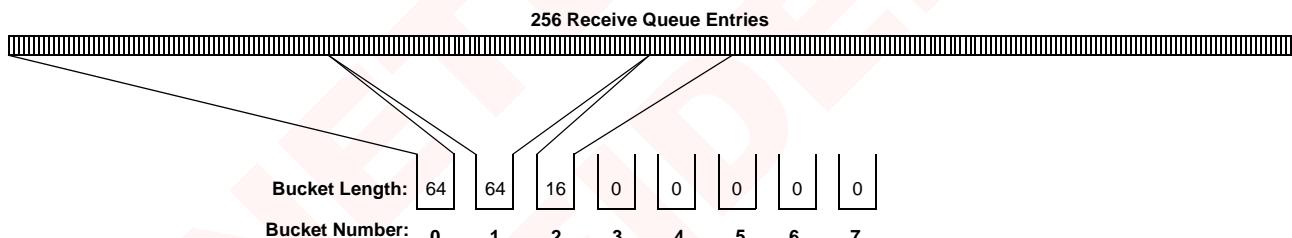


Alternately, software can distribute messages to multiple Threads to reduce latency. In this case, software can arrange to notify Threads in a round-robin fashion about incoming messages.

If some message sources have greater bandwidth, or if some Threads have longer processing latency, software may initialize larger Bucket sizes for these Threads. Threads that receive lower-bandwidth message traffic or that have short processing latency can have smaller Buckets. (See [Figure 12-7](#).)

Each Thread can receive messages from any Bucket in the Station. So, for example, one Thread can receive messages from up to 8 Buckets.

Figure 12-7. Mixed Bucket Sizes



12.1.5 Credit-based Message Management

The FMN is designed to avoid bottlenecks automatically, and to achieve top network throughput. During initialization, software establishes how FMN bandwidth is to be distributed among the sending and receiving Stations so that messages receive timely delivery. Once set up, the Stations on the FMN automatically administer their allotted bandwidth. Bandwidth usage can be adjusted dynamically, if desired. However, this should only take place when no messages are on the FMN.

Message flow-control is managed through a credit-based scheme. Stations must have enough Credits in order to send messages to destination Buckets. Each Station maintains a count of available Credits for each destination Bucket in the system. A Transmit Credit Manager within each Station (see [Figure 12-4](#)) is responsible for tracking available Credits for each destination Bucket.

There are 128 total Buckets in the system (see [Table 12-1](#)). Therefore, the Transmit Credit Manager in each Station maintains 128 Credit Counters, one for each Bucket in each Station.

2. Each network interface contains a Packet Director that distributes received packets to Threads for processing. See [Chapter 13, “Networking Accelerators”](#) for more about Packet Directors.

Each Station's Credit Counters determine whether the Station has enough Credits to send a message to a particular Bucket.

When a Station wishes to send a message to a Bucket, that Station's Transmit Credit Manager examines its Credit Counter for that Bucket. If the number of Credits available for that Bucket is less than the size of the message, then the message will not be sent. The Station must retry the message when it has enough Credits.

If the number of Credits available for that Bucket is greater than or equal to the size of the message, then the message can be sent immediately. Before sending a message, the sending Station's Credit Counter for that destination Bucket is automatically reduced by the size of the message. At this point, the FMN schedules and executes message delivery based on its traffic priorities. When the message is consumed by the recipient, the corresponding Credit Counter in the sending Station is automatically increased by the size of the message.

Note: Each Station is initially assigned zero Credits by default. Because no Station has any Credits after reset, no messages can be sent until Credits are assigned to Stations.

When a message is received by a Station, a copy of the message is placed in the Receive Queue Entries that were assigned to the targeted Bucket. The Bucket is organized as a FIFO, so multiple messages can be queued. An error is raised if more messages arrive at a Bucket than it can hold. This is an indication of failure to allocate Credits properly.

The fate of a message after it has been received by a Station depends upon the type of Station receiving the message. If it is a Core Station, the message is destined for a Thread; if it is a networking interface Station, it is destined to be transmitted via the targeted interface.

If the recipient is a Core Station, a Thread can be triggered to receive the message. The recipient Thread can detect newly arrived messages in one of three ways:

- Polling a bit in a status register
- Receiving an interrupt
- Resuming execution after being suspended, awaiting a message with the MsgWait instruction. (See [Section 5.3.9, “MSGWAIT — Wait for Message in the Receive Queue”](#).)

The recipient Thread pops the first available message from a Bucket by executing the MsgLd COP2 instruction. (See [Section 5.3.8, “MSGLD — Load Message Command”](#).) Any Thread can receive messages from any Bucket in its Station. Two actions are performed when the MsgLd instruction is executed:

1. The message is consumed by being transferred from the Receive Queue into the Receive Buffer of the Thread that executed the instruction (see [Figure](#)). Each Core has four dedicated Receive Buffers per Thread in COP2 register space. (If the Bucket is empty, an error condition is raised.)
2. Once the message has been consumed, the sending Station must be notified of the receipt of the message so it can increment its Credit Counter for the recipient Bucket. So, after the transfer is completed, a Free Credit message is returned to the sending Station via a special out-of-band channel, instructing it to increment the Credit Counter corresponding to the Bucket that received the message.

The use of an out-of-band channel to transmit Free Credit messages means that there is no throughput penalty for the FMN to process Free Credit messages.

Because this credit policy is enforced at every Station, the FMN automatically administers flow control once it has been set up by software.

If the message recipient is a networking interface Station, essentially the same steps are taken by the receiving Station. The arrival of the message in the Station's Receive Queue triggers the networking interface hardware:

1. The message is consumed by being transferred from the Bucket into the internal registers of the interface's hardware (see [Figure](#)).

2. After the transfer is completed, a Free Credit message is returned to the sending Station via a special out-of-band channel, instructing it to increment the Credit Counter corresponding to the Bucket that received the message.

Thus, the FMN provides a unified message-passing discipline for Cores, network interfaces, DMA, Security Acceleration Engine, and Compression/Decompression Engine. The only difference is how the hardware attached to a particular Station processes the messages received.

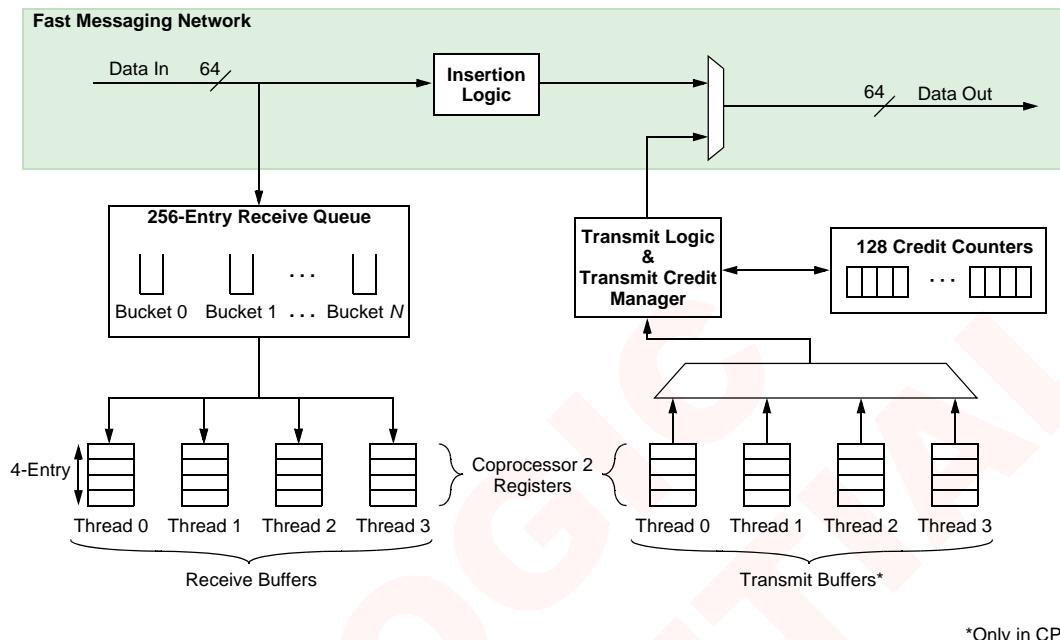
[Figure 12-8](#) shows a more detailed block diagram of a Core Station. Messages are serialized on the FMN as 64-bit Entries, one per FMN clock. Each 64-bit Entry is received in turn by the Station on the line marked Data In. The Insertion Logic in each Station gives priority to passing messages already on the FMN to the next Station on the ring. Therefore, new messages originating from this Station will be blocked while messages already on the FMN are passed from Data In to Data Out.

If the message is destined for a Bucket in this Station, then it is copied into the Receive Queue Entries dedicated to this Bucket by software. This new message is added behind any unprocessed messages already received by this Bucket. When all Entries of the message have been received, the awaiting Thread is notified, and the message can be popped by the recipient Thread. Each Thread has its own dedicated Receive Buffer into which popped messages are placed. (See [Figure 12-8](#).) When the transfer from Receive Queue to Receive Buffer has completed, an out-of-band Free Credit message is sent back to the sending Station.

Note that for reasons having to do with bandwidth fairness among the FMN Stations, the recipient Station does not remove the message from the FMN. The message continues around the ring until it returns to its originating Station. The originating Station removes the message, and is blocked from sending any new messages for the number of FMN clocks corresponding to the number of Entries in the message. This mechanism ensures FMN bandwidth fairness among Stations. This is described in greater detail below in [Section 12.1.6, “Message Precedence and Automatic Bandwidth Administration”](#).

Stations attached to network interfaces, DMA, and Security Acceleration Engine receive messages from the FMN in a manner essentially identical to the above.

When a Thread goes to transmit a message, it will execute a MsgSnd COP2 instruction. (See [Section 5.3.7, “MSG SND — Send Message Command”](#).) The Station’s Transmit Credit Manager determines whether there are enough Credits to transmit the message. If there are enough Credits, and if no other reason exists to block message transmission, then the MUX shown in [Figure 12-8](#) is set to transmit the contents of the Transmit Buffer onto the Data Out line.

Figure 12-8. Message Station Overview

*Only in CPUs

Note: The software designer must make a proper accounting of the various resources of the FMN system in order to utilize it. The sum of all Credits in all Stations for a particular Bucket must not exceed the number of Entries assigned to that Bucket. For example, if Bucket0 in Core0 Station has been allocated 32 Entries, then all the Credits for this Bucket in all Stations must add up to 32. This way, messages will only be sent to a Bucket if it has sufficient capacity to handle them, thereby ensuring that Buckets are not over-subscribed and do not overflow.

12.1.6 Message Precedence and Automatic Bandwidth Administration

Because the FMN is a ring-based design, it is likely that messages will be passing through other Stations en route to their destination Bucket. To ensure high performance and fair access, FMN Stations give precedence to traffic already in transit.

For example, suppose Core0 Station wishes to transmit a message, but simultaneously has a message passing through on its way to another Station. The message already in transit will be forwarded by the Station before the message originating in Core0.

Conversely, the FMN is designed so that in-transit traffic can't block Stations indefinitely from adding new messages. When a message makes a full traversal of the ring and comes back to the source Station, that Station cannot put any new messages onto the ring. This restriction is equal to the size of the message. This frees the next Station or Stations to add messages onto the FMN.

For example, suppose Core0 must send a message of size 2 to Bucket 87 in the Security Acceleration Engine. Core0 checks with its Transmit Credit Manager to see if it has two Credits available for that Bucket. The Credit Manager examines its credit counter for bucket 87. If it is greater than or equal to 2, then:

1. The message is given to the transmit logic of Core0's Station.
2. The Station's Transmit Credit Manager decrements available credits to Bucket 87 by 2.
3. The Station accepts the message for eventual transmission.

When it is not processing in-transit traffic, the message is placed on the FMN on two consecutive clocks.

The message travels around the FMN until it is received by the Security Acceleration Engine Station. A copy of the message is placed in that Station's Receive Queue for Bucket 87.

When this message is consumed by the Security Acceleration Engine and removed from its Receive Queue, an out-of-band Free Credit message is automatically sent back to the Transfer Credit Manager of Core0 Station instructing it to add two Credits to the Credit Counter for Bucket 87.

Note that the message is not removed from the FMN by the Security Acceleration Engine Station, even though it is the message recipient. The message continues to traverse the FMN until it returns to Core0. Core0, the sending Station, removes the message. It is blocked from putting new messages on the FMN for two cycles, which is the size of the original message. This mechanism guarantees fairness among the Stations.

In this manner, once the Receive Queues are partitioned into Buckets and the corresponding Credits have been distributed to the Stations, FMN hardware automatically manages message transmission bandwidth with no software overhead.

Note: Unused Buckets can be allocated zero credits.

In summary, the FMN is a high-speed network designed to pass messages between functional blocks of the XLS Processor in an optimal and fair way that ensures the greatest possible bandwidth utilization.

12.2 FMN Registers for Non-Core Stations

Setup of FMN resources for non-Core Stations (DMA, Networking Accelerators, CDE, and Security Acceleration Engine) is similar to setup of the Core Stations. However, non-Core Stations are configured through memory write operations. This section details the registers used to set up Bucket sizes and Credit Counters for these blocks.

12.2.1 Non-Core MsgBucketSize Register Format

The format of all non-Core MsgBucketSize registers is given below.

32	8 7	0
Reserved		SIZE

Bits		Description	R/W	Reset
#	Name			
32:8		Reserved		
7:0	SIZE0	Bucket size.	R/W	256/MaxBuckets ^a

a. See [Table 12-1](#).

Note: There are important restrictions on the sizes of Buckets. See [Section 12.4.1, “Setting Up Buckets”](#).

12.2.2 XAUI/SGMII Non-Core MsgBucketSize Register IDs

The Register ID offset, 0x320, is the same in each block with respect to the base address of each block. Blocks vary in how many Bucket size registers are present, depending upon the number of Buckets each block contains. The identity of non-Core MsgBucketSize registers is given in the tables below.

As shown below, the Network Accelerator interfaces contain six MsgBucketSize Registers for SGMII, and two for XAUI.

Table 12-2. XAUI/SGMII MsgBucketSize Registers

Register ID	Address Offset	Name
0x320	0xC80	Unused
0x321	0xC84	Free Credit Bucket
0x322	0xC88	Tx0 Bucket
0x323	0xC8C	Tx1 Bucket (SGMII only)
0x324	0xC90	Tx2 Bucket (SGMII only)
0x325	0xC94	Tx3 Bucket (SGMII only)
0x326	0xC98	Unused
0x327	0xC9C	Free Credit Bucket (Context 1 — Split Mode) (SGMII only)

Note that to ensure correct operation, all SGMII MsgfBucketSize registers, including those listed as “Unused” in [Table 12-2](#), must be programmed. Those marked “Unused” in [Table 12-2](#) MUST be programmed to zero.

In addition, if the Tx Bucket sizes are set (in the Txn Bucket registers) to values greater than 8, the buckets will overflow. When the buckets overflow, since there is no buffer region protection in hardware, messages will be placed in other buckets resulting in unpredictable software behavior.

The Security Acceleration Engine contains two MsgBucketSize Registers.

Table 12-3. Security Acceleration Engine MsgBucketSize Registers

Register ID	Address Offset	Name
0x320	0xC80	SEC_PIPE0
0x321	0xC84	SEC_RSA_PIPE

The DMA Engine contains four (4) MsgBucketSize Registers.

Table 12-4. DMA MsgBucketSize Registers

Register ID	Address Offset	Name
0x320	0xC80	DMA_CHANNEL0
0x321	0xC84	DMA_CHANNEL1
0x322	0xC88	DMA_CHANNEL2
0x323	0xC8C	DMA_CHANNEL3

The CDE contains two (2) MsgBucketSize Registers

Table 12-5. CDE MsgBucketSize Registers (XLS6xx, XLS4xx-Lite and XLS4xx devices)

Register ID	Address Offset	Name
0x320	0xC80	Free descriptor bucket
0x321	0xC84	Compression/decompression bucket

The PCIe interface contains eight (8) MsgBucketSize Registers.

Table 12-6. PCIe MsgBucketSize Registers

Register ID	Address Offset	Name	XLS Device Type
0x320	0xC80	TX0 message bucket size	These apply to all XLS devices.
0x321	0xC84	RX0 message bucket size	
0x322	0xC88	TX1 message bucket size	
0x323	0xC8C	RX1 message bucket size	
0x324	0xC90	TX2 message bucket size	These do not apply to XLS4xx-Lite, or XLS1xx devices.
0x325	0xC94	RX2 message bucket size	
0x326	0xC98	TX3 message bucket size	
0x327	0xC9C	RX3 message bucket size	

12.2.3 Non-Core Credit Counter Register Format

The format of all non-Core Credit Counter registers is given below.

32		8 7 0
Reserved		COUNT
Bits	#	Description
#	Name	R/W Reset
32:8		Reserved
7:0	COUNT	Credit Counter. To enable a non-Core station to send message to a destination bucket, its non-Core Credit Counter must be greater than or equal to 4.

12.2.4 Non-Core Credit Counter Register IDs

The Credit Counter Register ID offset, 0x380, is common for all blocks relative to the base address of each block. The number of these registers is the same in each block, because there must be a Credit Counter in each Station for each potential destination Bucket.

Table 12-7. Credit Counter Registers

Register ID	Address Offset	Name
0x380 – 0x3FF	0xE00 – 0xFFC	Credit Counter registers

12.2.5 PCIE_MSG_TX_THRESHOLD

(For XLS408Lite and XLS404Lite)

Address Offset: 0xC20

Bit#	Name	Description	Attribute	Reset
31:0	MSG_TX_THRESHOLD	Message transmit credit threshold size.	R/W	0x00000003

12.3

Interrupt Causes from Core Stations

Threads can detect that messages have arrived in the following ways:

- By polling status fields in COP2 registers
- By being awakened from the MsgWait instruction
- By receiving interrupts when the associated FMN Station's Receive Queue changes status.

This section discusses the sources of Thread interrupts from Core Stations.

The MsgWait instruction allows a Thread to be suspended pending the arrival of a message at a particular Bucket. When it issues the MsgWait command, the Thread indicates the Bucket that it is waiting on. The Thread is resumed when a message is delivered to that particular Bucket.

In contrast, interrupts are generated from Core Stations only in response to overall changes in Receive Queue status. Four interrupt types can be generated from Core FMN Stations:

- Functional interrupts:
 - Receive Queue Non-empty
A level interrupt indicates that the Station's Receive Queue is non-empty.
 - Receive Queue Watermark Exceeded
A level interrupt indicates that the number of Receive Queue Entries holding messages has exceeded a programmable threshold.
- Debug/status-related interrupts:
 - Receive Queue Overflow
The Receive Queue was full when a new message was received.
 - Invalid Bucket ID
A message originating from this Station had an invalid Bucket ID.

These interrupt conditions are discussed in the following sections.

12.3.1

Functional Interrupts

Interrupts can be generated by the Core Stations indicating a change in status of the Receive Queue as a whole. These interrupt conditions are configured in the MsgConfig register of each Core. See [Section 4.3.2.5, "MsgConfig"](#).

One or more Threads can be designated to receive these interrupts in round-robin fashion by setting the ITM mask field in the MsgConfig register to indicate the participating Threads. Note, there is no broadcast facility for these interrupts: only one of the participating Threads will receive each interrupt, based on the ITM mask and the ID of the Thread previously interrupted by the Station.

Note: Both Non-empty and Threshold Exceeded interrupts can be enabled simultaneously. Though the hardware supports this configuration, it is not recommended because they are intended for mutually-exclusive message-handling strategies.

Receive Queue Non-empty

Threads can be interrupted when the Receive Queue is non-empty. The interrupt service routine can determine the non-zero length Bucket by reading the RFBE field of the MsgStatus register.

This is a level interrupt condition that, if enabled, will hold the interrupt so long as the Receive Queue is non-empty. If this interrupt is enabled and the Receive Queue is empty, then the Core Station will generate this interrupt to the designated Thread when the next message arrives. If, on exit from the interrupt service routine, the Receive Queue is still non-empty, another

interrupt will be generated immediately. The interrupt condition persists until the Receive Queue is empty.

This facility can be used to architect an eager response to messages arriving in any Bucket.

Receive Queue Watermark Exceeded

This level interrupt indicates that the number of Receive Queue Entries holding messages has exceeded a programmable threshold. The threshold is determined by the ITM field of the MsgConfig register.

This is a level interrupt condition that, if enabled, will hold the interrupt so long as the number of Entries in the Receive Queue containing messages exceeds a programmable threshold. If this interrupt is enabled and the threshold is exceeded, then the Core Station will generate this interrupt to the designated Thread. If, on exit from the interrupt service routine, the threshold is still exceeded, another interrupt will be generated immediately. The interrupt condition persists until the threshold is no longer exceeded.

This facility can be used to architect a measured response to messages arriving in any Bucket.

12.3.2 Debugging/Status Interrupts

Interrupts can be generated by the Core Stations to provide debugging and status information. These interrupt conditions are configured in the MsgConfig1 register. See [Section 4.3.2.6 on page 132](#).

One or more Threads can be designated to receive these interrupts in round-robin fashion by setting the ITM mask field in the MsgConfig register to indicate the participating Threads. Note, there is no broadcast facility for these interrupts: only one of the participating Threads will receive each interrupt, based on the ITM mask and the ID of the Thread previously interrupted by the Station.

Receive Queue Overflow

Threads can be interrupted when an incoming message is detected but the Receive Queue is already full, indicating a message overrun condition. The F bit of the MsgStatus1 register indicates this condition. If this interrupt is not enabled, the F bit still indicates this condition, and can be tested by polling.

This interrupt condition indicates that software failed to allocate Credits appropriately. When the sum of Credits across all Stations equals the number of Entries in a Bucket, this condition cannot arise.

This is an edge interrupt condition that arises when the FMN determines that an incoming message was detected while the Receive Queue was full. The interrupt is held until cleared by writing the MsgStatus1 register.

Invalid Bucket ID

The address space of Bucket IDs is 0 to 127, however not all Bucket IDs are assigned to blocks in the XLS Processor. For example, Bucket IDs 108 – 111 are unused (see [Table 12-1](#)). Messages to these Buckets are invalid, and will trigger this interrupt if it is enabled. The T bit of the MsgStatus1 register indicates this condition. If this interrupt is not enabled, the T bit still indicates this condition, and can be tested by polling.

This is an edge interrupt condition that arises when the FMN determines that a message contains an invalid Bucket ID. The interrupt is held until cleared by writing the MsgStatus1 register.

12.4

Configuring the Messaging System

Before the messaging system can be used, it must be configured. Software designers must develop a system-level view about how it is to be used, and that approach will determine how the FMN should be configured. This section describes the required considerations.

The parameters that must be configured are:

- Bucket Sizes
- Credits
- Interrupts (applies only to Core Stations).

Setting up interrupts requires these steps:

- Enable the required Interrupts (see [Section 12.3, “Interrupt Causes from Core Stations”](#))
- Set required Interrupt Vectors
- Set watermark thresholds, if appropriate
- Set appropriate mask for Threads that are to participate in round-robin interrupt servicing.

Each Core Station is configured through COP2 registers. For a description of the registers and instructions supported for messaging in the Cores, see [Chapter 4, “XLS Processor Core Registers”](#).

Non-Core Stations are configured through memory write operations. See [Section 12.2, “FMN Registers for Non-Core Stations”](#).

12.4.1

Setting Up Buckets

The 256-entry Receive Queue (RcvQ) at each Station collects incoming messages. Each Station's RcvQ has a fixed number of Buckets that can divide up the RcvQ, each with its own distinct target address or ID. A Bucket acts like a mail box for receiving messages. Particular Buckets can be disabled if not needed by assigning them a size of zero.

It is important to determine how many Buckets each Station will use. Also the size of each Bucket must be determined.

Bucket size is subject to the following requirements:

- Bucket size must be a power of 2, or 0 (writing a value of 0, disables the bucket).
- Bucket size must be ≥ 4 , if not 0.
- The sum of all Buckets must not exceed the number of Entries in that Station's RcvQ.

Each core has a maximum of 8 Buckets per 256-Entry RcvQ. By default, each Bucket gets 32 entries. However by configuring the size of each Bucket, this default can be changed. For example, Core1 could be configured as follows:

Bucket	Bucket ID	Size	Offset
0	8	128	0
1	9	64	128
2	10	64	192
3	11	0	
4	12	0	
5	13	0	
6	14	0	
7	15	0	

In this example, only Buckets 0, 1, and 2 are being used, all others are disabled by setting their size of zero.

Note that the RcvQ has been divided into 3 segments:

- Bucket 0 uses entries 0–127 of the RcvQ, that is, RcvQ offset for Bucket 0 = 0
- Bucket 1 uses entries 128–191 of the RcvQ, that is, RcvQ offset for Bucket 1 = 128
- Bucket 2 uses entries 192–255 of the RcvQ, that is, RcvQ offset for Bucket 2 = 192.

The following conditions apply when configuring Buckets (where N refers to a Bucket ID):

$$\begin{aligned} \text{Offset}(0) &= 0 \\ \text{Offset}(N) &= \text{Offset}(N-1) + \text{Size}(N-1). \end{aligned}$$

The resulting offsets for each Bucket have the following additional requirement:

$$(\text{Offset mod Size}) == 0$$

Thus, specifying $\text{Size}(0) = 16$, $\text{Size}(1) = 4$, and $\text{Size}(2) = 16$ is illegal, because this results in:

$$\begin{aligned} \text{Offset}(0) &= 0 \\ \text{Offset}(1) &= 16 \\ \text{Offset}(2) &= 20 \quad \Rightarrow \text{Illegal, since } 20 \bmod 16 \text{ is not } 0 \end{aligned}$$

Specifying $\text{Size}(0) = 16$, and $\text{Size}(1) = 16$, and $\text{Size}(2) = 4$ is legal, because this results in:

$$\begin{aligned} \text{Offset}(0) &= 0 \\ \text{Offset}(1) &= 16 \\ \text{Offset}(2) &= 32 \end{aligned}$$

All offsets now meet the requirement that $(\text{Offset mod Size}) == 0$.

12.4.2 Credits

There is a total of 128 possible destinations; thus each Station maintains 128 Credit Counters for each of the destination Buckets.

By default, Credit Counters in each Station are all initialized to zero. Thus no Station can send messages until it is allocated Credits.

After each Bucket size is determined, Credits for each destination must be distributed to Stations that are going to be sending messages to that destination.

Note: The sum of Credits distributed for a particular bucket MUST NOT EXCEED the size of the Bucket. Hardware DOES NOT ENFORCE CORRECTNESS, and illegal configurations can result in lost messages, and in the worst case, a disabled Station.

When a Station sends out a message, the sender's Credit Counter for the destination is decremented by one Credit for each Entry in the message. On reaching the target, the message goes into the specified Bucket of its RcvQ. When the receiver pops the message from its Bucket, a Free Credit message is sent back to the Sender, which increments its Credit Counter associated with the particular target Bucket.

For example, suppose Core1/Bucket1 has a destination ID of 9, and has been set up for a bucket size of 64 (must be a power of two). Because of the bucket size that was set up, the total number of Credits distributed across all Credit Counters in all Stations should be ≤ 64 .

A sample of a credit distribution that would comply with the " ≤ 64 " requirement discussed above would be if the credits for Destination ID 8 are distributed as follows:

Core0[Credit(8)]	= 7
Core1[Credit(8)]	= 11
CDE	= 14
DMA[Credit(8)]	= 20
Security[Credit(8)]	= 12
Total:	64

For all other Stations, Credit(8) = 0.

Here's an example of Credit Counter management.

1. Core0 sends a message of size 4 to Core1/Bucket1 (Id == 9).
Core0[Credit(8)] is decremented by 4, changing from 7 to 3.
2. Core0 sends a message of size 2 to Core1/Bucket1 (Id == 9)
Core0[Credit(8)] decrements by 2, changing from 3 to 1
3. Core0 sends a message of size 3 to Core1/Bucket1 (Id == 9)
Core0[Credit(8)] is unchanged at 1
**** Send Message Fails because there are not enough Credits ****
**** It is up to software to retry a failed send message ****
4. Core1 pops a message from Bucket 1, resulting in a Free Credit message of 4 sent back to Bucket 8 to Core0.
5. Core0[Credit(8)] increments by 4, changes from 1 to 5.
6. Core1 sends a Free Credit message of 2 for Bucket 8 to Core0.
Core0[Credit(8)] increments by 2, changes from 5 to 7.

The third message from Core0 is not automatically re-queued to send when enough Credits are restored. It is up to software to retry a failed send message command.

If Credit Counters are given a non-zero count for a non-existent or a disabled Bucket, the sender will be able to send messages until there are no Credits available; however the messages will be lost, because the destination was disabled. The sender's Credits also will never be replenished because no Free Credit messages will be returned to it from disabled destinations, and when the Credits run out, the Station will not be able to send any more messages to that Bucket.

The following ideas are important to note:

- Each Bucket has its own unique Destination ID, so a message is sent addressed to a Bucket and not to a Station. For example, Core1 has 8 buckets with Destination IDs 8-15.
- On the receiving end however, the message source is only associated with the Station from which the message came, not the Bucket. For example, a message received from Core1 would have a source ID of 8.
- Messages can only be sent to a particular Bucket ID, not to a particular Thread. It is up to the software architecture to determine which Thread is notified of message arrival. One Thread can be designated, or a mask of Threads can participate in round-robin scheduling of message notification via interrupts.
- Buckets and Threads are completely orthogonal; there is no relationship between them in hardware. All Threads have access to all Buckets of the associated Station. However, software is free to associate Buckets with Threads, that is, each Thread can be limited (in software) to access only certain Buckets. For example:
 - Thread 0 could be assigned Buckets 0, 1.
 - Thread 1 could be assigned Buckets 2, 3.
 - Thread 2 could be assigned Buckets 4, 5.
 - Thread 3 could be assigned Buckets 6, 7.

12.4.3 Sending Messages

Sending a message is done through the (COP2) Instruction. See [Section 5.3.7, “MSG SND — Send Message Command”](#). The parameters of the message are set up in a GPR specified in the instruction. Here's how one would normally send a message:

- Set up the message by writing message data to be sent into the COP2 Transmit Buffer Registers. Each Thread has four of these registers. See [Section 4.3.2.1 on page 128](#). Message contents are arbitrary, depending on the semantics of the particular application. The networking interfaces expect data in Packet Descriptor format. (See [Section 13.3.6 on page 433](#).)
- Set up the parameters of the message transaction in a GPR. Information that must be set up includes:
 - Destination ID (a 7-bit field, corresponding to 128 destinations)
 - An optional 8-bit Software Code field (potentially useful to other Threads; not used by networking interfaces).
 - Message size – 1 (a 2-bit field)
- Execute a MsgSnd command.
- Check the MsgStatus register to determine success or failure. See [Section 4.3.2.3 on page 129](#). Each Thread has its own MsgStatus register. These registers contain the status information about their own last send and receive messages.
- Check the MsgStatus1 register for additional status. See [Section 4.3.2.4 on page 130](#).

When a Station receives a SndMsg command:

- It checks the destination Credit Counter to see if there is enough space; If not, MsgSnd fails, and the “Send Message Credit Fail” bit (MsgStatus.SCF) is set.
- It checks whether there is already a message pending because of a previous MsgSnd in the same Thread; If there is, MsgSnd fails, and the “Send Message Pending Fail” bit (MsgStatus.SPF) is set.

Both these bits are updated every time the MsgSnd command is executed. If neither of these checks are violated, then the MsgSnd command is accepted, and the message will be posted for delivery on the next available FMN slot.

The “Send Message Pending” bit (MsgStatus.SMP) register is set when the Station accepts the MsgSnd command, and is reset when the message has been posted to the FMN.

Note that writes to the COP2 Transmit Buffer registers will be blocked if there is a send message pending for that Thread. So if software does not want the execution of the Thread to be blocked, it should first check the “Send Message Pending” bit (MsgStatus.SMP) for any pending messages.

When sending a message, in addition to the 1 to 4 64-bit data which make up the body of the message, a Core can also include an 8-bit software code as part of the message. The hardware just passes this on to the destination.

Here's a simple pseudo-code example of sending a message:

```
// write to Transmit-Buffer registers to setup message data
MTC2 GPR, XmitBuffer
loop:
  MsgSnd rx (rx is any GPR, contains the dest, size and SW Code)
  MFC2 GPR, MsgStatus // Read MsgStatus Register
  Extract from VALUE whether msg send succeeded
  If not, goto loop
```

Network interfaces, DMA and the Security Acceleration Engine transmit messages on the FMN in an identical manner, except that they have dedicated hardware registers instead of Transmit Buffers, depending upon their design.

12.4.4

Receive/Load Messages

The MsgLd (COP2) instruction sends a command to the Station to pop a message from the RcvQ into that Thread's Receive Buffers (COP2 registers). See [Section 5.3.8, "MSGLD — Load Message Command"](#) and [Section 4.3.2.2 on page 128](#).

Each Station has a number of Buckets associated with it. For example, each Core has up to 8 Buckets. So the MsgLd instruction requires a 3-bit parameter to specify which Bucket to use for the Load message.

When the MsgLd command is received by the Station:

- If the Receive Queue bucket is empty, the command will fail, and the “Load Message Empty Fail” bit (MsgStatus.LEF) is set.
- If there is another Load message pending for the same Thread, the command fails, and the “Load Message Pending Fail” bit (MsgStatus.LPF) is set.

These two bits are also updated every time a MsgLd command is executed. The message is accepted if the above two conditions are not violated, and the “Load Message Pending” bit (MsgStatus.LMP) is set. The Station then schedules loads across the different Threads. When the message is transferred from the RcvQ to the Thread's Receive Buffer, the “Load Message Pending” bit (MsgStatus.LMP) is cleared. Software can then access the message through a MFC2 instruction. See [Section 5.3.5, "MFC2 — Move Word From Coprocessor 2"](#).

The MsgLd instruction could still fail even after it has been accepted if the RcvQ becomes empty by the time that particular MsgLd command is scheduled. This could happen if another Thread popped a message from the same Bucket and the Bucket became empty. Thus, race conditions could arise between Threads receiving from the same Bucket. A semaphore or mutex should be used in this case.

After the MsgLd instruction completes, the message has been transferred to the Thread's Receive Buffers, and the MsgStatus register will reflect the status of the operation and its parameters. See [Section 4.3.2.3 on page 129](#). The information available in MsgStatus includes:

- RMSID — Receive Message Source ID (7 bits), the ID of the sending Station
- RMSC — Receive Message Software Code (8 bits)
- RMS — Receive message size minus 1 (2 bits).

If two consecutive MsgLd instructions are executed without reading from the Receive Buffers, the second message will overwrite the information in the Receive Buffers, and the first message will be lost. It is up to software to ensure that this doesn't happen.

Below are three simple pseudocode examples to pop a message.

The following is a continuous polling loop:

```
loop:
    MSGLD rx    (rx is any GPR, specifies the Bucket to load from)
    MFC2 GPR, MsgStatus // Read MsgStatus Register
    Extract from VALUE whether msg load succeeded
    If not, goto loop
    Extract Message parameters from VALUE (Size, SW-Code, Msg-Src)
```

Threads can suspend execution until the arrival of a message in a particular Bucket using the MsgWait instruction. See [Section 5.3.9, "MSGWAIT — Wait for Message in the Receive Queue"](#). The following pseudocode uses the MSGWAIT instruction to free the Core resources as the Thread sleeps until a message is available:

```
loop:
    MSGWAIT    (Sleep until a msg is available)
    MSGLD rx    (rx is any GPR, specifies the Bucket to load from)
    MFC2 GPR, MsgStatus // Read MsgStatus Register
    Extract from VALUE whether msg load succeeded
```

```

    If not, goto loop
    Extract Message parameters from VALUE (Size, SW-Code, Msg-Src)

```

It is also possible for Threads to receive an interrupt when a message is available, as described in [Section 12.3, “Interrupt Causes from Core Stations”](#).

The following uses the interrupting capability of Messaging block:

```

Wait for Interrupt indicating a message is available
In the Interrupt Handler:
    MSGLD rx (rx is any GPR, specifies the Bucket to load from)
    MFC2 GPR, MsgStatus // Read MsgStatus Register
    Extract from VALUE whether msg load succeeded
    If not, return from interrupt handler
    Extract from VALUE Message parameters
        (Size, SW-Code, Msg-Src)

```

There is only one MsgConfig register per Core, NOT per Thread. So any Thread writing to this register writes to the same register.

12.4.5

Defeaturing and Debug

The following debugging aids are supported.

- Messaging Error Condition Detection and Interrupt

Two conditions are detected.

- An incoming message was detected when the RcvQ is full.
- An invalid target was specified in a Send Message, and nobody claimed the message.

An Interrupt can be generated if these conditions are detected. To enable the interrupt, the MsgConfig1 Register must be set up. See [Section 4.3.2.6 on page 132](#).

- Credit counter overrun

This error condition is detected when a Free Credit Message causes the Credit Counter to overflow.

An Interrupt can be generated if this condition is detected. To enable the interrupt, the MsgConfig1 Register must be set up. See [Section 4.3.2.6 on page 132](#).

- Trace Mode

Trace Mode can be enabled by setting the Trace-Mode bit in the MsgConfig1 register. When Trace Mode is enabled, all messages (regardless of target) are written into Bucket0.

Note: When a Station is configured to use Trace Mode, software must ensure that only Bucket0 is enabled, and that the size of Bucket 0 is 256.

NETLOGIC
CONFIDENTIAL



Chapter 13 Networking Accelerators

13.1 Introduction

This chapter discusses and describes operation and use of the Networking Accelerators in an XLS Processor. The Networking Accelerators ensure that the XLS Processor can continuously process packet data at peak levels of performance and workload efficiency. They support the SGMII/RGMII interfaces' GMAC FIFOs, as seen in [Figure 13-1](#).

This chapter provides the following information:

- Overview of architecture
- Theory of operation
 - how packets flow in the architecture
 - operational description
- Programming model
 - startup/initialization
 - bringup requirements
- Networking Accelerator registers
 - master list of registers
 - register descriptions

13.2 Architecture

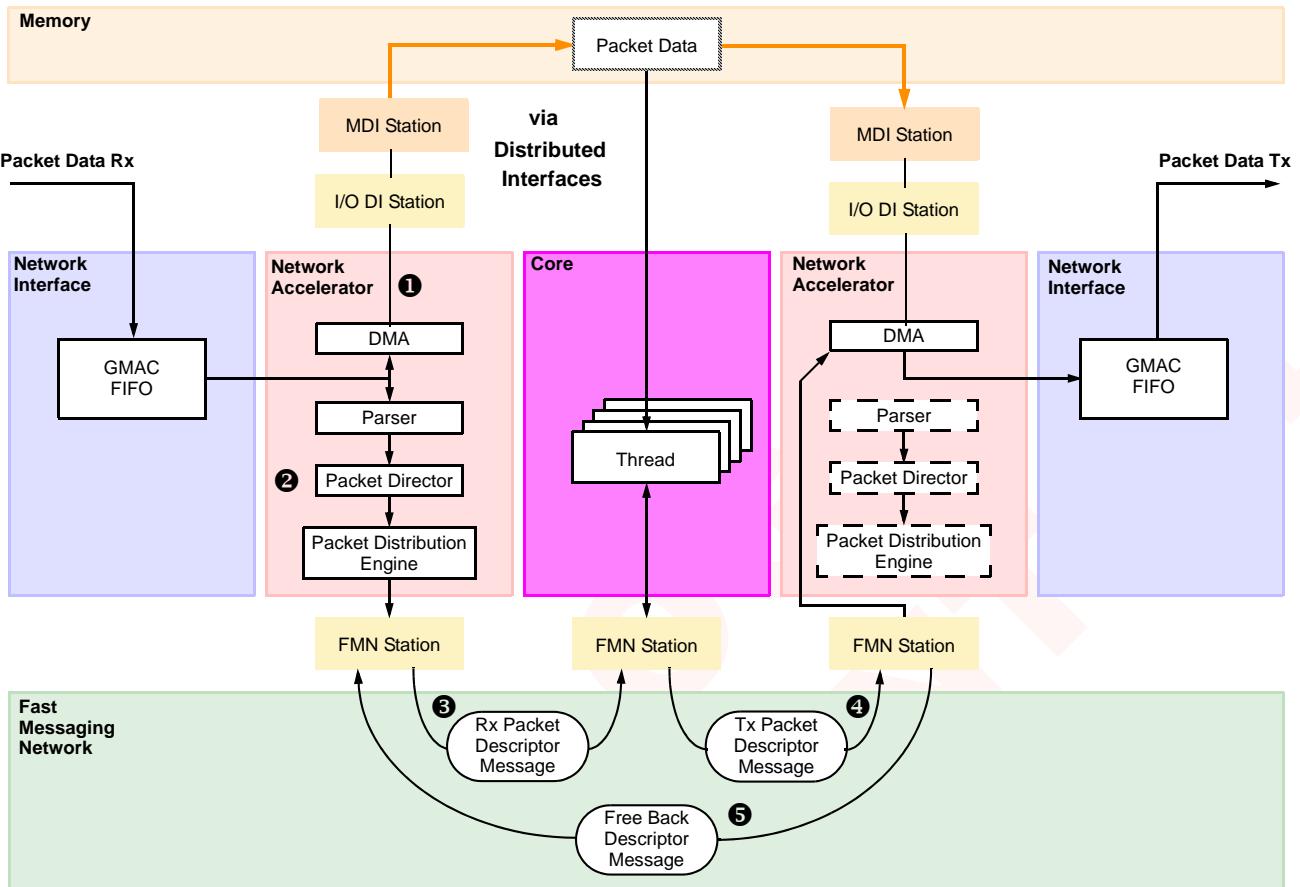
[Figure 1-1 on page 39](#) shows the position of the two Networking Accelerators in the architecture. They connect the interfaces to stations on the FMN and to the XLS device's internal I/O Distributed interconnect (I/O DI).

The Networking Accelerators provide a variety of functions:

- programmable packet acquisition
- parsing
- packet direction management
- checksum verification
- load-balanced packet distribution across multiple cores.

Each Networking Accelerator includes DMA, a Parser, the Packet Director, TCAM, and Packet Distribution Engine. DMA in the Networking Accelerators operates independently and is controlled by the Network Accelerators.

[Figure 13-1](#) shows a high-level overview of packet flow through the XLS Processor.

Figure 13-1. Packet Flow in the XLS Processor

Packets are received by a Network Interface such as the SGMII, RGMII, or FIFO interface. When a Network Interface begins to receive a packet, it must:

1. Store the packet data in memory, and
2. Notify software of the packet's arrival and location in memory.

Both steps are performed automatically by the Network Accelerator, based on parameters set up by software during initialization.

Step 1 includes allocating memory buffers to store the packet. As packet data arrives, the Network Accelerator's DMA consumes preallocated memory buffers and stores packet data in memory **①**.

Step 2 includes deciding which core bucket should be notified of the packet's arrival. Concurrently with step 1, the Network Accelerator parses and classifies the incoming packet data **②**. Based on this classification, a recipient core bucket is selected from a pool of candidate recipient Threads that are designed to handle packets of this kind. The Network Accelerator sends a message via the Fast Messaging Network (FMN) to the designated Thread announcing its arrival **③**. By providing flexible choice of recipient Thread, the networking interfaces achieve load balancing across a set of Threads.

A single FMN message may contain from one to four Packet Descriptors. Additional FMN messages will be generated as necessary to represent longer packets. Packet Descriptors contain address data, packet length and port of origin. One Packet Descriptor format includes a pointer to the packet data stored in memory. Another format includes a pointer to an array of

Packet Descriptors, allowing for packets of virtually unlimited size to be represented. A bit field indicates the last Packet Descriptor in a sequence.

Using Packet Descriptors, the Network Accelerators and Threads can send and receive packets, create new packets, forward packets to other Threads, to the Security Acceleration Engine, or to other Network Interfaces for transmission⁴.

When a packet is finally consumed, such as at a transmitting networking interface, the exhausted Packet Buffer must be returned to the originating interface so it can be reused. Facilities exist in the Network Accelerators to return freed Packet Descriptors back to their origin across the FMN without Thread intervention⁵.

Though the Network Accelerators transact FMN messages in Packet Descriptor format, the FMN itself is a general purpose message-passing system that can be used by Threads to communicate arbitrary information among themselves. The Fast Messaging Network is described in detail in [Chapter 12, “Fast Messaging Network”](#).

This chapter focuses on the architecture and management of the Network Accelerators associated with each Network Interface. The Network Accelerator function encompasses the operation of the Parser, the Packet Director, TCAM, and Packet Distribution Engine.

13.3

Theory of Operation

Network acceleration logic resides in the following Network Accelerators:

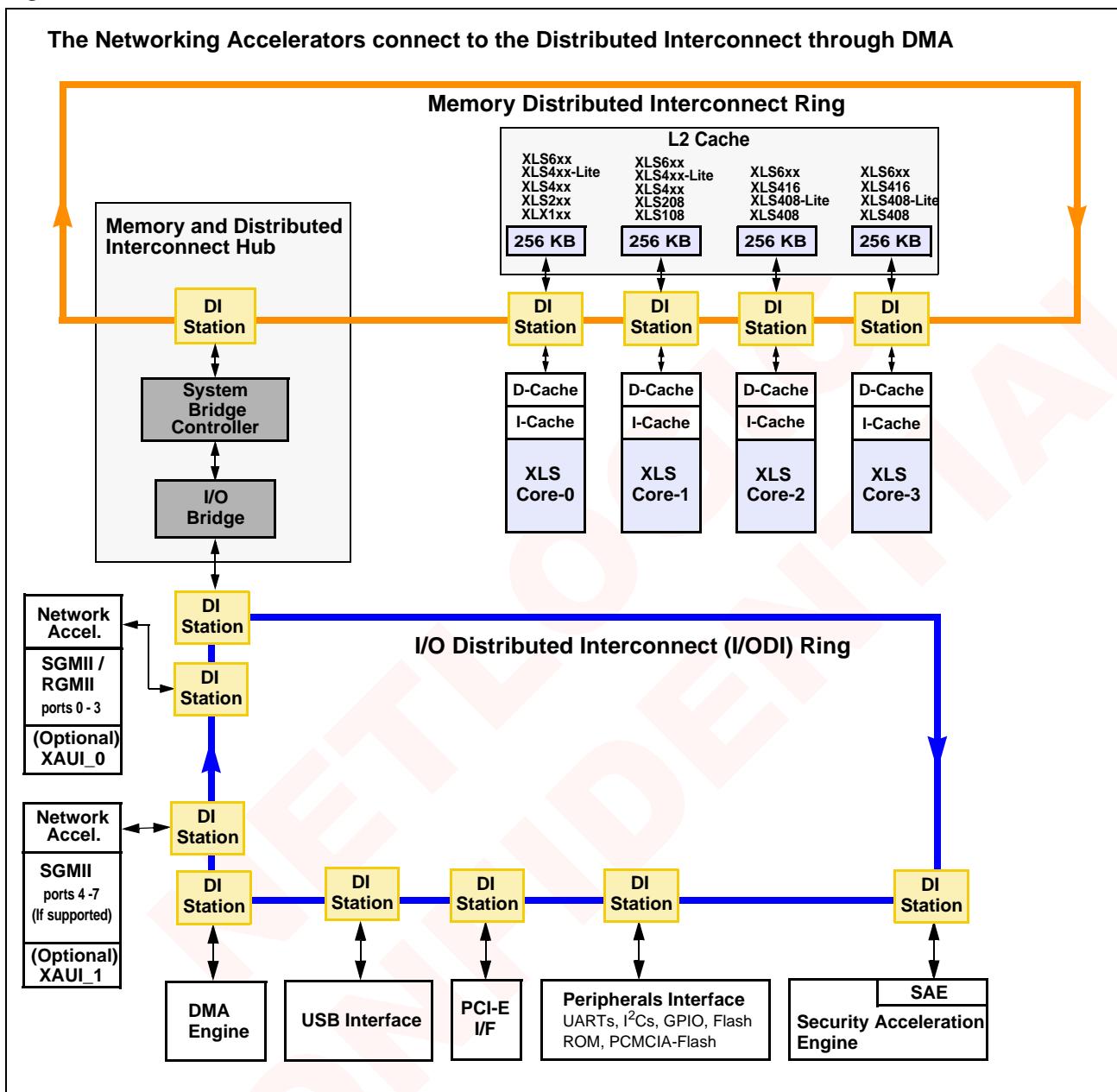
1. One Network Accelerator handling GMAC ports SGMII0 through SGMII3
2. One Network Accelerator handling GMAC ports SGMII4 through SGMII7 (if supported)

This section describes architecture, operation, and flow.

13.3.1

Packet Data Movement

As packet data arrives at a Network Interface, its Network Accelerator uses lists of preallocated memory buffers to store packet data in memory via its DMA. Packet data movement on the Distributed Interconnects is pictured in [Figure 13-2](#). This figure shows the Networking Accelerators connecting to the Distributed Interconnect through DMA.

Figure 13-2. Packet Data Movement on the Distributed Interconnects

13.3.2 Accessing the Networking Accelerator Registers

The following describes the position of the registers within the architecture.

As shown in [Figure 13-2](#), there are two Network Accelerator modules in the XLS Processor, as follows:

1. One shared by 4 GMACs, SGMII0 through SGMII3
2. One shared by 4 GMACs, SGMII4 through SGMII7 (if supported)

Each Accelerator is the portal for its companion GMACs to the internals of the XLS. Each Accelerator also processes interrupts from its companion GMACs, aggregates them, and passes an interrupt to the Programmable Interrupt Controller.

Each of the four GMACs in a quad share some common registers with the other GMACs in the quad. Each GMAC also has some individual (i.e., not shared) registers. Shared GMAC registers are in the range of 0x100 to 0x3FF; non-shared GMAC registers are in the range of 0x00 to 0xFF.

Each of the two Network Accelerators has its own individual set of registers.

Figure 13-3. Network Accelerator Register Layout

Network Accelerator A (GMAC QUAD 0)				Network Accelerator B (GMAC QUAD 1)				
0x00	SGMII0	SGMII1	SGMII2	SGMII3	SGMII4	SGMII5	SGMII6	SGMII7
0xFF	Individual	Individual	Individual	Individual	Individual	Individual	Individual	Individual
0x100	Common							
0x3FF								

Operation of the Networking Accelerators is governed by registers in sub-region address spaces as shown in the system memory map in [Figure 13-4](#). The offsets for the eight sub-region address spaces are given in [Table 13-1](#). All individual register offsets in this chapter apply to the corresponding sub-region for the Network Accelerator which shares the register space.

Table 13-1. Network Accelerator Sub-Regions within the Peripherals & I/O Configuration Region

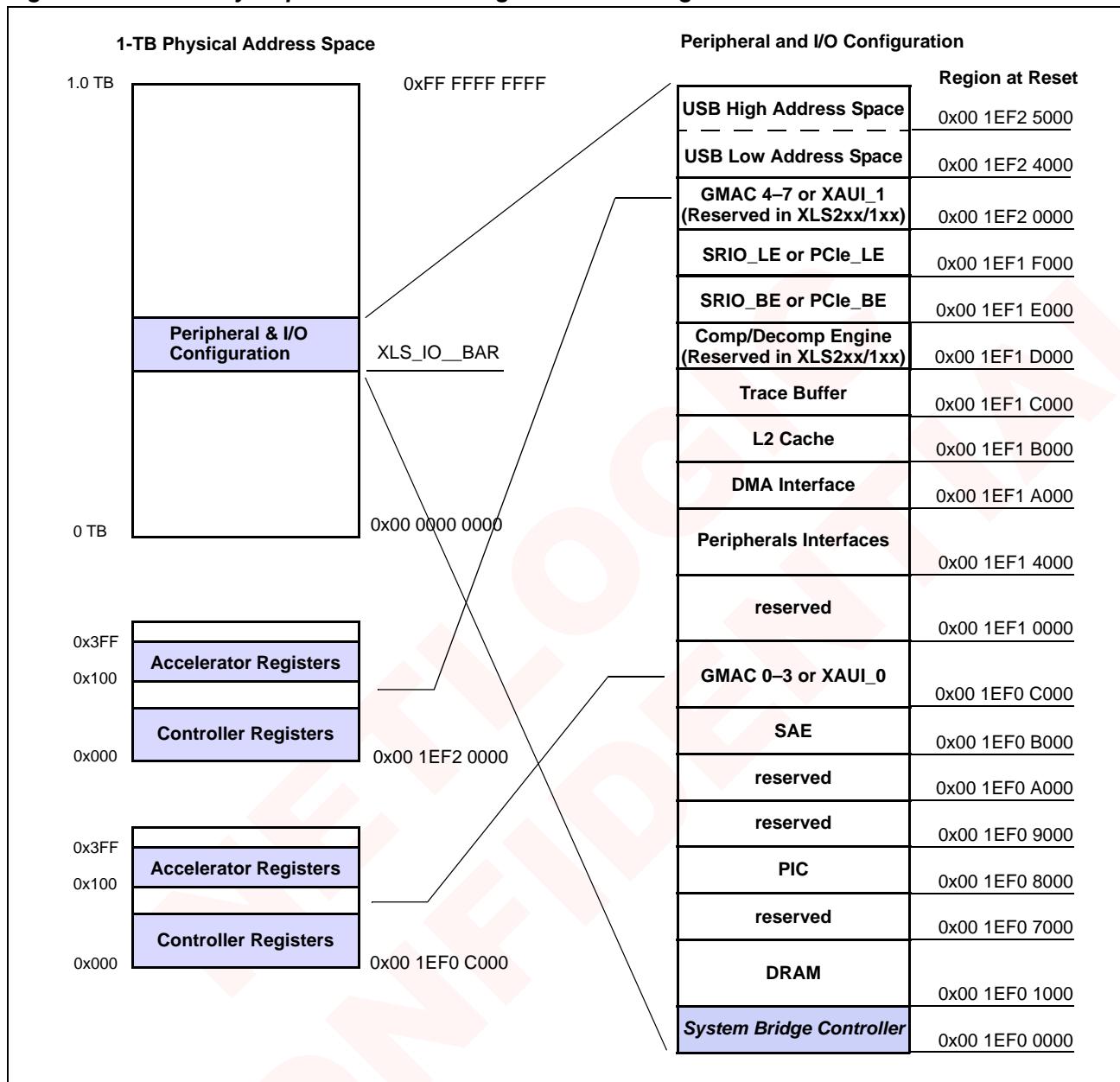
Peripheral	Peripheral Device Name	Sub-Region Offset	Sub-Region Offset Address at Reset
GMAC Quad 0			
(Optional) XAUI_0 Port ^a	XLS_IO_DEV_XAUI_0	0x0 C000	0x00 1EF0 C000
GMAC (or FIFO ^b) 0, Port SGMII0	XLS_IO_DEV_GMAC_0	0x0 C000	0x00 1EF0 C000
GMAC 1, Port SGMII1	XLS_IO_DEV_GMAC_1	0x0 D000	0x00 1EF0 D000
GMAC 2, Port SGMII2 ^c	XLS_IO_DEV_GMAC_2	0x0 E000	0x00 1EF0 E000
GMAC 3, Port SGMII3 ^c	XLS_IO_DEV_GMAC_3	0x0 F000	0x00 1EF0 F000
GMAC Quad 1			
(Optional) XAUI_1 Port ^a	XLS_IO_DEV_XAUI_1	0x2 0000	0x00 1EF2 0000
GMAC 4, Port SGMII4 ^d	XLS_IO_DEV_GMAC_4	0x2 0000	0x00 1EF2 0000
GMAC 5, Port SGMII5 ^d	XLS_IO_DEV_GMAC_5	0x2 1000	0x00 1EF2 1000
GMAC 6, Port SGMII6 ^d	XLS_IO_DEV_GMAC_6	0x2 2000	0x00 1EF2 2000
GMAC 7, Port SGMII7 ^d	XLS_IO_DEV_GMAC_7	0x2 3000	0x00 1EF2 3000

a. XAUI ports are in XLS616, XLS608, XLS416 and XLS408 devices only.

b. FIFO is optional configuration for NetLogic port when used

c. SGMII2-3 are not in XLS1xx devices

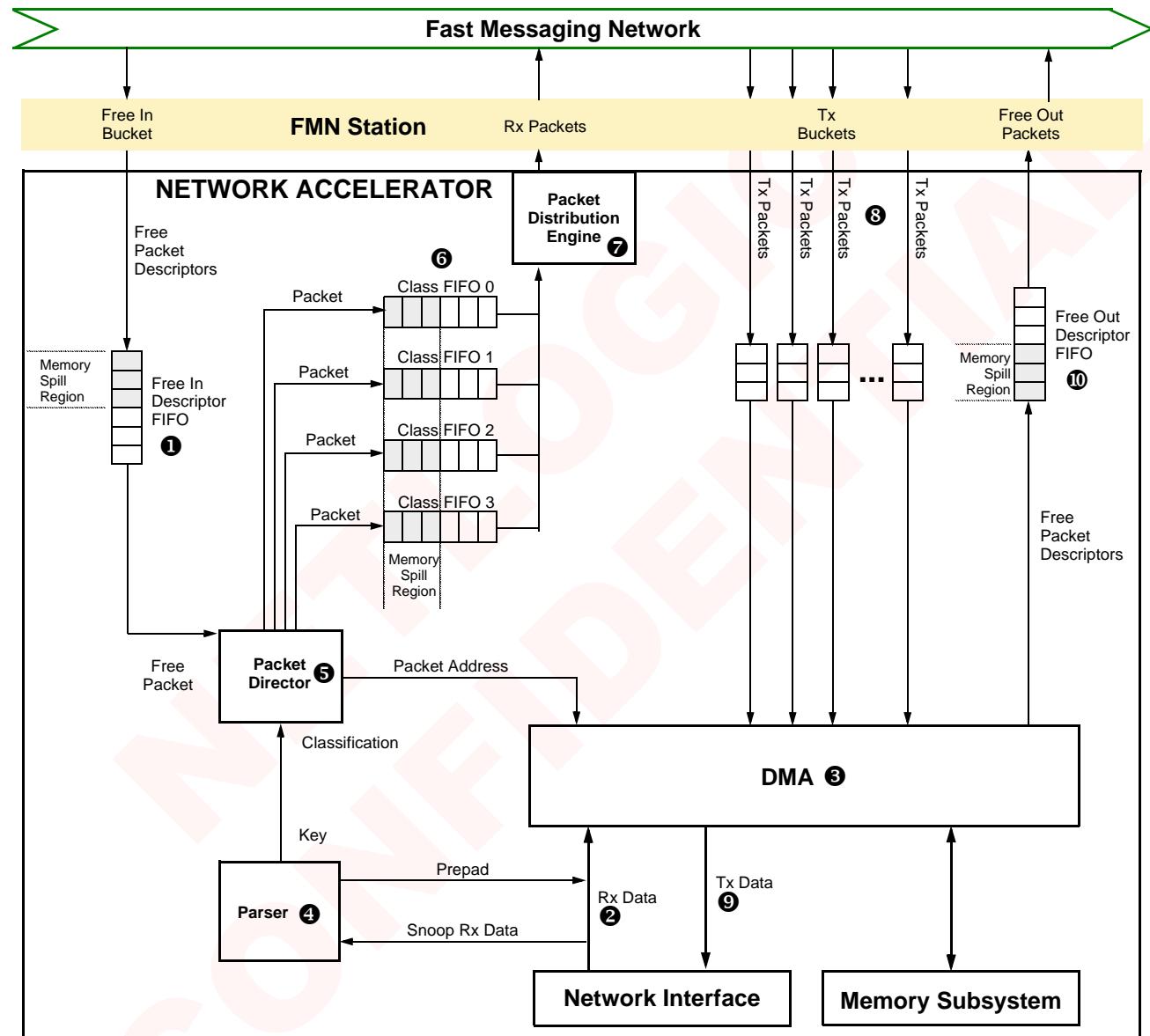
d. SGMII4-7 are not in the XLS2xx or XLS1xx devices

Figure 13-4. Memory Map for the Networking Accelerator Registers

13.3.3 Message Flow and Use of Packet Descriptors

Figure 13-5 presents a view of the Networking Accelerator connecting to the XLS device's internal Fast Messaging Network, showing message flow and the use of Packet Descriptors.

Figure 13-5. Networking Accelerator Connecting to the Fast Messaging Network



13.3.3.1 Operation Description

Packet Data Storage Setup

At system start-up, software provides the Network Accelerator with a number of fixed-size pre-allocated memory blocks to be used as Packet Buffers to store incoming packet data. Packet Buffer length is a GMAC parameter set by the RegularSize field of the [DescPackCtrl](#)

register. This size should be set the same for all GMACs. Each packet buffer must be a multiple of 32 bytes and can be up to 16 KB in length. Initialization software encapsulates pointers to the Packet Buffers in Packet Descriptors, and sends them via the FMN to the Network Accelerators.

Each Network Accelerator contains a Free In Descriptor FIFO❶ used to queue up these descriptors. Each such FIFO has a Bucket ID on the FMN. At startup, initialization software populates this FIFO with free Packet Descriptors. The Free In Descriptor FIFO can hold up to 128 Packet Descriptors on-chip and can be extended into memory using a “spill” mechanism. When a FIFO fills up, it uses spill regions in memory to store subsequent Descriptors. These spill regions can be made large enough to hold all descriptors necessary for a specific Network Accelerator. The spill regions holding the free Packet Descriptors may also be cached, if desired.

Packet Storage

When a packet comes in through the receive side❷ (marked RxData) of a Network Accelerator, a free Packet Descriptor is popped from the Free In Descriptor FIFO. The memory address pointer in the Descriptor is passed to the DMA engine❸, which starts sending the packet data to the memory subsystem.

As many additional Packet Descriptors are popped from the Free In Descriptor FIFO as are required to store the entire packet. The last Packet Descriptor has its end-of-packet (EOP) bit set.

Packet Parser

Concurrently, the incoming packet data is also snooped by the Parser❹, and predefined fields are extracted from it as it is received. The Parser concatenates the extracted fields into a 128-bit key that it delivers to the Packet Director❺ and TCAM.

Additionally, if the PrePadEnable bit of the [DescPackCtrl](#) is set, the Parser prepends a 256-bit header to the packet data containing the 128-bit key and other useful information designed to help the recipient Thread decode the packet. See [Section 13.8, “Prepad”](#).

Packet Director

The Packet Director❺ determines the classification method for the Network Accelerator. The Packet Director options are:

- A. Most-significant 7 Bits of the 128-bit key field from the packet (default classification method)
- B. Port-based classification
- C. Identification by combined L3/L4 protocols extracted by parser
- D. CRC7 Hash
- E. Ternary Content-Addressable Memory (TCAM) exact-match lookup
- E then A
- E then B
- E then C
- E then D

Note that enabling TCAM matching typically requires enabling one of the other classification methods as well, as an alternate method. In such cases, the TCAM exact-match test is performed first, with the alternate following if no exact match was found. [Table 13-2](#) illustrates.

Table 13-2. Packet Director Classification Modes

For this Director Mode	Enable This Classification Mechanism:				
	Most Significant 7 bits of Key Field	Port-Based ID HW	L3 / L4 Protocol-Based ID HW	CRC7 Hash Engine	TCAM
Default (Most Significant 7 bits of 128-bit key field)	ENABLE	DISABLE	DISABLE	DISABLE	DISABLE
Classification by Ingress Port	DISABLE	ENABLE	DISABLE	DISABLE	DISABLE
Classification by L3/L4 Protocol	DISABLE	DISABLE	ENABLE	DISABLE	DISABLE
Classification by Hash	DISABLE	DISABLE	DISABLE	ENABLE	DISABLE
TCAM Exact Match, then MS 7 bits of 128-bit key	ENABLE	DISABLE	DISABLE	DISABLE	ENABLE
TCAM Exact Match, then by-Port classification	DISABLE	ENABLE	DISABLE	DISABLE	ENABLE
TCAM Exact Match, then by-L3/L4 Protocol classification	DISABLE	DISABLE	ENABLE	DISABLE	ENABLE
TCAM Exact Match, then classification by Hashing	DISABLE	DISABLE	DISABLE	ENABLE	ENABLE

The Packet Director supplies the descriptor along with 9 bits of classification information to the Packet Distribution Engine (PDE) Class FIFOs, based upon the type of classification selected. ⑥

Packet Distribution Engine

The Packet Distribution Engine ⑦ (PDE) receives the classification information from the Class FIFOs and uses it to distribute the Packet Descriptors to Cores.

For each packet, the nine bits of classification information are: UseBucket, BucketID and ClassID.

These fields are interpreted as follows:

- UseBucket bit

The UseBucket bit specifies whether the PDE should use the Class distribution method (UseBucket==0), or should send the packet message to a specific Bucket ID (UseBucket==1).

- 6 bits that indicate a Bucket ID

If UseBucket==1, the packet will be directed to the Bucket indicated by the BucketID field received from the Packet Director.

- 2 bits of Packet Class ID

If UseBucket==0, the PDE will perform Class-based packet distribution, which provides for round-robin scheduling of packets among Threads. The ClassID field specifies which of the four Classes should determine the Packet Descriptor's destination. Each Class contains a 64-bit PDE_MASK indicating the set of CPU buckets that are allowed to receive these messages. The eligible set of buckets is a bitwise AND of PDE_MASK and

the set of buckets that have enough credits. The next Bucket is selected using a two-level hierarchical round-robin algorithm.

If UseBucket==1, the message will attempt to go only to the specified Bucket. The ClassID field is still used to allocate the Packet Descriptor to one of the four Class FIFOs. The PDE will only send the Packet Descriptor to the specified Bucket (PDE_MASK is ignored).

Algorithm Explanation

Assume cores are $C = \{C_0, C_1, \dots, C_7\}$, and their buckets are $B = \{B_0, \dots, B_7\}$. Assume the transformation $\text{ROUND_ROBIN}(V, P)$, where V is a binary vector, and P is an index into this vector. This is a function that searches for a "1" in non-zero binary vector V starting from $(P+1)$. When it reaches 7 it starts searching from index "zero".

$C(n).B$ represents the eligible bucket vector for Core n .

$C(n).P$ represents the last bucket selected from Core n .

Each element of C (i.e. C_0, C_1, \dots, C_7) is a logical OR of all the bits of $C(n).B$

$C.P$ represents the last core from which last bucket was selected

$k = \text{ROUND_ROBIN}(C, C.P)$

$\text{New_Selection} = (C(k).B, C(k).P)$

Each Class FIFO is 32 Entries deep and can spill into memory when the on-chip FIFO is full.

The Packet Distribution Engine packs from one to four Rx Packet Descriptors per FMN message, and will send as many FMN messages as necessary, depending upon the length of the packet and the size of the Descriptor Buffers. The EOP bit will be set in the first Packet Descriptor of the last FMN message.

The FMN drains the FIFOs as space is available and transmits the message to the designated Bucket, based on either the PDE Mask or BucketID.

PDE Algorithm

First, a Class FIFO is selected based on round-robin among the four Class FIFOs. The PDE skips over CLASS FIFOs that are blocked, and continues to process those remaining. For example, if a message is stalled on a CLASS FIFO because the destination bucket is full at a core, the PDE skips that CLASS FIFO.

If USE_BUCKET is set, then PDE checks to make sure enough credits are available to send the message. At least four credits are required, even if only one message need be sent. If enough credits are not there, then this class is skipped and the next class is selected.

If after the class is selected and the USE bucket is not set, then the 64-bit PDE MASK along with another 64-bit vector that contains an internally-used value called CREDIT_VALID, are used for selecting the next bucket. The CREDIT_VALID vector contains status for each bucket that informs whether enough credits are available for that bucket or not. These two 64-bit vectors are divided into 8 separate 8-bit vectors representing the 8 buckets in 8 CPU cores, and a two-level round-robin is done, one round-robin among the 8 CPU cores and another among the 8 buckets in each CPU core to select a candidate.

FMN Message Receipt by a Thread

When the FMN message containing the Packet Descriptors is received by the destination Bucket, the message is stored in the Receive Queue of the associated FMN Station. Ordinarily, the recipient will be a Thread executing in a Core. Once the message is received, the Core's FMN Station determines which Thread should be notified. The connection between Threads and Buckets in Core Stations is programmable, as described in [Chapter 12, "Fast Messaging Network"](#).

Thread Processing

Threads can simply wait for a message to arrive on one or more Buckets using the MsgWait instruction. The Thread is suspended pending receipt of a message by that Bucket. (See [Section 5.3.9, “MSGWAIT — Wait for Message in the Receive Queue”](#).) When a Bucket with a matching ID receives a message, the suspended Thread resumes execution. It reads the message from the bucket, or uses out of the Receive Queue, and processes it appropriately. Alternately, Threads can be interrupted when the Receive Queue becomes non-empty.

The processed message may be modified and sent to another Thread for more processing, which may send it, with appropriate format modifications, to the Security Acceleration Engine or yet another Thread, and so on. Finally, it may be forwarded, with appropriate format modifications, to the egress side of a Networking Interface⑧ for transmission. The egress side of a Networking Interface is used to transmit data⑨.

Appropriate format modifications include such efforts as, for example, converting a message to Security Engine message format prior to sending it to the SAE block, or converting the descriptor from an Rx Descriptor to a Tx Descriptor prior to sending the packet to a Networking port for transmission.

Automatic Buffer Management

The Packet Descriptor message format also contains a free-back Bucket ID containing the Bucket ID of the Free In Descriptor FIFO of the originating MAC. After the Packet Descriptor's buffer has been emptied by the transmitting Network Interface, the free-back Bucket ID may be used to return the exhausted buffers automatically back to the Free In Descriptor FIFO of the originating Network Interface without any software intervention. The MAC at the egress side places the exhausted Packet Descriptors on its Free Out Descriptor FIFO⑩ where they are sent to the Free In Descriptor FIFO associated with the free-back Bucket ID. In this way, exhausted buffers are recycled to their originators for reuse without intervention of Threads. The Free Out Descriptor FIFO also contains a spill region to hold free Packet Descriptors in case the Free Out FIFO is full.

13.3.4

Packet Descriptor Types

There are five basic Packet Descriptor formats:

- Rx Packet Descriptor format — used by the ingress side of Network Interfaces to pass pointers to Packet Buffers and other useful information to Threads.
Network Interfaces can store any length packet by using multiple Rx Packet Descriptors. Each Rx Packet Descriptor indicates the size of its buffer in bytes. For all Rx Descriptors except the last, the length field is equal to the value of DescPackCtrl.RegularSize. For the last descriptor of the packet, the length field equals the length of the packet modulo the value of DescPackCtrl.RegularSize. The EOP field in the Rx Packet Descriptor indicates the last Packet Descriptor in a sequence.
- Pointer-to-Data (P2D) type Tx Packet Descriptor — used by the egress side of Network Interfaces to access pointers to Packet Buffers to be transmitted.
P2D Packet Descriptors contain the physical address location from which the transmitting Network Interface's DMA engine will read packet data to be transmitted. The physical address can be byte-aligned (i.e., it need not be cache-line aligned). The length field contained within P2D Descriptors describes the length of useful packet data in bytes. A maximum of 4 such Pointer to Data type descriptors can be used to describe a particular TX packet without requiring to the use of a pointer-to-pointer type TX descriptor.
- Pointer-to-Pointer (P2P) type TX Descriptor — used by the egress side of Network Interfaces to access packet data of virtually unlimited size.

The Pointer-to-Pointer (P2P) type Tx Descriptors allow FMN messages to convey a virtually unlimited number of Tx Pointer-to-Data (P2D) type Descriptors, avoiding the 4-Entry limit the FMN imposes on P2D messages.

- Free Back Descriptor — used by the Network Interfaces to indicate completion of packet processing
- Free In Descriptor — sent from Threads during initialization to populate the various Descriptor FIFOs of the Network Accelerators with free Packet Descriptors.

The physical address field specified in the P2P type descriptor resolves to the address of a table of P2D type Descriptors. This table can contain a maximum of $((2^{14} = 16 \text{ KB}) / 8) = 2048$ P2D Descriptors. (The length of each P2D Descriptor is 8 bytes.) Each P2D type Descriptor can resolve to a packet data payload maximum of 16 KB, thereby allowing packets of $(2048 * 16 \text{ KB})$ to be constructed. Up to four P2P type descriptors may be used in each message. P2P and P2D type descriptors may be freely mixed in a single message.

Packet Descriptors of all types are uniformly 64 bits (one double-word, or 8 bytes) in length.

13.3.5

FMN Messages

As shown in [Figure 12-3](#), the FMN limits message size to a maximum of four 64-bit Entries each. Packet Descriptors are all uniformly 8 bytes in length, therefore a maximum of four Packet Descriptors may be sent in one FMN message.

As a packet is being received by a Network Interface, the Packet Director will consume free Packet Descriptors and store packet data in the associated Packet Buffer. The Packet Director constructs an FMN message containing the Packet Descriptors as they are filled. The FMN message will be delivered either when the message is full or the packet ends. Let *PacketSize* indicate the length in bytes of a packet. Then if:

$$\text{PacketSize} > (4 \cdot \text{DescPackCtrl.RegularSize}) \quad (\text{EQ } 13.a)$$

the packet data will require more than one FMN message. In this case, the Network Accelerator will send a P2D message to the designated Thread containing the currently-filled Packet Buffers, clearing the End of Packet (EOP) bit in the Descriptors so the Thread will expect additional Rx Packet Descriptors. The last descriptor will have the EOP bit set. A length field in the last descriptor indicates the byte length of the last buffer.

Threads can determine the number of Entries in an FMN message via the *MsgStatus* register, CP2 Reg 2, sel0 at the time of receipt. The size of received messages is encoded in bits [7:6] of that register. This field contains the number of 64-bit Entries – 1. See [Section 4.3.2.3, "MsgStatus"](#).

13.3.6 Packet Descriptor Definitions

Packet Descriptor formats are defined in the following subsections.

13.3.6.1 Rx Packet Descriptor Format

Rx Packet Descriptors ([Figure 13-3](#)) are used by Network Accelerators to send messages about packets they have received. The format of Rx Packet Descriptors is given in [Table 13-3](#).

Table 13-3. Rx Packet Descriptor Format

63	62:56	55:54	53:40	39:5	4	3:0
EOP	Status	ClassId	ParticleLength	Address	U	PortID

Bits	Name	Description
63	EOP	End of Packet. 0: Not the last descriptor of the packet 1: Last descriptor of the packet.
62:56	Status	Status is valid only for the first Packet Descriptor of the last FMN message of a packet. See text following for a discussion. Also, see Table 13-4 for status bit definitions. ^a
55:54	ClassId	Class identification for ingress packet, computed by the Parser. 0: Class 0 1: Class 1 2: Class 2 3: Class 3 These values are used by software in conjunction with the PortID value to identify packet data. The ClassId field is the 2-bits of classification identification generated by the Parser.
53:40	ParticleLength	Size of this buffer in bytes. For all Rx Descriptors except the last, this field is equal to the value of DescPackCtrl.RegularSize. For the last descriptor of the packet, this field equals the length of the packet modulo the value of DescPackCtrl.RegularSize.
39:5	Address	Cacheline-aligned Packet Buffer address. Specifies the physical address of the ingress packet data aligned on a 32-byte cache line boundary.
4	U	Unclassified. 0: Hardware classification results are valid. Software may use the hardware-provided classification results. 1: Hardware classification is invalid. Software classification will be necessary to handle non-fast-path cases.
3:0	PortID	Port ID of the incoming packet. 0000: (Optional XAUI_0 or XAUI_1 are always '0000') 0000: GMAC Port 0 (in Quad 0) or Port 4 (in Quad 1) (for XLS2xx and XLS1xx devices, this selects Port 0) 0001: GMAC Port 1 (in Quad 0) or Port 5 (in Quad 1) (for XLS2xx and XLS1xx devices, this selects Port 1) 0010: GMAC Port 2 (in Quad 0) or Port 6 (in Quad 1) (for XLS2xx devices, this selects Port 2) 0011: GMAC Port 3 (in Quad 0) or Port 7 (in Quad 1) (for XLS2xx devices, this selects Port 3) 0100 to 1111: Reserved

- a. The status of a packet can only be determined after the Network Interface has received the entire packet. Therefore, the Status field can only be sent in the last FMN message.

Note: The Status field for the entire packet is placed in the *first* Packet Descriptor of the last FMN message (not the last Packet descriptor of the last FMN message as one might expect).

The PortID indicates the GMAC within this Network Accelerator on which the ingress packet arrived. The Receive Message Source ID field (RMSID) provided by the Fast Messaging Network identifies the source station. (See [Section 4.3.2.3, “MsgStatus”](#). The combination of PortID and RMSID uniquely identify the ingress location.

The Unclassified bit signifies whether the results of the hardware classification performed by the Network Accelerator are valid. Special-case circumstances may require software assistance with packet classification. If the Unclassified bit is set, then the hardware classification results may not be used by software. If the Unclassified bit is cleared, then the hardware classification results can be used by software. See “[ParserConfig](#)” on page 449, for definition of “Unclassified bits”.

The Address field resolves to the physical address of the Packet Buffer, aligned to a 32-byte cache line boundary. This is the address used by the Network Accelerator’s DMA engine to store ingress packet data to memory.

The ParticleLength field gives the size of the Packet Buffer in bytes. For all Rx Descriptors except the last, this field is equal to the value of DescPackCtrl.RegularSize. For the last descriptor of the packet, this field equals the length of the packet modulo the value of DescPackCtrl.RegularSize. The maximum size of the entire packet can be 16K – 1 bytes.

The ClassID field indicates the Class FIFO from which the PDE took the message.

Each Rx Packet Descriptor includes a status field in bits[62:56]. For GMACs, the status field is as shown in [Table 13-4](#). If bit[6] of this field is set, an error occurred, and the remaining bits indicate the error condition. If bit[6] is cleared, then the other bits indicate attributes of the packet data as shown.

Table 13-4. Rx Packet Descriptor Status Field Values for GMACs

Rx Packet Descriptor Status Bit Position	Error	OK	Description
62	1	0	If set, an error occurred, and the remaining bits indicate the error condition
61	0	Broadcast	If Error==0 && Broadcast == 1, this packet is Broadcast
60	0	Multicast	If Error==0 && Multicast == 1, this packet is Multicast
59	0	Unicast	If Error==0 && Unicast == 1, this packet is Unicast
58:57	{Code Error, CRC Error}	Unicast Address [4 choices] Maps to 4 MAC addresses programmed in the GMAC	If Error==1, these bits indicate nature of the error: code or CRC. If Error==0 && Unicast == 1, these bits provide the Unicast address.
56	Length Check Error	Vlan	If Error==1, this bit indicates a length check error. If Error==0, this is the Vlan bit

13.3.6.2 Tx Packet Descriptor Formats

Tx Packet Descriptors ([Figure 13-3](#)) are used by the egress side of Network Accelerators to receive messages about packets they are to transmit. There are two types of Tx Packet Descriptors, as described in this section.

Pointer-to-Data (P2D) type Tx Packet Descriptor

Pointer-to-Data (P2D) type Tx Packet Descriptors point to one Packet Buffer to be transmitted. A maximum of four P2D Descriptors can be encapsulated in one FMN message. [Figure 13-5](#) shows the bit-fields for the P2D Packet Descriptor format.

Tx Pointer to Data (P2D) Packet Descriptors point to packet data. P2D descriptors differ from the P2P type as follows:

- The PointerType bit must be cleared.
- The Address field need not be aligned on a cache line boundary.
- The Length may be any byte length. The Length field can be any value ranging from 0 to 16KB – 1.

Table 13-5. Tx Packet Descriptor Pointer-to-Data (P2D) Format

63	62	61	60:54	53:40	39:0
EOP	PointerType	RdEx	FBID	Length	Address

Bits	Name	Description
63	EOP	End of Packet. 0: Not last descriptor of packet 1: Last descriptor of packet
62	PointerType	Pointer type = 0 for P2D Packet Descriptors. 0: Pointer-to-Data Type 1: Pointer-to-Pointer Type Note: The PointerType bit must be cleared for the Packet Descriptor to be interpreted as a P2D.
61	RdEx	Read Exclusive. 0: Do not invalidate the cache line for the packet data memory after it has been read. 1: Invalidate the cache line for the packet data memory after it has been read. This option prevents the cache line from being written to DRAM, saving DRAM bandwidth. Usage: To prevent corrupted packets, the Read Exclusive bit should <u>not</u> be set: <ul style="list-style-type: none"> • For multicast/broadcast packets, or • If it is possible for a packet to be retransmitted due to collisions in Half-Duplex networks, or due to packet TX underrun.
60:54	FBID	Free Back ID. After the transmitting Network Accelerator has consumed this Descriptor, it will direct the exhausted Descriptor to the Bucket ID indicated by this field. This allows the Descriptor to return to its original pool of Free Tx Descriptors. Setting this to 127 will prevent this descriptor from being freed back at all. Note: FBID may point to a Free In Descriptor FIFO Bucket or a Core Bucket.

Bits	Name	Description
53:40	Length	Packet data length in bytes. Specifies the length in bytes of the Packet Buffer physically addressed by bits [39:0]. Any length is allowed for P2D Packet Descriptors. A length of zero will simply cause the Descriptor to be sent to the Bucket indicated by FBID without any data fetch. Note that End Of Packet (EOP) must be set for the last non-zero-length descriptor sent for the current packet. Thus, if a zero-length descriptor is sent as a packet's last descriptor, then EOP must be set for that descriptor and also for the preceding (non-zero-length) descriptor of the current packet.
39:0	Address	Byte-aligned address. Address points to the physical base address of packet data. Address can be byte-aligned. This address will be used by the egress Network Accelerator's DMA to access Tx packet data.

Note: The PointerType bit must be cleared for the Packet Descriptor to be interpreted as a P2D.

The Free Back ID (FBID) field makes it possible for the consumer of a packet to recycle the packet data buffers directly back to the originator. For example, suppose a packet is received by a Network Accelerator and then forwarded to a Thread, which subsequently must forward the packet to another Network Interface for transmission. When the transmitting Network Interface has finished consuming the packet, it will dispose of the exhausted packet buffer memory by returning it to the originating Network Accelerator. To accomplish this, the Thread must include the Bucket ID of the originating Network Accelerator's Free In Descriptor FIFO in the FreeBackID field of the Tx packet. When it has exhausted the buffer, the transmitting Network Accelerator sends the Packet Descriptor to the indicated Free In FIFO for reuse without Thread intervention.

If packet data has been cached, the Read Exclusive (RdEx) bit can be set if the transmitting Network Interface should invalidate the cached data upon which the packet data memory resides once it has been read.

Pointer-to-Pointer (P2P) type Tx Packet Descriptor

Tx Pointer to Pointer (P2P) Packet Descriptors point to a table of P2D descriptors. Data of virtually unlimited size can be transmitted using P2P Packet Descriptors.

P2P descriptors differ from the P2D type in the following ways:

- The P2P PointerType bit must be set.
- The P2P Address field must be aligned on a cache line boundary.
- The Length of the P2P must be an integer multiple of 8 bytes (i.e., an integer multiple of P2D Packet Descriptors). For example, a P2P consisting of a single P2D will be 8 bytes long; a P2P consisting of two P2Ds will be 16 bytes long, etc.

Table 13-6 shows the bit-fields for the P2D Packet Descriptor format.

Table 13-6. Tx Packet Descriptor Pointer-to-Pointer (P2P) Format

63	62	61	60:54	53:40	39:0
EOP	PointerType	RdEx	FBID	Length	Address

Bits	Name	Description
63	EOP	End of Packet. 0: Not last descriptor of packet 1: Last descriptor of packet
62	PointerType	Pointer type = 1 for P2P Packet Descriptors. 0: Pointer-to-Data Type 1: Pointer-to-Pointer Type Note: The PointerType bit must be set for the Packet Descriptor to be interpreted as a P2P.
61	RdEx	Read Exclusive. 0: Do not invalidate the cache line for the packet data memory after it has been read. 1: Invalidate the cache line for the packet data memory after it has been read. This option prevents the cache line from being written to DRAM, saving DRAM bandwidth. Usage: To prevent corrupted packets, the Read Exclusive bit should <u>not</u> be set: <ul style="list-style-type: none">• For multicast/broadcast packets, or• If it is possible for a packet to be retransmitted due to collisions in Half-Duplex networks, or due to packet TX underrun.
60:54	FBID	Free Back ID Setting this to 127 will prevent this descriptor from being freed back at all.
53:40	Length	Packet data length in bytes. Note: for P2P descriptors, this must be a multiple of 8, because Packet Descriptors are 8 bytes long. Specifies the length in bytes of the packet data memory physically addressed by bits [39:0]. A length of zero will simply cause the Descriptor to be sent to the Bucket indicated by FBID without any data fetch.
39:5	Address	Cache-line aligned address. Address points to the physical base address of an array of P2D Packet Descriptors. Address must be cache-line aligned. This address will be used by the Network Interface's DMA to access an array of P2D Packet Descriptors. The length of the array in units of P2D Packet Descriptors is the byte length encoded in bits 53:40 divided by the size of a Packet Descriptor (8 bytes).
4:0	Reserved	Reserved

Note: The PointerType bit must be set for the Packet Descriptor to be interpreted as a P2P.

Note: Pointer-to-Pointer type descriptors can only be used to address Pointer-to-Data type descriptors. Pointer-to-Pointer type descriptors cannot be used to address other pointer-to-pointer type descriptors.

The Free Back ID (FBID) is used to identify the Descriptor as available for future use once the packet it was originally describing has been transmitted. Since a P2P descriptor points to an array of P2D descriptors, it cannot be used by the ingress side of Network Accelerators, which can only recycle P2D descriptors. Ordinarily, a Core Bucket is addressed by P2P FBID which manages these descriptors.

The Read Exclusive (RdEx) bit is set to invalidate the cache line upon which the P2D memory array resides once it has been read.

13.3.6.3

Read Exclusive (RdEx) Bit and Cache Line Invalidation

The Read Exclusive function is designed to increase performance by invalidating a cache line upon transmission instead of allowing the cache line to be evicted by a new allocation. The purpose is to conserve DRAM bandwidth which is otherwise consumed by eviction.

Setting the RdEx bit causes the cache line to be invalidated after it is read from the memory subsystem. The RdEx bit should not be set if packets may be retransmitted. If the packet is retransmitted, however, the cache line is read again from the memory subsystem and, since it had already been invalidated after the initial read, the cache line may have already been used for other purposes. In this case, the retransmitted data will be corrupted, but a correct Ethernet FCS will be added so that there will no indication that the packet is incorrect.

The RdEx bit should not be set if the interface is in half duplex mode. In full duplex mode, the RdEx bit should not be set if the packet might be retransmitted due to a packet underrun condition. The RdEx bit should also not be set if packet is a multicast or broadcast packet that will be sent multiple times to multiple ports.

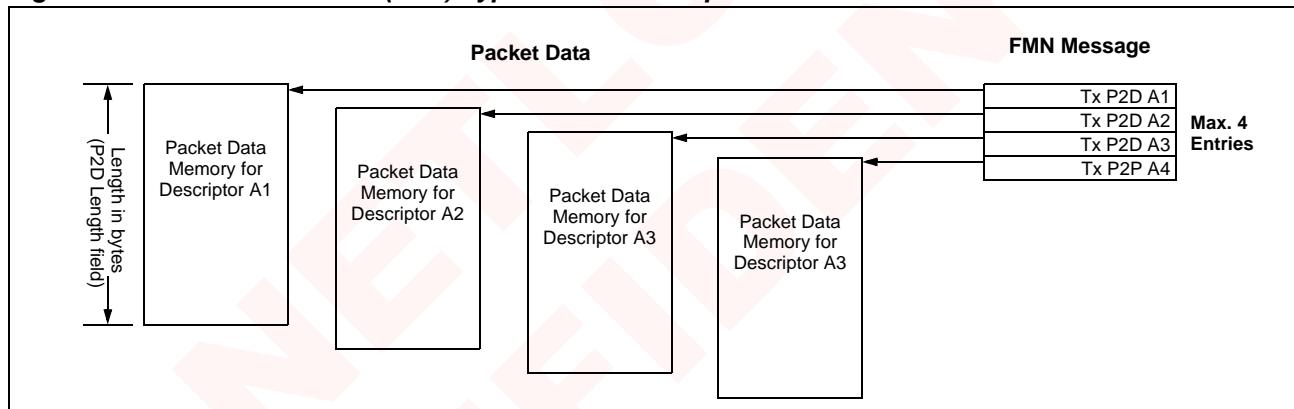
In these situations, if user wishes to invalidate a cache line after transmission, one can do so by freeing up the descriptor back to CPU. The software can then invalidate the cachelines.

13.3.7

Usage of Pointer-to-Data Type Tx Packet Descriptors

[Figure 13-6](#) shows the usage of the Tx Pointer-to-Data (P2D) type Packet Descriptors.

Figure 13-6. Pointer-to-Data (P2D) Type Packet Descriptors

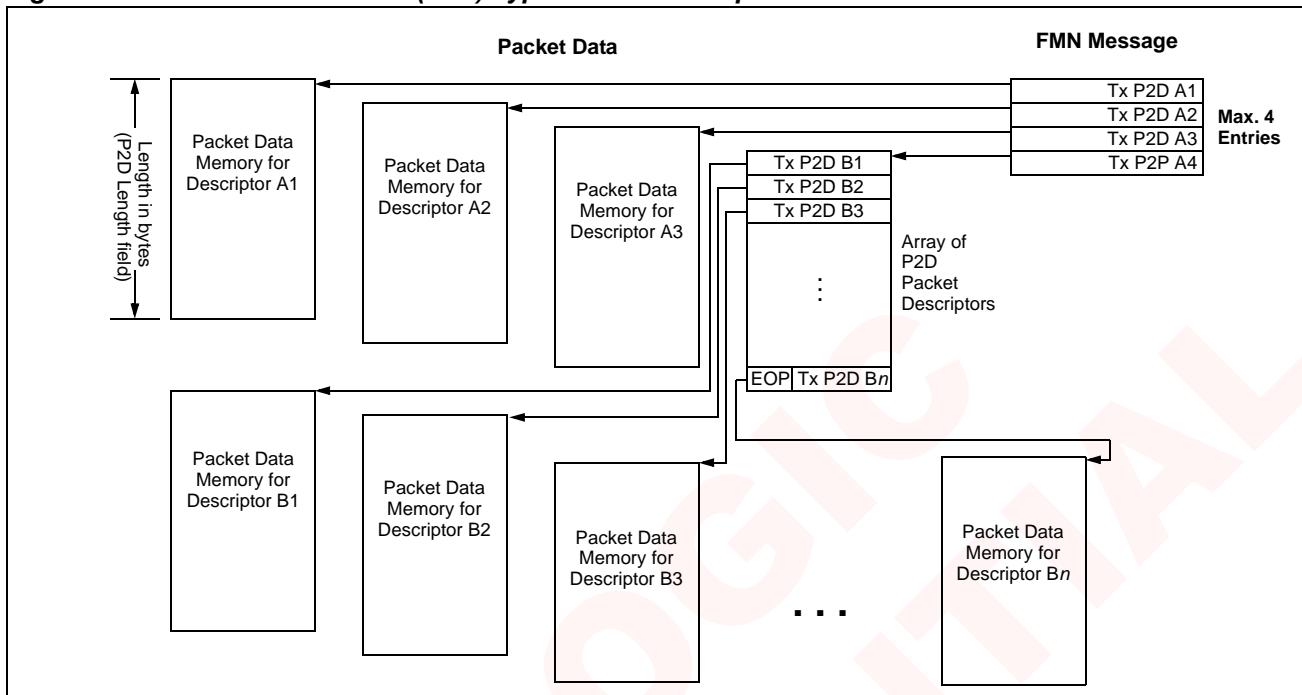


In this example, a total of 4 P2D Tx Packet Descriptors (the maximum number possible) are used to describe the packet data to be sent. Descriptor A1 contains a byte-aligned address which specifies the physical memory location containing the packet data used for constructing the packet to be transmitted, a total of 4 of which comprise the entire packet. The byte-aligned Length and byte-aligned Address fields in each Packet Descriptor are used to characterize the four components of the packet data to be transmitted.

Descriptor A4 has its EOP bit set to signify that this is the last descriptor for this packet.

Since P2D packets can represent up to four components of a packet, packet data need not be contiguous.

[Figure 13-7](#) shows the usage of the Tx Pointer-to-Pointer (P2P) type Packet Descriptors.

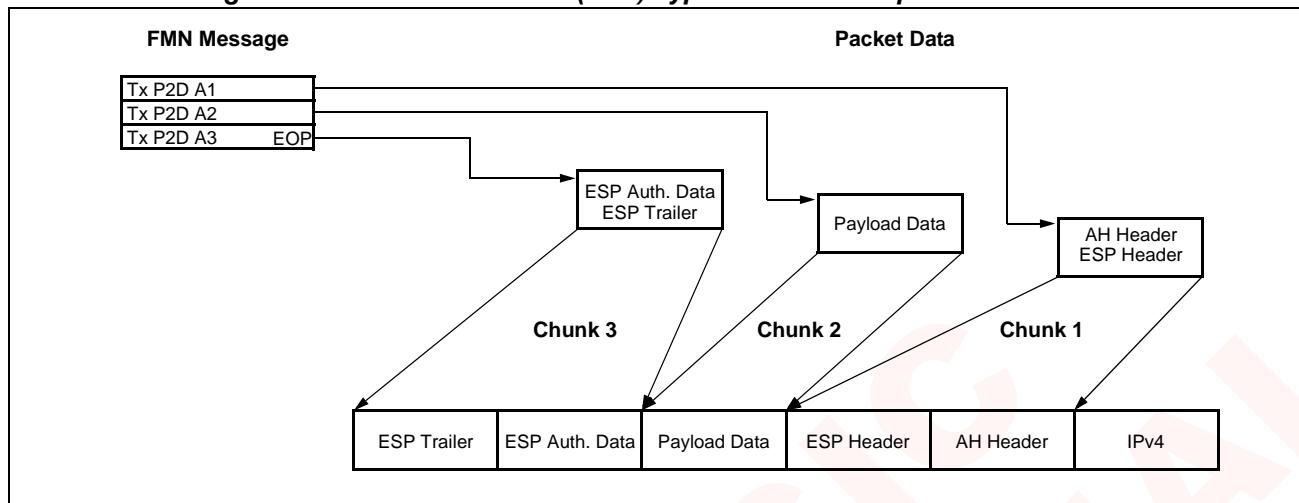
Figure 13-7. Pointer-to-Pointer (P2P) Type Packet Descriptors

An FMN message may contain up to four P2P descriptors. P2P and P2D descriptors may be freely combined in messages.

[Figure 13-7](#) shows a message described by 3 P2D descriptors and one P2P descriptor. The A1, A2 and A3 descriptors are P2D descriptors. The fourth descriptor, A4, is a P2P descriptor. It contains as its address a pointer to the starting address of a series of P2D descriptors. The number of P2D descriptors in the array is specified by A4's Length field divided by 8, the length of a P2D descriptor. The Length field divided by 8 equals the number of P2D descriptors addressed by A4. The last Descriptor in the list sets the EOP bit, indicating it is the last Descriptor.

13.3.8 Using Tx P2D Descriptors to Modify Egress Packets

[Figure 13-8](#) shows an example of performing egress packet modifications using the byte-addressable P2D type Tx Packet Descriptor.

Figure 13-8. Pointer-to-Data (P2D) Type Packet Descriptors

Descriptor A1 addresses a buffer containing the AH and ESP headers, which are the first chunk of data needed to build up the packet. Likewise, the second chunk of data required is the Payload Data, addressed by descriptor A2. The ESP Authentication Data and ESP Trailer are the last chunk of data needed to build the packet, and so must be pointed to by the last descriptor, A3, which also has the EOP bit set signifying that this is the last chunk of data being used to form the packet.

In a similar manner, other fields, such as VLAN tags, could be inserted into packets by using the byte-addressable pointers available in the P2D descriptors.

Note: When performing egress packet modifications in this manner, the first descriptor must resolve to the first data chunk (the chunk closest to the head of the packet data), the second descriptor to the second data chunk, and so on.

13.3.9 Tx Free Back Descriptor

The Free Back Descriptor is a one-Entry Packet Descriptor that is sent by the Network Interfaces upon completion of packet transmission. The destination is specified in the FBID (Free Back ID) field of the Tx Packet Descriptor that initiated processing. The Free Back Descriptor contains pointer information for the now-exhausted Packet Buffer associated with the packet data that has been sent. This allows the Descriptor to return to its original pool of Free Tx Descriptors. (An FBID of 127 will suppress sending a Free Back Descriptor. If the FBID is a P2D, the FBID should generally point to a Free In Descriptor FIFO Bucket.)

Table 13-7. Free Back Packet Descriptor Format

63	62	61	60	59	58	57:54	53:40	39:0
Reserved	P2P	Collision	Bus Error	Underrun	Abort	PortID	Length(0)	Address

Bits	Name	Description
63	Reserved	Reserved
62	P2P	Type of packet 0: P2D 1: P2P
61	Collision	Packet collision Set if an Ethernet collision was detected while the packet was being transmitted in 10/100 half duplex mode. The Network Accelerator retries the packet automatically up to a maximum set by FreeQCarve.MaxUnderRetryCount.
60	BusError	Bus error Set if a bus error (such as ECC error) was detected during DMA fetch of the packet by a Network Accelerator.
59	Underrun	Underrun Set if a data underrun condition arose while a GMAC Network Accelerator was fetching packet data while packet is being transmitted.
58	Abort	Abort Set if packet transmission was aborted for any reason. For example, Abort will be set on Bus Error, and will also be set on Underrun if the retry count is exceeded.
57:54	PortID	Port ID Indicates the port on which the packet was sent out. 0-3: GMAC 0-15: reserved
53:40	Length	Packet data length in bytes (0). This field will be zero for all Free Back Descriptors.
39:0	Address	The free back descriptor sends back the byte address Address points to the physical base address of the exhausted Packet Buffer.

13.3.10 Rx Free In Descriptor

The Free In Descriptor is a one-Entry Packet Descriptor that is sent to the Network Interfaces during initialization to set up the Free Descriptor FIFOs.

Table 13-8. Free In Packet Descriptor Format

63:40	39:5	4:0
Reserved	Address	Reserved

Bits	Name	Description
63:40	Reserved	Reserved
39:5	Address	Cache-line aligned address.
4:0	Reserved	Reserved

13.4 Parser

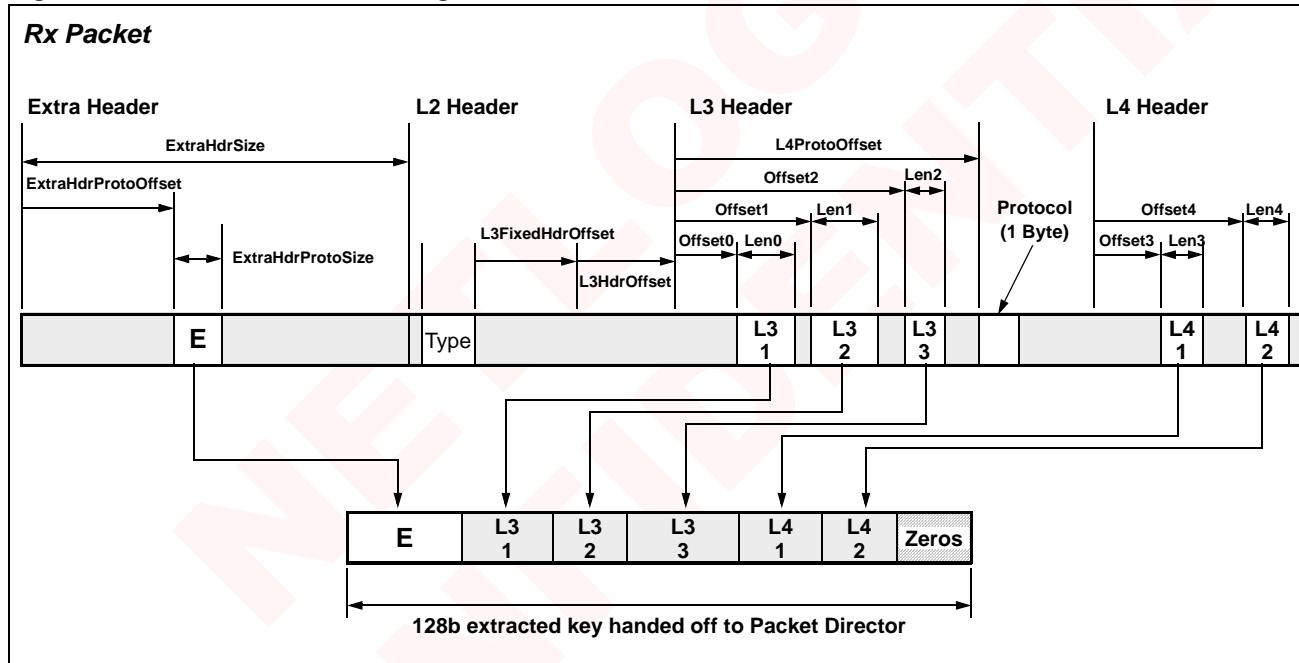
While packet data is being received and transferred to memory, the Parser snoops the arriving packet data and extracts user-definable fields (keys). It concatenates these fields together to form a 128-bit key used by the Packet Director to classify and dispatch the packet. The Parser can be programmed to understand any packet format.

The Parser can extract up to six separate fields, and can identify Layer 3 and Layer 4 headers. It supports up to sixteen Layer 3 Protocol types, and up to eight Layer 4 protocol types. Layer 3 and Layer 4 support includes TCP/UDP and IP checksum verification.

13.4.1 Theory of Operation

[Figure 13-9](#) shows a conceptual view of a packet containing a custom header, with standard L2, L3 and L4 headers. All symbols shown in [Figure 13-9](#) are fields from registers in the Parser block. Up to six fields from the packet can be concatenated together to create the classification key. The key is padded with zeros if the length of the fields is less than 128 bits.

Figure 13-9. Parser — Processing of Packet Header



Custom Header Extraction

The Parser can extract data from custom headers to be included as part of the 128-bit classification key. The `L2Type.ExtraHdrProtoOffset` field (5 bits) and `L2Type.ExtraHdrProtoSize` field (5 bits) allow up to 16 bytes of custom protocol to be included in the classification key. The extracted data, labeled "E" in [Figure 13-9](#), is inserted at the head of the 128-bit classification key.

L2 Header Extraction

The `L2Type.ExtraHdrSize` field (6 bits) indicates the offset to the beginning of the L2 header.

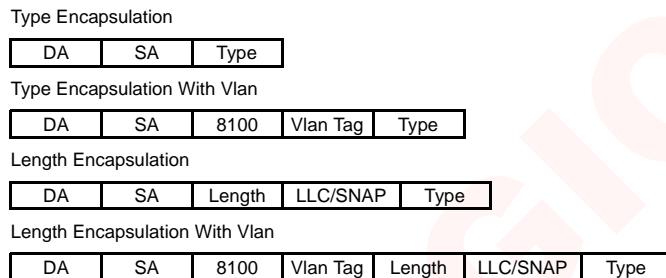
After extracting any specified custom header field, the Parser extracts the L2 header based on the L2 Type (port type). The L2 Type field specifies the L3 protocol (such as IPv4, IPv6, MPLS, ARP, etc.).

The L2 protocol is programmed per port. Each of the 4 GMAC ports are individually programmable via its own [L2Type](#) register.

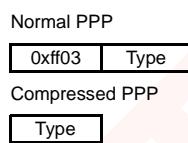
The L2Type.L2Proto field supports the following programming options for L2 protocol per port:

- 0: No key extractions or classification
- 1: Ethernet
- 2: Fixed Header Length — specifies length of L2
- 3: PPP (compressed, and uncompressed with FF03).

If the L2 protocol is programmed to be Ethernet, then the Parser detects one of the following four L2 formats and automatically extracts the L2 Type information from it, as follows:



If the L2 protocol is programmed to be PPP, the Parser detects one of the following two frame formats and automatically extracts the L2 Type information, as follows:



L3 Header Extraction

L2Type.L3FixedHdrOffset is used to find the beginning of the L3 header. Three fields can be extracted from the L3 header based on the L2 Protocol and L2 Type.

The sizes and offsets of the three L3 fields to be extracted are identified for each protocol via a 16-entry content-addressable memory (CAM) named L3CTable. The L3CTable CAM performs exact-match searches, with the last entry reserved for “no match”.

The L2 Protocol field (2-bit programmed value, fixed per port) and the 16-bit L2 Type field from a packet are concatenated and matched against the 16-entry L3CTable CAM, as shown in [Figure 13-10](#). Additionally, if the corresponding L3CTable.PortMask bit is set, then the four-bit L3CTable.Port field is also used during matching.

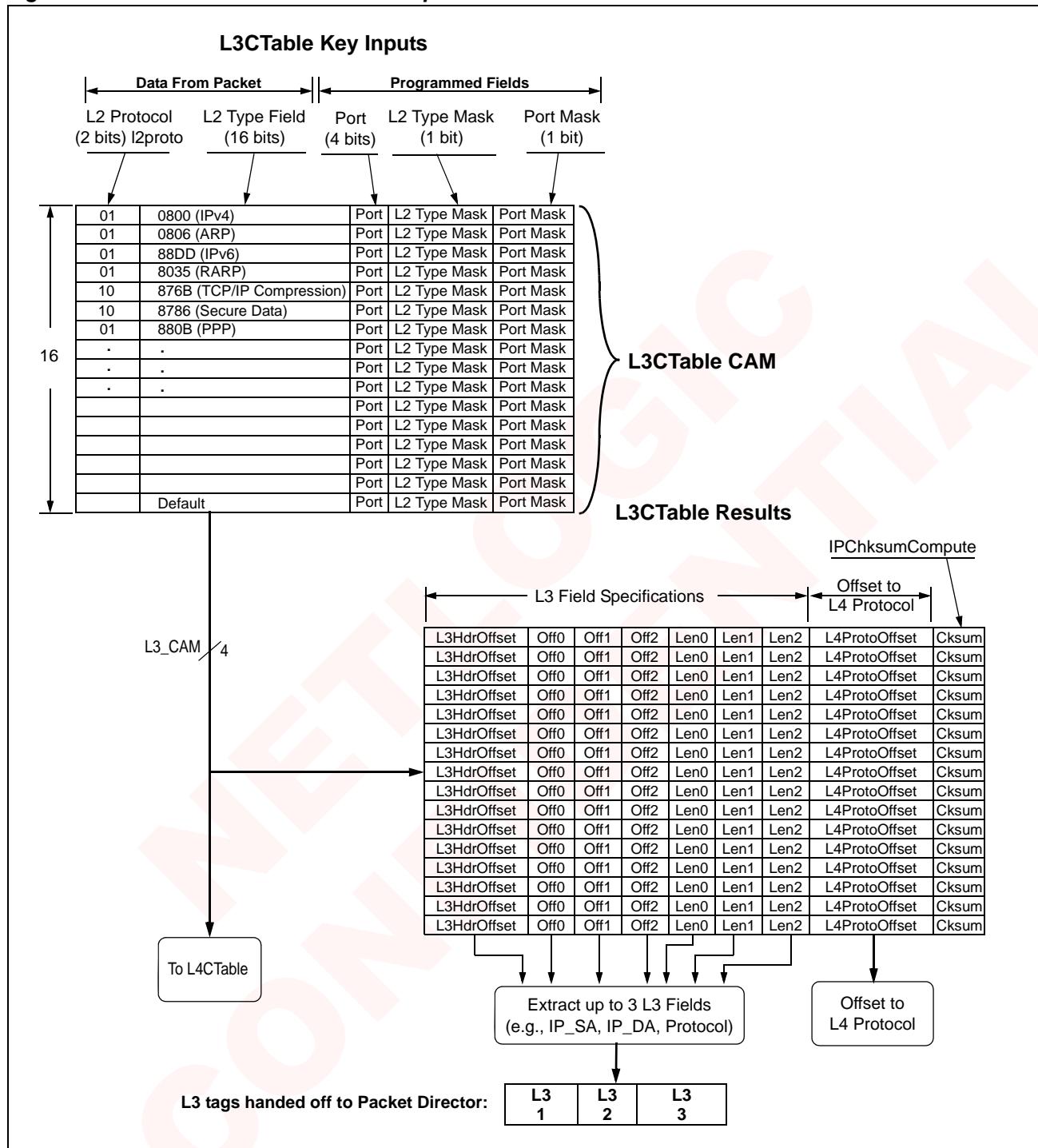
The resulting 4-bit number is labeled L3_CAM. It is used as an offset into the 16-entry table to determine which L3 and L4 fields to extract in order to help build the 128-bit classification key.

The indexed L3 and L4 fields identified in the table are named Len0, Offset0, Len1, Offset1, Len2, Offset2 in [Figure 13-9](#). These offsets and sizes are then used by the Parser to extract the three L3 fields from the packet header. The Parser then concatenates them to the 128-bit classification key being constructed. The L4ProtoOffset field in that table can be used to extract the L4 header, as described below.

The L3_CAM offset can also be used later for packet classification.

If there is not an exact match in L3CTable, the last entry of the CAM is selected as the default entry.

Figure 13-10. Parser — L3CTable Lookup



Once the L3 Fields have been extracted by the Parser, they are placed into the 128-bit classification key, as shown in [Figure 13-11](#). The 8-bit Protocol field in the L3 header can then be used to match into a CAM to obtain the offsets and lengths of the Layer4 fields to extract.

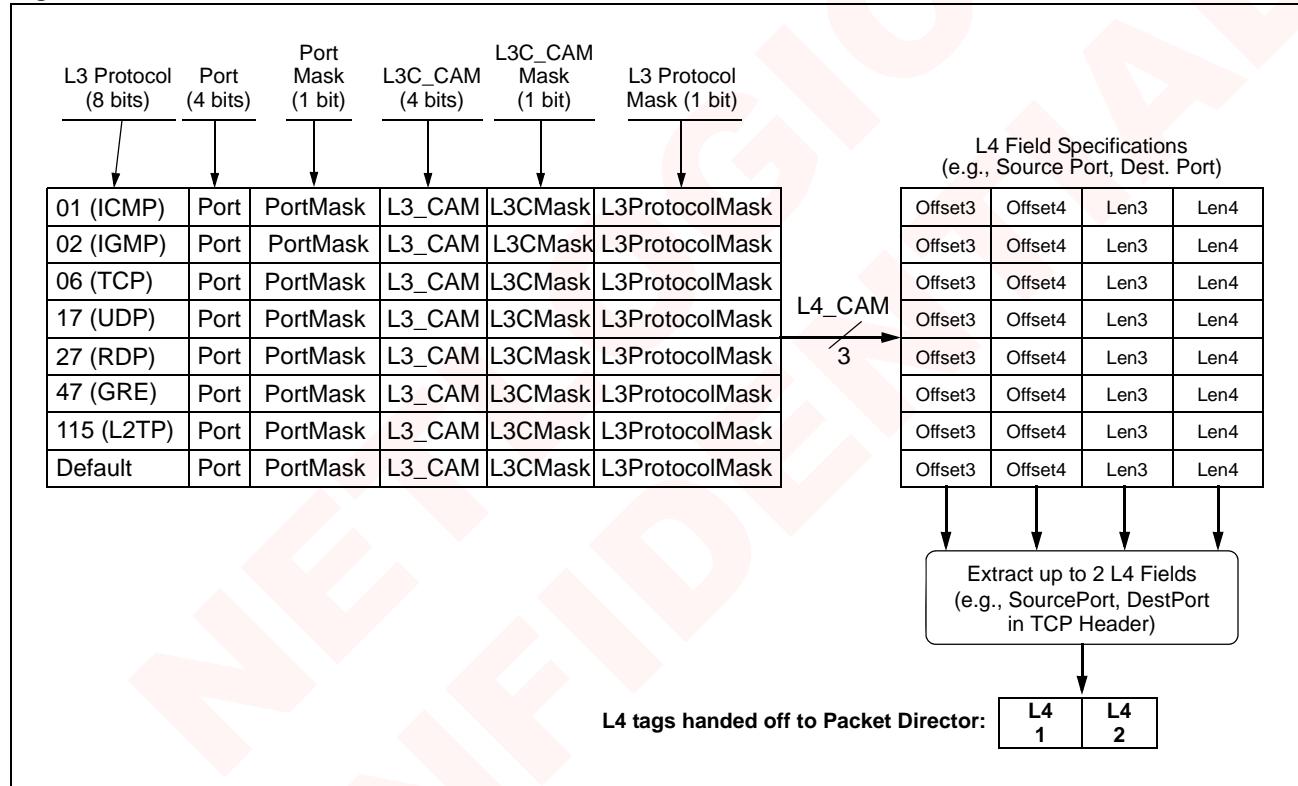
L4 Header Extraction

The Parser is automatically able to extract Layer-4 fields even if variable length IPv4 options are used in a packet. IP Checksum as well as TCP Checksum are also computed on TCP and IP packets, as described in [Section 13.5, "TCP Checksum Hardware Support"](#).

The L3 protocol field (one byte, which specifies TCP, UDP, etc.) is extracted, and looked up in an 8-entry CAM named L4CTable, as shown in [Figure 13-11](#). This field can be used to extract up to two Layer-4 fields: Offset3, Len3, Offset4 and Len4. The offset into this second 8-entry CAM search is a three-bit number that is referred to as L4_CAM.

If there is not an exact match in L4CTable, the last entry of the CAM is selected as the default entry.

Figure 13-11. Parser — L4CTable L4 Field Extraction

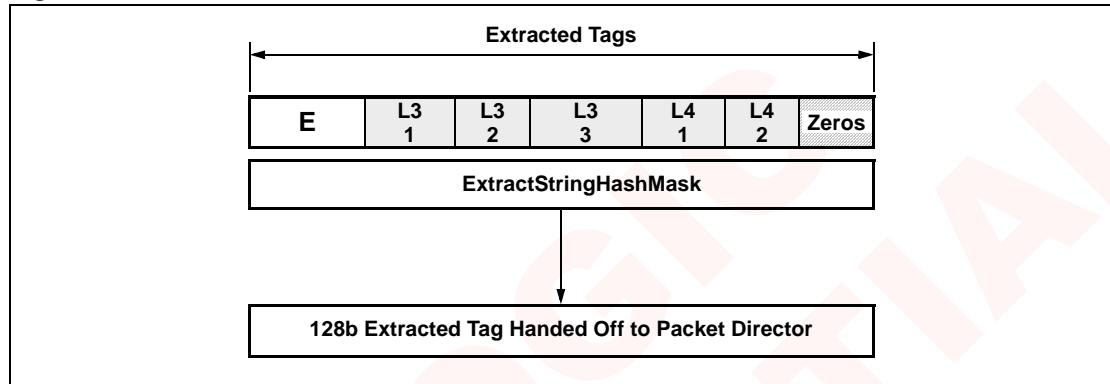


Classification Vector

The result of the Parser is a 128-bit classification vector that is passed to the Packet Director. It contains any Extra header information, up to three L3 fields, and up to two L4 fields. If it is less than 128 bits long, it is padded with zeros.

Finally, the classification vector is masked with the 128-bit ExtractStringHashMask register, as shown in [Figure 13-12](#).

Figure 13-12. Classification Vector



13.4.2 Parser Registers

Table 13-9. Parser Register Summary

Name	Register ID	Address Offset	Notes
L2Type	0x0F0 – 0x0FF	0x3C0 – 0x3FC	Unique register for each port: 1 for the 4 GMACs; Configures port settings for extra fixed header details, fixed L2 header size and fixed Layer 2 protocol mapping that the Parser will apply to the port.
ParserConfig	0x100	0x400	Determines whether packet direction will be based on a ternary CAM (TCAM), a hash table lookup, the {L3, L4} protocol, or the seven msbs of the Parser key.
ParseDepth	0x101	0x404	Maximum depth into the packet for parsing.
ExtractStringHashMask0	0x102	0x408	Mask value for Extract_String[31:0].
ExtractStringHashMask1	0x103	0x40C	Mask value for Extract_String[31:0].
ExtractStringHashMask2	0x104	0x410	Mask value for Extract_String[95:64].
ExtractStringHashMask3	0x105	0x414	Mask value for Extract_String[127:96].
L3_L4_PROTO_MASK	0x106	0x418	Provides L3 Protocol (Ethertype) and L4 Protocol global masks
L3CTable	0x140 – 0x15F	0x500-0x57C	16-entry content-addressable memory (CAM); performs exact-match searches, with the last entry reserved for “no match”.
L4CTable	0x160 – 0x16F	0x580 – 0x5BC	8-entry CAM used to extract up to two Layer-4 fields.
L3CTableMasks	0x1E0 – 0x1E3	0x780 – 0x78C	
L4_CTABLE_Masks	0x1E4 – 0x1E7	0x790 – 0x79C	
HASH7_BASE_MASK	0x1E8 – 0x1EB	0x7A0 – 0x7AC	Provides a way to segment the Translation Table.
HASH7_BASE_MASK_CAM_KEYS	0x1EC – 0x1EF	0x7B0 – 0x7BC	

13.4.2.1 L2Type

Register ID: 0x0F0 – 0x0FF

Address Offset: 0x3C0 – 0x3FC (See Note:)

Each port has an L2Type configuration register. L2Type is used to configure three port settings:

- Extra fixed header details, if required
- Extra fixed L2 header size if required
- L2Proto, the fixed Layer 2 protocol mapping that the Parser will apply to the port.

To avoid possible GMAC port hang these registers should be set up before the incoming traffic is enabled. Also, these register should not be read during traffic flow. If read access of these registers is required, software can maintain a shadow copy of the contents for read purposes.

Bits	Name	Description	R/W	Reset
31:26	ExtraHdrProtoSize	Extra header protocol size. Specifies the size of the Extra Header Protocol field to extract. Up to 16 bytes of custom protocol can be used for classification. The beginning byte is specified by ExtraHdrProtoOffset. The indicated bytes are inserted at the head of the 128-bit classification key generated by the Parser. If set to 0, no Extra Header Protocol data is inserted in the classification key.	R/W	0
25:20	ExtraHdrProtoOffset	Extra header protocol offset. Offset of the Extra Header Protocol field to be extracted from the custom header. Up to 16 bytes of custom protocol can be used for classification. The indicated bytes are inserted at the head of the 128-bit classification key generated by the Parser.	R/W	0
19:14	ExtraHdrSize	Extra Header Offset Size in front of L2 Header. This field indicates the size of the Extra Header that must be skipped to reach the L2 Header. This field should be set to zero if no extra header is present. Using ExtraHdrProtoSize and ExtraHdrProtoOffset, it is still possible to extract fields starting at the first byte of the packet, even if ExtraHdrSize=0	R/W	0
13:8	ProtoOffset	If L2Proto == 2, then this field tells where to pick up the L3 protocol field.	R/W	Not Reset
7:2	L3FixedHdrOffset	If L2Proto == 2, then this field gives the size of the Fixed Header. If L2Proto == 1 (Ethernet) or If L2Proto == 3 (PPP) then this field gives the size of a constant prefix after the L2 Header	R/W	Not Reset
1:0	L2Proto	Protocol 0: No L2 classification 1: Ethernet 2: Fixed Header Length (PPP) only uncompressed PPP with FF03 or proprietary header 3: PPP (compressed and uncompressed with FF03)	R/W	Not Reset

Note: This register is not shared among the GMACs. The unique addresses are:

- GMAC0 (or Optional – XAUI_0) - 0x0C3C0
- GMAC1 - 0x0D3C0
- GMAC2 - 0x0E3C0
- GMAC3 - 0x0F3C0
- GMAC4 (or Optional – XAUI_1) - 0x203C0
- GMAC5 - 0x213C0
- GMAC6 - 0x223C0
- GMAC7 - 0x233C0

13.4.2.2 ParserConfig

Register ID: 0x100

Address Offset: 0x400

See [Section 13.6, “Packet Director”.](#)

All four SGMII interfaces in a GMAC quad, the RGMII interface in GMAC_Quad_0, and the optional XAUI ports share this register in common.

(See [Section 13-3, “Network Accelerator Register Layout”.](#))

Bits	Name	Description	R/W	Reset
31:25	Reserved	Reserved	R	0
24:15	Unclassified Mask	Parser's Unclassified bit in the status field of the Rx Packet Descriptor is computed based on 10 different conditions described by this bit field. 24: L3_CAM_HIT 23: L4_CAM_HIT 22: Extra Header Extraction Valid 21: L3CTable Extract 0 Valid 20: L3CTable Extract 1 Valid 19: L3CTable Extract 2 Valid 18: L4CTable Extract 0 Valid 17: L4CTable Extract 1 Valid 16: L2 Header Done 15: L3 Header Done	R/W	0
14:8	CRCHashPoly	This is the 7-bit CRC polynomial.		0x7F
7:4	Reserved	Reserved	R	0
3	UsePort	Use Port Bind Mode. ^a Each GMAC (or optional XAUI port) vectors to a corresponding offset in the translation table. Only the bottom four locations of the translation table are used for GMACs, and the rest of the entries are unused.	R/W	0
2	UseTCAM	This bit must be set to enable the 4-entry 128-bit TCAM. ^{a, b, c}	R/W	0
1	UseHash	A seven-bit CRC HASH is used to lookup the translation table. ¹	R/W	0
0	UseProto	{L3_CAM, L4_CAM} is used to lookup the translation table. ¹	R/W	0

- a. UsePort, UseHash, and UseProto are mutually exclusive. UseCAM can be used in conjunction with one of UsePort, UseHash, or UseProto because UseCAM takes precedence, and others are only invoked if there is no TCAM hit. The TCAM is generally used to send control packets to a set of Threads; other packets go through the hashtable. If UseHASH, UseProto, and UsePort are all zero, then by default the 7 msbs of the 128-bit key are used to look up the Translate table.
- b. When the Network Accelerator’s Director is configured for HASH mode (to support flow-based packet distribution), a hardware-based CRC7 HASH algorithm is used. This HASH algorithm uses a 7-bit programmable polynomial set in ParserConfig to compute the translation table index.
- c. A separate CRC32 Hash is placed in the Prepad field named CRC32, for re-use by software. Since this CRC result is derived from the full key extracted by the parser, software may use it for fine-grain flow determination by applying additional XOR and shift operations. Further details and example index simulation code are available from NetLogic on request.

13.4.2.3 ParseDepth

Register ID: 0x101

Address Offset: 0x404

All four SGMII interfaces in a GMAC quad, the RGMII interface in GMAC_Quad_0, and the optional XAUI ports share this register in common.

(See [Section 13-3, “Network Accelerator Register Layout”](#).)

Maximum depth into the packet for parsing. The length of the Hash7 polynomial is set by POLY_MASK with a maximum limit of 7. Various other polynomial properties can be used if we keep this programmable.

Bits	Name	Description	R/W	Reset
31:12	Reserved	Reserved	R	0
11:5	POLY_MASK	Ideally should be $128 - 2^N = 127, 126, 124, 120, 112, 96$ or 64.	R/W	0x7F
4:0	ParseDepth	Maximum depth into the packet for parsing. Units specified in increments of 16 bytes. Value must be less than $((4 * \text{RegularSize})/16) - 10$. The maximum value that should be programmed is 15. Software must make sure that this value is less than 4 descriptors can hold, so we can fit the entire classification into the first FMN message.	R/W	0x1E

A 128-bit mask is applied to the extracted vector for computing the Hash7 and Hash32. The next four registers are for programming this mask.

13.4.2.4 ExtractStringHashMask0

Register ID: 0x102

Address Offset: 0x408

All four SGMII interfaces in a GMAC quad, the RGMII interface in GMAC_Quad_0, and the optional XAUI ports share this register in common.

(See [Section 13-3, “Network Accelerator Register Layout”](#).)

Bits	Name	Description	R/W	Reset
31:0	ES_MASK0	Mask value for Extract_String[31:0]. Mask applied to Parser 128-bit key. Can be used to mask out arbitrary bit fields.	R/W	0xFFFFFFFF

13.4.2.5 ExtractStringHashMask1

Register ID: 0x103

Address Offset: 0x40C

All four SGMII interfaces in a GMAC quad, the RGMII interface in GMAC_Quad_0, and the optional XAUI ports share this register in common.

(See [Section 13-3, “Network Accelerator Register Layout”](#).)

Bits	Name	Description	R/W	Reset
31:0	ES_MASK1	Mask value for Extract_String[63:32]. Mask applied to Parser 128-bit key. Can be used to mask out arbitrary bit fields.	R/W	0xFFFFFFFF

13.4.2.6 ExtractStringHashMask2

Register ID: 0x104

Address Offset: 0x410

All four SGMII interfaces in a GMAC quad, the RGMII interface in GMAC_Quad_0, and the optional XAUI ports share this register in common.

(See Section 13-3, “Network Accelerator Register Layout”.)

Bits	Name	Description	R/W	Reset
31:0	ES_MASK2	Mask value for Extract_String[95:64]. Mask applied to Parser 128-bit key. Can be used to mask out arbitrary bit fields.	R/W	0xFFFFFFFF

13.4.2.7 ExtractStringHashMask3

Register ID: 0x105

Address Offset: 0x414

All four SGMII interfaces in a GMAC quad, the RGMII interface in GMAC_Quad_0, and the optional XAUI ports share this register in common.

(See Section 13-3, “Network Accelerator Register Layout”.)

Bits	Name	Description	R/W	Reset
31:0	ES_MASK3	Mask value for Extract_String[127:96]. Mask applied to Parser 128-bit key. Can be used to mask out arbitrary bit fields.	R/W	0xFFFFFFFF

13.4.2.8 L3_L4_PROTO_MASK

Register ID: 0x106

Address Offset: 0x418

All four SGMII interfaces in a GMAC quad, the RGMII interface in GMAC_Quad_0, and the optional XAUI ports share this register in common.

(See Section 13-3, “Network Accelerator Register Layout”.)

This register provides L3 Protocol (Ethertype) and L4 Protocol global masks. This register allows the L3 protocol to be less than 16 bits and the L4 protocol to be less than 8 bits.

Bits	Name	Description	R/W	Reset
31:24	Reserved	Reserved	R	
23:16	L4_MASK	Bit Mask for L4 Protocol	R/W	0x3
15:0	L3_MASK	Bit Mask for L3 Protocol	R/W	0x4

13.4.2.9 L3CTable

Register ID: 0x140 – 0x15F

Address Offset: 0x500 – 0x57C

All four SGMII interfaces in a GMAC quad, the RGMII interface in GMAC_Quad_0, and the optional XAUI ports share this register in common.

(See [Section 13-3, “Network Accelerator Register Layout”](#).)

Bits	Name	Description	R/W	Reset
Result Fields				
Word 1 (Odd Words i.e. 0x141, 0x143,)				
31:25	Offset0	Offsets and lengths of 3 different words that can be extracted from the L3 header.	R/W	0
24:21	Len0		R/W	0
20:14	Offset1		R/W	0
13:10	Len1		R/W	0
9:4	Offset2		R/W	0
3:0	Len2		R/W	0
Word 0 (Even Words i.e. 0x140, 0x142,)				
31:26	L3HdrOffset	L3 header offset from end of L2 header	R/W	0
25:20	L4ProtoOffset	L4 protocol field	R/W	0
19	Reserved	Reserved	R	0
18	IPChksumCompute	IP Checksum Compute: 0: Do not compute the checksum on the packet. 1: Treat the packet as an IP packet and compute the IP checksum.	R/W	0
L2 Key for the CAM				
17:16	L2Proto	These fields are compared during CAM lookup.	R/W	0
15:0	L3ProtoKey		R/W	0

13.4.2.10 L4CTable

Register ID: 0x160 – 0x16F

Address Offset: 0x580 – 0x5BC

All four SGMII interfaces in a GMAC quad, the RGMII interface in GMAC_Quad_0, and the optional XAUI ports share this register in common.

(See Section 13-3, “Network Accelerator Register Layout”.)

Bits	Name	Description	R/W	Reset
Word 1 (Odd Words i.e. 0x161, 0x163,)				
31:27	Reserved	Reserved	R	0
26:21	Offset3	These are offsets and lengths of 2 different words that can be extracted from the L4 header.	R/W	0
20:17	Len3		R/W	0
16:11	Offset4		R/W	0
10:7	Len4		R/W	0
6:1	Reserved	Reserved	R	0
0	Reserved	Reserved	R	0
Word 0 (Even Words i.e. 0x160, 0x162,)				
Key For the CAM				
7:0	L4ProtoKey	These fields are compared during CAM lookup.	R/W	0

13.4.2.11 L3CTableMasks

Register ID: 0x1E0 – 0x1E3

Address Offset: 0x780 – 0x78C

All four SGMII interfaces in a GMAC quad, the RGMII interface in GMAC_Quad_0, and the optional XAUI ports share this register in common.

(See [Section 13-3, “Network Accelerator Register Layout”](#).)

The L3 CAM uses {ETHER_TYPE[15:0], L2_PROTO[1:0]} and {Port[3:0]} as keys. They can be masked off with individual masks specified in this register. See [Figure 13-10](#) for usage.

Bits	Name	Description	R/W	Reset
31:30	Reserved	Reserved		
29	L3_P_MASK	ENTRY 3*N (N = lowest 2 bits of index)	R/W	1
28	PortMask		R/W	0
27:24	Port		R/W	0
23:22	Reserved	Reserved		
21	L3_P_MASK	ENTRY 2*N	R/W	1
20	PortMask		R/W	0
19:16	Port		R/W	0
15:14	Reserved	Reserved		
13	L3_P_MASK	ENTRY 1*N	R/W	1
12	PortMask		R/W	0
11:8	Port		R/W	0
7:6	Reserved	Reserved		
5	L3_P_MASK	ENTRY 0*N	R/W	1
4	PortMask		R/W	0
3:0	Port		R/W	0

13.4.2.12 L4_CTABLE_Masks

Register ID: 0x1E4 – 0x1E7

Address Offset: 0x790 – 0x79C

All four SGMII interfaces in a GMAC quad, the RGMII interface in GMAC_Quad_0, and the optional XAUI ports share this register in common.

(See Section 13-3, “Network Accelerator Register Layout”.)

The L3 CAM uses {PROTO[7:0]} and {Port[3:0]} and {L3_CAM_ADDR[3:0]} as the keys. They can be masked off with individual masks. See [Figure 13-11](#).

Bits	Name	Description	R/W	Reset
31:27	Reserved	Reserved	R	
26	L4_P_MASK	ENTRY 1*N (N = lowest bit of index)	R/W	1
25	L3_P_MASK		R/W	0
24	PortMask		R/W	0
23:20	L3_ADDR		R/W	0
19:16	Port		R/W	0
15:11	Reserved		R	
10	L4_P_MASK	ENTRY 0*N	R/W	1
9	L3_P_MASK		R/W	0
8	PortMask		R/W	0
7:4	L3_ADDR		R/W	0
3:0	Port		R/W	0

13.4.2.13 HASH7_BASE_MASK

Register ID: 0x1E8 – 0x1EB

Address Offset: 0x7A0 – 0x7AC

All four SGMII interfaces in a GMAC quad, the RGMII interface in GMAC_Quad_0, and the optional XAUI ports share this register in common.

(See [Section 13-3, “Network Accelerator Register Layout”](#).)

This register provides a way to segment the Translation Table so that various sections are based on the {port, L3, L4, Unclassified} key. This is achieved with a separate 8-deep HASH_CAM. The CAM keys are port, L3, L4, and Unclassified. If there is no CAM hit then it defaults to the last entry. The output from this table is a HASH_BASE and HASH_MASK. The New Translate Table Address is TableAddress = (HASH7 & HASH_MASK) | HASH_BASE. By choosing suitable values for HASH_BASE and HASH_MASK, it is possible to get the effect of segmenting the Translation Table.

Bits	Name	Description	R/W	Reset
31	<i>Reserved</i>	ENTRY 1*N (N = lowest bit of index)	R	
30:24	HASH7_BASE		R/W	0
23	<i>Reserved</i>		R	
22:16	HASH7_MASK		R/W	0x7F
15	<i>Reserved</i>	ENTRY 0*N	R	
14:8	HASH7_BASE		R/W	0
7	<i>Reserved</i>		R	
6:0	HASH7_MASK		R/W	0x7F

13.4.2.14 HASH7_BASE_MASK_CAM_KEYS

Register ID: 0x1EC – 0x1EF

Address Offset: 0x7B0 – 0x7BC

All four SGMII interfaces in a GMAC quad, the RGMII interface in GMAC_Quad_0, and the optional XAUI ports share this register in common.

(See Section 13-3, “Network Accelerator Register Layout”.)

Bits	Name	Description	R/W	Reset
31	uncla_MASK	ENTRY 1*N (N = lowest bit of index)	R/W	0
30	L4_P_MASK		R/W	0
29	L3_P_MASK		R/W	0
28	PortMask		R/W	0
27	unclassified		R/W	0
26:24	L4_ADDR		R/W	0
23:20	L3_ADDR		R/W	0
19:16	Port		R/W	0
15	uncla_MASK	ENTRY 0*N	R/W	0
14	L4_P_MASK		R/W	0
13	L3_P_MASK		R/W	0
12	PortMask		R/W	0
11	unclassified		R/W	0
10:8	L4_ADDR		R/W	0
7:4	L3_ADDR		R/W	0
3:0	Port		R/W	0

13.5 TCP Checksum Hardware Support

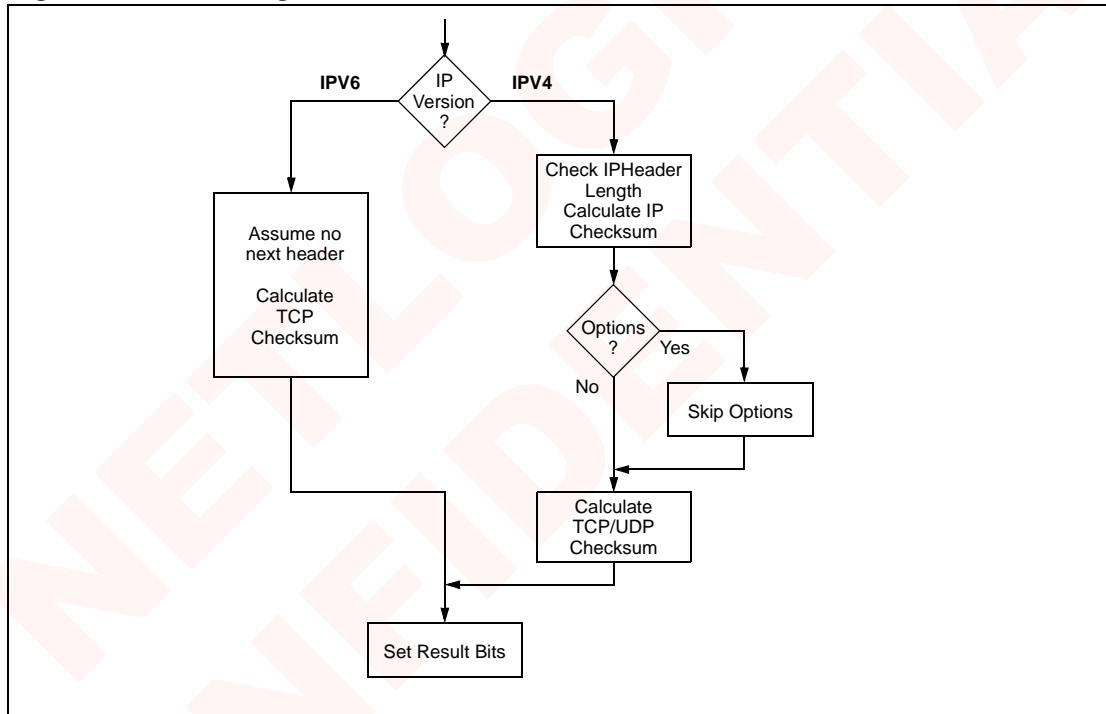
The Parser hardware checksum is relevant to all L2Type.L2Proto options except 00 (no classification).

Each Parser has an L3CTable which is a 16-entry CAM. (See [L3CTable](#) above.) One half of the L3CTable is the key (L2Proto and L3ProtoKey). The second part is the result for each table match.

In the result portion of the L3CTable is a bit IPChksumCompute, which tells the Parser to compute the IPV4 and TCP/UDP Header checksums. Software can use this bit to determine if it should look at the checksum result bits set when the packet is delivered. The packet is delivered with a prepended header that includes IP_chksum_valid and TCP_chksum_valid bits.

The actions of the Parser in setting the checksum result bits is shown in [Figure 13-13](#).

Figure 13-13. Setting Checksum Result Bits



First, the Parser checks the IP version number. For IPv4, the Parser checks the IP header length to see if it should skip Option fields when calculating the TCP/UDP checksum. For IP Header checksum, the Options are not skipped. The options are skipped only while calculating the TCP checksum.

For IPv6, the Parser performs the checksum calculation assuming a single IPv6 header, but software must look for “next header” indication in the packet header to determine if the checksum result for TCP is valid.

13.6 Packet Director

Packet direction is set by the [ParserConfig](#) register which determines whether packet direction will be based on a ternary CAM and one of the following:

- A hash table lookup
- The {L3, L4} protocol
- The seven msbs of the Parser key (see [Figure 13-14](#) and [Table 13-10](#)).

Figure 13-14. Packet Director

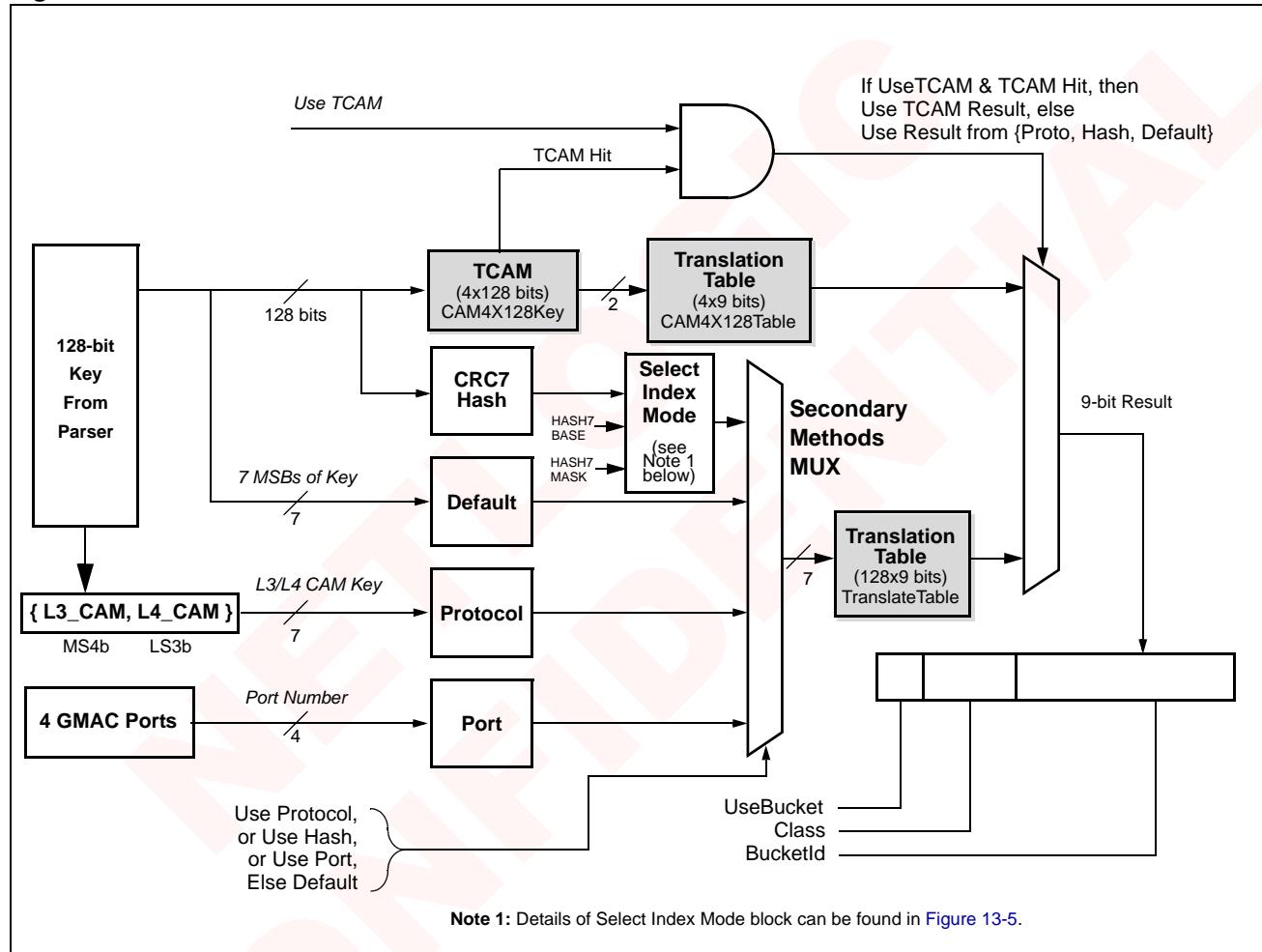
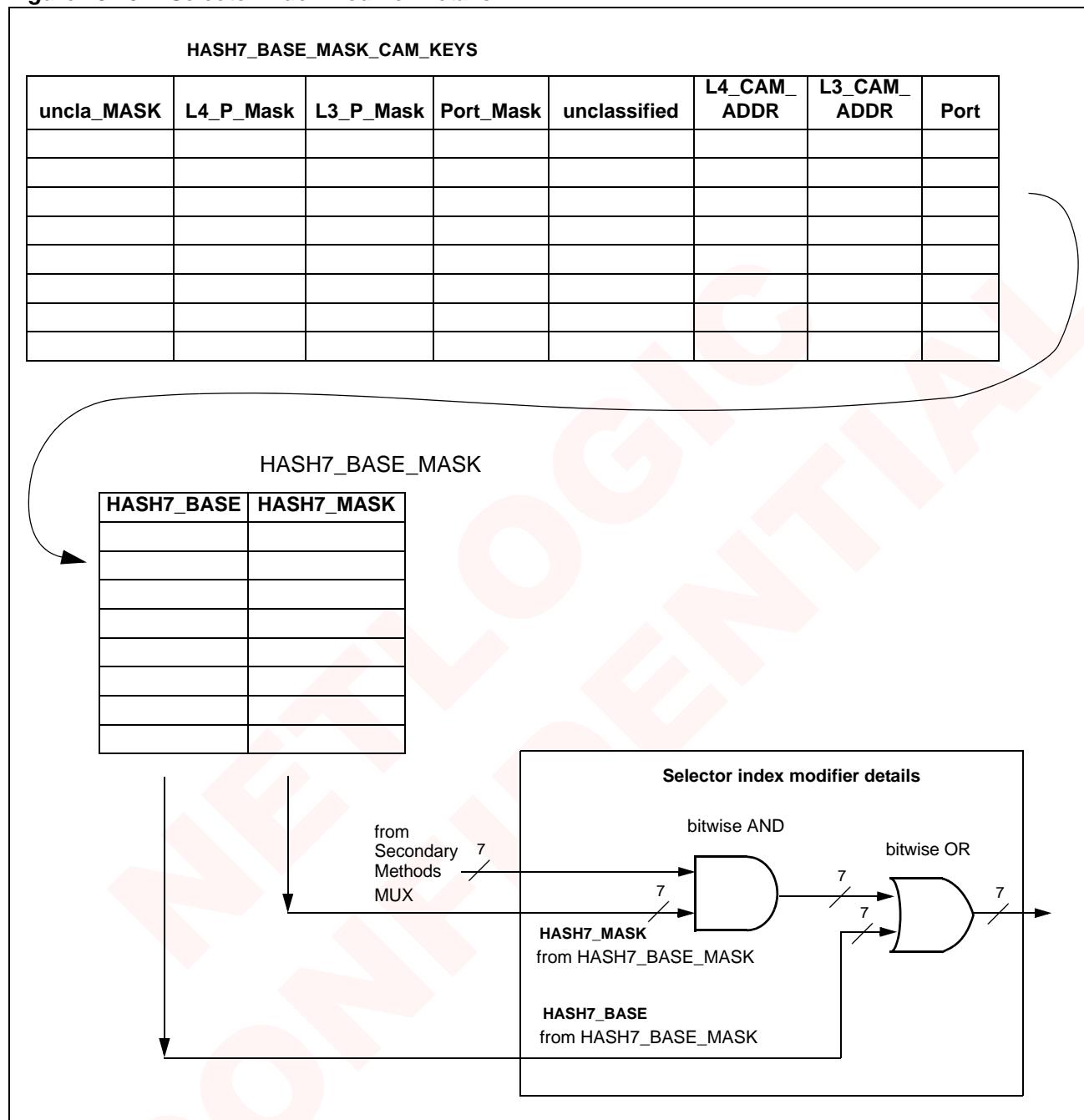


Table 13-10. Packet Director Classification Modes

For this Director Mode	Enable This Classification Mechanism:				
	Most Significant 7 bits of Key Field	Port-Based ID HW	L3 / L4 Protocol-Based ID HW	CRC7 Hash Engine	TCAM
Default (Most Significant 7 bits of 128-bit key field)	ENABLE	DISABLE	DISABLE	DISABLE	DISABLE
Classification by Ingress Port	DISABLE	ENABLE	DISABLE	DISABLE	DISABLE
Classification by L3/L4 Protocol	DISABLE	DISABLE	ENABLE	DISABLE	DISABLE
Classification by Hash	DISABLE	DISABLE	DISABLE	ENABLE	DISABLE
TCAM Exact Match, then MS 7 bits of 128-bit key	ENABLE	DISABLE	DISABLE	DISABLE	ENABLE
TCAM Exact Match, then by-Port classification	DISABLE	ENABLE	DISABLE	DISABLE	ENABLE
TCAM Exact Match, then by-L3/L4 Protocol classification	DISABLE	DISABLE	ENABLE	DISABLE	ENABLE
TCAM Exact Match, then classification by Hashing	DISABLE	DISABLE	DISABLE	ENABLE	ENABLE

Figure 13-15. Selector Index Modifier Details



The Packet Director contains a 128x4-entry ternary CAM (TCAM). The TCAM allows pattern matching with the use of “don’t care” conditions. “Don’t cares” act as wild cards during a search, and are useful for implementing longest-prefix-match searches in routing tables.

Each entry of the TCAM indexes a 128-bit mask that specifies which bits to compare, and which bits are “don’t care.” The main purpose of the TCAM is to quickly identify control packets. The UseTCAM bit in the [ParserConfig\[2\]](#) register indicates whether to use the TCAM for classification.

In addition to the TCAM, there is a 128x9-bit translation table, which takes as input a seven-bit selector index. See Selector Index Modifier details in [Figure 13-14](#). The seven-bit selector input can come from one of the following sources, as determined by the indicated fields of [ParserConfig](#):

- The input port (UsePort)
- A CRC7 hash on the 128-bit parser key (UseHash)
- A 7-bit value created by concatenating the L3_CAM and L4_CAM offsets (UseProto), or
- The 7 most-significant bits of the 128-bit key (Default).

Values programmed in the [ParserConfig](#) register determine the Packet Director classification modes. If UseTCAM is not set, or if it is set but there is a miss in the 4-entry TCAM, then one of the secondary choices is selected. The secondary choices are the CRC7 (UseHash), the input port (UsePort), the concatenation of the {L3, L4} protocol (UseProto), or the default method which uses the 7 most significant bits of the key.

Note: UsePort, UseHash, UseProto, and the Default (7 msbs) method mutually exclusive. UseCAM is used in conjunction with one of those other four because UseCAM takes precedence, and others are only invoked if there is no TCAM hit. See [Table 13-10](#).

When using the TCAM, note that any packet with an extracted key that will match any entry in the TCAM will also match on the entry “Value=00, Mask=00.” Thus, no 00:00 TCAM entry should be created. If it is intended that a given TCAM entry not be used, it should be defined as “Value=01, Mask=00.”

The result of the Packet Director is a 9-bit value that specifies to the Packet Distribution Engine (PDE) where to deliver the packet message. The nine bits include:

- 2 bits of packet classification (always used)
- 6 bits that indicate a Bucket ID (only used if UseBucket is set)
- 1 UseBucket bit

The UseBucket bit specifies whether the PDE should use the Class distribution method (UseBucket==0), or should send the packet message only to the specific Bucket ID (UseBucket not equal to 0). See the [CAM4x128Table Register](#).

If UseBucket==0, the PDE will perform Class-based packet distribution, which provides for round-robin scheduling of packets among Threads. The ClassID field specifies which of the four Classes should determine the Packet Descriptor’s destination. Each Class contains a 64-bit PDE_MASK indicating the set of CPU buckets that are allowed to receive these messages. The eligible set of buckets is a bitwise AND of PDE_MASK and the set of buckets that have enough credits. The next Bucket is selected using a two-level hierarchical round-robin algorithm.

If UseBucket==1, the message will attempt to go only to the specified Bucket. The ClassID field is still used to allocate the Packet Descriptor to one of the four Class FIFOs. The PDE will only send the Packet Descriptor to the specified Bucket (PDE_MASK is ignored).

The classification vector can also be pre-pended to the Packet in memory (this pre-pend process is called “Prepad”).

13.6.1 TCAM

The TCAM is a 128-bit wide 4-entry Ternary Content Addressable Memory. If enabled, the TCAM receives the 128-bit Parser key and matches it against each of its 4 entries. Each entry contains a 128-bit mask which specifies which bits to compare, and which bits are “don’t care” (a very specific definition of which is described in Section 13.6.1.1).

If enabled, the TCAM method takes precedence over the secondary methods. If the UseTCAM bit of the [ParserConfig\[2\]](#) register is set to ‘0’, then the TCAM is not used, and packet

distribution is determined by one of the secondary methods described in the following sections. (See “[ParserConfig](#)” on page 449.)

If the UseTCAM bit is set, then the Packet Director will attempt to find a match for the 128-bit Parser key in the TCAM as follows. The TCAM matches the Parser key against each of its four CAM4x128Key.CAMKey fields. (See [CAM4x128Key Register](#).) Each of the four entries contains a 128-bit mask (CAM4x128Key.CAMMask) Bits set in this mask register to ‘1’ force a comparison for that bit position; bits cleared to ‘0’ indicate a “don’t care” condition for that bit position.

If there is a TCAM hit, then the index of the hit is used to specify the criteria for selecting the destination Bucket via the [CAM4x128Table Register](#). If there is a TCAM miss, then the selected secondary method is used.

If the TCAM is hit, then its 2-bit offset is used to look up values in the [CAM4x128Table Register](#). The UseBucket field of this table specifies whether the ClassID or BucketID field should be used to drive packet distribution. If UseBucket is set to ‘1’, the BucketID field is taken as the destination Bucket ID. If UseBucket is cleared to ‘0’, the ClassID indicates which Class should be processed to determine the Bucket. This information is then delivered to the PDE to deliver the packet message.

It may be advisable to use Class 0 to indicate TCAM-matched packets and Classes 1-3 to indicate the GMAC, so that software can determine easily which packets had a TCAM match.

13.6.1.1 TCAM Set-up Procedure

The TCAM Key fields and the TCAM Mask fields must be preset by software to identify the types of TCAM matches that can be used by the incoming packet. The Key is unknown at TCAM programming time; it is parsed for each incoming packet. The Mask and TCAM values must be preset to either:

- Allow key bits of any value to be considered a match at each bit position, or
- Allow key bits of value ‘0’ to be considered a match at each bit position, or
- Allow key bits of value ‘1’ to be considered a match at each bit position

The choices that must be considered by the programmer in configuring the TCAM key and the Mask fields are:

- Don’t Care what the Key bit is – Configure to yield a local-bit match regardless of the value of the bit in the Key.
- Care what the Key bit is – There are two choices:
 - Configure to yield a local-bit match if the Key value is ‘0’
 - Configure to yield a local-bit match if the Key value is ‘1’

Table 13-11. TCAM Key and Mask Set-up Procedure

Intended Match Result		Required Preset Value	
		Mask bit	TCAM bit
Don’t Care – Any Key bit will match		0	0
Care	Key bits of ‘0’ will match this bit of the TCAM value	1	0
	Key bits of ‘1’ will match this bit of the TCAM value	1	1
Illegal Combination		0	1

When a Key arrives from the Parser, the local-bit match result for each of the 128 bits of the Key will be according to the results shown in [Table 13-12](#).

Table 13-12. Parser Key Bit Match Results

Parser Key bit	If Mask bit is	If TCAM bit is	Local-bit Match Result
0	0	0	Match
1	0	0	Match
0	1	0	Match
1	1	0	No Match
0	1	1	No Match
1	1	1	Match

13.6.2

Secondary Methods

If the UseTCAM bit is not set in [ParserConfig](#), or if it is set but the TCAM is not hit, then the destination Bucket is selected by one of the secondary methods described here.

All secondary methods result in a 7-bit index used to look up an entry in the 128x9 [TranslateTable Register](#). The entries of this table have the same format as the [CAM4x128Table Register](#): the UseBucket field of this table specifies whether the ClassID or BucketID field should be used to drive packet distribution. If UseBucket is set (1), the BucketId field is taken as the destination Bucket ID. If UseBucket is cleared (0), the ClassId indicates which Class should be processed to determine the Bucket. This information is then delivered to the PDE to deliver the packet message.

13.6.2.1

CRC7 Hash

If UseHash is set and either UseTCAM is not set or else UseTCAM is set but the TCAM misses, then the following method is used to generate a 7-bit index into the 128-entry [TranslateTable Register](#).

The 128-bit Parser key goes through a CRC7 Hash algorithm with a programmable 7-bit polynomial specified in the CRCHashPoly field of the [ParserConfig](#). The resulting 7-bit value is used to index into the 128-entry [TranslateTable Register](#).

This HASH algorithm uses the standard Ethernet polynomial to compute the CRC7 result, which is placed in the Prepad field

13.6.2.2

Identification by Combined L3/L4 Protocols

If UseProto is set and either UseTCAM is not set or else UseTCAM is set but the TCAM misses, then the following method is used to generate a 7-bit index into the 128-entry [TranslateTable Register](#).

The L3_CAM and L4_CAM indexes of the Parser can be used to provide sixteen L3 Protocol Types and eight L4 Protocol Types, providing 128 combinations. This is used to index into the 128-entry [TranslateTable Register](#).

The four L3_CAM bits form the MSB of the index and the three L4_CAM bits form the lsb.

13.6.2.3

Port-Based Classification

If UsePort is set and either UseTCAM is not set or else UseTCAM is set but the TCAM misses, then the following method is used to generate a 7-bit index into the 128-entry [TranslateTable Register](#).

For GMACs, the port number in the range of 0 to 3 is used to index the 128-entry [TranslateTable Register](#). Only indices 0-3 are used in this mode.

13.6.2.4 Most-significant 7 Bits of the 128-bit Key Field from the Packet (Default)

If none of UseHash, UseProto, or UsePort is set and if UseTCAM is not set or the TCAM misses, then the following method is used to generate a 7-bit index into the 128-entry [TranslateTable Register](#).

The Most-significant 7 bits of the Parser key is used to index into the 128-entry [TranslateTable Register](#). This can be useful for in-band communication of packet-handling instructions or other proprietary functions.

13.6.2.5 Selector Index Modifier Details

The 7 bit index output from the MUX used by the Secondary Methods is sent through a Selector Index Modifier as shown in [Figure 13-14](#).

The “Port”, “L3_CAM_ADDR”, “L4_CAM_ADDR” and “unclassified” information of the incoming packet is matched against an 8-entry CAM called the HASH7_BASE_MASK CAM. “L3_CAM_ADDR” is the index that matched in the L3_CAM, and similarly “L4_CAM_ADDR” is the index that matched in the L4_CAM.

“Unclassified” bit is defined based on the unclassified mask in the ParserConfig register. For deciding whether classification is successful, 10 different conditions are checked.

1. L3 CAM HIT (See [Figure 13-10](#))
2. L4 CAM HIT (See [Figure 13-11](#))
3. ExtraHeader Extract Valid (E field in the Rx Packet Format in [Figure 13-9](#))
4. L3CTable Extract 0 Valid (L3-1 field in the Rx Packet Format in [Figure 13-9](#))
5. L3CTable Extract 1 Valid (L3-2 field in the Rx Packet Format in [Figure 13-9](#))
6. L3CTable Extract 2 Valid (L3-3 field in the Rx Packet Format in [Figure 13-9](#))
7. L4CTable Extract 0 Valid (L4-1 field in the Rx Packet Format in [Figure 13-9](#))
8. L4CTable Extract 1 Valid (L4-2 field in the Rx Packet Format in [Figure 13-9](#))
9. L2 Header Done (Complete L2 Header such as ethernet or PPP was received. This will be based on the L2Proto field in the L2Type register)
10. L3 Header Done (Based on L3CTable Extract 0, 1 and 2 valid and also successful extraction of L4 protocol from the L3 header)

The “unclassified mask” field in the ParserConfig register defines which of these 10 conditions to use when defining a packet to be “unclassified”.

This same definition is used for the “unclassified” bit in the Receive Descriptor (See [Section 13.3.6.1 Rx Packet Descriptor Format](#)).

The 8-entry HASH7_BASE_MASK CAM indexes into an 8-deep table that gives out the HASH7_BASE and HASH7_MASK. The final index used to address the 128-deep Translate Table is = (SecondaryMUXout & HASH_MASK) | HASH_BASE.

Examples of using this HASH7_BASE_MASK CAM:

- 1) Partitioned Translate Table based on Port

Assume 4 ports (i.e. 4 GMACs). We want to use CRC7 based hash for distribution, however we also want packets from each port to be mapped to a different section of the 128-deep Translate Table.

Set UseHASH bit in ParserConfig to one.

Set up HASH7_BASE_MASK_CAM_KEYS in the following way:

```
uncla_MASK[0]= 0; L4_P_MASK[0]= 0; L3_P_MASK[0]= 0; PortMask [0]=
1; Port [0]= 0; // other fields are "don't care" because the masks
are zero
```

```
uncla_MASK[1]= 0; L4_P_MASK[1]= 0; L3_P_MASK[1]= 0; PortMask [1]=
1; Port [1]= 1; // other fields are "don't care" because the masks
are zero
```

```
uncla_MASK[2]= 0; L4_P_MASK[2]= 0; L3_P_MASK[2]= 0; PortMask [2]=
1; Port [2]= 2; // other fields are "don't care" because the masks
are zero
```

```
uncla_MASK[3]= 0; L4_P_MASK[3]= 0; L3_P_MASK[3]= 0; PortMask [3]=
1; Port [3]= 3; // other fields are "don't care" because the masks
are zero
```

Since one of the first 4 entries will always be hit, there is no need to program 4 to 7 entries.

Set up HASH7_BASE_MASK in the following way:

```
HASH_MASK[0]= 0x1F; HASH_BASE[0]= 0x00;
HASH_MASK[1]= 0x1F; HASH_BASE[1]= 0x20;
HASH_MASK[2]= 0x1F; HASH_BASE[2]= 0x40;
HASH_MASK[3]= 0x1F; HASH_BASE[3]= 0x60;
```

This will have the effect of partitioning the translate table equally among all the 4 ports.

2) Directing all Unclassified traffic to Translate Table Entry 127

Set up “Unclassified mask” appropriately in ParserConfig, based on application. This will define the exact meaning of “unclassified”.

Set up HASH7_BASE_MASK_CAM_KEYS in the following way:

```
uncla_MASK[0]= 1; L4_P_MASK[0]= 0; L3_P_MASK[0]= 0; PortMask [0]=
0; unclassified [0]= 1; // other fields are "don't care" because
the masks are zero
```

Set up HASH7_BASE_MASK in the following way:

```
HASH_MASK[0]= 0x00; HASH_BASE[0]= 0x7F;
```

This will cause all unclassified packets to translate table entry 127.

13.7 Packet Distribution Engine

The Packet Director provides a 9-bit value that specifies to the Packet Distribution Engine (PDE) where to deliver the packet message. The nine bits include:

- 2 bits of packet classification (always used)
- 6 bits that indicate a Bucket ID (only used if UseBucket is set)
- 1 UseBucket bit
- 1 I/O Bucket – Network *redirect* transmit bucket enable bit

The UseBucket bit specifies whether the PDE should use the Class distribution method (UseBucket==0), or should send the packet message only to the specific Bucket ID (UseBucket not equal to 0). See the [CAM4x128Table Register](#).

If UseBucket==0, the PDE will perform Class-based packet distribution, which provides for round-robin scheduling of packets among Threads. The ClassID field specifies which of the four Classes should determine the Packet Descriptor's destination. Each Class contains a 64-bit PDE_MASK indicating the set of CPU buckets that are allowed to receive these messages. The eligible set of buckets is a bitwise AND of PDE_MASK and the set of buckets that have enough credits. The next Bucket is selected using a two-level hierarchical round-robin algorithm.

If UseBucket==1, the message will attempt to go only to the specified Bucket. The ClassID field is still used to allocate the Packet Descriptor to one of the four Class FIFOs. The PDE will only send the Packet Descriptor to the specified Bucket (PDE_MASK is ignored).

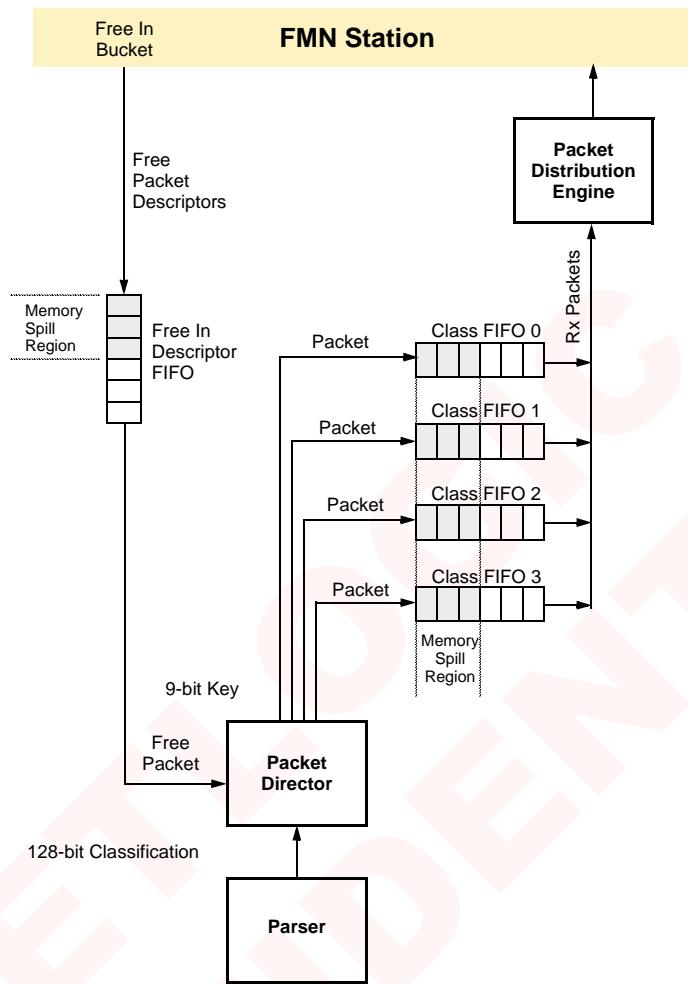
The classification vector can also be pre-pended to the Packet in memory (this pre-pend process is called "Prepad").

The Packet Distribution Engine (PDE), shown in [Figure 13-16](#), supports four distribution Classes. It performs a load-balanced distribution of messages based on the results of the Packet Director.

The Bucket chosen is determined in two stages:

1. PDE_CLASS[0-3] is a set of four 64-bit mask registers, one for each of the four Classes. (see [Section 13.11.1, "PDE_CLASS0"](#).) These registers indicate the set of Bucket IDs that are members of each Class. If UseBucket==0, the PDE goes through the bit mask using round-robin scheduling to determine a potential recipient Bucket.
2. The status of the Credit Counter for the potential recipient Bucket is checked. If the Bucket does not have sufficient credit, the PDE will continue searching through the bit mask until a Bucket in the Class with sufficient Credits is identified.

The FMN message will be directed to this Bucket. In this way, load balancing is achieved.

Figure 13-16. Packet Distribution Engine

13.7.1

Bypass Packets Capability

Bypass traffic is sent from ingress networking blocks to egress networking blocks without going through CPU processing.

Users can program network controllers to bypass packets using two mechanisms:

- Port-based redirect
- Classification-based redirect

13.7.1.1

Port-Based Redirect

Using port based redirection, packet redirection can be controlled on a per-port granularity. Each of the four GMAC ports can be re-directed to the corresponding egress port on the same GMAC controller or an egress port of a different GMAC controller. (For the optional XAUI ports the redirection is between ports XAUI_0 and XAUI_1). For this purpose, A 16-bit redirection control register, [GMAC PortBasedRedirect](#) on page 473, is added to the GMAC controller. Each GMAC port uses four bits from the register to determine the final destination of incoming packets.

Using this register, the traffic can be delivered to a CPU core or it can be delivered directly to one of the four GMAC egress controllers. The port number for bypass traffic cannot be changed.

Since packets that are being re-directed still go through the classifier, a global register is provided to select one of the four classes for the port-based redirected traffic.

13.7.1.2 Translate-Table Redirect

The BUCKET_ID field in the Translate Table and the 4-entry CAM are made 1-bit wider, so that a 7-bit destination bucket number can be programmed. Using buckets 64-127, the translate table can redirect traffic, based on classification results, either to a CPU core or to egress message buckets of any networking interface.

GMAC<->GMAC traffic can be redirected using Translate-Table redirection. This method also enables some percentage of traffic to a CPU core to be statistically redirected, and the remaining traffic sent to bypass ports, based on classification hash results.

13.7.1.3 FreeBackID Register

For all bypass traffic, the packet descriptor format must be modified to contain the FreeBack Bucket ID of the packet buffer.

For all bypassed traffic, the Status field of the [Rx Packet Descriptor Format\[62:56\]](#) register is masked out and replaced with the FreeBack Bucket ID (FBID_BYPASS) from the [BypassClassFreeBackID Register\[6:0\]](#). In addition to this, the port number contained in the lower four bits of each descriptor is masked out.

13.7.1.4 Limitations

1. If a particular egress port is being used to transmit bypassed traffic and the CPU core also needs to use this egress port to transmit packets, messages longer than 4 entries cannot be used for traffic on this port. Using packets > 4 entry messages cause interleaving between CPU sent traffic and network controller bypass traffic.
2. If a particular port on the egress GMAC is being back pressured from the remote GMAC device, the class spill area's on the incoming GMAC interface will build up. After the class spill area is full, all ingress ports writing packets to this class are back pressured.
3. Bypass traffic consumes DRAM bandwidth because bypass packets are still written to and read from the memory subsystem.
4. The XLS does not remove CRC for any incoming packet. Therefore, when using bypass mode (bypass capabilities), the programmer must configure the egress port to not compute and append CRC on outgoing packets. If a CPU needs to transmit a packet on the same egress port, it must compute and attach CRC to the outgoing packet.

13.7.2 Packet Director and Packet Distribution Engine Registers

13.7.2.1 CAM4x128Table Register

The TCAM receives a 128-bit key extracted by the Parser and matches it against each of its 4 entries in the CAM4x128Key.CAMKey field. Each entry contains a 128-bit mask (CAM4x128Key.CAMMask) which specifies which bits to compare when set to '1', and which bits are "don't care" when cleared to '0'. The result is an offset used to look up values in this register (CAM4x128Table). Using the UseBucket field, this table specifies whether the Class_ID or Bucket_ID field should be used to drive packet distribution. If UseBucket is set to '1', the Bucket_ID field is taken as the forced destination Bucket ID. If UseBucket is cleared to '0', the Class_ID indicates which Class should be used.

When the I/O Bucket bit is set to '1', the packets are redirected to the GMAC egress ports as shown in [Section 13.7.2.4, "GMAC PortBasedRedirect"](#).

Register ID: 0x172 – 0x173 (See [Table 13-13](#) for detailed bit mapping)

Address Offset: 5C8 – 5CC

Bits	Name	Description	R/W	Reset
31:26	Reserved	Reserved	RO	0
25	IO_BKT_1	I/O Bucket 1 – Network redirect transmit bucket enable bit	R/W	0
24:23	CID_1 ^a	Class ID 1	R/W	0
22:17	BID_1 ^a	Lower 6 bits of Bucket_ID_1	R/W	0
16	UB_1 ^a	Use Bucket_ID_1	R/W	0
15:10	Reserved	Reserved	RO	0
9	IO_BKT_0	I/O Bucket 0 – Network redirect transmit bucket enable bit	R/W	0
8:7	CID_0	Class ID 0	R/W	0
6:1	BID_0	Lower 6 bits of Bucket_ID_0	R/W	0
0	UB_0	Use Bucket_ID_0	R/W	0

a. See [Packet Distribution Engine](#) on page 429 for details.

Note: Register addresses for the CAM4x128 table are given in [Table 13-13](#).

Table 13-13. CAM4x128Table Register Addresses

Addr	[31:26]	[25]	[24:23]	[22:17]	[16]	[15:10]	[9]	[8:7]	[6:1]	[0]
0x172	Reserved	IO_BKT_1	CID_1	BID_1	UB_1	Reserved	IO_BKT_0	CID_0	BID_0	UB_0
0x173		IO_BKT_3	CID_3	BID_3	UB_3		IO_BKT_2	CID_2	BID_2	UB_2

13.7.2.2 CAM4x128Key Register

Register ID: 0x180 – 0x19F

Address Offset: 0x600 – 0x67C

All four GMACs in a quad share this register in common.

(See [Section 13-3, “Network Accelerator Register Layout”](#).)

This shows the organization of the first of four 128-bit fields of the TCAM key. The other three fields are identical in layout, with addresses as shown in [Table 13-14](#). If enabled, the 128-bit key from the Parser is matched against this key with TCAM masking by the CAMMask field. A bit set in the CAMMask forces a comparison; a bit cleared in the CAMMask indicates a “don’t care” condition. (Thus a mask of all zeros will cause a match with any value in the TCAM.)

Address#	Bit Name	Description	R/W	RESET
0x180	CAMKey0[31:0]	31:0 of 128-bit CAM entry	R/W	0
0x181	CAMKey0[63:32]	63:32 of 128-bit CAM entry	R/W	0
0x182	CAMKey0[95:64]	95:65 of 128-bit CAM entry	R/W	0
0x183	CAMKey0[127:96]	127:96 of 128-bit CAM entry	R/W	0
0x184	CAMMask[31:0]	31:0 of 128-bit CAM mask	R/W	0
0x185	CAMMask[63:32]	63:32 of 128-bit CAM mask	R/W	0
0x186	CAMMask[95:64]	95:65 of 128-bit CAM mask	R/W	0
0x187	CAMMask[127:96]	127:96 of 128-bit CAM mask	R/W	0

Note: The register addresses for the four CAM keys and the four CAM masks are given in [Table 13-14](#).

Table 13-14. CAM4x128Key Register Addresses

CAM Entry	CAM Key Address				CAM Mask Address			
0:	0x183	0x182	0x181	0x180	0x187	0x186	0x185	0x184
1:	0x18B	0x18A	0x189	0x188	0x18F	0x18E	0x18D	0x18C
2:	0x193	0x192	0x194	0x190	0x197	0x196	0x195	0x194
3:	0x19B	0x19A	0x199	0x198	0x19F	0x19E	0x19D	0x19C

Note: CAM Mask should be set to “1” to validate the corresponding bit position.

13.7.2.3 TranslateTable Register

Register ID: 0x1A0 – 0x1DF

Address Offset: 0x680 – 0x77C

All four GMACs in a quad share this register in common.

To avoid possible GMAC port hang these registers should be set up before the incoming traffic is enabled. Also, these register should not be read during traffic flow. If read access of these registers is required, software can maintain a shadow copy of the contents for read purposes.

(See [Section 13-3, “Network Accelerator Register Layout”](#).)

Bits	Name	Description	R/W	Reset
Entry 0 – 0x1A0				
15:10	<i>Reserved</i>	<i>Reserved</i>	RO	0
9	IO_BKT_0	I/O Bucket 0 – Network redirect transmit bucket enable bit	R/W	0
8:7	CID_0	Class ID 0	R/W	0
6:1	BID_0	Lower 6 bits of Bucket_ID_0	R/W	0
0	UB_0	Use Bucket ID 0	R/W	0
Entry 1 – 0x1A0				
31:26	<i>Reserved</i>	<i>Reserved</i>	RO	0
25	IO_BKT_1	I/O Bucket 1 – Network redirect transmit bucket enable bit	R/W	0
24:23	CID_1	Class ID 1	R/W	0
22:17	BID_1	Lower 6 bits of Bucket_ID_1	R/W	0
16	UB_1	Use Bucket ID 1	R/W	0
• • •				
Entry 126 – 0x1DF				
15:10	<i>Reserved</i>	<i>Reserved</i>	RO	0
9	IO_BKT_126	I/O Bucket 126 – Network redirect transmit bucket enable bit	R/W	0
8:7	CID_126	Class ID 126	R/W	0
6:1	BID_126	Lower 6 bits of Bucket_ID_126	R/W	0
0	UB_126	Use Bucket ID 126	R/W	0
Entry 127 – 0x1DF				
31:26	<i>Reserved</i>	<i>Reserved</i>	RO	0
25	IO_BKT_127	I/O Bucket 127 – Network redirect transmit bucket enable bit	R/W	0
24:23	CID_127	Class ID 127	R/W	0
22:17	BID_127	Lower 6 bits of Bucket_ID_127	R/W	0
16	UB_127	Use Bucket ID 127	R/W	0

Note: Register addresses for the Translate Table are given in [Table 13-15](#).

Table 13-15. CAM4x128Table Register Addresses

Addr	[31:26]	[25]	[24:23]	[22:17]	[16]	[15:10]	[9]	[8:7]	[6:1]	[0]
0x1A0	RSVD	IO_BKT_1	CID_1	BID_1	UB_1	RSVD	IO_BKT_0	CID_0	BID_0	UB_0
0x1A1		IO_BKT_3	CID_3	BID_3	UB_3		IO_BKT_2	CID_2	BID_2	UB_2
•••		•••	•••	•••	•••		•••	•••	•••	•••
0x1DF		IO_BKT_127	CID_127	BID_127	UB_127		IO_BKT_126	CID_126	BID_126	UB_126

13.7.2.4 GMAC PortBasedRedirect

This register is mapped into the GMAC space.

Register ID: 0x107

Address Offset: 0x41C

All four GMACs in a quad (and the optional XAUI port) share this register in common.

Bits	Name	Description	R/W	Reset
31:9	Reserved	Reserved	RO	0
15:12	RE_DIRECT_3	Re-direct control code for GMAC3/7 b0000; DEFAULT – Packets Redirected through Classification or go to CPU buckets	WO	0
11:8	RE_DIRECT_2	Re-direct control code for GMAC2/6 Packets Redirected to: ^a	WO	0
7:4	RE_DIRECT_1	Re-direct control code for GMAC1/5 0001; Egress port of GMAC0/4 or Optional XAUI_0/1	WO	0
3:0	RE_DIRECT_0	Re-direct control code for GMAC0/4 or XAUI_0/1 0010; Egress port of GMAC1/5 0100; Egress port of GMAC2/6 1000; Egress port of GMAC3/7	WO	0

a. These bits are write only. When read back they return all zeros.

13.7.2.5 BypassClassFreeBackID Register

Register ID: 0x108

Address Offset: 0x420

All four GMACs in a quad share this register.

Bits	Name	Description	R/W	Reset
31:9	Reserved	Reserved	RO	0
8:7	CLASS_BYPASS	Class ID for all port-based packets 00: Class FIFO 0 01: Class FIFO 1 10: Class FIFO 2 11: Class FIFO 3	R/W	0
6:0	FBID_BYPASS	Free-back ID to be used for all bypass traffic. 0 - 63: points to CPU buckets 64 - 127: See "Automatic Buffer Management" on page 431.	R/W	0

13.8 Prepad

Network Accelerators can be configured to prepad received packets with one cache line (256 bits) of Packet Director output for that packet. [Table 13-17](#) shows the format of the prepad.

Table 13-16. Prepad Format ‘00’

Bits ^a (DESCENDING)	Name	Description	Prepad Byte ^b (ASCENDING)
255:233	Reserved	Reserved	Prepad Byte 0 thru Byte 2 (bit 255 leads Byte 0)
232	UseBucket	Use Bucket (from PDE)	
231:226	Reserved	Reserved	Prepad Byte 3 (bit 231 leads Byte 3)
225:224	UseClass	Use Class (from PDE)	
223:222	Reserved	Reserved	Prepad Byte 4 (bit 223 leads Byte 4)
221:216	Bucket	The Bucket ID (from PDE)	
215	Reserved	Reserved	Prepad Byte 5 (bit 215 leads Byte 5)
214:208	Hash	Hash Index for UseHash mode.	
207:204	Reserved	Reserved	Prepad Byte 6 thru Byte 8 (bit 207 leads Byte 6)
203:184	SequenceNumber	Sequence Number: Each GMAC or XAUI-port generates a unique sequential “sequence number.”	
183:180	Reserved	Reserved	
179	L3_CAM_Hit	L3 CAM Hit 0: L3 CAM not hit 1: L3 CAM hit	
178	L4_CAM_Hit	L4 CAM Hit 0: L4 CAM not hit 1: L4 CAM hit	Prepad Byte 9 (bit 183 leads Byte 9)
177	Extra_Header_Extraction_Valid	Extra Header Extraction Valid 0: Extra header extraction is not valid 1: Extra header extraction is valid	
176	L3CTable_Extract_0_Valid	L3CTable Extract 0 Valid 0: L3CTable extract 0 is not valid 1: L3CTable extract 0 is valid	

Table 13-16. Prepad Format '00' (continued)

Bits ^a (DESCENDING)	Name	Description	Prepad Byte ^b (ASCENDING)
175	L3CTable_Extract_1_Valid	L3CTable Extract 1 Valid 0: L3CTable extract 1 is not valid 1: L3CTable extract 1 is valid	Prepad Byte 10 (bit 175 leads Byte 10)
174	L3CTable_Extract_2_Valid	L3CTable Extract 2 Valid 0: L3CTable extract 2 is not valid 1: L3CTable extract 2 is valid	
173	L4CTable_Extract_0_Valid	L4CTable Extract 0 Valid 0: L4CTable extract 0 is not valid 1: L4CTable extract 0 is valid	
172	L4CTable_Extract_1_Valid	L4CTable Extract 1 Valid 0: L4CTable extract 1 is not valid 1: L4CTable extract 1 is valid	
171	L2_Header_Done	L2 Header Done 0: L2 Header is not done 1: L2 Header is done	
170	L3_Header_Done	L3 Header Done 0: L3 Header is not done 1: L3 Header is done	
169	IP_CSUM_Valid	IP CSUM Valid 0: IP CSUM is not valid 1: IP CSUM is valid	
168	TCP/UDP_CSUM_Valid	TCP/UDP CSUM Valid 0: TCP/UDP is not valid 1: TCP/UDP is valid	
167-136	CRC32	This is a full CRC32 Hash of the extracted key. This is a CRC32 over the 128-bit extracted fields of the pre-pad.	Prepad Byte 11 thru Byte 14 (bit 167 leads Byte 11)
135	Reserved	Reserved	Prepad Byte 15 (bit 135 leads Byte 15)
134-131	L3_CAM_Addr	L3 CAM Addr	
130-128	L4_CAM_Addr	L4 CAM Addr	
127:0	128_Bit_Parser_Extract_Vector ^c	128 Bit Parser Extract Vector	Prepad Byte 16 thru Byte 31 (bit 127 leads Byte 16)

- a. Note that the “Bits” column numbers bits in DESCENDING order, such that Bit 255 is the first bit of the prepad transmitted; bit 0 is the last bit transmitted.
- b. Note that the “Prepad Bytes” column numbers bytes in ASCENDING order of prepad bytes. Thus, Prepad Byte 0 consists of bits 255, 254, 253, ... 248. Prepad Byte 0 precedes Prepad Byte 1 when transmitting at the front of a packet.
- c. This is raw Parser extracted key, it is not affected by the mask used to pass it to the Director.

Table 13-17. Prepad Format '01'

Bits ^a (DESCENDING)	Name	Description	Prepad Byte ^b (ASCENDING)
255:254	Class	Class	
253:248	Bucket	Bucket	Prepad Byte 0
247:192	TimeStamp	TimeStamp[55:0]	Prepad Byte 1 to Byte 7 Inclusive
191	UseBucket	Use Bucket	
190:184	Hash7	Hash 7	
183:180	Reserved	Reserved	
179	L3_CAM_Hit	L3 CAM Hit 0: L3 CAM not hit 1: L3 CAM hit	
178	L4_CAM_Hit	L4 CAM Hit 0: L4 CAM not hit 1: L4 CAM hit	
177	Extra_Header_Extraction_Valid	Extra Header Extraction Valid 0: Extra header extraction is not valid 1: Extra header extraction is valid	
176	L3CTable_Extract_0_Valid	L3CTable Extract 0 Valid 0: L3CTable extract 0 is not valid 1: L3CTable extract 0 is valid	
175	L3CTable_Extract_1_Valid	L3CTable Extract 1 Valid 0: L3CTable extract 1 is not valid 1: L3CTable extract 1 is valid	
174	L3CTable_Extract_2_Valid	L3CTable Extract 2 Valid 0: L3CTable extract 2 is not valid 1: L3CTable extract 2 is valid	
173	L4CTable_Extract_0_Valid	L4CTable Extract 0 Valid 0: L4CTable extract 0 is not valid 1: L4CTable extract 0 is valid	
172	L4CTable_Extract_1_Valid	L4CTable Extract 1 Valid 0: L4CTable extract 1 is not valid 1: L4CTable extract 1 is valid	
171	L2_Header_Done	L2 Header Done 0: L2 Header is not done 1: L2 Header is done	
170	L3_Header_Done	L3 Header Done 0: L3 Header is not done 1: L3 Header is done	
169	IP_CSUM_Valid	IP CSUM Valid 0: IP CSUM is not valid 1: IP CSUM is valid	
168	TCP/UDP_CSUM_Valid	TCP/UDP CSUM Valid 0: TCP/UDP is not valid 1: TCP/UDP is valid	

Table 13-17. Prepad Format '01' (continued)

Bits ^a (DESCENDING)	Name	Description	Prepad Byte ^b (ASCENDING)
167-136	CRC32	This is a full CRC32 Hash of the extracted key. This is a CRC32 over the 128-bit extracted fields of the pre-pad.	Prepad Byte 11 thru Byte 14 (bit 167 leads Byte 11)
135	<i>Reserved</i>	<i>Reserved</i>	Prepad Byte 15 (bit 135 leads Byte 15)
134-131	L3_CAM_Addr	L3 CAM Addr	
130-128	L4_CAM_Addr	L4 CAM Addr	
127:0	128_Bit_Parser_Extract_Vector	128 Bit Parser Extract Vector	Prepad Byte 16 thru Byte 31 (bit 127 leads Byte 16)

- a. Note that the "Bits" column numbers bits in DESCENDING order, such that Bit 255 is the first bit of the prepad transmitted; bit 0 is the last bit transmitted.
- b. Note that the "Prepad Bytes" column numbers bytes in ASCENDING order of prepad bytes. Thus, Prepad Byte 0 consists of bits 255, 254, 253, ... 248. Prepad Byte 0 precedes Prepad Byte 1 when transmitting at the front of a packet.

Table 13-18. Prepad Format '10'

Bits ^a (DESCENDING)	Name	Description	Prepad Byte ^b (ASCENDING)
255:192	TimeStamp	TimeStamp[63:0]	Prepad Byte 0 to Byte 7 Inclusive
191:190	Class	Class	Prepad Byte 8
189:184	Bucket	Bucket	
183:180	<i>Reserved</i>	<i>Reserved</i>	
179	L3_CAM_Hit	L3 CAM Hit 0: L3 CAM not hit 1: L3 CAM hit	Prepad Byte 9 (bit 183 leads Byte 9)
178	L4_CAM_Hit	L4 CAM Hit 0: L4 CAM not hit 1: L4 CAM hit	
177	Extra_Header_Extraction_Valid	Extra Header Extraction Valid 0: Extra header extraction is not valid 1: Extra header extraction is valid	
176	L3CTable_Extract_0_Valid	L3CTable Extract 0 Valid 0: L3CTable extract 0 is not valid 1: L3CTable extract 0 is valid	

Table 13-18. Prepad Format '10' (continued)

Bits ^a (DESCENDING)	Name	Description	Prepad Byte ^b (ASCENDING)
175	L3CTable_Extract_1_Valid	L3CTable Extract 1 Valid 0: L3CTable extract 1 is not valid 1: L3CTable extract 1 is valid	Prepad Byte 10 (bit 175 leads Byte 10)
174	L3CTable_Extract_2_Valid	L3CTable Extract 2 Valid 0: L3CTable extract 2 is not valid 1: L3CTable extract 2 is valid	
173	L4CTable_Extract_0_Valid	L4CTable Extract 0 Valid 0: L4CTable extract 0 is not valid 1: L4CTable extract 0 is valid	
172	L4CTable_Extract_1_Valid	L4CTable Extract 1 Valid 0: L4CTable extract 1 is not valid 1: L4CTable extract 1 is valid	
171	L2_Header_Done	L2 Header Done 0: L2 Header is not done 1: L2 Header is done	
170	L3_Header_Done	L3 Header Done 0: L3 Header is not done 1: L3 Header is done	
169	IP_CSUM_Valid	IP CSUM Valid 0: IP CSUM is not valid 1: IP CSUM is valid	
168	TCP/UDP_CSUM_Valid	TCP/UDP CSUM Valid 0: TCP/UDP is not valid 1: TCP/UDP is valid	
167-136	CRC32	This is a full CRC32 Hash of the extracted key. This is a CRC32 over the 128-bit extracted fields of the pre-pad.	Prepad Byte 11 thru Byte 14 (bit 167 leads Byte 11)
135	<i>Reserved</i>	<i>Reserved</i>	Prepad Byte 15 (bit 135 leads Byte 15)
134-131	L3_CAM_Addr	L3 CAM Addr	
130-128	L4_CAM_Addr	L4 CAM Addr	
127:0	128_Bit_Parser_Extract_Vector	128 Bit Parser Extract Vector	Prepad Byte 16 thru Byte 31 (bit 127 leads Byte 16)

- a. Note that the “Bits” column numbers bits in DESCENDING order, such that Bit 255 is the first bit of the prepad transmitted; bit 0 is the last bit transmitted.
- b. Note that the “Prepad Bytes” column numbers bytes in ASCENDING order of prepad bytes. Thus, Prepad Byte 0 consists of bits 255, 254, 253, ... 248. Prepad Byte 0 precedes Prepad Byte 1 when transmitting at the front of a packet.

Table 13-19. Prepad Format '11'

Bits^a (DESCENDING)	Name	Description	Prepad Byte^b (ASCENDING)
255:192	TimeStamp	TimeStamp[63:0]	Prepad Byte 0 to Byte 7 Inclusive
191	UseBucket	Use Bucket	Prepad Byte 8
190:184	Hash7	Hash7	
183:180	Reserved	Reserved	
179	L3_CAM_Hit	L3 CAM Hit 0: L3 CAM not hit 1: L3 CAM hit	
178	L4_CAM_Hit	L4 CAM Hit 0: L4 CAM not hit 1: L4 CAM hit	Prepad Byte 9 (bit 183 leads Byte 9)
177	Extra_Header_Extraction_Valid	Extra Header Extraction Valid 0: Extra header extraction is not valid 1: Extra header extraction is valid	
176	L3CTable_Extract_0_Valid	L3CTable Extract 0 Valid 0: L3CTable extract 0 is not valid 1: L3CTable extract 0 is valid	
175	L3CTable_Extract_1_Valid	L3CTable Extract 1 Valid 0: L3CTable extract 1 is not valid 1: L3CTable extract 1 is valid	
174	L3CTable_Extract_2_Valid	L3CTable Extract 2 Valid 0: L3CTable extract 2 is not valid 1: L3CTable extract 2 is valid	
173	L4CTable_Extract_0_Valid	L4CTable Extract 0 Valid 0: L4CTable extract 0 is not valid 1: L4CTable extract 0 is valid	
172	L4CTable_Extract_1_Valid	L4CTable Extract 1 Valid 0: L4CTable extract 1 is not valid 1: L4CTable extract 1 is valid	Prepad Byte 10 (bit 175 leads Byte 10)
171	L2_Header_Done	L2 Header Done 0: L2 Header is not done 1: L2 Header is done	
170	L3_Header_Done	L3 Header Done 0: L3 Header is not done 1: L3 Header is done	
169	IP_CSUM_Valid	IP CSUM Valid 0: IP CSUM is not valid 1: IP CSUM is valid	
168	TCP/UDP_CSUM_Valid	TCP/UDP CSUM Valid 0: TCP/UDP is not valid 1: TCP/UDP is valid	

Table 13-19. Prepad Format '11' (continued)

Bits ^a (DESCENDING)	Name	Description	Prepad Byte ^b (ASCENDING)
167-136	CRC32	This is a full CRC32 Hash of the extracted key. This is a CRC32 over the 128-bit extracted fields of the pre-pad.	Prepad Byte 11 thru Byte 14 (bit 167 leads Byte 11)
135	<i>Reserved</i>	<i>Reserved</i>	Prepad Byte 15
134-131	L3_CAM_Addr	L3 CAM Addr	(bit 135 leads Byte 15)
130-128	L4_CAM_Addr	L4 CAM Addr	
127:0	128_Bit_Parser_Extract_Vector	128 Bit Parser Extract Vector	Prepad Byte 16 thru Byte 31 (bit 127 leads Byte 16)

a. Note that the "Bits" column numbers bits in DESCENDING order, such that Bit 255 is the first bit of the prepad transmitted; bit 0 is the last bit transmitted.

b. Note that the "Prepad Bytes" column numbers bytes in ASCENDING order of prepad bytes. Thus, Prepad Byte 0 consists of bits 255, 254, 253, ... 248. Prepad Byte 0 precedes Prepad Byte 1 when transmitting at the front of a packet.

13.9 Programming Model

13.9.1 System Setup

Parent process:

- Initialize Phy
- Initialize counters
- Setup Classifier (optionally enable prepad)
- Fork Threads to receive packets
- Fork Threads to release packets
- Send free Descriptors to MAC
- Finally, enable ingress

13.9.2 Thread Processing

Thread Initialization

For each Thread process:

1. Have just one Thread in Core initialize FMN for this Core
2. Loop on MSGWAIT, reading and writing packets.

Thread Rx/Tx Loop

Pseudocode example of a wait loop for Rx:

```

while ( 1 ) {
    Issue MSGWAIT instruction with mask of Buckets to wait on
    // Thread goes to sleep,
    // is awakened when a msg. is received
    Get the message
    Extract physical address from Packet Descriptor
    Convert physical address to virtual address
    Process the Rx packet
    if ( Tx packet ready to transmit ) {
        Initialize Packet Descriptor for the packet
        Send the Tx message
    }
}

```

Rx Packet Processing

```

// Extract packet from Descriptor
Get Phy. Addr. from Descriptor
Convert to virtual addr.
Get Length from Descriptor
Get PortID from Descriptor

```

Transmit Packets

```

Get memory for new Packet Descriptor
Init new Packet Descriptor
Fill in fields
Send it

```

13.9.3**Software Reset**

The soft reset sequence is as follows:

1. Do not send any new TxDescriptors or Free In Descriptors
2. Reset RxEnable in the MAC (address 0x00, NOT RxEnable in [RxControl](#))
3. Poll RxHalt until it goes high
4. Set the SoftReset bit in the [RxControl](#)
5. Poll on the SoftResetDone bit the [RxControl](#)
6. Reset the SoftReset bit
7. Set the RxEnable bit

13.9.4**Interrupts**

Each Network Accelerator can generate a variety of interrupts. The interrupts from a Network Accelerator are aggregated to create a single interrupt that goes to the PIC.

In addition, the Network Accelerator itself can generate interrupts on various conditions, such as when its internal FIFOs reach a watermark condition.

Interrupt handling software must distinguish these conditions, as follows.

1. When a Thread receives an interrupt, it must then check the appropriate IntReg register in the appropriate Network Accelerator to determine what interrupt condition actually occurred. (See [Section 13.10.2.7, “IntReg”](#).)
For example, if the PIC receives a hardware interrupt for SGMII port A, Thread software must check the IntReg in the Network Accelerator addressing space for the SGMIIA interface.
2. The Thread must also clear the appropriate interrupt bit in the SGMII port's Network Accelerator.
3. The Thread must clear the interface's interrupt bit in the PIC in the Interrupt Acknowledge Register in the PIC.

13.10 Networking Accelerator Registers

Networking Accelerator registers reside in eight address sub-regions whose sub-region offsets are given in [Table 13-1 on page 425](#). All register offsets shown below apply to the individual Network Accelerator which shares a sub-register space with its interface controller (i.e., a GMAC controller and its Network Accelerator). See [Figure 1-1 on page 39](#) for clarification.

13.10.1 Register Summary

Key to [Table 13-20](#):

Three columns identify the port type. Below the heading, an “A” means all fields of the register in a row apply to the interface type for the column. An “S” means some fields (but not others) of the register also apply to the interface type shown in the columns. (Please see the Notes column and the register description for more details.) An “N” means that register does not apply to that interface type. Register address offsets are four times the Register ID value

Table 13-20. Networking Accelerator Register Summary

Name	Register ID	Address Offset	Description	Notes
Registers that are unique for each GMAC Interface or optional XAUI interface (all other registers are common to, shared by, all GMAC interfaces).^a				
TxControl	0x0A0	0x280	Transmit Control	The TxnHalt field bits[31:16] N/A for GMAC.
RxControl	0x0A1	0x284	Receive Control	SinglePort bit[6] N/A for GMAC.
DescPackCtrl	0x0A2	0x288	Packet Descriptor Control	
StatCtrl	0x0A3	0x28C	Statistics Control	
L2AllocCtrl	0x0A4	0x290	L2 Cache Allocation Register	
IntMask	0x0A5	0x294	Interrupt Mask Register	
IntReg	0x0A6	0x298	Interrupt Register	
Reserved	0x0A7	0x29C	Reserved	
CoreControl	0x0A8	0x2A0	Core Control Register	
Reserved	0x0A9	0x2A4	Reserved	
Reserved	0x0AA	0x2A8	Reserved	
PauseFrame	0x0AB	0x2AC	Pause Frame Register	
Reserved	0x0AC – 0x0AD	0x2B0 – 0x2B4	Reserved	
FIFOModeCtrl1	0x0AE	0x2B8	FIFO Mode Control	Applies only to port with RGMII pins
FIFOModeCtrl2	0x0AF	0x2BC	FIFO Mode Control	Applies only to port with RGMII pins
DMA Credit Registers				
DmaCr0	0x200	0x800	DMA Credit Register	Data4WrMaxCr and Data4RdMaxCr (bits [5:0]) do not apply for GMAC.
Reserved	0x201	0x804	Reserved	
Reserved	0x202	0x808	Reserved	

Table 13-20. Networking Accelerator Register Summary (continued)

Name	Register ID	Address Offset	Description	Notes
DmaCr3	0x203	0x80C	DMA Credit Register	
DMA Spill Control Registers				
RegFrInSpillMemStart0	0x204	0x810	Start address of Regular Free Descriptor In Spill-over	
RegFrInSpillMemStart1	0x205	0x814	Extended start address of Regular Free Descriptor In Spill-over	
RegFrInSpillMemSize	0x206	0x818	Size of Regular Free Descriptor Spill-over Memory	
FrOutSpillMemStart0	0x207	0x81C	Start address of Free Descriptor Out Spill-over	
FrOutSpillMemStart1	0x208	0x820	Extended Start address of Free Descriptor Out Spill-over	
FrOutSpillMemSize	0x209	0x824	Free Out Descriptor Spill-over Memory Size	
Class0SpillMemStart0	0x20A	0x828	Start address of Class0 Descriptor Spill-over	
Class0SpillMemStart1	0x20B	0x82C	Extended start address of Class0 Descriptor Spill-over	
Class0SpillMemSize	0x20C	0x830	Size of Class0 Descriptor Spill-over Memory	
Reserved	0x20D – 0x20F	0x834 – 0x83C	Reserved	
Class1SpillMemStart0	0x210	0x840	Start address of Class1 Descriptor Spill-over	
Class1SpillMemStart1	0x211	0x844	Extended start address of Class1 Descriptor Spill-over	
Class1SpillMemSize	0x212	0x848	Size of Class1 descriptor Spill-over Memory	
Class2SpillMemStart0	0x213	0x84C	Start address of Class2 Descriptor Spill-over	
Class2SpillMemStart1	0x214	0x850	Extended start address of Class2 Descriptor Spill-over	
Class2SpillMemSize	0x215	0x854	Size of Class2 Descriptor Spill-over Memory	
Class3SpillMemStart0	0x216	0x858	Start address of Class3 Descriptor Spill-over	
Class3SpillMemStart1	0x217	0x85C	Extended start address of Class3 Descriptor Spill-over	
Class3SpillMemSize	0x218	0x860	Size of Class3 Descriptor Spill-over Memory	
RegFrIn1SpillMemStart0	0x219	0x864	Start address of Regular Free Descriptor In Spill Over	

Table 13-20. Networking Accelerator Register Summary (continued)

Name	Register ID	Address Offset	Description	Notes
RegFrIn1SpillMemStart1	0x21A	0x868	Extended start address of Regular Free Descriptor In Spill Over.	
RegFrIn1SpillMemSize	0x21B	0x86C	Regular Free Descriptor Spill Over Memory	
TX_DATA_FIFO0 TX_DATA_FIFO1	0x221 to 0x222	0x870 – 0x874	TX Data FIFO Configuration	
Reserved	0x223 to 0x232	0x878 – 0x8C8	Reserved	
Virtual MIPS Mode Support				
FreeQCarve	0x233	0x8CC	Virtual MIPS mode support	
Reserved	0x234 – 0x243	0x8D0 – 0x90C	Reserved	
FIFO Watermark Registers				
ClassWatermarks	0x244	0x910	Watermarks for Classes	
RxWatermarks0	0x245	0x914	Watermarks for Rx data	
Reserved	0x246 to 0x248	0x918 – 0x920	Reserved	
FreeWatermarks	0x249	0x924	Watermark for Free Out Descriptor FIFO	
1588_PTP Source Registers				
1588_PTP_SOURCE_DIV	0x24A	0x928	1588 Source-Divider values	
1588_PTP_SOURCE	0x24B	0x92C	1588 Source Clock	
Reserved	0x24C to 0x2FF	0x930 – 0xBFF	Reserved	
PDE Class Registers				
PDE_CLASS0	0x300 / 0x301	0xC00 / 0xC04	Bit mask for Bucket membership in Class 0	Bits [31:0] are addressed at index 0x300 and bits [63:32] are addressed at index 0x301
PDE_CLASS1	0x302 / 0x303	0xC08 / 0xC0C	Bit mask for Bucket membership in Class 1	Bits [31:0] are addressed at index 0x302 and bits [63:32] are addressed at index 0x303.
PDE_CLASS2	0x304 / 0x305	0xC10 / 0xC14	Bit mask for Bucket membership in Class 2	Bits [31:0] are addressed at index 0x304 and bits [63:32] are addressed at index 0x305.
PDE_CLASS3	0x306 / 0x307	0xC18 / 0xC1C	Bit mask for Bucket membership in Class 3	Bits [31:0] are addressed at index 0x306 and bits [63:32] are addressed at index 0x307.
Debug Registers				
DebugEvent0_A	0x224	0x890	Debug Event Indicator 0A	
DebugEvent0_B	0x224	0x890	Debug Event Indicator 0B	
DebugCounter0_A	0x226	0x898	Debug Event Counter 0A	
DebugCounter0_B	0x227	0x89C	Debug Event Counter 0B	

Table 13-20. Networking Accelerator Register Summary (continued)

Name	Register ID	Address Offset	Description	Notes
DebugEvent1_A	0x22C	0x8B0	Debug Event Indicator 1A	
DebugEvent1_B	0x22D	0x8B4	Debug Event Indicator 1B	
DebugCounter1_A	0x22E	0x8B8	Debug Event Counter 1A	
DebugCounter1_B	0x22F	0x8BC	Debug Event Counter 1B	
DebugDescWordCounts	0x23C	0x8F0	Word count of the FIFO described by DEBUG_DESC_WORD_COUNTS_SEL	
DebugDescWordCountsSel	0x23D	0x8F4	Select Debug Description Word Counts	
DebugRxDescInMAC	0x23E	0x8F8	Word count of total number of Descriptors in the ingress part of the MAC	
DebugTxDescInMAC	0x23F	0x8FC	Word count of total number of Descriptors in the egress part of the MAC	
DebugEvent2_A	0x24C	0x930	Debug Event Indicator 2A	
DebugEvent2_B	0x24D	0x934	Debug Event Indicator 2B	
DebugCounter2_A	0x24E	0x938	Debug Event Counter 2A	
DebugCounter2_B	0x24F	0x93C	Debug Event Counter 2B	

a. Physically there are two separate instances of Network Accelerator, as follows:

- (1) the SGMII network Accelerator shared by all 4 SGMII (GMAC) interfaces SGMII0 / RGMII through SGMII3, and
- (2) the SGMII network Accelerator shared by all 4 SGMII (GMAC) interfaces SGMII4 through SGMII7. For GMACs, all registers from index 0x00 to 0xFF are in the unique space, and all registers from index 0x100 to 0x3FF are in the shared space.

13.10.2 GMAC Registers for SGMII Interfaces in the Network Accelerators

Each of the four SGMII interfaces in a GMAC quad (and the optional XAUI interfaces) has its own set of the following registers. That is, a unique set exists for each interface, for a total of four sets per quad. These sets of registers reside in the Network Accelerator servicing that set of four SGMII interfaces. Each set has its own base address. (See [Section 13-3, “Network Accelerator Register Layout”](#).)

13.10.2.1 TxControl

Register ID: 0x0A0

Address Offset: 0x280

Transmit Control Register

Bits	Name	Description	R/W	Reset
31:16	Reserved	Reserved	R	1
15	TxIdle	Tx is Idle. Set while the Network Interface is not enabled (TxEnable=0) and all FIFOs are empty.	R	1
14	TxEnable	Tx Enable	R/W	0
13:0	TxThreshold	Number of bytes that must be fetched from memory before transmission of packet begins.	R/W	0

13.10.2.2 RxControl

Register ID: 0x0A1

Address Offset: 0x284

Receive Control Register

Bits	Name	Description	R/W	Reset
31:13	Reserved	Reserved	R	0
12:11	RX1588TS	Time Stamp all incoming packets and put the 64-bit value in the Prepad. 00: Old Prepad format (See “Prepad Format ‘00” on page 474.) 01: 56-bit Time Stamp (See “Prepad Format ‘01” on page 476.) 10: 64-bit Time Stamp (See “Prepad Format ‘10” on page 477.) 11: 64-bit Time Stamp (See “Prepad Format ‘11” on page 479.)	R/W	0
10	RGMIIIMode	Selects if interface is in SGMII or RGMII mode – applies to GMAC_0 only. 0: SGMII mode 1: RGMII mode	R/W	0
9	Reserved	Reserved (resets to ‘1’, but can be set to ‘0’)	R/W	1
8:6	ErrStMask	Bit 8: Length Check Error Bit 7: CRC Error Bit 6: Code Error The mask bits must be set if error reporting in the Receive Descriptors is expected.	R/W	‘111’
5	IPReset	Hard-reset the MAC. 0: Not reset 1: Reset	R/W	0
4	DebugInt	Reserved for Factory testing only	R/W	0
3	SoftRstDone	Soft Reset Done. 0: Soft reset is in progress 1: Soft reset done Software polls this bit to determine when soft reset has completed.	R	0

Bits	Name	Description	R/W	Reset
2	SoftReset	Soft Reset. 0: No action 1: Force soft reset Software sets this bit to effect software reset of this Network Interface.	R/W	0
1	RxHalt	Rx is halted, waiting for RxEnable. 0: Not halted 1: Halted, waiting for RxEnable Software can poll this bit to determine when Rx has entered the Halted state.	R	1
0	RxEnable	Rx enable. 0: Disabled 1: Enabled Enable the Rx side of the interface.	R/W	0

See [Section 13.9.3, “Software Reset”](#) for a description of the soft reset sequence.

13.10.2.3 DescPackCtrl

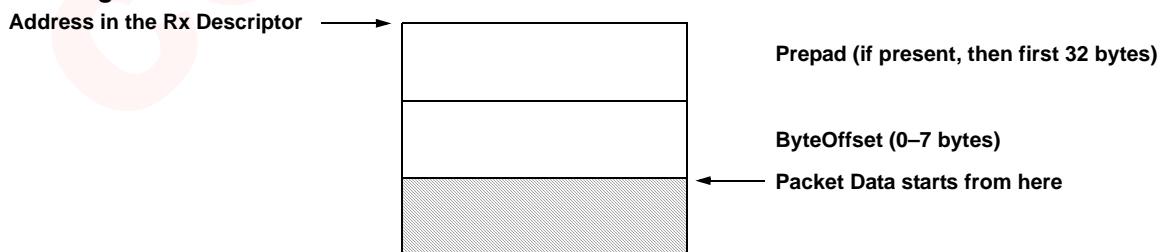
Register ID: 0x0A2

Address Offset: 0x288

Packet Descriptor Control Register

Bits	Name	Description	R/W	Reset
31:26	MaxTxDescP	Defines the number of buffer descriptors per packet that are held by the GMAC before they are freed back. It affects 10 and 100 Mbps Ethernet retry operation only.	R/W	0
25:20	MaxDescHold	Maximum number of Descriptors to hold in TmpFIFO for Tx retry function. After this, the packet is aborted in case of exceptions. Note: 0 is an Illegal Value.	R/W	1
19:17	ByteOffset	Byte offset of GMAC packet data. See Figure 13-17 . See ByteOffsets0 (0xA9) and ByteOffsets1 (0xAA). The offset value may be used to force cache alignment of IP or TCP headers if the Ethernet framing is known.	R/W	0
16	PrePadEnable	Enable prepad to packet. (Flags will be stored with packet.)	R/W	0
15:14	Reserved	Reserved	R/W	0
13:0	RegularSize	Size of a Packet Buffer (in bytes). This field cannot be changed freely at any time, but must be changed just prior to a GMAC reset, to take effect. The value of this field must be \geq 128 bytes, and must be a multiple of 32 bytes; it must be greater than one-quarter of MTU size or Maximum Frame Length of the interface.	R/W	0

Figure 13-17. RxPacket



13.10.2.4 StatCtrl

Register ID: 0x0A3

Address Offset: 0x28C

Statistics Control Register.

Controls statistics monitoring for all interfaces connected to this Network Accelerator.

Bits	Name	Description	R/W	Reset
31:5	Reserved	Reserved	R	0
4	OverFlowEn	Carry overflow interrupt control 0: do not enable 1: enable the statistics counters to generate an interrupt upon overflow, through the Network Accelerator	R/W	0
3	GIG	GMAC configuration, indicated statistics block is being driven by a GMAC core	R/W	0
2	Sten	Statistics operation control 0: do not enable statistics counting 1: Statistics enable	R/W	0
1	ClrCnt	0: (no action) 1: Clear all statistics counters	R/W	0
0	AutoZ	Control zeroing a statistics counter upon read. 0: do not clear a statistics register upon read 1: automatically zero addressed statistics counter upon a read.	R/W	0

13.10.2.5 L2AllocCtrl

Register ID: 0x0A4

Address Offset: 0x290

L2 Cache Allocation Register

Bits	Name	Description	R/W	Reset
31:18	Reserved	Reserved	R	0
17:9	TxL2Allocate	Tx number of cache lines to be marked L2 allocate. While reading data from memory to transmit, the data is stored in the cache. This field reserves the indicated number of cache lines for data to be cached pending transmit. This only should be used for multicast operations where it may be desirable to leave a Tx packet in the cache.	R/W	0
8:0	RxL2Allocate	Rx number of cache lines to be marked L2 allocate. Number of cache lines to use to store received packet data. (Packet data in excess of this will be stored in memory uncached.)	R/W	0

13.10.2.6 IntMask

Register ID: 0x0A5

Address Offset: 0x294

Interrupt Mask Register; setting the masking bit to “1” allows the particular interrupt to be issued.

Bits	Name	Description	R/W	Reset
31	TMR1_1588 ^a	This bit is set every time 1588_PTP Timer 1 expires	R/W	0
30	TMR2_1588 ^a	This bit is set every time 1588_PTP Timer 2 expires	R/W	0
29	TMR3_1588 ^a	This bit is set every time 1588_PTP Timer 3 expires	R/W	0
28	TxTS1588	This bit is set when a 1588 frame is transmitted out. This bit is unique for each interface.	R/W	0
27	Collision_Rx Interrupt	RGMII: this bit specifies that a collision occurred in half-duplex mode.	R/W	0
26	Abort	Packet aborted.	R/W	0
25	Underrun	Underrun occurred while fetching packet from memory.	R/W	0
24	Discard	GMAC discard. Set when the GMAC has been forced to discard packet data. (This is an indication that there are throughput bottlenecks.)	R/W	0
23	AFOverflow	Internal error in MAC (asynchronous FIFO overflow). Should never happen. (Indicates bottlenecks in Rx side. If set, there is the possibility that two packets would be coalesced into one.)	R/W	0
22	TagFull	Internal error in MAC (all 32 tags for Rx packets are being used). Should never happen. (Indicates bottlenecks in Rx side. If set, there is the possibility that packets may be dropped.)	R/W	0
21	Class3Full	Send interrupt when Class 3 FIFO is full (including spill region). Triggered on transition from full–1 to full.	R/W	0
20	C3EarlyFull	Send interrupt when Class 3 FIFO > watermark. Triggered only when threshold is crossed.	R/W	0
19	Class2Full	Send interrupt when Class 2 FIFO is full (including spill region). Triggered on transition from full–1 to full.	R/W	0
18	C2EarlyFull	Send interrupt when Class 2 FIFO > watermark. Triggered only when threshold is crossed.	R/W	0
17	Class1Full	Send interrupt when Class 1 FIFO is full (including spill region). Triggered on transition from full–1 to full.	R/W	0
16	C1EarlyFull	Send interrupt when Class 1 FIFO > watermark. Triggered only when threshold is crossed.	R/W	0
15	Class0Full	Send interrupt when Class 0 FIFO is full (including spill region). Triggered on transition from full–1 to full.	R/W	0
14	C0EarlyFull	Send interrupt when Class 0 FIFO > watermark. Triggered only when threshold is crossed.	R/W	0
13	RxDATAFull	Send interrupt when Rx Data FIFO is full	R/W	0
12	RxEarlyFull	Send interrupt when Rx Data FIFO > watermark. Triggered only when threshold is crossed.	R/W	0
11	Reserved	Reserved	R	0
10	Reserved	Reserved	R	0
9	RFreeEmpty	Send interrupt when Free-In Descriptor FIFO is empty. Triggered on empty+1 to empty.	R/W	0

Bits	Name	Description	R/W	Reset
8	RFEarlyEmpty	Send interrupt when Free-In Descriptor FIFO < watermark. Triggered only when threshold is crossed.	R/W	0
7	SpillOrP2PError	ECC error on Spill channel of P2P channel	R/W	0
6	DebugInt	Debug event interrupt Do not use.	R	0
5	FreeDescFull	Send interrupt when Free-Out Descriptor FIFO is full (including spill region). Triggered on transition from full-1 to full.	R/W	0
4	FreeEarlyFull	Send interrupt when Free Descriptor FIFO > watermark. Triggered only when threshold is crossed.	R/W	0
3	TxFetchError	Send interrupt when Tx data fetches encounter bus error.	R/W	0
2	Carry	Statistics counter overflow	R/W	0
1	MDInt	Management port interrupt. Interrupts when cable is connected/disconnected, speed changes, etc.	R/W	0
0	TxIllegal	Send interrupt when Illegal Tx Packet Descriptor is encountered.	R/W	0

a. Bits[31:29] are present only for GMAC0 base offset, these bits are reserved for other GMAC base offsets.

13.10.2.7 IntReg

Register ID: 0x0A6

Address Offset: 0x298

Interrupt Register; writing to a specific bit with a “1” clears the interrupt.

Bits	Name	Description	R/W	Reset
31	TMR1_1588 ^a	This bit is set every time 1588_PTP Timer 1 expires	R/W	0
30	TMR2_1588 ^a	This bit is set every time 1588_PTP Timer 2 expires	R/W	0
29	TMR3_1588 ^a	This bit is set every time 1588_PTP Timer 3 expires	R/W	0
28	TxTS1588	This bit is set when a 1588 frame is transmitted out. This bit is unique for each interface.	R/W	0
27	Collision_Rx Interrupt	RGMII: this bit specifies that a collision occurred in half-duplex mode.	R/W	0
26	Abort	Packet aborted.	R/W	0
25	Underrun	Underrun occurred while fetching packet from memory.	R/W	0
24	Discard	GMAC discard	R/W	0
23	AFOverflow	Async. FIFO overflow	R/W	0
22	TagFull	All 32 tags for Rx packets are being used	R/W	0
21	Class3Full	Class 3 FIFO is full	R/W	0
20	C3EarlyFull	Class 3 FIFO > watermark	R/W	0
19	Class2Full	Class 2 FIFO is full	R/W	0
18	C2EarlyFull	Class 2 FIFO > watermark	R/W	0
17	Class1Full	Class 1 FIFO is full	R/W	0
16	C1EarlyFull	Class 1 FIFO > watermark	R/W	0
15	Class0Full	Class 0 FIFO is full	R/W	0
14	C0EarlyFull	Class 0 FIFO > watermark	R/W	0
13	RxDATAFull	Send interrupt when Rx Data FIFO is full	R/W	0
12	RxEarlyFull	Rx Data FIFO > watermark	R/W	0
11	Reserved	Reserved	R	0
10	Reserved	Reserved	R	0

Bits	Name	Description	R/W	Reset
9	RFreeEmpty	Free-In Descriptor FIFO is empty	R/W	0
8	RFEarlyEmpty	Free-In Descriptor FIFO < watermark	R/W	0
7	SpillOrP2PError	ECC error on Spill channel of P2P channel	R/W	0
6	Reserved	Reserved	R	0
5	FreeDescFull	Free-Out Descriptor FIFO is full	R/W	0
4	FreeEarlyFull	Free Descriptor FIFO > watermark	R/W	0
3	TxFetchError	Tx Data fetch encountered bus error	R/W	0
2	Carry	Statistics counter overflow	R/W	0
1	MDInt	Management port interrupt. Interrupts when cable is connected/disconnected, speed changes, etc.	R/W	0
0	TxIllegal	Illegal Tx descriptor was encountered	R/W	0

a. Bits[31:29] are present only for GMAC0 base offset, these bits are reserved for other GMAC base offsets.

13.10.2.8 Reserved

Register ID: 0x0A7

Address Offset: 0x29C

Bits	Name	Description	R/W	Reset
31:0	Reserved	Reserved	R	0

13.10.2.9 CoreControl

Register ID: 0x0A8

Address Offset: 0x2A0

Core Control Register

Bits	Name	Description	R/W	Reset
31:10	Reserved	Reserved	R	0
9:3	ErrorID	See UseErrorID	R/W	0
2	UseErrorID	In case of Abort on Tx, if this bit is set then descriptors are sent to ErrorID instead of FBID in the descriptor.	R/W	0
1:0	Speed	Speed setting (controls clock divider) 00: 125 MHz 01: 25 MHz 10: 2.5 MHz 11: Reserved	R/W	0

13.10.2.10 PauseFrame

Register ID: 0x0AB

Address Offset: 0x2AC

Used to initiate a pause frame or determine if one has been sent.

Pause Frame Register

Bits	Name	Description	R/W	Reset
31:17	Reserved	Reserved	R	0
16	SendingPauseFrame	This bit can be polled to find out if Pause Frame has been sent. 0: Not sending pause frame 1: Sending pause frame	R	
15:0	PauseFrameTimer	Set Pause Frame Timer. Writing to this will send out a pause Frame. When software writes to this register, a pause frame is sent out. Software must poll SendingPauseFrame to determine when it has gone out.	R/W	0

13.10.2.11 Reserved

Register ID: 0x0AC – 0x0AD

Address Offset: 0x2B0 – 0x2B4

Bits	Name	Description	R/W	Reset
31:0	Reserved	Reserved	R	0

13.10.2.12 FIFOModeCtrl1

Register ID: 0x0AE

Address Offset: 0x2B8

This register is applicable only to the network port having RGMII pins.

Bits	Name	Description	R/W	Reset
31:13	Reserved	Reserved	R	0
12	FModeCtl1	FIFO Mode Ctrl1 Enable 0: Disable FIFO Mode 1: Enable FIFO Mode For chip-to-chip peer-to-peer communications. To enable FIFO Mode, you must set both this bit and bit 16 of FIFO Mode Control2 register. In XLS processors, the FIFO mode interface might behave unpredictably when the number of P2P descriptors in a packet is greater than 16. Note that this condition does not affect packets with less than 16 P2P descriptors, nor does it affect packets with P2D descriptors. See Chapter 16, "SGMII and RGMII Interface" .	R/W	0
11:0	Reserved	Reserved	R	0

13.10.2.13 FIFO Mode Ctrl2

Register ID: 0x0AF

Address Offset: 0x2BC

FIFO Mode Control; This register is applicable only to RGMII Port A.

Bits	Name	Description	R/W	Reset															
31:21	Reserved	Reserved	R	0															
20	RecvBPState	Receive Back-Pressure State 0: not active 1: Receive back-pressure state from FIFO-Mode link partner	R/W	0															
19	SendBPState	Send Back-Pressure State 0: not active 1: Send back-pressure state to FIFO-Mode link partner	R/W	0															
18:17	BackPressurePeriodSel	<table border="1"> <tr> <th>18</th><th>17</th><th>Period</th></tr> <tr> <td>0</td><td>0</td><td>Infinite</td></tr> <tr> <td>0</td><td>1</td><td>64</td></tr> <tr> <td>1</td><td>0</td><td>512</td></tr> <tr> <td>1</td><td>1</td><td>4K</td></tr> </table> <p>See Chapter 16, “SGMII and RGMII Interface”</p>	18	17	Period	0	0	Infinite	0	1	64	1	0	512	1	1	4K	R/W	0
18	17	Period																	
0	0	Infinite																	
0	1	64																	
1	0	512																	
1	1	4K																	
16	FModeCtl2	<p>FIFO Mode Ctl2 Enable 0: Disable FIFO Mode 1: Enable FIFO Mode</p> <p>For chip-to-chip peer-to-peer communications. To enable FIFO Mode, you must set both this bit and bit 12 of FIFO Mode Control1 register at 0xAE. In XLS processors, the FIFO mode interface might behave unpredictably when the number of P2P descriptors in a packet is greater than 16. Note that this condition does not affect packets with less than 16 P2P descriptors, nor does it affect packets with P2D descriptors.</p> <p>See Chapter 16, “SGMII and RGMII Interface”.</p>	R/W	0															

Bits	Name	Description	R/W	Reset
15:8	FModeHighWaterMark	<p>FIFO Word Count High Water Mark: When RxData FIFO Word Count is above this threshold, backpressure is generated. See Chapter 16, “SGMII and RGMII Interface”.</p> <p>The FIFO size is 96 cache lines by 32 bytes. This FIFO is shared between four GMACs. Thus, the portion of the FIFO allocated to a given port is 24 times 32 bytes (or 768 bytes).</p> <p>This High Water Mark increments in 32-byte portions (i.e., one cache line per increment). Thus, the maximum value that may be programmed into this field is 24 (18 hex). A High Water Mark value of ‘1’ indicates one cache line or 32 bytes. A value of ‘0x17’ indicates 23 cache lines or 736 bytes.</p> <p>Comparing RxData FIFO Word Count to this High Water Mark value is done in identical units, since the RxData FIFO Word Count is also incremented in numbers of cache lines (32-byte portions).</p> <p>XAUl: When the interface is configured as a XAUl port, the XAUl interface uses all of the FIFO normally allocated to the GMAC quad.</p>	R/W	0
7:0	FModeLowWatermark	<p>FIFO Word Count Low Water Mark: When RxData FIFO Word Count is below this threshold, backpressure is removed. See Chapter 16, “SGMII and RGMII Interface”.</p> <p>This Low Water Mark increments in 32-byte portions (i.e., one cache line per increment). Thus, the maximum value that may be programmed into this field is 24 (18 hex).</p> <p>Comparing RxData FIFO Word Count to this Low Water Mark value is done in identical units, since the RxData FIFO Word Count is also incremented in numbers of cache lines (32-byte portions).</p>	R/W	0

13.10.3 SGMII SERDES Control Registers

Each of the two Network Accelerators has a GMAC Quad, with four SGMII PHYs. For each quad, there is a set of SERDES Control Registers. Within that set, the SGMII_TX_BOOST is potentially useful to users. (Others are for internal validation/test use.)

For the Optional **XAU1** port, the four SERDES control registers are aligned with the four I/O lanes of each **XAU1** port.

To access the SERDES Control Registers of a particular Quad (or XAU1 port), the MDIO interface is used. The control registers reside at PHY ID 26. The SGMII_TX_BOOST register (or optional XAU1_TX_BOOST register) resides at Address Offset of PHY ID 26. Each Quad has its own PHY ID 26; thus each Quad has its own SGMII_TX_BOOST register (or optional XAU1_TX_BOOST register).

(Note that PHY ID 0 through 25 are assignable to external PHYs. PHY ID 27 through 30 are assigned to the four PHYs of the Quad.)

13.10.3.1 SGMII_TX_BOOST

This register controls transmit boost (pre-emphasis). There is the SGMII_TX_BOOST (or XAUI_TX_BOOST) register for each Network Accelerator (i.e., for each GMAC Quad or optional XAUI port). The programmed boost value (the ratio of the transition bit drive level to the non-transition bit drive level) is:

$$\text{boost} = -20 \log (1 - (\text{Tx_Boost} + 0.5) / 32) \text{ dB}$$

with one exception – Zeroing Tx_boost[3:0] produces 0 dB of boost.

Amplitude boost up to 5.75 dB in steps of ~0.37dB is achievable.

Tx Boost may be transitioned asynchronously. However, note that changing the register value while the SGMII PHY is transmitting may cause brief undesired transmit waveforms.

PHY ID: 26 (decimal)

Address Offset: 0x4

15	12 11	8 7	4 3	0
Tx_Boost_3	Tx_Boost_2	Tx_Boost_1	Tx_Boost_0	

Bits	Name	Description	R/W	Reset
15:12	Tx_Boost_3/7	If Accessing the Quad_0 Boost Register: This field sets the Tx boost for SGMII port 3. Incrementing this field increases the SGMII port 3 (or XAUI_0 lane 3) boost by approximately 0.37dB. If Accessing the Quad_1 Boost Register: This field sets the Tx boost for SGMII port 7. Incrementing this field increases the SGMII port 7 (or XAUI_1 lane 3) boost by approximately 0.37dB.	R/W	0
11:8	Tx_Boost_2/6	If Accessing the Quad_0 Boost Register: This field sets the Tx boost for SGMII port 2. Incrementing this field increases the SGMII port 2 (or XAUI_0 lane 2) boost by approximately 0.37dB. If Accessing the Quad_1 Boost Register: This field sets the Tx boost for SGMII port 6. Incrementing this field increases the SGMII port 6 (or XAUI_1 lane 2) boost by approximately 0.37dB.	R/W	0
7:4	Tx_Boost_1/5	If Accessing the Quad_0 Boost Register: This field sets the Tx boost for SGMII port 1. Incrementing this field increases the SGMII port 1 (or XAUI_0 lane 1) boost by approximately 0.37dB. If Accessing the Quad_1 Boost Register: This field sets the Tx boost for SGMII port 5. Incrementing this field increases the SGMII port 5 (or XAUI_1 lane 1) boost by approximately 0.37dB.	R/W	0
3:0	Tx_Boost_0/4	If Accessing the Quad_0 Boost Register: This field sets the Tx boost for SGMII port 0. Incrementing this field increases the SGMII port 0 (or XAUI_0 lane 0) boost by approximately 0.37dB. If Accessing the Quad_1 Boost Register: This field sets the Tx boost for SGMII port 4. Incrementing this field increases the SGMII port 4 (or XAUI_1 lane 0) boost by approximately 0.37dB.	R/W	0

13.10.4 DMA Credit Registers

These registers are shared among all Network Interfaces.

13.10.4.1 DmaCr0

Register ID: 0x200

Address Offset: 0x800

DMA Credit Register

Note: These credits are for the Memory Distributed Interface subsystem, not the Fast Messaging Network, which has its own credit system. They indicate how many outstanding reads and writes the DMA can have.

Bits	Name	Description	R/W	Reset
31:30	Reserved	Reserved	R	0
29:27	Data0WrMaxCr	Write credits for Interface 0 Rx duplex channel. See note above.	R/W	0
26:24	Data0RdMaxCr	Read credits for Interface 0 Tx duplex channel. See note above.	R/W	0
23:21	Data1WrMaxCr	Write credits for Interface 1 Rx duplex channel	R/W	0
20:18	Data1RdMaxCr	Read credits for Interface 1 Tx duplex channel	R/W	0
17:15	Data2WrMaxCr	Write credits for Interface 2 Rx duplex channel	R/W	0
14:12	Data2RdMaxCr	Read credits for Interface 2 Tx duplex channel	R/W	0
11:9	Data3WrMaxCr	Write credits for Interface 3 Rx duplex channel	R/W	0
8:6	Data3RdMaxCr	Read credits for Interface 3 Tx duplex channel	R/W	0
5:3	Reg_1_WrMaxCr	When Split Mode is enabled (See "FreeQCarve" on page 510.), this field contains spill-over write credits for GMACs 2 and 3. (This field has no meaning in Non-Split Mode.)	R	0
2:0	Reg_1_RdMaxCr	When Split Mode is enabled (See "FreeQCarve" on page 510.), this field contains spill-over read credits for GMACs 2 and 3. (This field has no meaning in Non-Split Mode.)	R	0

13.10.4.2 DmaCr3

Register ID: 0x203

Address Offset: 0x80C

DMA Credit Register

Note: These credits are for the Memory Distributed Interface subsystem, not the Fast Messaging Network, which has its own credit system. They indicate how many outstanding reads and writes the DMA can have.

Bits	Name	Description	R/W	Reset
31:30	Reserved	Reserved	R	0
29:27	Reserved	Reserved	R	0
26:24	Reserved	Reserved	R	0
23:21	SpClassWrMaxCr	Write credits for DMA Class Descriptor Spill-over	R/W	0
20:18	SpClassRdMaxCr	Read credits for DMA Class Descriptor Spill-over	R/W	0
17:15	Reserved	Reserved	R	0
14:12	P2PRdMaxCr	Read credits for P2P channel	R/W	0
11:9	RegFrInWrMaxCr	When in Non-Split Mode, this field contains spill-over write credits for DMA RegF Descriptor. When Split Mode is enabled (See “FreeQCarve” on page 510.), this field contains spill-over write credits for DMA RegF Descriptor for GMACs 0 and 1.	R/W	0
8:6	RegFrInRdMaxCr	When in Non-Split Mode, this field contains spill-over read credits for DMA RegF Descriptor. When Split Mode is enabled (See “FreeQCarve” on page 510.), this field contains spill-over read credits for DMA RegF Descriptor for GMACs 0 and 1.	R/W	0
5:3	FrOutWrMaxCr	Write credits for DMA Free Descriptor Out Spill-over	R/W	0
2:0	FrOutRdMaxCr	Read credits for DMA Free Descriptor Out Spill-over	R/W	0

13.10.5 DMA Spill Control Registers

These registers are shared among all Network Interfaces.

13.10.5.1 RegFrInSpillMemStart0

Register ID: 0x204

Address Offset: 0x810

Start address of Regular Free Descriptor In Spill-over. Cache-line aligned and cacheline sized memory addresses.

Bits	Name	Description	R/W	Reset
31:0	RegFrInSpillMemStart0	LSB — Start address of Regular Free Descriptor In Spill-over	R/W	0

13.10.5.2 RegFrInSpillMemStart1

Register ID: 0x205

Address Offset: 0x814

Extended start address of Regular Free Descriptor In Spill-over. Cache-line aligned and cacheline sized memory addresses.

Bits	Name	Description	R/W	Reset
31:6	Reserved	Reserved	R	0
5	RdExclusive	Read Exclusive. 0: Read operation will not affect the cached data (the data will not be purged from the cache after the read operation). 1: Cache data will be discarded without being written back to external memory Note that if this bit is set, bit 3 of this register (L2RdAlloc) must be cleared.	R/W	0
4	L2WrAlloc	L2 allocate for write.	R/W	0
3	L2RdAlloc	L2 allocate for read. Note that if this bit is set, bit 5 of this register (RdExclusive) must be cleared.	R/W	0
2:0	MemStartMSB	RegFrInSpillMemStart[39:37]	R/W	0

Note: Combined with FrInSpillMemStart0 to provide the 35-bit address (39:5) for RegFrIn Spill-over Memory Start.

13.10.5.3 RegFrInSpillMemSize

Register ID: 0x206

Address Offset: 0x818

Size of Regular Free Descriptor Spill-over Memory.

Bits	Name	Description	R/W	Reset
31:0	RegFrInSpillMemSize	Size of Regular Free Descriptor Spill-over Memory	R/W	0

13.10.5.4 FrOutSpillMemStart0

Register ID: 0x207

Address Offset: 0x81C

Start address of Free Descriptor Out Spill-over.

Bits	Name	Description	R/W	Reset
31:0	FrOutSpillMemStart0	LSB — Start address of Free Descriptor Out Spill-over	R/W	0

NETLOGIC
CONFIDENTIAL

13.10.5.5 FrOutSpillMemStart1

Register ID: 0x208

Address Offset: 0x820

Extended Start address of Free Descriptor Out Spill-over.

Bits	Name	Description	R/W	Reset
31:6	Reserved	Reserved	R	0
5	RdExclusive	Read Exclusive. 0: Read operation will not affect the cached data (the data will not be purged from the cache after the read operation). 1: Cache data will be discarded without being written back to external memory.	R/W	0
4	L2WrAlloc	L2 allocate for write.	R/W	0
3	L2RdAlloc	L2 allocate for read.	R/W	0
2:0	FrOutSpillMemStart1	MSB — Start address of Free Descriptor Out Spill-over	R/W	0

Note: Combined with FrOutSpillMemStart0 to provide the 35-bit address (39:5) for Free In Spill-over Memory Start.

13.10.5.6 FrOutSpillMemSize

Register ID: 0x209

Address Offset: 0x824

Free Out Descriptor Spill-over Memory Size.

Bits	Name	Description	R/W	Reset
31:0	FrOutSpillMemSize	Size — Free Out Descriptor Spill-over Memory	R/W	0

13.10.5.7 Class0SpillMemStart0

Register ID: 0x20A

Address Offset: 0x828

Start address of Class0 Descriptor Spill-over.

Bits	Name	Description	R/W	Reset
31:0	Class0SpillMemStart0	LSB — Start address of Class0 Descriptor Spill-over	R/W	0

13.10.5.8 Class0SpillMemStart1

Register ID: 0x20B

Address Offset: 0x82C

Extended start address of Class0 Descriptor Spill-over.

Bits	Name	Description	R/W	Reset
31:6	Reserved	Reserved	R	0
5	RdExclusive	Read Exclusive. 0: Read operation will not affect the cached data (the data will not be purged from the cache after the read operation). 1: Cache data will be discarded without being written back to external memory	R/W	0
4	L2WrAlloc	L2 allocate for write.	R/W	0

Bits	Name	Description	R/W	Reset
3	L2RdAlloc	L2 allocate for read.	R/W	0
2:0	MemStartMSB	Class0SpillMemStart[39:37]	R/W	0

NETLOGIC
CONFIDENTIAL

13.10.5.9 Class0SpillMemSize

Register ID: 0x20C

Address Offset: 0x830

Size of Class0 Descriptor Spill-over Memory.

Bits	Name	Description	R/W	Reset
31:0	Class0SpillMemSize	Size of Class0 Descriptor Spill-over Memory	R/W	0

13.10.5.10 Reserved

Register ID: 0x20D

Address Offset: 0x834

Bits	Name	Description	R/W	Reset
31:0	Reserved	Reserved	R	0

13.10.5.11 Reserved

Register ID: 0x20E

Address Offset: 0x838

Bits	Name	Description	R/W	Reset
31:0	Reserved	Reserved	R	0

13.10.5.12 Reserved

Register ID: 0x20F

Address Offset: 0x83C

Bits	Name	Description	R/W	Reset
31:0	Reserved	Reserved	R	0

13.10.5.13 Class1SpillMemStart0

Register ID: 0x210

Address Offset: 0x840

Start address of Class1 Descriptor Spill-over.

Bits	Name	Description	R/W	Reset
31:0	Class1SpillMemStart0	LSB — Start address of Class1 Descriptor Spill-over	R/W	0

13.10.5.14 Class1SpillMemStart1

Register ID: 0x211

Address Offset: 0x844

Extended start address of Class1 Descriptor Spill-over.

Bits	Name	Description	R/W	Reset
31:6	Reserved	Reserved	R	0
5	RdExclusive	Read Exclusive. 0: Read operation will not affect the cached data (the data will not be purged from the cache after the read operation). 1: Cache data will be discarded without being written back to external memory	R/W	0
4	L2WrAlloc	L2 allocate for write.	R/W	0
3	L2RdAlloc	L2 allocate for read.	R/W	0
2:0	MemStartMSB	Class1SpillMemStart [39:37]	R/W	0

13.10.5.15 Class1SpillMemSize

Register ID: 0x212

Address Offset: 0x848

Size of Class1 descriptor Spill-over Memory.

Bits	Name	Description	R/W	Reset
31:0	Class1SpillMemSize	Size of Class1 descriptor Spill-over Memory	R/W	0

13.10.5.16 Class2SpillMemStart0

Register ID: 0x213

Address Offset: 0x84C

Start address of Class2 Descriptor Spill-over.

Bits	Name	Description	R/W	Reset
31:0	Class2SpillMemStart0	LSB — Start address of Class2 Descriptor Spill-over	R/W	0

13.10.5.17 Class2SpillMemStart1

Register ID: 0x214

Address Offset: 0x850

Extended start address of Class2 Descriptor Spill-over.

Bits	Name	Description	R/W	Reset
31:6	Reserved	Reserved	R	0
5	RdExclusive	Read Exclusive. 0: Read operation will not affect the cached data (the data will not be purged from the cache after the read operation). 1: Cache data will be discarded without being written back to external memory	R/W	0
4	L2WrAlloc	L2 allocate for write.	R/W	0
3	L2RdAlloc	L2 allocate for read.	R/W	0
2:0	MemStartMSB	Class2SpillMemStart [39:37]	R/W	0

13.10.5.18 Class2SpillMemSize

Register ID: 0x215

Address Offset: 0x854

Size of Class2 Descriptor Spill-over Memory.

Bits	Name	Description	R/W	Reset
31:0	Class2SpillMemSize	Size of Class2 Descriptor Spill-over Memory	R/W	0

13.10.5.19 Class3SpillMemStart0

Register ID: 0x216

Address Offset: 0x858

Start address of Class3 Descriptor Spill-over.

Bits	Name	Description	R/W	Reset
31:0	Class3SpillMemStart0	LSB — Start address of Class3 Descriptor Spill-over	R/W	0

13.10.5.20 Class3SpillMemStart1

Register ID: 0x217

Address Offset: 0x85C

Extended Start address of Class3 Descriptor Spill-over.

Bits	Name	Description	R/W	Reset
31:6	Reserved	Reserved	R	0
5	RdExclusive	Read Exclusive. 0: Read operation will not affect the cached data (the data will not be purged from the cache after the read operation). 1: Cache data will be discarded without being written back to external memory	R/W	0
4	L2WrAlloc	L2 allocate for write. 0: disable 1: enable	R/W	0
3	L2RdAlloc	L2 allocate for read. 0: disable 1: enable	R/W	0
2:0	MemStartMSB	Class3SpillMemStart [39:37]	R/W	0

13.10.5.21 Class3SpillMemSize

Register ID: 0x218

Address Offset: 0x860

Size of Class3 Descriptor Spill-over Memory.

Bits	Name	Description	R/W	Reset
31:0	Class3SpillMemSize	Size of Class3 Descriptor Spill-over Memory	R/W	0

13.10.5.22 RegFrIn1SpillMemStart0

Register ID: 0x219

Address Offset: 0x864

Start address of Regular Free Descriptor In Spill Over.

Bits	Name	Description	R/W	Reset
31:0	RegFrInSpillMemStart0	LSB – Start address of Regular Free Descriptor In Spill Over	R/W	0

13.10.5.23 RegFrIn1SpillMemStart1

Register ID: 0x21A

Address Offset: 0x868

Extended start address of Regular Free Descriptor In Spill Over. Combined with FrInSpillMemStart0 to provide the 35-bit address (39:5) for RegFrIn Spill Over Memory Start..

Bits	Name	Description	R/W	Reset
31:6	Reserved	Reserved	R	0
5	RdExclusive	Read Exclusive. 0: Read operation will not affect the cached data (the data will not be purged from the cache after the read operation). 1: Cache data will be discarded without being written back to external memory	R/W	0
4	L2WrAlloc	L2 allocate for write. 0: disable 1: enable	R/W	0
3	L2RdAlloc	L2 allocate for read. 0: disable 1: enable	R/W	0
2:0	MemStartMSB	RegFrInSpillMemStart[39:37]	R/W	0

13.10.5.24 RegFrIn1SpillMemSize

Register ID: 0x21B

Address Offset: 0x86C

Regular Free Descriptor Spill Over Memory..

Bits	Name	Description	R/W	Reset
31:0	RegFrInSpillMemSize	Size – Regular Free Descriptor Spill Over Memory	R/W	0

13.10.6 Transmit Data FIFO Configuration

The following registers are used to configure the transmit data FIFOs. See [Figure 13-18](#) for description of the Transmit Data FIFO configuration.

13.10.6.1 TX_DATA_FIFO0

Register ID: 0x221

Address Offset: 0x870

Bits	Name	Description	R/W	Reset
31:24	TX0_START	Transmit Data FIFO_0 Start Address	R/W	0
23:16	TX0_DEPTH	Transmit Data FIFO_0 depth In XAUI mode set to 224 (which means 192), otherwise set to 48	R/W	48
15:8	TX1_START	Transmit Data FIFO_1 Start Address	R/W	48
7:0	TX1_DEPTH	Transmit Data FIFO_1 depth	R/W	48

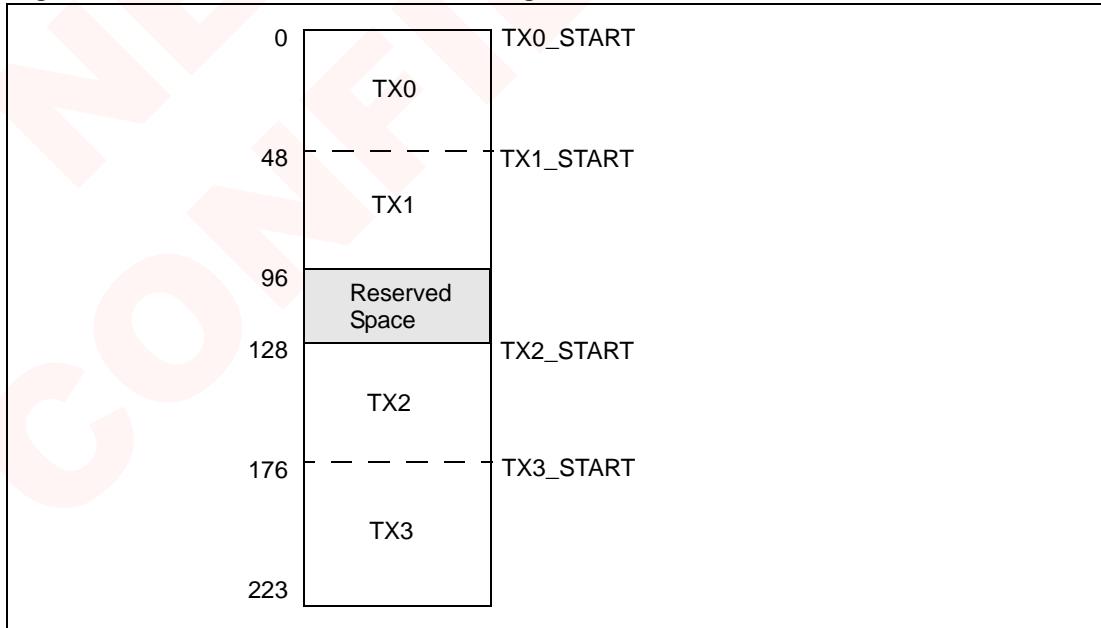
13.10.6.2 TX_DATA_FIFO1

Register ID: 0x222

Address Offset: 0x874

Bits	Name	Description	R/W	Reset
31:24	TX2_START	Transmit Data FIFO_2 Start Address	R/W	128
23:16	TX2_DEPTH	Transmit Data FIFO_2 depth	R/W	48
15:8	TX3_START	Transmit Data FIFO_3 Start Address	R/W	176
7:0	TX3_DEPTH	Transmit Data FIFO_3 depth	R/W	48

Figure 13-18. Transmit Data FIFO Configuration



NETLOGIC
CONFIDENTIAL

13.10.7 Split Mode Support

13.10.7.1 FreeQCarve

Register ID: 0x233

Address Offset: 0x8CC

Used in “Split Mode” (also known as “Virtual MIPS Mode”) to support multiple operating systems. See [Chapter 3, “Virtual MIPS Mode”](#). When FreeQCarvingMode is set, the Rx Free In Descriptor FIFO is carved into 2 segments for use by the two resident operating systems. When FreeOutQCarvingMode is set, the Rx Free Out Descriptor FIFO is carved into 2 segments for use by the two resident operating systems.

Note: When FreeOutQCarvingMode bit is set, then free out spill memory size should be set to 0. In split mode, In this case, Free Out Descriptors should be drained by the destination Threads (or Network Interface).

Bits	Name	Description	R/W	Reset
31:19	Reserved	Reserved	R	0
18:5	Reserved	Reserved	R	0
6	FreeOutQCarvingMode	Carve Free Out Descriptor FIFO into two segments. 0: one segment 1: two segments	R/W	0
5	FreeQCarvingMode	Carve Free In Descriptor FIFO into two segments. 0: one segment 1: two segments	R/W	0
4:0	MaxUnderRunRetryCount	Maximum Retry Count for Under-run Retries	R/W	0x1F

13.10.8 FIFO Watermark Registers

Class watermark registers are used to monitor the level of usage of the four Class FIFOs in the PDE.

13.10.8.1 ClassWatermarks

Register ID: 0x244

Address Offset: 0x910

Watermarks for Classes

Bits	Name	Description	R/W	Reset
31:29	Reserved	Reserved	R	0
28:24	Class0Watermark	Watermark for Class0 in number of 64-bit entries	R/W	0x10
23:21	Reserved	Reserved	R	0
20:16	Class1Watermark	Watermark for Class1 in number of 64-bit entries	R/W	0x10
15:13	Reserved	Reserved	R	0
12:8	Class2Watermark	Watermark for Class2 in number of 64-bit entries	R/W	0x10
7:5	Reserved	Reserved	R	0
4:0	Class3Watermark	Watermark for Class3 in number of 64-bit entries	R/W	0x10

13.10.8.2 RxWatermarks0

Register ID: 0x245

Address Offset: 0x914

Watermarks for Rx data.

Bits	Name	Description	R/W	Reset
31	<i>Reserved</i>	<i>Reserved</i>	R	0
30:24	Rx0DataWatermark	Watermark for Rx0Data in number of 256-bit entries	R/W	0x30
23	<i>Reserved</i>	<i>Reserved</i>	R	0
22:16	Rx1DataWatermark	Watermark for Rx1Data in number of 256-bit entries	R/W	0x30
15	<i>Reserved</i>	<i>Reserved</i>	R	0
14:8	Rx2DataWatermark	Watermark for Rx2Data in number of 256-bit entries.	R/W	0x30
7	<i>Reserved</i>	<i>Reserved</i>	R	0
6:0	Rx3DataWatermark	Watermark for Rx3Data in number of 256-bit entries	R/W	0x30

13.10.8.3 FreeWatermarks

Register ID: 0x249

Address Offset: 0x924

Watermark for Free Out Descriptor FIFO.

Bits	Name	Description	R/W	Reset
31:16	FreeOutWatermark	Watermark for Free Out Descriptor FIFO in number of 64-bit entries	R/W	0x10
15	<i>Reserved</i>	<i>Reserved</i>	R	0
14:8	<i>Reserved</i>	<i>Reserved</i>	R	0
7	<i>Reserved</i>	<i>Reserved</i>	R	0
6:0	RegFrWatermark	Watermark for Free In Descriptor FIFO in number of 64-bit entries	R/W	0x40

13.10.9 IEEE1588_PTP Source Registers

The following registers are used to configure the IEEE1588_PTP control registers.

13.10.9.1 1588_PTP_SOURCE_DIV

Register ID: 0x24A

Address Offset: 0x928

1588_PTP SourceClock Divider .

Bits	Name	Description	R/W	Reset
31:0	SOURCE_DIV	Source Clock is divided by (this value +1) 0x00: No divide - bypass the clock divider 0x01: Divide by 2 0x02: Divide by 3 0x03: Divide by 4 . . . 0x1F Divide by 32	R/W	0

13.10.9.2 1588_PTP_SOURCE

Register ID: 0x24B

Address Offset: 0x92C

1588_PTP Source

Bits	Name	Description	R/W	Reset
31:1	Reserved	Reserved	R	0
0	SOURCE	Source 0: GMAC/XGMAC RefCLK (125-MHz GMAC Mode, and 156.25-MHz in XGMAC Mode) 1: Core Clock (approximately 1.2 GHz) or Half Core Clock, depending on the value of the BHCM field in the GPIO RESET Configuration register (see Section 23.3.5.1, “GPIO Reset Configuration” for details)	R/W	0

13.11 PDE Class Registers

The PDE Class registers are bit masks that determine Bucket membership in each Class. Bits set in these registers cause the indicated Bucket to be included as a participant in the round-robin scheduling for this Class. Because they are bit masks specified per Class, it is possible for Buckets to appear as destinations in multiple Classes, or for Bucket groups to be mutually exclusive per Class.

13.11.1 PDE_CLASS0

Register ID: 0x300 / 0x301

Address Offset: 0xC00 / 0xC04

Bit mask for Bucket membership in Class 0.

Bits [31:0] are addressed at 0x300 and bits [63:32] are addressed at 0x301.

This is the bit mask for Bucket membership in Class 0. Bits set in this register include the indicated Bucket as a participant in the round-robin scheduling for this Class.

Bits	Name	Description	R/W	Reset
63:56	Reserved	Reserved	R	0
55:48	Reserved	Reserved	R	0
47:40	Reserved	Reserved	R	0
39:32	Reserved	Reserved	R	0
31:24	CPU3_Bucket_Mask	Bit 31: Bucket 7 for CPU3 Bit 24: Bucket 0 for CPU3	R/W	0
23:16	CPU2_Bucket_Mask	Bit 23: Bucket 7 for CPU2 Bit 16: Bucket 0 for CPU2	R/W	0
15:8	CPU1_Bucket_Mask	Bit 15: Bucket 7 for CPU1 Bit 8: Bucket 0 for CPU1	R/W	0
7:0	CPU0_Bucket_Mask	Bit 7: Bucket 7 for CPU0 Bit 0: Bucket 0 for CPU0	R/W	0

13.11.2 PDE_CLASS1

Register ID: 0x302 / 0x303

Address Offset: 0xC08 / C0C

Bit mask for Bucket membership in Class 1.

Bits [31:0] are addressed at 0x302 and bits [63:32] are addressed at 0x303.

This is the bit mask for Bucket membership in Class 1. Bits set in this register include the indicated Bucket as a participant in the round-robin scheduling for this Class.

Bits	Name	Description	R/W	Reset
63:56	Reserved	Reserved	R	0
55:48	Reserved	Reserved	R	0
47:40	Reserved	Reserved	R	0
39:32	Reserved	Reserved	R	0
31:24	CPU3_Bucket_Mask	Bit 31: Bucket 7 for CPU3 Bit 24: Bucket 0 for CPU3	R/W	0
23:16	CPU2_Bucket_Mask	Bit 23: Bucket 7 for CPU2 Bit 16: Bucket 0 for CPU2	R/W	0
15:8	CPU1_Bucket_Mask	Bit 15: Bucket 7 for CPU1 Bit 8: Bucket 0 for CPU1	R/W	0
7:0	CPU0_Bucket_Mask	Bit 7: Bucket 7 for CPU0 Bit 0: Bucket 0 for CPU0	R/W	0

13.11.3 PDE_CLASS2

Register ID: 0x304 / 0x305

Address Offset: 0xC10 / C14

Bit mask for Bucket membership in Class 2.

Bits [31:0] are addressed at 0x304 and bits [63:32] are addressed at 0x305.

This is the bit mask for Bucket membership in Class 2. Bits set in this register include the indicated Bucket as a participant in the round-robin scheduling for this Class.

Bits	Name	Description	R/W	Reset
63:56	Reserved	Reserved	R/W	0
55:48	Reserved	Reserved	R	0
47:40	Reserved	Reserved	R	0
39:32	Reserved	Reserved	R	0
31:24	CPU3_Bucket_Mask	Bit 31: Bucket 7 for CPU3 Bit 24: Bucket 0 for CPU3	R/W	0
23:16	CPU2_Bucket_Mask	Bit 23: Bucket 7 for CPU2 Bit 16: Bucket 0 for CPU2	R/W	0
15:8	CPU1_Bucket_Mask	Bit 15: Bucket 7 for CPU1 Bit 8: Bucket 0 for CPU1	R/W	0
7:0	CPU0_Bucket_Mask	Bit 7: Bucket 7 for CPU0 Bit 0: Bucket 0 for CPU0	R/W	0

13.11.4 PDE_CLASS3

Register ID: 0x306 / 0x307

Address Offset: 0xC18 / 0xC1C

Bit mask for Bucket membership in Class 3.

Bits [31:0] are addressed at 0x306 and bits [63:32] are addressed at 0x307.

This is the bit mask for Bucket membership in Class 3. Bits set in this register include the indicated Bucket as a participant in the round-robin scheduling for this Class.

Bits	Name	Description	R/W	Reset
63:56	Reserved	Reserved	R	0
55:48	Reserved	Reserved	R	0
47:40	Reserved	Reserved	R	0
39:32	Reserved	Reserved	R	0
31:24	CPU3_Bucket_Mask	Bit 31: Bucket 7 for CPU3 Bit 24: Bucket 0 for CPU3	R/W	0
23:16	CPU2_Bucket_Mask	Bit 23: Bucket 7 for CPU2 Bit 16: Bucket 0 for CPU2	R/W	0
15:8	CPU1_Bucket_Mask	Bit 15: Bucket 7 for CPU1 Bit 8: Bucket 0 for CPU1	R/W	0
7:0	CPU0_Bucket_Mask	Bit 7: Bucket 7 for CPU0 Bit 0: Bucket 0 for CPU0	R/W	0

13.11.5 Debug Registers

The Network Accelerators contain a number of debugging registers, such as performance counters and word count registers for the various FIFOs. See [Section 25.11, “Network Accelerator Debug Registers”](#).

NETLOGIC
CONFIDENTIAL



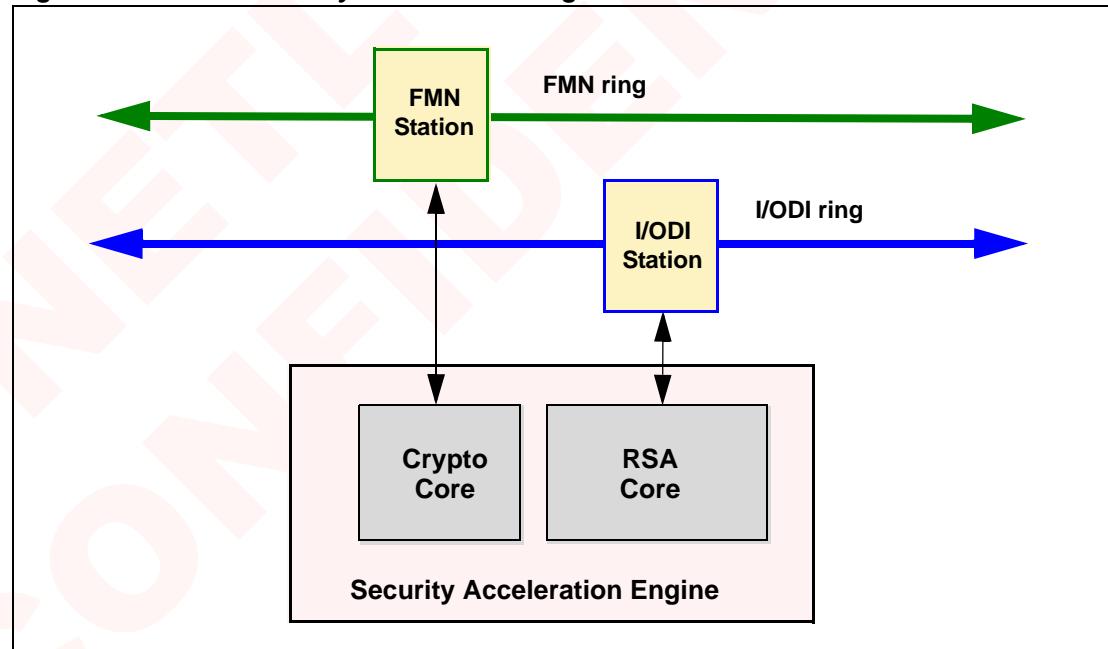
Chapter 14 Security Acceleration Engine

14.1 Introduction

The Security Acceleration Engine (SAE) enables up to 2.5 Gbps of bulk cryptographic processing for security protocols such as IPSec and SSL. The SAE provides encryption/decryption and authentication-related hashing acceleration support for industry-standard security algorithms. It supports fragmented payloads and arbitrary size payloads.

The SAE has one Crypto Core (for encryption/decryption and authentication functions) and one RSA core, each of which is designed to operate independently, as shown in [Figure 14-1](#). The RSA core provides combined RSA (Rivest Shamir Adleman key encryption algorithm) and ECC (Elliptic Curve Cryptography) functionality, to accelerate RSA-based and ECC-based key exchanges.

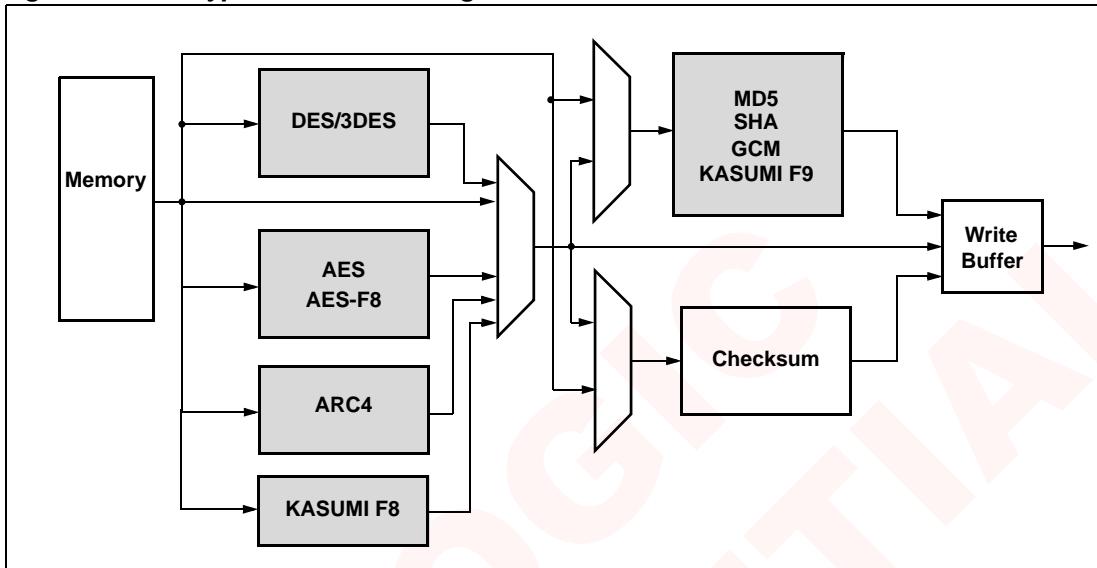
Figure 14-1. XLS Security Acceleration Engine



14.1.1 Crypto Core

The XLS Processor Crypto core block diagram is shown in [Figure 14-2](#).

Figure 14-2. Crypto Core Block Diagram



The Crypto core supports encryption, authentication, and TCP checksum generation. In addition to DES, 3DES, and AES, each Crypto core has an ARC4 and Kasumi F8 engine and corresponding cipher selection encodings. Furthermore, depending on the requirements of a given security application, a Crypto core can encrypt and authenticate a packet, or authenticate and decrypt a packet.

The XLS Architecture provides hardware support for the following security/hashing algorithms in the Crypto core:

- Encryption/Decryption:
 - AES
 - Key sizes: 128, 192 and 256 bits
 - ECB, CBC, CFB, OFB, CTR, and F8 modes
 - DES/3DES
 - DES ECB Mode
 - DES CBC Mode
 - 3DES ECB Mode
 - 3DES CBC Mode
 - ARC4
 - Kasumi F8
- Authentication:
 - SHA-1
 - SHA-256
 - SHA-384
 - SHA-512
 - MD5
 - GCM
 - Kasumi F9
- Input to the Authentication module in the Crypto core may be selected from input to, or output of, any of the encryption/decryption modules
- HMAC support for SHAs and MD5 authentication algorithms mentioned above
- Support for all padding modes

The performance of a Crypto core is described in [Table 14-1](#), shown at high and low processor clock speeds.

Table 14-1. Crypto Core Raw Performance (at Processor Clock Speed)

Metric	Gbps		
	for 1.2 GHz Processor Clock	for 750 MHz Processor Clock	for 600 MHz Processor Clock
Authentication			
SHA384	5.2	3.3	2.6
SHA512	5.2	3.3	2.6
GCM	4.0	2.5	2.0
KASUMI F9	2.2	1.4	1.1
HMAC-MD5	3.9	2.4	1.95
HMAC-SHA1	5.2	3.3	2.6
HMAC-SHA256	3.9	2.4	1.95
Encryption			
DES	7.7	4.8	3.85
3DES	3.0	1.9	1.5
AES128	3.0	1.9	1.5
AES192	2.6	1.6	1.3
AES256	2.3	1.4	1.15
ARC4	5.5	3.4	2.75
KASUMI F8	2.2	1.4	1.1

14.1.1.1

ARC4 Cipher Engine

In addition to DES, 3DES, and AES, each Crypto core has an ARC4 module and corresponding cipher selection encoding. The ARC4 Cipher module supports keys between eight and 256 bits in length with 8-bit granularity. The ARC4 module communicates with the Crypto core datapath the same way the DES, 3DES, and AES engines communicate. The Arc4 State may be loaded from memory, saved to memory, or preserved for each operation. Its data path is eight bits wide so that each 64-bit word is processed in eight passes. The raw throughput of the engine is 4.57 bits per clock.

In the stateless case where the Arc4 Sbox is recalculated for every operation, the actual raw throughput becomes a function of the payload size, due to the 256 clocks required for Sbox initialization.

Although the Security Engine datapath is 8 bytes wide, the user may specify exactly how many byte to encrypt in the last double word (Arc4ByteCount in [Table 14-12: Cipher Destination \(dstDataSettings\)](#)).

Table 14-2. Raw ARC4 Performance in Stateless Case

Payload Size (Bytes)	Bits / Clock	Gbps@ 1.2 GHz	for 750 MHz Processor Clock	for 600 MHz Processor Clock
64	1.39	1.7	1.063	0.85
128	2.13	2.5	1.563	1.75
256	2.91	3.5	2.188	1.75
1.75	3.55	4.3	2.688	2.15
1024	4.00	5.0	3.125	2.5
2048	4.26	5.0	3.125	2.5
4096	4.41	5.3	3.313	2.65
8192	4.49	6.5	4.063	3.25
16384	4.53	5.5	3.438	2.75

14.1.1.2

Fragmented Payload Support and Arbitrary Size Payloads

Data Descriptors can instruct the executing Crypto Core to read the next cache line following the current Data Descriptor and interpret it as a subsequent Data Descriptor after the current operation on the present data block is complete. An arbitrary number of such Data Descriptors may be concatenated to describe a particular payload. Each such Data Descriptor may resolve to a different data source and different data destination memory location. The payload blocks described by each Data Descriptor will be processed in order, and the cipher, authentication, and checksum engines will preserve state between blocks, treating them as one whole payload with possibly scattered (non-contiguous) source and destination addresses.

Similarly, the destination pointers may be specified so as to obtain one contiguous payload result (i.e. gather if the source was non-contiguous) or separate blocks (i.e. scatter if the source was contiguous). See [Section 14.2.3, “Fragmented Payload Support and Arbitrary Size Payloads”](#).

14.1.2 RSA Core

The SAE also contains an RSA core providing combined RSA (Rivest Shamir Adleman key encryption algorithm) and ECC (Elliptic Curve Cryptography) functionality, to accelerate RSA-based and ECC-based key exchanges ([Figure 14-3](#)). Because the RSA core is separate, the XLS Processor can support both IPSec and SSL VPNs concurrently on the same platform.

The RSA core can perform 1440 RSA exponentiations/sec at 1.2 GHz for 1024-bit keys. Utilizing the Chinese Remainder Theorem (CRT), the RSA Core can perform 2,800 exponentiations per second for 1024-bit keys. A summary of the ECC (Elliptic Curve Cryptography) performance characteristics is shown in [Table 14-3](#).

Note that the following requirements must be met for the RSA/DH engine to operate correctly:

- Modulus must be odd
- Arbitrary length (\leq 1024 bits) modulus is allowed, but it must be zero-extended to the next larger size supported by the engine:
 - Zero-extend to 512 bits if block-width bit is set to 0
 - Zero-extend to 1024 bits if block-width bit is set to 1

The ECC hardware in the RSA core supports binary and prime algorithms for a total of 9 key lengths. The performance varies based on the chosen algorithm and key size. At 1.2 GHz, the highest throughput (3840 point multiplications per second) is obtained from the 163-bit binary field algorithm, and the lowest (152 point multiplications per second) is obtained from the 512-bit prime field algorithm.

Figure 14-3. RSA Core

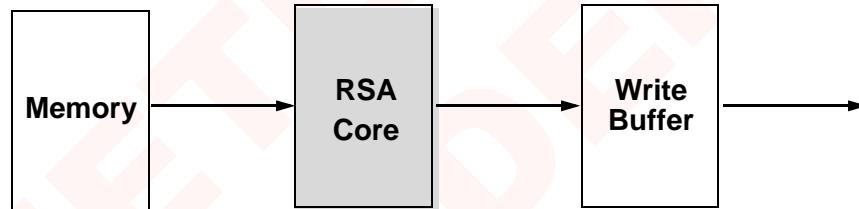


Table 14-3. ECC (Elliptic Curve Cryptography) Performance Characteristics

OP	Nb ECC core cycles	Nb core cycles	Ops/sec @ 1.2 Ghz	Ops/sec @ 1.0 Ghz	Ops/sec @ 0.8 Ghz	Ops/sec @ 0.6 Ghz
512 prime p-mul r1-curve	2.70e6	8.10e6	148	123	98.8	74
384 prime p-mul r1-curve	2.30e6	6.90e6	174	145	116	87
256 prime p-mul r1-curve	1.13e6	3.39e6	354	295	236	177
224 prime p-mul r1-curve	1.04e6	3.12e6	385	321	256	193
192 prime p-mul r1-curve	9.00e5	2.70e6	444	370	296	222
160 prime p-mul r1-curve	8.53e5	2.56e6	469	391	313	234
233 binary p-mul k-curve	2.13e5	6.39e5	1878	1565	1252	939
191 binary p-mul k-curve	1.43e5	4.29e5	2797	2331	1865	1398
163 binary p-mul k-curve	1.05e5	3.15e5	3810	3175	2540	1905

14.2 Theory of Operation

The Security Acceleration Engine (SAE) connects via an FMN Station on the Fast Messaging Network to send packet data to, and receive packet data from, processor Threads. The SAE intercepts FMN messages posted to bucket 120 for the Crypto core and bucket 121 for the RSA core ([Table 12-1, “Stations and Addressable Buckets on the Fast Messaging Network,” on page 402](#) shows all associations between Bucket IDs and functional units).

The posted request is constructed out of two 64-bit message entries (refer to chapter 12 - FMN for more details), where the first 64-bit entry points to the control descriptor block, and the second 64-bit entry is dedicated to the data descriptor block. After fetching the control block information, the SAE loads:

- the source address of the message to be processed,
- the destination addresses for the processed data,
- the authentication code, and
- the check sum.

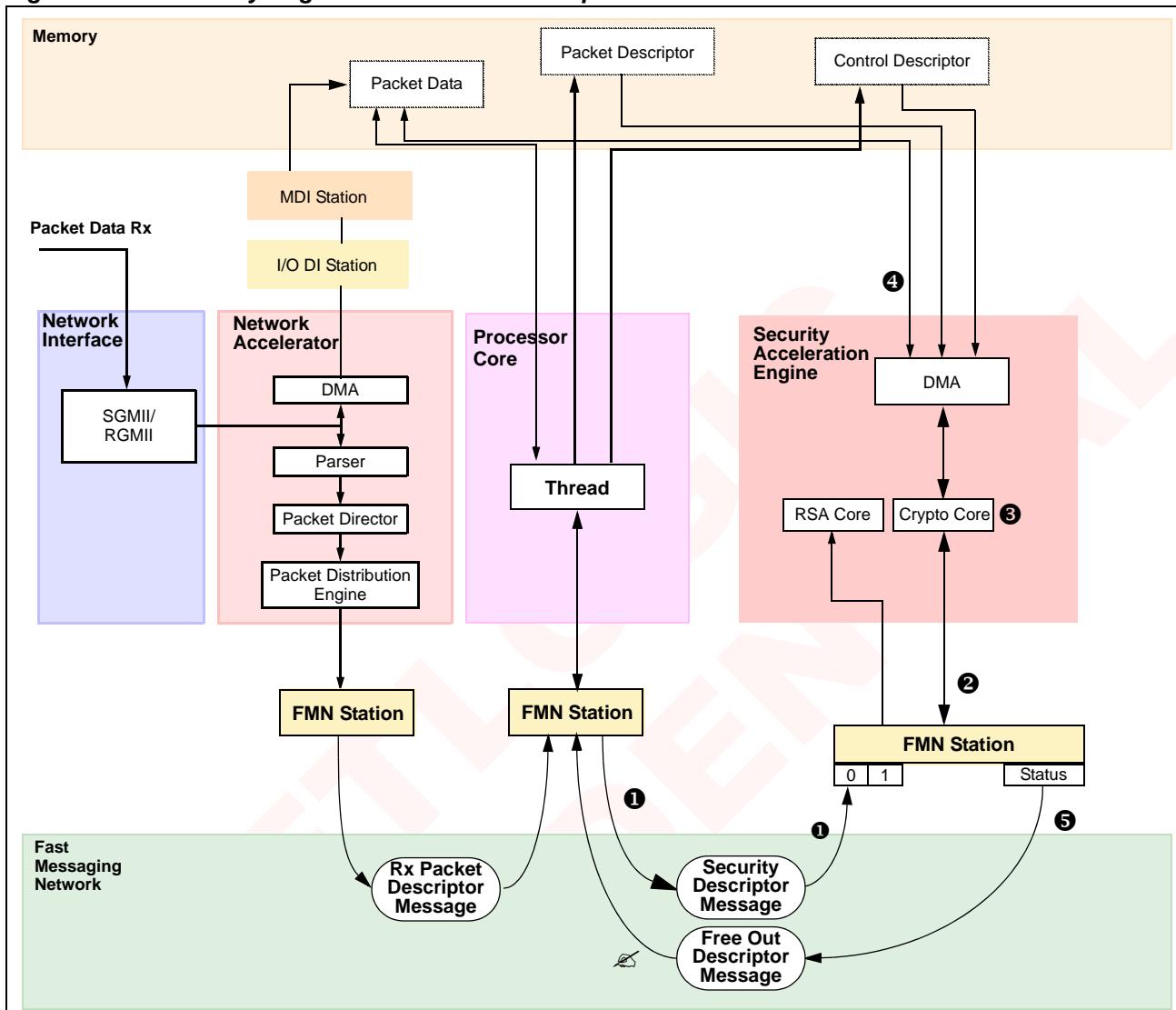
Upon completion, the SAE sends a message back to the programmed FMN station notifying of any errors and reporting the status of the previous operation.

14.2.1 Security Acceleration Engine Flow

[Figure 14-4](#) shows an example of security processing flow using the XLS Security Acceleration Engine (SAE). First, the originating nCPU constructs a Security Descriptor, which contains pointers to control and data information needed to process the packet:

1. The originating nCPU sends the Security Descriptor to Bucket 0 of the Security Acceleration Engine (Bucket ID 120).
2. The Crypto core receives the Descriptor and parses it.
3. Based on the Descriptor, the Crypto core determines:
 - Type of instruction (cipher, hash) and keys
 - Source Address, Destination Address, and data size
 - Packet data
 - Bucket ID of originating agent
4. The Crypto core processes the packet data and writes the data back to memory when encryption processing is completed.
5. At the end of processing, the Crypto core informs the Status Dispatcher that it is finished.
6. The Status Dispatcher sends a Security Free Out Descriptor to indicate the completion of operations.

The Bucket ID used as the destination of the Security Free Out Descriptor is derived from a field of the originating Security Descriptor. The originating nCPU can program this field to cause the Free Out Descriptor to return to one of its own Buckets, or to be forwarded to another nCPU’s bucket. The mechanism is not unlike a postal return receipt.

Figure 14-4. Security Engine Packet Flow - Example

14.2.2 Security Block Data and Control Exchange

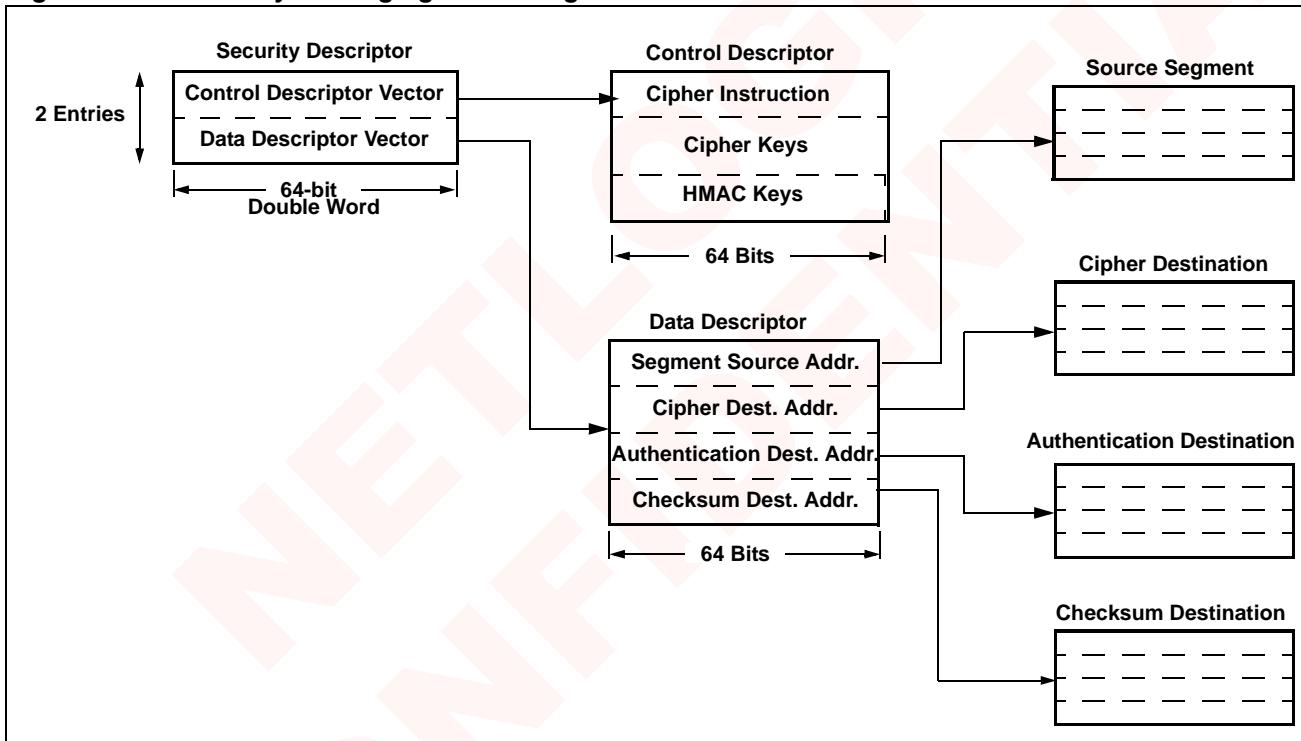
Before using the Security Acceleration Engine, software must set up a Control Descriptor and a Data Descriptor in memory. These Descriptors contain information about the packet to be processed.

A two-entry Security Packet Descriptor message containing pointers to the Control Descriptor and Data Descriptor is sent from the originating Thread to the SAE via the Fast Messaging Network. The two entries of a Security Packet Descriptor are:

- **Control Descriptor Vector** — 64-bit entry that provides a pointer to the type of instruction (cipher, hash) and keys in memory
- **Data Descriptor Vector** — 64-bit entry that provides a pointer to the data descriptor. The data descriptor contains a pointer to the address of the source segment and the address where the cipher result, authentication result, and checksum result will be stored.

The layout of the data structures is shown in [Figure 14-5](#).

Figure 14-5. Security Messaging Block Diagram



14.2.3 Fragmented Payload Support and Arbitrary Size Payloads

A Next field in the Data Descriptor instructs the executing Crypto core to read the next cache line following the current Data Descriptor line and interpret it as a subsequent Data Descriptor after the current operation on the present data chunk is complete.

An arbitrary number of such Data Descriptors may be concatenated to describe a particular payload. Each such Data Descriptor may resolve to a different data source and different data destination memory location. The payload chunks described by each Data Descriptor will be processed in order, and the cipher, authentication, and checksum engines will preserve state between chunks, treating them as one whole payload with possibly scattered (non-contiguous) source and destination addresses.

Processing continues until the Next field of the current Data Descriptor is set to 0. The operation completes and the Crypto core returns a Security Free Out Descriptor.

In order to process very large (> 16 kB) payloads of contiguous data, software must partition data in chunks of 16 kB or less and create a linked list of Data Descriptors pointing to the beginning of each chunk in the payload. Similarly, the destination pointers may be specified so as to obtain one contiguous payload result (i.e. gather if the source was non-contiguous) or separate chunks (i.e. scatter if the source was contiguous).

Minimum chunk merging granularity is 8-byte chunks, but only for chunks following the first one.

To support this feature, the Use_IV and IV_Offset fields of PacketDescriptor have been overloaded to have different functions on fragments following the first.

**NETLOGIC
CONFIDENTIAL**

14.3 Security Descriptor Message Format

Table 14-4 and **Table 14-5** shows the Security Control Descriptor. **Table 14-6** and **Table 14-7** show the Security Data Descriptor—the 2-entry message sent by the originating nCPU to the SAE through the Fast Messaging Network. The contents of the message supply the SAE with an instruction and pointers to all relevant data.

Addresses are assumed to be cacheline-aligned. That is, Address[4:0] is set to zero.

Note: All reserved fields must always be cleared to zero.

Table 14-4. Control Descriptor

Control Descriptor Vector

63:61		60:54		53	52:49	48:45	44:40	39:5	4:0
Ctrl	Rsvd	IF_L2 ALLOC		Reserved		Control Length	Rsvd	ControlAddress (35 MSB) of Control Descriptor data structure	Software Scratch0

Table 14-5. Control Descriptor Vector

Bits	Name	Description
63:61	Ctrl	Control. 0x6: Start of Packet (SOP) All other values are Reserved.
60:54	Reserved	Reserved
53	IF_L2ALLOC	L2 Allocation after read operation: 0: Structure pointed to by control address is guaranteed <i>not</i> to be in L2 cache 1: Structure pointed to by control address might be in L2 cache This option should be avoided when there is no need for further processing to minimize L2 trashing.
52:49	Reserved	Reserved = 0
48:45	ControlLength	Control length is the number of control cache lines to be read. The control length must be rounded up to the closest integer multiple of 32 bytes. The longest (sensical) control structure is \leq 416 bytes.
44:40	Reserved	Reserved = 0
39:5	ControlAddress	35 MSB of address for the Control Descriptor. Addresses are assumed to be cacheline-aligned, i.e., Address[4:0] is ignored.
4:0	SoftwareScratch0	Ignored by hardware and copied as-is to the Free Out Descriptor. Software can use these bits (for example as an ID tracking number).

Table 14-6. Data Descriptor**Data Descriptor Vector**

Data Descriptor Vector																			
63:61		60:54		53		52		51		50:46		45		44:40		39:5		4:0	
Ctrl	Resp Dest Id Entry1	WRB COH	WRB L2ALLOC	DF PTR L2ALLOC	Rsvd	Data Length	Rsvd	Address (35 MSB) of Packet Descriptor data structure		Software Scratch1									

Table 14-7. Data Descriptor Vector

Bits	Name	Description
63:61	Ctrl	Control: 0x5: End of Packet (EOP) All other values are Reserved.
60:54	RespDestIdEntry1	Destination Bucket to which the Security Free Out Descriptor FMN message will be sent. See Section 14.4, “Security Free Out Descriptor Message Format” for the format of this message. This field is copied to the DestinationID field of both the Control Return Descriptor and the Data Return Descriptor components of the Security Free Out Descriptor.
53	WRB_COH	Reserved
52	WRB_L2ALLOC	L2 Allocation after write operation: 0: Data structure that the SAE writes back, (i.e. cipher destination, authentication destination and checksum destination) is guaranteed <i>not</i> to be in L2 cache 1: Data structure that the SAE writes back, (i.e. cipher destination, authentication destination and checksum destination) might be in L2 cache
51	DF_PTR_L2ALLOC	After read operation: 0: Structure pointed to by data address is guaranteed <i>not</i> to be in L2 cache 1: Structure pointed to by data address might be in L2 cache Setting the DF_PTR_L2ALLOC bit will force the data fetch pointer to go through the L2 cache on its way from DRAM to the security block, although it is impossible to know how long it will remain in L2 before it is invalidated. If this bit is not set though, it is guaranteed that the data will not go through L2.
50:46	Reserved	Reserved (value = 0)
45	Data Length	The packet descriptor data structure size is fixed at one cacheline (32 bytes). This effectively makes Data Length a Load (1)/NoLoad (0) bit. NoLoad causes an abort.
44:40	Reserved	Reserved (value = 0)
39:5	Data Address	35 msb of address of Packet Descriptor. Addresses are assumed to be cacheline-aligned, i.e., Address[4:0] is ignored.
4:0	Software_Scratch1	Ignored by hardware and copied as-is to the Free Out Descriptor. Software can use these bits as an ID tracking number, etc.

14.3.1 Control Descriptor Data Structure

This Control Descriptor Data Structure is pointed to by the control address of the 1st entry for the message. It consists of 2 parts:

- **Cipher Instruction** — specifies the operation to perform
- **Cipher Hash Information Formats** — contains the keys required by the operation and the Arc4cipher state

Note: Cipher Instruction and Cipher Hash Information data structures must be cacheline aligned.

14.3.1.1 Cipher Instruction

The Cipher Instruction tells the SAE what kind of operation to perform on the message, for example, MD5 and 3DES. The Cipher Hash Information format required by the specified operation is determined by the Cipher, HMAC, and Hash fields.

Table 14-8. Cipher Instruction

63:44		43	42	41	40	39:35		34:32
Reserved	Override Cipher	Arc4 Wait4 Save	Save Arc4 State	Load Arc4 State	Arc4KeyLen	Cipher		
31:29	28	27:25	24	23	22:21	20	19:17	16
Mode	InCp_Key	Reserved	Hash Group	HMAC	Hash	InHs_Key	Reserved	CkSum

Table 14-9. Cipher Instruction Fields

Bits	Name	Description
63:44	Reserved	Reserved = 0
43	OverrideCipher	Relevant to GCM mode only. 1: Engine will calculate E(K,0) then write data in the clear. Engine behavior is undefined if this bit is set and CipherPrefix is not set. (See Cipher Destination (dstDataSettings)[63]: CipherPrefix)
42	Arc4Wait4Save	Relevant to Arc4 mode only. 1: If Op is Arc4 and it requires state saving, then setting this bit will cause the current Op to delay subsequent uploading until saved state data becomes visible.
41	SaveArc4State	Relevant to Arc4 mode only. 1: Save Arc4 state at the end of the Arc4 operation. This operation is needed for stateful ARC4 operation.
40	LoadArc4State	Relevant to Arc4 mode only. 1: Load Arc4 state at the beginning of Arc4 operation. This is overwritten by the bit[28]InCp_Key setting for Arc4 cipher.
39:35	ARC4KeyLen	Relevant to Arc4 mode only. Length in bytes of the ARC4 key (0 is interpreted as 32). This field is ignored for other ciphers. Note: For ARC4, IFetch / IDecode always reads exactly four consecutive 64-bit words into its CipherKey{0,3} regardless of this quantity. It will, however, only use the number of bytes specified by this quantity.

Table 14-9. Cipher Instruction Fields (continued)

Bits	Name	Description
34:32	Cipher	Cipher 0: Bypass 1: DES 2: 3DES 3: AES 128-bit key 4: AES 192-bit key 5: AES 256-bit key 6: ARC4 7: Kasumi F8
31:29	Mode	Mode 0: ECB 1: CBC 2: CFB (AES only, otherwise undefined) 3: OFB (AES only, otherwise undefined) 4: CTR (AES only, otherwise undefined) 5: F8 (AES only, otherwise undefined) 6: Undefined 7: Undefined
28	InCp_Key	0: Preserve old Cipher Keys 1: Load new Cipher Keys from memory to local registers. This overrides LoadArc4State (bit [40]).
27:25	Reserved	Reserved (value = 0)
24	Hashgroup	Controls usage of contents of field 22:21. See Hash below.
23	HMAC	HMAC 0: Hash without HMAC 1: Hash with HMAC Behavior is undefined if GCM or Kasumi F9 authentication is chosen.
22:21	Hash	Usage of this field depends on value of bit 24, Hashgroup. If [24]Hashgroup = '0' then this field's values are: 00: Hash NOP 01: MD5 10: SHA-1 11: SHA-256 If Hashgroup = '1' then this field has the following meaning: 00: SHA-384 01: SHA-512 10: GCM 11: Kasumi F9
20	InHs_Key	0: Preserve old HMAC Keys 1: Load new HMAC Keys from memory to local registers; if GCM is selected as authentication, both H and SCI will be loaded from memory to the decoder. H will be loaded directly to the engine; SCI is only loaded if CFBMask[1:0] = 3. If Kasumi F9 is selected, new keys will be loaded from memory to decoder.
19:17	Reserved	Reserved (value = 0)
16	CkSum	Checksum 0: CkSum NOP 1: Internet_checksum
15:0	Reserved	Reserved (value = 0)

14.3.1.2 Cipher Hash Information Formats

The section follows the cipher instruction contains the information about keys required by the operation, and initial vectors for the HMAC. Due to different ciphers keys lengths, this section has several possible combinations in terms of key fields and HMAC. The generic format of this block (shown in [Figure 14-6](#)) has the keys on the top part and the HMAC keys on the bottom. If no HMAC functionality is needed, the lower part is ignored and isn't loaded by the SAE.

Figure 14-6. Cipher and HMAC Keys

Cipher Key[0]
Cipher Key[1]
...
Cipher Key[n]
HMAC Key[0]
HMAC Key[1]
...
...
HMAC Key[7]

DES/AES/KASUMI Formats

The following tables show the different combinations of Cipher Key lengths and HMAC Key lengths for DES/3DES, AES and KASUMI cipher blocks:

AES256, (ECB, CBC, OFB, CTR, CFB), HMAC (MD5, SHA-1, SHA-256) — 96 bytes in 12 64-bit words

1	2	3	4	5	6	7	8	9	10	11	12
Cipher Key0	Cipher Key1	Cipher Key2	Cipher Key3	HMAC Key0	HMAC Key1	HMAC Key2	HMAC Key3	HMAC Key4	HMAC Key5	HMAC Key6	HMAC Key7

AES256 (ECB, CBC, OFB, CTR, CFB), Non-HMAC (MD5, SHA-1, SHA-256) — 32 bytes in four 64-bit words

1	2	3	4
CipherKey0	CipherKey1	CipherKey2	CipherKey3

AES192, (ECB, CBC, OFB, CTR, CFB), HMAC (MD5, SHA-1, SHA-256) — 88 bytes in 11 64-bit words

1	2	3	5	6	7	8	9	10	11	12
Cipher Key0	Cipher Key1	Cipher Key2	HMAC Key0	HMAC Key1	HMAC Key2	HMAC Key3	HMAC Key4	HMAC Key5	HMAC Key6	HMAC Key7

AES192, (ECB, CBC, OFB, CTR, CFB), Non-HMAC (MD5, SHA-1, SHA-256) — 24 bytes in three 64-bit words

1	2	3
CipherKey0	CipherKey1	CipherKey2

AES128, (ECB, CBC, OFB, CTR, CFB), HMAC (MD5, SHA-1, SHA-256) — 80 bytes in 10 64-bit words

1	2	3	4	5	6	7	8	9	10
Cipher Key0	Cipher Key1	HMAC Key0	HMAC Key1	HMAC Key2	HMAC Key3	HMAC Key4	HMAC Key5	HMAC Key6	HMAC Key7

AES128, (ECB, CBC, OFB, CTR, CFB), Non-HMAC (MD5, SHA-1, SHA-256) — 16 bytes in two 64-bit words

1	2
CipherKey0	CipherKey1

DES, (ECB, CBC), HMAC (MD5, SHA-1, SHA-256) — 72 bytes in nine 64-bit words

1	2	3	4	5	6	7	8	9
Cipher Key0	HMAC Key0	HMAC Key1	HMAC Key2	HMAC Key3	HMAC Key4	HMAC Key5	HMAC Key6	HMAC Key7

DES, (ECB, CBC), Non-HMAC (MD5, SHA-1, SHA-256) — 8 bytes

1
CipherKey0

3DES, (ECB, CBC), HMAC (MD5, SHA-1, SHA-256) — 88 bytes in 11 64-bit words

1	2	3	4	5	6	7	8	9	10	11	
Cipher Key0	Cipher Key1	Cipher Key2	Cipher Key0	HMAC Key0	HMAC Key1	HMAC Key2	HMAC Key3	HMAC Key4	HMAC Key5	HMAC Key6	HMAC Key7

3DES, (ECB, CBC), Non-HMAC (MD5, SHA-1, SHA-256) — 24 bytes

1	2	3
Cipher Key0	Cipher Key1	Cipher Key2

HMAC (MD5, SHA-1, SHA-256) — 64 bytes

1	2	3	4	5	6	7	8
HMAC Key0	HMAC Key1	HMAC Key2	HMAC Key3	HMAC Key4	HMAC Key5	HMAC Key6	HMAC Key7

ARC4 Cipher Formats

When using ARC4 cipher, the “Cipher Key and Hash Key” block has an additional functionality related to the ARC4 state. Since SAE ARC4 supports stateful operation, part of the “Cipher Key and Hash Key” block are used to preserve the state of the operation. The structure of block maintain the structure mentioned above, with the additional space reserved for the state as shown in [Figure 14-7](#) and the tables following it.

Figure 14-7. Cipher and HMAC Keys

Cipher Key[0]	Cipher Key[0]
Cipher Key[1]	Cipher Key[1]
Cipher Key[2]	Cipher Key[2]
Cipher Key[3]	Cipher Key[3]
PAD[0]	HMAC Key[0]
PAD[1]	HMAC Key[1]
PAD[2]	...
ARC4Sbox_data[0]	...
ARC4Sbox_data[1]	HMAC Key[7]
...	PAD[0]
...	PAD[1]
ARC4Sbox_data[31]	PAD[2]
ARC4IJ state	ARC4Sbox_data[0]
PAD[0]	ARC4Sbox_data[1]
PAD[1]	...
PAD[2]	...
	ARC4Sbox_data[31]
	ARC4IJ state
	PAD[0]
	PAD[1]
	PAD[2]

ARC4, Non-HMAC (MD5, SHA-1, SHA-256) — 32 bytes

1	2	3	4
Cipher Key0	Cipher Key1	Cipher Key2	Cipher key3

ARC4, HMAC only key (MD5, SHA-1, SHA-256) — 96 bytes in 12 64-bit words

1	2	3	4	5	6	7	8	9	10	11	12
Cipher Key0	Cipher Key1	Cipher Key2	Cipher Key3	HMAC Key0	HMAC Key1	HMAC Key2	HMAC Key3	HMAC Key4	HMAC Key5	HMAC Key6	HMAC Key7

ARC4 if state loaded or saved, Non-HMAC (MD5, SHA-1, SHA-256) — 344 bytes in 43 64-bit words

1	2	3	4	5	6	7
Cipher Key0	Cipher Key1	Cipher Key2	Cipher key3	Pad0	Pad1	Pad2
Arc4Sbox Data0
40	Arc4IJ Data	41	42	43		
N/A	N/A	.	.	N/A	J	I

byte
0 1 2 3 4 5 6 7

Note: N/A = ignored.

Note: User provides shaded words; Security Accelerator provides unshaded words, which are to be persisted (unchanged) between fragments.

ARC4 if state loaded or saved, HMAC (MD5, SHA-1, SHA-256) — 408 bytes in 51 64-bit words

1	2	3	4												
Cipher Key0	Cipher Key1	Cipher Key2	Cipher Key3												
5	6	...	12	13	14	15									
HMAC Key0	HMAC Key1	...	HMAC Key7	Pad0	Pad1	Pad2									
16	47								
Arc4Sbox Data0	Arc4Sbox Data31								
48	49	50	51												
N/A	N/A	J	I	Pad3	Pad4	Pad5					
byte	0	1	2	3	4	5	6	7							

Note: N/A = ignored

Note: User provides shaded words; Security Accelerator provides unshaded.

14.3.2 Data Descriptor Data Structure

The Data Descriptor data structure occupies four 64-bit words in memory in the order shown in [Table 14-10](#). Contents of each 64-bit word are given in [Table 14-11](#), [Table 14-12](#), [Table 14-13](#), and [Table 14-14](#).

The Security Acceleration Engine (SAE) accepts payload data byte-aligned. No 64-bit alignment need be performed by software for caching payload data.

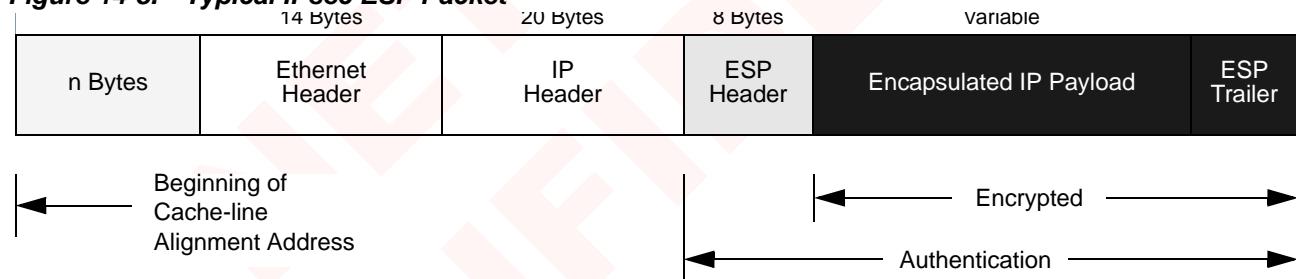
Table 14-10. Data Descriptor Structure

Table	Address	63	Data Descriptor	0
14-11	Source Address	0x00	srcLengthIVOffUseIVNext	
14-12	Cipher Destination Address	0x08	dstDataSettings	
14-13	Authentication Destination Address	0x10	authDstNonceLow	
14-14	Check-Sum Destination Address	0x18	ckSumDstNonceHiCFBMaskLLWMask	

Alignment Concept

The SAE is designed as a 64-bit machine requiring several alignments to point to the actual data from the given source address. This allows flexibility when processing typical VPN/IPsec frames with minimized data shuffling. The address given to the SAE must follow cache line alignment, 32 bytes, expressed as 35 bits in the Data Descriptor section. If the data to be processed isn't aligned with the cache line, a "Global Offset" should be used to point into the data. The Global Offset describes the distance, in byte quantities, from the given address to the actual data. Using a typical IPsec ESP packet, see example in [Figure 14-8](#).

Figure 14-8. Typical IPsec ESP Packet



As shown in [Figure 14-8](#), the packet may (or may not) start at the cache line alignment, thus leaving n bytes before the packet starts. However, the SAE will expect the aligned address as a reference. The SAE offers several offsets pointing to the proper location of the specific section, such as cipher offset, HASH offset, IV offset and checksum offset. Each of these offsets uses 64-bit count (1 DW) from a given Global Offset (the same global offset is used for all three). Using the example above, we can calculate the required offsets:

- Global Offset = $(n + 14 + 20) \bmod 8$
- HASH Offset = $(n + 14 + 20) \bmod 8$
- Cipher Offset = $(n + 14 + 20 + 8) \bmod 8$

If an Initialization Vector is used, then:

- IV Offset = $(n + 14 + 20 + 8) \bmod 8$
- Cipher Offset = $(n + 14 + 20 + 8 + 8) \bmod 8$ (for 8-byte ciphers such as DES and 3DES)
Cipher Offset = $(n + 14 + 20 + 8 + 16) \bmod 8$ (for 16-byte ciphers such as AES)

Table 14-11. Segment Source Address Data Structure (*srcLengthIVOffUseIVNext*)

63	62	61:59	58	57	56:54	53:43	42	41	40	39:5	4	3	2:0
Load HMAC key	Pad Hash	Hash Byte Count	Next	UseIV	IV Offset	Packet Length	NLHMAC	Break	Wait	Segment Source Address	SRTCP	Hash_Offset_Hi	Global Src Data Offset

Bits	Name	Description
63	LoadHMACkey	Load HMAC Key: 0: Preserve old HMAC key stored in Authentication engine (this is not useful if HASH.HMAC equals 0) 1: Load HMAC key from ID registers at beginning of operation. If GCM is selected as authenticator, 'H' from the decoder will be loaded to the engine. If Kasumi F9 is selected, the key will be loaded from the decoder to the engine.
62	PadHash	Pad Hash: 0: HASH will assume the data was padded to be a multiple of 512 bits in length and that the last 64-bit word expresses the total data length in bits seen by Hash engine. 1: If the data was not padded to be a multiple of 512 bits in length; the Hash engine will do its own padding to generate the correct digest. If GCM is selected, the authenticator does its own padding (i.e., PadHash is ignored).
61:59	HashByteCount	Number of bytes on last 64-bit data word to use in digest calculation. Note: this is relevant only if Pad Hash (bit [62]) is set. Note: This is applied to data after alignment for global_src_data_offset has been done, so that the total amount of data to be authenticated does not have to be a multiple of 8 bytes. 0x0: Use all 8 0x1: Use first byte (MSB) only; clear other bytes to zero (i.e., 0xDDxxxxxxxxxxxx) 0x2: Use first two bytes only; clear other bytes to zero (i.e., 0xDDDDxxxxxxxxxx) 0x3: ... 0x7: Use seven most-signif. bytes; clear 8th to zero (i.e., 0xDDDDDDDDDDDDDDxx)
58	Next	The Next bit allows for fragmentation/defragmentation and processing of large (>16kB) packets. The sequence of adjacent Data Descriptors acts as a contiguous linked list of pointers to the actual packets. Clear this bit to zero on the last PacketDescriptor to end the processing. 0: Finish (return Descriptor) at end of operation. 1: Access the next Data Descriptor (i.e. next cache-line) when the current is complete.
57	Use IV	0: On first fragment: Use old IV On subsequent fragments: Do <i>not</i> write out to DST the (64-bit word) offset 1: On first fragment: Use data @ Segment_address + IV_Offset as IV. On subsequent fragments: Write out the (64-bit) offset data to DST.
56:54	IV Offset	<ul style="list-style-type: none"> On first fragment: Offset in number of 64-bit words from beginning of packet (i.e. potentially byte-shifted Segment address) to cipher IV On subsequent fragments: Offset to beginning of data to process. Data to offset won't be given to engines and will be written out to destination in the clear. <p>IMPORTANT: on subsequent fragments, IV_Offset may not exceed 3; larger values will cause an error. See error conditions in Table 14-16 and Table 14-17.</p>

Bits	Name	Description
53:43	Packet Length	<p>Packet Length: If a read operation, this field is defined as the number of double words to read by SAE. This is the total amount of data read by the security engine, as given by the equation:</p> $\text{PktLen} = (\text{Address_of_last_Byte_of_Data} - (\text{segment_source_address} + \text{src_global_offset})) / 8;$ <p>where segment_source_address is cacheline-aligned and src_global_offset is the source offset in bytes.</p> <p>If a write operation, this field is defined as the total number of DWords written. (Multiply this field by eight for the number of bytes written.)</p> <p>See Table 14-16 and Table 14-17 for corner cases and error conditions associated with this field.</p>
42	NLHMAC	1: Setting this bit will prevent the transmission of the last DWORD in the payload to the authenticator. Relevant typically when data to the authenticator is byte-shifted.
41	Break	1: Break a wait state. Causes the operation to be flushed and the free Descriptor to be returned. Activated if DFetch is blocked by a Wait and the Wait is still active.
40	Wait	1: Setting this bit causes the operation to block in the DFetch stage. DFetch will keep polling the memory location until the bit is reset, at which time the pipe resumes normal operation. This feature is convenient for software dealing with fragmented packets.
39:5	Segment Src Address	35 MSBs of pointer to (cacheline-aligned) source data.
4	SRTCP	1: Bypass the cipher for the last 4 bytes of data. The last 4 bytes will be sent to memory and the authenticator in the clear. This is applicable to last fragment only.
3	Hash_Offset_Hi	The most-significant-bit of the Hash DWORD offset count. It is prepended to dstDataSettings.Hash_Offset_Lo (see Table 14-11) to form Hash_Offset (3 bits total)
2:0	Global Src Data Offset	<p>Number of bytes to right-shift data before presenting it to engine. Allows realignment of byte-aligned, non-double-word-aligned data.</p> $[(\text{CipherDstAddress} - \text{DstDwordoff}) * 8] - \text{GlobalDestDataOffset}$

Table 14-12. Cipher Destination (dstDataSettings)

63	62:60	59	58:56	55:54	53	52:41	40	39:5	4:3	2:0
Cipher Prefix	Arc4 Byte Count	E/D	Cipher_Offset	Hash_Offset_Lo	Hash_Src	CkSum_Offset	CkSum_SRC	CipherDestAddress	Dst Dword Off	Global Dest Data Offset

Bits	Name	Description										
63	CipherPrefix	1: Two double words set to zero will be sent to the selected cipher after the IV is loaded and before the actual data goes in the result of that encryption - also known as E(K,0) - will be stored locally and X-ORed with the authentication digest to create the final digest at the end of the authentication operation.										
62:60	Arc4ByteCount	<p><i>When Arc4 is used, this field's function is:</i></p> <p>Number of bytes on last 8 byte word to encrypt, starting from most significant.</p> <p>000: Encrypt all 8 bytes. (Must be zero for non-RC4 ciphers)</p> <p>001: Encrypt first byte only:</p> <table border="1"> <tr> <td>0x</td> <td>DD</td> <td>.</td> <td>.</td> <td>.</td> <td>.</td> <td>.</td> <td>.</td> <td>.</td> <td>.</td> </tr> </table>	0x	DD
0x	DD			
		<p>010: Encrypt first 2 bytes:</p> <table border="1"> <tr> <td>0x</td> <td>DD</td> <td>DD</td> <td>.</td> <td>.</td> <td>.</td> <td>.</td> <td>.</td> <td>.</td> <td>.</td> </tr> </table>	0x	DD	DD
0x	DD	DD			
		<p>...</p> <p>111: Encrypt first 7 bytes:</p> <table border="1"> <tr> <td>0x</td> <td>DD</td> <td>DD</td> <td>DD</td> <td>DD</td> <td>DD</td> <td>DD</td> <td>DD</td> <td>.</td> </tr> </table>	0x	DD	.							
0x	DD	DD	DD	DD	DD	DD	DD	.				
		<p>In reality, the engine keeps encrypting and all bytes are encrypted. However, the Sbox and I, J values are held (i.e., not changed) at the value they had at the last byte encrypted.</p> <p>If the cipher is not Arc4, Arc4ByteCount should be written as 0.</p> <p><i>When Arc4 is not used, this field contains IV byte offset.</i></p>										
59	E/D	<p>0: Decrypt</p> <p>1: Encrypt</p>										
58:56	Cipher_Offset	Number of 64-bit words between the first data segment and 64-bit word on which to start Cipher operation.										
55:54	Hash_Offset_Lo	The least significant bits of Hash_Offset. Hash_Offset is the number of 64-bit words between the first data segment and 64-bit word on which to start hashing. Hash_Offset is obtained by concatenating Hash_Offset_Hi (see Table 14-10) and Hash_Offset_Lo.										
53	Cipher_Offset_Hi	<p>0: DMA Channel</p> <p>1: Cipher if word count exceeded Cipher_offset, DMA channel otherwise</p>										
52:41	CkSum_Offset	Number of 32-bit words between the first data segment and the 32-bit word on which to start checksum calculation. If multiple fragments are specified, the value for check sum offset is taken from the first fragment only.										
40	CkSum_SRC	<p>0: DMA Channel</p> <p>1: Cipher if word count exceeded Cipher_Offset. DMA channel otherwise.</p>										

Bits	Name	Description
39:5	CipherDestAddress	Cipher destination address 35 msbs of pointer to destination (i.e., cacheline-aligned)
4:3	DstDwordOff	Number of 64-bit words to left-shift data from specified Cipher destination address before writing it to memory.
2:0	GlobalDestDataOffset	Global destination data offset: Number of bytes to left-shift (double-word boundary aligned) data by before writing it to memory.

Table 14-13. Authentication Destination Address (authDstNonceLow)

63:40	39:5	4:0
Nonce_Low	Authentication Destination Address	Cipher_Offset_Hi

Bits	Name	Description
63:40	Nonce_Low	Nonce[23:0] 24 Least-significant-bits of 32-bit long nonce. Used by AES in counter mode
39:5	Authentication Destination Address	35 msbs of pointer to (cacheline-aligned) destination location.
4:0	Cipher_Offset_Hi	5 msbs of 8-bit Cipher Offset. Will be concatenated to the top of dstDataSettings[58:56]. It is ignored for fragments other than first.

Table 14-14. Checksum Destination Address (ckSumDstNonceHiCFBMaskLLWMask)

63:61	60:58	57:56	55:48	47:40	39:5	4:0
Hash_Byt_Offset	Packet_length_bytes	LLW_Mask	CFB_Mask	Nonce_Hi	ChecksumDestAddress	IV_Offset_Hi

Bits	Name	Description
63:61	Hash_Byt_Offset	Additional offset in bytes to be added to Hash_Offset to obtain the full offset applied to the data before submitting it to authenticator
60:58	Packet_length_bytes	Use only for multiple fragments. Sum of packet length in bytes for fragments. If mod8 (sum) is nonzero then an error flag is sent in the return descriptor. On one fragment payloads: Ignored (i.e. assumed to be 0, last dword used in its entirety) On fragments before last: Number of bytes on last fragment dword On last fragment: Ignored (i.e. assumed to be 0, last dword used in its entirety)
57:56	LLWMask	Last Long Word Mask — applicable in AES CTR only 00: No masking performed on Input data 11: Mask (zero-out) 32 least significant bits 10: Mask 64 LSBs 01: Mask 96 LSBs

Bits	Name	Description
55:48	CFB_Mask[1:0]	<p>CFB Mask: This field's function is:</p> <p><i>If CFB mode, then</i></p> <ul style="list-style-type: none"> - 8-bit mask used by AES in CFB mode. - In CTR mode, bit 0 enables CCM; remaining bits should be 0. - In CTR mode, setting CFB_Mask[0] to 1 allows the 32 least-significant-bits of the counter to be initialized to the 32 least-significant-bits of the second IV: IV1[31:0] <p><i>If CTR mode is set, then</i></p> <p>See full width text in footnote below ^a</p>
47:40	Nonce_Hi	<p>Nonce[31:24] 8 most-significant bits of 32-bit nonce value. Used by AES in counter mode.</p> <p>The full counter will be formed as follows:</p> <ul style="list-style-type: none"> - if CFBMask[0] = 0: Counter = Nonce_Hi[47:40] Nonce_Low[63:40] IV0[63:0] 32'h1 - if CFBMask[0] = 1: Counter = Nonce_Hi[47:40] Nonce_Low[63:40] IV0[63:0] IV1[31:0]
39:5	ChecksumDestAddress	35 MSB of pointer to (cacheline-aligned) destination location
4:0	IV_Offset_Hi	<p>This is the extension of IV_Offset. See Table 14-11.</p> <p>5 msbs of 8-bit IVOffset will be concatenated to the top of IVOffset, which is srcLengthIVOffsetUseIVNext[56:54].</p> <p>It is ignored on fragments other than first.</p>

a. In CTR mode

```

CFB_Mask[1:0] =
2'b00->Counter[127:0]=Nonce[31:0]|| IV0[63:0]||4'h00000000 (only 1 IV expected)
regular CTR de
2'b01->Counter[127:0]=Nonce[31:0]|| IV0[63:0] || IV1[31:0] (2 IV expected) CCMP
mode
2'b10->Counter[127:0]=IV1[63:0] || IV0[31:0] || Nonce[31:0] (2 IV expected) GCM
with SCI in SA
2'b11->Counter[127:0]=IDecode.SCI[63:0] || IV0[31:0] || Nonce[31:0] (1 IV expected)
GCM w/o SCI in SA

```

14.4 Security Free Out Descriptor Message Format

Upon completion of an operation, the SAE returns a 2-Entry Free Out Descriptor in the following format.

Table 14-15. Security Free Out Descriptor

Control Return Descriptor						
63:61	60:54	53:52	51:49	48:40	39:5	4:0
Ctrl	DestinationID	Reserved	Desc Ctrl	Control Error	Source Address (35 MSB)	Software_Scratch0
Data Return Descriptor						
63:61	60:54	53:52	51:49	48:40	39:5	4:0
Ctrl	Destination ID	Reserved	Desc Ctrl	Data Error	Dest Address (35 MSB)	Software_Scratch1

Bits	Name	Description
Control Descriptor Return		
63:61	Ctrl	Field will always be marked: 2: Free
60:54	DestinationID	Destination FMN bucket ID set by sending FMN agent
53:52	Reserved	Reserved = 0
51:49	DescCtrl	Control field of first entry of received (ORIG) descriptor (expected to be SOP)
48:40	ControlError	See Table 14-16 .
39:5	ControlAddress	Address of Control Descriptor, as sent in the original message
4:0	SoftwareScratch0	Copied from original descriptor. Not used by the engine; the field is returned exactly as it was received. Software may use it, for example, to identify the security engine that the free descriptor returns from, or to identify the descriptor/data structures themselves.
Data Descriptor Return		
63:61	Ctrl	Field will always be marked: 2: Free
60:54	DestinationID	Destination FMN bucket ID set by sending FMN agent
53:52	Reserved	Reserved = 0
51:49	DescCtrl	Control field of second entry of received (ORIG) descriptor (expected to be EOP)
48:40	DataError	See Table 14-17 .
39:5	DestAddress	Address of Data Descriptor, as sent in the original messages
4:0	SoftwareScratch1	Copied from original descriptor. Not used by the engine; the field is returned exactly as it was received. Software may use it, for example, to identify the security engine that the free descriptor returns from, or to identify the descriptor/data structures themselves.

The Instruction and Data Error codes are enumerated in the following sections.

The Control and Data Error codes are defined in [Table 14-16](#) and [Table 14-17](#).

Table 14-16. Control Error Conditions

Code	Condition	Description
0x000	No Error	No Error
0x001	<i>Reserved</i>	
0x002	Unknown or Illegal Mode	((Mode == {2,3,4} & {DES 3DES}) (Mode == {5,6,7}) (Mode != 0 & (ARC4 Kasumi F8 Bypass)))
0x004	<i>Reserved</i>	
0x008	<i>Reserved</i>	
0x010	<i>Reserved</i>	
0x020	<i>Reserved</i>	
0x040	<i>Reserved</i>	
0x080	Data Read Error	
0x100	Descriptor Control Field Error	D0.Ctrl != SOP D1.Ctrl != EOP

Table 14-17. Data Error Conditions

Code	Condition	Description
0x000	No Error	No Error
0x001	Insufficient Data To Cipher	$ \begin{aligned} & ((\text{Packet_Length} \leq (\text{Cipher_Offset} \text{ or } \text{IV_Offset})) \\ & (\text{Packet_Length} == 0 \text{ & FirstFrag \& Next}) \\ & (\text{Packet_Length} == 0 \text{ & Packet_Byte_Length} == 0 \text{ & LastFrag}) \\ & (\text{Total_Packet_Byte_Length} \text{ & LastFrag}) \\ & (\text{Packet_Byte_Length} != 0 \text{ & FirstFrag \& LastFrag}) \\ & (\text{Packet length} == 11'h7ff \text{ & (Global src data offset} != 0 \\ & \quad \text{Global dst data offset} != 0 \text{)}) \\ & (\text{Packet_Length} == \text{IV_Offset} \text{ & (} \\ & \quad \text{Packet_Byte_Length} == 0 \text{ FirstFrag LastFrag} \\ & \quad (\text{Accumulated byte-count crosses 8-byte boundary}) \\ & \quad \text{UseIV} \text{)}) \\ \end{aligned} $ <p>Masked by: Memory error #Tot Odd Dword Count to AES on last frag #Tot Odd Dword Count to AES on previous frag and UseIV #Subseq frag and IVOffset > 3 Break on Wait</p>
0x002	Illegal IV Location	$ \begin{aligned} & ((\text{Cipher_Offset} < \text{IV_Offset}) \\ & (\text{Cipher_Offset} \leq \text{IV_Offset} \text{ & ((AES \& (~CTR} \\ & \quad (\text{CFBMask}[0] \wedge \text{CFBMask}[1])) \\ & \quad \text{KasumiF9})) \\ & (\text{IV_Offset} > 3 \text{ & PrevNext}) \\ & (\text{IV_Offset} == 0xff \text{ & IV_Byte_Offset} > 0)) \\ \end{aligned} $
0x004	Illegal Wordcount To AES	$ \begin{aligned} & (\text{AES} \text{ & ((Packet_Length}[3] != \text{Cipher_Offset}[0]) \text{ & !PrevNext \& !Next}) \\ & (\text{(TotDWordCount} == 2k + 1) \text{ & PrevNext \& !Next}) \\ & (\text{(TotDWordCount} == 2k + 1) \text{ & PrevNext \& IVOff} > 0 \text{ & UseIV})) \\ \end{aligned} $
0x008	Illegal Pad and ByteCount Spec	$ \begin{aligned} & (\text{Hash_Byte_Count} != 0 \text{ & !Pad_Hash \& FirstFrag \& Auth_Op}) \\ & (\text{!Pad_Hash \& FirstFrag \& GCM}) \\ \end{aligned} $
0x010	Insufficient Data To Checksum	$ \{(\text{Packet_Length}, 1'b0} \leq \text{CkSum_Offset} \text{ & FirstFrag \& CkSum_Op}) $

Table 14-17. Data Error Conditions (continued)

Code	Condition	Description
0x020	Forbidden CFB Mask	(AES & CFBMode & InCp_Key & CFBMask[7] & (CFBMask[6:0]) & FirstFrag)
0x040	Insufficient Data To Authenticate	({Packet_Length} <= Auth_Offset & FirstFrag & Auth_Op)
0x080	Data Read Error	
0x100	<i>Reserved</i>	

14.5 RSA Core

14.5.1 RSA / ECC Control Descriptor Message Format

Table 14-18 shows the RSA / ECC (Elliptic Curve Cryptography) Descriptor, the 2-entry message that is sent by the originating Thread to the RSA core through the Fast Messaging Network. The contents of the message supply the RSA core with an instruction and pointers to all necessary data to use.

Table 14-18. RSA / ECC Control Descriptor

Control Descriptor Vector

63:61		60:54		53	52:40		39:5		4:3	2:0
Ctrl	Op Class	Valid Op		53	OpCtrl0		Source Address (35 MSB)	Software Scratch0		Global src data offset

Data Descriptor Vector

63:61		60:54		53	52	51	50:40	39:5	4:3	2:0
Ctrl	Dest ID	WRB_COH	WRB_L2ALLOC	DF_L2ALLOC		OpCtrl1	Destination Address (35 MSB)	Software Scratch1		Global dst data offset

Table 14-19. RSA / ECC Control Descriptor Vector

Bits	Name	Description
63:61	Ctrl	Start of Packet (SOP)
60:54	OpClass	Operation Class: This field identifies the class of operation (RSA or ECC): 0x00: RSA (Modular exponentiation) 0x01: ECC (including prime modular ops and binary GF ops) Other bit values: Undefined
53	ValidOp	0: No operation performed; descriptors are sent back immediately. 1: Will cause operation to start; descriptors are sent back at end of operation.
52:40	OpCtrl0	Operation Control 0: This field's definition depends on the operation type selected in the OpClass field (bits [60:54]). If the OpClass indicates an RSA operation, bits [52:40] are defined as follows: bit [52]: <u>Block Width</u> 0 = 512-bit blocks 1 = 1024-bit blocks bit [51]: <u>Load Constant</u> 0 = Preserve old constant; assumes Source Addr pointer (RSAData_pt) points to Exponent or that the length of constant is zero. 1 = Load constant from data structure bits [50:40]: <u>Exponent Width</u> (expressed in number of bits). This corresponds to the highest 1-bit of the exponent plus one. For example, if the exponent were 278928823 (0x10A01DB7), the highest 1-bit would be bit 28, and thus Exponent Width would be 29.

Table 14-19. RSA / ECC Control Descriptor Vector

Bits	Name	Description
52:40 (Cont.)	OpCtrl0 (Cont.)	<p>If the OpClass indicates an ECC (Elliptical Curve Cryptography) operation, bits [52:40] are defined as follows:</p> <p>bits [52:46]: <u>Type</u></p> <ul style="list-style-type: none"> 0x00 = ECC prime 160 0x01 = ECC prime 192 0x02 = ECC prime 224 0x03 = ECC prime 256 0x04 = ECC prime 384 0x05 = ECC prime 512 0x06 through 0x1F = Undefined 0x20 = ECC binary 163 0x21 = ECC binary 191 0x22 = ECC binary 233 0x23 through 0x6F = Undefined 0x70 = ECC UC load 0x71 through 0x7F = Undefined <p>bits [45:40]: <u>Function</u></p> <p>If bits [52:46] selected an ECC Prime type (of any precision), these Function bits are interpreted according to the following table:</p> <ul style="list-style-type: none"> 0x00 = ECC prime - Point Multiplication: $R = k.P$ 0x01 = ECC prime - Point Addition: $R = P + Q$ 0x02 = ECC prime - Point Double: $R = 2 \times P$ 0x03 = ECC prime - Point Verification: '1' if R is on curve; else '0' 0x04 = ECC prime - Modular Addition: $c = x + y \text{ mod } m$ 0x05 = ECC prime - Modular Subtraction: $c = x - y \text{ mod } m$ 0x06 = ECC prime - Modular Multiplication: $c = x * y \text{ mod } m$ 0x07 = ECC prime - Modular Division: $x / y \text{ mod } m$ 0x08 = ECC prime - Modular Inversion: $c = 1 / y \text{ mod } m$ 0x09 = ECC prime - Modular Reduction: $c = x \text{ mod } m$ 0x0A through 0x3F = Undefined <p>If bits [52:46] selected an ECC Binary type (of any precision), these Function bits are interpreted according to the following table:</p> <ul style="list-style-type: none"> 0x00 = ECC binary - Point Multiplication: $R = k.P$ 0x01 = ECC binary - Binary GF Inversion: $C(x) = 1 / A(x) \text{ mod } F(x)$ 0x02 = ECC binary - Binary GF Multiplication: $C(x) = B(x) * A(x) \text{ mod } F(x)$ 0x03 = ECC binary - Binary GF Addition: $C(x) = B(x) + A(x) \text{ mod } F(x)$ 0x04 through 0x3F = Undefined
39:5	SourceAddress	35 most significant bits of pointer to source address (i.e., cacheline-aligned)
4:3	Software_Scratch0	Ignored by hardware and copied as-is to the Free Out Descriptor.
2:0	GlobalSrcDataOffset	Number of bytes to right-shift data before presenting it to engine; allows realignment of byte-aligned, non-double-word aligned data.

Table 14-20. RSA / ECC Data Descriptor Vector

Bits	Name	Description
63:61	Ctrl	End of Packet (EOP)
60:54	DestID	Destination Bucket to which the return message will be sent.
53	WRB_COH	Reserved
52	WRB_L2ALLOC	After write operation: 0: Data structure that the RSA writes back, i.e. destination address, <i>is guaranteed not to be in L2 cache</i> 1: Data structure that the RSA writes back, i.e. destination address, <i>might be in L2 cache</i>
51	DF_L2ALLOC	After read operation: 0: Structure pointed to by data address <i>is guaranteed not to be in L2 cache</i> 1: Structure pointed to by data address <i>might be in L2 cache</i>
50:40	OpCtrl1	Operation Control 1: This field's definition depends on the operation type selected in the OpClass field (bits [60:54]). If the OpClass indicates an RSA operation, bits [50:40] are defined as follows: Modulus Width (expressed in number of bits). This corresponds to the highest 1-bit of the modulus plus one. For example, if the modulus were 278928823 (0x10A01DB7), the highest 1-bit would be bit 28, and thus Modulus Width would be 29. (If the OpClass indicates an ECC operation, bits [50:40] are not used.)
39:5	DestinationAddress	35 most significant bits of pointer to destination address (i.e., cacheline-aligned)
4:3	Software_Scratch1	Ignored by hardware and copied as-is to the Free Out Descriptor.
2:0	GlobalDstDataOffset	Number of bytes to left-shift (double-word boundary aligned) data before writing it to memory

14.5.2 RSA Data Formats

All RSA data formats may be shifted by 0 to 7 bytes from a cache line aligned address. The shift may be specified as GlobalSrcDataOffset (for reading), or GlobalDstDataOffset (for writing).

The data block at SourceAddress + GlobalSrcDataOffset specifies, in order, the ‘constant’, exponent, modulus, and message data. The length of the message data is specified by Block Width (bit 52) in the control descriptor vector.

This section shows the data formats; all diagrams in this section have a 64-bit data granularity. The quantity known as ‘constant’:

$$2^{(2 \times \text{BlockWidth}+4)} \bmod \text{Modulus}$$

is to be calculated in software once for each (Exponent, Modulus) combination.

The number of double words used to store the exponent and modulus is calculated from the bit widths, ExponentWidth (see [Table 14-19](#)) and ModulusWidth (see [Table 14-20](#)):

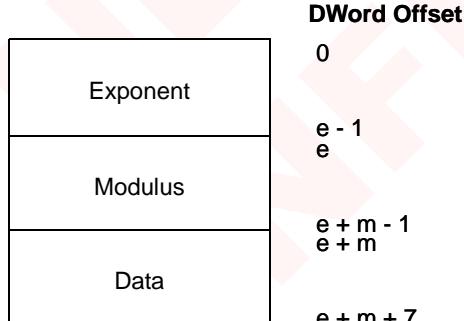
- The value **e** is the minimum number of double words required to express the exponent, and depends on ExponentWidth .
- The value **m** is the minimum number of double words required to express the modulus, and it depends on ModulusWidth .

These quantities may be calculated in the ‘c’ programming language as follows:

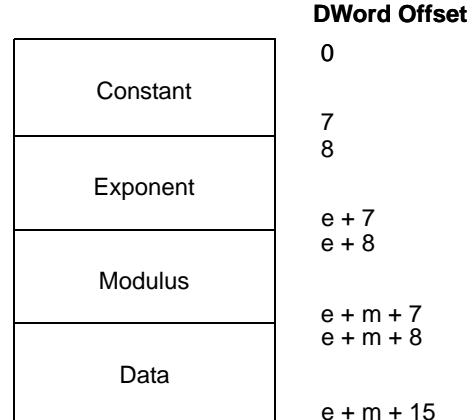
```
e = (ExponentWidth>>6)+( (ExponentWidth&0x3F)?1:0)
m = (ModulusWidth>>6)+( (ModulusWidth&0x3F)?1:0)
```

Note that either e or m may be 0, in which case the stored exponent or modulus is used. Similarly, if LoadConstant is 0, the previously stored constant is used.

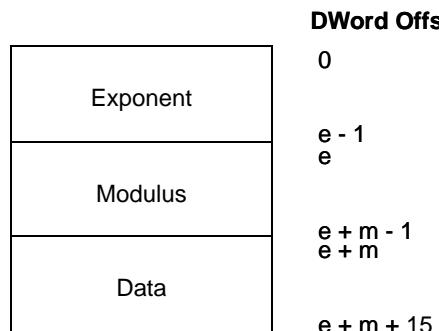
**Block Width = 0 (512 bits)
Load Constant = 0**



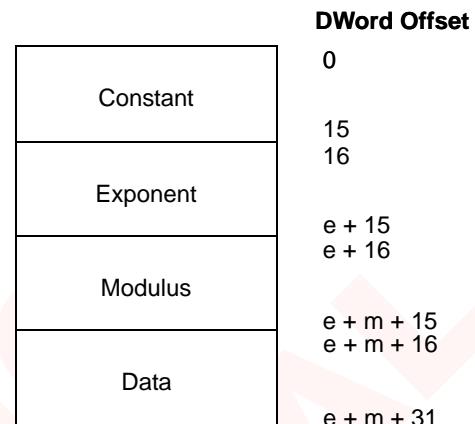
**Block Width = 0 (512 bits)
Load Constant = 1**



Block Width = 1 (1024 bits)
Load Constant = 0



Block Width = 1 (1024 bits)
Load Constant = 1



14.5.3

ECC Prime Data Formats

The quantities in this section are equal to the precision of the chosen operation (160, 192, ...) rounded up to a multiple of 8 bytes (i.e., rounded to 3 DWords for precision 160 and 192, to 4 DWords for precision 224 and 256, to 6 DWords for precision 384, and to 8 DWords for precision 512), and then padded with zeroes when necessary.

The only exception to this is the key quantity (k) which must be rounded up to 32 bytes in all cases and padded with zeroes. Therefore the key must be 4 DWords (160, 192, 224, 256) or 8 DWords (384, 512).

For each operation and length group, the total length (in DWords) read or written is specified at the bottom of each data structure.

In all content in this section, m is the modulus and CST is the constant such that

$$CST = 2^{(2 * \text{length} + 4)} \bmod m \quad (\text{EQ 14.a})$$

In all content in this section, parameters a and b are the curve parameters.

Table 14-21. ECC Prime Data Formats

Input Action	Data In	Output Action	Data Out
UC Load			
src+glb_off->	DWord_0 . . DWord_511 512 dw		N/A
Point Multiplication $R(x_r, y_r) = P(x_p, y_p) + Q(x_q, y_q)$			
src+glb_off->	x_p x_p y_p y_p a k m cst $7x(3/4/6/8)+(4/4/8/8) = 25/32/50/64$ dw	dst+glb_off->	x_r y_r $2x(3/4/6/8) = 6/8/12/16$ dw
Point Addition $R(x_r, y_r) = P(x_p, y_p) + Q(x_q, y_q)$			
src+glb_off->	x_p y_p x_q y_q a m cst $7x(3/4/6/8) = 21/28/42/56$ dw	dst+glb_off->	x_r y_ra $2x(3/4/6/8) = 6/8/12/16$ dw
Point Double $R(x_r, y_r) = 2 \cdot P(x_p, y_p)$			
src+glb_off->	x_p y_p a m cst $5x(3/4/6/8) = 15/20/30/40$ dw	dst+glb_off->	x_r y_r $2x(3/4/6/8) = 6/8/12/16$ dw
Point Verification $\text{Is_On_Curve} = P(x_p, y_p) \text{ on curve ? } 1 : 0$			
src+glb_off->	x_p y_p a b m cst $6x(3/4/6/8) = 18/24/36/48$ dw	dst+glb_off->	Is_On_Curve 1 dw
Modular Addition $c = x + y \bmod m$			
src+glb_off->	x y m cst $3x(3/4/6/8) = 9/12/18/24$ dw	dst+glb_off->	c $3/4/6/8$ dw

Table 14-21. ECC Prime Data Formats (*continued*)

Input Action	Data In	Output Action	Data Out
Modular Subtraction $c = x - y \bmod m$			
src+glb_off->	x y m $3x(3/4/6/8) = 9/12/18/24$ dw	dst+glb_off->	c 3/4/6/8 dw
Modular Multiplication $c = x * y \bmod m$			
src+glb_off->	x y m cst $4x(3/4/6/8) = 12/16/24/32$ dw	dst+glb_off->	c 3/4/6/8 dw
Modular Division $c = x / y \bmod m$			
src+glb_off->	x y m $3x(3/4/6/8) = 9/12/18/24$ dw	dst+glb_off->	c 3/4/6/8 dw
Modular Inversion $c = 1 / y \bmod m$			
src+glb_off->	y m $2x(3/4/6/8) = 6/8/12/16$ dw	dst+glb_off->	c 3/4/6/8 dw
Modular Reduction $c = x \bmod m$			
src+glb_off->	x m $2x(3/4/6/8) = 6/8/12/16$ dw	dst+glb_off->	c 3/4/6/8 dw

14.5.4 ECC *Binary* Data Formats

The quantities in this section are equal to the precision of the chosen operation (163, 191, 233) rounded up to a multiple of 8 bytes (i.e., rounded to 3 DWords for precision 163 and 191, or to 4 DWords for precision 233), and then padded with zeroes as necessary.

For each operation and length group, the total length (in DWords) read or written is specified at the bottom of each data structure.

In all content in this section, parameter b is the curve parameter.

Table 14-22. ECC Binary Data Formats

Input Action	Data In	Output Action	Data Out
Point Multiplication $R(x_r, y_r) = k \cdot P(x_p, y_p)$			
src+glb_off->	b k x_p y_p $4x(3/4) = 12/16$ dw	dst+glb_off->	x_r y_r $2x(3/4) = 6/8$ dw
Binary GF Inversion $C(x) = 1 / A(x) \bmod F(x)$			
src+glb_off->	A $1x(3/4) = 3/4$ dw	dst+glb_off->	c $1x(3/4) = 3/4$ dw
Binary GF Multiplication $C(x) = B(x) * A(x) \bmod F(x)$			
src+glb_off->	A B $2x(3/4) = 6/8$ dw	dst+glb_off->	c $1x(3/4) = 3/4$ dw
Binary GF Addition $C(x) = B(x) + A(x) \bmod F(x)$			
src+glb_off->	A B $2x(3/4) = 6/8$ dw	dst+glb_off->	c $1x(3/4) = 3/4$ dw

14.5.5 RSA / ECC Free Out Descriptor Message Format

Upon completion of an operation, the RSA core returns a 2-entry Free Out Descriptor in the following format.

Table 14-23. Security Engine Return Message

Control Descriptor Return							
63:61	60:54	53:52	51:49	48:40	39:5	4:3	2:0
Ctrl	DestinationID	Reserved	Desc Ctrl	Control Error	Source Address (35 msb)	Software_Scratch0	Global src data offset
Data Descriptor Return							
63:61	60:54	53:52	51:49	48:40	39:5	4:3	2:0
Ctrl	DestinationID	Reserved	Desc Ctrl	Data Error	Dest Address (35 msb)	Software_Scratch1	Global dst data offset

Bits	Name	Description
Source Descriptor Return		
63:61	Ctrl	Free encoding
60:54	DestinationID	Destination FMN bucket ID set by sending FMN agent
53:52	Reserved	Reserved = 0
51:49	DescCtrl	Control field of first entry of received (ORIG) descriptor (expected to be SOP)
48:40	ControlError	See Table 14-24 .
39:5	SourceAddress	Address of Source, as sent in the original message
4:3	SoftwareScratch0	Copied from original descriptor. Not used by the engine; the field is returned exactly as it was received. Software may use, for example, to identify the security engine that the free descriptor returns from, or to identify the descriptor/data structure themselves.
2:0	Global src data offset	Number of bytes to right-shift data by before presenting it to engines (0-7); allows realignment of byte-aligned, non-double-word aligned data.
Destination Descriptor Return		
63:61	Ctrl	Free encoding
60:54	DestinationID	Destination FMN bucket ID set by sending FMN agent
53:52	Reserved	Reserved = 0
51:49	DescCtrl	Control field of second entry of received (ORIG) descriptor (expected to be EOP)
48:40	DataError	See Table 14-25 .
39:5	DestinationAddress	Address of Data Descriptor, as sent in the original messages
4:3	SoftwareScratch1	Copied from original descriptor. Not used by the engine; the field is returned exactly as it was received. Software may use, for example, to identify the security engine that the free descriptor returns from, or to identify the descriptor/data structure themselves.
2:0	Global dst data offset	Number of bytes to left-shift (double-word boundary aligned) data by before writing it to memory

The Control and Data Error codes for the RSA core are enumerated in [Table 14-24](#) and [Table 14-25](#), respectively.

Table 14-24. RSA / ECC Control Error Conditions

Code	Condition	Description
0x000	No Error	No Error
0x001	Undefined Op Class	
0x002	Undefined ECC Type	
0x004	Undefined ECC Function	
0x08	ECC time-out	
0x10		<i>Reserved</i>
0x20		<i>Reserved</i>
0x40		<i>Reserved</i>
0x80	Data Read Error	
0x100	Descriptor Ctrl Field Error	(D0.Ctrl != SOP D1.Ctrl != EOP)

Table 14-25. RSA Core Data Error Conditions

Code	Condition	Description
0x000	No Error	No Error
0x001	Exponent Width > Block Width	
0x002	Modulus Width > Block Width	
0x004		<i>Reserved</i>
0x08		<i>Reserved</i>
0x10		<i>Reserved</i>
0x20		<i>Reserved</i>
0x40		<i>Reserved</i>
0x80	Data Read Error	
0x100		<i>Reserved</i>

14.6 Clock Gating

There are two levels of clock gating used in the SAE. One is used at the pipe level and the other is used at the engine level. At the pipe level, the clock will be turned off for a whole pipe unless the pipe is processing data or responding to memory-mapped register read/write requests to its internal registers. For ciphers and authenticators not selected by an operation, cipher and authentication engine clocks are kept off even while the pipe is processing data.

Two defeating bits are associated with this function both of which are located inside the CONFIG2 register (0x508):

- CLK_GATE_DEFEATURE
 - 0: Clock gating enabled (Default case)
 - 1: Clock gating disabled for all pipes.
- PIPE_ENABLE_DEFEATURE
 - 0: Enable register-access clock activation (Default case)
 - 1: Disable register-access clock activation. Disabling register-access clock activation will prevent pipes from responding to LW or SW from CPUs to their internal registers while the pipe is inactive. The LW, SW operations will then stall the CPU. Disabling this operation is preempted by disabling clock gating.

14.7 Security Acceleration Registers

14.7.1 Register Access Model

The SAE's registers are accessed via an address formed as follows:

$$\begin{aligned}
 & \text{XLS IO Base Address} \\
 & + \\
 & \text{SAE block address } (=0x0B000) \\
 & + \\
 & \text{register address offset}
 \end{aligned}$$

14.7.2 Register Summary

Table 14-26. Security Acceleration Engine Registers Summary

Addr Offset	Name	Description	R/W	Reset
0x000 – 0x01FC	Reserved	Reserved	R	0
0x0500	DMA_CREDIT	See DMA_Credit Register .	R/W	0
0x0504	CONFIG1	See Config1 Register .	R/W	0
0x0508	CONFIG2	See Config2 Register .	R/W	0
0x050C	CONFIG3	See Config3 Register .	R/W	0x81312D00
0x0510 – 0x0C1C	Reserved	Reserved	R	0
0x0C20	SecBiu	Number of destination Bucket credits needed to send a message. 0: 1 credit 1: 2 credits 2: 3 credits 3: 4 credits		0x2
0x0C24 – 0x0C7C	Reserved	Reserved	R	0
0x0C80	SEC_MSG_BUCKET0_SIZE	Message Bucket 0 Size in Bytes	R/W	0x20
0x0C84	SEC_MSG_BUCKET1_SIZE	Message Bucket 1 Size in Bytes	R/W	0x20
0x0C88 – 0x0DFC	Reserved	Reserved		
0x0E00	CPU0_MSG_BUCKET0_CREDIT	Number of credits SEC block has to write to Bucket0 of Core0	R/W	0
0x0E04	CPU0_MSG_BUCKET1_CREDIT	Number of credits SEC block has to write to Bucket1 of Core0	R/W	0
0x0E08	CPU0_MSG_BUCKET2_CREDIT	Number of credits SEC block has to write to Bucket2 of Core0	R/W	0
0x0E0C	CPU0_MSG_BUCKET3_CREDIT	Number of credits SEC block has to write to Bucket3 of Core0	R/W	0
0x0E10	CPU0_MSG_BUCKET4_CREDIT	Number of credits SEC block has to write to Bucket4 of Core0	R/W	0
0x0E14	CPU0_MSG_BUCKET5_CREDIT	Number of credits SEC block has to write to Bucket5 of Core0	R/W	0

Table 14-26. Security Acceleration Engine Registers Summary (continued)

Addr Offset	Name	Description	R/W	Reset
0x0E18	CPU0_MSG_BUCKET6_CREDIT	Number of credits SEC block has to write to Bucket6 of Core0	R/W	0
0x0E1C	CPU0_MSG_BUCKET7_CREDIT	Number of credits SEC block has to write to Bucket7 of Core0	R/W	0
0x0E20	CPU1_MSG_BUCKET0_CREDIT	Number of credits SEC block has to write to Bucket0 of Core1	R/W	0
0x0E24	CPU1_MSG_BUCKET1_CREDIT	Number of credits SEC block has to write to Bucket1 of Core1	R/W	0
0x0E28	CPU1_MSG_BUCKET2_CREDIT	Number of credits SEC block has to write to Bucket2 of Core1	R/W	0
0x0E2C	CPU1_MSG_BUCKET3_CREDIT	Number of credits SEC block has to write to Bucket3 of Core1	R/W	0
0x0E30	CPU1_MSG_BUCKET4_CREDIT	Number of credits SEC block has to write to Bucket4 of Core1	R/W	0
0x0E34	CPU1_MSG_BUCKET5_CREDIT	Number of credits SEC block has to write to Bucket5 of Core1	R/W	0
0x0E38	CPU1_MSG_BUCKET6_CREDIT	Number of credits SEC block has to write to Bucket6 of Core1	R/W	0
0x0E3C	CPU1_MSG_BUCKET7_CREDIT	Number of credits SEC block has to write to Bucket7 of Core1	R/W	0
0x0E40	CPU2_MSG_BUCKET0_CREDIT	Number of credits SEC block has to write to Bucket0 of Core2	R/W	0
0x0E44	CPU2_MSG_BUCKET1_CREDIT	Number of credits SEC block has to write to Bucket1 of Core2	R/W	0
0x0E48	CPU2_MSG_BUCKET2_CREDIT	Number of credits SEC block has to write to Bucket2 of Core2	R/W	0
0x0E4C	CPU2_MSG_BUCKET3_CREDIT	Number of credits SEC block has to write to Bucket3 of Core2	R/W	0
0x0E50	CPU2_MSG_BUCKET4_CREDIT	Number of credits SEC block has to write to Bucket4 of Core2	R/W	0
0x0E54	CPU2_MSG_BUCKET5_CREDIT	Number of credits SEC block has to write to Bucket5 of Core2	R/W	0
0x0E58	CPU2_MSG_BUCKET6_CREDIT	Number of credits SEC block has to write to Bucket6 of Core2	R/W	0
0x0E5C	CPU2_MSG_BUCKET7_CREDIT	Number of credits SEC block has to write to Bucket7 of Core2	R/W	0
0x0E60	CPU3_MSG_BUCKET0_CREDIT	Number of credits SEC block has to write to Bucket0 of Core3	R/W	0
0x0E64	CPU3_MSG_BUCKET1_CREDIT	Number of credits SEC block has to write to Bucket1 of Core3	R/W	0
0x0E68	CPU3_MSG_BUCKET2_CREDIT	Number of credits SEC block has to write to Bucket2 of Core3	R/W	0
0x0E6C	CPU3_MSG_BUCKET3_CREDIT	Number of credits SEC block has to write to Bucket3 of Core3	R/W	0
0x0E70	CPU3_MSG_BUCKET4_CREDIT	Number of credits SEC block has to write to Bucket4 of Core3	R/W	0

Table 14-26. Security Acceleration Engine Registers Summary (continued)

Addr Offset	Name	Description	R/W	Reset
0x0E74	CPU3_MSG_BUCKET5_CREDIT	Number of credits SEC block has to write to Bucket5 of Core3	R/W	0
0x0E78	CPU3_MSG_BUCKET6_CREDIT	Number of credits SEC block has to write to Bucket6 of Core3	R/W	0
0x0E7C	CPU3_MSG_BUCKET7_CREDIT	Number of credits SEC block has to write to Bucket7 of Core3	R/W	0
0x0E80 – 0x0FFC	<i>Reserved</i>	<i>Reserved</i>	R	0

14.7.3 SEC_STATE0 Register

Address Offset: 0x0620

On Writes:

Bits	Name	Description	R/W	Reset
31:0	<i>Reserved</i>	<i>Reserved</i>	R	0

On Reads:

Bits	Name	Description	R/W	Reset
31:0	<i>Reserved</i>	<i>Reserved</i>	R	0

14.7.4 DMA_Credit Register

Address Offset: 0x0500

Bits	Name	Description	R/W	Reset
31:12	Reserved	Reserved	R	0
11:9	Rsa0Out	RSA to DMA write credits	R/W	0
8:6	Rsa0In	DMA to RSA read credits	R/W	0
5:3	Pipe0Out	Pipe0 to DMA write credits	R/W	0
2:0	Pipe0In	DMA to Pipe0 read credits	R/W	0

14.7.5 Config1 Register

Address Offset: 0x0504

Bits	Name	Description	R/W	Reset
31:2	Reserved	Reserved	R	0
1	RSA_LITTLE_ENDIAN	RSA little endian mode	R/W	0

14.7.6 Config2 Register

Address Offset: 0x0508

Bits	Name	Description	R/W	Reset
31:3	Reserved	Reserved	R	0
2	PIPE_ENABLE_DEFFEATURE	0: Enable register-access clock activation (Default case) 1: Disable register-access clock activation. Disabling register-access clock activation will prevent pipes from responding to LW or SW from CPUs to their internal registers while the pipe is inactive. The LW, SW operations will then stall the CPU. Disabling this operation is preempted by disabling clock gating.	R/W	0
0	PIPE0_DEF_DBL_ISS	0: Allow security pipe0 to grab next pending descriptor as soon as its front end is done with the previous descriptor, while the back end is still processing it 1: Process one descriptor at a time	R/W	0

14.7.7 Config3 Register

Address Offset: 0x050C

Bits	Name	Description	R/W	Reset
31	RSA_ECC_TIMEOUT_EN	RSA/ECC Timeout Operation Enable 0: Disable 1: Enable	R/W	1
30:0	ECC_TIMEOUT	RSA/ECC Timeout Count. The preset value of the ECC timeout counter, which counts down at the core clock rate. If a timeout occurs and the operation completes, the Free Out message from the Security Accelerator to the CPU will have the ECC timeout bit set in its status field (see Table 14-24).	R/W	0x1312D00

14.8 Fragmentation and Offset

14.8.1 Rebuilding Packets from Fragments on 64-bit Boundaries

The Offset before data/IV on the first fragment is always written back. Non-zero destination 64-bit word or global offsets may cause more data to be written than the user-specified length.

Below is a source (first fragment) packet (@ ADD0 cache-aligned address). Assume we just copy it and relevant data starts on 64-bit word 3 so Cipher_Offset = IV_Offset = 3 (64-bit words). D0X denotes relevant data and G denotes don't care data. Offset data is also copied so Packet_Length = 9 (64-bit words) * 8 = 72 (bytes) Segment_src_address = ADD0

For example, if we want to copy so that the relevant (i.e., D0X) data starts at (cache-aligned address) ADD1, we need to specify Dst_dword_offset = 1 so D00 is moved from 64-bit word position 3 to 0 on next cache-line Cipher_dst_address = ADD1 - 0x20 so D00 is written to ADD1.

Note that the security engine always writes full cachelines therefore, data written to 64-bit word 0 of ADD1 (denoted with "?" below) is what the sec_pipe write back buffer contained from the previous operation.

Source					Destination									
Segment_src_address = ADD0					Cipher_dst_address = ADD1 - 0x20									
Packet_Legth = 72					Dst_dword_offset = 1									
Cipher_Offset = 3														
IV_Offset = 3														
Use_IV = ANY														
3	2	1	0		3	2	1	0						
D00	G	G	G	<- ADD0	G	G	G	?	<- ADD1 - 0x20					
D04	D03	D02	D01		D03	D02	D01	D00	<- ADD1					
			D05				D05	D04						

On fragments following the first, IV_Offset is overloaded to mean data offset (number of 64-bit words to skip from beginning of cacheline before starting processing) and Use_IV is overloaded to mean writeback the offset (in the clear). These fields in combination with Dst_dword_offset allow packet fragments with arbitrary boundaries/length to be reassembled.

Assume the data above was the first fragment of a packet we'd like to merge to a (second) fragment below located at ADD2. The written data should follow the previous data without gaps or overwrites. To achieve this, one should assert the "Next" field on the previous fragment and use the parameters below.

Source					Destination				
Segment_src_address = ADD2					Cipher_dst_address = ADD1 + 0x20				
Packet_Legth = 104					Dst_dword_offset = 1				
IV_Offset = 1									
Use_IV = 0									
3	2	1	0		3	2	1	0	
D12	D11	D10	G	<- ADD2	G	G	G	?	<- ADD1 - 0x20

Source					Destination				
D16	D15	D14	D13	<- ADD1	D03	D02	D01	D00	<- ADD1
D1a	D19	D18	D17		D11	D10	D05	D04	<- ADD1 + 0x20
			D1b		D15	D14	D13	D12	
					D19	D18	D17	D16	
						D1b	D1a		

It is note-worthy that the merging can only be achieved if Use_IV is 0. Indeed, the security engine always writes full lines, therefore ADD1 + 0x20 will be re-written. Setting Use_IV to 0 will allow the security pipe write-back buffer to preserve D04, D05 from the previous fragment and only receive D10, D11 thereby preserving the integrity of the previous data.

On fragments following the first, negation of UseIV in combination with the condition Dst_dword_offset >= (4 - IV_Offset) will cause a wraparound of the write, thus achieving all 16 possible (Initial_Location, Final_Location) combinations for the data. That is:

```
if ( !UseIV && (Dst_dword_offset >= (4 - IV_Offset) ) ) {
    // wraparound
}
```

This example contiguously merges the two data sets above with a third located at ADD3. If this is the last fragment, reset its Next bit.

Source					Destination				
Segment_src_address = ADD3					Cipher_dst_address = ADD1 + 0x80				
Packet_Legth = 152					Dst_dword_offset = 3				
IV_Offset = 3									
Use_IV = 0					3	2	1	0	
3	2	1	0		3	2	1	0	
D20	G	G	G	<- ADD2	G	G	G	?	<- ADD1 - 0x20
D24	D23	D22	D21		D03	D02	D01	D00	<- ADD1
D28	D27	D26	D25		D11	D10	D05	D04	<- ADD1 + 0x20
D2C	D2B	D2A	D29		D15	D14	D13	D12	
	D2F	D2E	D2D		D19	D18	D17	D16	
					D21	D20	D1b	D1a	<- ADD1 + 0x80
					D25	D24	D23	D22	
					D29	D28	D27	D26	
					D2D	D2C	D2B	D2A	
					(D2D)	(D2C)	D2F	D2E	

It is worth noticing that always writing full cache-lines causes the last 2 64-bit words in the reconstituted packet to be unnecessarily written: (D2d) and (D2c).

14.8.2**Implications of Fragmentation on AES**

AES is a 128 bit block cipher; therefore it requires an even 64-bit word total data length. Data fragments (provided there are more than 1) are allowed to have odd 64-bit word data lengths provided the total length (cumulated over fragments) is an even 64-bit word count; an error will be generated otherwise, upon receiving the last fragment descriptor.

While using fragments with AES, a fragment (other than first) starting with a != 0 (IV) offset while the subsequent total 64-bit word count given to AES is odd may not be required to write its offset (UseIV). Doing so will cause an error.

Suppose the first fragment has an odd DATA 64-bit word count and uses AES (as shown below).

Source					Destination									
Segment_src_address = ADD0					Cipher_dst_address = ADD1									
Packet_Legth = 64					Dst_dword_offset = 1									
Cipher_Offset = 3														
IV_Offset = 1														
Use_IV = 1														
Cipher = Any AES														
Next = 1														
3	2	1	0		3	2	1	0						
D00	IV1	IV0	G	<- ADD0	E00	IV1	IV0	G	<- ADD1					
D04	D03	D02	D01		X	E03	E02	E01	<- ADD1 + 0x20					

At the end of processing of the previous fragment, the AES engine input buffer has D04 and waits for the next 64-bit word, therefore the writeback buffer cannot finish writing the fragment to destination (X instead of E04).

If a second fragment now arrives with a non-0 offset and requires the offset data to be written to the destination, the previous write (still needing the arrival of the last 64-bit word required by the AES to complete the previous operation) cannot complete before the present should start, causing a deadlock.

14.9 Programming Model

This section describes how to set up and operate the SAE.

- [Section 14.9.1](#) presents the settings that may not be ignored if the security block is to work. It mostly refers to resettable registers requiring non-zero values.
- [Section 14.9.2](#) presents additional features/options.
- [Section 14.9.3](#) deals with idiosyncrasies and things to be avoided.

For more details on how messaging works, refer to XLS Family Programming Reference Manual Chapter 12, “Fast Messaging Network”.

For coprocessor registers, see XLS Family Programming Reference Manual Chapter 2, “Processor Core Registers”.

14.9.1 Basic Configuration

This section presents the required settings for security block operation. It mostly refers to resettable registers requiring non-zero values.

14.9.1.1 Setting Credits for Sending Messages to the SAE

Each FMN Station keeps count of the number of messages it may send to each Bucket. The initial value is set as a Credit count, decremented for every message sent, then incremented (when the message is used).

The count is the number of 64-bit Entries in the FMN message, thus each Security Descriptor costs 2 Credits. See discussion of credit accelerators in the XLS Programmer’s Reference Manual.

The SAE only uses two of the available 8 Buckets in its FMN station, one for each pipe. Bucket 0 is hardwired to pipe0. Bucket 1 is hardwired to the RSA pipe.

Note: The sum of all Credits of all FMN Stations for any given security Bucket MAY NOT EXCEED the size of that Bucket, lest that bucket be overrun. See Chapter 12.

Note: All threads of a CPU core share these registers; writes from different threads of same CPU core will overwrite each other.

The following example sets up an XLS Processor with 4 and 2 Credits for interacting with SecPipe0 and RSApipe, respectively.

Assembly example:

```
/* Set up secpipe0 thread credits to 4 */
li      t0, 4
mtc2   t0, $31,0
/* Set up rsapipe thread credits to 2 */
li      t0, 2
mtc2   t0, $31,1
```

C-example:

```
write_32bit_cp2_register_sel($31, 4, 0);
write_32bit_cp2_register_sel($31, 2, 1);
```

Default (out of reset) value for those registers: 0x0.

14.9.1.2 Setting Security Credits for Sending Messages from the SAE

This is the reciprocal of the previous example. The security block has a certain (separate) number of credits for sending messages to each Bucket of each FMN Station. In the case of the SAE, these are only used for sending back Free Descriptors.

Note that there is no a priori correspondence between Buckets and Threads (see Chapter 12). The security Free Descriptors have two entries, therefore they cost 2 Credits each.

Note: The sum of all Credits of all blocks for any Bucket of any Station MAY NOT EXCEED the size of that Bucket.

In the examples below, the number of credits of the security block is set up to send messages to bucket 2 of XLS Processor 0 to 8:

Assembly Example

```
/* Set SEC->CPU3.BCKT5 credits to 8          */
/*                                         */
/* PHNX_IO_DEV_SECURITY_BASEDIGITS = 0x0B      */
/* SEC_CNFG_MSG_CREDIT0_SEL = 0x0E              */
#define CPU_IDX      0
#define BUCKET_IDX   2
daddiu t0,      gp,      boot1_info_io_base_off
ld      t0,      0x0(t0)
li      t1,      (PHNX_IO_DEV_SECURITY_BASEDIGITS << 12)
ori      t1,      t1,      (SEC_CNFG_MSG_CREDIT0_SEL << 8)
ori      t1,      t1,      (CPU_IDX << 5)
ori      t1,      t1,      (BUCKET_IDX << 2)
daddu  t0,      t0,      t1
li      t1,      0x8
sw      t1,      0(t0)
```

C-example:

```
#define CPU_IDX      0
#define BUCKET_IDX   2
sec_write_reg(SEC_CC_CPU0_0 + CPU_IDX * 8 + BUCKET_IDX, 0x8);
```

Default (out of reset) value for this registers: 0x0

Note that there is a total of 128 such registers in the security block. The first 64 are for sending to CPUs and should be the only ones to be set, because the SAE can only send meaningfully to Cores. The other 64, based at (SEC_CNFG_MSG_CREDIT1_SEL = 0x0F) should remain 0.

14.9.1.3

Setting Credits for Security DMA to Security BIU

Each of two pipes has two credit settings (reads, writes) ranging between 0 and 7. The register holding these settings resides in the CONFIG group of the security block register space. The counter decrements based on DMA->BIU outstanding (not yet honored) r/w transaction requests.

Note: FMN Credits and BIU Credits are not to be confused. The former are for sending messages on the FMN; the latter are for transacting memory access.

Table 14-27. Security to DMA Credit Register Format

31:12	11:9	8:6	5:3	2:0
	RSA0Out	Rsa0In	Pipe0Out	Pipe0In

Assembly Example

```

/* Set all 4 DMA-BIU credits to 4 */
/*
 * PHNX_IO_DEV_SECURITY_BASEDIGITS = 0x0B */
/* SEC_CNFG_GEN_SEL = 0x05 */
daddiu t0,      gp,      boot1_info_io_base_off
ld     t0,      0x0(t0)
li     t1,      (PHNX_IO_DEV_SECURITY_BASEDIGITS << 12)
ori   t1,      t1,      (SEC_CNFG_GEN_SEL << 8)
daddu t0,      t0,      t1
li     t1,      0x924
sw     t1,      0(t0)

```

C-example

```
sec_write_reg(SEC_DMA_CREDIT, 0x924);
```

Default (out of reset) value for this registers: 0x0

14.9.2**Optional Configuration Settings**

This section presents additional features/options.

14.9.2.1**Setting SAE Bucket Sizes**

The SAE (as do all others) has 8 reconfigurable size buckets.

Bucket 0 is hardwired to pipe0 (Crypto core), and Bucket 1 is hardwired to the RSApipe.

Buckets 2 through 7 are not used.

The cumulated size of all buckets has to be 256. Each bucket size may only be a power of 2 >= 4. A bucket size set to 0 disables that bucket. Refer to mips64.txt for additional restrictions regarding bucket size/adjacency.

The example below sets size of bucket 1 to 64.

```
BCKT0_SIZE_ADDR 0x80
BCKT1_SIZE_ADDR 0x84
```

Assembly Example

```

/* Set SEC.BCKT2_SIZE to 8
 */
/* PHNX_IO_DEV_SECURITY_BASEDIGITS = 0x0B */
/* SEC_CNFG_MSG_BUCKET_SEL = 0x0C */
#define BCKT2_SIZE_ADDR 0x88
daddiu t0,      gp,      boot1_info_io_base_off
ld     t0,      0x0(t0)
li     t1,      (PHNX_IO_DEV_SECURITY_BASEDIGITS << 12)
ori   t1,      t1,      (SEC_CNFG_MSG_BUCKET_SEL << 8)
ori   t1,      t1,      BCKT2_SIZE_ADDR
daddu t0,      t0,      t1
li     t1,      0x40
sw     t1,      0(t0)

```

C-example

```
#define BCKT2_SIZE_IDX 2
sec_write_reg(SEC_MSG_BUCKET0_SIZE + BCKT2_SIZE_IDX, 0x40);
```

Default (out of reset) value for this registers: 0x20.

14.9.2.2**Security to DMA and BIU to Bridge Credit Settings**

Both are hardwired. For the SEC/DMA side, there is no credit setting for writes; the DMA takes write requests based on BIU availability. The meaning of read credits roughly amounts to the number of cachelines the client (SEC) can receive in a burst without being overrun. The FIFO receiving the data is 4-deep, therefore the read credits are hard-wired to 4.

The BIU/bridge Credit read and write Credit settings are hard-wired to 8.

14.9.2.3**Round-Robin Mode*****Assembly Example***

```
/* Set bit 0 of CONFIG2 register */  
/* PHNX_IO_DEV_SECURITY_BASEDIGITS = 0x0B */  
/* SEC_CNFG_GEN_SEL = 0x05 */  
/* SEC_CNFG_CNFG2 = 0x8 */  
daddiu t0, gp, boot1_info_io_base_off  
ld t0, 0x0(t0)  
li t1, (PHNX_IO_DEV_SECURITY_BASEDIGITS << 12)  
ori t1, t1, (SEC_CNFG_GEN_SEL << 8)  
ori t1, t1, SEC_CNFG_CNFG2  
daddu t0, t0, t1  
lw t1 0(t0)  
ori t1, t1, 0x1  
sw t1 0(t0)
```

C-example

```
uint32_t val;  
val = sec_read_reg(SEC_DBLISS_RR);  
val |= 0x1;  
sec_write_reg(SEC_DBLISS_RR, val);
```

14.9.3**Additional Considerations**

This section deals with idiosyncrasies and issues to be avoided.

14.9.3.1**Offset and Padding**

Both 8-byte boundary with arbitrary byte count masking as well as 512-bit boundary padding modes are supported.

The authentication module has a 512 bit block size which is why this mode exists; however the IP Encapsulation Security Payload (ESP) RFC (RFC 2406) specifies in section 3.3.4 that as far as IPSEC is concerned, SHA-1 and MD5 are viewed as having a 1-byte block size, hence the necessity for the 8-byte mode and mask. In the 8-byte mode, the authentication module does its own padding and masks out (with zeros) the number of LSB bytes of the last 8 byte word specified by the user.

To use the 512-bit boundary mode:

- Set the PadHash field of srcLengthIVOffUseIVNext to 0 (see [Table 14-11](#)).
- Pad the data sent to the authentication module so that its size is a multiple of 512 bits. The padding should be done with zeros except for
 - The first pad byte which should be 0x80
 - The last 64-bit word of data seen by the authentication module which should be the total data length excluding padding, including keys (if in HMAC mode and if you choose to load the keys), expressed in bits and right-aligned.

- The HashByteCount field of srcLengthIVOffUseIVNext is ignored in this mode.

To use the 8-byte boundary mode:

- Set the PadHash field of srcLengthIVOffUseIVNext to 1.
- Set the HashByteCount field of srcLengthIVOffUseIVNext to the number of most significant bytes (0-7) of the last 8-byte word to use in the calculation (the rest will be zeroed out); note that 0 means all 8 will be used:

Hash Byte Count	Mask Applied to Last 8-byte Word
0	0xFFFFFFFFFFFFFFF
1	0xFF00000000000000
2	0xFFFF000000000000
3	0xFFFFFFF0000000000
...	...
7	0xFFFFFFFFFFFFFFF00

The 8-byte mode is much easier to use.

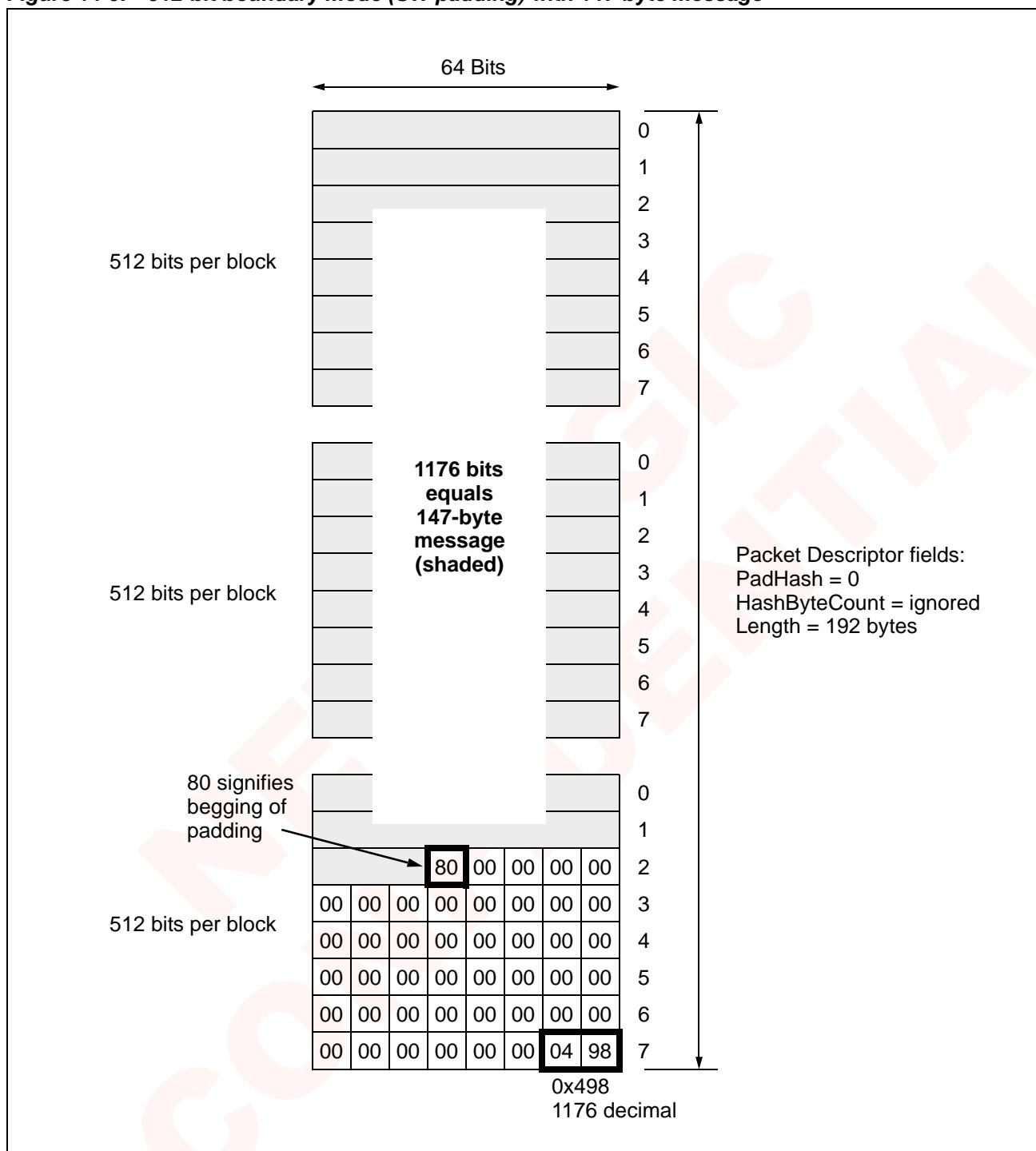
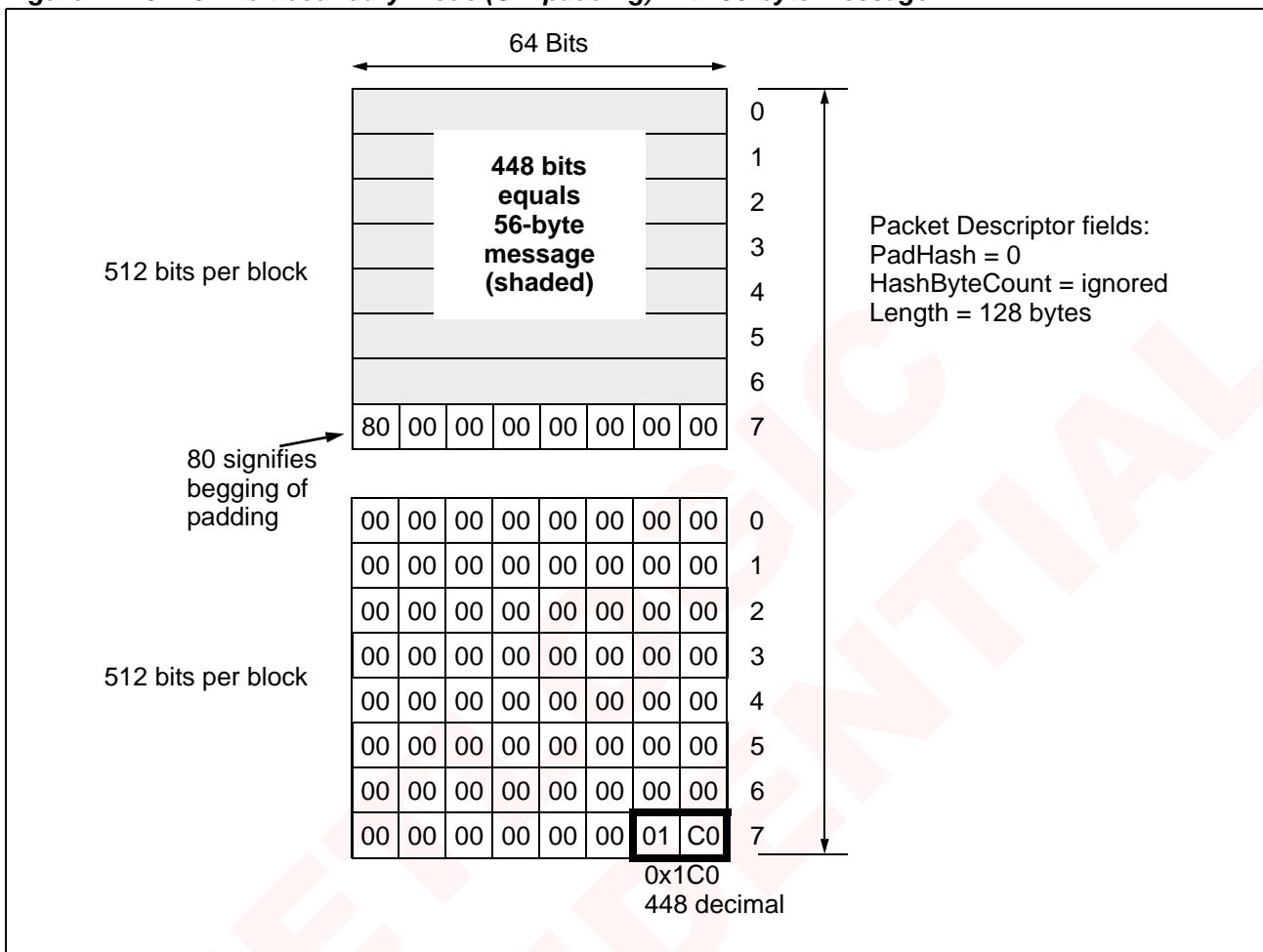
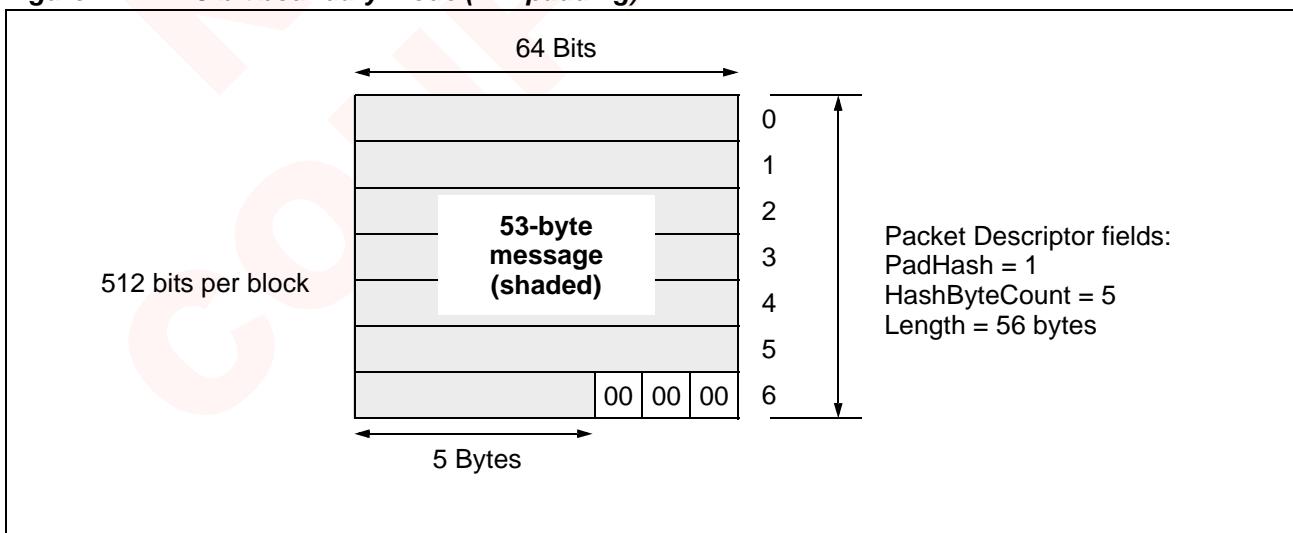
Figure 14-9. 512-bit boundary mode (SW padding) with 147-byte message

Figure 14-10. 512-bit boundary mode (SW padding) with 56-byte message**Figure 14-11. 8-bit boundary mode (HW padding)**

14.10 Application Example

Figure 14-12. Tunneling an IP Packet

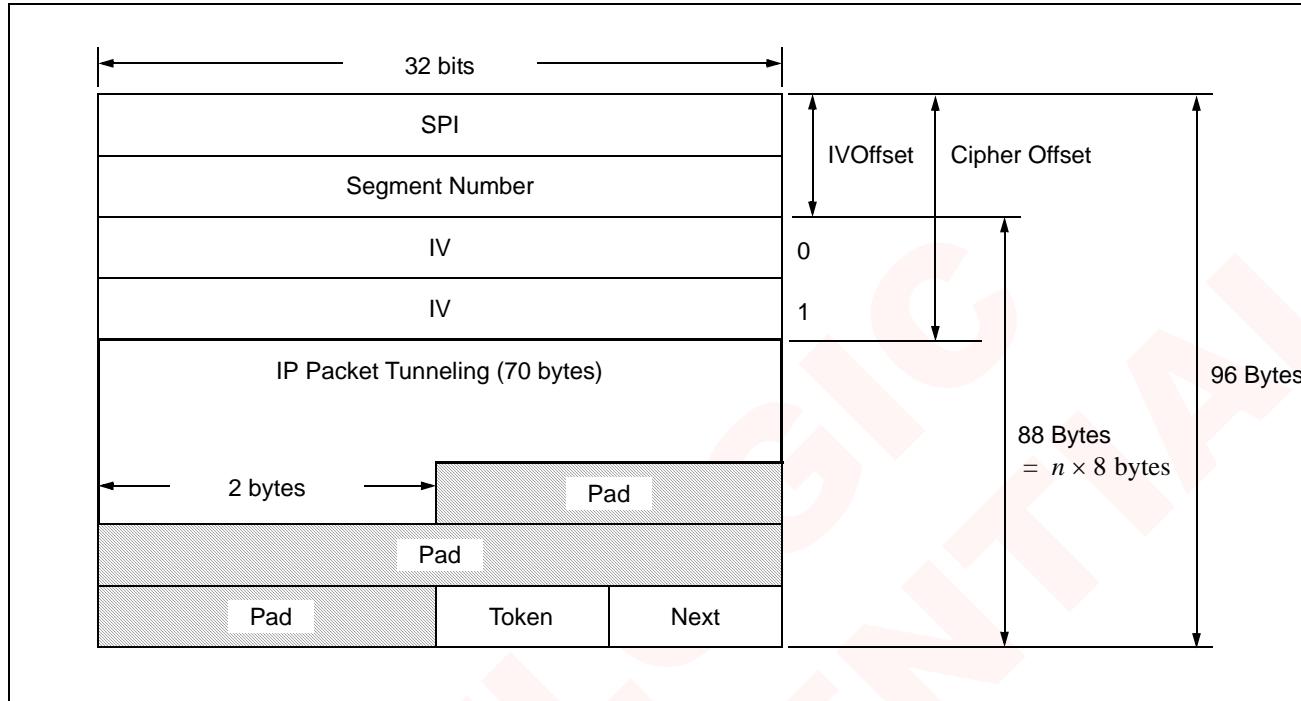


Figure 14-13. In-Memory Representation

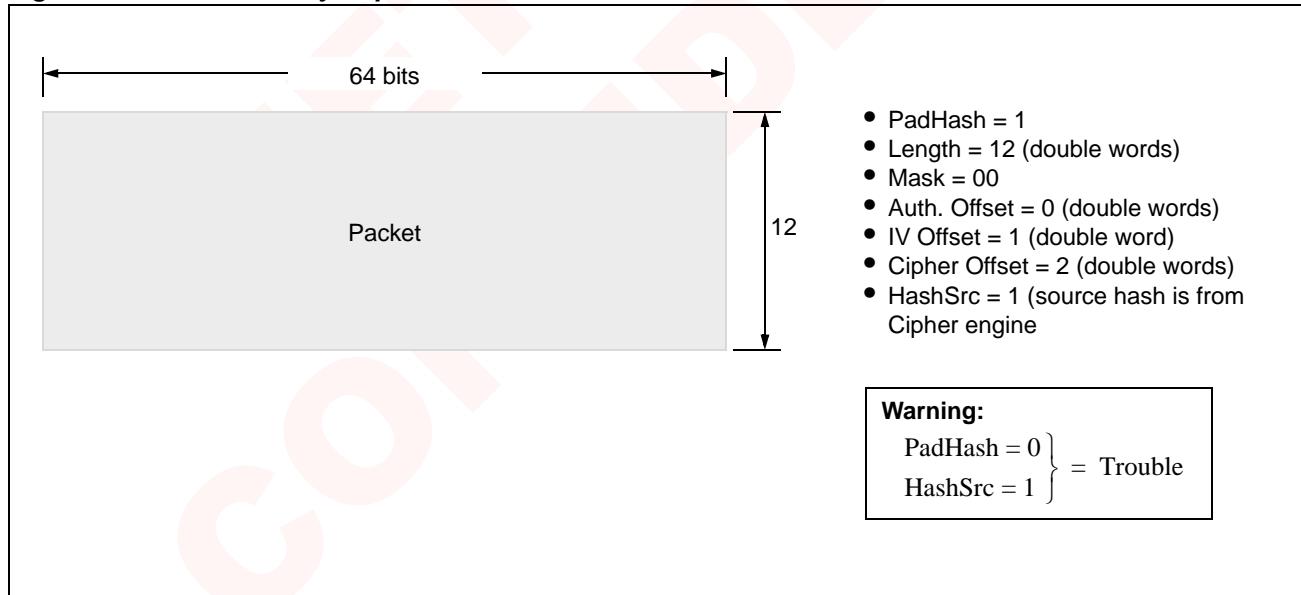


Figure 14-14. Cipher Destination

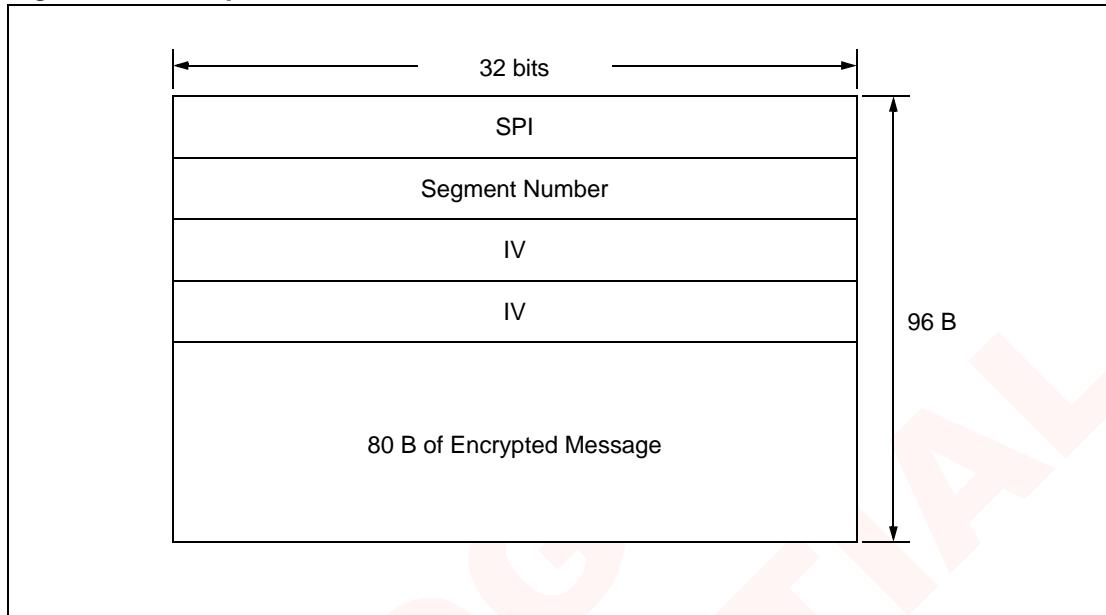
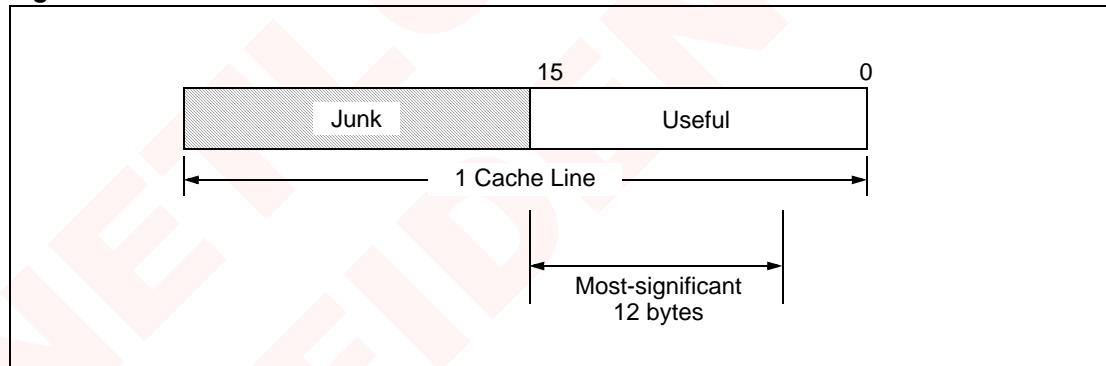


Figure 14-15. Authentication Destination



NETLOGIC
CONFIDENTIAL



Chapter 15 Compression/Decompression Engine

15.1 Introduction

Note: This chapter applies to the XLS6xx, XLS4xx-Lite and XLS4xx devices only. It does not apply to the XLS2xx or XLS1xx devices.

The Compression/Decompression Engine complies with these specifications:

- RFC1950 “ZLIB Compressed Data Format” P. Deutsch, J-L. Gailly, 1996
- RFC1951 “DEFLATE Compressed Data Format” P. Deutsch, 1996
- RFC1952 “GZIP file format” P. Deutsch, 1996

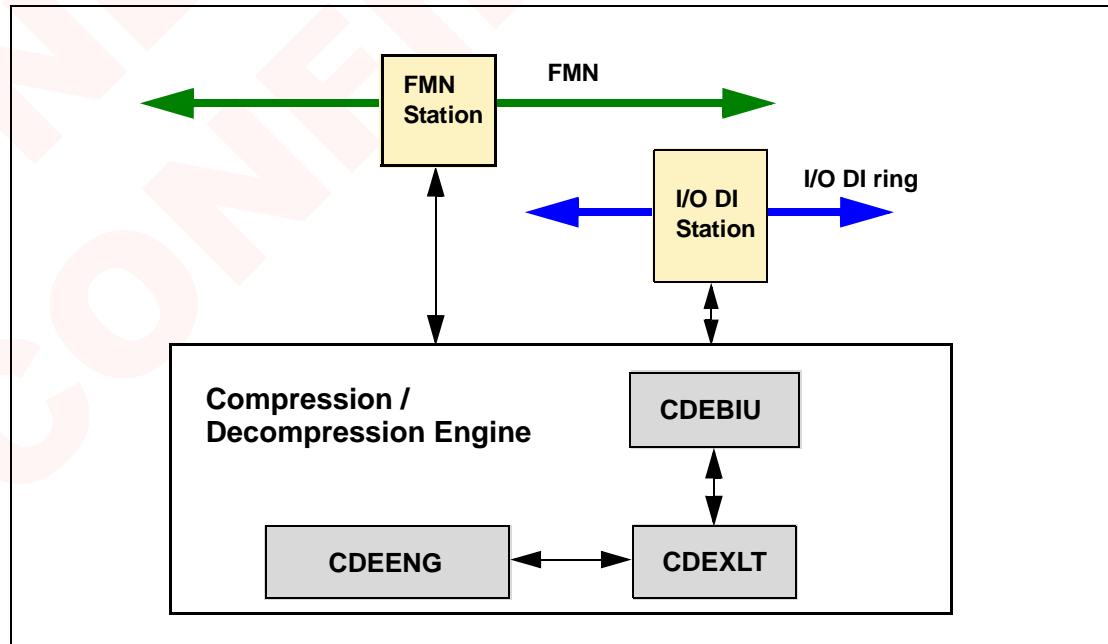
In this document, the Compression/Decompression Engine is abbreviated as “CDE.”

15.2 Capabilities and Functions

15.2.1 Key Capabilities Summary

- 2.5 Gbps operation: Deflate or Inflate or combination
- Operates on string of File Descriptors
- Deflate context save and restore (at block boundaries)
- Inflate context save and restore (at arbitrary file position)

Figure 15-1. Compression/Decompression Engine Block Diagram



15.3 Theory of Operation

At system start-up, software must provide free descriptors to the Compression block. The free descriptors are stored in the Free Descriptor Pool, a FIFO which can hold up to eight descriptors on-chip and which can be extended into memory using a “spill mechanism.” The spill region in memory must be set up to handle cases wherein the Compression block is initialized with more than eight descriptors.

When a Compression/Decompression Message is sent from the CPU to the Compression Block, the message is first decoded and the list of data pointers is fetched from memory. The first pointer in the list points to a scratch page. This page needs to be at least 1.5 Kbyte and is used by the compression engine to store intermediate results. This is useful in case of the save and restore feature, which allows the compression engine to store the intermediate state and work on a different file. When more data for the first file is received, the state can be restored by using the restore feature.

The CDEXLT then walks down the list of data pointers fetching the data and sending it to the CDEENG. The CDEENG performs the transformations on the data and returns it back to the CDEXLT. The CDEXLT then pops 2 free descriptors. The first free descriptor is used for forming the return list. The second descriptor is the location where the transformed data is written. If the data does not fit into 1 descriptor, then more free descriptors are popped from the Free Pool and used to store the data. When all the transformed data has been written to memory, the return message is formed and sent to the Return Bucket. The Return Bucket is a field in the message that was sent from CPU to the CDE.

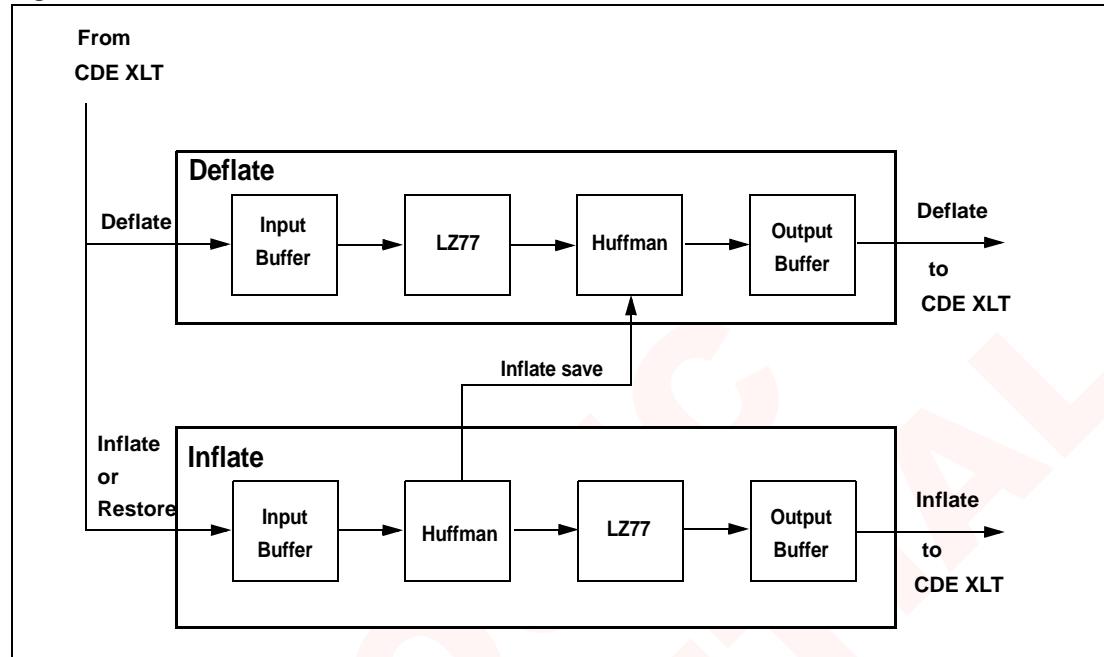
For files that use save/restore, software may return free descriptors back to the compression block after it has received a message and read the transformed data. However, for files that use save/restore, for proper operation these free descriptors must not be sent back to the compression block until the whole file is transformed because following segments might point back to data in previous segments.

15.3.1 Architecture

The CDE is divided into 3 blocks:

- CDEBIU - Bus Interface to the IO Distributed Interface ring.
- CDEXLT - Translate Block interfacing from the CDEENG to the CDEBIU
- CDEENG - performs Compression and decompression operations for the CDE

The Compression/Decompression Engine has a Station on the Fast Messaging Network (FMN) that it uses to send and receive packet data pointers from Threads. See [Figure 15-1](#). FMN messages containing Security Descriptors can be addressed to one of the Buckets associated with the CDE FMN Station.

Figure 15-2. CDEENG Architecture

15.3.2 Modes of Operation

The CDE compression process is called *Deflate* and the CDE decompression process is called *Inflate*. This terminology comes directly from RFC1951.

For both Deflate and Inflate there are three modes of operation:

- *static* compression
- *dynamic* compression
- *no compression*

A file may be split into *blocks*, and each block may use the mode that suits it best.

15.3.2.1 Mode Descriptions

For Deflate, the splitting is done as a pre-process before the file is presented to the CDE. The CDE then performs the requested form of compression before bit-stitching the block to previous blocks in the deflated bit stream. For Inflate, the entire deflated bit stream is fed to the CDE and the CDE itself will switch mode using block header information within the bit stream.

The Deflate and Inflate algorithms use two algorithms to achieve compression: LZ77 and Huffman encoding. LZ77 creates a dictionary of strings of bytes (minimum length 3 bytes) that have occurred previously in the file. It then replaces strings with a *distance* (up to 32768 bytes) to a previous match, and a *length* (up to 258 bytes) for the matching string. If no match exists then the incoming byte is output as a *literal* character. Huffman coding replaces the literal, length and distance codes with codes whose length depends on the frequency of occurrence of the LZ77 codes in the block. For static coding a predefined code is used which will not usually be ideal for the block in question but will still usually achieve good compression. It also has the advantage of speed of execution. Dynamic coding on the contrary is slower since it requires two passes: one pass to create a statistics table of the frequency of occurrence of each LZ77 code, and to use that to generate an optimized Huffman code, and a second pass to make use of the Huffman code to encode the LZ77 data. Its advantage, though, is a higher compression ratio.

Some input data, such as an embedded image file, may already be in a compressed format. Such data will not compress well and may even increase in size. This data can be put into the *no compression* format. The data is split into blocks, each no larger than 65472 bytes. The compression process adds a header for this data type and then outputs the stream as *is*.

15.3.3 Compression/Decompression Engine Processing Flow

15.3.3.1 Deflate LZ77

Within the CDE, data is received from CDEXLT by the Deflate LZ77 Input FIFO. The data is received as 32-byte cache lines - with a byte count to indicate how many bytes are valid on the last word. Words are then written to both a 128-byte Input Buffer, and to the 32Kbyte Byte buffer. The Byte buffer stores the last 32Kbytes of the stream and is used as reference data whenever a match is being determined.

A Dictionary stores buffer locations for matching to. Every 3 input bytes are hashed to give an 11-bit address; so the dictionary has 2K entries. Up to 4 possible match entries are stored at each dictionary entry. Each match entry has a match position, a valid bit and the first two characters of the string. These characters allow rapid rejection of bad matches because many 3 character combinations will match each dictionary entry. Good matches (ones that match the first two characters) proceed to the next stage. Here the location in the Byte buffer is read 16 bytes at a time. It is matched against the string at the current position in the Input buffer. (At the same time up to 3 other strings can be compared, assuming they too are good matches, by using interleaved Byte buffer reads.) The match continues until the longest match is found or until a match exceeds 258 bytes.

The best match provides a *Distance* and *Length* pair, which is output. If there are no good matches, the latest byte is output as a *Literal*. When the block is completed a special code of decimal 256 is output. This code only occurs once within a block and is used to indicate its completion.

The Deflate LZ77 output is passed to the Deflate Huffman block.

15.3.3.2 Deflate Huffman

There are several phases within the Deflate Huffman block. The static Huffman code skips most of them. Instead, a static Look up Table (LUT) is built at startup, and is used to provide a Huffman code for every literal, length or distance subsequently presented to it. All distance codes are 5 bits and there are 30 of them. Literal and Length codes are part of the same 286-entry Look Up. They are 7, 8 or 9 bits in size. Many of the length and distance codes have “extra” data which follows directly after the code word. The code word provides a range of possible lengths or distances and the extra data is used to define an exact length or distance. The number of bits of extra data is a function of the code; the longer code values have longer extra data values.

Dynamic Huffman coding has these phases:

- Pass 1
 - Statistics gathering for each of 286 literal/length codes
 - Statistics gathering for each of 30 distance codes
- Pass2
 - Literal & length Huffman heap build
 - Literal & length Huffman tree build
 - Literal & length Huffman code generation
 - Distance Huffman heap build
 - Distance Huffman tree build
 - Distance Huffman code generation
 - Bit length Huffman heap build
 - Bit length Huffman tree build
 - Bit length Huffman code generation
 - Output bit length code sizes
 - Output literal/length codes using bit length code
 - Output distance codes using bit length code
- Static OR Pass2
 - Output literal/length and distance bit stream

Of these the static Huffman only has to perform the last, using the static LUTs created during initialization. The Deflate Huffman output is a serial bit stream which is sent a byte at a time to CDEXLT. The bit stream is packed with zeroes at the end of file in order to finish on a byte boundary. At 400MHz the maximum transfer rate is 3.2Gbps which provides the upper performance bound.

15.3.3.3 Inflate Huffman

Within CDE in Inflate mode, data is received from CDEXLT by the Huffman Inflate Input FIFO, and from there they are written into the Inflate Input Buffer which is an 8-byte shift register, with new data arriving on the left-most side, and the current data being removed at the right.

The Inflate Huffman block operates in a single phase whether the data is statically or dynamically encoded.

The static LUT is programmed during initialization. A set of comparators is used to determine the length of each incoming literal/length code – 7, 8 or 9 bits; the distance codes are all length 5. An offset is then added to the code to put it into the correct range within the LUT. The output of the LUT gives both the value of the code and the length of any extra data that was appended.

The dynamic LUT is programmed on the fly. These phases of code building are performed which are not required for static codes:

- Read and store the size (1-7 bits) of each bit length code
- Determine the sum of codes of each size for the bit length codes
- Determine the start code for each code size for the bit length codes
- Write the bit length look up table
- Using the bit length LUT, read and store the size (1-15 bits) of each literal/length code
- Using the bit length LUT, determine the sum of codes of each size for the literal/length codes
- Using the bit length LUT, determine the start code for each code size for the literal/length codes
- Write the literal/length look up table
- Using the bit length LUT, read and store the size (1-15 bits) of each distance code
- Using the bit length LUT, determine the sum of codes of each size for the distance codes
- Using the bit length LUT, determine the start code for each code size for the distance codes
- Write the distance look up table

Like the static LUT, a set of comparators is used to determine the size of each incoming literal/length and distance code, but 14 are required since the size can vary from 1 to 15 bits. The output of the LUT gives both the value of the code and the length of any extra data that was appended.

Together, the code and extra data is used to recover the length or distance value (literals are treated like lengths but have no extra data). In this way the original LZ77 sequence is recovered and this is output to the Inflate LZ77 block.

15.3.3.4

Inflate LZ77

Inflate LZ77 receives LZ77 coded data from Inflate Huffman and uses it to reconstruct the original file format. It uses the identical 32Kbyte buffer used by Deflate LZ77, this time using it as the source of the strings specified by the distance and length codes provided by Inflate Huffman. Each decoded byte is both output to CDEXLT and written to the Byte Buffer. In this manner, the previous 32Kbytes of data is always available for reference.

Inflate LZ77 provides its output via a 16-byte wide bus to CDEXLT.

15.3.3.5

Save & Restore

The CDE is designed to perform either Deflate or Inflate sequentially, not both at once. To make it easier to deal with long files which could provide a bottleneck, the CDE architecture allows a **Context Save** to be performed, followed at some arbitrary time later by a **Context Restore** in order to continue (or complete) the stream.

Deflate Save

Deflate Save can be performed only at block boundaries. This is not really a handicap because the processor can make blocks as large as desired. However, the first block must be greater than 32 Bytes. For dynamic blocks, saves always occur at the end of Pass 2, never after Pass 1.

Note: If the total file size is 32 bytes or less and the file must be deflated, then it should be deflated using software compression rather than hardware compression.

Deflate Restore

Deflate save and restore always occurs on block boundaries and no special processing with CDENG is required.

Inflate Save

Inflate Save is performed each time a dynamic Inflate preamble is received. The preamble data is sent to the Deflate Output Buffer where they are sent back to CDEXLT. The saved preamble is always byte-aligned at the beginning, whether the incoming data were aligned or not. Subsequently, if a save is requested 2 cases are handled. If the save request follows the preamble, then CDEXLT has already received and stored the entire preamble; this is called a Full Save. If a save is requested during the preamble, then a Partial Save results. In either case CDE sends an 8-byte Context word which contains any fragmentary data remaining, the state of the Inflate state machine, the type of data: dynamic, static or non-compressed, and a bit to distinguish the Full and Partial Save types.

Inflate Restore

The same 3 restore stages are required:

- Preamble restore
- Load warm-up data and context data (compression software must accumulate at least 300 bytes of Inflate data for the first segment of any inflate save/restore context.)
- Continue stream from where it was terminated

(Dynamic Only) Inflate Restore also begins by reloading the LUTs using the saved Preamble data. This could be Partial, which leaves the LUTs partially programmed.

Next the previous 32Kbytes of Byte Buffer data is restored, and the Context data is reloaded. This reloads the Input Buffer with any fragmentary data and sets the Inflate state appropriately.

Finally the stream is restarted from where it terminated. Whether a Full or Partial Restore was performed, CDE will continue from where it was halted.

15.3.3.6**Adler Checksum & CRC**

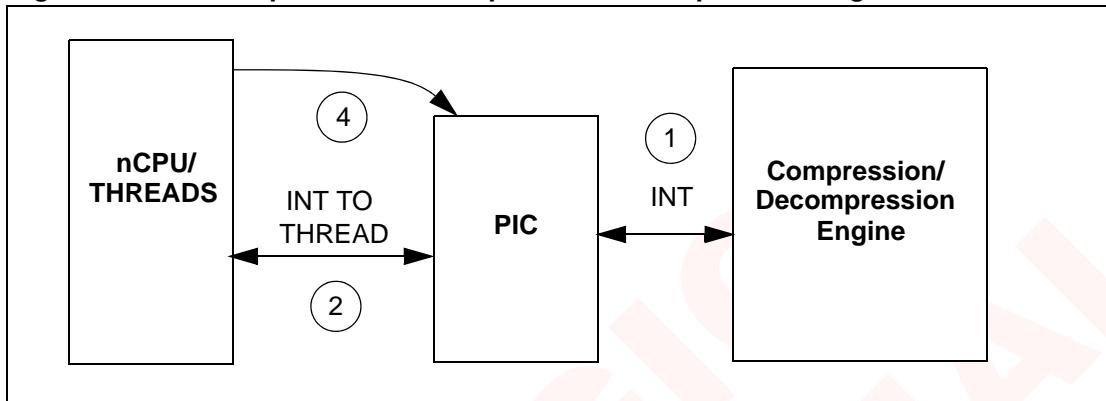
One should understand that the CDE receives only data to be Deflated or the bit stream data portion of packets to be Inflated. One task for system software is to strip the header information and CRC data from Inflate packets before sending the raw Inflate data to CDE.

But whether Deflating or Inflating, data is sent to a CRC block within CDE which computes **both** an Adler checksum and a CRC value. These checksums are then written to the Scratch space and are available for software to either append to a Deflate packet or compare with an Inflate packet checksum as required.

15.3.4**Interrupts**

The Engine is integrated into the system interrupt architecture as follows:

Figure 15-3. Interrupt Model for Compression/Decompression Engine



1. The Engine generates an interrupt to the PIC.
2. The PIC generates an interrupt to a target nCPU.
3. The nCPU reads the PIC to determine which system device generated the interrupt. For this example, the PIC identifies the CDE as the sourcing system device.
4. The nCPU then checks the identified system device to determine interrupt source. In this case, the CPU must read the CDE ISR. (See [Section 15.5.13 INTERRUPT_VEC](#) in this chapter and also Chapter 12, Programmable Interrupt Controller, in this manual.)

15.3.5**Errors and Events Handling for CDE in the System Architecture**

Please refer to Chapter 8 for details.

15.4 Programming Model

This section discusses programmatic set up for operation of the Engine, and programmatic access to it.

There are two main areas in which setup must be done:

- System-level setup
- Engine-level setup

15.4.1 System Setup for Startup and Initialization

The Engine is tightly integrated into the XLS processor's I/O distributed interconnect. Operation at the system level for activity involves the following setup:

15.4.1.1 System Bridge Controller Setup

The SBC resides in the XLS's Memory and Distributed Interconnect Hub. (See [Chapter 8, "Memory and I/O Access"](#) for details.) This handles device-internal address mappings.

Setup here for the CDE access involves XLS_IO_BAR, which is set up once to define the base address for the entire Peripheral and I/O Configuration region in which all internal peripheral configuration header registers and control and status registers lie. All CDE internal registers lie at a group offset from this base, and each register is at an additional offset or index from that group offset.

15.4.1.2 Interrupt Setup

This involves setup in two areas:

- Interrupt Redirection Table (IRT) in the PIC. See discussions in Chapter 12, Programmable Interrupt Controller, of this manual.
- the EIRR for each nCPU that will interact with the CDE. See discussion of EIRR registers in CPUs.

15.4.1.3 Reported Block Length Setup

When the CPU is not processing messages from the Compression Engine fast enough, under certain outstanding message credit and free descriptor conditions, the length of the block reported by the Compression Engine may be 32 bytes less than the actual value. However, the data received is complete and correct.

To ensure the receipt of the correct Block Length the following two setups must be implemented:

1. The number of message credits from CPU to CDE should be less than or equal to one-half the number of credits assigned to the Compression engine. For instance, if 8 allocated CDE credits transmit to nCPU#0, the software on nCPU#0 should maintain 4 message outstanding credits for the compression block, to avoid this issue.
2. In addition, the total number of free descriptors sent to the Compression Engine should be less than or equal to

$$[\text{FreeDescSize} / 8]$$

where FreeDescSize is a programmable field in the CDE [CONTROL_REG_0](#) Register. This prevents the CDE from sending more than two entries of return message when receiving one message from the nCPU.

For example, if FreeDescSize = 1K bytes, the number of free descriptors sent to the Compression Engine should be < 128.

Software initially sends $(\text{FreeDescSize} / 8)$ free descriptors to the CDE. When starting the file process, the free descriptor pool may cross the free descriptor threshold. If it does, it will generate a FreeFifoBelowThreshold interrupt (interrupt Vector Register INTERRUPT_VEC[25]). The nCPU can replenish the correct amount of free descriptors back to the CDE to keep the total number $< (\text{FreeDescSize} / 8)$ until the file operation completes. After completing the file operation, software should only return $(\text{FreeDescSize} / 8)$ free descriptors back to the Compression Engine for the next file process.

If multiple nCPUs are performing transactions with the CDE simultaneously, the same rule must be applied. The total number of free descriptors in the CDE should be less than or equal to $(\text{FreeDescSize} / 8)$.

15.4.2 Interface Setup for Startup and Initialization

When this block is initialized, all the configuration registers should be set in case some reset value must be changed. For example, the Spill regions must be set up, and then free descriptors must be sent to the CDE.

15.4.2.1 Reset Sequence for Engine

See [RESET_REG](#) description on [page 595](#).

15.4.3 Standard Message Interface (Two Buckets)

Bucket 0 — Free Descriptors. These are destination descriptors.

Bucket 1 — Message with SRC descriptions

15.4.4 Standard I/O Ring Interface (Three Channel DMA)

Channel 0 — Data load store

Channel 1 — Chained descriptor load/store

Channel 2 — Spill channel for free descriptors

15.4.5 Message Format

The CDEXLT module interfaces the CDE to the XLS's Fast Messaging Network Ring and I/O DI Ring. This section describes the descriptor format.

To send messages to the CDE, use this format:

15.4.5.1 Send Message (CPU - to - CompressionBlock)

63:62	61	60	59:54	53	52:40	39:0
Reserved	ReadX	Dfl	RET_BKT	0	Length	SRC_ADDR
Bits	Name			Description		
63:62	Reserved			Reserved		
61	ReadX			Read Exclusive bit to DMS		
60	Dfl			1 = Deflate 0 = Inflate		

Bits	Name	Description
63:62	Reserved	Reserved
61	ReadX	Read Exclusive bit to DMS
60	Dfl	1 = Deflate 0 = Inflate

Bits	Name	Description
59:54	RET_BKT	Buckets per CPU 0-7: cpu0 8-15: cpu1 16-23: cpu2 24-31: cpu3
53	Reserved	Reserved—must be set to 0.
52:40	Length	Number of descriptors in memory
39:0	SRC_ADDR	Pointer to list of NEXT_IN pointers

15.4.5.2 Free Pointers (CPU - to - CompressionBlock)

The following addresses are used by the compression block to send back the data.

63:40	39:0
Reserved	FREE_ADDR

15.4.5.3 Return Message (Compression Block - to - CPU)

63:62	60	59:54	53	52:40	39:0
Last	Dfl	RET_BKT	0	Length	SRC_ADDR
0			Length	DST_ADDR	

Bits	Name	Description
63:62	Last	1: For last message 0: More messages to continue
60	Dfl	1 = Deflate 0 = Inflate
59:54	RET_BKT	Buckets per CPU 0-7: cpu0 8-15: cpu1 16-23: cpu2 24-31: cpu3
52:40	Length	Number of descriptors in memory
39:0	SRC_ADDR	Pointer to list of NEXT_IN pointers
39:0	DST_ADDR	Pointer to list of NEXT_IN pointers

15.4.5.4 Descriptor Format

63	62:61	60	59	58	57	56	55:40	39:0
EOF	TYPE	SOD	SOB	Save	Restore	EOB	Length	DATA_ADDR

Bits	Name	Description
63	EOF	EOF has to be asserted during SOB of the last block
62:61	TYPE	No Compression / St Huffman / Dy Huffman 00 - No Compression 01 - St Huffman 10 - Dy Huffman - Pass 1 11 - Dy Huffman - Pass 2
60	SOD	Start of Data useful for setting up a dictionary and restore context The first pointer from which real data starts shall have this bit set.
59	SOB	Start of Block
58	SC	Save Context
57	RC	Restore Context
56	EOB	End Of Block
55:40	Length	Length of Data in bytes. The minimum length is 300 bytes and the maximum length is 64572 bytes.
39:0	DATA_ADDR	Pointer to Data

15.4.5.5 Example Message

The first pointer in the list is to the scratch page and may or may not have the Restore bit set.

63:62	61	60	59:54	53:40	39:0
Reserved	ReadX	Dfl	RET_BKT	Length	SRC_ADDR

SRC_ADDR has to be cache-line aligned.

The SRC_ADDR points to a list of Descriptors in memory. This list looks like the following:

Table 15-1. SRC_ADDR Descriptors List

EOF	Type1	Type0	SOD	SOB	Save	Restore	EOB	Length	Address	Description
0	0	0	0	0	0	1	0	Length_SCRATCH	SCRATCH_PAGE_ADDR	First descriptor always points to Scratch Page
0	0	0	0	1	0	0	0	Length_W0	ADDR_W0	Warm up data
0	0	0	0	0	0	0	0	Length_W1	ADDR_W1	
0	0	1	1	1	0	0	0	Length_0	ADDR_0	Static Huffman block
0	0	1	0	0	0	0	1	Length_1	ADDR_1	
0	1	0	0	1	0	0	0	Length_2	ADDR_2	Dynamic Huffman pass 1
0	1	0	0	0	0	0	1	Length_3	ADDR_3	
0	1	1	0	1	0	0	0	Length_2	ADDR_2	Dynamic Huffman pass 2. ADDR_2 and ADDR_3 are repeated
0	1	1	0	0	0	0	1	Length_3	ADDR_3	
0	0	0	0	1	0	0	1	Length_4	ADDR_4	No Compressed Blocks
1	0	0	0	1	0	0	1	Length_5	ADDR_5	

If the RESTORE bit is set then the first pointer is for the scratch page. The scratch page holds data used by the Compression Engine to store or retrieve intermediate results. Intermediate results are partial CRC, partial Adler or Dynamic Huffman Codes.

The next set of descriptors may be for warm up data. This data is used by the CDE for warming up its dictionary. This does not generate any output. The SOD (Start Of Data) bit is set for the page where the first data starts. For Dynamic Huffman Blocks, the descriptors have to be repeated twice. All descriptors starting with the Start of Block and ending with End Of Block of each Dynamic Huffman Block has to be repeated in the same sequence. On the first pass the TYPE[1:0] should be set to 0x2, and on the second pass TYPE[1:0] should be set to 0x3.

The Last descriptor in the list may or may not have the SAVE bit set. If the SAVE bit is set, the intermediate results are stored back into the SCRATCH_PAGE_ADDR.

15.4.5.6 Save Context Data Structure

Total Size of Scratch Page = minimum 48 cachelines

Cache Line =0 (Contains various Control Bits)

Bits	Name	Description
255:224	CRC32	CRC
223:192	ADLER32	ADLER Checksum
193:186	Reserved	
185:176	DHTL	Dynamic Huffman Tree Length
175	SFWPOverflow	Scratch FIFO Write Pointer Overflow
174:170	Reserved	
169:160	SFWP	Scratch FIFO Write Pointer
159:155	Reserved	
154:152	NumBitsValid	Number of bits valid in Last Partial Byte
151:144	PartialByte	Deflate Last Partial Byte
143:89	Reserved	
Error Status Bits: 88:64		
Other Status		
88	InfHuffInitDone	Inflate Huffman Initialization Done
87	DefHuffInitDone	Deflate Huffman Initialization Done
Inflate LZ77 Error		
86	TimerInt	Timer Interrupt
85	LenCode286_287	inflate LZ77 length_code 286 or 287
84	LongDistCode	inflate LZ77 distance greater than current pointer in file
83	IllegalLenCode	inflate LZ77 illegal length
82	IllegalDistcode	inflate LZ77 illegal distance
Inflate Error		
81	RdIndexOutOfRange	Read index out of range (bl > 18 or litdist > 319)
80	IllegalBFinal	Bfinal and eob, but not last word
79	IBufUnderflow	Input Buffer Underflow
78	IllegalLen	Noncomp ~ Length doesn't match Length
77	IllegalBType	Btype is set to 2'h3
76	IllegalPreamble	Last distance code in preamble has zero length (and there is more than one code)
75	ILenLiteralZero	Last literal/length code in preamble has zero length
74	LastLenZero	Last bit length code in preamble has zero length
Deflate Error		
73	ZeroLenHuffCode	A zero-length Huffman code was read
72	LastHuffNotOnesDist	Last Huffman code for longest distance code is not all 1s
71	LastHuffNotOnesLit	Last Huffman code for longest literal/length code is not all 1s
70	LastHuffNotOnesBitLen	Last Huffman code for longest bit length code is not all 1s
69		A zero-length Huffman code was written
Inflate Status		
68	NonCompZeroLen	Noncomp byte length specified is 0 (not an error but unlikely to be used)

Bits	Name	Description
Deflate Status		
67	DistStatClamp	Distance statistics have clamped
66	LitLenStatClamp	Literal/length statistics have clamped
65	LitLenCodeOverflow	Literal/length or distance code length overflow (not an error since we switch to static)
64	BLCodeOverflow	BL code length overflow (not an error since we switch to predefined table)
63:62	Mode	Inflate: Mode: 0=non compress, 1=static, 2=dynamic, 3=unknown Deflate: Mode: 0=non compress, 1=static, 2=dynamic, 3=unknown (Static encoding will be signaled on dynamic blocks whenever a Huffman code overflow occurs)
61:57	BLDecodeState	BL decode state bits
56	PartialSave	Partial save: The preamble data were incomplete when the save occurred.
55:54	<i>Reserved</i>	
53:48	Count	Count: Number of unused bits (0-47) of received data
47:0	Data	Data: The last 6 bytes of data received from Xlate

Notes:

- cacheline = 1 to 15 (ScratchFifoStartPointer-1) contains the dynamic huffman tree.
- cacheline = 16 (ScratchFifoStartPointer) to 47 (ScratchPageSize-1) contains the egress descriptor list.

15.5 CDE Configuration Registers

15.5.1 Definition of Terms

The following terms are used in the register descriptions to describe field modifiability type:

Table 15-2. Field Modifiability Codes

Type Code	Meaning	Description
R/O	Read Only	The field contains a static value and cannot be written.
R/W	Read/Write	The field can be modified by software. Note that some fields, where indicated, are modifiable only indirectly, by writing to a shadow register then performing a soft reset.
R/C	Read/Clear	The field can be read to give status information. Bits may be cleared by writing a '1' to them; writes of '0' are ignored.

15.5.2 Endianness for the Compression/Decompression Engine

The CDE receives and outputs data in big Endian data format. Within individual bytes, data is stored according to the manner specified in RFC 1951, with Deflated bit streams starting with the least significant bit of the least significant byte.

15.5.3 Register Access

CDE registers are accessed using the combination of individual register offset, plus device base address, plus base address for all XLS_IO devices. See Chapter 8 for information about addressing system elements.

15.5.4 Compression/Decompression Engine Register Summary

The following table summarizes the CDE registers.

Address offsets are byte addressing. The 32-bit registers are on 4-byte boundaries.

Table 15-3. Compression/Decompression Engine Configuration Registers

Addr Offset	Register Name	Reset
0x0	CONTROL_REG_0	0x39CE0400
0x4	DMA_CREDITS	0xFFFFFFFF
0x8	FREE_IN_SPILL_MEM_START_0	0x00000000
0xC	FREE_IN_SPILL_MEM_START_1	0x00000000
0x10	FREE_IN_SPILL_MEM_SIZE	0x00000000
0x14	FREE_IN_SPILL_MEM_BYTES	0x00000000
0x18	CRC_ADLER_SPILL_MBUF	0x00002000
0x1C	SCRATCH_PAGE	0x00402000
0x20	INTERRUPT_VEC	0x00000000
0x24	INTERRUPT_MASK	0x00000000
0x28	FREE_DESC_THRESHOLD	0xFFFF0000
0x2C	DESC_FIFO_WORD_COUNTS	0x00000000
0x30	RESET_REG	0x00000008
0x3A	ERROR_RESET_MASK	0x003FFC00
0x38	ADDRESS_OF_READ_ERROR_LIST0	0x00000000
0x3C	ADDRESS_OF_READ_ERROR_LIST1	0x00000000

15.5.5 CONTROL_REG_0

Reg Address Offset: 0x0
Register default value: 0x39CE0400

Bits	Name	Description	R/W	Reset
31	Reserved	Reserved	R/W	0
30	DataDMAChannelRdAllocate	L2 Cache Read Control 0: Do not store reads in cache 1: When set, if the data is not present in the L2 Cache, then data is stored in L2 Cache during read.	R/W	0
29	Reserved	Reserved	R/W	1
28	DataDMAChannelRdCoherent	Read Coherency Control 0: Do not strongly order reads 1: When set, reads are strongly ordered	R/W	1
27	DataDMAChannelWrCoherent	0: Do not strongly order writes 1: When set, writes are strongly ordered, and the bridge does a read-modify-write for unaligned addresses.	R/W	1
26	DataDMAChannelRdExclusive	L2 Cache Read Exclusive 0: Read operation will not affect the cached data (the data will not be purged from the cache after the read operation). 1: Cache data will be discarded without being written back to external memory	R/W	0
25	FreeInSpillDMAChannelRdAllocate	L2 Cache Read Control 0: Do not store reads in cache 1: When set, if the data is not present in the L2 Cache, then data is stored in L2 Cache during read.	R/W	0
24	FreeInSpillDMAChannelWrAllocate	L2 Cache Write Control 0: For writes not going to L2 cache 1: Set to cause writes to go to L2 cache	R/W	1
23	FreeInSpillDMAChannelRdCoherent	Read Coherency Control 0: Do not strongly order reads 1: When set, reads are strongly ordered	R/W	1
22	FreeInSpillDMAChannelWrCoherent	0: Do not strongly order writes 1: When set, writes are strongly ordered, and the bridge does a read-modify-write for unaligned addresses.	R/W	1
21	FreeInSpillDMAChannelRdExclusive	L2 Cache Read Exclusive 0: Read operation will not affect the cached data (the data will not be purged from the cache after the read operation). 1: Cache data will be discarded without being written back to external memory	R/W	0
20	DesCListDMAChannelRdAllocate	L2 Cache Read Control 0: Do not store reads in cache 1: When set, if the data is not present in the L2 Cache, then data is stored in L2 Cache during read.	R/W	0
19	Reserved	Reserved	R/W	1

Bits	Name	Description	R/W	Reset
18	DescListDMAChannelRdCoherent	Read Coherency Control 0: Do not strongly order reads 1: When set, reads are strongly ordered	R/W	1
17	DescListDMAChannelWrCoherent	0: Do not strongly order writes 1: When set, writes are strongly ordered, and the bridge does a read-modify-write for unaligned addresses.	R/W	1
16	DescListDMAChannelRdExclusive	L2 Cache Read Exclusive 0: Read operation will not affect the cached data (the data will not be purged from the cache after the read operation). 1: Cache data will be discarded without being written back to external memory	R/W	0
15:0	FreeDescSize	Size of Free Descriptors in bytes. This must be a multiple of cacheline size (32)	R/W	0x400

15.5.6 DMA_CREDITS

Reg Address Offset: 0x4
 Register default value 0xFFFFFFFF

Bits	Name	Description	R/W	Reset
31:24	Reserved	Reserved		
23:20	DescListDmaRdCr	Maximum read credits for DMA channel 2	R/W	0xF
19:16	DescListDmaWrCr	Maximum write credits for DMA channel 2	R/W	0xF
15:12	FreeInSpillDmaRdCr	Maximum read credits for DMA channel 1	R/W	0xF
11:8	FreeInSpillDmaWrCr	Maximum write credits for DMA channel 1	R/W	0xF
7:4	DataDmaRdCr	Maximum read credits for DMA channel 0	R/W	0xF
3:0	DataDmaWrCr	Maximum write credits for DMA channel 0	R/W	0xF

15.5.7 FREE_IN_SPILL_MEM_START_0

Reg Address Offset: 0x8
 Register default value

Bits	Name	Description	R/W	Reset
31:0	FreeInSpillMemStart[36:5]	Start of Spill Memory for Free Descriptors. This is in units of cachelines.	R/W	0

15.5.8 FREE_IN_SPILL_MEM_START_1

Reg Address Offset: 0xC

Register default value

Bits	Name	Description	R/W	Reset
31:3	Reserved			
2: 0	FreeInSpillMemStart[39:37]	3 msbs of Start of Spill Memory for Free Descriptors	R/W	0

15.5.9 FREE_IN_SPILL_MEM_SIZE

Reg Address Offset: 0x10

Register default value

Bits	Name	Description	R/W	Reset
31:0	FreeInSpillMemSize	Size of Spill Memory in bytes. This has to be a multiple of cacheline (32)	R/W	0

15.5.10 FREE_IN_SPILL_MEM_BYTES

Reg Address Offset: 0x14

Register default value Read Only

Bits	Name	Description	R/W	Reset
31:0	FreeInSpillMemBytes	Number of bytes in spill memory	R/W	0

15.5.11 CRC_ADLER_SPILL_MBUF

Reg Address Offset: 0x18
Register default value Read Only

Bits	Name	Description	R/W	Reset
31:14	<i>Reserved</i>			
13	CRCReverseByte	Reverse the order of bits in a byte when computing CRC	R/W	1
12	AdlerReverseByte	Reverse the order of bits in a byte when computing Adler32	R/W	0
11:8	FreeSpillOnChipMicroBuffer0	Number of bytes in FreeIn Spill write cacheline buffer 0	R/W	0
7:4	FreeSpillOnChipMicroBuffer1	Number of bytes in FreeIn Spill write cacheline buffer 1	R/W	0
3:0	FreeSpillOnChipMicroBuffer2	Number of bytes in FreeIn Spill read cacheline buffer	R/W	0

15.5.12 SCRATCH_PAGE

Reg Address Offset: 0x1C
Register default value

Bits	Name	Description	R/W	Reset
31:28	<i>Reserved</i>			
27:18	ScratchPageSize	Size of scratch page in units of cachelines	R/W	0x20
17:9	ScratchFifoStartPointer	Start of WarmUp data fifo (cacheline unit)	R/W	0x10
8:0	<i>Reserved</i>			

15.5.13 INTERRUPT_VEC

Reg Address Offset: 0x20
Register default value: Read, Write 1 to clear

Bits	Name	Description	R/W	Reset
31:29	Reserved			
28	DmaChannel2Error		R/W	0
27	DmaChannel1Error		R/W	0
26	DmaChannel0Error		R/W	0
25	FreeFifoBelowThreshold		R/W	0
24	InflateHuffmanInitDone	Inflate Huffman Initialization Done	R/W	0
23	DeflateHuffmanInitDone	Deflate Huffman Initialization Done	R/W	0
22	InflateLZ77ErrorTimerInterrupt	Inflate LZ77 timer Interrupt	R/W	0
21	InflateLZ77ErrorLengthError	Inflate LZ77 length_code 286 or 287	R/W	0
20	InflateLZ77ErrorDistanceError	Inflate LZ77 distance greater than current pointer in file	R/W	0
19	InflateLZ77ErrorIllegalLength	Inflate LZ77 illegal length	R/W	0
18	InflateLZ77ErrorIllegalDistance	Inflate LZ77 illegal distance	R/W	0
17	InflateErrorReadIndexOutOfRange	Read index out of range (bl > 18 or litdist > 319)	R/W	0
16	InflateErrorBfinalError	Bfinal and eob, but not last word	R/W	0
15	InflateErrorInputBufferUnderflow	Input Buffer Underflow	R/W	0
14	InflateErrorNoncompError	Noncomp ~Length doesn't match Length	R/W	0
13	InflateErrorLastDistCodeError	Btype is set to 2'h3	R/W	0
12	InflateErrorLastDistCodeError	Last distance code in preamble has zero length (and there is more than one code)	R/W	0
11	InflateErrorLastLitlenCodeError	Last literal/length code in preamble has zero length	R/W	0
10	InflateErrorLastBLCodeError	Last bit length code in preamble has zero length	R/W	0
9	DeflateErrorZeroLengthReadError	A zero-length Huffman code was read from the LUT		0
8	DeflateErrorLastDistCodeError	Last Huffman code for longest distance code is not all 1s	R/W	0
7	DeflateErrorLastLitlenCodeError	Last Huffman code for longest literal/length code is not all 1s	R/W	0
6	DeflateErrorLastBLCodeError	Last Huffman code for longest bit length code is not all 1s	R/W	0
5	DeflateErrorZeroLengthWriteError	A zero-length Huffman code was written to the LUT	R/W	0
4	InflateStatusNoncompLengthZero	Noncomp byte length specified is 0 (not an error but unlikely to be used)	R/W	0
3	DeflateStatusDistStatsClamp	Distance statistics have clamped	R/W	0
2	DeflateStatusLitlenStatsClamp	Literal/length statistics have clamped	R/W	0
1	DeflateStatusOverflow	Literal/length or distance code length overflow (not an error because codes are fixed)	R/W	0
0	DeflateStatusBLOverflow	BL code length overflow (not an error since we switch to predefined table)	R/W	0

Writing a specific bit to a '1' clears the interrupt

15.5.14 INTERRUPT_MASK

Reg Address Offset: 0x24

Register default value Write 0 to mask

Bits	Name	Description	R/W	Reset
31:29	<i>Reserved</i>			
28:0	Interrupt Mask	Writing a specific bit to a “0” will mask out the corresponding interrupt bit in INTERRUPT_VEC from being used to generate an interrupt to the PIC.	R/W	0

15.5.15 FREE_DESC_THRESHOLD

Reg Address Offset: 0x28

Register default value 0xfffff0000

Bits	Name	Description	R/W	Reset
31:16	TimerInitVal	Should be set to 0xffff	R/W	65535
15:0	Free Descriptor FIFO Threshold	Generates the FreeFifoBelowThreshold when total number of free descriptors is below this threshold. This is a coarse threshold, so it may be off by up to 8 descriptors.	R/W	0

15.5.16 DESC_FIFO_WORD_COUNTS

Reg Address Offset: 0x2C

Register default value

Bits	Name	Description	R/W	Reset
31:11	<i>Reserved</i>			
10:8	MsgOut FIFO	Number of descriptors in egress send message fifo	R	0
7:4	MsgIn FIFO	Number of descriptors in the incoming message fifo	R	0
3:0	FreeIn FIFO	Number of pointers in the On-chip Free In FIFO	R	0

15.5.17 RESET_REG

Reg Address Offset: 0x30
Register default value 0x0

Bits	Name	Description	R/W	Reset
31:8	Reserved			
7	Abort On OCP Error	In case of Double Bit ECC or Address Error, then stop the Cmp Engine from processing any more messages and wait for software action which may include soft reset.	R/W	1
6	Back2BackAllow	Reserved (set to 0)	R/W	0
5	Flush Complete	Message flushing complete	R/W	0
4	Flush Incoming Messages	Before Soft Reset, all incoming messages need to be flushed by setting this bit.	R/W	0
3	DMAChannel2AbortInProgress	In case "Abort On OCP Error" is set, and an ECC occurs, the DMA transactions are automatically aborted. Software should wait for these 3 bits to be cleared before resetting CmpEngine.	R/W	0
2	DMAChannel1AbortInProgress		R/W	0
1	DMAChannel0AbortInProgress		R/W	0
0	Reset Compression Engine	Soft Reset to Cmp Engine. All register values are maintained.	R/W	0

Automatic Abort happens whenever any DMA channel gets data errors. The corresponding interrupt bit also gets set.

Software must read the DMAChannel[0-2]AbortInProgress bits and wait till they are zero, and then Reset the Compression Engine.

There are two available reset sequences:

1. Soft Reset Sequence:

```
write RESET_REG 0x10
poll RESET_REG {
    when RESET_REG[5] = 1 then exit from poll loop
}
write RESET_REG 0x1;
write RESET_REG 0x0;
```

2. Reset Sequence after Data Error:

Data Error (like un-correctable ECC) on any of the DMA channels should set the interrupt. The interrupt sub-routine should follow the following steps:

```
poll RESET_REG {
    when (RESET_REG & 0xE) = 0 then exit from poll loop
}
```

then follow the same software sequence as in 1.

15.5.18 ERROR_RESET_MASK

Reg Address Offset: 0x34
Register default value: 23'h0e7c00

Bits	Name	Description	R/W	Reset
31:29	Reserved		R/W	0
28	DmaChannel2Error		R/W	0
27	DmaChannel1Error		R/W	0
26	DmaChannel0Error		R/W	0
25	FreeFifoBelowThreshold	Automatic Abort happens whenever any DMA channel gets data errors. The corresponding interrupt bit also gets set.	R/W	0
24	InflateHuffmanInitDone	Set the Mask to reset Cmp Engine in case any of the selected errors occur. The Mask by default is set to all Inflate data errors.	R/W	0
23	DeflateHuffmanInitDone		R/W	0
22	InflateLZ77ErrorTimerInterrupt		R/W	1
21	InflateLZ77ErrorLengthError		R/W	1
20	InflateLZ77ErrorDistanceError		R/W	1
19	InflateLZ77ErrorIllegalLength		R/W	1
18	InflateLZ77ErrorIllegalDistance		R/W	1
17	InflateErrorReadIndexOutOfRange		R/W	1
16	InflateErrorBfinalError		R/W	1
15	InflateErrorInputBufferUnderflow		R/W	1
14	InflateErrorNoncompError		R/W	1
13	InflateErrorBtypeError		R/W	1
12	InflateErrorLastDistCodeError		R/W	1
11	InflateErrorLastLitlenCodeError		R/W	1
10	InflateErrorLastBLCodeError		R/W	1
9	DeflateErrorZeroLengthReadError		R/W	0
8	DeflateErrorLastDistCodeError		R/W	0
7	DeflateErrorLastLitlenCodeError		R/W	0
6	DeflateErrorLastBLCodeError		R/W	0
5	DeflateErrorZeroLengthWriteError		R/W	0
4	InflateStatusNoncompLengthZero		R/W	0
3	DeflateStatusDistStatsClamp		R/W	0
2	DeflateStatusLitlenStatsClamp		R/W	0
1	DeflateStatusOverflow		R/W	0
0	DeflateStatusBLOverflow		R/W	0

15.5.19 ADDRESS_OF_READ_ERROR_LIST0

Reg Address Offset: 0x38

Register default value

Bits	Name	Description	R/W	Reset
31:0	Address[31:0]	In case OCP Error is detected, this register will hold the lower 32 bits of the corresponding message.	R/W	0

15.5.20 ADDRESS_OF_READ_ERROR_LIST1

Reg Address Offset: 0x3C

Register default value

Bits	Name	Description	R/W	Reset
31:0	Address[63:32]	In case OCP Error is detected, this register will hold the upper 32 bits of the corresponding message.	R/W	0

NETLOGIC
CONFIDENTIAL



Chapter 16 SGMII and RGMII Interface

16.1 Introduction

This chapter covers the SGMII and RGMII Interfaces. It provides the following information:

- Overview of the [Capabilities and Functions](#)
- [Theory of Operation](#)
 - description of interface position within the chip architecture
 - operational description
 - device modes/states/access model
 - interrupt handling
 - statistics monitoring
- [Programming Model](#)
 - startup/initialization
 - bringup requirements
- Interface external signals
- [SGMII and RGMII Registers](#)
 - master list of general registers
 - register descriptions
 - master list of statistics registers
 - statistics register descriptions

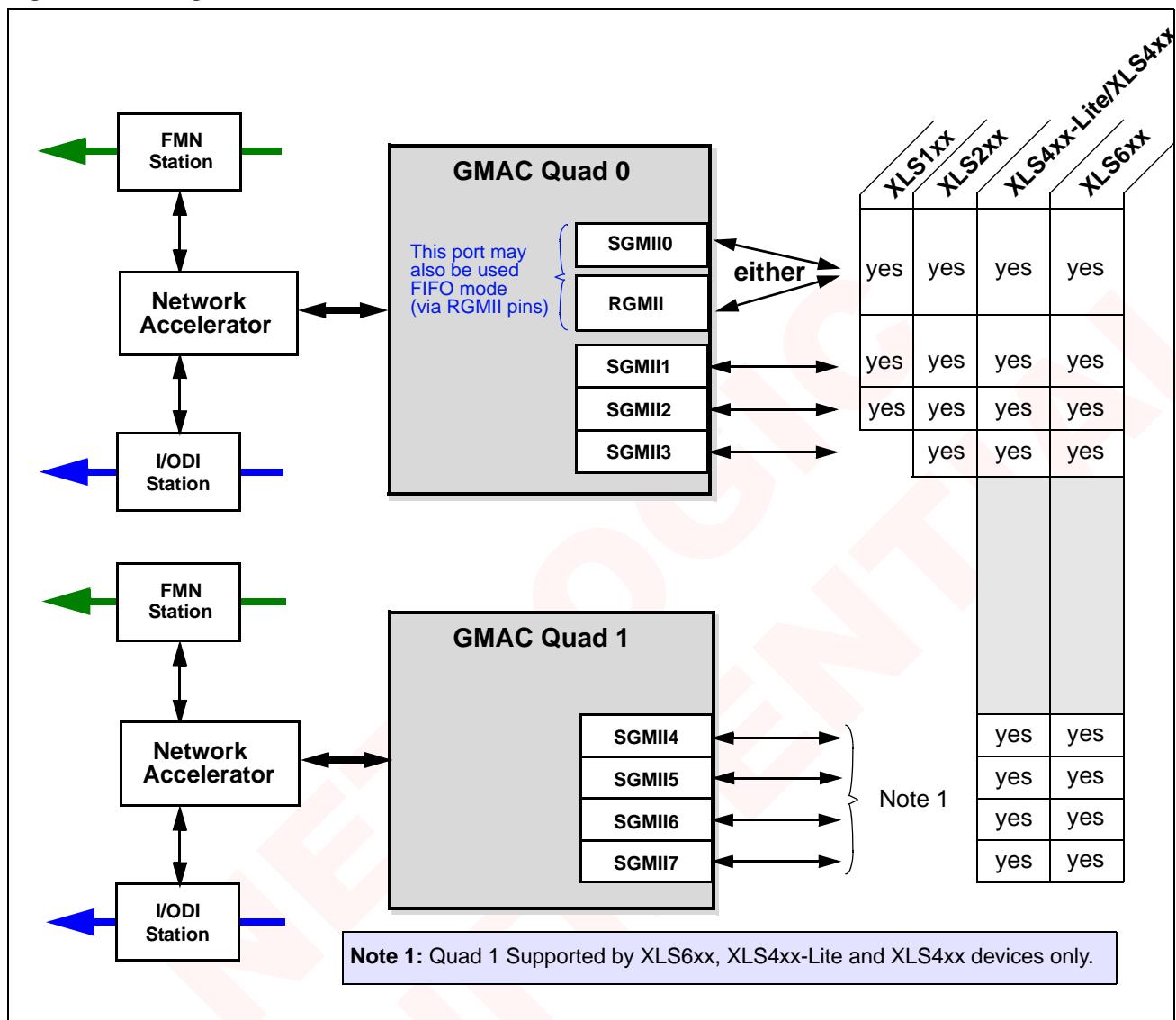
16.1.1 Optional XAUI Interface

The XLS616, XLS608 and XLS416 offer two optional XAUI links that share signal pins with the SGMII quad interfaces. The XLS408 offers one optional XAUI link that shares signal pins with the SGMII_4-7 quad interface. At power-up time, either or both XAUI links can be configured to replace the SGMII quads as follows:

- The XAUI_0 port can be configured to replace the SGMII_0-3 quad by strapping the IO_AD[7] pin high at power-up/reset. When strapped low, the SGMII_0-3 quad is selected.
- The XAUI_1 port can be configured to replace the SGMII_4-7 quad by strapping the IO_AD[6] pin high at power-up/reset. When strapped low, the SGMII_4-7 quad is selected.

Once a XAUI port is selected, all of the SGMII signal pins in the quad are replaced with the equivalent XAUI signal pins. See XLS Data Book for details.

[Figure 16-1](#) shows the high-level position of the interface in the architecture.

Figure 16-1. High-Level View of GMAC Interface in XLS Architecture

16.2 Capabilities and Functions

16.2.1 Key Capabilities Summary

SGMII:

- Up to eight ports, depending on device type
- SGMII ports support Revision 1.7

RGMII:

The ports have the following features:

- IEEE 802.3, 802.3u, 802.3x, 802.3z, 802.3ac Compliant
- 10/100/1000-Mbps Operation
- MAC Control Sub-layer with PAUSE, extended opcode support
- MII Management with suppressed preamble, sequential cycle, variable clock features
- One RGMII port with 4-bit transmit, 4-bit receive engine optimized for size, reduced latency
- Internal software-accessible module supports specific resets and enables
- Per frame and default programmable padding and FCS appending
- Supports huge frames
- Full statistics collection for both transmit and receive
- Supports Draft Version 2.0 of the RGMII Specification
- Optional Half-Duplex Back-Pressure (10/100 Mbps Only)
- Support for externally-generated proprietary control frames
- The RGMII port supports FIFO mode operation: NetLogic custom functionality to aid on-board interconnection

The GMAC functionality is full-featured and capable of full-duplex operation at 1000 Megabits per second, and also half-duplex and full-duplex operation at 10 and 100 Mbps. The GMAC supports Control frames, including PAUSE frames.

The Interface communicates through the Network Accelerator with the internal architecture, including interrupts. Details can be found in the Network Accelerator chapter.

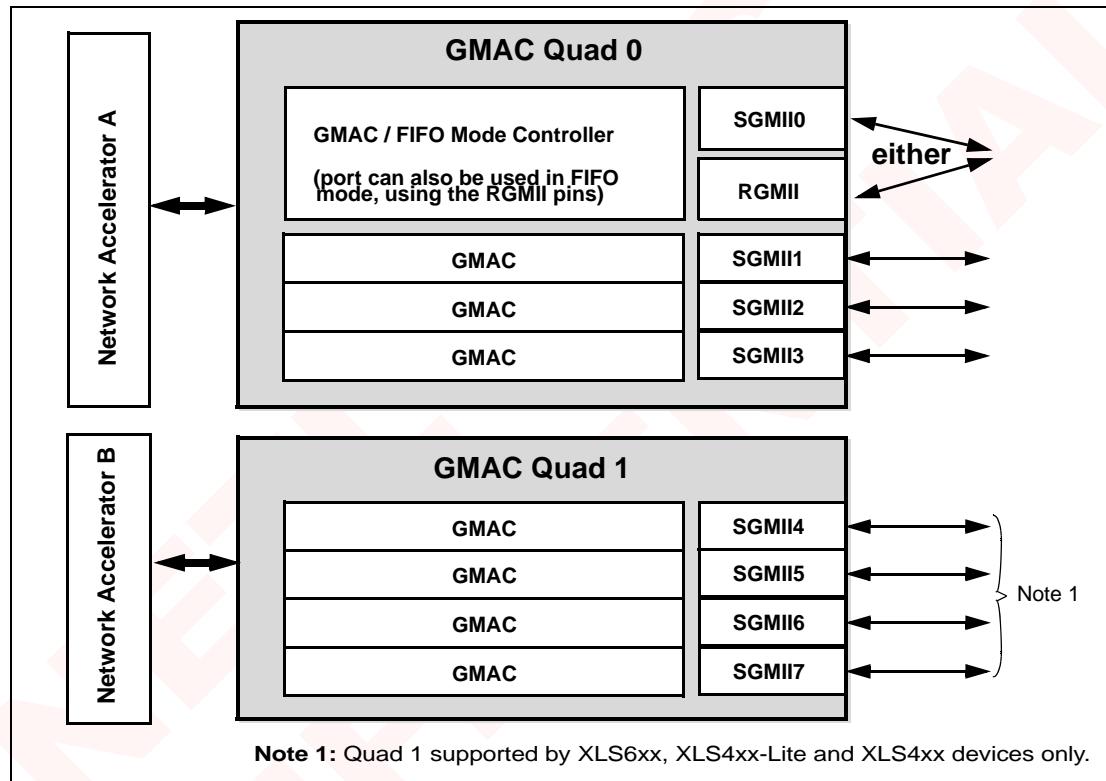
In addition, the interface provides controllable packet filter functions. These are described in the section covering the [MAC_FILTER_CONFIG](#) register (0x5C), and illustrated there.

When GMAC 0 is configured for RGMII operation, the MAC can alternatively be configured to operate in FIFO mode with backpressure. FIFO mode is used for communicating with other chips on the same board without the need of a PHY. The 4-bit FIFOs are clocked at 125 MHz DDR supporting up to 1 Gbps per interface.

16.3 Theory of Operation

This section discusses architecture and operating details of the GMAC and FIFO Mode Controller. As shown in [Figure 16-2](#), XLS Processors support up to 8 GMACs, depending on model. GMACs are grouped in “quads” of four interfaces each; each quad has its own Network Accelerator. Thus, in the XLS6xx, XLS4xx-Lite and XLS4xx with 8 GMACs, there are two GMAC quads of four interfaces each, and two Network Accelerators. The XLS2xx devices have a single GMAC quad with four interfaces and one Network Accelerator for those four ports. The XLS1xx devices have a three GMAC ports and one Network Accelerator for those three ports.

Figure 16-2. GMAC Architecture



16.3.1 Position of Interface Within the System Architecture

Key points for understanding the system-level architectural model are:

- The GMAC interacts with the Network Accelerator block for access to the XLS internal architecture.
- The GMAC registers lie in a sub-group within the Network Accelerator register addressing region.
- A module outside the GMAC monitors statistics for traffic between the interface and the Network Accelerator, and its registers are accessed through the Network Accelerator.
- Interrupts for the interface are routed through the Network Accelerator to the PIC (Programmable Interrupt Controller)

The internal-facing side of the Network Accelerator is integrated into the XLS architecture for data transport functions. The interface's external side complies with IEEE 802.3, 802.3U, 802.3X, 802.3Z, 802.3AC standards and supports Draft Version 2.0 of the RGMII Specification and Revision 1.7 for SGMII Interfaces

16.3.2 Overview of SGMII/RGMII Port GMAC Mode Operation

The system interface presents frames to be transmitted by the GMAC. Frames may or may not be padded and may or may not contain the Frame Check Sequence (FCS) field. Depending upon the programmed configuration internal to the GMAC and per-packet overrides, the GMAC may pad the frame and/or append a valid FCS. In half-duplex mode, the GMAC adheres to the Carrier Sense Multiple Access / Collision Detect (CSMA/CD) access method as defined in IEEE 802.3 and its several supplements including IEEE 802.3u. In full-duplex mode, the GMAC follows IEEE 802.3x, which says to ignore carrier and collisions.

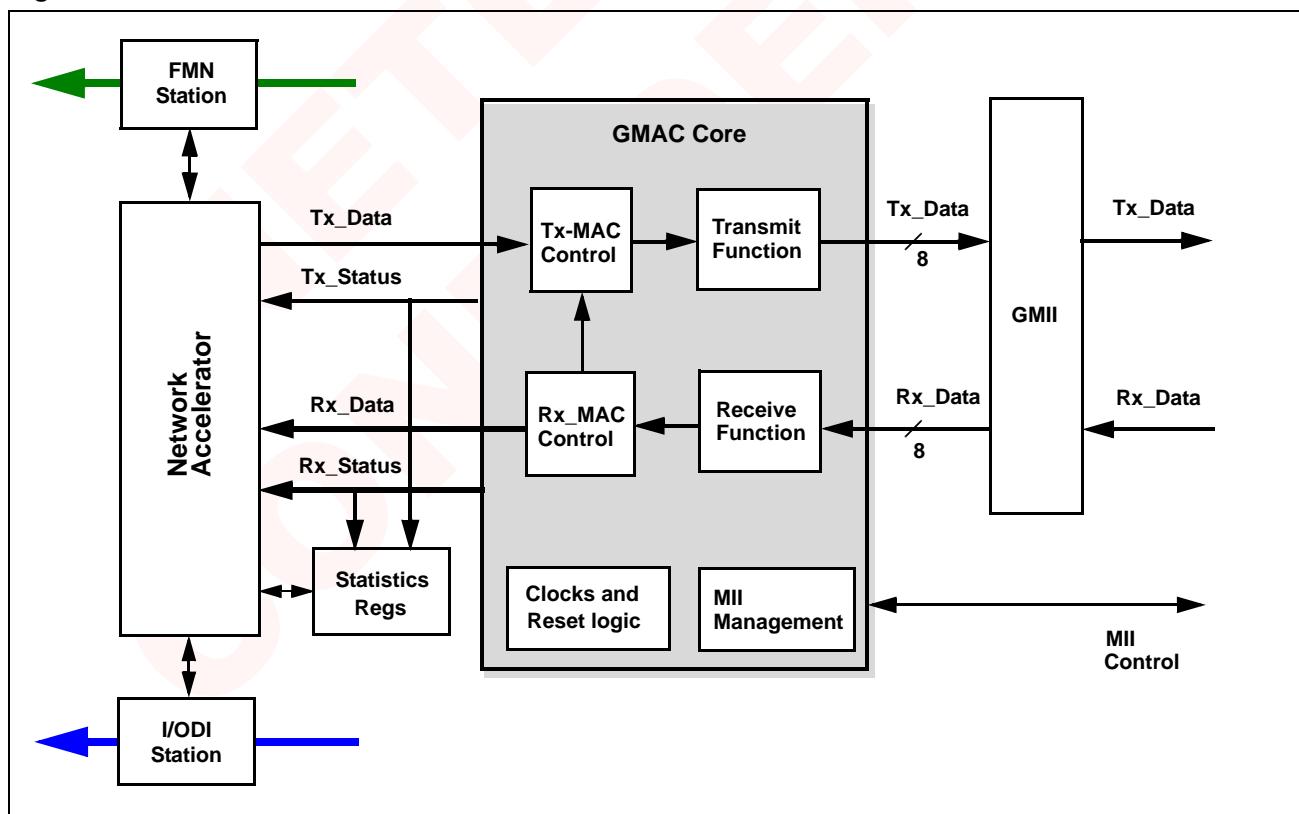
Note: Software must ensure that the transmit packet size equals or exceeds 9 bytes.

If a packet smaller than 9 bytes must be sent, the Free_Back descriptor of the small packet must be received by the CPU before sending any subsequent packet to the GMAC.

The external Physical layer device (PHY) presents receive packets to the GMAC. The GMAC scans the preamble looking for the Start Frame Delimiter (SFD). When found, the preamble and SFD are stripped and the frame is passed to the system. Following each frame reception, a Receive Statistics Vector is output for frame filtering and statistics collection. The GMAC also outputs 9-bits of the Cyclic Redundancy Check sequence of the Destination Address (DA) for hash table lookup and filtering common in many Network Interface Card (NIC) applications.

The XLS Processor also contains the GMAC Control Sublayer, which adheres to IEEE 802.3x and provides support for Control frames. The GMAC supports both Symmetric and Asymmetric Flow Control. At present, only PAUSE Control frames have been defined. Via optional configuration bits, the GMAC can support other Control frames.

Figure 16-3. Internal Details of SGMII/RGMII Port in GMAC Mode



The following sections describes operation of the major sub-blocks in the GMAC and the related statistics registers:

16.3.2.1 Transmit Function (TFN)

The Transmit Functional Block accepts frames from the system transmit interface, prepends a seven-byte preamble and SFD, and outputs the packet across the interface to the outside. After the TFN waits the programmed IPG, the preamble is output.

Reduced Gigabit Media Independent Interface (RGMII)

For the RGMII port, the XLS 10/100/1000 MAC outputs packets via the RGMII. Half-Duplex is only supported for 10 Mbps and 100 Mbps operation.

16.3.2.2 Transmit MAC Control (TMC)

The TMC is responsible for the multiplexing of normal transmit frames and Control frames requested by the system. Control frames are described in IEEE Standard 802.3x.

16.3.2.3 Receive Function (RFN)

The RFN accepts packets via the SGMII/RGMII external interface, extracts frames and presents these to the system. The preamble and SFD fields are removed from the packet by the RFN. The RFN calculates the CRC of the received frame to be used in checking the Frame Check Sequence field.

16.3.2.4 Receive MAC Control (RMC)

It is the responsibility of the RMC to detect Control frames. The RFN passes receive frames to the RMC module. Frames are examined to determine if they are Control frames, and if so whether or not they are PAUSE frames.

Control Frame Detection

Valid length properly received frames that contain the EtherType 0x8808 are considered Control frames. The length must be in the range of 64 octets to 1518 octets inclusive. The frame must also not contain any errors and have a valid CRC.

PAUSE Control Frame Detection

Once a frame has been determined to be a Control frame, two additional fields are checked. They are the DA and the Opcode fields. If the DA is either the *Reserved* multicast address used by PAUSE (01-80-c2-00-00-01) or the station's unique address, and the Opcode is 0x0001, the Control frame is considered a PAUSE Control frame. For statistics purposes, only the Opcode is examined. For PAUSE Flow Control, both the DA and Opcode fields are examined.

PAUSE Flow Control

After a PAUSE Control frame is received the parameter field is loaded into the pause timer. This 16-bit down counter decrements every pause quanta — a speed-independent constant of 64 byte-times (unlike slot time). Whenever the pause time counter is non-zero the MAC is considered to be paused and no new data transmit frames will be sent. When either the pause time counter counts down to 0, or a subsequent PAUSE Flow Control with a zero-value parameter field is received, normal frame transmissions resume.

Station Address Detection

The four 48-bit programmed station address values per Mac are compared to each receive frame's destination address, and appropriate action taken.

16.3.2.5**MII Management (MGT)**

Control and status to and from the PHY is provided via the two-wire MII Management Interface described in IEEE 802.3u. The MGT sub-module provides this functionality.

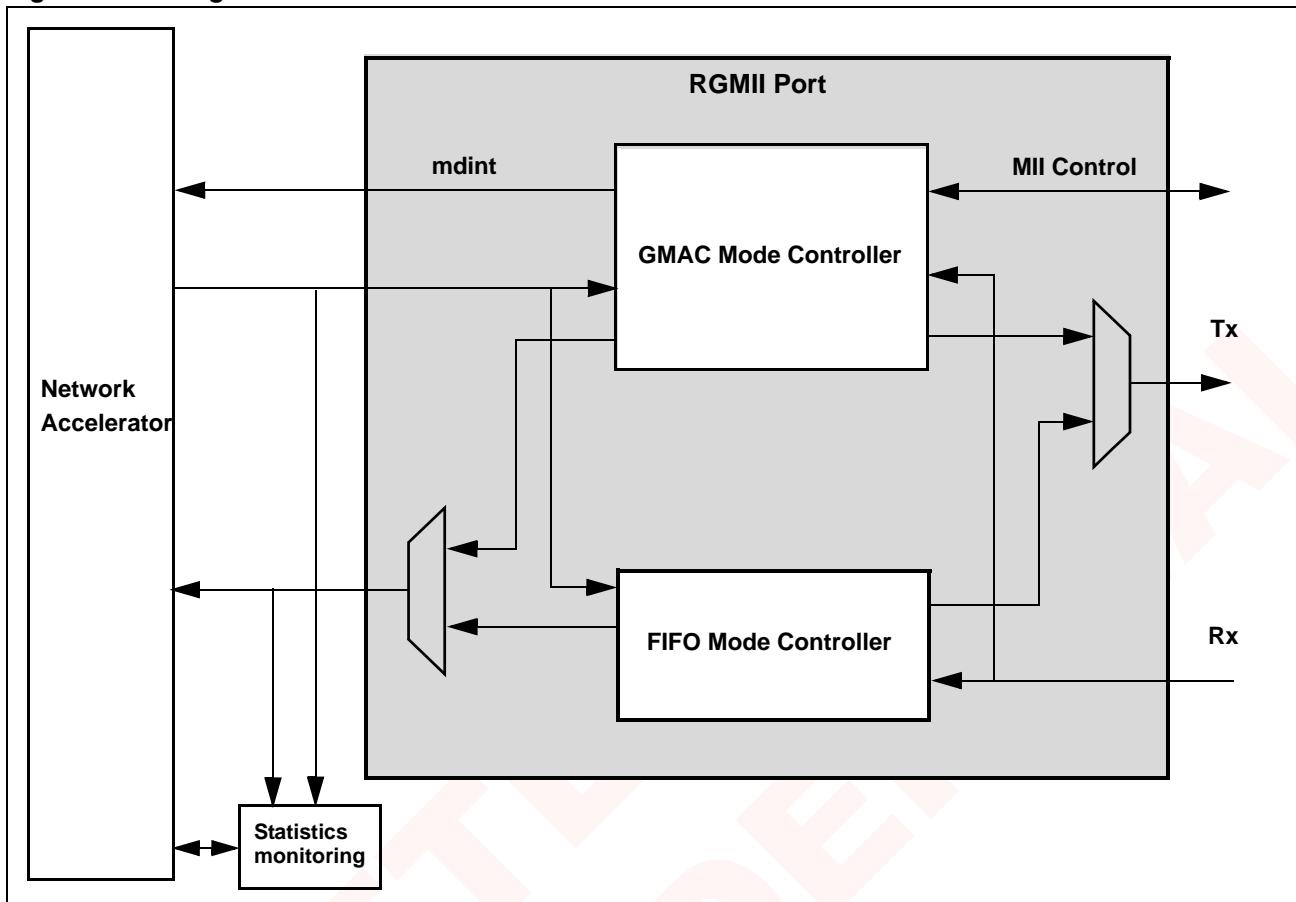
16.3.2.6**Statistics Monitoring**

The XLS includes capability for monitoring traffic between each SGMII/RGMII port and its Network Accelerator connection. Each of the four SGMII ports and the RGMII port has its own dedicated set of statistics registers. These registers are described in the register section of this chapter.

16.3.3**RGMII Operation**

For Port SA/RA, selection of SGMII or RGMII mode is set in the Network Accelerator for GMAC Block A. [Figure 16-4](#) shows a high-level view of the internals of the port when configured for RGMII operation.

Additionally, when operating in RGMII mode, the port may be configured for either GMAC or FIFO mode operation.

Figure 16-4. High-Level View of XLS RGMII Port A

16.3.4 Overview of RGMII Mode FIFO Mode Operation

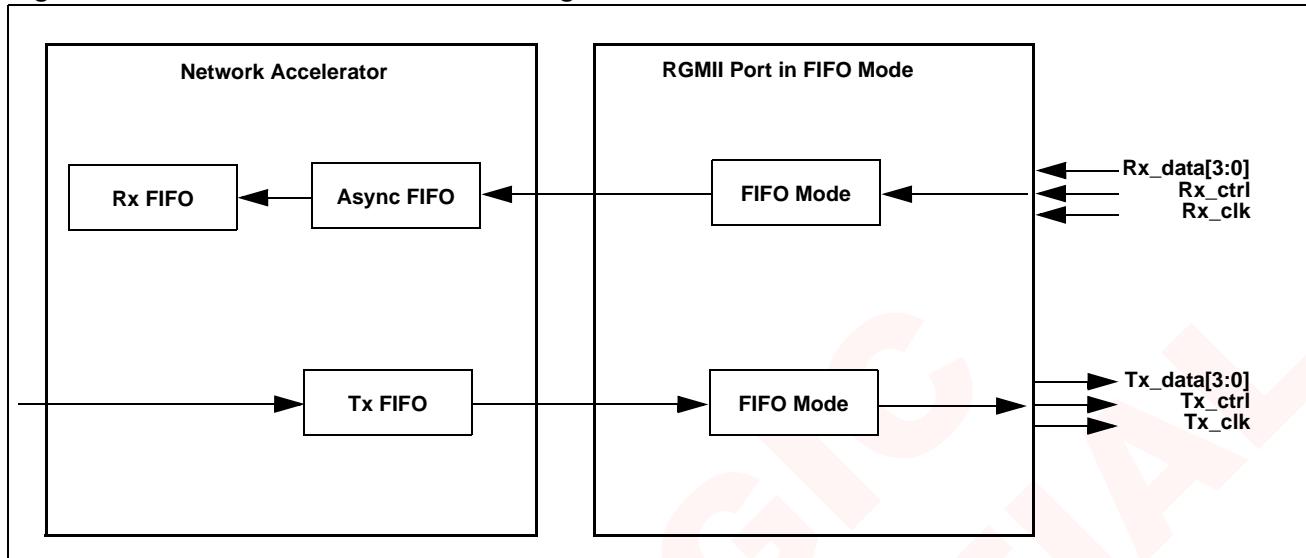
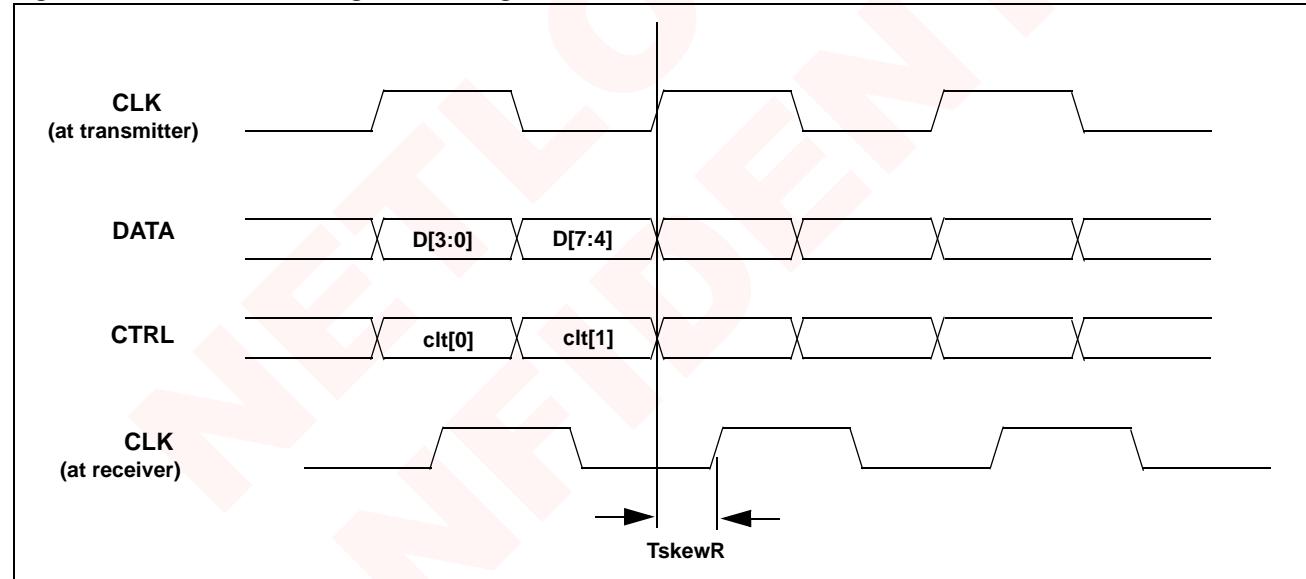
When Port RA is configured for RGMII Mode, it can be configured to be in FIFO Mode operation. For FIFO Mode operation, the electrical specifications of the data, CTRL and CLK are identical to the RGMII standard - data is 4 bits at 125-Mhz DDR. This supports data rate up to 1 Gbps per port.

The logical specifications are modified.

Highlights:

- Low latency flow control for loss-less operation
- Support mid-frame idle cycles
- Reduced inter-frame gap

Identical to Ethernet, packet data is protected by a per-frame CRC at the end of each frame.

Figure 16-5. Port RA FIFO Mode Block Diagram**Figure 16-6. Bus Encoding and Timing**

The transmitter drives D[3:0] and clt[0] at positive-edge of CLK, and drives D[7:4] and CLT[1] at negative-edge of CLK. CLK at receiver is skewed relative to the CLK at transmitter. This allows the receiver to sample DATA[3:0] and CTRL[0] at positive-edge of the received CLK. This skew is achieved on purpose by adding delay to the CLK trace on the PC board. The required {Min/Typical/Max} value of this skew TskewR is 1/1.8/2.6 ns. Ref: RGMII 2.0 specification.

Table 16-1. States

State	ctl[0]	ctl[1]	DATA
SOP	1	0	D[7:0] is first word of packet.
EOP	0	1	D[7:0] is last word of packet.

Table 16-1. States (continued)

State	ctl[0]	ctl[1]	DATA
Valid	1	1	D[7:0] is middle of a packet
Idle	0	0	D[7:0] gives the BP stats <ul style="list-style-type: none"> • Back pressure off (BP-off, i.e. start) if D is 8'b0000 0001, • Back pressure on (BP-on, i.e. stop) if data[1] is D is 8'b0000 0010 any other code are disallowed. • it is allowed and necessary to transmit idle in the middle of a packet

16.3.4.1**CRC Handling**

Each packet is trailed by a 4-byte CRC. The last byte of the CRC is transmitted at the same time with the EOP character. It is not allowed to have idle character during CRC transmission.

- Example 1

	idl	sop	packet data										CRC				eop	idl		
Ctrl	0	0	1	0	1	1	1	1	1	1	1	1	1	1	1	1	0	1	0	0
Data	B	B	N	N	N	N	N	N	N	N	N	N	N	C	C	C	C	B	B	

- Nx = the x-th nibble of packet
- BpBp = back pressure status. Note that at idle cycle we must transmit backpressure status.
- The packet data is bracketed by the SOP and EOP.
- Note that there is valid packet data at SOP and EOP.

- Example 2

It is allowed to have idles mid packet, as shown:

	idl	sop	packet data					idl	data				CRC				eop	idl		
Ctrl	0	0	1	0	1	1	1	1	0	0	1	1	1	1	1	1	0	1	0	0
Data	B	B	N	N	N	N	N	N	B	B	N	N	N	N	C	C	C	B	B	

16.3.4.2**Min Inter-frame Gap and Min Frame Size**

- The transmitter is expected to insert a minimum 4-byte IPG and append a 4-byte CRC to each packet.
- There must not be any idle in the last 4 bytes, i.e. CRC of a packet.
- The EOP CTRL code will accompany the last cycle of the CRC word on the data pins.
- If the link partner transmits packet smaller than Min IPG to XLS, the behavior is unpredictable.
- The XLS transmitter is designed to transmit packet with at least 4-byte IPG and append a CRC.
- In FIFO Mode, the GMAC transmit logic will always append CRC. Unlike GMAC mode, this is not a configurable option.

- Min frame size is 32bytes. Packets smaller than Min frame size might cause unpredictable behavior.

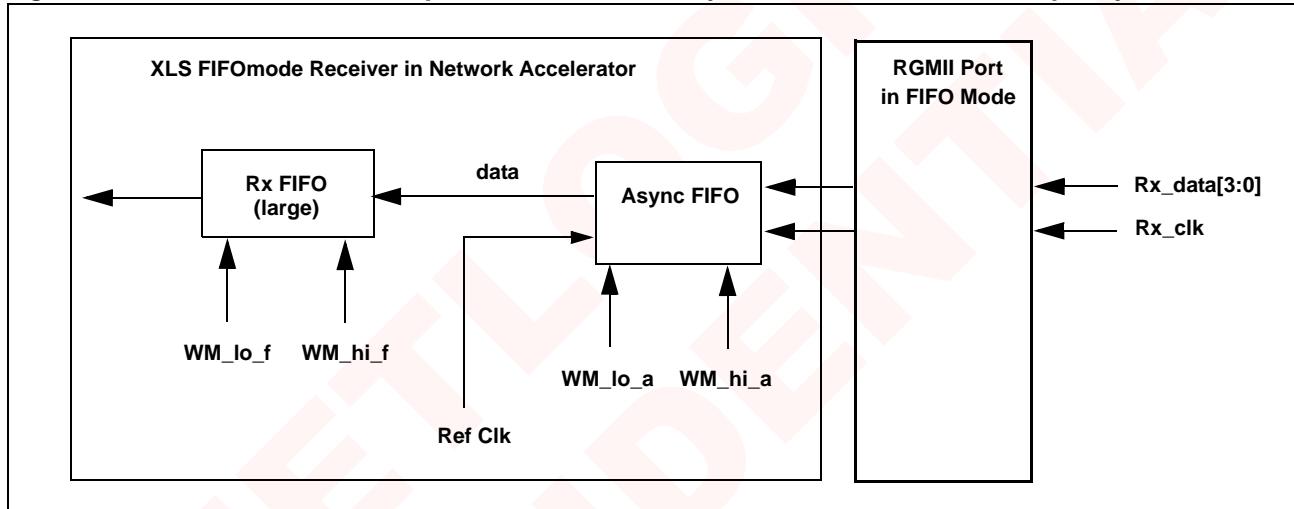
16.3.4.3 CRC Addition/Removal in FIFO Mode

When the XLS receives a FIFO mode packet, it will check the CRC. Then the CRC will be removed and the packet is DMA'd to the memory system. Thus, when software sees this packet, it does not have the CRC. If a packet has the wrong CRC, the packet descriptor will get flagged 'error'. The SW will see this flag, and treat this packet accordingly.

Unlike the ethernet MAC mode, where adding CRC is an option, when the XLS transmits a FIFO mode packet, it will always add the CRC. So SW should not append the CRC when it forms a packet for transmission.

16.3.4.4 Generation of Backpressure

Figure 16-7. Generation of Backpressure Based on Async FIFO and RxFIFO Occupancy



The Async FIFO could build up due to CLK tolerance between RefClk and RxClk. To handle this, when the Async FIFO builds up and it passes a threshold (WM_hi_a), a BP-ON signal will be sent to link partner D. D will transmit a series of idles, which are not written into the Async FIFO. This allows the Async FIFO to drain.

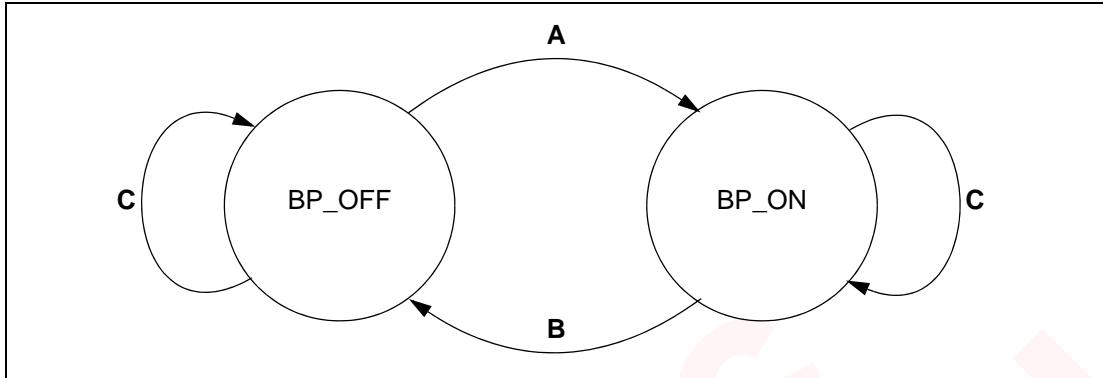
When the Async FIFO has drained below WM_lo_a, an BP-OFF signal will be sent, assuming that the RxFIFO is not generating a BP-ON.

WM_hi_a and WM_lo_a are hard-wired and not programmable.

WM_hi_f and WM_lo_f are programmable values, controlled by the FIFOModeCtrl (offset 0x0af) in the Network Accelerator.

16.3.4.5 Pseudocode for Directed Graph

Figure 16-8 shows the state diagram that corresponds to the pseudo-code shown below. The letters in the figure correspond to the marked pseudo-code statements.

Figure 16-8. State Diagram for Backpressure

```

current_state = current state of back pressure, either BP_ON or BP_OFF
  BP_ON means the link partner is not allowed to send. i.e. STOP
  BP_OFF means the link partner is allowed to send. i.e. Resume
  
```

```

next_state = next state of back pressure
fifo_occupancy = current occupancy of the FIFO
positive-edge_clk = true if a clock signal positive edge is detected
  
```

```

A   if (current_state == BP_OFF &&
      (AsyncFifo_occupancy>= Wm_Hi_a || Rxfifo_occupancy>= Wm_Hi_f) {
      next_state = BP_ON;
}
B   else if (current_state == BP_ON &&
      ((AsyncFifo_occupancy < Wm_Lo_a && Rxfifo_occupancy < Wm_lo_f) {
      next_state = BP_OFF;
}
C   else {
      next_state = current_state;
}
if (positive-edge_clk) {
  if (current_state != next_state) {
    if (packet_is_currently_in_transmission) {
      stop midpacket and send next_state in an idle character;
      resume sending packet data afterwards;
    } else {
      send next_state in idle character;
    }
    current_state = next_state;
}
}   
```

16.3.4.6 Periodic BP Transmission

The XLS transmitter will transmit a gratuitous idle character once in X cycles, to refresh the link partner's state. This happens even if the BP state has not changed.

The periodic X is selectable from 64, 512, 4K or infinite.

The infinite setting makes this an optional feature.

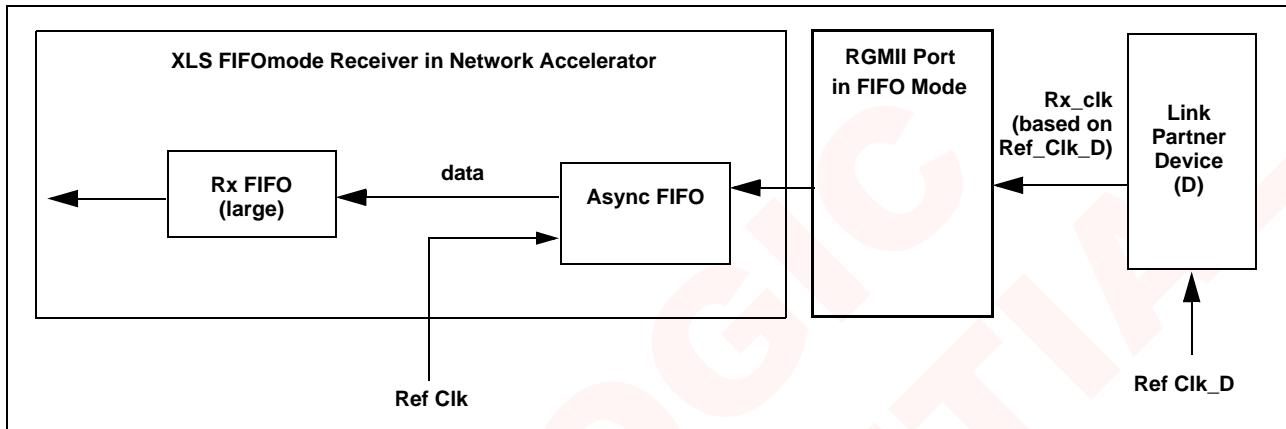
As a result, the XLS receiver does not expect its link-partner to periodically transmit BP.

16.3.4.7 Responding to Backpressure State

When receiving a BP-on character, the XLS transmitter stops mid-packet and starts transmitting idle characters. There is an exception. If the XLS transmitter is transmitting the CRC, it will finish the current packet and stop.

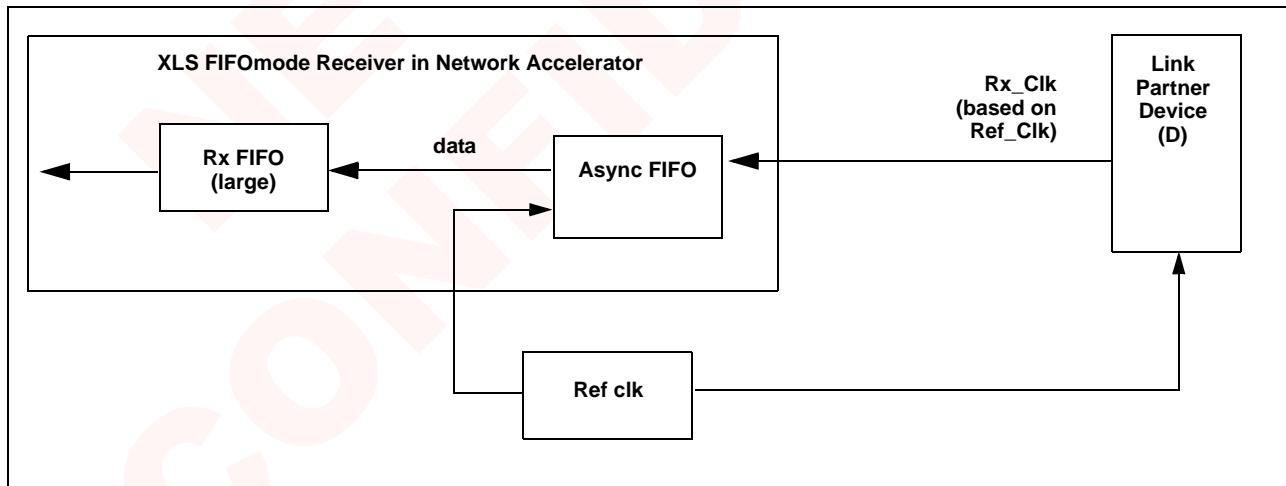
The XLS receiver expects its link partner to have the same behavior.

Figure 16-9. Async FIFO and Clocking Scheme in FIFO Mode — FIFO Mode (Async)



- Since the Async FIFO generates BP-on if it grows too large, there is no limit on max burst. Max burst is the max number of data characters between idles.
- Note that idles are not written in the Async FIFO.
 - so the BP data are read in the RxClk domain, before writing into Async FIFO.
 - The BP data are passed to the RefClk domain, using a separate set of Async flops

Figure 16-10. Async FIFO and Clocking Scheme in FIFO Mode — FIFO mode (sync)



- In this case, the Async FIFO will have at most one (or two) entries in it.
- The Async FIFO will not grow, since the RxClk has same frequency as the Refclk.
- There is no limit in the Max burst length.

16.4 Programming Model

This section discusses programmatic set up for operation of the interface, and programmatic access to the interface. It also discusses statistics monitoring for traffic between the RGMII interface and its companion Network Accelerator.

There are two main areas in which setup must be done:

- XLS system-level setup
- Interface-level setup

16.4.1 System Setup for Startup and Initialization

The inward side of the interface is tightly integrated into the XLS processor's I/O distributed interconnect, through its FMN and I/O DI stations. Setup areas are in addressing, data transport credits, interrupts, and statistics monitoring.

16.4.1.1 System Bridge Controller (SBC) Setup for Address Access

The SBC resides in the XLS's Memory and Distributed Interconnect Hub. (See Chapter 8, Memory and I/O Access, for details.) The Hub handles device-internal address mappings.

Access to SGMII/RGMII registers depends on the base address register setting in the SBC: Additionally, register access for certain register groups, as detailed later, requires special consideration.

The SBC's XLS_IO_BAR register defines the base address for the entire Peripheral and I/O Configuration region in which all internal peripheral configuration header registers and control and status registers lie.

Each SGMII/RGMII group:

- GMAC Block A (SGMII0/RGMII through SGMII3), and
- GMAC Block B (SGMII4 through SGMII7)

has its own group base address setting the base address for access to that port's registers. These group addresses lie at fixed offsets from the base address.

Each port has its own set of statistics registers, and each set is accessed from that port's group base address.

16.4.1.2 Credits Setup for Network Accelerator

Each SGMII/RGMII group's data is sent internally through a Network Accelerator for that block to an I/O DI station for distribution to other XLS system elements as necessary.

The NA's bandwidth usage within the XLS is controlled by the system I/O credits mechanism. Thus, optimizing bandwidth usage of the SGMII/RGMII interface within the XLS's internal data transport mechanism involves understanding I/O credits. See Fast Messaging Network chapter discussions of credits and buckets.

16.4.1.3 Interrupt Setup

Interrupt handling for the interfaces at the system level is routed through the Network Accelerator to the XLS's PIC (Programmable Interrupt Controller). This involves setup in several areas:

- 1) Interrupt Redirection Table (IRT) in the PIC. See discussions in Chapter 12, Programmable Interrupt Controller, of this manual, and the Networking Accelerator chapter.
- 2) the EIRR for each CPU/thread that will interact with the Network Accelerator. See this manual's discussion of EIRR registers in CPUs.

16.4.2 Interface Setup for Startup and Initialization

SGMII/RGMII internal registers must be configured as needed.

16.4.2.1 Reset Handling

Although all internal logic of the interface is reset upon system reset, some internal modules can be reset independently by means of bits in the interface's register set. (See register descriptions.) This can be useful in debugging or other circumstances.

16.4.3 Interrupts

As mentioned before, interrupt handling for the interfaces is routed through the Network Accelerator to the XLS's PIC (Programmable Interrupt Controller). This involves setup in several areas:

- 1) Interrupt Redirection Table (IRT) in the PIC. See discussions in Chapter 12, Programmable Interrupt Controller, of this manual, and the Networking Accelerator chapter.
- 2) the EIRR for each CPU/thread that will interact with the Network Accelerator. See discussion of EIRR registers in CPUs.

Each SGMII/RGMII interface can generate a variety of interrupts. The interrupts from an interface are aggregated in its companion Network Accelerator to create a single interrupt that goes to the PIC. When a thread is given the interrupt, it must then check the appropriate IntReg register at 0x0A6 in the appropriate Network Accelerator to determine what interrupt condition actually occurred. The software must clear the interface's interrupt bit in the PIC, and must also clear the appropriate interrupt bit in the correct Network Accelerator. In some system designs, it may also be necessary for the software to interact with an interrupt source external to the XLS, in a PHY device.

16.4.4 Statistics Monitoring

Statistics values for traffic between each SGMII/RGMII port and its Network Accelerator can be monitored in the SGMII/RGMII statistics registers.

- Values can be obtained by reading the appropriate statistics monitoring register. The monitoring activity is enabled through a register in the Network Accelerator for the GMAC Block.
- A read clears a statistics counter register if the 'AutoZ' feature is enabled for all statistics registers. This feature is enabled by setting the AutoZ bit-field in the StatCtrl (0x0A3) register in the companion Network Accelerator for that GMAC block.

For details refer to the Network Accelerator chapter register descriptions.

The interface can generate an interrupt to the PIC when a statistics counter overflows. This is enabled in the Network Accelerator for that GMAC block, and requires setting appropriate masking bits in the SGMII/RGMII's statistics CAM1 and CAM2 registers.

16.5 SGMII and RGMII Registers

16.5.1 Introduction

The following is discussed in this section:

- The interface's GMAC block registers and additional related registers. Each SGMII/RGMII interface (SGMII0/RGMII through SGMII7) within a GMAC block has its own unique set of configuration registers controlling that interface's GMAC.
- In addition, a single set of registers in the companion Network Accelerator for a GMAC block affect flow control for all four SGMII/RGMII interfaces in that block, and control how those interfaces interact with the XLS internal architecture. Though in the Network Accelerator, these registers are covered here.
- Statistics registers for monitoring data between the GMAC and the NA.
- PCS registers are described here.
- Not described in this section:
- IO credits registers for control of bandwidth allocation in the XLS between interfaces. See Fast Messaging Network chapter for the relevant registers.

For proper operation, all appropriate GMAC and NA registers must be configured.

Bits that are not implemented or are *Reserved*, are set to zero.

16.5.1.1 Definition of Terms

The following terms are used in the register descriptions to describe field modifiability type:

Table 16-2. Field Modifiability Codes

Type Code	Meaning	Description
R/O	Read Only	The field contains a static value and cannot be written.
R/W	Read/Write	The field can be modified by software. Note that some fields, where indicated, are modifiable only indirectly, by writing to a shadow register then performing a soft reset.
R/C	Read/Clear	The field can be read to give status information. Bits may be cleared by writing a 1 to them; writes of 0 are ignored.

16.5.1.2 Data Access and Endianness

The local space in the interface is little endian.

16.5.1.3 Register Access

SGMII/RGMII registers are address accessed using the combination of base address for all XLS_IO devices, plus interface group base address or offset, plus individual register offset.

Register offsets are listed with each register in this chapter.

Each SGMII/RGMII interface has a group offset base address as described in Chapter 8 and shown below.

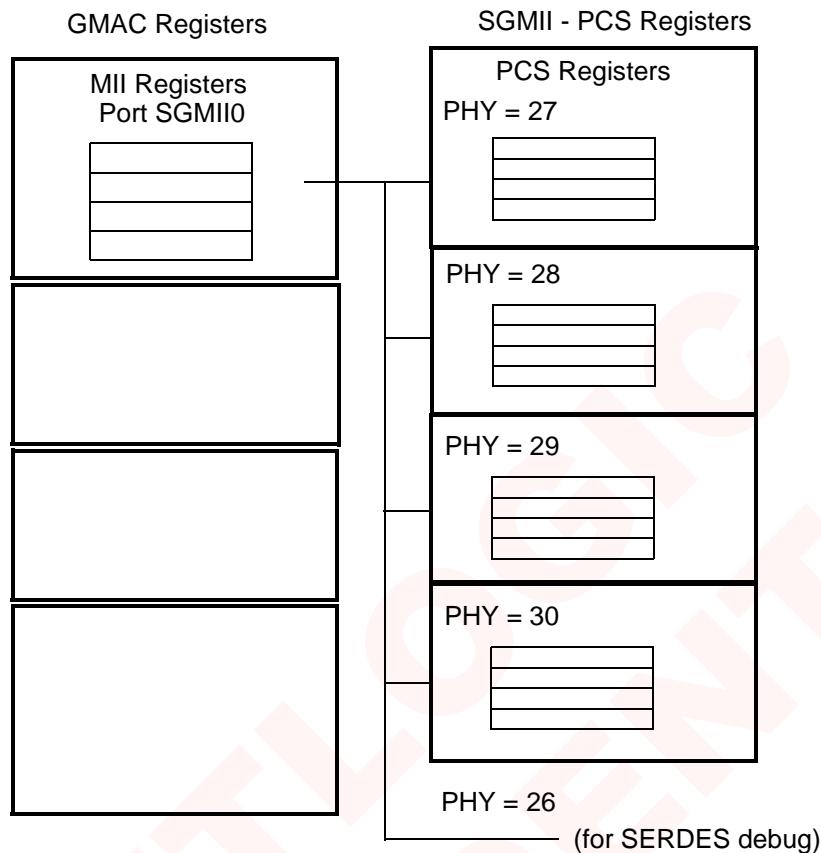
XLS processor models have varying number of SGMII interfaces, and the addresses shown apply only up to the number present in a model.

Table 16-3. Port Register Access

Peripheral	Device Name	Region Offset	Region Offset Address at Reset
SGMII-Interface_0, Port SGMII0/ RGMII-interface, RGMII port	XLS_IO_DEV_GMAC_0	0x0 C000	0x00 1EF0 C000
SGMII-Interface_1, Port SGMII1	XLS_IO_DEV_GMAC_1	0x0 D000	0x00 1EF0 D000
SGMII-Interface_2, Port SGMII2	XLS_IO_DEV_GMAC_2	0x0 E000	0x00 1EF0 E000
SGMII-Interface_3, Port SGMII3	XLS_IO_DEV_GMAC_3	0x0 F000	0x00 1EF0 F000
SGMII-Interface_4, Port SGMII4	XLS_IO_DEV_GMAC_4	0x2 0000	0x00 1EF2 0000
SGMII-Interface_5, Port SGMII5	XLS_IO_DEV_GMAC_5	0x2 1000	0x00 1EF2 1000
SGMII-Interface_6, Port SGMII6	XLS_IO_DEV_GMAC_6	0x2 2000	0x00 1EF2 2000
SGMII-Interface_7, Port SGMII7	XLS_IO_DEV_GMAC_7	0x2 3000	0x00 1EF2 3000

16.5.1.4 PCS Register Access

The SGMII PCS layer registers are accessed indirectly through the MII Management Interface (MDIO Interface) of GMAC Quad 0. The PHY addresses used for the 4 SGMII ports are 27, 28, 29, and 30. Also there is a common set of registers for the SERDES_control which can be accessed with PHY address 26. The board / system designer must guarantee that these PHY addresses are not used by external PHY chips that need to be controlled using the same Management Interface.

Figure 16-11. Accessing PCS Registers in SGMII**16.5.1.5****Access for Network Accelerator Registers for the Interface**

There are two unique aspects to accessing Network Accelerator registers for SGMII/RGMII interfaces.

- 1) **Per-interface registers:** For each SGMII/RGMII interface, there is one set of Network Accelerator registers at offsets *below 0x100* within the addressing space for the interface. That is, the area below 0x100 actually contains both interface registers and Network Accelerator registers. This set of registers in the Network Accelerator are unique for that interface, i.e., each SGMII/RGMII interface (SGMII0/RGMII0 through SGMII7) has a unique set of Network Accelerator registers dedicated to it which reside below 0x100.
- 2) **Per-block registers:** Additionally, there is a set of control function registers in the Network Accelerator *that are used for, and apply to, all four SGMII/RGMII interfaces in the block*, i.e., these Network Accelerator registers are in common for all four ports.

These registers are at or above 0x100 within the addressing space of the lowest interface in the GMAC quad. For GMAC Quad 0, these NA registers are in SGMII0; for GMAC Quad 1, these are in port SGMII4.

16.5.2 GMAC Register Descriptions

The following section describes the RGMII/GMAC registers. Bits that are not implemented or are Reserved, are set to zero.

16.5.2.1 GMAC Register Summary

Each of the GMACs (0 through 7) contains its own unique set of the following registers. There is a separate group base address for each GMAC. The register address offset shown below is the offset from the subregion (group) base address shown in Chapter 8.

Note: In RGMII mode, the Reserved registers may not return “0x0000_0000” when read.

Table 16-4. GMAC Registers

Register ID	Address Offset	Mnemonic	Register	Default/Reset
0x00	0x00	MAC_CONF1	MAC Configuration #1	0x8000_0000
0x01	0x04	MAC_CONF2	MAC Configuration #2	0x0000_7000
0x02	0x08	IPG/IFG	IPG / IFG	0x4060_5060
0x03	0x0C	HLF_DUP	Half-Duplex	0x00A1_F037
0x04	0x10	MAX_FRM	Maximum Frame	0x0000_0600
0x05	0x14	Reserved	Reserved	0x0000_0000
0x06	0x18	Reserved	Reserved	0x0000_0000
0x07	0x1C	TEST	Test Register	0x0000_0000
0x08	0x20	MIIM_CONF	MII Mgmt: Configuration	0x0000_0000
0x09	0x24	MIIM_CMD	MII Mgmt: Command	0x0000_0000
0x0A	0x28	MIIM_ADDR	MII Mgmt: Address	0x0000_0000
0x0B	0x2C	MIIM_CTRL	MII Mgmt: Control	0x0000_0000
0x0C	0x30	MIIM_STAT	MII Mgmt: Status	0x0000_0000
0x0D	0x34	MIIM_IND	MII Mgmt: Indicators	0x0000_0000
0x0E	0x38	I/O_CTRL	Interface Control	0x0000_0000
0x0F	0x3C	I/O_STAT	Interface Status	0x0000_0000
0x10	0x40	Reserved	Reserved	0x0000_0000
0x11	0x44	Reserved	Reserved	0x0000_0000
0x12	0x48	Reserved	Reserved	0x0000_0000
-0x1F	-0x7C			
0x20	0x80	GMAC statistics register set See Section 16.5.3 GMAC Statistics Registers Descriptions	GMAC statistics	
-0x4F	-0x13C			
0x50	0x140	MAC_ADDR0_LO	MAC_ADDR0 low word	0x0000_0000
0x51	0x144	MAC_ADDR0_HI	MAC_ADDR0 high word	0x0000_0000
0x52	0x148	MAC_ADDR1_LO	MAC_ADDR1 low word	0x0000_0000
0x53	0x14C	MAC_ADDR1_HI	MAC_ADDR1 high word	0x0000_0000
0x54	0x150	MAC_ADDR2_LO	MAC_ADDR2 low word	0x0000_0000
0x55	0x154	MAC_ADDR2_HI	MAC_ADDR2 high word	0x0000_0000
0x56	0x158	MAC_ADDR3_LO	MAC_ADDR3 low word	0x0000_0000
0x57	0x15C	MAC_ADDR3_HI	MAC_ADDR3 high word	0x0000_0000
0x58	0x160	MAC_ADDR_MASK0_LO	MAC_ADDR_MASK0 low word	0x0000_0000
0x59	0x164	MAC_ADDR_MASK0_HI	MAC_ADDR_MASK0 high word	0x0000_0000

Table 16-4. GMAC Registers (continued)

Register ID	Address Offset	Mnemonic	Register	Default/Reset
0x5A	0x168	MAC_ADDR_MASK1_LO	MAC_ADDR_MASK1 low word	0x0000_0000
0x5B	0x16C	MAC_ADDR_MASK1_HI	MAC_ADDR_MASK1 high word	0x0000_0000
0x5C	0x170	MAC_FILTER_CONFIG	MAC Filter Configuration	0x0000_0000
0x60	0x180	HASH TABLE VECTOR_B31_0	HASH TABLE VECTOR bits 31 to 0	0x0000_0000
0x61	0x184	HASH TABLE VECTOR_B63_32	HASH TABLE VECTOR bits 63 to 32	0x0000_0000
0x62	0x188	HASH TABLE VECTOR_B95_64	HASH TABLE VECTOR bits 95 to 64	0x0000_0000
0x63	0x18C	HASH TABLE VECTOR_B127_96	HASH TABLE VECTOR bits 127 to 96	0x0000_0000
0x64	0x190	HASH TABLE VECTOR_B159_128	HASH TABLE VECTOR bits 159 to 128	0x0000_0000
0x65	0x194	HASH TABLE VECTOR_B191_160	HASH TABLE VECTOR bits 191 to 160	0x0000_0000
0x66	0x198	HASH TABLE VECTOR_B223_192	HASH TABLE VECTOR bits 223 to 192	0x0000_0000
0x67	0x19C	HASH TABLE VECTOR_B255_224	HASH TABLE VECTOR bits 255 to 224	0x0000_0000
0x68	0x1A0	HASH TABLE VECTOR_B287_256	HASH TABLE VECTOR bits 287 to 256	0x0000_0000
0x69	0x1A4	HASH TABLE VECTOR_B319_288	HASH TABLE VECTOR bits 319 to 288	0x0000_0000
0x6A	0x1A8	HASH TABLE VECTOR_B351_320	HASH TABLE VECTOR bits 351 to 320	0x0000_0000
0x6B	0x1AC	HASH TABLE VECTOR_B383_352	HASH TABLE VECTOR bits 383 to 352	0x0000_0000
0x6C	0x1B0	HASH TABLE VECTOR_B415_384	HASH TABLE VECTOR bits 415 to 384	0x0000_0000
0x6D	0x1B4	HASH TABLE VECTOR_B447_416	HASH TABLE VECTOR bits 447 to 416	0x0000_0000
0x6E	0x1B8	HASH TABLE VECTOR_B479_448	HASH TABLE VECTOR bits 479 to 448	0x0000_0000
0x6F	0x1BC	HASH TABLE VECTOR_B511_480	HASH TABLE VECTOR bits 511 to 480	0x0000_0000

16.5.2.2 MAC_CONF1 - MAC Configuration Register #1

Register ID: 0x0 Bits [31:0]

Address Offset: 0x00

31	30	29							20	19	18	17	16
Soft Reset													

15		9	8	7	6	5	4	3	2	1	0		

MAC Configuration Register #1: default value: 0X8000_0000

Bits	Name	Description	R/W	Reset
31	SOFT RESET	The Interface stays in reset until the user clears this bit. 0: enable operation 1: resets all internal logic of the GMAC except for the configuration registers.	R/W	1
30:16	Reserved	Reserved	R/W	0
15:9	Reserved	Reserved		0
8	LOOP BACK	Loop Back control. 0: clearing this bit results in normal operation. 1: causes the MAC Transmit outputs to be looped back to the MAC Receive inputs.	R/W	1
7:6	Reserved	Reserved		0
5	RECEIVE FLOW CONTROL ENABLE	0: clearing this bit causes the Receive MAC Control to ignore PAUSE Flow Control frames. 1: causes the Receive MAC Control to detect and act on PAUSE Flow Control frames.	R/W	0
4	TRANSMIT FLOW CONTROL ENABLE	0: clearing this bit prevents the Transmit MAC Control from sending Flow Control frames. 1: allows the Transmit MAC Control to send PAUSE Flow Control frames when requested by the system.	R/W	0
3	SYNCHRONIZED RECEIVE ENABLE	0: no action 1: Receive Enable is synchronized to the receive stream.	R	0
2	RECEIVE ENABLE	0: clearing this bit will prevent the reception of frames. 1: allows the MAC to receive frames from the PHY.	R/W	0
1	SYNCHRONIZED TRANSMIT	0: no action 1: Transmit Enable is synchronized to the transmit stream.	R	0
0	TRANSMIT ENABLE	0: clearing this bit will prevent the transmission of frames. 1: allows the MAC to transmit frames from the system.	R/W	0

16.5.2.3 MAC_CONF2 - MAC Configuration Register #2

Register ID: 0x01 Bits [31:0]

Address Offset: 0x04

31	<i>Reserved</i>										16
15	12 11	10 9	8 7	5	4	3	2	1	0		
	Preamble Length	Reserved	I/F Mode	Reserved	Length Check	Resv	Pad /CRC	CRC Enable	Full Duplex		

MAC Configuration Register #2: default value is 0X0000_7000

Bits	Name	Description	R/W	Reset
31:16	Reserved	<i>Reserved</i>		0
15:12	PREAMBLE LENGTH	Length of the preamble field of the packet, in bytes.	R/W	0x7
11:10	Reserved	<i>Reserved</i>		0
9:8	INTERFACE MODE	Type of interface connection for the MAC: 00: Reserved 01: Nibble mode (10/100 Mbps, MII/RMII/SMII) 10: Byte mode (1000 Mbps GMII/TBI) 11: Reserved	R/W	0
7:6	Reserved	<i>Reserved</i>		0
5	Reserved	always set to 0	R/W	0
4	LENGTH FIELD CHECKING	0: clear this bit if no length field checking is desired. 1: causes the MAC to check the frame's length field to ensure it matches the actual data field length. 802.3 Ethernet packets have a PDU of less than 1518 bytes. The Ethernet Type field represents the length of the PDU. If the field type is 0x03fe or less, with the Length Field Checking bit set, the XLS will post an error if the packet being DMA'd does not match the value of the Ethernet Type/Length field.	R/W	0
3	Reserved	<i>Reserved</i>		0
2	PAD/CRC ENABLE	0: clear this bit if frames presented to the MAC have a valid length and contain a CRC. 1: set this bit to 1 to have the MAC pad all short frames and append a CRC to every frame whether or not padding was required.	R/W	0
1	CRC ENABLE	0: Clear this bit if frames presented to the MAC have a valid length and contain a valid CRC. 1: Set this bit to 1 to have the MAC append a CRC on all frames. If the configuration bit PAD/CRC ENABLE or the per-packet PAD/CRC ENABLE is set, CRC ENABLE is ignored.	R/W	0
0	FULL-DUPLEX	0: Clearing this bit will configure the GMAC to operate in Half-Duplex mode only. 1: Setting this bit to 1 configures the GMAC to operate in Full-Duplex mode.	R/W	0

16.5.2.4 IPG/IFG - Inter-packet Gap/IFG Register

Register ID: 0x02 Bits [31:0]

Address Offset: 0x08

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Rsvd								Rsvd							NBBIPG_2
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
								Rsvd							BBIPG

IPG / IFG Register: default value is: 0X4060_5060

Bits	Name	Description	R/W	Reset
31	Reserved	Reserved		0
30:24	NBBIPG_1	NON_BACK_TO_BACK_INTERPACKET_GAP PART_1 This is a programmable field representing the optional carrier Sense window referenced in IEEE 802.3/4.2.3.2.1 "Carrier Deference." If carrier is detected during the timing of IPGR1, the MAC defers to carrier. If, however, carrier becomes active after IPGR1, the MAC continues timing IPGR2 and transmits, knowingly causing a collision, thus ensuring fair access to medium. Its range of values is 0x00 to IPGR2. Its default is 0x40 (64d) which follows the two-thirds/one-thirds guideline.	R/W	0x40
23	Reserved	Reserved		0
22:16	NBBIPG_2	NON_BACK_TO_BACK_INTERPACKET_GAP PART_2 This is a programmable field representing the Non-Back-to-Back Inter-Packet-Gap in bits. Its default is 0x60 (96d), which represents the minimum IPG of 96 bits.	R/W	0x60
15:8	MIN_IFG_ENFORCEMENT	MINIMUM_IFG_ENFORCEMENT This is a programmable field representing the minimum number of bits of IFG to enforce between frames. A frame whose IFG is less than that programmed is dropped. The default setting of 0x50 (80d) represents half of the nominal minimum IFG which is 160 bits.	R/W	0x50
7	Reserved	Reserved		0
6:0	BBIPG	BACK-TO-BACK_INTER-PACKET_GAP This is a programmable field representing the IPG between Back-to-Back packets. This is the IPG parameter used exclusively in Full-Duplex mode when two transmit packets are sent back-to-back. Set this field to the number of bits of IPG desired. The default setting of 0x60 (96d) represents the minimum IPG of 96 bits.	R/W	0x60

16.5.2.5 HLF_DUP - Half-Duplex Register

Register ID: 0x03 **Bits [31:0]**

Address Offset: 0x0C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>								ABEBT	ABEBE	BP_NB	No Back Off	EX_DEF			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RETRANS_MAX		<i>Reserved</i>		COLLISION_WIN											

Half-Duplex register.

Bits	Name	Description	R/W	Reset
31:24	Reserved	Reserved		0
23:20	ABEBT	ALTERNATE BINARY EXPONENTIAL BACKOFF TRUNCATION This field is used when ALTERNATE BINARY EXPONENTIAL BACKOFF ENABLE is set. The value programmed is substituted for the Ethernet standard value of ten.	R/W	0xA
19	ABEBE	ALTERNATE BINARY EXPONENTIAL BACKOFF ENABLE 0: Clearing this bit will cause the Tx MAC to follow the standard binary exponential back off rule. 1: Setting this bit will configure the Tx MAC to use the ALTERNATE BINARY EXPONENTIAL BACKOFF TRUNCATION setting instead of the 802.3 standard tenth collision. The Standard specifies that any collision after the tenth uses 210 – 1 as the maximum backoff time.	R/W	0
18	BP_NB	BACK PRESSURE NO BACKOFF 0: Clearing this bit will cause the Tx MAC to follow the binary exponential back off rule. 1: Setting this bit will configure the Tx MAC to immediately re-transmit, following a collision, during back pressure operation.	R/W	0
17	NO BACKOFF	NO BACKOFF 0: Clearing this bit will cause the Tx MAC to follow the binary exponential back off rule. 1: Setting this bit will configure the Tx MAC to immediately re-transmit following a collision.	R/W	0
16	EX_DEF	EXCESSIVE DEFER 0: Clearing this bit will cause the Tx MAC to abort the transmission of a packet that has been excessively deferred. 1: Setting this bit will configure the Tx MAC to allow the transmission of a packet that has been excessively deferred.	R/W	1
15:12	RETRANS_MAX	RETRANSMISSION MAXIMUM This is a programmable field specifying the number of retransmission attempts following a collision before aborting the packet due to excessive collisions. The Standard specifies the attempt Limit to be 0xF (15d). NOTE: If a packet under-run occurs on the last byte, the under-run counter will get reset to zero. Thus, a packet might get retransmitted more times than the configured value in RETRANS_MAX. It is possible, but very unlikely, that this could occur on subsequent packet transmissions.	R/W	0xF

Bits	Name	Description	R/W	Reset
11:10	Reserved	Reserved		0
9:0	COLLISION_WIN	COLLISION WINDOW This is a programmable field representing the slot time or collision window during which collisions occur in properly configured networks. Since the collision window starts at the beginning of transmission, the preamble and SFD are included. Its default of 0x37 (55d) corresponds to the count of frame bytes at the end of the window. This value must always be <=0x3F.	R/W	0x37

16.5.2.6 MAX_FRM - Maximum Frame Length Register

Register ID: 0x04 Bits [31:0]

Address Offset: 0x10

31	Reserved	16
15	MAX_FM_LEN	0

Maximum Frame Length Register: default value: 0X0000_0600

Bits	Bits	Description	R/W	Reset
31:16	Reserved	Reserved		0
15:0	MAX_FM_LEN	MAXIMUM FRAME LENGTH This field resets to the maximum frame size in both the transmit and receive directions. If a different maximum length restriction is desired, program this 16-bit field. This field should be set to a value that is not greater than four packet buffers to guarantee that no incoming packets will require multiple messages.	R/W	0x0600 (1536d)

16.5.2.7 GMAC Reserved 05

Register ID: 0x05 Bits [31:0]

Address Offset: 0x14

31	Reserved	16
15	Reserved	0

Default value: 0X0000_0000

Bits	Bits	Description	R/W	Reset
31:0	Reserved	Reserved		0

16.5.2.8 GMAC Reserved 06

Register ID: 0x06 Bits [31:0]

Address Offset: 0x18

31	<i>Reserved</i>	16
----	-----------------	----

15	<i>Reserved</i>	0
----	-----------------	---

Default value: 0X0000_0000

Bits	Bits	Description	R/W	Reset
31:0	<i>Reserved</i>	<i>Reserved</i>		0

16.5.2.9 TEST Register

Register ID: 0x07 Bits [31:0]

Address Offset: 0x1C

31	<i>Reserved</i>	16
----	-----------------	----

15	4	3	2	1	0
				<i>Test PAUSE</i>	<i>SHORT_SLOT_TIME</i>

Test Register: default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
31:4	<i>Reserved</i>	<i>Reserved</i>		0
3	MAX_BACKOFF	MAXIMUM BACKOFF 0: no action 1: causes the MAC to backoff for the maximum possible length of time. This test bit is used to predict backoff times in Half-Duplex mode.	R/W	0
2	REGTX_FLOW_EN	REGISTERED TRANSMIT FLOW ENABLE 0: no action 1: enables the Register Transmit Half-Duplex Flow.	R/W	0

Bits	Name	Description	R/W	Reset
1	TEST_PAUSE	TEST PAUSE 0: no pause 1: allows the MAC to be paused for testing purposes.	R/W	0
0	SHORT_SLOT_TIME	SHORTCUT SLOT TIME 0: no action 1: allows the slot time counter to expire regardless of the current count. This bit is for testing purposes only.	R/W	0

16.5.2.10 MIIM_CONF - MII Management Configuration Register

Register ID: 0x08 Bits [31:0]

Address Offset: 0x20

31	30							16
RST_MII_MGMT	Reserved							
15	6	5	4	3	2	0		
Reserved		SCAN_AUTOINC	PRE_SUPPR	Rsvd	MGMT_CLK_SEL			

MII Management Configuration Register: default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
31	RST_MII_MGMT	Reset MII Management: 0: clearing this bit allows the MII MGMT to perform Mgmt read/write cycles as requested via the Network Accelerator module. 1: resets the MII MGMT.	R/W	0
30:6	Reserved	Reserved		0
5	SCAN_AUTOINC	Scan Auto Increment: 0: no action 1: causes the MII MGMT to continually read from a set of PHYs of contiguous address space. The starting address of the PHY is specified by the content of the PHY address field, the very next PHY to be read is PHY address + 1. The last PHY to be queried in this read sequence is that which resides at address 0x31, after which the read sequence will continue again for the PHY specified by the PHY address field.	R/W	0
4	PRE_SUPPR	PREAMBLE SUPPRESSION 0: clearing this bit causes the MII MGMT to perform Mgmt read/write cycles with the 32 clocks of preamble. 1: causes the MII MGMT to suppress preamble generation and reduce the Mgmt cycle from 64 clocks to 32 clocks. This is in accordance with IEEE 802.3/22.2.4.4.2.	R/W	0

Bits	Name	Description	R/W	Reset
3	Reserved	Reserved		0
2:0	MGMT_CLK_SEL	Management Clock Select: This field determines the clock frequency of the Mgmt Clock (MDC). 000: Source Clock Divided by 4 001: Source Clock Divided by 4 010: Source Clock Divided by 6 011: Source Clock Divided by 8 100: Source Clock Divided by 10 101: Source Clock Divided by 14 110: Source Clock Divided by 20 111: Source Clock Divided by 54 The source clock which is divided to obtain the selected frequency is the 125 MHz RGMII reference clock (RGMII_GE_CLK).	R/W	000

16.5.2.11 MIIM_CMD - MII Management Command Register

Register ID: 0x09 Bits [31:0]

Address Offset: 0x24

31	Reserved			16
15	Reserved	2	1	0

MII Management Command Register: default value: 0X0000_0000

Bits	Bits	Description	R/W	Reset
31:2	Reserved	Reserved		0
1	SCAN CYCLE	SCAN CYCLE 0: no action 1: causes the MII Management to perform Read cycles continuously. This is useful for monitoring Link Fail for example.	R/W	0
0	READ CYCLE	READ CYCLE 0: no action 1: causes the MII Management to perform a single Read cycle. The Read data is returned in Register 0xC (MII Mgmt Read Data).	R/W	0

16.5.2.12 MIIM_ADDR - MII Management Address Register

Register ID: 0x0A Bits [31:0]

Address Offset: 0x28

31					16
<i>Reserved</i>					
15	13 12	8 7	5 4	0	
<i>Reserved</i>	PHY Address	<i>Reserved</i>	Register Address		

MII Management Address Register: default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
31:2	Reserved	<i>Reserved</i>		0
12:8	PHY ADDRESS	This field represents the 5-bit PHY Address field of Mgmt cycles. Up to 31 PHYs can be addressed (0 is Reserved).	R/W	0x00
7:5	Reserved	<i>Reserved</i>		0
4:0	REGISTER ADDRESS	This field represents the 5-bit Register Address field of Mgmt cycles. Up to 32 registers can be accessed.	R/W	0x00

16.5.2.13 MII MGMT CONTROL - MII Management Control Register

Register ID: 0x0B Bits [31:0]

Address Offset: 0x2C

31					16
<i>Reserved</i>					
15					0
MII MGMT CONTROL (PHY Control)					

MII Management Control Register. Default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
31:16	Reserved	<i>Reserved</i>		0
15:0	MII MGMT CONTROL	MII MGMT CONTROL. This is the PHY Control – When written, an MII Mgmt write cycle is performed using the 16-bit data and the pre-configured PHY and Register addresses from the MII Mgmt Register (0x0A).	R/W	0x0000

16.5.2.14 MIIM_STAT - MII Management Status Register

Register ID: 0x0C Bits [31:0]

Address Offset: 0x30

31	Reserved	16
----	----------	----

15	MII MGMT STATUS (PHY Status)	0
----	------------------------------	---

MII Management Status Register: default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
31:16	Reserved	Reserved		0
15:0	MII MGMT STATUS	MII MGMT STATUS. This is the PHY status – Following an MII Mgmt Read Cycle, the 16-bit data can be read from this location.	R	0x0000

16.5.2.15 MIIM_IND - MII Management Indicators Register

Register ID: 0x0D Bits [31:0]

Address Offset: 0x34

31	Reserved	16
----	----------	----

15	Reserved	3	2	1	0
		Not Valid	Scan	Busy	

MII Management Indicators Register: default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
31:2	Reserved	Reserved		0
2	NOT VALID	1: When '1' is returned - indicates MII Mgmt Read cycle has not completed and the Read Data is not yet valid.	R	0
1	SCANNING	1: When '1' is returned - indicates a scan operation (continuous MII Mgmt Read cycles) is in progress.	R	0
0	BUSY	1: When '1' is returned - indicates MII Mgmt block is currently performing an MII Mgmt Read or Write cycle.	R	0

16.5.2.16 I/O_CTRL - Interface Control Register

Register ID: 0x0E Bits [31:0]

Address Offset: 0x38

31	30	28	27	26	25	24	23	22	17	16
Reserved		TBI Mode	Rsv	LHD Mode					Reserved	
15	14	11	10	9	8	7	6		1	0
								Reserved		EN_JAB_PROT

Interface Control Register: default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
31:28	Reserved	Reserved		0
27	TBIMODE	TBIMODE enable 0: disable 1: Setting this bit configures the RGMII interface to expect TBI signals. When not using this mode this bit should not be asserted.	R/W	0
26	Reserved / SGMII_SPD_1	This bit has two functions: RGMII Mode: (Reserved) SGMII Mode: Gbps Select, as follows: 0: 10 / 100 Mbps SGMII speed 1: 1000 Mbps SGMII speed		0
25	LHDMODE / SGMII_SPD_0	This bit has two functions: RGMII Mode: LHDMODE, as follows: 0: LHD mode ISA not selected. 1: Setting this bit configures the RGMII interface to expect 10 or 100 half duplex MII and will enable the use of CRS and COL signals SGMII Mode: When in SGMII Mode and bit[26] is cleared to '0', this bit selects between 10 Mbps and 100 Mbps, as follows: 0: 10 Mbps SGMII speed 1: 100 Mbps SGMII speed Note that when in SGMII Mode and bit[26] is set to '1', this bit MUST be cleared to '0'. Bits [26:25] = '11' in SGMII mode is not allowed.	R/W	0
24:1	Reserved	Reserved	R/W	0
0	EN_JAB_PROT	ENABLE JABBER PROTECTION This bit enables the Jabber Protection logic in ENDEC mode. Jabber is the condition where a transmitter is stuck 'on' for longer than 50 ms preventing other stations from transmitting. 0: disable 1: enable	R/W	0

16.5.2.17 I/O_STAT - Interface Status Register

Register ID: 0x0F Bits [31:0]

Address Offset: 0x3C

31	Reserved												16									
15	10	9	8	7	6	5	4	3	2	1	0	Reserved	Excess Defer	Clash	Jabber	Link OK	Full Duplex	Speed	Link Fail	Loss Carrier	SQE Error	Jabber

Interface Status Register: default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
30:10	Reserved	Reserved		0
9	EXCESS DEFER	1: This bit sets when the MAC excessively defers a transmission. It clears when read. This bit latches high.	R	0
8	CLASH	0: the Serial MII module is either in PHY mode or in a properly configured MAC to MAC mode. 1: the Serial MII module is in MAC to MAC mode with the partner in 10 Mbps and/or Half-Duplex mode indicative of a configuration error.	R	0
7	JABBER	0: the Serial MII PHY has not detected a jabber condition. 1: the Serial MII PHY has detected a jabber condition on the link.	R	0
6	LINK OK	0: the Serial MII PHY has not detected a valid link. 1: the Serial MII PHY has detected a valid link.	R	0
5	FULL DUPLEX	0: the Serial MII PHY is operating in Half-Duplex mode. 1: the Serial MII PHY is operating in Full-Duplex mode.	R	0
4	SPEED	0: the Serial MII PHY is operating at 10 Mbps. 1: the Serial MII PHY is operating at 100 Mbps mode.	R	0
3	LINK FAIL	Note that for asynchronous host accesses, this bit must be read at least once every scan read cycle of the PHY. 0: the MII management module has read PHY link fail register to be 0'. 1: the MII management module has read PHY link fail register to be 1'.	R	0
2	LOSS OF CARRIER	This bit latches high. 0: the module has not detected a Loss of Carrier. 1: the module has detected a Loss of Carrier.	R	0
1	SQE ERROR	This bit latches high. 0: the module has not detected an SQE Error. 1: the module has detected an SQE Error.	R	0
0	JABBER	This bit latches high. 0: the module has not detected a Jabber condition. 1: the module has detected a Jabber condition.	R	0

16.5.2.18 MAC_ADDR0_LO - Station Address 4-Most-Significant-Bytes Register**Register ID:** 0x50 Bits [31:0]**Address Offset:** 0x140

MAC_ADDR0_LO/HI and MAC_ADDR1_LO/HI are used for full or exact match.

31	24 23	16
Station Address, 1st octet		Station Address, 2nd octet
15	8 7	0
Station Address, 3rd octet		Station Address, 4th octet

Station Address 4-Most-Significant-Bytes Register: default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
31:24	STATION ADDRESS, 1st octet	This field holds the first octet of the station address.	R/W	0
23:16	STATION ADDRESS, 2nd octet	This field holds the second octet of the station address.	R/W	0
15:8	STATION ADDRESS, 3rd octet	This field holds the third octet of the station address.	R/W	0
7:0	STATION ADDRESS, 4th octet	This field holds the fourth octet of the station address.	R/W	0

16.5.2.19 MAC_ADDR0_H - Station Address 2-Least-Significant-Bytes Register**Register ID:** 0x51 Bits [31:0]**Address Offset:** 0x144

31	24 23	16
Station Address, 5th octet		Station Address, 6th octet
15	8 7	0
<i>Reserved</i>		

Station Address 2-Least-Significant-Bytes Register: default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
31:24	STATION ADDRESS, 5th octet	This field holds the fifth octet of the station address.	R/W	0
23:16	STATION ADDRESS, 6th octet	This field holds the sixth octet of the station address.	R/W	0
15:0	<i>Reserved</i>	<i>Reserved</i>		0

16.5.2.20 MAC_ADDR1_LO - Station Address 4-Most-Significant-Bytes Register**Register ID:** 0x52 Bits [31:0]**Address Offset:** 0x148

MAC_ADDR0_LO/HI and MAC_ADDR1_LO/HI are used for full or exact match.

31	24 23	16
Station Address, 1st octet	Station Address, 2nd octet	
15	8 7	0
Station Address, 3rd octet	Station Address, 4th octet	

Station Address 4-Most-Significant-Bytes Register: default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
31:24	STATION ADDRESS, 1st octet	This field holds the first octet of the station address.	R/W	0
23:16	STATION ADDRESS, 2nd octet	This field holds the second octet of the station address.	R/W	0
15:8	STATION ADDRESS, 3rd octet	This field holds the third octet of the station address.	R/W	0
7:0	STATION ADDRESS, 4th octet	This field holds the fourth octet of the station address.	R/W	0

16.5.2.21 MAC_ADDR1_H- Station Address 2-Least-Significant-Bytes Register**Register ID:** 0x53 Bits [31:0]**Address Offset:** 0x14C

31	24 23	16
Station Address, 5th octet	Station Address, 6th octet	
15	0	0
Reserved		

Station Address 2-Least-Significant-Bytes Register: default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
31:24	STATION ADDRESS, 5th octet	This field holds the fifth octet of the station address.	R/W	0
23:16	STATION ADDRESS, 6th octet	This field holds the sixth octet of the station address.	R/W	0
15:0	Reserved	Reserved		0

16.5.2.22 MAC_ADDR2_LO - Station Address 4-Most-Significant-Bytes Register

Register ID: 0x54 Bits [31:0]

Address Offset: 0x150

31	24 23	16
Station Address, 1st octet		Station Address, 2nd octet

15	8 7	0
Station Address, 3rd octet		Station Address, 4th octet

Station Address 4-Most-Significant-Bytes Register: default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
31:24	STATION ADDRESS, 1st octet	This field holds the first octet of the station address.	R/W	0
23:16	STATION ADDRESS, 2nd octet	This field holds the second octet of the station address.	R/W	0
15:8	STATION ADDRESS, 3rd octet	This field holds the third octet of the station address.	R/W	0
7:0	STATION ADDRESS, 4th octet	This field holds the fourth octet of the station address.	R/W	0

16.5.2.23 MAC_ADDR2_H - Station Address 2-Least-Significant-Bytes Register

Register ID: 0x55 Bits [31:0]

Address Offset: 0x154

31	24 23	16
Station Address, 5th octet		Station Address, 6th octet

15	0
<i>Reserved</i>	

Station Address 2-Least-Significant-Bytes Register: default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
31:24	STATION ADDRESS, 5th octet	This field holds the fifth octet of the station address.	R/W	0
23:16	STATION ADDRESS, 6th octet	This field holds the sixth octet of the station address.	R/W	0
15:0	Reserved	Reserved		0

16.5.2.24 MAC_ADDR3_LO - Station Address 4-Most-Significant-Bytes Register

Register ID: 0x56 Bits [31:0]

Address Offset: 0x158

31	24 23	16
Station Address, 1st octet		Station Address, 2nd octet
15	8 7	0
Station Address, 3rd octet		Station Address, 4th octet

Station Address 4-Most-Significant-Bytes Register: default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
31:24	STATION ADDRESS, 1st octet	This field holds the first octet of the station address.	R/W	0
23:16	STATION ADDRESS, 2nd octet	This field holds the second octet of the station address.	R/W	0
15:8	STATION ADDRESS, 3rd octet	This field holds the third octet of the station address.	R/W	0
7:0	STATION ADDRESS, 4th octet	This field holds the fourth octet of the station address.	R/W	0

16.5.2.25 MAC_ADDR3_H - Station Address 2-Least-Significant-Bytes Register

Register ID: 0x57 Bits [31:0]

Address Offset: 0x15C

31	24 23	16
Station Address, 5th octet		Station Address, 6th octet
15	24 23	16
Reserved		0

Station Address 2-Least-Significant-Bytes Register: default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
31:24	STATION ADDRESS, 5th octet	This field holds the fifth octet of the station address.	R/W	0
23:16	STATION ADDRESS, 6th octet	This field holds the sixth octet of the station address.	R/W	0
15:0	Reserved	Reserved		0

16.5.2.26 MAC_ADDR_MASK0_LO**Register ID:** 0x58 Bits [31:0]**Address Offset:** 0x160

This register contains the 32-bit low-mask value to be ANDed with MAC_ADDR2_LO, which is then compared with DestAddr for a match. The same activity is performed with the high word. If a match is detected, and ADDR_MATCH_DISC is enabled, then the packet is declared valid.

16.5.2.27 MAC_ADDR_MASK0_HI**Register ID:** 0x59 Bits [31:0]**Address Offset:** 0x164

This register contains the 32-bit high-mask value to be ANDed with MAC_ADDR2_HI, which is then compared with DestAddr for a match. The same activity is performed with the low word. If a match is detected, and ADDR_MATCH_DISC is enabled, then the packet is declared valid.

16.5.2.28 MAC_ADDR_MASK1_LO**Register ID:** 0x5A Bits [31:0]**Address Offset:** 0x168

This register contains the 32-bit high-mask value to be ANDed with MAC_ADDR3_LO, which is then compared with DestAddr for a match. The same activity is performed with the high word. If a match is detected, and ADDR_MATCH_DISC is enabled, then the packet is declared valid.

16.5.2.29 MAC_ADDR_MASK1_HI**Register ID:** 0x5B Bits [31:0]**Address Offset:** 0x16C

This register contains the 32-bit high-mask value to be ANDed with MAC_ADDR3_HI, which is then compared with DestAddr for a match. The same activity is performed with the low word. If a match is detected, and ADDR_MATCH_DISC is enabled, then the packet is declared valid.

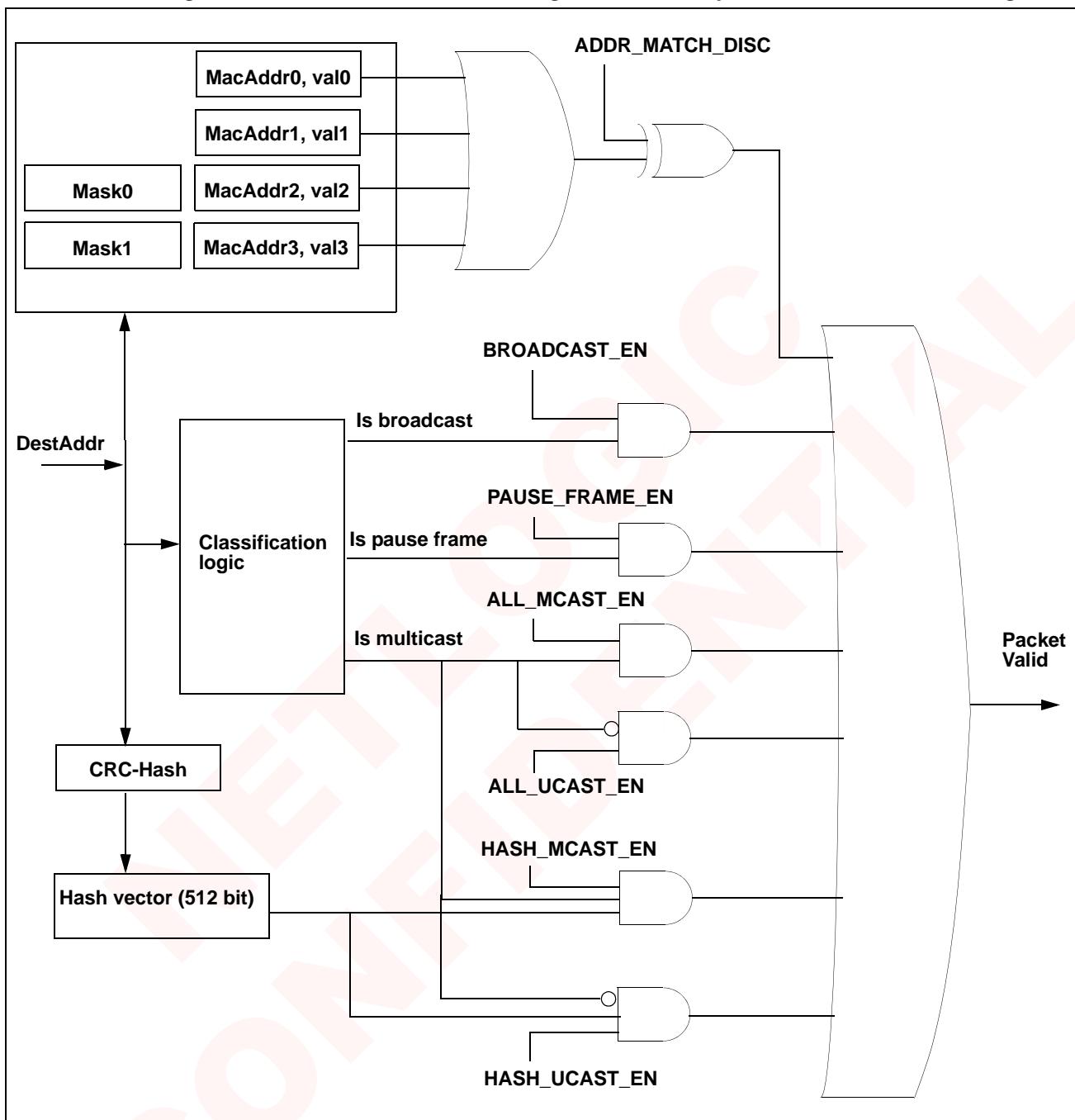
16.5.2.30 MAC_FILTER_CONFIG

Register ID: 0x5C **Bits [31:0]**

Address Offset: 0x170

The bit fields in this register are used to configure the packet address filtering mechanism, as shown in [Figure 16-12](#).

Bits	Name	Description	R/W	Reset
31:11	Reserved	<i>Reserved</i>		
10	BROADCAST_EN	1: Accept all broadcast packets	R/W	0
9	PAUSE_FRAME_EN	1: Accept all pause frames	R/W	0
8	ALL_MCAST_EN	1: Accept All mcast packets	R/W	0
7	ALL_UCAST_EN	1: Accept all ucast packets	R/W	0
6	HASH_MCAST_EN	1: Accept if packet is mcast and hash matches	R/W	0
5	HASH_UCAST_EN	1: Accept if packet is ucast and hash matches	R/W	0
4	ADDR_MATCH_DISC	1: When set, discard if address matches any of the 4 entries, or else accept if address matches	R/W	0
3	MAC_ADDR3_VALID	1: MAC_ADDR3 entry is valid	R/W	1
2	MAC_ADDR2_VALID	1: MAC_ADDR2 entry is valid	R/W	1
1	MAC_ADDR1_VALID	1: MAC_ADDR1 entry is valid	R/W	1
0	MAC_ADDR0_VALID	1: MAC_ADDR0 entry is valid	R/W	1

Figure 16-12. MAC Filter Control Logic Controlled by MAC_FILTER_CONFIG Register**16.5.2.31 HASH_TABLE_VECTOR**

The hash table vector used for packet filtering appears in these registers:

Table 16-5. Hash Table Vector Registers

Register ID	Address Offset	Name	Function
0x60	0x180	HASH TABLE VECTOR_B31_0	HASH TABLE VECTOR bits 31 to 0
0x61	0x184	HASH TABLE VECTOR_B63_32	HASH TABLE VECTOR bits 63 to 32
0x62	0x188	HASH TABLE VECTOR_B95_64	HASH TABLE VECTOR bits 95 to 64
0x63	0x18C	HASH TABLE VECTOR_B127_96	HASH TABLE VECTOR bits 127 to 96
0x64	0x190	HASH TABLE VECTOR_B159_128	HASH TABLE VECTOR bits 159 to 128
0x65	0x194	HASH TABLE VECTOR_B191_160	HASH TABLE VECTOR bits 191 to 160
0x66	0x198	HASH TABLE VECTOR_B223_192	HASH TABLE VECTOR bits 223 to 192
0x67	0x19C	HASH TABLE VECTOR_B255_224	HASH TABLE VECTOR bits 255 to 224
0x68	0x1A0	HASH TABLE VECTOR_B287_256	HASH TABLE VECTOR bits 287 to 256
0x69	0x1A4	HASH TABLE VECTOR_B319_288	HASH TABLE VECTOR bits 319 to 288
0x6A	0x1A8	HASH TABLE VECTOR_B351_320	HASH TABLE VECTOR bits 351 to 320
0x6B	0x1AC	HASH TABLE VECTOR_B383_352	HASH TABLE VECTOR bits 383 to 352
0x6C	0x1B0	HASH TABLE VECTOR_B415_384	HASH TABLE VECTOR bits 415 to 384
0x6D	0x1B4	HASH TABLE VECTOR_B447_416	HASH TABLE VECTOR bits 447 to 416
0x6E	0x1B8	HASH TABLE VECTOR_B479_448	HASH TABLE VECTOR bits 479 to 448
0x6F	0x1BC	HASH TABLE VECTOR_B511_480	HASH TABLE VECTOR bits 511 to 480

16.5.3 GMAC Statistics Registers Descriptions

16.5.3.1 GMAC Statistics Registers Summary

Each of the GMACs (A through D) contains its own unique set of the following registers. That is, there are four sets of these registers. Registers in each set are named the same and are therefore uniquely identified by their name together with their group base address and offset.

The group base address for each set of statistics registers is the same as the group base address for that GMAC (A though D).

Table 16-6. GMAC Statistics Registers

Register ID	Address Offset	Mnemonic	Register	Type
Transmit and Receive Counters				
0x20	0x80	TR64	Transmit and Receive 64 Byte Frame Counter	R/W
0x21	0x84	TR127	Transmit and Receive 65 to 127 Byte Frame Counter	R/W
0x22	0x88	TR255	Transmit and Receive 128 to 255 Byte Frame Counter	R/W
0x23	0x8C	TR511	Transmit and Receive 256 to 511 Byte Frame Counter	R/W
0x24	0x90	TR1K	Transmit and Receive 512 to 1023 Byte Frame Counter	R/W
0x25	0x94	TRMAX	Transmit and Receive 1024 to 1518 Byte Frame Counter	R/W
0x26	0x98	TRMVG	Transmit and Receive 1519 to 1522 Byte Good VLAN Frame Count	R/W
Receive Counters				
0x27	0x9C	RBYT	Receive Byte Counter	R/W
0x28	0xA0	RPKT	Receive Packet Counter	R/W
0x29	0xA4	RFCS	Receive FCS Error Counter	R/W
0x2A	0xA8	RMCA	Receive Multicast Packet Counter	R/W
0x2B	0xAC	RBCA	Receive Broadcast Packet Counter	R/W
0x2C	0xB0	RXCF	Receive Control Frame Packet Counter	R/W
0x2D	0xB4	RXPF	Receive PAUSE Frame Packet Counter	R/W
0x2E	0xB8	RXUO	Receive Unknown OP code Counter	R/W
0x2F	0xBC	RALN	Receive Alignment Error Counter	R/W
0x30	0xC0	RFLR	Receive Frame Length Error Counter	R/W
0x31	0xC4	RCDE	Receive Code Error Counter	R/W
0x32	0xC8	RCSE	Receive Carrier Sense Error Counter	R/W
0x33	0xCC	RUND	Receive Undersize Packet Counter	R/W
0x34	0xD0	ROVR	Receive Oversize Packet Counter	R/W
0x35	0xD4	RFRG	Receive Fragments Counter	R/W
0x36	0xD8	RJBR	Receive Jabber Counter	R/W
0x37	0xDC	Reserved	Reserved	R/W
Transmit Counters				
0x38	0xE0	TBYT	Transmit Byte Counter	R/W
0x39	0xE4	TPKT	Transmit Packet Counter	R/W
0x3A	0xE8	TMCA	Transmit Multicast Packet Counter	R/W
0x3B	0xEC	TBCA	Transmit Broadcast Packet Counter	R/W
0x3C	0xF0	TXPF	Transmit PAUSE Control Frame Counter	R/W
0x3D	0xF4	TDFR	Transmit Deferral Packet Counter	R/W
0x3E	0xF8	TEDF	Transmit Excessive Deferral Packet Counter	R/W

Table 16-6. GMAC Statistics Registers (continued)

Register ID	Address Offset	Mnemonic	Register	Type
0x3F	0xFC	TSCL	Transmit Single Collision Packet Counter	R/W
0x40	0x100	TMCL	Transmit Multiple Collision Packet Counter	R/W
0x41	0x104	TLCL	Transmit Late Collision Packet Counter	R/W
0x42	0x108	TXCL	Transmit Excessive Collision Packet Counter	R/W
0x43	0x10C	TNCL	Transmit Total Collision Counter	R/W
0x44	0x110	Reserved	Reserved	R/W
0x45	0x114	Reserved	Reserved	R/W
0x46	0x118	TJBR	Transmit Jabber Frame Counter	R/W
0x47	0x11C	TFCS	Transmit FCS Error Counter	R/W
0x48	0x120	TXCF	Transmit Control Frame Counter	R/W
0x49	0x124	TOVR	Transmit Oversize Frame Counter	R/W
0x4A	0x128	TUND	Transmit Undersize Frame Counter	R/W
0x4B	0x12C	TFRG	Transmit Fragments Frame Counter	R/W
General Registers				
0x4C	130	CAR1	Carry Register One Register*	RO
0x4D	134	CAR2	Carry Register Two Register*	RO
0x4E	138	CAM1	Carry Register One Mask Register	R/W
0x4F	13C	CAM2	Carry Register Two Mask Register	R/W

* Counter values will be discretely cleared on read if AUTOZ is asserted in the Network Accelerator StatCtrl register (0x0A3).

Use of Carry and Carry Mask Registers

The CAR1 and CAR2 registers allow the programmer to have a given statistics counter register generate an interrupt when it rolls over.

These interrupts are processed as follows: if the OverFlowEn bit field is asserted in the Network Accelerator StatCtrl register (0x0A3), then if any unmasked statistics counter overflows in the RGMII/SGMII, the Network Accelerator receives an interrupt to be passed on to the PIC.

For software to handle a statistics carry interrupt in the PIC:

1. PIC generates the interrupt.
2. User must first read the Int register of the Network Accelerator for this Network Interface to find the source of the interrupt.
3. Then read CAR1 and CAR2 to determine the original source causing the overflow.
4. To clear the interrupt from the Network Accelerator for this Network Interface, write a '1' to bit[2] of that Network Accelerator's Int register.

16.5.3.2 TR64 – Transmit & Receive 64 Byte Frame Counter

Register ID: 0x20 Bits [31:0]

Address Offset: 0x80

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TR64(17:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR64 (15:0)															
Bits	Name	Description			R/W	Reset									
31:18	Reserved	<i>Reserved</i>													
17:0	TR64	Transmit and Receive 64 Byte Frame Counter. Incremented for each good or bad frame transmitted and received which is 64 bytes in length inclusive (excluding framing bits but including FCS bytes).			R/W	0									

16.5.3.3 TR127 – Transmit & Receive 65 to 127 Byte Frame Counter

Register ID: 0x21 Bits [31:0]

Address Offset: 0x84

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TR127(17:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR127(15:0)															
Bits	Name	Description			R/W	Reset									
31:18	Reserved	<i>Reserved</i>													
17:0	TR127	Transmit and Receive 65 to 127 Byte Frame Counter: Incremented for each good or bad frame transmitted and received which is 65 to 127 bytes in length inclusive (excluding framing bits but including FCS bytes).			R/W	0									

16.5.3.4 TR255 – Transmit & Receive 128 to 255 Byte Frame Counter

Register ID: 0x22 Bits [31:0]

Address Offset: 0x88

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TR255(17:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR255(15:0)															
Bits	Name	Description				R/W	Reset								
31:18	Reserved	<i>Reserved</i>													
17:0	TR255	Transmit and Receive 128 to 255 Byte Frame Counter. Incremented for each good or bad frame transmitted and received which is 128 to 255 bytes in length inclusive (excluding framing bits but including FCS bytes).				R/W	0								

16.5.3.5 TR511 – Transmit & Receive 256 to 511 Byte Frame Counter

Register ID: 0x23 Bits [31:0]

Address Offset: 0x8C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TR511(17:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR511(15:0)															
Bits	Name	Description				R/W	Reset								
31:18	Reserved	<i>Reserved</i>													
17:0	TR511	Transmit and Receive 256 to 511 Byte Frame Counter. Incremented for each good or bad frame transmitted and received which is 256 to 511 bytes in length inclusive (excluding framing bits but including FCS bytes).				R/W	0								

16.5.3.6 TR1K – Transmit & Receive 512 to 1023 Byte Frame Counter

Register ID: 0x24 Bits [31:0]

Address Offset: 0x90

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TR1K(17:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR1K(15:0)															
Bits	Name	Description			R/W	Reset									
31:18	Reserved	<i>Reserved</i>													
17:0	TR1K	Transmit and Receive 512 to 1023 Byte Frame Counter. Incremented for each good or bad frame transmitted and received which is 512 to 1023 bytes in length inclusive (excluding framing bits but including FCS bytes).			R/W	0									

16.5.3.7 TRMAX – Transmit & Receive 1024 to 1518 Byte Frame Counter

Register ID: 0x25 Bits [31:0]

Address Offset: 0x94

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TRMAX(17:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRMAX(15:0)															
Bits	Name	Description			R/W	Reset									
31:18	Reserved	<i>Reserved</i>													
17:0	TRMAX	Transmit and Receive 1024 to 1518 Byte Frame Counter. Incremented for each good or bad frame transmitted and received which is 1024 to 1518 bytes in length inclusive (excluding framing bits but including FCS bytes).			R/W	0									

16.5.3.8 TRMGV – Transmit & Receive 1519 to 1522 Byte VLAN Frame Counter

Register ID: 0x26 Bits [31:0]

Address Offset: 0x98

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TRMGV(17:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRMGV(15:0)															
Bits	Name	Description				R/W	Reset								
31:18	Reserved	<i>Reserved</i>													
17:0	TRMGV	Transmit and Receive 1024 to 1518 Byte Frame Counter. Incremented for each good or bad frame transmitted and received which is 1024 to 1518 bytes in length inclusive (excluding framing bits but including FCS bytes).				R/W	0								

16.5.3.9 RBYT - Receive Byte Counter

Register ID: 0x27 Bits [31:0]

Address Offset: 0x9C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														RBYT(23:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBYT(15:0)															
Bits	Name	Description				R/W	Reset								
31:23	Reserved	<i>Reserved</i>													
22:0	RBYT	Receive Byte Counter. The Statistic Counter register is incremented by the byte count of frames received with 0 to 1518 bytes, including those in bad packets, excluding framing bits but including FCS bytes.				R/W	0								

16.5.3.10 RPKT - Receive Packet Counter

Register ID: 0x28 Bits [31:0]

Address Offset: 0xA0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Reserved														RPKT(17:16)			
RPKT (15:0)																	
Bits	Name		Description												R/W	Reset	
31:18	Reserved		Reserved														
17:0	RPKT		Receive Packet Counter. Incremented for each frame received packet (including bad packets, all Unicast, Broadcast, and Multicast packets).												R/W	0	

16.5.3.11 RFCS - Receive FCS Error Counter

Register ID: 0x29 Bits [31:0]

Address Offset: 0xA4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Reserved																	
Reserved																	
Bits	Name		Description												R/W	Reset	
31:12	Reserved		Reserved														
11:0	RFCS		Receive FCS Error Counter. Incremented for each frame received that has a integral 64 to 1518 length and contains a Frame Check Sequence error.												R/W	0	

16.5.3.12 RMCA - Receive Multicast Packet Counter

Register ID: 0x2A Bits [31:0]

Address Offset: 0xA8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														RMCA(17:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RMCA(15:0)															
Bits	Name	Description				R/W	Reset								
31:18	Reserved	<i>Reserved</i>													
17:0	RMCA	Receive Multicast Packet Counter. Incremented for each Multicast good frame of lengths 64 to 1518 (non VLAN) or 1522 (VLAN) excluding Broadcast frames. This does not look at range/length errors.				R/W	0								

16.5.3.13 RBCA - Receive Broadcast Packet Counter

Register ID: 0x2B Bits [31:0]

Address Offset: 0xAC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														RBRA(21:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBRA(15:0)															
Bits	Name	Description				R/W	Reset								
31:22	Reserved	<i>Reserved</i>													
21:0	RBRA	Receive Broadcast Packet Counter. Incremented for each Broadcast good frame of lengths 64 to 1518 (non VLAN) or 1522 (VLAN) excluding Multicast frames. This does not look at range/length errors.				R/W	0								

16.5.3.14 RXCF - Receive Control Frame Packet Counter

Register ID: 0x2C Bits [31:0]

Address Offset: 0xB0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														RXCF(17:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCF(15:0)															
Bits	Name	Description												R/W	Reset
31:18	Reserved	<i>Reserved</i>													
17:0	RXCF	Receive Control Frame Packet Counter. Incremented for each MAC Control frame received (PAUSE & Unsupported).												R/W	0

16.5.3.15 RXPF - Receive PAUSE Frame Packet Counter

Register ID: 0x2D Bits [31:0]

Address Offset: 0xB4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
<i>Reserved</i>																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved			RXPF(11:0)													
Bits	Name	Description												R/W	Reset	
31:12	Reserved	<i>Reserved</i>														
11:0	RXPF	Receive PAUSE Frame Packet Counter. Incremented each time a valid PAUSE MAC Control frame is received.												R/W	0	

16.5.3.16 RXUO - Receive Unknown OPCode Packet Counter

Register ID: 0x2E Bits [31:0]

Address Offset: 0xB8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16														
<i>Reserved</i>																													
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
<i>Reserved</i>		RXUO(11:0)																											
Bits																													
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Bits</th><th style="text-align: left; padding: 2px;">Name</th><th style="text-align: left; padding: 2px;">Description</th><th style="text-align: left; padding: 2px;">R/W</th><th style="text-align: left; padding: 2px;">Reset</th></tr> </thead> <tbody> <tr> <td style="padding: 2px;">31:12</td><td style="padding: 2px;"><i>Reserved</i></td><td style="padding: 2px;"><i>Reserved</i></td><td style="padding: 2px;"></td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;">11:0</td><td style="padding: 2px;">RXUO</td><td style="padding: 2px;">Receive Unknown OPcode Counter. Incremented each time a MAC Control Frame is received which contains an opcode other than a PAUSE.</td><td style="padding: 2px;">R/W</td><td style="padding: 2px;">0</td></tr> </tbody> </table>															Bits	Name	Description	R/W	Reset	31:12	<i>Reserved</i>	<i>Reserved</i>			11:0	RXUO	Receive Unknown OPcode Counter. Incremented each time a MAC Control Frame is received which contains an opcode other than a PAUSE.	R/W	0
Bits	Name	Description	R/W	Reset																									
31:12	<i>Reserved</i>	<i>Reserved</i>																											
11:0	RXUO	Receive Unknown OPcode Counter. Incremented each time a MAC Control Frame is received which contains an opcode other than a PAUSE.	R/W	0																									

16.5.3.17 RALN - Receive Alignment Error Counter

Register ID: 0x2F Bits [31:0]

Address Offset: 0xBC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16															
<i>Reserved</i>																														
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0															
<i>Reserved</i>		RALN(11:0)																												
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left; padding: 2px;">Bits</th><th style="text-align: left; padding: 2px;">Name</th><th style="text-align: left; padding: 2px;">Description</th><th style="text-align: left; padding: 2px;">R/W</th><th style="text-align: left; padding: 2px;">Reset</th></tr> </thead> <tbody> <tr> <td style="padding: 2px;">31:12</td><td style="padding: 2px;"><i>Reserved</i></td><td style="padding: 2px;"><i>Reserved</i></td><td style="padding: 2px;"></td><td style="padding: 2px;"></td></tr> <tr> <td style="padding: 2px;">11:0</td><td style="padding: 2px;">RALN</td><td style="padding: 2px;">Receive Alignment Error Counter. Incremented for each received frame from 64 to 1518 (non VLAN) or 1522 (VLAN) which contains an invalid FCS and is not an integral number of bytes.</td><td style="padding: 2px;">R/W</td><td style="padding: 2px;">0</td></tr> </tbody> </table>																Bits	Name	Description	R/W	Reset	31:12	<i>Reserved</i>	<i>Reserved</i>			11:0	RALN	Receive Alignment Error Counter. Incremented for each received frame from 64 to 1518 (non VLAN) or 1522 (VLAN) which contains an invalid FCS and is not an integral number of bytes.	R/W	0
Bits	Name	Description	R/W	Reset																										
31:12	<i>Reserved</i>	<i>Reserved</i>																												
11:0	RALN	Receive Alignment Error Counter. Incremented for each received frame from 64 to 1518 (non VLAN) or 1522 (VLAN) which contains an invalid FCS and is not an integral number of bytes.	R/W	0																										

16.5.3.18 RFLR - Receive Frame Length Error Counter]

Register ID: 0x30 Bits [31:0]

Address Offset: 0xC0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFLR(15:0)															
Bits	Name	Description										R/W	Reset		
31:16	Reserved	Reserved													
15:0	RFLR)	Receive Frame Length Error Counter. Incremented for each frame received in which the 802.3 length field did not match the number of data bytes actually received (46 - 1500 bytes). The counter is not incremented if the length field is not a valid 802.3 length, such as an EtherType value.										R/W	0		

16.5.3.19 RCDE - Receive Code Error Counter

Register ID: 0x31 Bits [31:0]

Address Offset: 0xC4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RCDE(11:0)															
Bits	Name	Description										R/W	Reset		
31:12	Reserved	Reserved													
11:0	RCDE	Receive Code Error Counter. Incremented each time a valid carrier was present and at least one invalid data symbol was detected.										R/W	0		

16.5.3.20 RCSE - Receive Carrier Sense Error Counter

Register ID: 0x32 Bits [31:0]

Address Offset: 0xC8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		RCSE[11:0]													
Bits	Name	Description		R/W	Reset										
31:12	Reserved	Reserved													
11:0	RCSE	Receive False Carrier Counter. Incremented each time a false carrier is detected during idle, as defined by a 1 on RX_ER and an '0xE' on RXD. The event is reported along with the statistics generated on the next received frame. Only one false carrier condition can be detected and logged between frames.		R/W	0										

16.5.3.21 RUND- Receive Undersize Packet Counter

Register ID: 0x33 Bits [31:0]

Address Offset: 0xCC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		RUND(11:0)													
Bits	Name	Description		R/W	Reset										
31:12	Reserved	Reserved													
11:0	RUND	Receive Undersize Packet Counter. Incremented each time a frame is received which is less than 64 bytes in length and contains a valid FCS and were otherwise well formed. This does not look at Range Length errors.		R/W	0										

16.5.3.22 ROVR - Receive Oversize Packet Counter

Register ID: 0x34 Bits [31:0]

Address Offset: 0xD0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				ROVR(11:0)											
Bits	Name	Description												R/W	Reset
31:12	Reserved	<i>Reserved</i>													
11:0	ROVR	Receive Oversize Packet Counter. Incremented each time a frame is received which exceeded 1518 (non VLAN) or 1522 (VLAN) and contains a valid FCS and were otherwise well formed. This does not look at Range Length errors.												R/W	0

16.5.3.23 RFRG - Receive Fragments Counter

Register ID: 0x35 Bits [31:0]

Address Offset: 0xD4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				RFRG(11:0)											
Bits	Name	Description												R/W	Reset
31:12	Reserved	<i>Reserved</i>													
11:0	RFRG	Receive Fragments Counter. Incremented for each frame received which is less than 64 bytes in length and contains an invalid FCS, includes integral and non-integral lengths.												R/W	0

16.5.3.24 RJBR - Receive Jabber Counter

Register ID: 0x36 Bits [31:0]

Address Offset: 0xD8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>															
Bits	Name	Description										R/W	Reset		
31:12	Reserved	<i>Reserved</i>													
11:0	RJBR	Receive Jabber Counter. Incremented for frames received which exceed 1518 (non VLAN) or 1522 (VLAN) bytes and contains an invalid FCS, includes alignment errors.										R/W	0		

16.5.3.25 TBYT - Transmit Byte Counter

Register ID: 0x38 Bits [31:0]

Address Offset: 0xE0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>										TBYT(23:16)					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>TBYT(15:0)</i>															
Bits	Name	Description										R/W	Reset		
31:24	Reserved	<i>Reserved</i>													
23:0	TBYT	Transmit Byte Counter. Incremented by the number of bytes that were put on the wire including fragments of frames that were involved with collisions. This count does not include preamble/SFD or jam bytes.										R/W	0		

16.5.3.26 TPKT - Transmit Packet Counter

Register ID: 0x39 Bits [31:0]

Address Offset: 0xE4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TPKT(17:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPKT(15:0)															
Bits	Name	Description			R/W	Reset									
31:18	Reserved														
17:0	TPYT	Transmit Packet Counter. Incremented for each transmitted packet (including bad packets, excessive deferred packets, excessive collision packets, late collision packets, all Unicast, Broadcast, and Multicast packets).			R/W	0									

16.5.3.27 TMCA - Transmit Multicast Packet Counter

Register ID: 0x3A Bits [31:0]

Address Offset: 0xE8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TMCA(17:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMCA(15:0)															
Bits	Name	Description			R/W	Reset									
31:18	Reserved	Reserved													
17:0	TMCA	Transmit Multicast Packet Counter. Incremented for each Multicast valid frame transmitted (excluding Broadcast frames).			R/W	0									

16.5.3.28 TBCA - Transmit Broadcast Packet Counter

Register ID: 0x3B Bits [31:0]

Address Offset: 0xEC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TBCA(17:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBCA(15:0)															
Bits	Name	Description				R/W	Reset								
31:18	Reserved	<i>Reserved</i>													
17:0	TBCA	Transmit Broadcast Packet Counter. Incremented for each Broadcast frame transmitted (excluding Multicast frames).				R/W	0								

16.5.3.29 TXPF - Transmit PAUSE Control Frame Counter

Register ID: 0x3C Bits [31:0]

Address Offset: 0xF0

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TXPF(11:0)											
Bits	Name	Description				R/W	Reset								
31:12	Reserved	<i>Reserved</i>													
11:0	TXPF	Transmit PAUSE Frame Packet Counter. Incremented each time a valid PAUSE MAC Control frame is transmitted.				R/W	0								

16.5.3.30 TDFR - Transmit Deferral Packet Counter

Register ID: 0x3D Bits [31:0]

Address Offset: 0xF4

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved		TDFR[11:0]														
Bits	Name	Description													R/W	Reset
31:12	Reserved	Reserved														
11:0	TDFR	Transmit Deferral Packet Counter. Incremented for each frame, which was deferred on its first transmission attempt. Does not include frames involved in collisions.													R/W	0

16.5.3.31 TEDF - Transmit Excessive Deferral Packet Counter]

Register ID: 0x3E Bits [31:0]

Address Offset: 0xF8

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
Reserved																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved		TEDF[11:0]														
Bits	Name	Description													R/W	Reset
31:12	Reserved	Reserved														
11:0	TEDF	Transmit Excessive Deferral Packet Counter. Incremented for frames aborted which were deferred for an excessive period of time (3036 byte times).													R/W	0

16.5.3.32 TSCL - Transmit Single Collision Packet Counter

Register ID: 0x3F Bits [31:0]

Address Offset: 0xFC

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
<i>Reserved</i>																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
<i>Reserved</i>		TSCL[11:0]														
Bits	Name		Description												R/W	Reset
31:12	<i>Reserved</i>		<i>Reserved</i>													
11:0	TSCL		Transmit Single Collision Packet Counter. Incremented for each frame transmitted which experienced exactly one collision during transmission.												R/W	0

16.5.3.33 TMCL - Transmit Multiple Collision Packet Counter

Register ID: 0x40 Bits [31:0]

Address Offset: 0x100

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
<i>Reserved</i>																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
<i>Reserved</i>		TMCL[11:0]														
Bits	Name		Description												R/W	Reset
31:12	<i>Reserved</i>		<i>Reserved</i>													
11:0	TMCL		Transmit Multiple Collision Packet Counter. Incremented for each frame transmitted which experienced 2-15 collisions (including any late collisions) during transmission as defined using the RETRY[3:0] field of the TX function control register.												R/W	0

16.5.3.34 TLCL - Transmit Late Collision Packet Counter

Register ID: 0x41 Bits [31:0]

Address Offset: 0x104

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
<i>Reserved</i>																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
<i>Reserved</i>		TLCL[11:0]														
Bits	Name	Description													R/W	Reset
31:12	Reserved	<i>Reserved</i>														
11:0	TLCL	Transmit Late Collision Packet Counter. Incremented for each frame transmitted which experienced a late collision during a transmission attempt. Late collisions are defined using the LCOL[5:0] field of the TX Function control register.													R/W	0

16.5.3.35 TXCL - Transmit Excessive Collision Packet Counter

Register ID: 0x42 Bits [31:0]

Address Offset: 0x108

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
<i>Reserved</i>																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
<i>Reserved</i>		TXCL[11:0]														
Bits	Name	Description													R/W	Reset
31:12	Reserved	<i>Reserved</i>														
11:0	TXCL	Transmit Excessive Collision Packet Counter. Incremented for each frame that experienced 16 collisions during transmission and was aborted.													R/W	0

16.5.3.36 TNCL - Transmit Total Collision CounterRegister ID: **0x43 Bits [31:0]**Address Offset: **0x10C**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>		TNCL[12:0]													
Bits	Name	Description		R/W	Reset										
31:12	<i>Reserved</i>	<i>Reserved</i>													
12:0	TNCL	Transmit Total Collision Counter. Incremented by the number of collisions experienced during the transmission of a frame as defined as the simultaneous presence of signals on the DO and RD circuits (i.e. transmitting and receiving at the same time). Note, this register does not include collisions that result in an excessive collision condition)		R/W	0										

16.5.3.37 TJBR - Transmit Jabber Frame CounterRegister ID: **0x46 Bits [31:0]**Address Offset: **0x118**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>		TJBR(11:0)													
Bits	Name	Description		R/W	Reset										
31:12	<i>Reserved</i>	<i>Reserved</i>													
11:0	TJBR	Transmit Jabber Frame Counter. Incremented for each oversized transmitted frame with an incorrect FCS value.		R/W	0										

16.5.3.38 TFCS - Transmit FCS Error Counter

Register ID: 0x47 Bits [31:0]

Address Offset: 0x11C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>															
Bits	Name	Description										R/W	Reset		
31:12	Reserved	<i>Reserved</i>													
11:0	TFCS	Transmit FCS Error Counter. Incremented for every valid sized packet with an incorrect FCS value.										R/W	0		

16.5.3.39 TXCF - Transmit Control Frame Counter

Register ID: 0x48 Bits [31:0]

Address Offset: 0x120

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>															
Bits	Name	Description										R/W	Reset		
31:12	Reserved	<i>Reserved</i>													
11:0	TXCF	Transmit Control Frame Counter. Incremented for every valid size frame with a Type Field signifying a Control frame.										R/W	0		

16.5.3.40 TOVR - Transmit Oversize Frame CounterRegister ID: **0x49 Bits [31:0]**Address Offset: **0x124**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>		TOVR(11:0)													
Bits	Name	Description		R/W	Reset										
31:12	<i>Reserved</i>	<i>Reserved</i>													
11:0	TOVR	Transmit Oversize Frame Counter. Incremented for each oversized transmitted frame with an correct FCS value.		R/W	0										

16.5.3.41 TUND - Transmit Undersize Frame CounterRegister ID: **0x4A Bits [31:0]**Address Offset: **0x128**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>		TUND(11:0)													
Bits	Name	Description		R/W	Reset										
31:12	<i>Reserved</i>	<i>Reserved</i>													
11:0	TUND	Transmit Undersize Frame Counter. Incremented for every frame less than 64 bytes, with a correct FCS value.		R/W	0										

16.5.3.42 TFRG - Transmit Fragment Counter

Register ID: 0x4B Bits [31:0]

Address Offset: 0x12C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>				TFRG(11:0)											
Bits	Name			Description										R/W	Reset
31:12	<i>Reserved</i>			<i>Reserved</i>											
11:0	TFRG			Transmit Fragment Counter. Incremented for every frame less than 64 bytes, with a incorrect FCS value.										R/W	0

16.5.3.43 CAR1 - Carry Register 1

Register ID: 0x4C Bits [31:0]

Address Offset: 0x130

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C1 64	C1 127	C1 255	C1 511	C1 1K	C1 MAX	C1 MGV	<i>Reserved</i>								C1 RBY

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C1 RPK	C1 RFC	C1 RMC	C1 RBC	C1 RXC	C1 RXP	C1 RXU	C1 RAL	C1 RFL	C1 RCD	C1 RCS	C1 RUN	C1 ROV	C1 RFR	C1 RJB	C1 RDR

Carry Register One *

Bits	Name	Description	R/W	Reset
31	C164	Carry register 1 TR64 Counter Carry bit	R	0
30	C1127	Carry register 1 TR127 Counter Carry bit	R	0
29	C1255	Carry register 1 TR255 Counter Carry bit	R	0
28	C1511	Carry register 1 TR511 Counter Carry bit	R	0
27	C11k	Carry register 1 TR1K Counter Carry bit	R	0
26	C1MAX	Carry register 1 TRMAX Counter Carry bit	R	0
25	C1MGV	Carry register 1 TRMGV Counter Carry bit	R	0
24:17	Reserved	<i>Reserved</i>	R	0
16	C1RBY	Carry register 1 RBYT Counter Carry bit	R	0
15	C1RPK	Carry register 1 RPKT Counter Carry bit	R	0
14	C1RFC	Carry register 1 RFCS Counter Carry bit	R	0
13	C1RMC	Carry register 1 RMCA Counter Carry bit	R	0
12	C1RBC	Carry register 1 RBCA Counter Carry bit	R	0
11	C1RXC	Carry register 1 RXCF Counter Carry bit	R	0
10	C1RXP	Carry register 1 RXPF Counter Carry bit	R	0
9	C1RXU	Carry register 1 RXUO Counter Carry bit	R	0
8	C1RAL	Carry register 1 RALN Counter Carry bit	R	0
7	C1RFL	Carry register 1 RFLR Counter Carry bit	R	0
6	C1RCD	Carry register 1 RCDE Counter Carry bit	R	0
5	C1RCS	Carry register 1 RCSE Counter Carry bit	R	0
4	C1RUN	Carry register 1 RUND Counter Carry bit	R	0
3	C1ROV	Carry register 1 ROVR Counter Carry bit	R	0
2	C1RFR	Carry register 1 RFRG Counter Carry bit	R	0
1	C1RJB	Carry register 1 RJBR Counter Carry bit	R	0
0	C1RDR	Carry register 1 RDRP Counter Carry bit	R	0

* Carry register bits are cleared on carry register write while respective bit is asserted

16.5.3.44 CAR2 - Carry Register 2

Register ID: 0x4D Bits [31:0]

Address Offset: 0x134

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
Reserved														C2 TJB	C2 TFC	C2 TCF	C2 TOV
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
C2 TUN	C2 TFG	C2 TBY	C2 TPK	C2 TMC	C2 TBC	C2 TPF	C2 TDF	C2 TED	C2 TSC	C2 TMA	C2 TLC	C2 TXC	C2 TNC	C2 TPH	C2 TDP		

Carry Register 2 *

Bits	Name	Description	R/W	Reset
31:20	Reserved	Reserved	R	0
19	C2TJB	Carry register 2 TJBR Counter Carry bit	R	0
18	C2TFC	Carry register 2 TFCS Counter Carry bit	R	0
17	C2TCF	Carry register 2 TXCF Counter Carry bit	R	0
16	C2TOV	Carry register 2 TOVR Counter Carry bit	R	0
15	C2TUN	Carry register 2 TUND Counter Carry bit	R	0
14	C2TFG	Carry register 2 TFRG Counter Carry bit	R	0
13	C2TBY	Carry register 2 TBYT Counter Carry bit	R	0
12	C2TPK	Carry register 2 TPKT Counter Carry bit	R	0
11	C2TMC	Carry register 2 TMCA Counter Carry bit	R	0
10	C2TBC	Carry register 2 TBCA Counter Carry bit	R	0
9	C2TPF	Carry register 2 TXPF Counter Carry bit	R	0
8	C2TDF	Carry register 2 TDFR Counter Carry bit	R	0
7	C2TED	Carry register 2 TEDF Counter Carry bit	R	0
6	C2TSC	Carry register 2 TSCL Counter Carry bit	R	0
5	C2TMA	Carry register 2 TMCL Counter Carry bit	R	0
4	C2TLC	Carry register 2 TLCL Counter Carry bit	R	0
3	C2TXC	Carry register 2 TXCL Counter Carry bit	R	0
2	C2TNC	Carry register 2 TNCL Counter Carry bit	R	0
1	C2TPH	Carry register 2 TPPH Counter Carry bit	R	0
0	C2TDP	Carry register 2 TDRC Counter Carry bit	R	0

* Carry register bits are cleared on carry register write while respective bit is asserted

16.5.3.45 CAM1 - Carry Mask Register 1

Register ID: 0x4E Bits [31:0]

Address Offset: 0x138

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16						
M1 64	M1 127	M1 255	M1 511	M1 1K	M1 MAX	M1 MGV	Reserved														M1 RBY
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
M1 RPK	M1 RFC	M1 RMC	M1 RBC	M1 RXC	M1 RXP	M1 RXU	M1 RAL	M1 RFL	M1 RCD	M1 RCS	M1 RUN	M1 ROV	M1 RFR	M1 RJB	M1 RDR						

When a mask bit is set to zero, the corresponding interrupt bit is allowed to cause interrupt indications to the Network Accelerator. Upon a statistics register overflow, the Carry bit is set in the Network Accelerator Int register, which is then passed on to the PIC.

Bits	Name	Description	R/W	Reset
31	M164	Mask register 1 TR64 Counter Carry bit Mask	R/W	1
30	M1127	Mask register 1 TR127 Counter Carry bit Mask	R/W	1
29	M1255	Mask register 1 TR255 Counter Carry bit Mask	R/W	1
28	M1511	Mask register 1 TR511 Counter Carry bit Mask	R/W	1
27	M11k	Mask register 1 TR1K Counter Carry bit Mask	R/W	1
26	M1MAX	Mask register 1 TRMAX Counter Carry bit Mask	R/W	1
25	M1MGV	Mask register 1 TRMGV Counter Carry bit Mask	R/W	1
24:17	Reserved	Reserved	R/W	
16	M1RBY	Mask register 1 RBYT Counter Carry bit Mask	R/W	1
15	M1RPK	Mask register 1 RPKT Counter Carry bit Mask	R/W	1
14	M1RFC	Mask register 1 RFCS Counter Carry bit Mask	R/W	1
13	M1RMC	Mask register 1 RMCA Counter Carry bit Mask	R/W	1
12	M1RBC	Mask register 1 RBCA Counter Carry bit Mask	R/W	1
11	M1RXC	Mask register 1 RXCF Counter Carry bit Mask	R/W	1
10	M1RXP	Mask register 1 RXPF Counter Carry bit Mask	R/W	1
9	M1RXU	Mask register 1 RXUO Counter Carry bit Mask	R/W	1
8	M1RAL	Mask register 1 RALN Counter Carry bit Mask	R/W	1
7	M1RFL	Mask register 1 RFLR Counter Carry bit Mask	R/W	1
6	M1RCD	Mask register 1 RCDE Counter Carry bit Mask	R/W	1
5	M1RCS	Mask register 1 RCSE Counter Carry bit Mask	R/W	1
4	M1RUN	Mask register 1 RUND Counter Carry bit Mask	R/W	1
3	M1ROV	Mask register 1 ROVR Counter Carry bit Mask	R/W	1
2	M1RFR	Mask register 1 RFRG Counter Carry bit Mask	R/W	1
1	M1RJB	Mask register 1 RJBR Counter Carry bit Mask	R/W	1
0	M1RDR	Mask register 1 RDRP Counter Carry bit Mask	R/W	1

16.5.3.46 CAM2 - Carry Mask Register 2

Register ID: 0x4F Bits [31:0]

Address Offset: 0x13C

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
<i>Reserved</i>														M2 TJB	M2 TFC	M2 TCF	M2 TOV
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
M2 TUN	M2 TFG	M2 TBY	M2 TPK	M2 TMC	M2 TBC	M2 TPF	M2 TDF	M2 TED	M2 TSC	M2 TMA	M2 TLC	M2 TXC	M2 TNC	M2 TPH	M2 TDP		

When a mask bit is set to zero, the corresponding interrupt bit is allowed to cause interrupt indications to the Network Accelerator. Upon a statistics register overflow, the Carry bit is set in the Network Accelerator Int register, which is then passed on to the PIC.

Bits	Name	Description	R/W	Reset
31:20	Reserved	Reserved	R/W	
19	M2TJB	Mask register 2 TJBR Counter Carry bit Mask	R/W	1
18	M2TFC	Mask register 2 TFCS Counter Carry bit Mask	R/W	1
17	M2TCF	Mask register 2 TXCF Counter Carry bit Mask	R/W	1
16	M2TOV	Mask register 2 TOVR Counter Carry bit Mask	R/W	1
15	M2TUN	Mask register 2 TUND Counter Carry bit Mask	R/W	1
14	M2TFG	Mask register 2 TFRG Counter Carry bit Mask	R/W	1
13	M2TBY	Mask register 2 TBYT Counter Carry bit Mask	R/W	1
12	M2TPK	Mask register 2 TPKT Counter Carry bit Mask	R/W	1
11	M2TMC	Mask register 2 TMCA Counter Carry bit Mask	R/W	1
10	M2TBC	Mask register 2 TBKA Counter Carry bit Mask	R/W	1
9	M2TPF	Mask register 2 TXPF Counter Carry bit Mask	R/W	1
8	M2TDF	Mask register 2 TDFR Counter Carry bit Mask	R/W	1
7	M2TED	Mask register 2 TEDF Counter Carry bit Mask	R/W	1
6	M2TSC	Mask register 2 TSCL Counter Carry bit Mask	R/W	1
5	M2TMA	Mask register 2 TMCL Counter Carry bit Mask	R/W	1
4	M2TLC	Mask register 2 TLCL Counter Carry bit Mask	R/W	1
3	M2TXC	Mask register 2 TXCL Counter Carry bit Mask	R/W	1
2	M2TNC	Mask register 2 TNCL Counter Carry bit Mask	R/W	1
1	M2TPH	Mask register 2 TPFH Counter Carry bit Mask	R/W	1
0	M2TDP	Mask register 2 TDRP Counter Carry bit Mask	R/W	1

16.5.4 PCS Register Descriptions

16.5.4.1 PCS Register Summary Table

Table 16-7 presents registers for PCS-specific control functions:

Table 16-7. PCS-Specific Registers

Register ID	Address Offset	Register
0x00	PHY 27 through 30, as needed	Control
0x01		Status
0x02/0x03		Reserved
0x04		AN SGMII Advertisement
0x05		AN Link Partner Base Page Ability
0x06		AN Expansion
0x07		AN Next Page Transmit
0x08		AN Link Partner Ability Next Page
0x0F		Extended Status
0x10		Jitter Diagnostics
0x11		TBI Control

16.5.4.2 Control

Register ID: 0x00 Bits [15:0]

Address Offset: 0x00

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PHY Reset	Loop Back	Reserved	AN Enable	Reserved	Reset AN										Reserved

Default value: 0x0000

Bits	Name	Description	R/W	Reset
15	PHY RESET	Setting this bit will cause the PETEX, PEREX, and PEANX sub-modules in the M-SGMII core to be reset. This bit is self-clearing. Its default is '0'.	RW, SC	0
14	LOOP BACK	Setting this bit will cause the transmit outputs of the M-SGMII to be connected to the receive inputs. Clearing this bit results in normal operation. Its default is '0'. Note: This bit does not affect the clock signals, which for loop back must be handled external to the core.	RW	0
13	Reserved	Reserved		0
12	AUTO-NEGOTIATION ENABLE	Setting this bit enables the Auto-negotiation process. Its default is '0'.	RW	0
11:10	Reserved	Reserved		0
9	RESET AUTO-NEGOTIATION	Setting this bit causes the Auto-negotiation process to restart. This action is only available when Auto-Negotiation has been enabled. Its default is '0'.	RW, SC	0
8:0	Reserved	Reserved		0

16.5.4.3 Status**Register ID:** 0x01 **Bits [15:0]****Address Offset:** 0x04

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
					Reserved		Extend Status		No Pre	AN Done	Remote Fault	AN Ability	Link Status		Extend Ability

Default value: 0x0001

Bits	Name	Description	R/W	Reset
15:9	Reserved	Reserved		0
8	EXTENDED STATUS	This bit indicates that PHY status information is also contained in Register 15 – EXTENDED STATUS. Returns ‘1’ on read.	RO	0
7	Reserved	Reserved		0
6	MF PREMABLE SUPPRESSION ENABLE	This bit indicates whether the PHY is capable of handling MII Management Frames without the 32-bit preamble field. Returns ‘1’ indicating support for suppressed preamble MII Management Frames.	RO	0
5	AUTO-NEGOTIATION COMPLETE	When ‘1’, this bit indicates that the Auto-negotiation process has completed. This bit returns ‘0’ when either the Auto-negotiation process is underway or when the Auto-negotiation function is disabled. Its default is ‘0’.	RO	0
4	REMOTE FAULT	When ‘1’, a remote fault condition has been detected between the M-SGMII and the PHY. This bit latches high in order for software to detect the condition. Each read of the STATUS register clears this bit. Its default is ‘0’.	RO/LH	0
3	AUTO-NEGOTIATION ABILITY	When ‘1’, this bit indicates that the M-SGMII has the ability to perform Auto-negotiation. Returns ‘1’ on read.	RO	0
2	LINK STATUS	When ‘1’, this bit indicates that a valid link has been established between the M-SGMII and the PHY. When ‘0’, no valid link has been established. This bit latches low to allow software polling to detect a failure condition. Its default is ‘0’.	RO/LL	0
1	Reserved	Reserved		0
0	EXTENDED CAPABILITY	This bit indicates that the M-SGMII contains the extended set of registers (those beyond CONTROL and STATUS). Returns ‘1’ on read.	RO	1

16.5.4.4 AN SGMII Advertisement

Register ID: 04 Bits [15:0]

Address Offset: 0x10

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Link Up	Ack	Reserved	Full Duplex	Link Speed											Reserved

Default value: 0x0000

Bits	Name	Description	R/W	Reset																		
15	LINK UP	When the M-SGMII is integrated into a PHY and is communicating with the SGMII module in a MAC, assertion of this bit indicates that the link is up. When the M-SGMII is integrated into a MAC, such as the PE-MCXMAC, this bit must be written '0'.	R/W	0																		
14	ACK	Write '1'. Assert Acknowledge bit in the Advertisement word.		0																		
13	Reserved	This bit must be written '0' for correct M-SGMII operation.	RO	0																		
12	FULL DUPLEX	When the M-SGMII is integrated into a PHY and is communicating with the SGMII module in a MAC, assertion of this bit indicates that the link is transferring data in full duplex mode. When the M-SGMII is integrated into a MAC, such as the PE-MCXMAC, this bit must be written '0'.	R/W	0																		
11:10	LINK SPEED	<p>When the M-SGMII is integrated into a PHY and is communicating with the SGMII module in a MAC, assertion of these 2 bits indicate the speed that the link is transferring data as per the table below. Note that external pins msgmii_speed[1:0] must be set appropriately. When the M-SGMII is integrated into a MAC, such as the PE-MCXMAC, these bits must be written '00'.</p> <p>Link Speed Encoding:</p> <table border="1"> <thead> <tr> <th colspan="2">Link Speed</th><th>Capability</th></tr> <tr> <th>[11]</th><th>[10]</th><th></th></tr> </thead> <tbody> <tr> <td>1</td><td>1</td><td>Reserved</td></tr> <tr> <td>1</td><td>0</td><td>1000 Mbps</td></tr> <tr> <td>0</td><td>1</td><td>100 Mbps</td></tr> <tr> <td>0</td><td>0</td><td>10 Mbps</td></tr> </tbody> </table>	Link Speed		Capability	[11]	[10]		1	1	Reserved	1	0	1000 Mbps	0	1	100 Mbps	0	0	10 Mbps	R/W	0
Link Speed		Capability																				
[11]	[10]																					
1	1	Reserved																				
1	0	1000 Mbps																				
0	1	100 Mbps																				
0	0	10 Mbps																				
9:0	Reserved	These bits must always be written '0000000001' for correct M-SGMII operation.	R/W	0																		

16.5.4.5 AN SGMII Link Partner Base Page Ability

Register ID: 0x05 Bits [15:0]

Address Offset: 0x14

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Link Up	Ack	Reserved	Full Duplex	Link Speed											Reserved

Default value: 0x0000

Bits	Name	Description	R/W	Reset
15	LINK UP	When the M-SGMII is integrated into a MAC, such as the PE-MCXMAC, and is communicating with another SGMII module in the PHY, assertion of this bit indicates that the link is up. When the M-SGMII is integrated into a PHY, this bit is invalid.	RO	0
14	ACK (Reserved)	Ignore on read.	RO	0
13	Reserved	Reserved		0
12	FULL DUPLEX	When the M-SGMII is integrated into a MAC, such as the PE-MCXMAC, and is communicating with another SGMII module in the PHY, assertion of this bit indicates that the link is transferring data in full duplex mode. When the M-SGMII is integrated into a PHY, this bit is invalid.	RO	0
11:10	LINK SPEED	When the M-SGMII is integrated into a MAC, such as the PE-MCXMAC, and is communicating with another SGMII module in the PHY, assertion of these 2 bits indicate the speed that the link is transferring data as per Table 2. Note that the external msgmii_speed[1:0] pins must be set appropriately. When the M-SGMII is integrated into a PHY, these bits are invalid.	RO	0
9:0	Reserved	Reserved		0

16.5.4.6 AN Expansion

Register ID: 0x06 Bits [15:0]

Address Offset: 0x18

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>												NP Able	Page Received	Reserved	

Default value: 0x0000

Bits	Name	Description	R/W	Reset
15:0	Reserved	Reserved		0
2	NEXT PAGE ABLE	When '1', indicates that the local device supports the Next Page function. Returns '1' on read.	RO	0
1	PAGE RECEIVED	When '1', indicates that a new page has been received and stored in the applicable AN LINK PARTNER ABILITY or AN NEXT PAGE register. This bit latches high for detection by software when polling. The bit is cleared on a read to the register.	RO, LH	0
0	Reserved	Reserved		0

16.5.4.7 AN Next Page Transmit

Register ID: 0x07 Bits [15:0]

Address Offset: 0x1C

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Next Page	Ack	Msg Page	Ack2	Toggle	Message / Unformatted Code field										

Default value: 0x0000

Bits	Name	Description	R/W	Reset
15	NEXT PAGE	Assert this bit to indicate additional Next Pages to follow. Clear the bit to indicate the last page. [Reference MII bit 7.15]	R/W	0
14	ACK (Reserved)	Write '1'. Assert Acknowledge bit in the Advertisement word.. [Reference MII bit 7.14]	RO	0
13	MESSAGE PAGE	Assert this bit to indicate a Message Page. Clear the bit to indicate an Unformatted Page. [Reference MII bit 7.13]	R/W	0
12	ACKNOWLEDGE 2	Used by the Next Page function to indicate that the device has the ability to comply with the message. Assert this bit if the local device will comply with the message. Clear the bit if the local device cannot comply with message. [Reference MII bit 7.12]	R/W	0

Bits	Name	Description	R/W	Reset
11	TOGGLE	Used to ensure synchronization with the Link Partner during Next Page exchange. This bit always takes the opposite value to the Toggle bit of the previously exchanged Link Code Word. The initial value in the first Next Page transmitted is the inverse of bit 11 in the base Link Code Word. [Reference MII bit 7.11]	RO	0
10:0	MESSAGE / UNFORMATTED CODE FIELD	Message pages are formatted pages that carry a predefined Message Code, which is enumerated in IEEE 802.3u/Annex 28C. Unformatted Code Fields take an arbitrary value. [Reference MII field 7.10:0]	R/W	0

NETLOGIC CONFIDENTIAL

16.5.4.8 AN Link Partner Ability Next Page

Register ID: 0x08 Bits [15:0]

Address Offset: 0x20

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Next Page	Ack	Msg Page	Ack2	Toggle											Message / Unformatted Code field

Default value: 0x0000

Bits	Name	Description	R/W	Reset
15	NEXT PAGE	The Link Partner asserts this bit to indicate additional Next Pages to follow. When '0', indicates last Next Page from link partner.	RO	0
14	ACK	Write '1'. Assert Acknowledge bit in the Advertisement word. [Reference MII bit 8.14]	RO	0
13	MESSAGE PAGE	When '1', indicates Message Page. When '0', indicates Unformatted Page.	RO	0
12	ACKNOWLEDGE 2	Indicates Link Partner's ability to comply with the message. When '1', Link Partner will comply with message. When '0', Link Partner cannot comply with message.	RO	0
11	TOGGLE	Used to ensure synchronization with the Link Partner during Next Page exchange. This bit always takes the opposite value to the Toggle bit of the previously exchanged Link Code Word. The initial value in the first Next Page transmitted is the inverse of bit 11 in the base Link Code Word.	RO	0
10:0	MESSAGE / UNFORMATTED CODE FIELD	Message pages are formatted pages that carry a predefined Message Code, which is enumerated in IEEE 802.3u/Annex 28C. Unformatted Code Fields take an arbitrary value.	RO	0

16.5.4.9 Extended Status

Register ID: 0x0F Bits [15:0]

Address Offset: 0x3C

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1000 X Full	1000 X Half	1000 T Full	1000 T Half												Reserved

Default value: 0xA000

Bits	Name	Description	R/W	Reset
31:16	Reserved	Reserved		0
15	1000BASE-X FULL-DUPLEX	When '1', indicates PHY can operate in 1000BASE-X Full- Duplex mode. When '0', indicates PHY cannot operate in this mode. Returns '1' on read.	RO	1
14	1000BASE-X HALF-DUPLEX	When '1', indicates PHY can operate in 1000BASE-X Half- Duplex mode. When '0', indicates PHY cannot operate in this mode. Returns '0' on read.	RO	0
13	1000BASE-T FULL-DUPLEX	When '1', indicates PHY can operate in 1000BASE-T Full- Duplex mode. When '0', indicates PHY cannot operate in this mode. Returns '1' on read.	RO	1

Bits	Name	Description	R/W	Reset
12	1000BASE-T HALF-DUPLEX	When '1', indicates PHY can operate in 1000BASE-T Half-Duplex mode. When '0', indicates PHY cannot operate in this mode. Returns '0' on read.	RO	0
11:0	Reserved	Reserved		0

16.5.4.10 Jitter Diagnostics

Register ID: 0x10 Bits [15:0]

Address Offset: 0x40

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Jitter Enable	Jitter Select	Reserved	Custom Jitter Pattern												

Default value: 0x0000

Bits	Name	Description	R/W	Reset
15	JITTER DIAGNOSTIC ENABLE	Set this bit to enable the M-SGMII to transmit the jitter test patterns defined in IEEE 802.3z 36A. Clear this bit to enable normal transmit operation. Its default is '0'.	RO	0
14:12	JITTER PATTERN SELECT	Selects the jitter pattern to be transmitted in diagnostics mode. Encoding of this field is shown in Table 3. Its default is '000'. See Table 16-8 for Jitter Pattern Select Encoding	RO	0
11:10	Reserved	Reserved	RO	0
9:0	CUSTOM JITTER PATTERN	Used in conjunction with JITTER PATTERN SELECT and JITTER DIAGNOSTIC ENABLE. Set this field to the desired custom pattern, which will be transmitted continuously. Its default is '0x000'.	RO	0

Table 16-8. Jitter Pattern Select Encoding

User Defined Custom Pattern	0	0	0
Annex 36A Defined High Frequency 10101010101010101010...	0	0	1
Annex 36A Defined Mixed Frequency 11111010110000010100...	0	1	0
Custom Defined Low Frequency 11111000001111100000...	0	1	1
Random Jitter Pattern	1	0	0
Annex 36A Defined Low Frequency 11111000001111100000...	1	0	1
Reserved	1	1	0
Reserved	1	1	1

16.5.4.11 TBI Control

Register ID: 0x11 **Bits [31:0]**

Address Offset: 0x44

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Soft Reset	Short Timer	Disable Rx Dis	Disable Tx Dis	Rsvd	AN Sense	<i>Reserved</i>		Clock Select	MII Mode	<i>Reserved</i>		Enable Wrap	Comma Detect		

Default value: 0x0000

Bits	Name	Description	R/W	Reset
31:16	Reserved	Reserved		0
15	SOFT RESET	This bit resets the functional modules in the M-SGMII. Clear it for normal operation. Its default is '0'.	R/W	0
14	SHORTCUT LINK TIMER	Set this bit to reduce the amount of simulation time needed to time the 1.6ms Link Timer. Clear it for normal operation. Its default is '0'.	R/W	0
13	DISABLE RECEIVE RUNNING DISPARITY	Set this bit to disable the running disparity calculation and checking in the receive direction. This bit must be '0' for correct M-SGMII operation. Its default is '0'.	R/W	0
12	DISABLE TRANSMIT RUNNING DISPARITY	Set this bit to disable the running disparity calculation and checking in the transmit direction. This bit must be '0' for correct M-SGMII operation. Its default is '0'.	R/W	0
11:10	Reserved	Reserved		0
9	AN Sense			0
8	AUTO-NEGOTIATION SENSE	Set this bit to allow the Auto-Negotiation function to sense either a Gigabit MAC in Auto-Negotiation bypass mode or an older Gigabit MAC without Auto-Negotiation capability. When sensed, Auto-Negotiation Complete will become true; however Page Received will be low, indicating no page was exchanged. Management can then act accordingly. Clear this bit when IEEE 802.3z Clause 37 behavior is desired, which results in the link not coming up. Its default is '0'.	R/W	0
7:6	Reserved	Reserved		0
5	RECEIVE CLOCK SELECT	Set this bit to configure the M-SGMII to accept a 125 MHz Receive Clock from the SERDES PHY. The 125 MHz clock should be physically connected to 'PMA Receive Clock 0'. Clear this bit to allow the M-SGMII to accept dual split-phase 62.5 MHz Receive Clocks. This bit must be '0' for correct M-SGMII operation. Its default is '0'.	R/W	0
4	GMII MODE	When cleared, this bit defines the M-SGMII as being in 1000BASE-X SERDES mode. This bit must be '0' for correct M-SGMII operation. Its default is '0'.	R/W	0
3:2	Reserved	Reserved		0
1	ENABLE WRAP	Set this bit to configure the SERDES in loop-back mode. Clear this bit to permit normal operation. Its default is '0'.	R/W	0
0	ENABLE COMMA DETECT	Set this bit to allow the SERDES PHY to perform code group alignment based upon the detection of a comma. Its default is '0'.	R/W	0



Chapter 17 XAUI Interface

(XLS616, XLS608, XLS416 and XLS408)

17.1 Introduction

The XLS616, XLS608 and XLS416 provides two optional XAUI interfaces, and the XLS408 provides one optional XAUI interface which share I/O signal pins with the SGMII network ports. (Other XLS devices do not offer the XAUI configuration option.)

This chapter covers the 4-Lane XAUI 10-Gb Ethernet Interface. It provides the following information:

- Overview of interface capabilities
- Theory of Operation
 - description of interface position within the chip architecture
 - operational description
 - device modes/states/access model
 - interrupt handling
 - statistics monitoring
- Programming Model
 - startup/initialization
 - bringup requirements
- XAUI Registers
 - register descriptions
- XAUI Statistics Registers Summary
 - statistics register descriptions
- XAUI PCS Registers

17.1.1 XAUI Interface Selection and Configuration

Either or both optional XAUI ports can be configured to replace the SGMII quads at power-up reset time, as follows:

- The XAUI_0 port can be configured to replace the SGMII_0-3 quad by tying the IO_AD[7] pin high at reset. When IO_AD[7] is strapped low, the SGMII_0-3 quad is selected.
- The XAUI_1 port can be configured to replace the SGMII_4-7 quad by tying the IO_AD[6] pin high. When IO_AD[6] is strapped low, the SGMII_0-3 quad is selected.

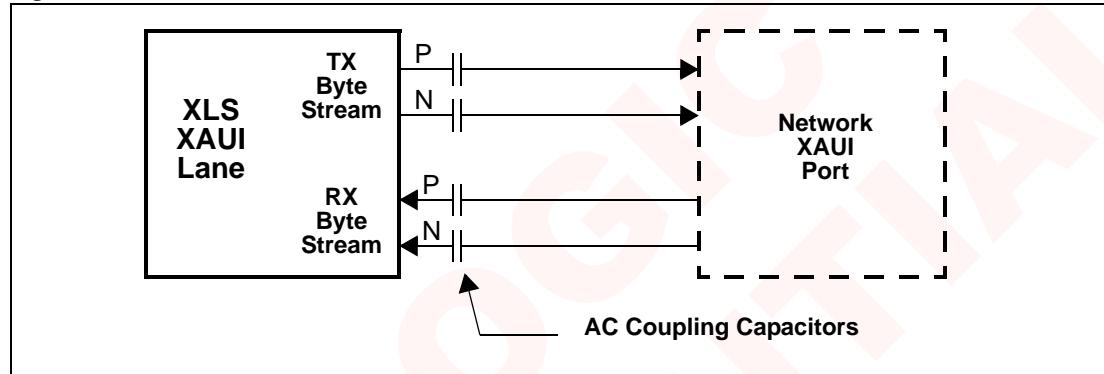
Once a XAUI port is selected, all of the related SGMII signal pins/pads are replaced with the equivalent XAUI signal pins/pads.

17.2 XAUI Overview

The XAUI ports conform to the IEEE 802.3ae 10-Gb Ethernet specification and consists of four lanes (16 total signals), each of which has an Tx pair and an Rx pair. In each the XAUI lane, a Tx and Rx pair provides single serial differential signaling carrying 8b/10b encoded data operating at 3.125 Gbps.

A XAUI port (or link) consists of four XAUI lanes. Each XAUI lane is comprised of a differential pair for Rx and a differential pair for Tx signal pins, which can carry 8b/10b encoded data simultaneously. Thus, one XAUI lane is comprised of four traces. [Figure 17-1](#) shows a diagram of a XAUI lane.

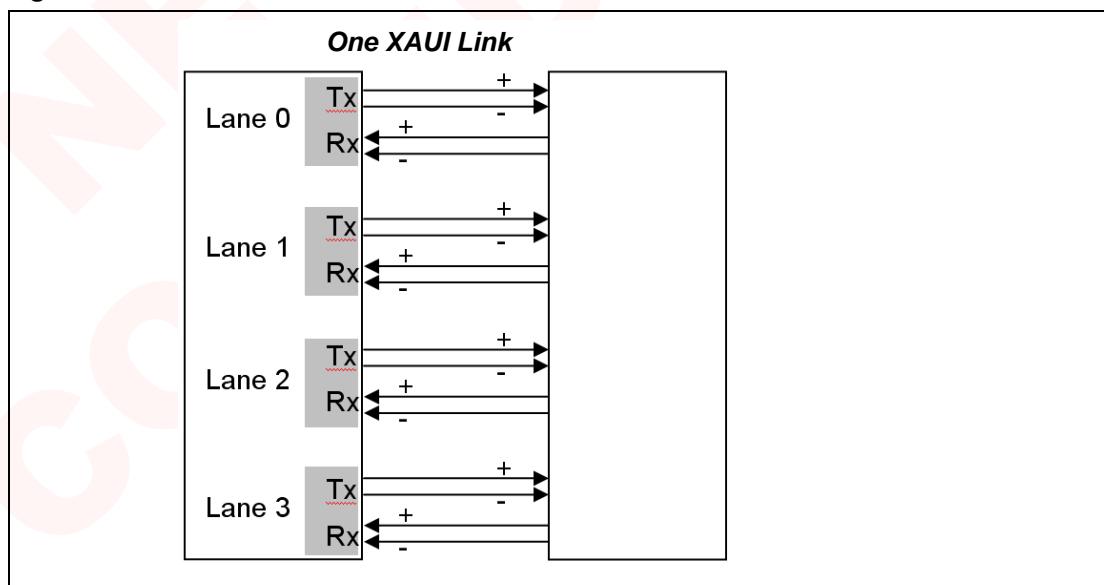
[Figure 17-1. XAUI Lane](#)



The XAUI specification uses the term “byte stream” to describe one half of a data lane (i.e., one differential signal pair). An Rx pair or Tx pair operates at a nominal rate of 2.5 Gbps in the 8-bit domain (3.125 Gbaud in the 10-bit domain).

Four XAUI lanes are used to convey a total of 16-bit self-clocking data and control, resulting in a 10 Gbps effective data rate. [Figure 17-2](#) shows a diagram of a XAUI link.

[Figure 17-2. XAUI Link](#)



The XAUI implementation includes IEEE 802.3ae Clause 45 support to provide control and status through the MDIO interface.

In the Transmit direction, the XAUI module takes the Transmit data stream, encodes it into 10-bit symbols that include clock information and then serializes the data.

In the Receive direction, the data is de-serialized, and the 10-bit symbols are decoded and converted into the into 8-bit symbol Receive signal set and the recovered data clock.

17.2.1

XAU^I Interface Features

The XLS616, XLS608, XLS416 and XLS408 XAU^I implementation is a full-featured interface capable of full-duplex operation at a total of 10 Gbps per four-lane XAU^I link. The primary features are:

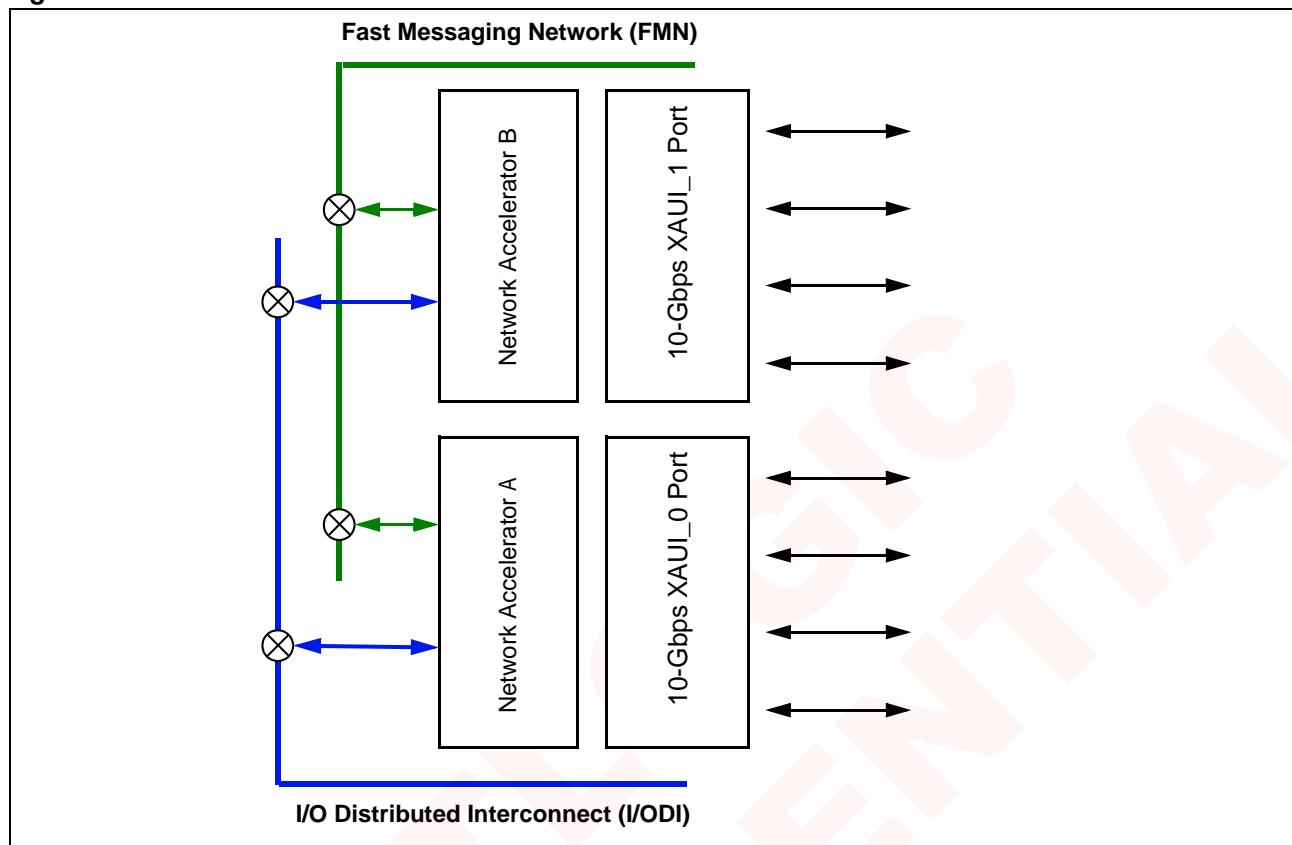
- Full compliance with IEEE 802.3ae
- Support for all mandatory and optional features in IEEE 802.3ae clause 47 & 48
- IEEE 802.3ae- clause 45 MDIO interface
- IEEE 802.3ae- clause 48 state machines
- Pseudo-random idle insertion (PRBS Polynomial X₇ + X₃ + 1)
- 156.25MHz clock frequency
- Single-width DDR or double-width SDR XAU^I interface
- Optional comma alignment function
- Low power mode
- PHY-XS and DTE-XS loopback
- IEEE 802.3ae- annex 48A jitter test pattern support
- IEEE 802.3 clause 36 8b/10b encoding compliance
- Tolerance of lane skew up to 16 ns (50 UI)
- IEEE 802.3 PICs Compliance Matrix

17.2.2

XAU^I Architecture

Under the International Standards Organization's Open Systems Interconnection (OSI) model, Ethernet is fundamentally a Layer 2 protocol. An Ethernet Physical layer device (PHY), which corresponds to Layer 1 of the OSI model, connects the media (optical or copper) to the MAC layer, which corresponds to OSI Layer 2.

Figure 17-3 shows the Fast Messaging Network and the IO Distributed Interconnect relative to their relationship to the XAU^I ports.

Figure 17-3. XAUI Ports

17.2.2.1 XAUI Functional Description

XAUI employs the same robust 8b/10b transmission code as 1000BASE-X. As there are more than enough code possibilities for mapping an 8-bit word, some code groups are used for control signaling (SOF, EOF, IDLE, link configurations, etc.). Control words during the Inter-Packet Gap (IPG) and idle times ensure word and lane alignment, without upper layer support requirement.

XAUI compensates for differences in clock domains between each side of the link. For increased signal integrity, the XLS XAUI implementation isolates clock domains between the four Rx lanes, a Tx domain, and the MDIO host clock. For each of the six domains there is a separate reset signal. Rx and Tx reset signals are derived identically and are released within a cycle of each other.

Receive data on the XAUI interface is edge-aligned with the receive clock. Transmit XAUI PMA interface data is similarly edge-aligned with the transmit clock.

Primary functionality is described in the following sections.

Transmit Direction

Function highlights are as follows:

- **Data/Idle Encoding and Transmission:** The Encoding/Transmit path is double-width. Encoding occurs per-lane. The transmit source state diagram is as shown in IEEE 802.3ae, Figure 48-6. Frames must begin with a start character and end with a terminate character. Violations of the XAUI interface protocol are identified and set the transmit

fault error flag in the MMD registers. The encoded data is forwarded on the outputs without modification.

- **Sequence-Ordered Set Transmission:** As specified in IEEE 802.3ae clause 48, the M-XGXS cannot transmit sequence ordered sets as they arrive. Rather, they must be inserted within the Inter Packet Gap (IPG) at very specific times. Therefore, when a sequence-ordered set arrives, it is held in memory until a transmission opportunity arrives. The implementation has storage for one sequence ordered set. If more than one message arrives before one can be transmitted, the most recent message is forwarded.
- **Test Mode:** Input data is ignored and specific test patterns are transmitted, as specified in IEEE 802.3ae Annex 48A Jitter Test Pattern support. The XAUI PMA interface is double width, since all PMA SerDes Solutions accommodate 20-bit serialization.

Receive Direction

Function highlights are as follows:

- **Comma Alignment:** The comma alignment logic aligns the incoming non-aligned data to 10-bit word boundaries (by aligning to the unique bit sequence--7'b0011111 or 7'b1100000--contained in the Idle Control Characters K28.1 and K28.7. See IEEE 802.3, Section 36.2.4.9). Comma alignment is only enabled during the synchronization process. (The process of comma alignment is described in IEEE 802.3ae Figure 48-7.)
- **Lane Synchronization / Phase Alignment:** Incoming lane data is generated from a common crystal; therefore, there is no clock drift between the lanes of data. However there can be significant lane-to-lane skew (up to 13.5 ns), making phase alignment of the four XAUI lanes unknown. The synchronization function phase-aligns the four lanes of data to a common clock (the lane 0 recovered clock).
Phase alignment is performed only on initialization and only if excessive bit errors occur. (The process of enabling and disabling synchronization is described in IEEE 802.3ae Figure 48-7.)
- **Lane Alignment:** Even after lane synchronization (phase alignment), there can still be significant lane-to-lane skew between the four lanes of data. Thus, at the input of the lane alignment logic, all four data lanes are synchronized to a common Rx clock. Lane alignment is accomplished by observing the special alignment Idle character K28.4.
Lane alignment is only performed following a successful synchronization process. If lane-to-lane skew is identified, a Loss Of Alignment (LOA) occurs (due to a high density of lane skew errors) and the synchronization process is initiated, after which the lane alignment process is begun.
- **8b/10b Decoding:** The decoders operate per-lane, decoding data received on the XAUI bus and generating XAUI data and control bits for each lane. If invalid codes are received on the XAUI bus, then un-encoded XAUI output data is replaced with an error byte. If the Receive XAUI PMA SerDes has lost lock for any lane, then all incoming data is ignored and local faults are transmitted to the XAUI bus, per IEEE 802.3 Figure 48-7.

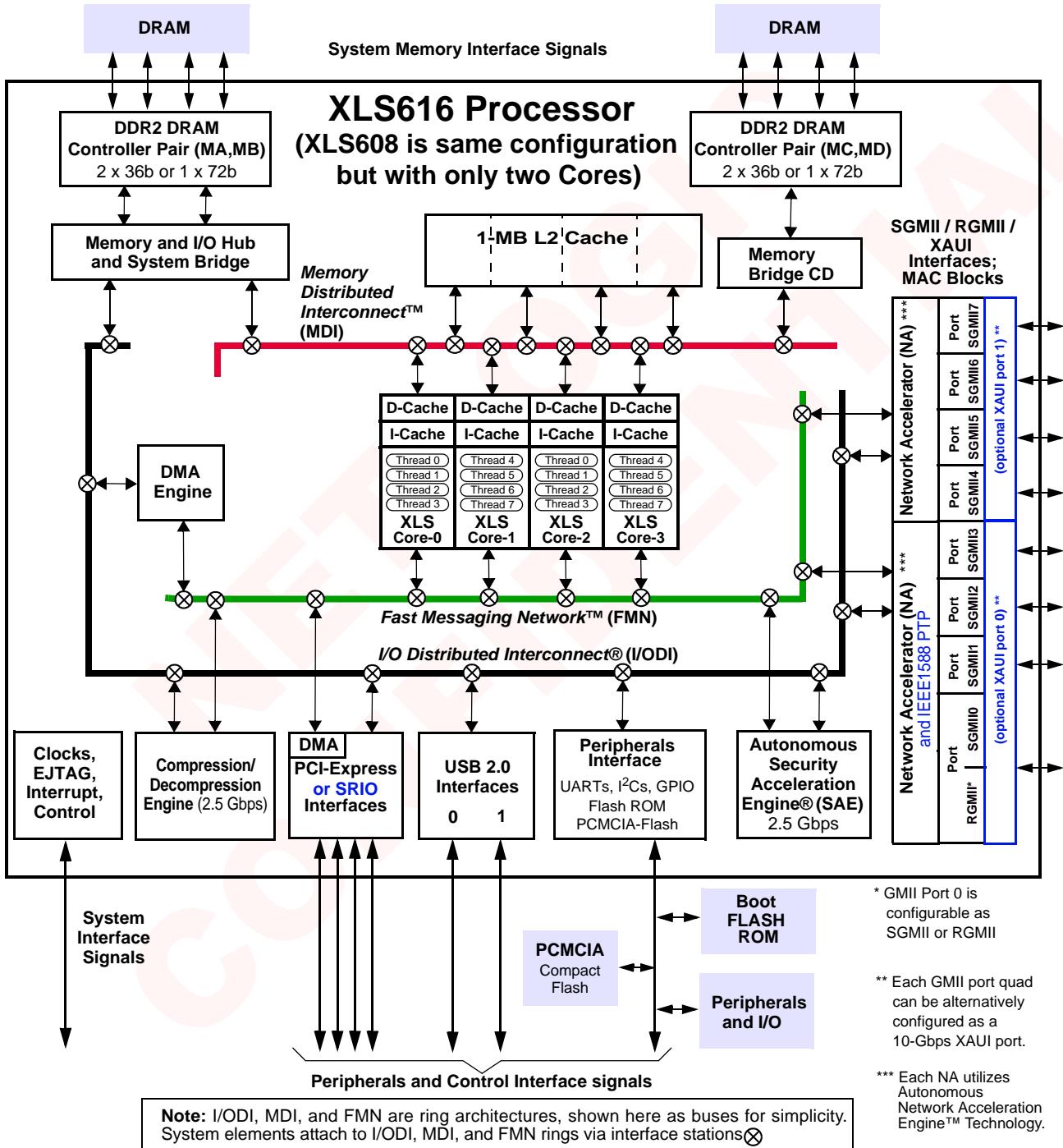
Link Management

The XAUI implementation includes the registers described in IEEE 802.3ae clause 45, section 45.2.5. The device address is equal to 0x05. This implementation is a generic MDIO-manageable device (MMD) when connected to a Station Management Entity (STA).

17.3 Theory of Operation

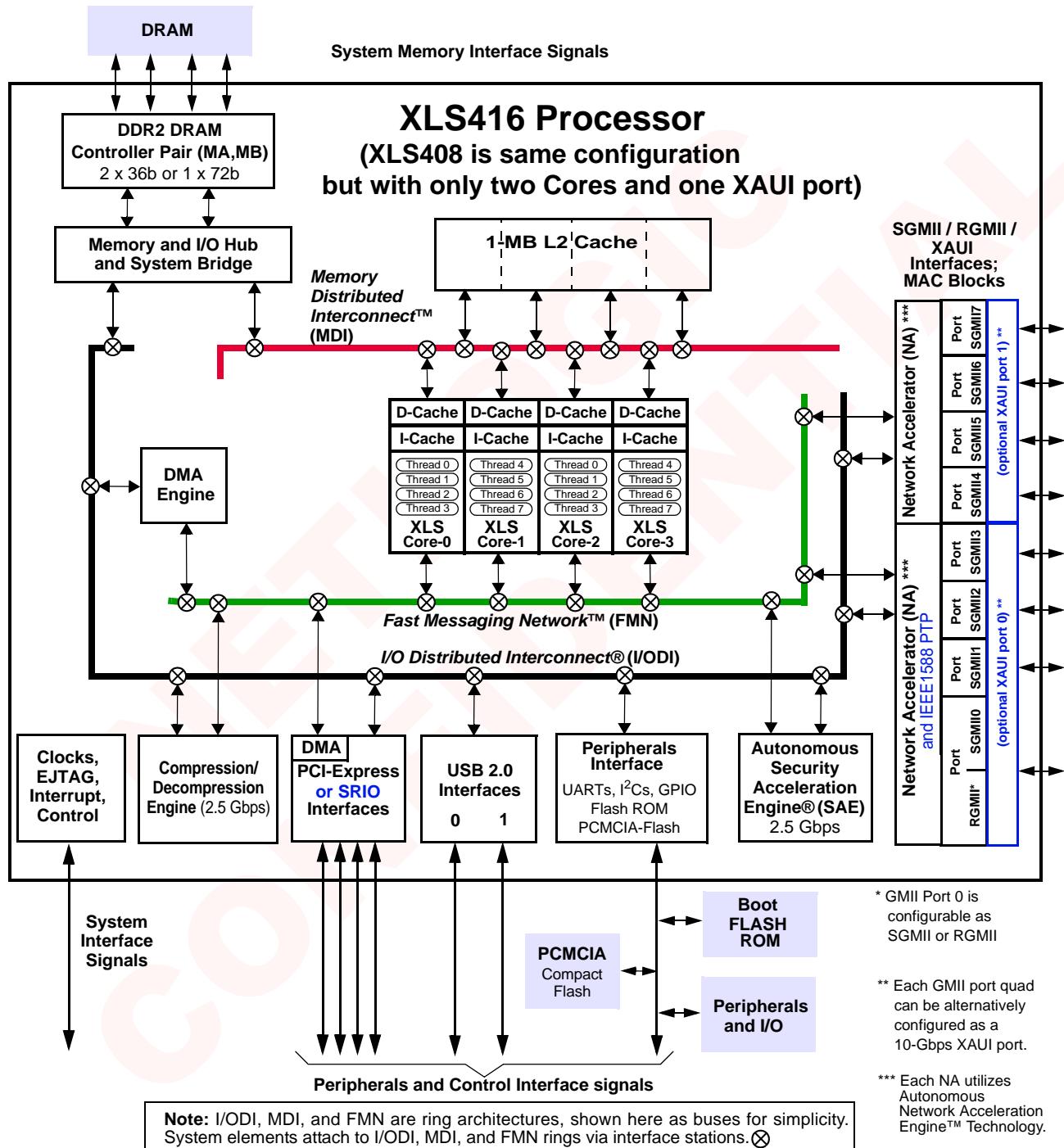
This section discusses architecture and operating details of the XAUI Controllers and Interface. As shown in Figure 17-4 and Figure 17-5 the Processors support eight GMACs, in two groups of four network ports each with its own Network Accelerator. Each quad is individually configurable as a 4-lane XAUI port.

Figure 17-4. XLS616 and XLS608 Block Diagram



The XLS416 is identical to the XLS616 except for the number of DRAM channels supported. Figure 17-5 presents an example hardware block diagram for the quad-core XLS416 device. The XLS408 is the same as the XLS416, except it has only two cores and only one XAUI interface on SGMII_0-3.

Figure 17-5. XLS416 and XLS408 Block Diagram



17.3.1

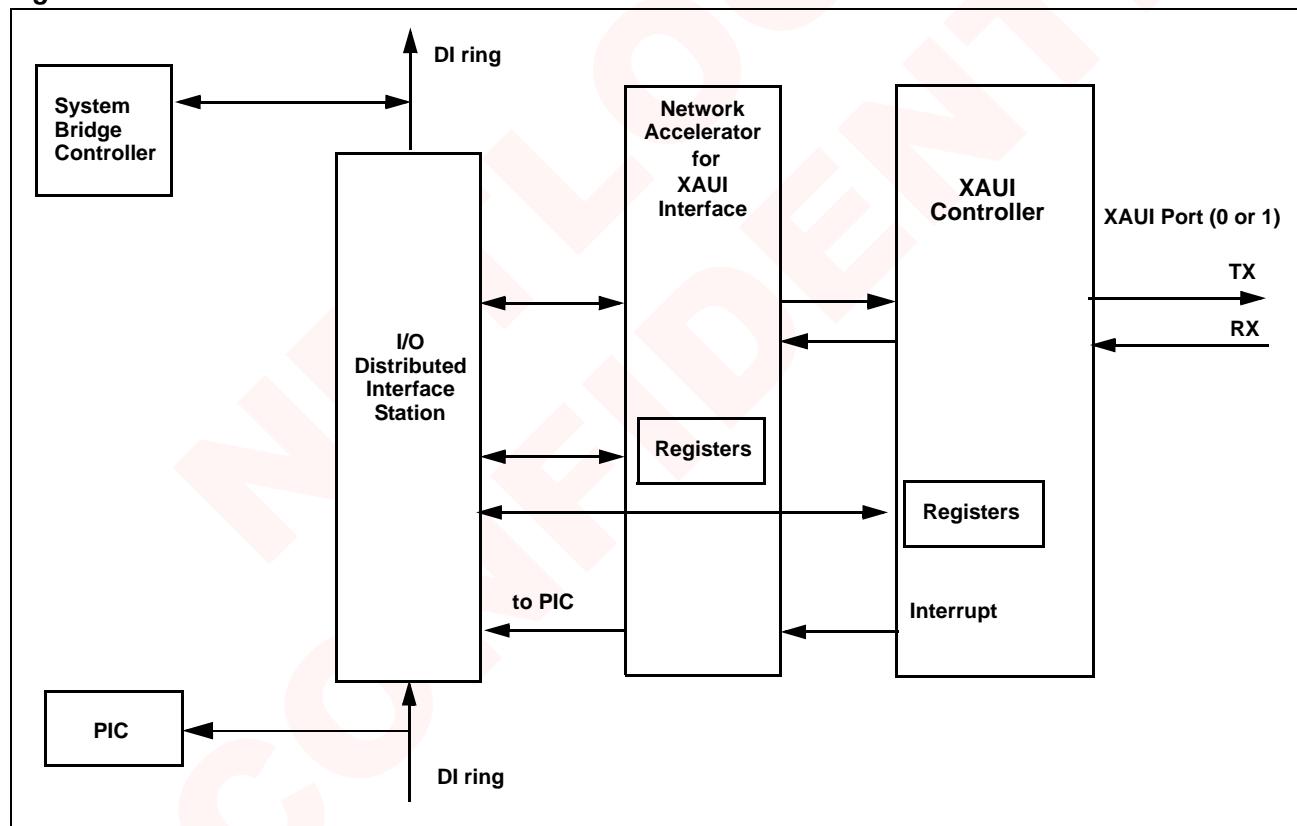
Position of XAUI Interface Within the System Architecture

Key point for understanding the system-level architectural model are:

- Each XAUI Network Interface interacts with its companion Network Accelerator for access to the XLS internal architecture.
- For Ports XA and XB, pin-strapping determines whether a port uses the XAUI interface
- Certain registers in the Network Accelerator affect or control XAUI interface operation. For register details, please refer Chapter 13, “Networking Accelerators”.
- The registers for the Network Accelerator handling an XAUI interface lie in the same group base-addressing region as the interface.
- A module outside the XAUI interface monitors statistics for traffic between the XAUI interface and its Network Accelerator, and its statistics registers are accessed through the Network Accelerator.
- XAUI interrupts are routed through its Network Accelerator to the PIC (Programmable Interrupt Controller)

The figure below shows the general architectural connection of the Network Accelerator, the XAUI, and external devices.

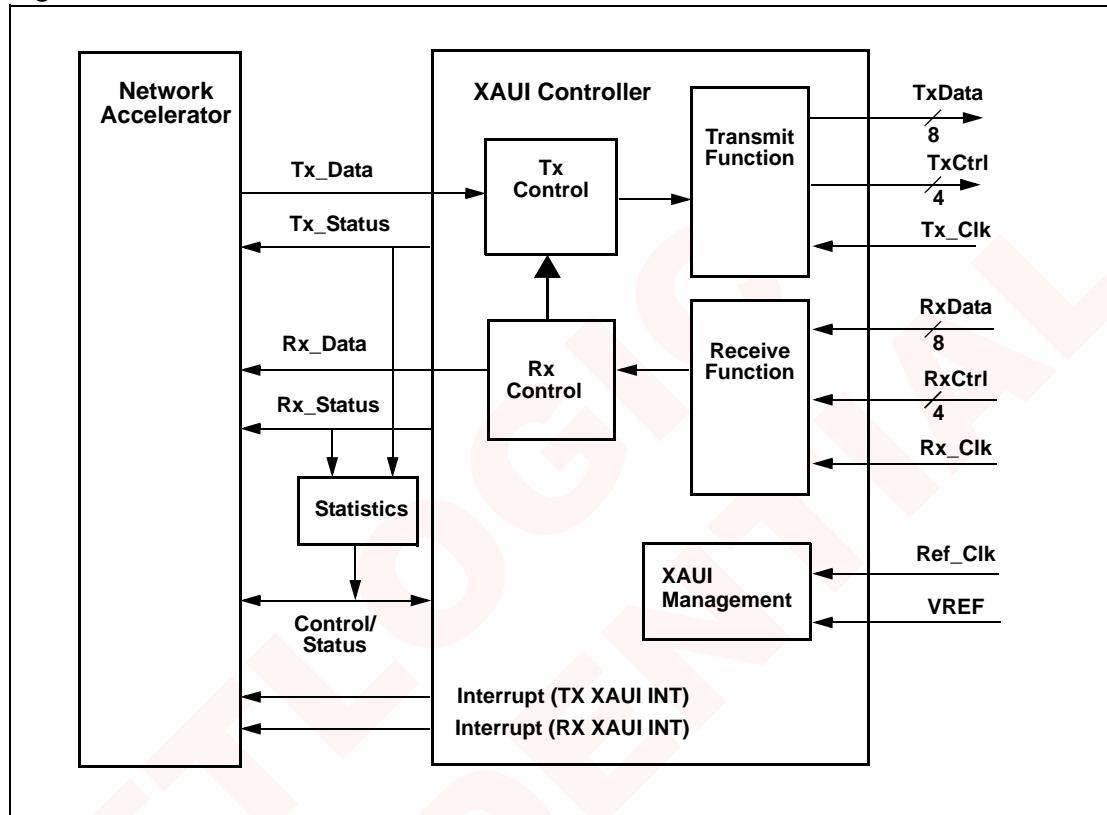
Figure 17-6. XAUI Interface Access to XLS Internal Architecture



17.3.2 XAUI Block Diagram

Figure 17-7 shows a block diagram of an XLS XAUI Interface.

Figure 17-7. Internal Details of an XLS XAUI Port



Details of control of the functional blocks is given in the register descriptions section following.

17.4 Programming Model

This section discusses programmatic set up for operation of the interface, and programmatic access to the interface.

There are two main areas in which setup must be done which are described in the following sections:

- System-level setup
- Interface-level setup

Control related to the XAUI interfaces exists in several places.

1. Control registers for each XAUI interface reside in an address region assigned to that interface.
 - a. XAUI_0 Interface registers should be used when the XAUI_0 port is enabled.
 - b. XAUI_1 Interface registers should be used when the XAUI_1 port is enabled.
 In each case, the registers control the internal function of the interface. For base address information, refer to Chapter 8.
2. Data flow for the XAUI interfaces passes through Network Accelerators. Control registers affecting that data flow reside in the Network Accelerators and must be initialized as needed. The Network Accelerator's registers are accessed using the Network Accelerator register offsets applied in the XAUI interface's addressing space. See [Section 17.5.2, "Register Access"](#) discussion.
3. Each XAUI interface's interrupts are output to the companion Network Accelerator, which contains interrupt masking that controls which of its associated XAUI interface's interrupts are aggregated into a single XAUI interrupt that goes to the PIC. The user must initialize this masking in the Network Accelerator as needed.
4. Programmable parameters within the XAUI interface are identified in this chapter.
5. Programmable parameters within the Network Accelerator are listed in register descriptions in the chapter describing that module.

17.4.1

System Setup for Startup and Initialization

The inward side of the interface is tightly integrated into the XLS processor's I/O Distributed Interconnect, through its FMN and I/O DI stations. Setup areas are in addressing, data transport credits, interrupts, and statistics monitoring.

17.4.1.1

System Bridge Controller Setup

The SBC resides in the XLS's Memory and Distributed Interconnect Hub. (See Chapter 8, External Memory and I/O Configuration.) This handles device-internal address mappings.

The XAUI interface access is affected by the address set in the System Bridge Controller's XLS_IO_BAR register, which defines the base address for the entire Peripheral and I/O Configuration region. All internal peripheral configuration header registers and control and status registers are in this region. Each XAUI Port (0 & 1) has its own group base offset for access to that port's registers.

17.4.1.2

Credits Setup for Network Accelerator

Each XAUI interface sends data through its Network Accelerator onto the internal XLS I/O Distributed Interface (I/O DI). The Network Accelerator's bandwidth usage within the XLS I/O DI is controlled by the XLS system I/O credits mechanism. Therefore, optimizing bandwidth usage of the XAUI interface within the XLS's internal data transport mechanism involves these areas. See [Chapter 8, "Memory and I/O Access"](#) for details.

17.4.1.3 Interrupt Setup

Interrupt handling for the XAUI interface is routed through the Network Accelerator to the XLS's PIC (Programmable Interrupt Controller). This involves setup in several areas:

1. Interrupt masking must be set up appropriately in the Network Accelerator for this interface.
2. The parameters in the Interrupt Redirection Table (IRT) in the PIC for this interrupt type (type 21 or 22) must be set up. See discussions in [Chapter 10, "Programmable Interrupt Controller"](#), of this manual, and the [Chapter 13, "Networking Accelerators"](#).
3. The EIRR for each nCPU that will interact with the Network Accelerator must be set up. See discussion of EIRR registers in CPUs.

The XAUI interface can generate a variety of interrupts. The interrupts from an interface are aggregated in its companion Network Accelerator to create a single interrupt that goes to the PIC. When an nCPU is given the interrupt, it must then check the appropriate IntReg register at 0x0A6 in the appropriate Network Accelerator to determine what interrupt condition actually occurred. The software must clear the interface's interrupt bit in the PIC, and must also clear the appropriate interrupt bit in the correct Network Accelerator. In some system designs, it may also be necessary for the software to interact with an interrupt source external to the XLS, in a PHY device.

17.4.2 Reset Handling

Although all internal logic of the XAUI interface are reset upon system reset, some XAUI internal modules can be reset independently by means of bits in the XAUI register set. This can be useful in debugging or other circumstances.

17.4.3 Statistics Monitoring

Statistics values for traffic between each XAUI port and its Network Accelerator can be monitored in the XAUI statistics registers. The values are read simply by reading the appropriate statistics monitoring register. The monitoring activity is controlled by a register in the Network Accelerator that is companion to the XAUI interface. Note that a read clears a statistics counter register if the 'AutoZ' feature is enabled for all statistics registers. This feature is controlled in the Network Accelerator and is enabled by setting the AutoZ bit field in its StatCtrl (0x0A3) register. For details refer to the Network Accelerator chapter register descriptions. To enable having an interrupt generated to the PIC through the Network Accelerator when a statistics counter overflows, enable it in the Network Accelerator and then set appropriate masking bits in the XAUI's statistics CAM1 and CAM2 registers. See also: [Section 17.6.4](#).

17.5 XAUI Registers

17.5.1 Introduction

This section discusses the XAUI interface registers. Each XAUI interface has its own set of configuration registers. These registers control operation of the interface. A companion set of registers exists in the companion Network Accelerator for each XAUI interface which affect flow control and to control how that interface interacts with the XLS internal architecture. For proper operation, both sets of registers must be configured.

Bits that are not implemented or are *Reserved*, are set to zero.

17.5.1.1 Definition of Terms

The following terms are used in the register descriptions to describe field Read/Write capability.

Table 17-1. Field Code Description

Type Code	Meaning	Description
R/O	Read Only	The field contains a static value and cannot be written.
R/W	Read/Write	The field can be modified by software. Note that some fields, where indicated, are modifiable only indirectly, by writing to a shadow register then performing a soft reset.
R/C	Read/Clear	The field can be read to give status information. Bits may be cleared by writing a '1' to them; writes of '0' are ignored.

17.5.1.2 Data Access and Endianness

The local space in the interface is little endian.

17.5.2 Register Access

XAUI registers are accessed using the combination of individual register offset, plus interface group base address, plus base address for all XLS_IO devices. Register offsets are listed with each register in this chapter.

Each XAUI interface has a group offset base address as described in Chapter 8. However, note that each XAUI interface shares a port with an SGMII interface, as determined by pin-strapping of the XLS. Only one interface in these shared pairs should be register-accessed; that is, if the XAUI interface for a port is selected, do not attempt access the SGMII registers of that pair or undefined results will occur.

Note that the offsets used to access registers for the Network Accelerator associated with each XAUI interface lie within the address region occupied by that SGMII/XAUI space.

For example, addressing for a register in the Network Accelerator handling XAUI-Port (when Port XA is set for XAUI by strapping pin IO_AD[7] High) is computed using the sum of the Network Accelerator register offset plus the XAUI-Interface_A group base offset of 0x1 1000 plus the XLS_IO_BAR value.

Similarly, the registers for XAUI-Interface_B (when Port XB is set for XAUI by strapping pin IO_AD[8] High) lie at base Offset 0x1 3000 in the XAUI space, and the interface's companion Network Accelerator registers are in that space. See chapter 8 for diagrams.

Unless otherwise noted, all registers have read and write access from the XLS internal side of the interface. All registers are word registers and are accessed on 4-byte boundaries.

- ADDR is the byte offset from the base address for the XAUI interface. The base address for each XAUI interface is described in Chapter 8 in the System Bridge Controller information.

- Register name is the name of the register; in some cases names are indexed.
- Reset is the reset value of the register.

Multiply register indexes by 4 to get the byte address offset from the sub-region base address shown in [Chapter 8, “Memory and I/O Access”](#).

Bits that are not implemented or are *Reserved*, are set to zero.

17.5.3 XAUI Interface Register Summary

The following table describes registers for XAUI_0 Interface and XAUI_1 Interface, which have identical but separate sets of control registers.

Multiply index values by 4 to get the byte address offset from the subregion base address shown in [Chapter 8, “Memory and I/O Access”](#).

Table 17-2. XAUI Register Map

Index	Register	Default/Reset
0x00	Configuration Register 0	0x8000
0x01	Configuration Register 1	0x3D
0x02	Configuration Register 2	0x40D0000
0x03	Configuration Register 3	0xFFFF
0x04	MAC_ADDR0_LO (same as register at 0x50)	0x0000_0000
0x05	MAC_ADDR0_HI (same as register at 0x51)	0x0000_0000
0x06 – 0x07	Reserved	
0x08	Maximum Frame Length	0x01800600
0x09 – 0x0A	Reserved	
0x0B	Revision Level	0x0608
0x0C – 0x0F	Reserved	
0x10	MII Management Command Register	0
0x11	MII Management Field Register	0x2086000
0x12	MII Management Configuration Register	0x3E
0x13	MII Link Fail Vector	0
0x14	MII Management Indicator Register	0
0x20 - 4F	XAU1 Statistics Registers	
0x50	MAC_ADDR0_LO (same as register at 0x04)	0x0000_0000
0x51	MAC_ADDR0_HI (same as register at 0x05)	0x0000_0000
0x52	MAC_ADDR1_LO	0x0000_0000
0x53	MAC_ADDR1_HI	0x0000_0000
0x54	MAC_ADDR2_LO	0x0000_0000
0x55	MAC_ADDR2_HI	0x0000_0000
0x56	MAC_ADDR3_LO	0x0000_0000
0x57	MAC_ADDR3_HI	0x0000_0000
0x58	MAC_ADDR_MASK0_LO	
0x59	MAC_ADDR_MASK0_HI	
0x5A	MAC_ADDR_MASK1_LO	
0x5B	MAC_ADDR_MASK1_HI	
0x5C	MAC_FILTER_CONFIG	
0x60 - 6F	HASH_TABLE_VECTOR	

17.5.4 Configuration Register 0

Register Index: 0x00 Bits [31:0]

31	30:23	22	21	20:19	18	17	16	15:0
MACRST	Reserved	RSTRCTL	RSTRFN	Reserved	RSTTCTL	RSTTFN	RSTMIIIM	Reserved

Bits	Name	Description	R/W	Reset
31	MACRST	1: Setting this bit resets the XAUI Interface	R/W	1
30:23	Reserved	Reserved	0	0
22	RSTRCTL	1: Setting this bit resets the Receive Control module. This module detects Control frames and contains the pause timers.	R/W	0
21	RSTRFN	1: Setting this bit resets the Receive Function module. This module performs the receive frame protocol.	R/W	0
20:19	Reserved	Reserved	0	0
18	RSTTCTL	1: Setting this bit resets the Transmit Control module. This module multiplexes data and Control frame transfers. It also responds to XOFF PAUSE Control frames.	R/W	0
17	RSTTFN	1: Setting this bit resets the Transmit Function module. This module performs the frame transmission protocol.	R/W	0
16	RSTMIIIM	1: Setting this bit resets the MII Management module. Clearing this bit allows the MII Management module to perform management read/write cycles as requested via the Host Interface.	R/W	0
15:0	Reserved	Reserved	0	0

17.5.5 Configuration Register 1

Register Index: 0x01 Bits [31:0]

31	30	29	28	27	26	25	24	23:13	12	11:10	9:8	7	6	5	4	3	2	1:0
TCTLEN	TFEN	RCTLLEN	RFEN	RSVD	TFEN	RSVD	RFEN	RSVD	RCTLHSRTP	DLYFCSTX	DLYFCSRX	PPEN	BYTSWP	DRPLT64	PRMSCRX	GENFCS	PADMODE1	PADMODE2

Bits	Name	Description	R/W	Reset
31	TCTLEN	After this bit is set by a write to the Configuration register, the Transmit Control module is not able to send PAUSE control frames until the first break in the transmission of frames by the XAUI interface. 0: Clearing this bit disables the PAUSE Control frames by the Transmit Control module. 1: Setting this bit enables the Transmit Control module to send PAUSE Control frames when requested by the system.	R/W	0
30	TFEN	After clearing this bit the XAUI interface waits until it is no longer transmitting to disable the Transmit module. 0: Clearing this bit disables the transmission of frames. After this bit is set by a write to the configuration register, the XAUI interface is able to begin transmission of frames on the next rising edge of the MTCLK. 1: Setting this bit enables the XAUI interface to transmit frames from the system.	R/W	0
29	RCTLLEN	After this bit is set via a write to the configuration register, the XAUI interface waits until the first break in the reception of frames before enabling the reception of PAUSE Control frames. This prevents partial PAUSE Control frames from being received after the Receive Control module is enabled. 0: Clearing this bit causes the Receive Control module to ignore PAUSE Control frames. 1: Setting this bit enables the Receive Control module to detect and act on PAUSE Control frames.	R/W	0
28	RFEN	After this bit is set by a write to the Configuration register, the XAUI interface waits until the first idle on the wire before enabling the reception of frames. This prevents partial frames from being received after the Receive Function is enabled. 0: Clearing this bit disables the reception of frames. 1: Setting this bit enables the XAUI interface to receive frames from the PHY.	R/W	0
27	Reserved	Reserved	0	0
26	TFEN	Read-Only bit that reflects the current enable state of the XAUI interface Transmit module. 0: inactive 1: active	R	0
25	Reserved	Reserved	0	0
24	RFEN	Read-Only bit that reflects the current enable state of the XAUI interface Receive module. 0: inactive 1: active	R	0
23:13	Reserved	Reserved	0	0

Bits	Name	Description	R/W	Reset
12	RCTLSHRTP	0: Clear this bit if shortening the XOFF pause time is not desired. This bit is normally used for testing purposes. 1: Setting this bit causes the reception of XOFF PAUSE Control frames to pause the Transmit module for only 64 slot times (4096 byte times), thereby shortening the XOFF control frame pause time. All other PAUSE Control frame timing parameters are not affected.	R/W	0
11:10	DLYFCSTX	The setting of these two bits determines the number of 4-byte words to delay the Transmit module's FCS calculation on the transmit frame as follows: 00: Does not delay the FCS calculation on the transmit frame. 01: Delays the FCS calculation by 1 4-byte word. 10: Delays the FCS calculation by 2 4-byte words. 11: Delays the FCS calculation by 3 4-byte words.	R/W	0
9:8	DLYFCSRX	These two bits determine the number of 4-byte words to delay the Receive module's FCS calculation on the receive frame as follows: 00: Does not delay the FCS calculation on the transmit frame. 01: Delays the FCS calculation by 1 4-byte word. 10: Delays the FCS calculation by 2 4-byte words. 11: Delays the FCS calculation by 3 4-byte words.	R/W	0
7	PPEN	0: Clear this bit if the use of per-packet overrides is not desired. 1: Setting this bit enables the use of the per-packet settings on transmit frames.	R/W	0
6	BYTSPWP	1: When this bit is set the system data bytes are swapped in the following manner: tfdat[31:24]= tfdat[7:0], tfdat[23:16]= tfdat[15:8], tfdat[15:8]= tfdat[23:16], tfdat[7:0]= tfdat[31:24]. rfdat[63:56]= rfdat[7:0], rfdat[55:48]= rfdat[15:8], rfdat[47:40]= rfdat[23:16], rfdat[39:32]= rfdat[31:24], rfdat[31:24]= rfdat[39:32], rfdat[23:16]= rfdat[47:40], rfdat[15:8]= rfdat[55:48], rfdat[7:0]= rfdat[63:56]. <i>This bit should be asserted for test only.</i>	R/W	0
5	DRPLT64	1: Setting this bit sets the SRDRPFRM output High, if a receive frame is less than 64 bytes in length.	R/W	1
4	PRMSCRX	1: Setting this bit enables the XAUI interface Receive module to accept all receive frames that are greater than 8 bytes in length with a valid start of frame delimiter on XGRMII[31:24]. When not asserted, the XAUI interface Receive module strictly expects the IEEE802.3ae specified control and data sequences to validate the start of frame delimiter on XGRMII[31:24]. This is in addition to the minimum 9-byte frame requirement.	R/W	0
3	GENFCS	1: Setting this bit causes the XAUI interface to check the frame's length field to ensure it matches the actual data field length and report the results to the TSV.	R/W	1

Bits	Name	Description	R/W	Reset
2	PADMODE1	0: Clear this bit if frames presented to the XAUI interface have a valid length and contain a valid FCS. This bit must be set if any PADMODE bits are set. 1: Setting this bit causes the XAUI interface to generate and append a FCS on all frames not using per-packet overrides.	R/W	1
1:0	PADMODE2	These two bits determine the pad mode for the XAUI interface Transmit Function as follows: 00: Does not pad the transmit frame. 01: Conditionally pads frames to 64 bytes. Appends an FCS on all frames. 10: Detects VLAN frames and conditionally pads them to 68 bytes. Detects non-VLAN frames and conditionally pads to 64 bytes. Appends an FCS to all frames. 11: Conditionally pads frames to 68 bytes. Appends an FCS to all frames	R/W	1

17.5.6 Configuration Register 2

Register Index: 0x02 Bits [31:0]

31 30:28 27 26 25:24 23:21 20:16 15 14:6 5 4:0											
TCTL FRCP	Reserved	MILNKFLTH	ALNKFLTH	RFLNKFLT	Reserved	IPGEXT MOD	RCTFRCP	Reserved	IPG EXTN	MIPG EXT	
Bits	Name	Description							R/W	Reset	
31	TCTLFRCP	Note: This bit is intended for test purposes only and should not be relied on for use in real-time operation. 1: Setting this bit forces the XAUI interface to transmit a single PAUSE Control frame. If the XAUI interface is currently transmitting a frame, it will wait until the end of that frame before transmitting the PAUSE Control frame. Note: This bit must remain set for a minimum of a MAX sized frame + an IPG + a minimum sized frame, in order to assure that an XOFF PAUSE Control frame is sent. If the bit is cleared to early, the exact operation of the XAUI interface in regards to the PAUSE Control frame cannot be guaranteed.							R/W	0	
30:28	Reserved	Reserved							0		
27	MILNKFLTH	MI KLink Fault Handler 1: Manual link fault handler which forces local fault sequences onto the XAUI transmit data block. May be changed during normal run time operation.							R/W	0	
26	ALNKFLTH	1: Steady state configuration bit which, when asserted, enables XAUI transmit data block to automatically, but abruptly, output local or remote fault sequences based on RFLNKFLT assertion by XAUI receive block.							R/W	1	
25:24	RFLNKFLT	Asserted when link fault detected as per 802.3ae clause 46.3.4. 0: no fault detected. 1: local fault detected. 2: remote fault detected.							R		
23:21	Reserved	Reserved							0		
20:16	IPGEXTMOD	These configuration bits represent the denominator in the Inter-Packet-Gap (IPG) extension algorithm required for optional SONET OC-192 data rate control. The IPG extension algorithm extends the minimum IPG byte count in bytes, by the floor resultant of the last transmit frame's total byte count divided by the value of these configuration bits. If a different denominator is desired, program this 4-bit field. Valid possible values are: 0x0D thorough 0x16. Do not program any values other than these.							R/W	0xD	
15	RCTFRCP	0: Clearing this bit re-enables the Transmit module. 1: Setting this bit forces the Transmit module to pause. If the XAUI interface is currently transmitting a frame, it will wait until the end of that frame before pausing.							R/W	0	
14:6	Reserved	Reserved							0		

Bits	Name	Description	R/W	Reset
5	IPGEXTEN	0: Clear this bit if extending the minimum IPG is not desired. 1: Setting this bit enables the IPG extension algorithm required for optional SONET OC-192 data rate control.	R/W	0
4:0	MIPGEXT	These configuration bits represent the additional minimum average IPG byte count above the default 12 bytes for transmitting back-to-back frames during normal LAN operation. For SONET OC-192 operation, this value is added to the extension.	R/W	0

17.5.7 Configuration Register 3

Register Index: 0x03 Bits [31:0]

31:16		15:0		
FLTRFRM		FLTRFRMDC		
Bits	Name	Description		
31:16	FLTRFRM	<p>These configuration bits are used to signal the drop frame conditions for the SRDRPFRM output. The bits correspond to the Receive Statistics Vector on a one per one basis. For example, bit 31 corresponds to RSV[31], and bit 30 corresponds to RSV[30]. When these bits compare and the don't-care is not asserted, SRDRPFRM is asserted. The setting of these bits, along with their don't-care values in the FLTRFRMDC configuration registers, create the filter that sets the SRDRPFRM output, if the receive frame does not pass the acceptable conditions and should be dropped by the System.</p> <p>For example, if it's desired to drop a frame that contains a FCS Error, Bit[20] would be set, and all receive frames that have RSV[20] set would set the SRDRPFRM output.</p>	R	0
15:0	FLTRFRMDC	<p>These configuration bits indicate which Receive Statistics Vectors are don't cares for the SRDRPFRM output. The bits correspond to the Receive Statistics Vector follows: Bit[15] = RSV[31] Bit[14] = RSV[30] Bit[0] = RSV[16]</p> <p>0: Clearing the bit checks for a matching condition on the corresponding configuration bit.</p> <p>1: Setting a configuration bit indicates a don't care for that RSV bit.</p>	R/W	0xFFFF

17.5.8 MAC_ADDR0_LO

MAC Station Address Least-Significant Word.

Default value: 0X0000_0000

Note: This register is also accessible at index 0x50.**Register Index: 0x04 Bits [31:0]**

31	24 23	16		
Station Address, 1st octet		Station Address, 2nd octet		
15	8 7	0		
Station Address, 3rd octet		Station Address, 4th octet		
Bits Bits Description R/W Reset				
31:24	MACADR[7:0]	First Octet of Station Address	R/W	0
23:16	MACADR[15:8]	Second Octet of Station Address	R/W	0
15:8	MACADR[23:16]	Third Octet of Station Address	R/W	0
7:0	MACADR[31:24]	Fourth Octet of Station Address	R/W	0

17.5.9 MAC_ADDR0_HI

Default value: 0X0000_0000

Register Index: 0x05 Bits [31:0]

31	24 23	16		
Station Address, 5th octet		Station Address, 6th octet		
15		0		
Reserved				
Bits Name Description R/W Reset				
31:24	MACADR0[39:32]	Fifth Octet of Station Address	R/W	0
23:16	MACADR0[47:40]	Sixth Octet of Station Address	R/W	0
15:0	Reserved	Reserved	0	

17.5.10 Maximum Frame Length

Register Index: 0x08 Bits [31:0]

31:30		29:16	15:0
Reserved	MXFRMWCTX	MXFRMBCRX	

Bits	Name	Description	R/W	Reset
31:30	Reserved	Reserved	0	
29:16	MXFRMWCTX	This field represents the word count (4 bytes) of the maximum allowable frame size in the transmit direction. All transmit frames larger than the maximum frame size are aborted at the maximum frame size by the XAUI interface and is indicated as such in the Transmit Statistics Vector. If a different maximum length restriction other than 0x0180 (384d) is desired, program this 14-bit field.	R/W	0x0180
15:0	MXFRMBCRX	This field represents the byte count of the maximum allowable frame size in the receive direction. All receive frames larger than the maximum frame size are truncated to the maximum frame size by the XAUI interface and is indicated as such in the Receive Statistics Vector. If a different maximum length restriction other than 0x0600 (1536d) is desired, program this 16-bit field.	R/W	0x0600

17.5.11 Revision Level

Register Index: 0x0B Bits [31:0]

31:16		15:0
Reserved		REVLVL

Bits	Name	Description	R/W	Reset
31:16	Reserved	Reserved		0
15:0	REVLVL	This field represents the revision level of the XAUI module.	R	0x0608

17.5.12 MII Management Command Register

Register Index: 0x10 Bits [31:0]

31:4			3:2:0	
Reserved			LDCMD	MIIMCMD
Bits	Bits	Description	R/W	Reset
31:4	Reserved	Reserved		0
3	LDCMD	Signal which indicates to the MIIM block that an MIIM sequence command has been loaded and to start the operation specified by the Network Accelerator.	R/W	0
2:0	MIIMCMD	<p>Data bus that specifies which operation is performed by the MIIM block. The six different values are as follows:</p> <ul style="list-style-type: none"> 000: Idle, no operation. 001: Legacy (10/100/1000 Mbps PH) - Write 010: Legacy (10/100/1000 MBS PHY) - Read 011: Single Why Monitor operation The MIIM continually reads from a single PHY register specified by the contents of the PHYADX and REGADX fields. 100: Multiple PHY Monitor operation. The MIIM continually reads from a set of PHYs of contiguous address space. The starting address of the PHY is specified by the content of the PHYADX field, the very next PHY to be read will be PHYADX + 1. The last PHY to be queried in this read sequence is that which resides at address 0x31, after which the read sequence continues again for the PHY specified by the PHYADX field. 101: 10-Gigabit MMD operation. The MIIM presents the contents of the MIIM Field Register to the MDIO pins in the order specified by the Management Frame Format in IEEE Standard 802.3 clause 45. 110: Clear Link Fail. Causes the Link Fail bit in the MIIM Indicators Register to be cleared. This value also clears each bit in the MIIM Link Fail Vector Register 	R/W	0

17.5.13 MII Management Field Register

Register Index: 0x11 Bits [31:0]

31:30		29:28	27:23	22:18	17:16	15:0	
STFIELD		OPFIELD	PHYADX	REGADX	TAFIELD	MIIMWRDAT	
Bits	Name	Description				R/W	Reset
31:30	STFIELD	<p>Signal which represents the 2-bit ST field of the Management frame format specified by IEEE standard 802.3 Clause 22, and augmented by Clause 45. This field is used by PHY monitor and multiple PHY monitor operations either for either legacy PHY access or 10-Gig PHY access and must be programmed accordingly.</p> <p>00: 10-Gigabit PHY operation 01: For legacy PHY operations (10/100/1000 Mb) this field is ignored by the MIIM block and is internally hard set to 1.</p>				R/W	1
29:28	OPFIELD	<p>Data bus which represents the 2-bit OP field of the Management frame format specified by IEEE standard 802.3 Clause 22 and augmented by Clause 45. For 10-Gigabit operation this field is defined for the following values:</p> <p>00: Address Operation for indirect access into 10-Gigabit MMD device. 01: Write Access into 10-Gigabit MMD. 11: Read Access from 10-Gigabit MMD. 10: Post Read Increment Address access into 10-Gigabit MMD.</p> <p>For any legacy PHY operations (10/100/1000 Mbps) this field is ignored by the MIIM block and is internally hard set to a value of 1 for write operations and to 2 for read operations. This field is used by PHY monitor and multiple PHY monitor operations either for either legacy PHY access or 10-Gig PHY access and must be programmed accordingly.</p>				R/W	2
27:23	PHYADX	<p>Data bus which represents the 5-bit PHY address field of the Management frame format specified by IEEE 802.3 Clause 22 or the equivalent PRTAD field defined in Clause 45. For legacy (10/100/1000 Mbps) PHY access, up to 31 PHYs can be addressed (0 is reserved). For 10-Gbps MMD access all 32 ports are allowed to be accessed through this programmable field.</p>				R/W	1
22:18	REGADX	<p>Data bus which represents the 5-bit Register Address field of Management frame format specified by IEEE standard 802.3 Clause 22 or the equivalent DEVAD field defined in Clause 45. Up to 32 registers can be addressed. For 10 Gigabit operation this signal is used to define the MMD specific address</p>				R/W	1
17:16	TAFIELD	<p>Data bus which represents the 2-bit TA field of the Management frame format specified by IEEE standard 802.3 Clause 22 and by Clause 45. For 10-Gigabit operation this signal must be programmed to a value of 2. For any legacy PHY operations (10/100/1000 Mbps) this field is ignored by the MIIM block and is internally hard set to a value of 2.</p>				R/W	2
15:0	MIIMWRDAT	<p>Data bus which represents the 16-bit Date field of the Management frame format specified by IEEE 802.3 Clause 22 and augmented by Clause 45. This is the data that is written into a PHY device on write operation. For 10-Gbps operation, this signal is the 16-bit address field for an indirect read operation or the data field for direct writes.</p>				R/W	0x0000

17.5.14 MII Management Configuration Register

Register Index: 0x12 Bits [31:0]

31:8		7	6:0		
Reserved		NOPRAM	CLKDIV		
Bits	Name	Description		R/W	Reset
31:8	Reserved	Reserved			0
7	NOPRAM	Bypass preamble. 0: do not bypass. This bit should be set to 0 unless it is known that the PHY to be accessed can support a preamble suppressed format. 1: Setting this bit to 1 causes the MIIM to bypass the preamble portion of the Management frame format.		R/W	0
6:0	CLKDIV	These 7 bits specify the half-clock duration of the MDC output in number of MAC Transmit clock ticks. The MIIM uses the MTCLK for its sequential logic and also to generate the MDC frequency. The calculation for MDC frequency in MHz is: $F(MDC) = F(MTCLK) / 2 * (CLKDIV + 1)$ The value of CLKDIV should never be set to zero.		R/W	0x3E

17.5.15 MIIM Link Fail Vector

Register Index: 0x13 Bits [31:0]

31:0				
MIMMLFVEC				
Bits	Name	Description	R/W	Reset
31:0	MIMMLFVEC	Data bus that reports a link fail condition in the PHY device or MMD whose port address is that which matches the bus' index number. This bus is active during a multiple PHY monitor operation and remains valid after the host system terminates the multiple PHY monitor command. The data bus is cleared by any new MIIM command after the monitor command has been terminated	R	0

17.5.16 MII Management Indicator Register**Register Index: 0x14 Bits [31:0]**

31:5	4	3	2	1	0
Reserved	MIIM_PHYLF	MIIM_MONCPLT	MIIM_MONVLD	MIIM_MON	MIIM_BUSY

Bits	Name	Description	R/W	Reset
31:5	Reserved	Reserved	0	
4	MIIM_PHYLF	1: This bit is an inverted and latched copy of Link Status bit read from a legacy PHY or an MMD while in PHY monitor or Multi-PHY monitor mode. If bit position #2 of a legacy PHY or MMD is read as zero then this bit is set to 1. This bit remains valid after the host system terminates the PHY monitor command. 0: This bit is cleared when any MIIM command is executed after the termination of the PHY monitor command. Note the REGADX[4:0] can be a variable on a given monitor operation and this field needs to be programmed appropriately for valid link fail reports.	R	0
3	MIIM_MONCPLT	1: This bit will be set when the AMIIM is doing a multiple PHY monitor operation and all the PHY devices in the operational set have been queried for status information	R	0
2	MIIM_MONVLD	1: This bit will be set on a single PHY monitor operation after one read operation into the PHY's status register has resulted in acquiring the Link Status bit value	R	0
1	MIIM_MON	0: This bit is cleared when the host terminates either command. 1: This bit is always active from the beginning of and during the complete operation of a single PHY monitor command or a multiple PHY monitor command.	R	0
0	MIIM_BUSY	1: This 'busy' bit signals the operational time for the MIIM to execute a command. For any single sequence operation this bit is set to 1 at the time the command is written into the MIIM command register and remains 1 until the conclusion of the operation. For a single PHY monitor command, this bit is set to 1 at the beginning of the operation and remains 1 until either a Link Fail condition is reported, or until the host terminates that command. In either case, MIIM_BUSY remains 1 until the end of the read operation that was terminated, or until the end of the read operation that resulted in Link Fail. For multiple PHY monitor operation, this bit remains 1 until the host terminates that operation.	R	0

17.5.17 MAC_ADDR0_LO

MAC Station Address Least-Significant Word.

This register is also accessible at index 0x50

Default value: 0X0000_0000

Register Index: 0x50 Bits [31:0]

MAC_ADDR0_LO/HI and MAC_ADDR1_LO/HI are used for full or exact match.

31	24 23	16		
Station Address, 1st octet		Station Address, 2nd octet		
15	8 7	0		
Station Address, 3rd octet		Station Address, 4th octet		
Bits	Name	Description	R/W	Reset
31:24	MACADR0[7:0]	First Octet of Station Address	R/W	0
23:16	MACADR0[15:8]	Second Octet of Station Address	R/W	0
15:8	MACADR0[23:16]	Third Octet of Station Address	R/W	0
7:0	MACADR0[31:24]	Fourth Octet of Station Address	R/W	0

17.5.18 MAC_ADDR0_HI

Register Index: 0x51 Bits [31:0]

31	24 23	16
Station Address, 5th octet		Station Address, 6th octet
15	0	0
<i>Reserved</i>		

Default value: 0X0000_0000

Bits	Name	Description	R/W	Reset
31:24	MACADR0[39:32]	Fifth Octet of Station Address	R/W	0
23:16	MACADR0[47:40]	Sixth Octet of Station Address	R/W	0
15:0	<i>Reserved</i>	<i>Reserved</i>	0	

17.5.19 MAC_ADDR1_LO

Default value: 0X0000_0000

Register Index: 0x52 Bits [31:0]

MAC_ADDR0_LO/HI and MAC_ADDR1_LO/HI are used for full or exact match.

31	24 23	16		
Station Address, 1st octet	Station Address, 2nd octet			
15	8 7	0		
Station Address, 3rd octet		Station Address, 4th octet		
Bits	Name	Description	R/W	Reset
31:24	MACADR1[7:0]	First Octet of Station Address	R/W	0
23:16	MACADR1[15:8]	Second Octet of Station Address	R/W	0
15:8	MACADR1[23:16]	Third Octet of Station Address	R/W	0
7:0	MACADR1[31:24]	Fourth Octet of Station Address	R/W	0

17.5.20 MAC_ADDR1_HI

Default value: 0X0000_0000

Register Index: 0x53 Bits [31:0]

31	24 23	16		
Station Address, 5th octet	Station Address, 6th octet			
15	8 7	0		
Reserved				
Bits	Name	Description	R/W	Reset
31:24	MACADR1[39:32]	Fifth Octet of Station Address	R/W	0
23:16	MACADR1[47:40]	Sixth Octet of Station Address	R/W	0
15:0	Reserved	Reserved	0	

17.5.21 MAC_ADDR2_LO

Default value: 0X0000_0000

Register Index: 0x54 Bits [31:0]

31	24 23	16
Station Address, 1st octet		Station Address, 2nd octet

15	8 7	0
Station Address, 3rd octet		Station Address, 4th octet

Bits	Name	Description	R/W	Reset
31:24	MACADR2[7:0]	First Octet of Station Address	R/W	0
23:16	MACADR2[15:8]	Second Octet of Station Address	R/W	0
15:8	MACADR2[23:16]	Third Octet of Station Address	R/W	0
7:0	MACADR2[31:24]	Fourth Octet of Station Address	R/W	0

17.5.22 MAC_ADDR2_HI

Default value: 0X0000_0000

Register Index: 0x55 Bits [31:0]

31	24 23	16
Station Address, 5th octet		Station Address, 6th octet

15	0	0
<i>Reserved</i>		

Bits	Name	Description	R/W	Reset
31:24	MACADR2[39:32]	Fifth Octet of Station Address	R/W	0
23:16	MACADR2[47:40]	Sixth Octet of Station Address	R/W	0
15:0	Reserved	<i>Reserved</i>	0	

17.5.23 MAC_ADDR3_LO

Default value: 0X0000_0000

Register Index: 0x56 Bits [31:0]

31	24 23	16			
	Station Address, 1st octet	Station Address, 2nd octet			
15	8 7	0			
	Station Address, 3rd octet	Station Address, 4th octet			
Bits		Name	Description	R/W	Reset
31:24	MACADR3[7:0]	First Octet of Station Address		R/W	0
23:16	MACADR3[15:8]	Second Octet of Station Address		R/W	0
15:8	MACADR3[23:16]	Third Octet of Station Address		R/W	0
7:0	MACADR3[31:24]	Fourth Octet of Station Address		R/W	0

17.5.24 MAC_ADDR3_HI

Default value: 0X0000_0000

Register Index: 0x57 Bits [31:0]

31	24 23	16			
	Station Address, 5th octet	Station Address, 6th octet			
15	0				
	Reserved				
Bits		Name	Description	R/W	Reset
31:24	MACADR3[39:32]	Fifth Octet of Station Address		R/W	0
23:16	MACADR3[47:40]	Sixth Octet of Station Address		R/W	0
15:0	Reserved	Reserved		0	

17.5.25 MAC_ADDR_MASK0_LO**Register Index: 0x58 Bits [31:0]**

This register contains the 32-bit low mask value to be ANDed with MAC_ADDR2_LO, which is then compared with DEST_ADDR for a match. The same activity is performed with the high word. If a match is detected, and ADDR_MATCH_DISC is enabled, then the packet is declared valid.

17.5.26 MAC_ADDR_MASK0_HI**Register Index: 0x59 Bits [31:0]**

This register contains the 32-bit high mask value to be ANDed with MAC_ADDR2_HI, which is then compared with DEST_ADDR for a match. The same activity is performed with the low

word. If a match is detected, and ADDR_MATCH_DISC is enabled, then the packet is declared valid.

17.5.27 MAC_ADDR_MASK1_LO

Register Index: 0x5A Bits [31:0]

This register contains the 32-bit high mask value to be ANDed with MAC_ADDR3_LO, which is then compared with DEST_ADDR for a match. The same activity is performed with the high word. If a match is detected, and ADDR_MATCH_DISC is enabled, then the packet is declared valid.

17.5.28 MAC_ADDR_MASK1_HI

Register Index: 0x5B Bits [31:0]

This register contains the 32-bit high mask value to be ANDed with MAC_ADDR3_HI, which is then compared with DEST_ADDR for a match. The same activity is performed with the low word. If a match is detected, and ADDR_MATCH_DISC is enabled, then the packet is declared valid.

17.5.29 MAC_FILTER_CONFIG

Register Index: 0x5C Bits [31:0]

The bit fields in this register are used to configure the packet address filtering mechanism.

Bits		Description	R/W	Reset
#	Name			
31:11	Reserved	Reserved		
10	BROADCAST_EN	Accept all broadcast packets	R/W	0
9	PAUSE_FRAME_EN	Accept all pause frames	R/W	0
8	ALL_MCAST_EN	Accept All MCAST packets	R/W	0
7	ALL_UCAST_EN	Accept all UCAST packets	R/W	0
6	HASH_MCAST_EN	Accept if packet is MCAST and hash matches	R/W	0
5	HASH_UCAST_EN	Accept if packet is UCAST and hash matches	R/W	0
4	ADDR_MATCH_DISC	When set, discard if address matches any of the 4 entries, or else accept if address matches	R/W	0
3	MAC_ADDR3_VALID	MAC_ADDR3 entry is valid	R/W	1
2	MAC_ADDR2_VALID	MAC_ADDR2 entry is valid	R/W	1
1	MAC_ADDR1_VALID	MAC_ADDR1 entry is valid	R/W	1
0	MAC_ADDR0_VALID	MAC_ADDR0 entry is valid	R/W	1

17.5.30 HASH_TABLE_VECTOR

Register Index: 0x60-6F

0x6F-6E	0x6D-6C	0x6B-6A	0x69-68	0x67-66	0x65-64	0x63-62	0x61-60
511-448	447-384	383-320	319-256	255-192	191-128	127-64	63-0

MAC_ADR

17.6 XAUI Statistics Registers Summary

Each XAUI interface contains its own unique set of the following registers. That is, Port XA contains one set of these registers for use when the Port is in XAUI mode. Port XB likewise has one set for its XAUI interface. Each set's registers have the same names as the other set, and are therefore uniquely identified using their name together with their group base address and offset.

The group base address for each set of statistics registers is the same as the group base address for that XAUI port.

Table 17-3. XAUI Statistics Registers

Offset	Mnemonic	Register	Type	Default
Transmit and Receive Counters				
0x20	TR64	TR64 – Transmit & Receive 64 Byte Frame Counter	R/W	
0x21	TR127	Transmit and Receive 65 to 127 Byte Frame Counter	R/W	
0x22	TR255	Transmit and Receive 128 to 255 Byte Frame Counter	R/W	
0x23	TR511	Transmit and Receive 256 to 511 Byte Frame Counter	R/W	
0x24	TR1K	Transmit and Receive 512 to 1023 Byte Frame Counter	R/W	
0x25	TRMAX	Transmit and Receive 1024 to 1518 Byte Frame Counter	R/W	
0x26	TRMGV	Transmit and Receive 1519 to 1522 Byte Good VLAN Frame Count	R/W	
Receive Counters				
0x27	RBYT	Receive Byte Counter	R/W	
0x28	RPKT	Receive Packet Counter	R/W	
0x29	RFCS	Receive FCS Error Counter	R/W	
0x2A	RMCA	Receive Multicast Packet Counter	R/W	
0x2B	RBCA	Receive Broadcast Packet Counter	R/W	
0x2C	RXCF	Receive Control Frame Packet Counter	R/W	
0x2D	RXPF	Receive PAUSE Frame Packet Counter	R/W	
0x2E	RXUO	Receive Unknown OP code Counter	R/W	
0x2F	RALN	Receive Alignment Error Counter	R/W	
0x30	RFLR	Receive Frame Length Error Counter	R/W	
0x31	RCDE	Receive Code Error Counter	R/W	
0x32	RCSE	Receive Carrier Sense Error Counter	R/W	
0x33	RUND	Receive Undersize Packet Counter	R/W	
0x34	ROVR	Receive Oversize Packet Counter	R/W	
0x35	RFRG	Receive Fragments Counter	R/W	
0x36	RJBR	Receive Jabber Counter	R/W	
0x37	Reserved	Reserved	R/W	
Transmit Counters				
0x38	TBYT	Transmit Byte Counter	R/W	
0x39	TPKT	Transmit Packet Counter	R/W	
0x3A	TMCA	Transmit Multicast Packet Counter	R/W	
0x3B	TBCA	Transmit Broadcast Packet Counter	R/W	
0x3C	TXPF	Transmit PAUSE Control Frame Counter	R/W	
0x3D	TDFFR	Transmit Deferral Packet Counter	R/W	
0x3E	TEDF	Transmit Excessive Deferral Packet Counter	R/W	

Table 17-3. XAUI Statistics Registers (continued)

Offset	Mnemonic	Register	Type	Default
0x3F	TSCL	Transmit Single Collision Packet Counter	R/W	
0x40	TMCL	Transmit Multiple Collision Packet Counter	R/W	
0x41	TLCL	Transmit Late Collision Packet Counter	R/W	
0x42	TXCL	Transmit Excessive Collision Packet Counter	R/W	
0x43	TNCL	Transmit Total Collision Counter	R/W	
0x44	Reserved	Reserved	R/W	
0x45	Reserved	Reserved	R/W	
0x46	TJBR	Transmit Jabber Frame Counter	R/W	
0x47	TFCS	Transmit FCS Error Counter	R/W	
0x48	TXCF	Transmit Control Frame Counter	R/W	
0x49	TOVR	Transmit Oversize Frame Counter	R/W	
0x4A	TUND	Transmit Undersize Frame Counter	R/W	
0x4B	TFRG	Transmit Fragments Frame Counter	R/W	
Carry and Carry Mask Registers				
0x4C	CAR1	Carry Register One Register*	RO	
0x4D	CAR2	Carry Register Two Register*	RO	
0x4E	CAM1	Carry Register One Mask Register	R/W	
0x4F	CAM2	Carry Register Two Mask Register	R/W	

* Counter values will be discretely cleared on read when AUTOZ is asserted in the Network Accelerator StatCtrl register (0x0A3).

17.6.1 Transmit and Receive Counters

17.6.1.1 TR64 – Transmit & Receive 64 Byte Frame Counter

Register Index: 0x20 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TR64(17:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR64 (15:0)															
Bits	Name	Description			R/W	Reset									
31:18	Reserved	<i>Reserved</i>													
17:0	TR64	Transmit and Receive 64 Byte Frame Counter. Incremented for each good or bad frame transmitted and received which is 64 bytes in length inclusive (excluding framing bits but including FCS bytes).			R/W										

17.6.1.2 TR127 – Transmit & Receive 65 to 127 Byte Frame Counter

Register Index: 0x21 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TR127(17:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR127(15:0)															
Bits	Name	Description			R/W	Reset									
31:18	Reserved	<i>Reserved</i>													
17:0	TR127	Transmit and Receive 65 to 127 Byte Frame Counter: Incremented for each good or bad frame transmitted and received which is 65 to 127 bytes in length inclusive (excluding framing bits but including FCS bytes).			R/W										

17.6.1.3 TR255 – Transmit & Receive 128 to 255 Byte Frame Counter**Register Index:** 0x22 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TR255(17:16)	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR255(15:0)															

Bits	Name	Description	R/W	Reset
31:18	Reserved	Reserved		
17:0	TR255	Transmit and Receive 128 to 255 Byte Frame Counter. Incremented for each good or bad frame transmitted and received which is 128 to 255 bytes in length inclusive (excluding framing bits but including FCS bytes).	R/W	

17.6.1.4 TR511 – Transmit & Receive 256 to 511 Byte Frame Counter**Register Index:** 0x23 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TR511(17:16)	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR511(15:0)															

Bits	Name	Description	R/W	Reset
31:18	Reserved	Reserved		
17:0	TR511	Transmit and Receive 256 to 511 Byte Frame Counter. Incremented for each good or bad frame transmitted and received which is 256 to 511 bytes in length inclusive (excluding framing bits but including FCS bytes).	R/W	

17.6.1.5 TR1K – Transmit & Receive 512 to 1023 Byte Frame Counter

Register Index: 0x24 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														TR1K(17:16)	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR1K(15:0)															

Bits	Name	Description	R/W	Reset
31:18	Reserved	Reserved		
17:0	TR1K	Transmit and Receive 512 to 1023 Byte Frame Counter. Incremented for each good or bad frame transmitted and received which is 512 to 1023 bytes in length inclusive (excluding framing bits but including FCS bytes).	R/W	

17.6.1.6 TRMAX – Transmit & Receive 1024 to 1518 Byte Frame Counter

Register Index: 0x25 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														TRMAX(17:16)	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRMAX(15:0)															

Bits	Name	Description	R/W	Reset
31:18	Reserved	Reserved		
17:0	TRMAX	Transmit and Receive 1024 to 1518 Byte Frame Counter. Incremented for each good or bad frame transmitted and received which is 1024 to 1518 bytes in length inclusive (excluding framing bits but including FCS bytes).	R/W	

17.6.1.7 TRMGV – Transmit & Receive 1519 to 1522 Byte VLAN Frame Counter**Register Index: 0x26 Bits [31:0]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TRMGV(17:16)	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRMGV(15:0)															

Bits	Name	Description	R/W	Reset
31:18	Reserved	Reserved		
17:0	TRMGV	Transmit and Receive 1024 to 1518 Byte Frame Counter. Incremented for each good or bad frame transmitted and received which is 1024 to 1518 bytes in length inclusive (excluding framing bits but including FCS bytes).	R/W	

17.6.2 Receive Counters**17.6.2.1 RBYT - Receive Byte Counter****Register Index: 0x27 Bits [31:0]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														RBYT(23:16)	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBYT(15:0)															

Bits	Name	Description	R/W	Reset
31:23	Reserved	Reserved		
22:0	RBYT	Receive Byte Counter. The Statistic Counter register is incremented by the byte count of frames received with 0 to 1518 bytes, including those in bad packets, excluding framing bits but including FCS bytes.	R/W	

17.6.2.2 RPKT - Receive Packet Counter**Register Index:** 0x28 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														RPKT(17:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPKT (15:0)															
Bits	Name	Description												R/W	Reset
31:18	Reserved	<i>Reserved</i>													
17:0	RPKT	Receive Packet Counter. Incremented for each frame received packet (including bad packets, all Unicast, Broadcast, and Multicast packets).												R/W	

17.6.2.3 RFCS - Receive FCS Error Counter**Register Index:** 0x29 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>				RFCS(11:0)											
Bits	Name	Description												R/W	Reset
31:12	Reserved	<i>Reserved</i>													
11:0	RFCS	Receive FCS Error Counter. Incremented for each frame received that has a integral 64 to 1518 length and contains a Frame Check Sequence error.												R/W	

17.6.2.4 RMCA - Receive Multicast Packet Counter**Register Index: 0x2A Bits [31:0]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														RMCA(17:16)	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RMCA(15:0)															

Bits	Name	Description	R/W	Reset
31:18	Reserved	Reserved		
17:0	RMCA	Receive Multicast Packet Counter. Incremented for each Multicast good frame of lengths 64 to 1518 (non VLAN) or 1522 (VLAN) excluding Broadcast frames. This does not look at range/length errors.	R/W	

17.6.2.5 RBCA - Receive Broadcast Packet Counter**Register Index: 0x2B Bits [31:0]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														RBCA(21:16)	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RBCA(15:0)															

Bits	Name	Description	R/W	Reset
31:22	Reserved	Reserved		
21:0	RBCA	Receive Broadcast Packet Counter. Incremented for each Broadcast good frame of lengths 64 to 1518 (non VLAN) or 1522 (VLAN) excluding Multicast frames. This does not look at range/length errors.	R/W	

17.6.2.6 RXCF - Receive Control Frame Packet Counter]

Register Index: 0x2C Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														RXCF(17:16)	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RXCF(15:0)															

Bits	Name	Description	R/W	Reset
31:18	<i>Reserved</i>	<i>Reserved</i>		
17:0	RXCF	Receive Control Frame Packet Counter. Incremented for each MAC Control frame received (PAUSE & Unsupported).	R/W	

17.6.2.7 RXPF - Receive PAUSE Frame Packet Counter

Register Index: 0x2D Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>				RXPF(11:0)											

Bits	Name	Description	R/W	Reset
31:12	<i>Reserved</i>	<i>Reserved</i>		
11:0	RXPF	Receive PAUSE Frame Packet Counter. Incremented each time a valid PAUSE MAC Control frame is received.	R/W	

17.6.2.8 RXUO - Receive Unknown OPCode Packet Counter**Register Index: 0x2E Bits [31:0]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>															
Bits	Name	Description										R/W	Reset		
31:12	Reserved	Reserved													
11:0	RXUO	Receive Unknown OPcode Counter. Incremented each time a MAC Control Frame is received which contains an opcode other than a PAUSE.										R/W			

17.6.2.9 RALN - Receive Alignment Error Counter**Register Index: 0x2F Bits [31:0]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>															
Bits	Name	Description										R/W	Reset		
31:12	Reserved	Reserved													
11:0	RALN[11:0]	Receive Alignment Error Counter. Incremented for each received frame from 64 to 1518 (non VLAN) or 1522 (VLAN) which contains an invalid FCS and is not an integral number of bytes.										R/W			

17.6.2.10 RFLR - Receive Frame Length Error Counter]

Register Index: 0x30 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RFLR(15:0)															

Bits	Name	Description	R/W	Reset
31:16	Reserved	Reserved		
15:0	RFLR (15:0)	Receive Frame Length Error Counter. Incremented for each frame received in which the 802.3 length field did not match the number of data bytes actually received (46 - 1500 bytes). The counter is not incremented if the length field is not a valid 802.3 length, such as an EtherType value.	R/W	

17.6.2.11 RCDE - Receive Code Error Counter

Register Index: 0x31 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved		RCDE(11:0)													

Bits	Name	Description	R/W	Reset
31:12	Reserved	Reserved		
11:0	RCDE (11:0)	Receive Code Error Counter. Incremented each time a valid carrier was present and at least one invalid data symbol was detected.	R/W	

17.6.2.12 RCSE - Receive Carrier Sense Error Counter

Register Index: 0x32 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>		RCSE[11:0]													

Bits	Name	Description	R/W	Reset
31:12	Reserved	Reserved		
11:0	RCSE[11:0]	Receive False Carrier Counter. Incremented each time a false carrier is detected during idle, as defined by a 1 on RX_ER and an '0xE' on RXD. The event is reported along with the statistics generated on the next received frame. Only one false carrier condition can be detected and logged between frames.	R/W	

17.6.2.13 RUND- Receive Undersize Packet Counter

Register Index: 0x33 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>		RUND(11:0)													

Bits	Name	Description	R/W	Reset
31:12	Reserved	Reserved		
11:0	RUND (11:0)	Receive Undersize Packet Counter. Incremented each time a frame is received which is less than 64 bytes in length and contains a valid FCS and were otherwise well formed. This does not look at Range Length errors.	R/W	

17.6.2.14 ROVR - Receive Oversize Packet Counter

Register Index: 0x34 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
<i>Reserved</i>																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
<i>Reserved</i>				ROVR(11:0)												
Bits	Name		Description												R/W	Reset
31:12	<i>Reserved</i>		<i>Reserved</i>													
11:0	ROVR (11:0)		Receive Oversize Packet Counter. Incremented each time a frame is received which exceeded 1518 (non VLAN) or 1522 (VLAN) and contains a valid FCS and were otherwise well formed. This does not look at Range Length errors.												R/W	

17.6.2.15 RFRG - Receive Fragments Counter

Register Index: 0x35 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
<i>Reserved</i>																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
<i>Reserved</i>				RFRG(11:0)												
Bits	Name		Description												R/W	Reset
31:12	<i>Reserved</i>		<i>Reserved</i>													
11:0	RFRG (11:0)		Receive Fragments Counter. Incremented for each frame received which is less than 64 bytes in length and contains an invalid FCS, includes integral and non-integral lengths.												R/W	

17.6.2.16 RJBR - Receive Jabber Counter

Register Index: 0x36 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>				RJBR(11:0)											

Bits	Name	Description	R/W	Reset
31:12	<i>Reserved</i>	<i>Reserved</i>		
11:0	RJBR (11:0)	Receive Jabber Counter. Incremented for frames received which exceed 1518 (non VLAN) or 1522 (VLAN) bytes and contains an invalid FCS, includes alignment errors.	R/W	

17.6.3 Transmit Counters**17.6.3.1 TBYT - Transmit Byte Counter**

Register Index: 0x38 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>								TBYT(23:16)							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBYT(15:0)															

Bits	Name	Description	R/W	Reset
31:24	<i>Reserved</i>	<i>Reserved</i>		
23:0	TBYT	Transmit Byte Counter. Incremented by the number of bytes that were put on the wire including fragments of frames that were involved with collisions. This count does not include preamble/SFD or jam bytes.	R/W	

17.6.3.2 TPKT - Transmit Packet Counter

Register Index: 0x39 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TPKT(17:16)	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TPKT(15:0)															

Bits	Name	Description	R/W	Reset
31:18	Reserved	Reserved		
17:0	TPYT	Transmit Packet Counter. Incremented for each transmitted packet (including bad packets, excessive deferred packets, excessive collision packets, late collision packets, all Unicast, Broadcast, and Multicast packets).	R/W	

17.6.3.3 TMCA - Transmit Multicast Packet Counter

Register Index: 0x3A Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>														TMCA(17:16)	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TMCA(15:0)															

Bits	Name	Description	R/W	Reset
31:18	Reserved	Reserved		
17:0	TMCA	Transmit Multicast Packet Counter. Incremented for each Multicast valid frame transmitted (excluding Broadcast frames).	R/W	

17.6.3.4 TBCA - Transmit Broadcast Packet Counter

Register Index: 0x3B Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved														TBCA(17:16)	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TBCA(15:0)															
Bits	Name	Description				R/W	Reset								
31:18	Reserved	Reserved													
17:0	TBCA	Transmit Broadcast Packet Counter. Incremented for each Broadcast frame transmitted (excluding Multicast frames).				R/W									

17.6.3.5 TXPF - Transmit PAUSE Control Frame Counter

Register Index: 0x3C Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved				TXPF(11:0)											
Bits	Name	Description				R/W	Reset								
31:12	Reserved	Reserved													
11:0	TXPF	Transmit PAUSE Frame Packet Counter. Incremented each time a valid PAUSE MAC Control frame is transmitted.				R/W									

17.6.3.6 TDFR - Transmit Deferral Packet Counter

Register Index: 0x3D Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
<i>Reserved</i>																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
<i>Reserved</i>				TDFR[11:0]												
Bits	Name		Description												R/W	Reset
31:12	<i>Reserved</i>		<i>Reserved</i>													
11:0	TDFR		Transmit Deferral Packet Counter. Incremented for each frame, which was deferred on its first transmission attempt. Does not include frames involved in collisions.												R/W	

17.6.3.7 TEDF - Transmit Excessive Deferral Packet Counter]

Register Index: 0x3E Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
<i>Reserved</i>																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
<i>Reserved</i>				TEDF[11:0]												
Bits	Name		Description												R/W	Reset
31:12	<i>Reserved</i>		<i>Reserved</i>													
11:0	TEDF		Transmit Excessive Deferral Packet Counter. Incremented for frames aborted which were deferred for an excessive period of time (3036 byte times).												R/W	

17.6.3.8 TSCL - Transmit Single Collision Packet Counter

Register Index: 0x3F Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>				TSCL[11:0]											
Bits	Name	Description				R/W	Reset								
31:12	Reserved	<i>Reserved</i>													
11:0	TSCL	Transmit Single Collision Packet Counter. Incremented for each frame transmitted which experienced exactly one collision during transmission.				R/W									

17.6.3.9 TMCL - Transmit Multiple Collision Packet Counter

Register Index: 0x40 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>				TMCL[11:0]											
Bits	Name	Description				R/W	Reset								
31:12	Reserved	<i>Reserved</i>													
11:0	TMCL	Transmit Multiple Collision Packet Counter. Incremented for each frame transmitted which experienced 2-15 collisions (including any late collisions) during transmission as defined using the RETRY[3:0] field of the TX function control register.				R/W									

17.6.3.10 TLCL - Transmit Late Collision Packet Counter

Register Index: 0x41 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>															
TLCL[11:0]															
Bits	Name			Description								R/W	Reset		
31:12	<i>Reserved</i>			<i>Reserved</i>											
11:0	TLCL			Transmit Late Collision Packet Counter. Incremented for each frame transmitted which experienced a late collision during a transmission attempt. Late collisions are defined using the LCOL[5:0] field of the TX Function control register.								R/W			

17.6.3.11 TXCL - Transmit Excessive Collision Packet Counter

Register Index: 0x42 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>															
TXCL[11:0]															
Bits	Name			Description								R/W	Reset		
31:12	<i>Reserved</i>			<i>Reserved</i>											
11:0	TXCL			Transmit Excessive Collision Packet Counter. Incremented for each frame that experienced 16 collisions during transmission and was aborted.								R/W			

17.6.3.12 TNCL - Transmit Total Collision Counter

Register Index: 0x43 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>		TNCL[12:0]													

Bits	Name	Description	R/W	Reset
31:12	<i>Reserved</i>	<i>Reserved</i>		
12:0	TNCL	Transmit Total Collision Counter. Incremented by the number of collisions experienced during the transmission of a frame as defined as the simultaneous presence of signals on the DO and RD circuits (i.e. transmitting and receiving at the same time). Note, this register does not include collisions that result in an excessive collision condition)	R/W	

17.6.3.13 TJBR - Transmit Jabber Frame Counter

Register Index: 0x46 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>		TJBR(11:0)													

Bits	Name	Description	R/W	Reset
31:12	<i>Reserved</i>	<i>Reserved</i>		
11:0	TJBR	Transmit Jabber Frame Counter. Incremented for each oversized transmitted frame with an incorrect FCS value.	R/W	

17.6.3.14 TFCS - Transmit FCS Error Counter

Register Index: 0x47 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved		TFCS(11:0)														
Reserved		TFCS(11:0)														
Bits	Name		Description												R/W	Reset
31:12	Reserved		Reserved													
11:0	TFCS		Transmit FCS Error Counter. Incremented for every valid sized packet with an incorrect FCS value.												R/W	

17.6.3.15 TXCF - Transmit Control Frame Counter

Register Index: 0x48 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Reserved		TXCF(11:0)														
Reserved		TXCF(11:0)														
Bits	Name		Description												R/W	Reset
31:12	Reserved		Reserved													
11:0	TXCF		Transmit Control Frame Counter. Incremented for every valid size frame with a Type Field signifying a Control frame.												R/W	

17.6.3.16 TOVR - Transmit Oversize Frame Counter

Register Index: 0x49 Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>				TOVR(11:0)											
Bits	Name	Description				R/W	Reset								
31:12	Reserved	Reserved													
11:0	TOVR	Transmit Oversize Frame Counter. Incremented for each oversized transmitted frame with a correct FCS value.				R/W									

17.6.3.17 TUND - Transmit Undersize Frame Counter

Register Index: 0x4A Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
<i>Reserved</i>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>				TUND(11:0)											
Bits	Name	Description				R/W	Reset								
31:12	Reserved	Reserved													
11:0	TUND	Transmit Undersize Frame Counter. Incremented for every frame less than 64 bytes, with a correct FCS value.				R/W									

17.6.3.18 TFRG - Transmit Fragment Counter

Register Index: 0x4B Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	
<i>Reserved</i>																
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
<i>Reserved</i>				TFRG(11:0)												
Bits	Name		Description												R/W	Reset
31:12	<i>Reserved</i>		<i>Reserved</i>													
11:0	TFRG		Transmit Fragment Counter. Incremented for every frame less than 64 bytes, with an incorrect FCS value.												R/W	

17.6.4 Carry and Carry Mask Registers

The CAR1 and CAR2 registers allow the programmer to have a given statistics counter register generate an interrupt when it rolls over. The interrupt is triggered in the Network Accelerator handling the XAUI interface.

These interrupts are processed as follows: if the OverFlowEn bit field is asserted in the Network Accelerator StatCtrl register (0x0A3), then if any unmasked statistics counter overflows in the XAUI, the Network Accelerator receives an interrupt to be passed on to the PIC. (All the overflow interrupts are OR'd within the XAUI interface.)

For software to handle a statistics carry interrupt in the PIC:

1. PIC generates the interrupt.
2. User must first read the Network Accelerator Int register to find the source of the interrupt.
3. Then read CAR1 and CAR2 to determine the original source causing the overflow.
4. To clear the Network Accelerator of the interrupt, write a '1' to bit[2] of the Network Accelerator Int register.

17.6.4.1 CAR1 - Carry Register 1**Register Index: 0x4C Bits [31:0]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
C1 64	C1 127	C1 255	C1 511	C1 1K	C1 MAX	C1 MGV	Reserved								C1 RBY
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C1 RPK	C1 RFC	C1 RMC	C1 RBC	C1 RXC	C1 RXP	C1 RXU	C1 RAL	C1 RFL	C1 RCD	C1 RCS	C1 RUN	C1 ROV	C1 RFR	C1 RJB	C1 RDR

Carry Register One *

Bits	Name	Description	R/W	Reset
31	C164	Carry register 1 TR64 Counter Carry bit	R	0
30	C1127	Carry register 1 TR127 Counter Carry bit	R	0
29	C1255	Carry register 1 TR255 Counter Carry bit	R	0
28	C1511	Carry register 1 TR511 Counter Carry bit	R	0
27	C11k	Carry register 1 TR1K Counter Carry bit	R	0
26	C1MAX	Carry register 1 TRMAX Counter Carry bit	R	0
25	C1MGV	Carry register 1 TRMGV Counter Carry bit	R	0
24:17	Reserved	Reserved	R	
16	C1RBY	Carry register 1 RBYT Counter Carry bit	R	0
15	C1RPK	Carry register 1 RPKT Counter Carry bit	R	0
14	C1RFC	Carry register 1 RFCS Counter Carry bit	R	0
13	C1RMC	Carry register 1 RMCA Counter Carry bit	R	0
12	C1RBC	Carry register 1 RBCA Counter Carry bit	R	0
11	C1RXC	Carry register 1 RXCF Counter Carry bit	R	0
10	C1RXP	Carry register 1 RXPF Counter Carry bit	R	0
9	C1RXU	Carry register 1 RXUO Counter Carry bit	R	0
8	C1RAL	Carry register 1 RALN Counter Carry bit	R	0
7	C1RFL	Carry register 1 RFLR Counter Carry bit	R	0
6	C1RCD	Carry register 1 RCDE Counter Carry bit	R	0
5	C1RCS	Carry register 1 RCSE Counter Carry bit	R	0
4	C1RUN	Carry register 1 RUND Counter Carry bit	R	0
3	C1ROV	Carry register 1 ROVR Counter Carry bit	R	0
2	C1RFR	Carry register 1 RFRG Counter Carry bit	R	0
1	C1RJB	Carry register 1 RJBR Counter Carry bit	R	0
0	C1RDR	Carry register 1 RDRP Counter Carry bit	R	0

* Carry register bits are cleared on carry register write while respective bit is asserted

17.6.4.2 CAR2 - Carry Register 2

Register Index: 0x4D Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
<i>Reserved</i>														C2 TJB	C2 TFC	C2 TCF	C2 TOV

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C2 TUN	C2 TFG	C2 TBY	C2 TPK	C2 TMC	C2 TBC	C2 TPF	C2 TDF	C2 TED	C2 TSC	C2 TMA	C2 TLC	C2 TXC	C2 TNC	C2 TPH	C2 TDP

Carry Register 2 *

Bits	Name	Description	R/W	Reset
31:20	Reserved	Reserved	R	0
19	C2TJB	Carry register 2 TJBR Counter Carry bit	R	0
18	C2TFC	Carry register 2 TFCS Counter Carry bit	R	0
17	C2TCF	Carry register 2 TXCF Counter Carry bit	R	0
16	C2TOV	Carry register 2 TOVR Counter Carry bit	R	0
15	C2TUN	Carry register 2 TUND Counter Carry bit	R	0
14	C2TFG	Carry register 2 TFRG Counter Carry bit	R	0
13	C2TBY	Carry register 2 TBYT Counter Carry bit	R	0
12	C2TPK	Carry register 2 TPKT Counter Carry bit	R	0
11	C2TMC	Carry register 2 TMCA Counter Carry bit	R	0
10	C2TBC	Carry register 2 TBCA Counter Carry bit	R	0
9	C2TPF	Carry register 2 TXPF Counter Carry bit	R	0
8	C2TDF	Carry register 2 TDFR Counter Carry bit	R	0
7	C2TED	Carry register 2 TEDF Counter Carry bit	R	0
6	C2TSC	Carry register 2 TSCL Counter Carry bit	R	0
5	C2TMA	Carry register 2 TMCL Counter Carry bit	R	0
4	C2TLC	Carry register 2 TLCL Counter Carry bit	R	0
3	C2TXC	Carry register 2 TXCL Counter Carry bit	R	0
2	C2TNC	Carry register 2 TNCL Counter Carry bit	R	0
1	C2TPH	Carry register 2 TPFH Counter Carry bit	R	0
0	C2TDP	Carry register 2 TDRP Counter Carry bit	R	0

17.6.4.3 CAM1 - Carry Mask Register 1

Register Index: 0x4E Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
M1 64	M1 127	M1 255	M1 511	M1 1K	M1 MAX	M1 MGV	<i>Reserved</i>								M1 RBY
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M1 RPK	M1 RFC	M1 RMC	M1 RBC	M1 RXC	M1 RXP	M1 RXU	M1 RAL	M1 RFL	M1 RCD	M1 RCS	M1 RUN	M1 ROV	M1 RFR	M1 RJB	M1 RDR

When a mask bit is set to zero, the corresponding interrupt bit is allowed to cause interrupt indications to the Network Accelerator. Upon a statistics register overflow, the Carry bit is set in the Network Accelerator Int register, which is then passed on to the PIC.

Bits	Name	Description	R/W	Reset
31	M164	Mask register 1 TR64 Counter Carry bit Mask	R/W	1
30	M1127	Mask register 1 TR127 Counter Carry bit Mask	R/W	1
29	M1255	Mask register 1 TR255 Counter Carry bit Mask	R/W	1
28	M1511	Mask register 1 TR511 Counter Carry bit Mask	R/W	1
27	M11k	Mask register 1 TR1K Counter Carry bit Mask	R/W	1
26	M1MAX	Mask register 1 TRMAX Counter Carry bit Mask	R/W	1
25	M1MGV	Mask register 1 TRMGV Counter Carry bit Mask	R/W	1
24:17	Reserved	Reserved	R/W	
16	M1RBY	Mask register 1 RBYT Counter Carry bit Mask	R/W	1
15	M1RPK	Mask register 1 RPKT Counter Carry bit Mask	R/W	1
14	M1RFC	Mask register 1 RFCS Counter Carry bit Mask	R/W	1
13	M1RMC	Mask register 1 RMCA Counter Carry bit Mask	R/W	1
12	M1RBC	Mask register 1 RBCA Counter Carry bit Mask	R/W	1
11	M1RXC	Mask register 1 RXCF Counter Carry bit Mask	R/W	1
10	M1RXP	Mask register 1 RXPF Counter Carry bit Mask	R/W	1
9	M1RXU	Mask register 1 RXUO Counter Carry bit Mask	R/W	1
8	M1RAL	Mask register 1 RALN Counter Carry bit Mask	R/W	1
7	M1RFL	Mask register 1 RFLR Counter Carry bit Mask	R/W	1
6	M1RCD	Mask register 1 RCDE Counter Carry bit Mask	R/W	1
5	M1RCS	Mask register 1 RCSE Counter Carry bit Mask	R/W	1
4	M1RUN	Mask register 1 RUND Counter Carry bit Mask	R/W	1
3	M1ROV	Mask register 1 ROVR Counter Carry bit Mask	R/W	1
2	M1RFR	Mask register 1 RFRG Counter Carry bit Mask	R/W	1
1	M1RJB	Mask register 1 RJBR Counter Carry bit Mask	R/W	1
0	M1RDR	Mask register 1 RDRP Counter Carry bit Mask	R/W	1

17.6.4.4 CAM2 - Carry Mask Register 2

Register Index: 0x4F Bits [31:0]

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16		
<i>Reserved</i>														M2 TJB	M2 TFC	M2 TCF	M2 TOV
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
M2 TUN	M2 TFG	M2 TBY	M2 TPK	M2 TMC	M2 TBC	M2 TPF	M2 TDF	M2 TED	M2 TSC	M2 TMA	M2 TLC	M2 TXC	M2 TNC	M2 TPH	M2 TDP		

When a mask bit is set to zero, the corresponding interrupt bit is allowed to cause interrupt indications to the Network Accelerator. Upon a statistics register overflow, the Carry bit is set in the Network Accelerator Int register, which is then passed on to the PIC.

Bits	Name	Description	R/W	Reset
31:20	<i>Reserved</i>	<i>Reserved</i>	R/W	
19	M2TJB	Mask register 2 TJBR Counter Carry bit Mask	R/W	1
18	M2TFC	Mask register 2 TFCS Counter Carry bit Mask	R/W	1
17	M2TCF	Mask register 2 TXCF Counter Carry bit Mask	R/W	1
16	M2TOV	Mask register 2 TOVR Counter Carry bit Mask	R/W	1
15	M2TUN	Mask register 2 TUND Counter Carry bit Mask	R/W	1
14	M2TFG	Mask register 2 TFRG Counter Carry bit Mask	R/W	1
13	M2TBY	Mask register 2 TBYT Counter Carry bit Mask	R/W	1
12	M2TPK	Mask register 2 TPKT Counter Carry bit Mask	R/W	1
11	M2TMC	Mask register 2 TMCA Counter Carry bit Mask	R/W	1
10	M2TBC	Mask register 2 TBCA Counter Carry bit Mask	R/W	1
9	M2TPF	Mask register 2 TXPF Counter Carry bit Mask	R/W	1
8	M2TDF	Mask register 2 TDFR Counter Carry bit Mask	R/W	1
7	M2TED	Mask register 2 TEDF Counter Carry bit Mask	R/W	1
6	M2TSC	Mask register 2 TSCL Counter Carry bit Mask	R/W	1
5	M2TMA	Mask register 2 TMCL Counter Carry bit Mask	R/W	1
4	M2TLC	Mask register 2 TLCL Counter Carry bit Mask	R/W	1
3	M2TXC	Mask register 2 TXCL Counter Carry bit Mask	R/W	1
2	M2TNC	Mask register 2 TNCL Counter Carry bit Mask	R/W	1
1	M2TPH	Mask register 2 TPFH Counter Carry bit Mask	R/W	1
0	M2TDP	Mask register 2 TDRP Counter Carry bit Mask	R/W	1

17.7 XAUI PCS Registers

17.7.1 XAUI PCS-Register Summary Table

Table 17-4 presents registers for PCS-specific control and status functions:

Table 17-4. XAUI PCS-Specific Registers

Register Offset	Device Address Offset	Register	Type	Default
0x0000	Device Address 0x0000 - 0x0005	Control 1	R/W	0x2040
0x0001		Status 1	RO	
0x0002 - 0x0003		Reserved	RO	
0x0004		XS Speed Ability	RO	
0x0005		XS_Devices_Present Low	RO	
0x0006 - 0x0007		Reserved	RO	
0x0008		Status 2	RO	
0x0009 - 0x0017		Reserved	RO	
0x0018		XGXS Lane Status	RO	
0x0019		PHY/DTE XGXS Test Control	R/W	
0x001A - 0xFFFF		Reserved		

17.7.1.1 Control 1

Register ID: **0x0000 Bits [15:0]**

Address
Offset:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reset	Loop Back	Speed Select	Reser ved	Low Power Mode	Reserved				Speed Select		Speed Select		Reserved		

Bits	Name	Description	R/W	Reset
15	RESET	Setting this bit will cause the sub-modules in the XAUI core to be reset. This bit is self-clearing.	RW	0
14	LOOP BACK	0 = normal operation 1 = connects the transmit output (tx_code_grpr0/tx_code_grpr1) back into the receive input (rx_ua_r0/rx_ua_r1) Note: For loopback to work properly, the transmit clock (tx_clk) must also be shunted back into the receive clock inputs (rx_clkiX). This gating must be performed pre-clock-tree.	RW	0

Bits	Name	Description	R/W	Reset
13	Speed Select	0 = Unspecified 1 = 10 Gbps and above	RO	1
12	Reserved	Reserved		0
11	Low-Power Mode	0 = Normal Operation 1 = Low-Power Mode When low-power mode is enabled, the tx_reset and rx_reset[3:0] signals are asserted and the output low-power is asserted. In order to truly enter a low-power state, the tx_clk and rx_clkiX (X = 0,1,2,3) must be disabled pre-clock-tree using the low-power output of M-XGXS. The mmd_mdc clock can also be disabled; however, if this is the case, low power can only be removed by asserting mstr_reset.	RW	0
10:7	Reserved	Reserved		0
6	Speed Select	0 = Unspecified 1 = 10 Gbps and above	RO	1
5:2	Speed Select	1xxx = Reserved x1xx = Reserved xx1x = Reserved 0001 = Reserved 0000 = 10 Gbps	RO	0
1:0	Reserved	Reserved		0

17.7.1.2 Status 1

Register ID: 0x0001 Bits [15:0]

Address
Offset: 0x

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved						Fault			Reserved			TX/RX Link Status	Low-Power Enable	Reserved	

Bits	Name	Description	R/W	Reset
15:8	Reserved	Reserved		0
7	Fault	0 = No Fault detected 1 = Fault detected; set if either TX or RX Fault bit is set in Status 2 [11:10] register	RO	0
6:3	Reserved	Reserved		0
2	PHY/DTE Transmit/Receive Link Status	0 = Link is down 1 = Link is up Latched Low	RO	0
1	Low-Power Enable	0 = Low-Power mode not enable 1 = Low-Power mode enabled	RO	1
0	Reserved	Reserved		0

17.7.1.3 XS Speed Ability

Register ID: 0x0004 Bits [15:0]

Address

Offset:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>														10G Capable	

Bits	Name	Description	R/W	Reset
15:1	Reserved	Reserved		0
0	10G Capable	0 = Not capable of 10 Gbps operation 1 = Capable of 10 Gbps operation	RO	1

17.7.1.4 XS_Devices_Present Low

Register ID: 0x0005 Bits [15:0]

Address

Offset:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>						DTE_XS Present	PHY_XS Present	PCS Present	WIS Present	PMD/PMA Present	Clause-22 Regs Present				

Bits	Name	Description	R/W	Reset
15:6	Reserved	Reserved		0
5	DTE_XS_Present	0 = DTE_XS not present 1 = DTE_XS is present	RO	1
4	PHY XS Present	0 = PHY_XS not present 1 = PHY_XS is present	RO	0
3	PCS Present	0 = PCS not present 1 = PCS is present	RO	0
2	WIS Present	0 = WIS not present 1 = WIS is present	RO	0
1	PMD/PMA Present	0 = PMD/PMA not present 1 = PMD/PMA is present	RO	0
0	Clause-22 Registers Present	0 = Clause-22 Registers not present 1 = Clause-22 Registers are present	RO	0

17.7.1.5 Status 2**Register ID: 0x0008 Bits [15:0]****Address****Offset:**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Device Present	Reserved	Transmit Fault	Receive Fault												Reserved

Default value: 0x0000

Bits	Name	Description	R/W	Reset
15:14	Device Present	00 = No device responding at this address 01 = No device responding at this address 10 = Device responding at this address 11 = No device responding at this address	RO	10
13:12	Reserved	Reserved		00
11	Transmit Fault	0 = No transmit fault 1 = Transmit fault Latched High, clear on read	RO	0
10	Receive Fault	0 = No receive fault 1 = Receive fault Latched High, clear on read	RO	0
9:0	Reserved	Reserved		0

17.7.1.6 XGXS Lane Status**Register ID: 0x0018 Bits [15:0]****Address****Offset: 0x**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved	PHY/DTE XGXS Lane Alignment Status	Pattern Testing Enable	PHY XGXS Loopback Enable		Reserved							Lane 3 SYNCed	Lane 2 SYNCed	Lane1 SYNCed	Lane 0 SYNCed

Bits	Name	Description	R/W	Reset
15:13	Reserved	Reserved		0
12	PHY/DTE XGXS Lane Alignment Status	0 = Lanes not aligned 1 = Lanes aligned	RO	0
11	Pattern Testing Enable	0 = (PHY/DTE)XS is not able to generate test patterns 1 = (PHY/DTE)XS is able to generate test patterns	RO	1
10	PHY XGXS Loop Back Enable	0 = PHY XGXS is not able to perform Loop Back function 1 = PHY XGXS is able to perform Loop Back function	RO	1
9:4	Reserved	Reserved		0

Bits	Name	Description	R/W	Reset
3	Lane 3 SYNCed	0 = Lane <i>n</i> is not SYNCed 1 = Lane <i>n</i> is SYNCed	RO	0
2	Lane 2 SYNCed		RO	0
1	Lane 1 SYNCed		RO	0
0	Lane 0 SYNCed		RO	0

17.7.1.7 PHY/DTE XGXS Test Control

Register ID: 0x0019 Bits [15:0]

Address

Offset:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<i>Reserved</i>										Transmit /Receive Test Pattern Enable	Test Pattern Select				

Bits	Name	Description	R/W	Reset
15:3	Reserved	<i>Reserved</i>		0
2	Transmit/Receive Test Pattern Enable	0 = Transmit/Receive test pattern disabled 1 = Transmit/Receive test pattern enabled	R/W	0
1:0	Test Pattern Select	00 = High-frequency test pattern 01 = Low-frequency test pattern 10 = Mixed-frequency test pattern 11 = Reserved	R/W	00



Chapter 18 USB Interface

18.1 Introduction

This chapter discusses the XLS Processor USB Interface. The XLS6xx, XLS4xx-Lite, XLS4xx and XLS2xx devices have two USB interfaces/ports. The XLS108 and XLS104 devices have only one USB port which uses Port_0.

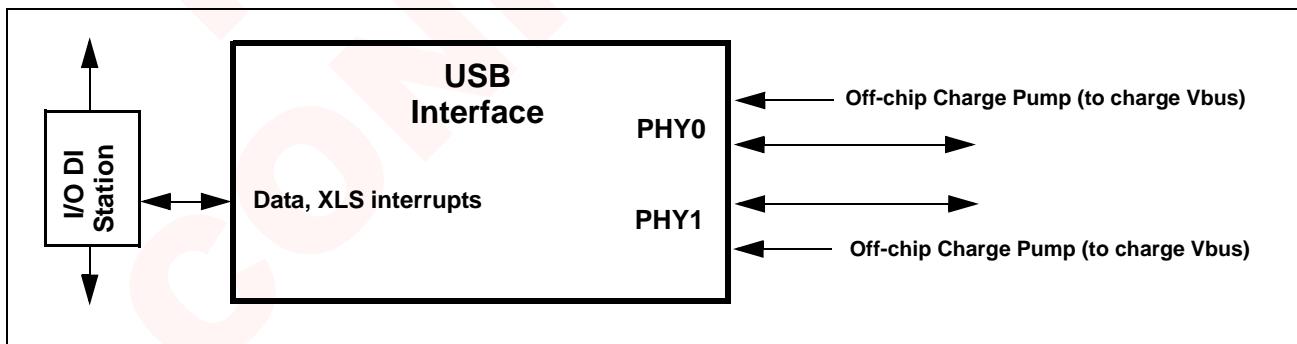
The XLS-Processor USB interfaces support the following capabilities and features.

- Root Hub functionality in Host mode
- Integrated PHY
- Ping and split transactions
- In-band wake-up protocol in Device mode
- Multipoint application support
- Control, Bulk, or Interrupt traffic. Isochronous traffic is supported in host mode.
- High-speed (480 Mbps), Full speed (12 Mbps), and low speed (1.5Mbps) operation
- Number of Endpoints: 3 Bidirectional in addition to Control Endpoint 0
- Supports 4 IN Endpoints including Control Endpoint 0 (Device mode)
- Maximum allowed packet size is 1024 bytes for a periodic USB packet, and 512 bytes for a non-periodic packet
- Dynamic FIFO sizing with threshold setting in Device mode
- Packet prefetching in Host mode

The XLS-Processor USB interfaces are fully compliant with the following specifications:

- USB Specification Rev. 2.0
- OHCI Specification Rev 1.0a
- EHCI for USB Specification Rev 1.0

Figure 18-1. XLS Processor USB Interface



18.2 Theory of Operation

This discussion divides the theory of operation into: USB Interface Overall Operation, Device Controller Operation, and Host Controller Operation.

18.2.1 Overall Operation

The XLS USB Interface connects through an I/O Distributed Interface station on the DI ring. The DI station operates autonomously and does not require user programming for data or interrupt conveyance. However, fine adjustments to data transfer efficiency are possible using specific control registers:

- In the [USB_GEN_CTRL3](#) register, the PREFETCH_SIZE field controls how many cachelines are read.
- The [USB_INTERRUPT_STATUS](#) and [USB_INTERRUPT_ENABLE](#) registers determine the source of an interrupt.

18.2.1.1 Major Operating Modes

The XLS USB interface can be configured to function as:

- One USB Device Controller functioning on Port_0, or
- Two USB Host Controllers on Ports 0 and 1 configured as shown in [Table 18-1](#).

This is illustrated in the diagrams below.

Figure 18-2. Interface as USB Device

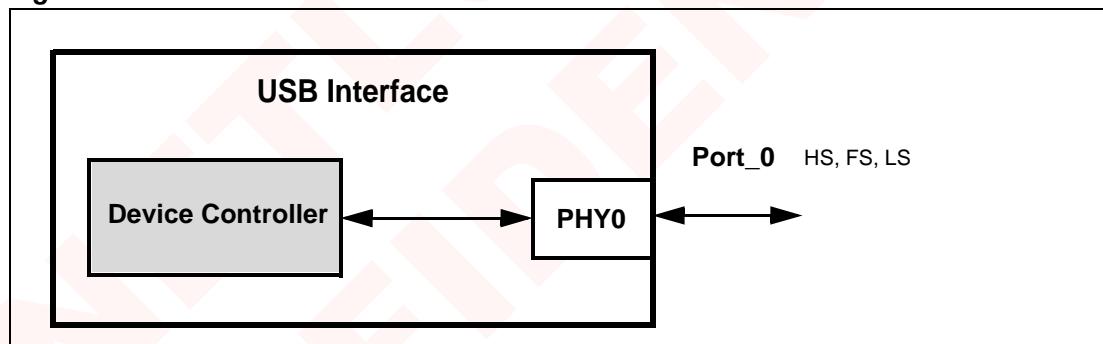


Figure 18-3. Interface as USB Host Controller

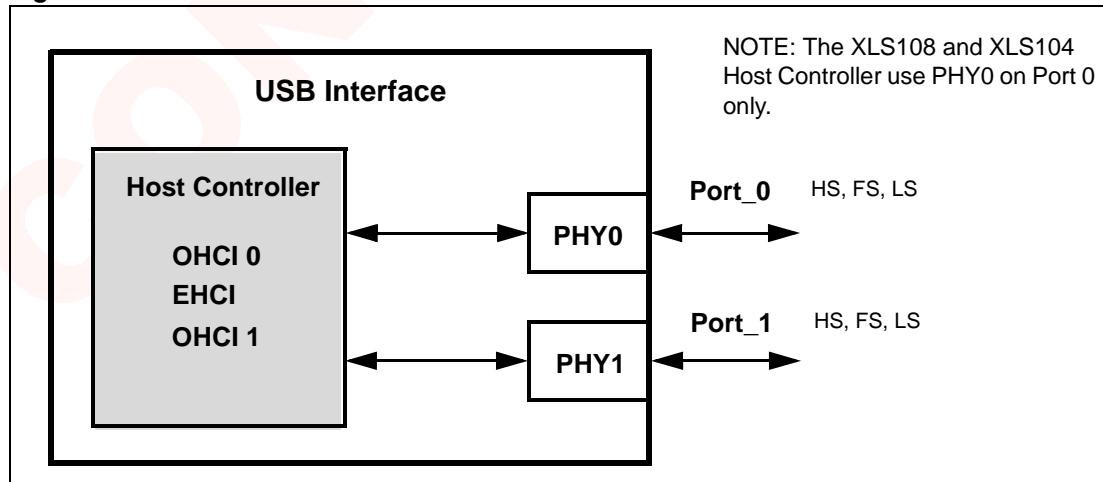


Table 18-1. Host Mode Configurations

Port_0	Port_1
EHCI	EHCI
ECHI	OHCI 1
OHCI 0	EHCI
OHCI 0	OHCI 1
EHCI	x
OHCI 0	x
x	EHCI
x	OHCI 1

18.2.1.2 USB Interface Signals

The XLS USB Interface port signals are listed in [Table 18-2](#).

Table 18-2. USB Interface Signals (Per Port) - (XLS1xx devices use Port 0 only)

Signal Name	Number of Signals	Direction	Description
USB_RKELVIN	1	IO	R _{EXT} voltage feedback
USB_DM_0	1	IO	USB Port_0 D+ signal
USB_DP_0	1	IO	USB Port_0 D- signal
USB_DM_1	1	IO	USB Port_1 D+ signal
USB_DP_1	1	IO	USB Port_1 D- signal
USB_VDD	1	I	Digital power supply from dedicated off-chip supply
USB_VDDA33C	1	I	3.3V analog power supply for common block
USB_VDDA33T_0	1	I	3.3V analog power supply for Port_0 transceiver block
USB_VDDA33T_1	1	I	3.3V analog power supply for Port_1 transceiver block
USB_VSS	1	I	Digital ground supply from dedicated off-chip supply
USB_VSSA33C	1	I	3.3V analog ground supply for common block
USB_VSSA33T_0	1	I	3.3V analog ground supply for Port_0 transceiver block
USB_VSSA33T_1	1	I	3.3V analog ground supply for Port_1 transceiver block
USB_XI	1	I	Crystal oscillator XO pin. The crystal must have a fundamental frequency of 48 MHz, with a frequency tolerance of +/-400 ppm, peak jitter of +/-100 ps, and an output differential voltage of no less than 500 mV with respect to the XI signal.
USB_XO	1	I	Crystal oscillator xi pin; 48 MHz

18.2.2 Device Controller Theory of Operation

This section discusses theory of operation of the Device Controller portion of the USB Interface.

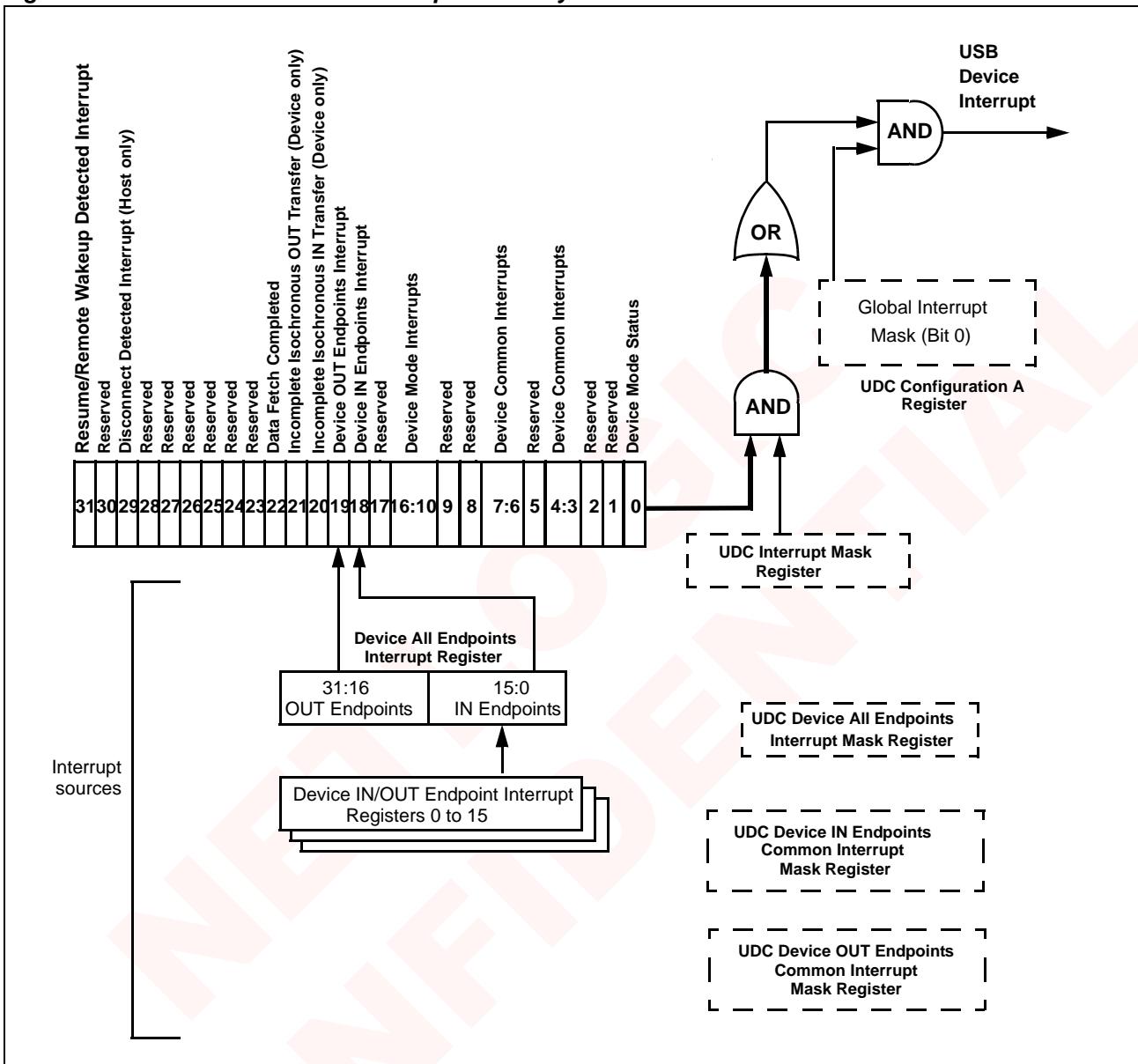
18.2.2.1 Device Operating Mode

Operation of the USB interface for Device Mode is selected using IO_AD[22]. For more information, see [Section 23.3.5 GPIO System Control Registers](#) in this manual or Chapter 8 in the XLS Data Sheet.

When configured for Device Mode, the interface functions as one USB Device Controller using Port_0.

18.2.2.2 Internal Interrupts in the USB Device Controller

The XLS USB interface's Device Controller has a variety of internal interrupts. These interrupts are managed as shown in [Figure 18-4](#). The Device Controller's single interrupt output is routed to the XLS's Programmable Interrupt Controller for handling there. When an interrupt is detected, the XLS must determine how to handle the interrupt, then interact with the USB driver software as needed to handle the interrupt within the Device Controller.

Figure 18-4. Device Controller Interrupt Hierarchy

18.2.3 Host Controller Theory of Operation

This section discusses theory of operation of the Host Controller portion of the USB Interface.

18.2.3.1 Host Operating Modes

Operation of the USB interface for Host Mode is selected using a bootstrap bit. For information, refer to the GPIO chapter in this document.

The USB Interface may be configured for any of several forms of Host Mode operation. These are explained in [Section 18.2.1.1 Major Operating Modes](#).

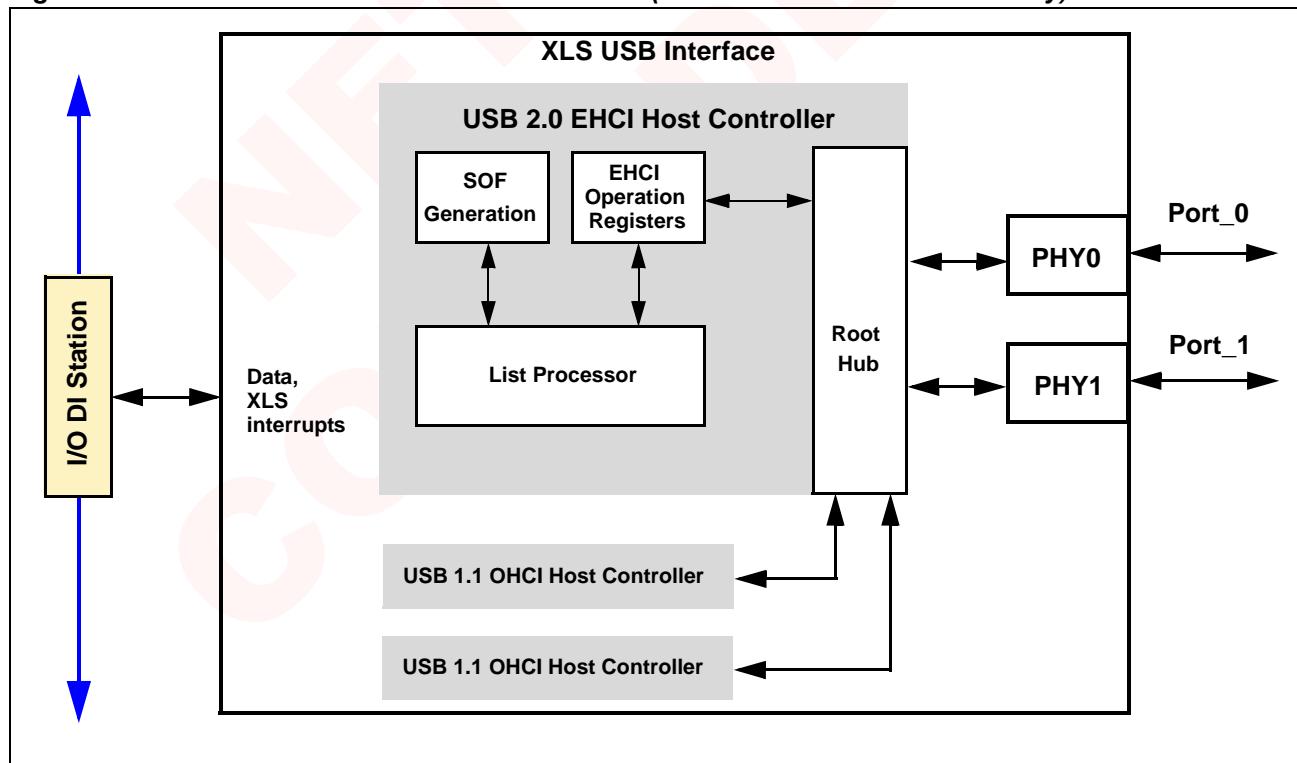
USB 2.0 Supported Features:

- Complies with the following specifications:
 - USB 2.0 Specification
 - OHCI Specification Rev 1.0a
 - EHCI Specification Rev 1.0
- Supports ping and split transactions
- Packet prefetching

18.2.3.2 Overview

The Host Controller is fully compliant with the USB 2.0 specification, Enhanced Host Controller Interface (EHCI) Specification, Revision 1.0, and the Open Host Controller Interface (OHCI) Specification Release 1.0a. The controller supports high-speed, 480-Mbps transfers using an EHCI Host Controller, as well as full and low speeds through one or two integrated OHCI Host Controllers.

Figure 18-5. USB Host Controller Main Modules - (XLS1xx devices use Port 0 only)



18.2.3.3 EHCI Controller Functional Blocks

- List Processor
- Operational Registers
- Start-of-Frame (SOF) Generator
- Root Hub (RH)

List Processor

The List Processor block is the main controller. This block performs the list service flow, which is set up by the Host Controller Driver according to the priority set in the Operational registers. It interfaces and act with the Packet Buffer, EHCI Operational registers, SOF Generators, and the Root Hub.

Operational Registers

The EHCI Capability and Operational registers are in this block.

Start-of-Frame (SOF) Generator

SOF packets are generated with an SOF counter to generate micro-SOFs for each microframe. Microframe duration is derived from the Frame Length Adjustment (FLADJ) register value, which can be configured through the Application Strap Signals interface. The same FLADJ values must be configured through strap signals. This ensures that the host microframe duration and per-port microframe duration remain the same.

Root Hub (RH)

The Root Hub propagates Reset and Resume signals to downstream ports and handles port connections and disconnections. In addition, the Root Hub is implemented with Port Router logic to route the ports to either the EHCI Host Controller or OHCI Host Controller.

18.3 Global Programming Model

This section discusses programmatic set up for operation of the interface, and programmatic access to the interface.

There are two main areas in which setup must be done:

- System-level setup
- Interface-level setup, the nature of which depends on which mode the interface will be in

18.3.1 System Setup for Startup and Initialization

The inward side of the interface is tightly integrated into the XLS processor's I/O distributed interconnect. Operation at the system level for interface activity involves the following setup.

18.3.1.1 System Bridge Controller Setup

The SBC resides in the XLS's Memory and Distributed Interconnect Hub. (See Chapter 8, External Memory and I/O Configuration, for details.) This handles device-internal address mappings.

Setup here for USB Interface access involves:

- 1) XLS_IO_BAR, which is set up once to define the base address for the entire Peripheral and I/O Configuration region in which all internal peripheral configuration header registers and control and status registers lie. All USB Interface internal registers lie at a group offset from this base, and each register is at an additional offset or index from that group offset.
- 2) Once XLS_IO_BAR has been set up, and the USB Interface operating mode configured in the appropriate GPIO register field for this purpose, USB Interface registers can be accessed. See [Section 18.6 USB Interface Registers](#) for more information.

18.3.1.2 Interrupt Setup in the XLS PIC

This involves setup in two areas:

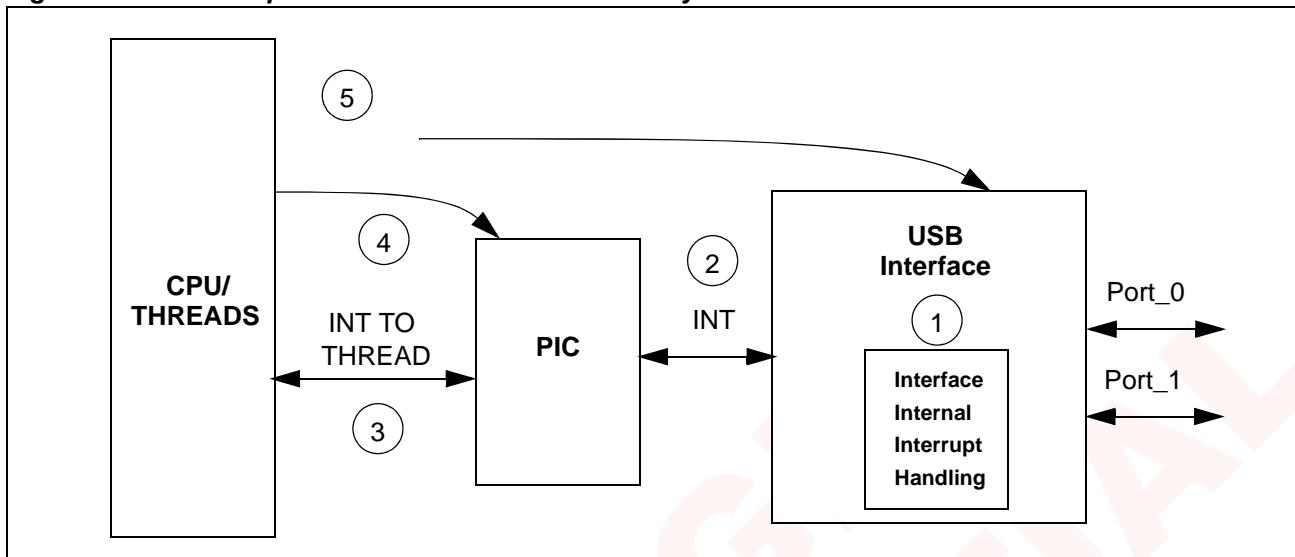
- 1) Interrupt Redirection Table (IRT) in the PIC. See discussions in Chapter 12, Programmable Interrupt Controller, of this manual.
- 2) The EIRR for each nCPU that will interact with this interface. See discussion of EIRR registers in CPUs.

The USB Interface is integrated into the system interrupt architecture as follows:

18.3.1.3 USB Mode Setup

The USB interface is configured for Device or Host mode operation by means of a bootstrap bit. The selection is based on the level of the IO_AD9 pin, latched at reset time. The options are:

- 0 = Run Device mode operation
- 1 = Run Host Mode operation

Figure 18-6. Interrupt Model for USB Interface and System Architecture

1. An interrupt occurs internally in the USB interface and is captured in a USB interface register.
2. The USB interface generates an interrupt to the PIC.
3. The PIC generates an interrupt to a target nCPU/thread.
4. The nCPU reads the PIC to determine which system device generated the interrupt. For this example, the PIC identifies the USB interface as the sourcing system device.
5. The nCPU then checks the identified system device to determine interrupt source. In this case, the nCPU must read the appropriate USB ISR. For example, the Host Controller UHCInterruptStatus register or the Device Controller's UDCInterrupt register. (See also Chapter 12, Programmable Interrupt Controller, in this manual.)

18.3.2 Clock and Reset

The USB Controller and USB PHY resets are generated by registers residing in UsbCtrlr.

There is a flush mechanisms used to synchronize the bridge and the USB block. The flush is used when the USB core needs to be reset but the XLS system does not.

When the system must reset the USB core but not the remainder of the XLS, the following steps must be followed:

1. Put the USB core in reset with register write.
2. Start link flush with register write.
3. Wait until all requests and responses are flushed from the bridge (as there is no specific status bit indicating flush completion, this step requires waiting several cycles before clearing the USB_FLUSH bit in the USB_GEN_CTRL3 register).
4. Stop link flush with register write.
5. Pull the USB core out of reset with register write.

18.3.3 Errors and Events Handling for USB Interface in the System Architecture

Note that there are event analysis bit fields for the USB Interface in the System Bridge Control registers 54 and 56. These may be used for performance analysis. See Chapter 8 description of SBC registers.

18.4 Host Programming Model

18.4.0.1 Setup for Host Mode

The USB interface is configured for Host Mode operation by strapping the IO_AD9 signal to a logical '1' at reset time. Once the interface is configured, GPIO RESET configuration register bit 22 indicates the USB mode (0 for Device and 1 for Host). For more information refer to [Section 18.2.2.1 Device Operating Mode](#).

Interrupt Setup in the USB Host Controller

1. Enable USB interface subsystem interrupt reporting

Enable summary interrupt for the EHCI and/or OHCI host controllers by writing 1 to the EHCI, OHCI1 or OHCI2 bit of the UBS_INTERRUPT_ENABLE register.

2. Enable EHCI host controller interrupt reporting

Once the USB interface subsystem EHCI summary interrupt has been enabled, one also needs to enable the USBINTR register in the EHCI host controller. Please refer to the standard document "Enhanced Host Controller Interface Specification for Universal Serial Bus Revision 1.0" for detail. This document will be later referred to as the EHCI standard document.

3. Enable OHCI host controller interrupt reporting

Once the USB interface subsystem OHCI summary interrupt has been enabled, one also needs to enable the interrupts from the OHCI host controller. The OHCI host controller interrupts can be enabled by setting HcInterruptEnable register or clearing HcInterruptDisable register. Please refer to the standard document "OpenHCI Open Host Controller Interface Specification for USB Release 1.0a" for detail. This document will later be referred to as the OHCI standard document.

Reset Sequence for Host Mode

1. Release PHY reset

Right after chip reset release, one must first release the PHY reset. This is done by writing '1' to the PhyRstN bit in the USB_GEN_CTRL2 register.

The PHY port reset also needs to be released. Write '1' to PhyPortRstN0 to release the port reset for Port_1, and write '1' to the PhyPortRstN1 for port 2. The PHY port reset bits can be found in the USB_GEN_CTRL2 register.

2. Release Host reset

The Host reset HostRstN bit in the USB_GEN_CTRL1 register can be released only after the PHY and PHY port resets are released. The HostRstN should be released at least one ms after the release of the PHY reset. The HostRstN is released when '1' is written into it.

3. Initialize the EHCI host controllers

The following initialization sequence for the EHCI host controller is provided as an example. Please refer to the standard document "Enhanced Host Controller Interface Specification for Universal Serial Bus Revision 1.0" for more detail.

- a. Issue Software reset to the EHCI host controller. This is done by writing 1 to the HCRESET bit in the USBCMD register.
- b. Set the PORTSC1 and PORTSC2 register to initialize the ports. With the exception of bits WKOC_E, WKDSCNNT_E, WKCNNT_E, and PP, all other bits are initialized to '0'. Bits WKOC_E, WKDSCNNT_E, WKCNNT_E, and PP are initialized to '1'.

- c. Clear the EHCI host controller interrupt by writing ‘1’s to all the interrupt bits of the USBINTR register.
 - d. Initialize the PERIODICLISTBASE register to indicate to the EHCI host controller the beginning address of the Periodic Frame list in system memory.
 - e. Enable the USB interface subsystem EHCI summary interrupt, and enable EHCI host-controller interrupts.
 - f. Set the CONFIGFLAG register to ‘1’, setting the port routing controller logic default-routes all ports to the EHCI host controller.
 - g. Read the HCSPARAMS register for the Port Power Control (PCC) bit. If the PPC bit is set to ‘1’, set the PortPower bit in the PortSC register to ‘1’.
4. Initialize the OHCI host controller for Port N
- The following initialization sequence for the OHCI host controller is provided as an example. Please refer to the standard document “OpenHCl Open Host Controller Interface Specification for USB Release 1.0a” for detail.
- Note:** The OHCI host controller_1 can only control USB Port_1. The OHCI host controller_2 can only control USB port 2. Each OHCI host controller requires its own separate initialization.
- a. Put the OHCI host controller to the OPERATIONAL state by writing to the HcControl register.
 - b. Set the HcHCCA register to specify the Host Controller Communication Area in the system memory.
 - c. Clear the HcDoneHead register to ‘0’.
- Note that the interrupt for this OHCI host controller is not enabled until a device attachment has been detected and the root hub has routed the port to this host controller.

5. Device Attachment

Once the USB interface subsystem is out of reset, and the EHCI and/or OHCI host controllers are initialized, the host controllers will be waiting for the attachment of an USB device.

The following discusses the attachment for high-speed and full-speed devices separately. For high speed attachment the EHCI host controller will be controlling the device. For full speed attachment the OHCI host controller will be controlling the device.

(5.1) High Speed Attachment

Whenever a high speed device is attached to one of the ports, an EHCI interrupt will be reported. The “Port Change Detected” status bit in the USBSTS register (please refer to the EHCI standard document) will be set.

By reading the "Connect Status Change" and "Current Connect Status" bit of the PORTSC for port N, one can determine if the port is connected to a device.

Next, initiate the port reset for the connected port by writing ‘1’ to the “Port Reset” bit and clear the “Port Enabled/Disabled” bit in the PORTSC register of the port.

After 50 ms, wait for the completion of the port reset sequence by reading the “Port Reset” bit in PORTSC. The port reset sequence is completed if the “Port Reset” bit returns to ‘0’. If the connected device is a high speed device, the “Port Enabled/Disabled” bit will be set to ‘1’.

If the “Port Enabled/Disabled” bit is set to ‘0’, it indicates a full speed device has been attached. The “Port Owner” bit in the PORTSC register should be set to ‘1’, thereby switching control of the port to the OHCI controller for the given port. Note that this

switching of port ownership from EHCI to OHCI will generate another EHCI interrupt, with the “Port Change Status” bit set in the USBSTS register.

If the “Run/Stop” bit in the USBCMD register is not yet set to ‘1’, it should be initialized to ‘1’ when there is a high-speed device connected to one of the ports. If there are no high-speed devices connected, the bit can be set to ‘0’. Writing ‘1’ to the “Run/Stop” bit enables the EHCI host controller to proceed with execution of the transaction schedule.

6. Full Speed Attachment

Upon an EHCI switching of port ownership to OHCI host controller, the “SetGlobalPower” bit in the HcRhStatus (refer to the OHCI standard document), should be set to ‘1’ to enable the port power.

The USB interface subsystem OHCI summary interrupt should be enabled. All the other OHCI host controller interrupts should also be enabled.

Read the CurrentConnectStatus and ConnectStatusChange bits of the HcRhPortStatus[1] for the OHCI host controller to determine if the host controller detected the connected device.

Initiate a port reset sequence by setting the SetPortReset bit and clear the ConnectStatusChange bit (in the HcRhPortStatus[1] register. This should result in a OHCI interrupt with the RootHubStatusChange set to ‘1’ in the HcInterruptStatus register.

The schedule for a type of transaction (Bulk, Periodic/Asynchronous, Isochronous, and Control) is to be enabled with the schedule’s respective “ListEnabled” and “ListFilled” bits. The “ListEnabled” bits can be found in the HcControl register. The “ListFilled” bit can be found in the HcCommandStatus register. Please refer to the OHCI standard document.

18.5 Device Programming Model

This section describes the interface setup for startup and initialization.

18.5.1 Interface Setup

The USB interface is configured for Device Mode operation by strapping the IO_AD9 signal to a logical '0' at reset time. Once the interface is configured, GPIO RESET configuration register bit 22 indicates the USB mode (0 for Device and 1 for Host). For more information refer to [Section 18.2.2.1 Device Operating Mode](#).

18.5.1.1 Interrupt Setup in the USB Device Controller

Enable the following fields in the USB_Interrupt_Enable register

- Device Interrupt Enable = 1
- Phy Interrupt Enable = 1

18.5.1.2 Reset Sequence for Device Mode Operation

1. After XLS system reset, the USB Device controller and PHY are still in reset.
2. Set the USB_VBUS_TIMER to an appropriate debounce value as required by the system. The default is 1msec but an appropriate value could be in the range of 100msec depending on the system.
3. Once the connect is detected (PHY Interrupt will be asserted), the PHY and the Device controller have to be reset separately. First bring the PHY out of reset. If the clock to the PHY is from the crystal, wait for 810 us for the PHY clock to stabilize and be available.
4. Now bring the Device Controller out of reset.

18.5.2 Core Initialization

The application must perform the core initialization sequence. If the B plug is connected during power-up, the Current Mode of Operation bit in the Core Interrupt register (UDC_GINTSTS.CurMod) reflects the device mode.

This section explains the initialization of the USB device controller after power-on. All core global registers are initialized according to the core's configuration as follows.

1. Read the User Hardware Configuration registers (UDC_GHWCFG1, 2, 3, and 4) to find the configuration parameters selected for the USB device controller.
2. Program the following fields in the Global AHB Configuration (UDC_GAHBCFG) register.
 - DMA Mode bit
 - AHB Burst Length field
 - Global Interrupt Mask bit = 1
3. Program the following fields in UDC_GUSBCFG register.
 - PHY Interface bit
 - HS/FS TimeOUT Time-Out Calibration field
 - USB Turnaround Time field

18.5.2.1 Device Initialization

The application must perform the following steps to initialize the core as a device on power-up.

1. Program the following fields in the UDC_DCFG register.
 - Device Speed

- Non-Zero-Length Status OUT Handshake
 - Periodic Frame Interval (when periodic endpoints are supported)
2. Program the Device threshold control register. This is required only if you are using DMA mode and you are planning to enable threshold setting.
 3. Program the UDC_GINTMSK register to unmask the following interrupts.
 - USB Reset
 - Enumeration Done
 - Early Suspend
 - USB Suspend
 - SOF
 4. Wait for the UDC_GINTSTS.USBReset interrupt, which indicates a reset has been detected on the USB and lasts for about 10 ms. On receiving this interrupt, the application must perform the steps listed in “Initialization on USB Reset” on page 751.
 5. Wait for the UDC_GINTSTS.EnumerationDone interrupt. This interrupt indicates the end of reset on the USB. On receiving this interrupt, the application must read the UDC_DSTS register to determine the enumeration speed and perform the steps listed in “Initialization on Enumeration Completion” on page 752.

At this point, the device is ready to accept SOF packets and perform control transfers on Control Endpoint 0.

18.5.3 Modes of Operation

The application operates the core in DMA mode where the core fetches the data to be transmitted or updates the received data on the AHB.

18.5.3.1 DMA Mode

The USB device controller uses the AHB Master interface for transmit packet data fetch (AHB to USB) and receive data update (USB to AHB). The AHB Master uses the programmed DMA address (UDC_DIEPDMAAn/UDC_DOEPDMAAn register) to access the data buffers.

Transfer-Level Operation

The application is interrupted only after the programmed transfer size is transmitted or received (provided the USB device controller detects no Timeout/CRC Error). All the USB errors are handled by the controller itself.

Transaction-Level Operation

This mode is similar to transfer-level operation, with the programmed transfer size equal to one packet size (maximum size or short packet).

18.5.3.2 Threshold Setting in DMA mode

The application can program the core to do FIFO threshold setting when operating as a device in DMA mode. With threshold support, the core can be configured to operate with less than maximum packet size FIFOs for a particular endpoint. This results in a smaller FIFO requirement when compared to non-threshold setting mode.

- The core allows both receive and transmit FIFO threshold setting.
- Device Threshold Control Register bit DTHRCTL.RxThrRn must be set to enable receive threshold setting. DTHRCTL.RxThrLen specifies the receive threshold size.

- Transmit uses separate Threshold Enable controls for isochronous and non-isochronous endpoints. Bits DTHRCTL.NonISOThrEn and DTHRCTL.ISOThrEn specify these Threshold Enable controls.
- The register field DTHRCTL.TxThrLen specifies the transmit threshold length and is common for isochronous and non-isochronous endpoints. The minimum threshold length supported by the core is four DWORDs.
- Threshold enable controls cannot be changed randomly. The application can set or reset the threshold enable bits only after ensuring that the core is not programmed to do any transfers (FIFOs are flushed, NAK bits are set, all endpoints are disabled).
- One of the limitation of threshold setting mode is in ping protocol, and could violate PING protocol. A PING token will be responded with an ACK handshake and the following OUT token could result in a NAK handshake. This behavior is a result of receive fifo overflow, and cannot be avoided in threshold setting mode. This scenario will not occur if there are no overflows.

When transmit threshold setting is enabled, the core starts transmitting data on the USB for a particular endpoint when there is threshold amount of data available in the corresponding transmit FIFO.

When receive threshold setting is enabled, the core starts transferring data from the receive FIFO to the system memory as soon as there is threshold amount of data available in the receive FIFO. Any underrun or overflow conditions are handled by the core internally.

18.5.4 Device Programming Model — Endpoint Initialization

18.5.4.1 Initialization on USB Reset

- Set the NAK bit for all OUT endpoints UDC_DOEPCTLn.SNAK = 1 (for all OUT endpoints)
- Unmask the following interrupt bits
 - UDC_DAINTMSK.INEP0 = 1 (control 0 IN endpoint)
 - UDC_DAINTMSK.OUTEP0 = 1 (control 0 OUT endpoint)
 - UDC_DOEPMSK.SETUP = 1
 - UDC_DOEPMSK.XferCompl = 1
 - UDC_DIEPMSK.XferCompl = 1
 - UDC_DIEPMSK.TimeOut = 1
- To transmit or receive data, the device must initialize more registers as follows: In DMA mode, UDC_GINTMSK.NPTxFEmpMsk and UDC_GINTMSK.RxFLvIMsk must be masked.
-
- Set up the Data FIFO RAM for each of the FIFOs (only if Dynamic FIFO Sizing is enabled)
 - Program the UDC_GRXFSIZ Register, to be able to receive control OUT data and setup data. If threshold setting is not enabled, at a minimum, this must be equal to 1 max packet size of control endpoint 0 + 2 DWORDs (for the status of the control OUT data packet) + 10 DWORDs (for setup packets). If threshold setting is enabled, at a minimum, this must be equal to $2^* (\text{Rx_threshold_length}/4 + 1) + 2$ DWORDs (for the status of the control OUT data packet) + 10 DWORDs (for setup packets)
 - Program the UDC_GNPTXFSIZ Register to be able to transmit control IN data. At a minimum, this must be equal to 1 max packet size of control endpoint 0. If threshold setting is enabled, this can be programmed to less than one max packet size.

6. Program the following fields in the endpoint-specific registers for control OUT endpoint 0 to receive a SETUP packet DOEPPTSIZ0.SetUP Count = 3 (to receive up to 3 back-to-back SETUP packets) In DMA mode, DOEPDMA0 register with a memory address to store any SETUP packets received

At this point, all initialization required to receive SETUP packets is done, except for enabling control OUT endpoint 0 in DMA mode.

18.5.4.2

Initialization on Enumeration Completion

1. On the Enumeration Done interrupt (UDC_GINTSTS.EnumDone, read the UDC_DSTS register to determine the enumeration speed.
2. Program the DIEPCTL0.MPS field to set the maximum packet size. This step configures control endpoint 0. The maximum packet size for a control endpoint depends on the enumeration speed.
3. Program the UDC_DOEPCTL0 register to enable control OUT endpoint 0, in order to receive a SETUP packet
 - UDC_DOEPCTL0.EPEna = 1

At this point, the device is ready to receive SOF packets and is configured to perform control transfers on control endpoint 0.

18.5.4.3

Initialization on SetAddress Command

This section describes what the application must do when it receives a SetAddress command in a SETUP packet.

1. Program the UDC_DCFG register with the device address received in the SetAddress command
2. Program the core to send out a status IN packet.

18.5.4.4

Initialization on SetConfiguration/SetInterface Command

This section describes what the application must do when it receives a SetConfiguration or SetInterface command in a SETUP packet.

1. When a SetConfiguration command is received, the application must program the endpoint registers to configure them with the characteristics of the valid endpoints in the new configuration.
2. When a SetInterface command is received, the application must program the endpoint registers of the endpoints affected by this command.
3. Some endpoints that were active in the prior configuration or alternate setting are not valid in the new configuration or alternate setting. These invalid endpoints must be deactivated.
4. For details on a particular endpoint's activation or deactivation, see “[Endpoint Activation](#)” on page [753](#) and “[Endpoint Deactivation](#)” on page [753](#).
5. Unmask the interrupt for each active endpoint and mask the interrupts for all inactive endpoints in the UDC_DAINTMSK register.
6. Set up the Data FIFO RAM for each FIFO (only if Dynamic FIFO Sizing is enabled). See “[Data FIFO RAM Allocation](#)” on page [786](#) for more detail.
7. After all required endpoints are configured, the application must program the core to send a status IN packet.

At this point, the device core is configured to receive and transmit any type of data packet.

18.5.4.5 Endpoint Activation

This section describes the steps required to activate a device endpoint or to configure an existing device endpoint to a new type.

1. Program the characteristics of the required endpoint into the following fields of the UDC_DIEPCTLn register (for IN or bidirectional endpoints) or the UDC_DOEPCTLn register (for OUT or bidirectional endpoints).
 - Maximum Packet Size
 - USB Active Endpoint = 1
 - Endpoint Start Data Toggle (for interrupt and bulk endpoints)
 - Endpoint Type
 - TxFIFO Number
2. Once the endpoint is activated, the core starts decoding the tokens addressed to that endpoint and sends out a valid handshake for each valid token received for the endpoint.

18.5.4.6 Endpoint Deactivation

This section describes the steps required to deactivate an existing endpoint.

1. In the endpoint to be deactivated, clear the USB Active Endpoint bit in the UDC_DIEPCTLn register (for IN or bidirectional endpoints) or the UDC_DOEPCTLn register (for OUT or bidirectional endpoints).
2. Once the endpoint is deactivated, the core ignores tokens addressed to that endpoint, resulting in a timeout on the USB.

18.5.4.7 Device DMA Mode Initialization

The application must meet the following condition to set up the device core to handle traffic: In DMA mode, UDC_GINTMSK.NPTxFEmpMsk, and UDC_GINTMSK.RxFLvlMsk must be masked.

18.5.5 Device Programming Model — Operational Model**18.5.5.1 SETUP and OUT Data Transfers**

This section describes the internal data flow and application-level operations during data OUT transfers and SETUP transactions.

SETUP Transactions

This section describes how the core handles SETUP packets and the application's sequence for handling SETUP transactions.

Application Requirements

1. To receive a SETUP packet, the UDC_DOEPTSIZn.SUPCnt field in a control OUT endpoint must be programmed to a non-zero value. When the application programs the SUPCnt field to a non-zero value, the core receives SETUP packets and writes them to the receive FIFO, irrespective of the UDC_DOEPCTLn.NAK status and UDC_DOEPCTLn.EPEna bit setting. The SUPCnt field is decremented every time the control endpoint receives a SETUP packet. If the SUPCnt field is not programmed to a proper value before receiving a SETUP packet, the core still receives the SETUP packet and decrements the SUPCnt field, but the application possibly is not be able to determine the correct number of SETUP packets received in the Setup stage of a control transfer.

- UDC_DOEPTSIZn.SUPCn = 3
- 2. In DMA mode, the OUT endpoint must also be enabled, to transfer the received SETUP packet data from the internal receive FIFO to the external memory.
 - UDC_DOEPCTLn.EPEna = 1'b1
- 3. The application must always allocate some extra space in the Receive Data FIFO, to be able to receive up to three SETUP packets on a control endpoint.
 - The space to be Reserved is $(4 * n) + 6$ DWORDs, where n is the number of control endpoints supported by the device. Three DWORDs are required for the first SETUP packet, 1 DWORD is required for the Setup Stage Done DWORD, and 6 DWORDs are required to store two extra SETUP packets among all control endpoints.
 - 3 DWORDs per SETUP packet are required to store 8 bytes of SETUP data and 4 bytes of SETUP status (Setup Packet Pattern). The core reserves this space in the receive data
 - FIFO to write SETUP data only, and never uses this space for data packets.

Internal Data Flow

1. When a SETUP packet is received, the core writes the received data to the receive FIFO, without checking for available space in the receive FIFO and irrespective of the endpoint's NAK and Stall bit settings.
 - The core internally sets the IN NAK and OUT NAK bits for the control IN/OUT endpoints on which the SETUP packet was received.
2. For every SETUP packet received on the USB, 3 DWORDs of data is written to the receive FIFO, and the SUPCn field is decremented by 1.
 - The first DWORD contains control information used internally by the core
 - The second DWORD contains the first 4 bytes of the SETUP command
 - The third DWORD contains the last 4 bytes of the SETUP command
3. When the Setup stage changes to a Data IN/OUT stage, the core writes an entry (Setup Stage Done DWORD) to the receive FIFO, indicating the completion of the Setup stage.
4. On the AHB side, SETUP packets are emptied by the DMA. In DMA mode, the SETUP packets (2 DWORDs) are written to the memory location programmed in the UDC_DOEPDMA register, only if the endpoint is enabled. If the endpoint is not enabled, the data remains in the receive FIFO until the enable bit is set.
5. When the DMA pops the Setup Stage Done DWORD from the receive FIFO, the core interrupts the application with a UDC_DOEPINTn.SETUP interrupt, indicating it can process the received SETUP packet.
 - The core clears the endpoint enable bit for control OUT endpoints.

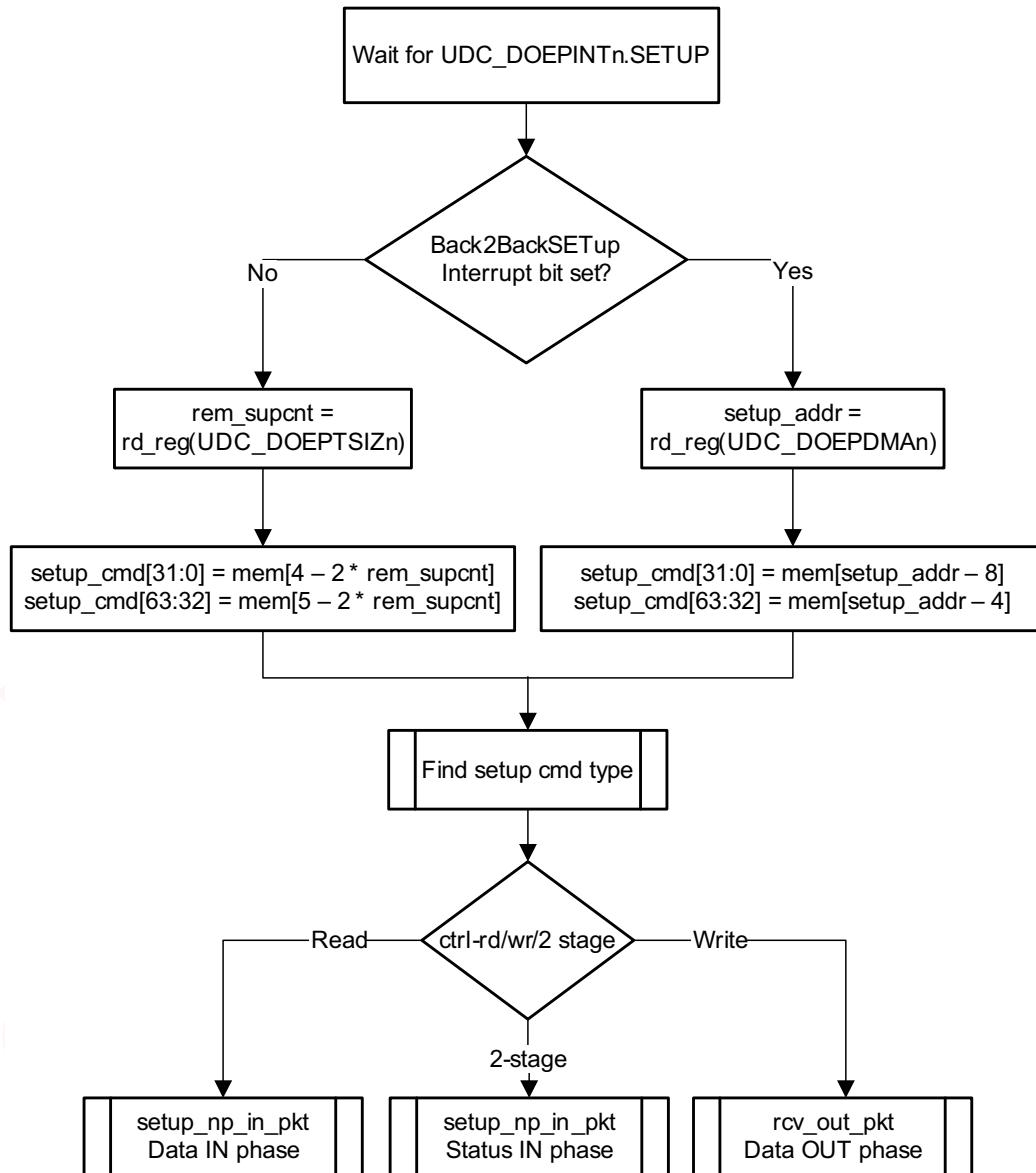
Application Programming Sequence

1. Program the UDC_DOEPTSIZn register.
 - UDC_DOEPTSIZn.SUPCn = 3
2. In DMA mode, program the UDC_DOEPDMA register and UDC_DOEPCTLn register with the endpoint characteristics and set the Endpoint Enable bit (UDC_DOEPCTLn.EPEna).
 - Endpoint Enable = 1
3. Assertion of the UDC_DOEPINTn.SETUP interrupt marks a successful completion of the SETUP Data Transfer.

- On this interrupt, the application must read the UDC_DOEPTSI n register to determine the number of SETUP packets received and process the last received SETUP packet.
- In DMA mode, the application must also determine if the interrupt bit UDC_DOEPINT n .Back2BackSETUp is set. This bit is set if the core has received more than three back-to-back SETUP packets. If this is the case, the application must ignore the UDC_DOEPTSI n .SUPCn value and use the UDC_DOEPDMA n directly to read out the last SETUP packet received. UDC_DOEPDMA n -8 provides the pointer to the last valid SETUP data.

Figure 18-7 charts this flow.

Figure 18-7. Processing a SETUP Packet



Handling More Than Three Back-to-Back SETUP Packets

Per the USB 2.0 specification, normally, during a SETUP packet error, a host does not send more than three back-to-back SETUP packets to the same endpoint. However, the USB 2.0 specification does not limit the number of back-to-back SETUP packets a host can send to the same endpoint. When this condition occurs, the USB device controller generates an interrupt (UDC_DOEPINTn.Back2BackSETUp). In DMA mode, the core also rewinds the DMA address for that endpoint (UDC_DOEPDMAAn) and overwrites the first SETUP packet in system memory with the fourth, second with the fifth, and so on. If the Back2BackSETUp interrupt is asserted, the application must read the OUT endpoint DMA register (UDC_DOEPDMAAn) to determine the final SETUP data in system memory.

In DMA mode, the application can mask the Back2BackSETUp interrupt, but after receiving the DOEPINT.SETUP interrupt, the application can read the DOEPINT.Back2BackSETUp interrupt bit.

Setting the Global OUT NAK

Internal Data Flow

1. When the application sets the Global OUT NAK (UDC_DCTL.SGOUTNak), the core stops writing data, except SETUP packets, to the receive FIFO. Irrespective of the space availability in the receive FIFO, non-isochronous OUT tokens receive a NAK handshake response, and the core ignores isochronous OUT data packets
2. The core writes the Global OUT NAK pattern to the receive FIFO. The application must reserve enough receive FIFO space to write this data pattern. See “[Data FIFO RAM Allocation](#)” on page [786](#).
3. When the core (in DMA mode) pops the Global OUT NAK pattern DWORD from the receive FIFO, the core sets the UDC_GINTSTS.GOUTNakEff interrupt.
4. Once the application detects this interrupt, it can assume that the core is in Global OUT NAK mode. The application can clear this interrupt by clearing the UDC_DCTL.SGOUTNak bit.

Application Programming Sequence

1. To stop receiving any kind of data in the receive FIFO, the application must set the Global OUT NAK bit by programming the following field.
 - UDC_DCTL.SGOUTNak = 1'b1
2. Wait for the assertion of the interrupt UDC_GINTSTS.GOUTNakEff. When asserted, this interrupt indicates that the core has stopped receiving any type of data except SETUP packets.
3. The application can receive valid OUT packets after it has set UDC_DCTL.SGOUTNak and before the core asserts the UDC_GINTSTS.GOUTNakEff interrupt.
4. The application can temporarily mask this interrupt by writing to the UDC_GINTMSK.GINNakEffMsk bit.
 - UDC_GINTMSK.GINNakEffMsk = 1'b0
5. Whenever the application is ready to exit the Global OUT NAK mode, it must clear the UDC_DCTL.SGOUTNak bit. This also clears the UDC_GINTSTS.GOUTNakEff interrupt.
 - UDC_DCTL.CGOUTNak = 1'b1
6. If the application has masked this interrupt earlier, it must be unmasked as follows:
 - UDC_GINTMSK.GINNakEffMsk = 1'b1

Disabling an OUT Endpoint

The application must use this sequence to disable an OUT endpoint that it has enabled.

Application Programming Sequence

1. Before disabling any OUT endpoint, the application must enable Global OUT NAK mode in the core, as described in “[Setting the Global OUT NAK](#)” on page [756](#).
 - UDC_DCTL.SGOUTNak = 1'b1
2. Wait for the UDC_GINTSTS.GOUTNakEff interrupt
3. Disable the required OUT endpoint by programming the following fields.
 - UDC_DOEPCTLn.EPDisable = 1'b1 UDC_DOEPCTLn.SNAK = 1'b1
4. Wait for the UDC_DOEPINTn.EPDisabled interrupt, which indicates that the OUT endpoint is completely disabled. When the EPDisabled interrupt is asserted, the core also clears the following bits.
 - UDC_DOEPCTLn.EPDisable = 1'b0 UDC_DOEPCTLn.EPEnable = 1'b0
5. The application must clear the Global OUT NAK bit to start receiving data from other non-disabled
 - OUT endpoints. UDC_DCTL.SGOUTNak = 1'b0

Generic Non-Isochronous OUT Data Transfers without Threshold Setting

This section describes a regular non-isochronous OUT data transfer (control, bulk, or interrupt).

Application Requirements

1. Before setting up an OUT transfer, the application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer, then program that buffer's size and start address (in DMA mode) in the endpoint-specific registers.
2. For OUT transfers, the Transfer Size field in the endpoint's Transfer Size register must be a multiple of the maximum packet size of the endpoint, adjusted to the DWORD boundary.
 - transfer size[epnum] = n * (mps[epnum] + 4 – (mps[epnum] mod 4))
 - packet count[epnum] = n
 - n > 0
3. In DMA mode, the core stores a received data packet in the memory, always starting on a DWORD boundary. If the maximum packet size of the endpoint is not a multiple of 4, the core inserts byte pads at end of a maximum-packet-size packet up to the end of the DWORD.
4. On any OUT endpoint interrupt, the application must read the endpoint's Transfer Size register to calculate the size of the payload in the memory. The received payload size can be less than the programmed transfer size.
 - Payload size in memory = application-programmed initial transfer size – core updated final transfer size
 - Number of USB packets in which this payload was received = application-programmed initial packet count – core updated final packet count

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers, clear the NAK bit, and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving data and writes it to the receive FIFO, as long as there is space in the receive FIFO. For every data packet received on the USB, the data packet and its status are written to the receive FIFO. Every packet (maximum packet size or short packet) written to the receive FIFO decrements the Packet Count field for that endpoint by 1.
 - OUT data packets received with Bad Data CRC are flushed from the receive FIFO automatically.
 - After sending an ACK for the packet on the USB, the core discards non-isochronous OUT data packets that the host, which cannot detect the ACK, re-sends. The application does not detect multiple back-to-back data OUT packets on the same endpoint with the same data PID. In this case the packet count is not decremented.
 - If there is no space in the receive FIFO, isochronous or non-isochronous data packets are ignored and not written to the receive FIFO. Additionally, non-isochronous OUT tokens receive a NAK handshake reply.
 - In all the above three cases, the packet count is not decremented because no data is written to the receive FIFO.
3. When the packet count becomes zero or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the isochronous or non-isochronous data packets are ignored and not written to the receive FIFO, and non-isochronous OUT tokens receive a NAK handshake reply.
4. After the data is written to the receive FIFO, either the application (in Slave mode) or the core's DMA engine (in External or Internal DMA mode), reads the data from the receive FIFO and writes it to external memory, one packet at a time per endpoint.
5. At the end of every packet write on the AHB to external memory, the transfer size for the endpoint is decremented by the size of the written packet.
6. The OUT Data Transfer Completed pattern for an OUT endpoint is written to the receive FIFO on one of the following conditions.
 - The transfer size is 0 and the packet count is 0
 - The last OUT data packet written to the receive FIFO is a short packet ($0 \leq$ packet size < maximum packet size)
7. When either the application or the DMA pops this entry (OUT Data Transfer Completed), a Transfer Completed interrupt is generated for the endpoint and the endpoint enable is cleared.

Application Programming Sequence

1. Program the UDC_DOEPTSIZn register for the transfer size and the corresponding packet count. Additionally, in DMA mode, program the UDC_DOEPDMA register.
2. Program the UDC_DOEPCTLn register with the endpoint characteristics, and set the Endpoint Enable and ClearNAK bits.
 - UDC_DOEPCTLn.EPEna = 1
 - UDC_DOEPCTLn.CNAK = 1
3. Asserting the UDC_DOEPINTn.XferCompl interrupt marks a successful completion of the non-isochronous OUT data transfer.
4. Read the UDC_DOEPTSIZn register to determine the size of the received data payload.

Generic non-Isochronous OUT Data Transfer with Threshold Setting

This section describes a regular non-ISO OUT data transfer (Control/Bulk/Intr) when threshold setting is enabled.

Application Requirements:

Application requirements is the same as without thresholding.

Internal Data Flow:

1. The application should set the transfer size and packet count fields in the endpoint specific registers, clear the NAK bit and enable the endpoint to receive the data.
2. Once the NAK bit is cleared, the core starts receiving the data and writes it into the receive FIFO, as long as there is threshold amount of space in the receive FIFO. For every threshold amount of data received on the USB, the data packet and the threshold status are written into the receive FIFO. At the end of a packet, a last threshold status and also a data update status is written into the receive FIFO. If it was the last packet of the transfer, then a transfer complete status is also written into the receive FIFO. On every packet (mps sized or short packet) written into the receive FIFO, the packet count field for that endpoint is decremented by 1.
 - OUT data packets received with Bad Data CRC are flushed out of the receive FIFO automatically. The core also rewind the DMA pointers internally.
 - Non-ISO OUT data packet re-sent by the host, because the ACK was not seen by the host, will be discarded by the core, after sending an ACK for the packet on the USB. The application will not see multiple back to back data OUT packets on the same endpoint, with the same data PID. In this case the packet count is not decremented.
 - If there is no space for at least threshold amount of data in the receive FIFO, the ISO/non-ISO data packets are ignored and not written into the receive FIFO. In addition, the non-ISO OUT tokens are responded with NAK handshake.
 - If the core sees an overflow case (no space in the fifo in the middle of a packet reception), then the core stops writing the remaining data into the fifo and sends a NAK handshake on the USB. The core rewinds the fifo pointer to the threshold boundary, so that the portion of the threshold data that is in the fifo is flushed out. The core also rewinds the DMA pointers. The core also sets UDC_DOEPINTn.OutPktErr (This interrupt bit is mainly used for debug purpose).

In all the above cases, the packet count is not decremented because no data is written into the receive FIFO.

In High Speed, after the core has received a packet, the core sends a NYET handshake if the core does not find threshold amount of free space available in the FIFO.

3. When the packet count becomes zero or when a short packet is received on the endpoint, the NAK bit for that endpoint is set. Once the NAK bit is set, the ISO/non-ISO data packets are ignored and not written into the receive FIFO and the non-ISO OUT tokens are responded with a NA
4. The DMA engine will transfer data from the receive FIFO to the system memory as soon as it sees one threshold amount of data in the FIFO.
5. At the end of every packet write on the AHB into the external memory, the transfer size for the endpoint is decremented by the size of packet written into the memory.
6. The OUT Data Transfer Complete pattern for an OUT endpoint is written into the receive FIFO, on one of the following conditions.
 - The last threshold is written into FIFO and the packet count is decremented to zero
 - If it is a short packet and the core sees the end of packet within a threshold.

7. When this entry (OUT Data Transfer Complete) is popped out by the DMA engine, Transfer Complete interrupt for the endpoint is generated and the endpoint enable is cleared.
8. “Rewind OUT Data Transfer” pattern is written into the receive FIFO, on one of the following conditions:
 - On seeing Overflow condition.
 - On seeing a CRC error.
9. When this entry (Rewind OUT Data Transfer) is popped out by the DMA engine, it does the DMA pointer rewind.

Application Programming Sequence

This sequence is the same as in non-thresholding case.

Generic Isochronous OUT Data Transfer without Threshold Setting

This section describes a regular isochronous OUT data transfer.

Application Requirements:

1. All the application requirements for non-isochronous OUT data transfers also apply to isochronous OUT data transfers
2. For isochronous OUT data transfers, the Transfer Size and Packet Count fields must always be set to the number of maximum-packet-size packets that can be received in a single microframe and no more. Isochronous OUT data transfers cannot span more than 1 microframe.
 - $1 \leq \text{packet count[epnum]} \leq 3$
3. In DMA mode, the application must guarantee enough bandwidth to allow emptying the isochronous OUT data packet from the receive FIFO before the end of each periodic frame.
4. To receive data in the following frame/microframe, an isochronous OUT endpoint must be enabled after the UDC_GINTSTS.EOPF and before the UDC_GINTSTS.SOF.

Internal Data Flow

1. The internal data flow for isochronous OUT endpoints is the same as that for non-isochronous OUT endpoints, but for a few differences.
2. When an isochronous OUT endpoint is enabled by setting the Endpoint Enable and clearing the NAK bits, the Even/Odd frame/microframe bit must also be set appropriately. The core receives data on a isochronous OUT endpoint in a particular microframe only if the following condition is met.
 - $\text{UDC_DOEPCTLn.Even/Odd microframe} = \text{UDC_DSTS.SOFTN}[0]$
3. When the internal DMA completely reads an isochronous OUT data packet (data and status) from the receive FIFO, the core updates the UDC_DOEPTSIzn.Received DPID field with the data PID of the last isochronous OUT data packet read from the receive FIFO.

Application Programming Sequence

1. Program the UDC_DOEPTSIzn register for the transfer size and the corresponding packet count. When in DMA mode, also program the UDC_DOEPDMA register.
2. Program the UDC_DOEPCTLn register with the endpoint characteristics and set the Endpoint Enable, ClearNAK, and Even/Odd frame/microframe bits.

- Endpoint Enable = 1
 - CNAK = 1
 - Even/Odd frame/microframe = (0: Even/1: Odd)
3. The assertion of the UDC_DOEPINTn.XferCompl interrupt marks the completion of the isochronous OUT data transfer. This interrupt does not necessarily mean that the data in memory is good.
 - This interrupt can not always be detected for isochronous OUT transfers. Instead, the application can detect the UDC_GINTSTS.incomplete Isochronous OUT data interrupt. See “[Incomplete Isochronous OUT Data Transfers](#)” on page [762](#), for more details.
 4. Read the UDC_DOEPTSIZn register to determine the size of the received transfer and to determine the validity of the data received in the microframe. The application must treat the data received in memory as valid only if one of the following conditions is met.
 - UDC_DOEPTSIZn.RxDPID = D0 and the number of USB packets in which this payload was received = 1
 - UDC_DOEPTSIZn.RxDPID = D1 and the number of USB packets in which this payload was received = 2
 - UDC_DOEPTSIZn.RxDPID = D2 and the number of USB packets in which this payload was received = 3
 - The number of USB packets in which this payload was received = App Programmed Initial Packet Count – Core Updated Final Packet Count

The application can discard invalid data packets.

Generic Isochronous OUT Data Transfer with Threshold Setting

This section describes a regular isochronous OUT data transfer.

Application Requirements:

There is no change in this section from a non-thresholding mode.

Internal Data Flow:

1. The internal data flow for isochronous OUT Endpoints when thresholding is enabled, is the same as that for the non isochronous OUT endpoints when thresholding is enabled, but for a few differences.
2. If MAC sees an overflow condition when writing a packet, it stops writing. The current threshold amount of data that is being written into the receive FIFO is flushed out at the end of packet. This will eventually result in UDC_GINTSTS.incomplete ISO OUT Data interrupt. Refer to the section “[Incomplete Isochronous OUT Data Transfers](#)” on page [762](#), for more details.
3. If MAC sees a CRC error for the receiving packet, the last threshold being written into the receive FIFO is flushed out. This will eventually result in UDC_GINTSTS.incomplete isochronous OUT Data interrupt. Refer to the section “[Incomplete Isochronous OUT Data Transfers](#)” on page [762](#), for more details.
4. Assertion of UDC_DOEPINTn.XferCompl interrupt marks a completion of the isochronous OUT Data Transfer. This interrupt may not necessarily mean that data in the memory is good data.
5. Read the UDC_DOEPTSIZn register, to find out the size of the received transfer and to find out the validity of the data received in the microframe. The application should treat the data received into the memory as valid only if one of the following conditions is met. Invalid data packets may be discarded by the application.

- UDC_DOEPTSIZn.RxDPID = D0 and Number of USB Packets in which this payload was received = 1
- UDC_DOEPTSIZn.RxDPID = D1 and Number of USB Packets in which this payload was received = 2
- UDC_DOEPTSIZn.RxDPID = D2 and Number of USB Packets in which this payload was received = 3

Number of USB Packets in which this payload was received = App Programmed Initial Packet Count - Core Updated Final Packet Count

Incomplete Isochronous OUT Data Transfers

This section describes the application programming sequence when isochronous OUT data packets are dropped inside the core.

Internal Data Flow

1. For isochronous OUT endpoints, the UDC_DOEPINTn.XferCompl interrupt possibly is not always asserted. If the core drops isochronous OUT data packets, the application could fail to detect the UDC_DOEPINTn.XferCompl interrupt under the following circumstances.
 - When the receive FIFO cannot accommodate the complete ISO OUT data packet, the core drops the received ISO OUT data. In thresholding this is same as overflow.
 - When the isochronous OUT data packet is received with CRC errors
 - When the isochronous OUT token received by the core is corrupted
 - When the application is very slow in reading the data from the receive FIFO
2. When the core detects an end of periodic frame before transfer completion to all isochronous OUT endpoints, it asserts the UDC_GINTSTS.incomplete Isochronous OUT data interrupt, indicating that a UDC_DOEPINTn.XferCompl interrupt is not asserted on at least one of the isochronous OUT endpoints. At this point, the endpoint with the incomplete transfer remains enabled, but no active transfers remains in progress on this endpoint on the USB.

Application Programming Sequence

1. Asserting the UDC_GINTSTS.incomplete Isochronous OUT data interrupt indicates that in the current microframe, at least one isochronous OUT endpoint has an incomplete transfer.
 - If this occurs because isochronous OUT data is not completely emptied from the endpoint, the application must ensure that the DMA or the application empties all isochronous OUT data (data and status) from the receive FIFO before proceeding.
 - When all data is emptied from the receive FIFO, the application can detect the UDC_DOEPINTn.XferCompl interrupt. In this case, the application must re-enable the endpoint to receive isochronous OUT data in the next microframe, as described in “[Generic Isochronous OUT Data Transfer without Threshold Setting](#)” on page [760](#).
2. When it receives a UDC_GINTSTS.incomplete Isochronous OUT data interrupt, the application must read the control registers of all isochronous OUT endpoints (UDC_DOEPCTLn) to determine which endpoints had an incomplete transfer in the current microframe. An endpoint transfer is incomplete if both the following conditions are met.
 - UDC_DOEPCTLn.Even/Odd microframe bit = UDC_DSTS.SOFFN[0]
 - UDC_DOEPCTLn.Endpoint Enable = 1

3. The previous step must be performed before the UDC_GINTSTS.SOF interrupt is detected, to ensure that the current microframe number is not changed.
4. For isochronous OUT endpoints with incomplete transfers, the application must discard the data in the memory and disable the endpoint by setting the UDC_DOEPCTLn.Endpoint Disable bit.
5. Wait for the UDC_DOEPINTn.Endpoint Disabled interrupt and enable the endpoint to receive new data in the next microframe as explained in "[Generic Isochronous OUT Data Transfer without Threshold Setting](#)" on page [760](#).
 - Because the core can take some time to disable the endpoint, the application possibly is not able to receive the data in the next microframe after receiving bad isochronous data.

Stalling a Non-Isochronous OUT Endpoint

This section describes how the application can stall a non-isochronous endpoint.

1. Put the core in the Global OUT NAK mode, as described in "[Setting the Global OUT NAK](#)" on page [756](#).
2. Disable the required endpoint, as described in "[Disabling an OUT Endpoint](#)" on page [757](#).
 - When disabling the endpoint, instead of setting the DOEPCTL.SNAK bit, set DOEPCTL.STALL = 1.
 - The Stall bit always takes precedence over the NAK bit.
3. When the application is ready to end the STALL handshake for the endpoint, the UDC_DOEPCTLn.STALL bit must be cleared.
4. If the application is setting or clearing a STALL for an endpoint due to a SetFeature.Endpoint Halt or ClearFeature.Endpoint Halt command, the Stall bit must be set or cleared before the application sets up the Status stage transfer on the control endpoint.

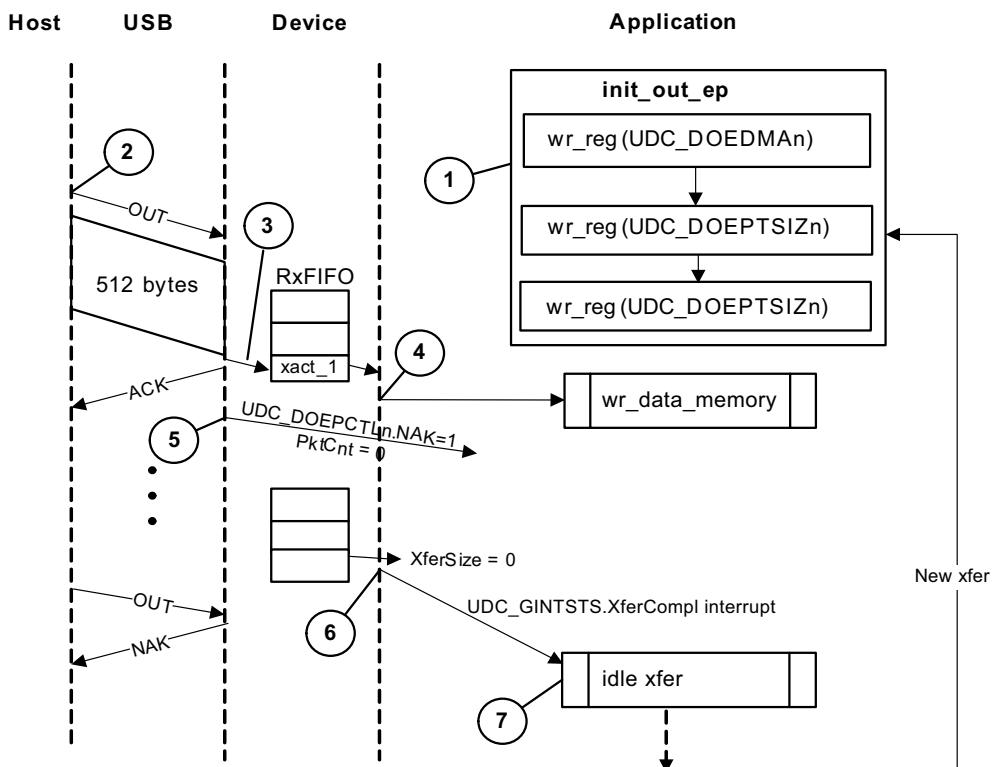
Examples

This section describes and depicts some fundamental transfer types and scenarios.

DMA Mode Bulk OUT Transaction

The following figure depicts the reception of a single Bulk OUT Data packet from the USB to the AHB and describes the events involved in the process.

Figure 18-8. DMA Mode Bulk OUT Transaction



1. After a SetConfiguration/SetInterface command, for Bulk OUT transfers application must allocate a buffer in the memory to accommodate all data to be received as part of the OUT transfer. Application must program the buffer's size (XferSize) and PktCnt in the UDC_DOEPTSIzn register and start address in the UDC_DOEPDMAAn register. Application initializes Bulk OUT endpoint by setting UDC_DOEPCTLn.CNAK = 1 and UDC_DOEPCTLn.EPEna = 1 bits.
2. Host attempts to send data (OUT token) to an endpoint.
3. When the core receives the OUT token on the USB, it stores the packet in the RxFIFO because space is available there.
4. After writing the complete packet in the RxFIFO, the core's DMA engine reads the data from the RxFIFO and writes it to the external memory starting at the address programmed in the UDC_DOEPDMAAn register.
5. On receiving the PktCnt number of USB packets, the core sets the NAK bit for this endpoint internally to prevent it from receiving any more packets.
6. The core then asserts the UDC_GINTSTS.XferCompl interrupt.
7. The application processes the setting of the UDC_DOEPINTn.XferCompl interrupt bit to determine that the intended transfer is complete.

18.5.5.2 IN Data Transfers

This section describes IN data transfer details.

Packet Write in Slave Mode: Dedicated Transmit FIFOs

This section describes how the application writes data packets to the endpoint FIFO in Slave mode when dedicated transmit FIFOs are enabled.

1. The application can either choose polling or interrupt mode.
 - In polling mode, application monitors the status of the endpoint transmit data FIFO, by reading the UDC_DTXFSTS_n register, to determine, if there is enough space in the data FIFO.
 - In interrupt mode, application waits for the UDC_DIEPINT_n.TxFEmp interrupt and then reads the UDC_DTXFSTS_n register, to determine, if there is enough space in the data FIFO.
 - To write a single non-zero length data packet, there must be space to write the entire packet in the data FIFO.
 - For writing zero length packet, application must not look for FIFO space.
2. Using one of the above mentioned methods, when the application determines that there is enough space to write a transmit packet, the application must first write into the endpoint control register, before writing the data into the data FIFO. The application, typically must do a read modify write on the UDC_DIEPCTL_n, to avoid modifying the contents of the register, except for setting the Endpoint Enable bit.

The application can write multiple packets for the same endpoint, into the transmit FIFO, if space is available. For periodic IN endpoints, application must write packets only for one microframe. It can write packets for the next periodic transaction, only after getting transfer complete for the previous transaction.

Setting IN Endpoint NAK

Internal Data Flow

1. When the application sets the IN NAK for a particular endpoint, the core stops transmitting data on the endpoint, irrespective of data availability in the endpoint's transmit FIFO.
 - Non-isochronous IN tokens receive a NAK handshake reply
 - Isochronous IN tokens receive a zero-data-length packet reply
2. The core asserts the UDC_DIEPINT_n.IN NAK Effective interrupt in response to the DIEPCTL.Set NAK bit.
3. Once this interrupt is seen by the application, the application can assume that the endpoint is in IN NAK mode. This interrupt can be cleared by the application by setting the UDC_DIEPCTL_n.Clear NAK bit.

Application Programming Sequence

1. To stop transmitting any data on a particular IN endpoint, the application must set the IN NAK bit. To set this bit, the following field must be programmed.
 - UDC_DIEPCTLn.SetNAK = 1'b1
2. Wait for assertion of the UDC_DIEPINTn.NAK Effective interrupt. This interrupt indicates the core has stopped transmitting data on the endpoint.
3. The core can transmit valid IN data on the endpoint after the application has set the NAK bit, but before the assertion of the NAK Effective interrupt.
4. The application can mask this interrupt temporarily by writing to the UDC_DIEPMSK.NAK Effective bit.
 - UDC_DIEPMSK.NAK Effective = 1'b0
5. To exit Endpoint NAK mode, the application must clear the UDC_DIEPCTLn.NAK status. This also clears the UDC_DIEPINTn.NAK Effective interrupt.
 - UDC_DIEPCTLn.ClearNAK = 1'b1
6. If the application masked this interrupt earlier, it must be unmasked as follows:
7. UDC_DIEPMSK.NAK Effective = 1'b1

IN Endpoint Disable in Dedicated FIFO Operation

Use the following sequence to disable a specific IN endpoint (periodic/non-periodic) that has been previously enabled in dedicated FIFO operation.

Application Programming Sequence:

1. The application must set the endpoint in NAK mode. Refer to the section Setting the Endpoint NAK
2. UDC_DIEPCTLn.SetNAK = 1'b1
3. Wait for UDC_DIEPINTn.NAK Effective interrupt.
4. Set the following bits in the UDC_DIEPCTLn register for the endpoint that must be disabled.
 - 5. UDC_DIEPCTLn.Endpoint Disable = 1
 - UDC_DIEPCTLn.SetNAK = 1
6. Assertion of UDC_DIEPINTn.Endpoint Disabled interrupt indicates that the core has completely disabled the specified endpoint. Along with the assertion of the interrupt, the core also clears the following bits.
 - UDC_DIEPCTLn.EPEnable = 1'b0
 - UDC_DIEPCTLn.EPDDisable = 1'b0
7. The application must read the UDC_DIEPTSIzN register for the periodic IN EP, to calculate how much data on the endpoint was transmitted on the USB.
8. The application must flush the data in the Endpoint transmit FIFO, by setting the following fields in the UDC_GRSTCTL register.
 - UDC_GRSTCTL.TxFIFONum = Endpoint Transmit FIFO Number
 - UDC_GRSTCTL.TxFFFlush = 1

The application must poll the UDC_GRSTCTL register, until the TxFFFlush bit is cleared by the core, which indicates the end of flush operation. To transmit new data on this endpoint, the application can re-enable the endpoint at a later point.

Generic Non-periodic IN Data Transfers without Threshold Setting

This section describes a regular non periodic IN data transfer when transmit thresholding is not enabled.

Application Requirements

1. Before setting up an IN transfer, the application must ensure that all data to be transmitted as part of the IN transfer is part of a single buffer, and must program the size of that buffer and its start address (in DMA mode) to the endpoint-specific registers.
2. For IN transfers, the Transfer Size field in the Endpoint Transfer Size register denotes a payload that constitutes multiple maximum-packet-size packets and a single short packet. This short packet is transmitted at the end of the transfer.
 - To transmit a few maximum-packet-size packets and a short packet at the end of the transfer:
 - Transfer size[epnum] = $n * \text{mps}[epnum] + sp$ (where n is an integer ≥ 0 , and $0 \leq sp < \text{mps}[epnum]$)
 - If ($sp > 0$), then packet count[epnum] = $n + 1$. Otherwise, packet count[epnum] = n
 - To transmit a single zero-length data packet:
 - Transfer size[epnum] = 0
 - Packet count[epnum] = 1
 - To transmit a few maximum-packet-size packets and a zero-length data packet at the end of the transfer, the application must split the transfer in two parts. The first sends maximum-pocket-size data packets and the second sends the zero-length data packet alone.
 - First transfer: transfer size[epnum] = $n * \text{mps}[epnum]$; packet count = n ;
 - Second transfer: transfer size[epnum] = 0; packet count = 1;
3. In DMA mode, the core fetches an IN data packet from the memory, always starting at a DWORD boundary. If the maximum packet size of the IN endpoint is not a multiple of 4, the application must arrange the data in the memory with pads inserted at the end of a maximum-packet-size packet so that a new packet always starts on a DWORD boundary.
4. Once an endpoint is enabled for data transfers, the core updates the Transfer Size register. At the end of IN transfer, which ended with a Timeout (only in Shared FIFO operation) or Endpoint Disabled interrupt, the application must read the Transfer Size register to determine how much data posted in the transmit FIFO was already sent on the USB.
 - Data fetched into transmit FIFO = Application-programmed initial transfer size – core-updated final transfer size
 - Data transmitted on USB = (application-programmed initial packet count – Core updated final packet count) * $\text{mps}[epnum]$
 - Data yet to be transmitted on USB = (Application-programmed initial transfer size – data transmitted on USB)

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. In DMA mode, the core fetches the data from memory according to the application setting for the endpoint.
3. Every time a packet is written into the transmit FIFO by the core's internal DMA (in DMA mode), the transfer size for that endpoint is decremented by the packet size. The data is fetched from the memory (DMA/Application) until the transfer size for the endpoint

becomes zero. In Dedicated Transmit FIFO operation, after writing the data into the FIFO, the “number of packets in FIFO” count is incremented (this is a 3-bit count, internally maintained by the core for each IN endpoint transmit FIFO. The maximum number of packets maintained by the core at any time in an IN endpoint FIFO is eight). For zero-length packets, a separate flag is set for each FIFO, without any data in the FIFO.

4. Once the data is written to the transmit FIFO, the core reads it out upon receiving an IN token. For every non-isochronous IN data packet transmitted with an ACK handshake, the packet count for the endpoint is decremented by one, until the packet count is zero. The packet count is not decremented on a TIMEOUT.
5. For zero length packets (indicated by zero length entry in the queue for Shared FIFO operation or by an internal zero length flag in Dedicated FIFO operation), the core sends out a zero-length packet for the IN token and decrements the Packet Count field.
6. If there is no data in the FIFO for a received IN token and the packet count field for that endpoint is zero, the core generates a IN Tkn Rcvd When FIFO Empty Interrupt for the endpoint, provided the endpoint NAK bit is not set. The core responds with a NAK handshake for non-isochronous endpoints on the USB.
7. If there is a TIMEOUT condition in Dedicated FIFO operation, the core internally rewinds the FIFO pointers and no timeout interrupt is generated.
8. When the transfer size is zero and the packet count is zero, the transfer complete interrupt for the endpoint is generated and the endpoint enable is cleared.

Application Programming Sequence

1. Program the UDC_DIEPTSI $_n$ register with the transfer size and corresponding packet count. In DMA mode, also program the UDC_DIEPDMA $_n$ register.
2. Program the UDC_DIEPCTL $_n$ register with the endpoint characteristics and set the CNAK and Endpoint Enable bits.
In DMA mode, ensure that the NextEp field is programmed so that the core fetches the data for IN endpoints in the correct order.
3. UDC_DIEPINT $_n$.TimeOut Interrupt is not generated in Dedicated FIFO operation and the core handles this internally.

Generic non-periodic IN Data Transfers with Threshold Setting

This section describes a regular non periodic IN data transfer when transmit threshold setting is enabled.

Application Requirements:

Application requirements are the same as those for DMA mode with no thresholding.

Internal Data Flow:

1. The application should set the transfer size and packet count fields in the Endpoint Specific registers, and enable the endpoint to transmit the data.
2. The core fetches threshold amount of data for one endpoint before switching to the next in a round robin fashion with equal fairness. The priority is given to periodic endpoints. non-periodic endpoint data is fetched only if data for the periodic endpoints has been fetched or if the currently active token on the USB is a non-periodic one. The core will not switch, and continue to fetch till the complete packet, if it finds that the currently active IN token on the USB is for that particular endpoint and the FIFO does not have the complete packet.

3. In response to an IN token on the USB, the core starts transmitting data, if the MAC finds at least threshold amount of data in the FIFO for that particular endpoint.
4. With each threshold amount of data written into the FIFO, the transfer size for that endpoint is decremented by the threshold size, except for the last packet. For the last packet, the transfer size is not decremented by the core. After writing the first threshold amount of data into the FIFO, the “number of packets in FIFO” count is incremented. For zero-length packets, a separate flag is set for that endpoint FIFO, without any data in the FIFO. This count is internally maintained by the core and is decremented when a full packet has been read out of the FIFO.
5. If the MAC sees an underrun case, where there is not enough data in the FIFO, the core will corrupt the data (invert the CRC) on the USB. The core will internally flush the FIFO, rewinding the DMA pointers and re-fetch the packet(s). Also UDC_DIEPINTn.Tx fifo Undr bit will be set as an indication to the application.
6. For every non-ISO data IN packet transmitted, with an ACK handshake, the packet count for the endpoint is decremented by 1, until the packet count becomes zero. The packet count is not decremented on a TIMEOUT or underrun condition.
7. If the zero length flag in the FIFO is set (internally set by the core, when the application enables and endpoint for zero length), then core sends out zero length packet for the IN token and decrements the packet count field.
8. If there is no data (or partial threshold data) in the FIFO for a received IN token, and the packet count field for that endpoint is 0, the core generates a IN Tkn Rcvd When FIFO Empty Interrupt for the endpoint. The core responds with a NAK handshake for the non-ISO endpoints on the USB.
9. If the core does not receive a handshake after sending the packet (TIMEOUT), the core internally flush the FIFO, rewind the DMA pointers and re-fetch the packet(s).
10. When the packet count is zero, the transfer complete interrupt for the endpoint is generated, and then the endpoint enable and transfer size are both cleared by the core.

Application Programming Sequence:

1. Program the UDC_DIEPTSI n register, for the transfer size and the corresponding packet count and the UDC_DIEPDMan register.
2. Program the UDC_DIEPCTL n register, with the Endpoint Characteristics and set the CNAK and the Endpoint Enable bit. Also specify the Tx FIFO number in the UDC_DIEPCTL n .TXFNum field.
3. Assertion of UDC_DIEPINT n .XferCompl interrupt marks the successful completion of the non-periodic IN transfer. Read to UDC_DIEPTSI n register should indicate, a transfer size = 0 and packet count = 0, indicating all the data is transmitted on the USB.

Generic Periodic IN Data Transfers without Threshold Setting

This section describes a typical Periodic IN data transfer when thresholding is not enabled.

Application Requirements

1. Application requirements 1, 2, 3, and 4 of “[Generic Non-periodic IN Data Transfers without Threshold Setting](#)” on page 767 also apply to periodic IN data transfers, except for a slight modification of Requirement 2.
 - The application can only transmit multiples of maximum-packet-size data packets or multiples of maximum-packet-size packets, plus a short packet at the end. To transmit a few maximum-packet-size packets and a short packet at the end of the transfer, the following conditions must be met.

- transfer size[epnum] = n * mps[epnum] + sp (where n is an integer ≤ 0 , and $0 \leq sp < mps[epnum]$)
 - If ($sp > 0$), packet count[epnum] = n + 1 Otherwise, packet count[epnum] = n;
 - mc[epnum] = packet count[epnum]
 - The application cannot transmit a zero-length data packet at the end of transfer. It can transmit a single zero-length data packet by itself. To transmit a single zero-length data packet,
 - transfer size[epnum] = 0
 - packet count[epnum] = 1
 - mc[epnum] = packet count[epnum]
2. The application can only schedule data transfers 1 microframe at a time.
- $(UDC_DIEPTSIzn.MC - 1) * UDC_DIEPCTLn.MPS \leq UDC_DIEPTSIzn.XferSiz \leq UDC_DIEPTSIzn.MC * UDC_DIEPCTLn.MPS$
 - $UDC_DIEPTSIzn.PktCnt = UDC_DIEPTSIzn.MC$
 - If $UDC_DIEPTSIzn.XferSiz < UDC_DIEPTSIzn.MC * UDC_DIEPCTLn.MPS$, the last data packet of the transfer is a short packet.
3. The application can schedule data transfers for multiple microframes, only if multiples of max packet sizes (up to 3 packets), should be transmitted every microframe. This is can be done, only when the core is operating in DMA mode. This is not a recommended mode of operation though.
- $((n * UDC_DIEPTSIzn.MC) - 1) * UDC_DIEPCTLn.MPS \leq UDC_DIEPTSIzn.TransferSize \leq n * UDC_DIEPTSIzn.MC * UDC_DIEPCTLn.MPS$
 $UDC_DIEPTSIzn.PacketCount = n * UDC_DIEPTSIzn.MC$
 - n is the number of microframes for which the data transfers are scheduled
- Data Transmitted per microframe in this case would be $UDC_DIEPTSIzn.MC * UDC_DIEPCTLn.MPS$, in all the microframes except the last one. In the microframe "n", the data transmitted would be $(UDC_DIEPTSIzn.TransferSize - (n-1) * UDC_DIEPTSIzn.MC * UDC_DIEPCTLn.MPS)$
4. For Periodic IN endpoints, the data must always be prefetched one microframe ahead for transmission in the next microframe. This can be done, by enabling the Periodic IN endpoint 1 (micro)frame ahead of the (micro)frame in which the data transfer is scheduled.
5. The complete data to be transmitted in the (micro)frame must be written into the transmit FIFO (by the DMA), before the Periodic IN token is received. Even when 1 DWORD of the data to be transmitted per microframe is missing in the transmit FIFO when the Periodic IN token is received, the core behaves as when the FIFO was empty. When the transmit FIFO is empty,
- A zero data length packet would be transmitted on the USB for ISO IN endpoints
 - A NAK handshake would be transmitted on the USB for INTR IN endpoints
6. In dedicated FIFO mode for a High Bandwidth IN endpoint with three packets in a microframe, the application can program the endpoint FIFO size to be $2 * \text{max_pkt_size}$ and have the third packet load in after the first packet has been transmitted on the USB.

Internal Data Flow

1. The application must set the Transfer Size and Packet Count fields in the endpoint-specific registers and enable the endpoint to transmit the data.
2. In DMA mode, the core fetches the data for the endpoint from memory, according to the application setting.

3. Every time the core's internal DMA (in DMA mode) writes a packet to the transmit FIFO, the transfer size for that endpoint is decremented by the packet size. The data is fetched from DMA memory until the transfer size for the endpoint becomes zero.
4. When an IN token is received for an periodic endpoint, the core transmits the data in the FIFO, if available. If the complete data payload (complete packet, in dedicated FIFO mode) for the microframe is not present in the FIFO, then the core generates an IN Tkn Rcvd When TxF Empty Interrupt for the endpoint.
 - A zero-length data packet is transmitted on the USB for isochronous IN endpoints
 - A NAK handshake is transmitted on the USB for interrupt IN endpoints
5. The packet count for the endpoint is decremented by 1 under the following conditions:
 - For isochronous endpoints, when a zero- or non-zero-length data packet is transmitted
 - For interrupt endpoints, when an ACK handshake is transmitted
6. When the transfer size and packet count are both zero, the Transfer Completed interrupt for the endpoint is generated and the endpoint enable is cleared.
7. At the “Periodic frame Interval” (controlled by UDC_DCFG.PerFrint), when the core finds non-empty any of the isochronous IN endpoint FIFOs scheduled for the current (micro)frame non-empty, the core generates a UDC_GINTSTS.incompISOIN interrupt.

Application Programming Sequence (Transfer Per Microframe)

1. Program the UDC_DIEPTSI n register. In DMA mode, also program the UDC_DIEPDMA n register.
2. Program the UDC_DIEPCTL n register with the endpoint characteristics and set the CNAK and Endpoint Enable bits.
3. Asserting the UDC_DIEPINT n .In Token Rcvd When TxF Empty interrupt indicates that the DMA has not yet written all data to be transmitted to the transmit FIFO.
 - If the interrupt endpoint is already enabled when this interrupt is detected, ignore the interrupt. If it is not enabled, enable the endpoint so that the data can be transmitted on the next IN token attempt.
 - If the isochronous endpoint is already enabled when this interrupt is detected, see “Incomplete Isochronous IN Data Transfers” on page 773 for more details.
4. The core handles time-outs internally on interrupt IN endpoints programmed as periodic endpoints, or when Dedicated FIFO operation is used, without application intervention. The application, thus, never detects a UDC_DIEPINT n .TimeOUT interrupt for periodic interrupt IN endpoints.
5. Asserting the UDC_DIEPINT n .XferCompl interrupt with no UDC_DIEPINT n .In Tkn Rcvd When TxF Empty interrupt indicates the successful completion of an isochronous IN transfer. A read to the UDC_DIEPTSI n register must indicate transfer size = 0 and packet count = 0, indicating all data is transmitted on the USB.
6. Asserting the UDC_DIEPINT n .XferCompl interrupt, with or without the UDC_DIEPINT n .In Tkn Rcvd When TxF Empty interrupt, indicates the successful completion of an interrupt IN transfer. A read to the UDC_DIEPTSI n register must indicate transfer size = 0 and packet count = 0, indicating all data is transmitted on the USB.
7. Asserting the UDC_GINTSTS.incomplete Isochronous IN Transfer interrupt with none of the aforementioned interrupts indicates the core did not receive at least 1 periodic IN token in the current microframe.
 - For isochronous IN endpoints, see “Incomplete Isochronous IN Data Transfers” on page 773, for more details.

Generic Periodic IN Data Transfers with Threshold Setting

This section describes a typical Periodic IN data transfer when thresholding is enabled.

Application Requirements:

Application requirements are the same as that for DMA mode with no thresholding.

Internal Data Flow:

1. The application should set the transfer size and packet count fields in the Endpoint Specific registers, and enable the endpoint to transmit the data.
2. The core fetches threshold amount of data for one endpoint before switching to the next in a round robin fashion with equal fairness. The priority is given to periodic endpoints. The core will not switch, and continue to fetch till the complete packet, if it finds that the currently active IN token on the USB side is for that particular endpoint and the FIFO does not have the complete packet.
3. In response to an IN token on the USB, the core starts transmitting, if the MAC finds at least threshold amount of data in the FIFO for that particular endpoint.
4. After a full packet has been written into the FIFO, the transfer size for that endpoint is decremented by the packet size (or less). After writing the first threshold amount of data into the FIFO, the “number of packets in FIFO” count is incremented. For zero length packets, a separate flag will be set in the FIFO, without any data in the FIFO. This count is internally maintained by the core and is decremented when a full packet has been read out of the FIFO.
5. If the MAC sees an underrun case, where there is not enough data in the FIFO, the core will corrupt the data (invert the CRC) on the USB. For isochronous endpoints, Underrun interrupt (UDC_DIEPINTn.TxFifoUndrn) is generated by the core for this endpoint. For interrupt endpoints, the core will flush the FIFO, rewind the DMA pointers and re-fetch the data.
6. If the core sees a timeout or an underrun condition for an interrupt endpoint, the core flushes the fifo, rewinds the DMA pointers and re-fetches the packet. No interrupt is generated to the application.
7. When an IN token is received for an periodic endpoint, the core transmits the data in the FIFO, if the core sees at least threshold amount of data. If the threshold amount of data for the packet is not present in the FIFO, the core generates a IN Tkn Rcvd When TxF Empty Interrupt for the endpoint.
 - A zero data length packet would be transmitted on the USB for ISO IN endpoints
 - A NAK handshake would be transmitted on the USB for INTR IN endpoints
8. The packet count for the endpoint is decremented by 1, on the following conditions
 - When a zero or non zero data length for ISO endpoints When an ACK handshake is transmitted for INTR endpoints
9. The packet count is not decremented when the core has to corrupt a packet because of underrun condition.
10. When the transfer size is 0 and the packet count is 0, the transfer complete interrupt for the endpoint is generated and the endpoint enable is cleared.

Application Programming Sequence (Transfer Per Microframe)

1. Program the UDC_DIEPTSI n register and in addition, in DMA mode program the UDC_DIEPDMA n register.
2. Program the UDC_DIEPCTL n register, with the Endpoint Characteristics and set the CNAK bit and the Endpoint Enable bit. Also specify the Tx FIFO number in the UDC_DIEPCTL n .TXFNum field.
3. Assertion of UDC_DIEPINT n .In Token Rcvd When TxF Empty interrupt indicates that the complete data to be transmitted is not written into the transmit FIFO.
 - If the INTR endpoint is already enabled, when this interrupt is seen, ignore the interrupt. If the IN Endpoint in Dedicated Tx FIFO config is not enabled, enable the endpoint, so that the data could be transmitted on the next attempt of the IN token.
 - If the ISO endpoint is already enabled, when this interrupt is seen, refer to the section Incomplete ISO IN Data Transfers, for more details.
4. TimeOUT on Interrupt IN endpoints, is handled by the core internally, without any application intervention. The application never sees any interrupt for timeout.
5. Assertion of UDC_DIEPINT n .XferCompl interrupt without any UDC_DIEPINT n .In Tkn Rcvd when TxF Empty interrupt, marks the successful completion of the ISO IN transfer. Read to UDC_DIEPTSI n register should indicate, a transfer size = 0 and packet count = 0, indicating all the data is transmitted on the USB.
6. Assertion of UDC_DIEPINT n .XferCompl interrupt with/without any UDC_DIEPINT n .In Tkn Rcvd when TxF Empty interrupt, marks the successful completion of the INTR IN transfer. Read to UDC_DIEPTSI n register should indicate, a transfer size = 0 and packet count = 0, indicating all the data is transmitted on the USB.
7. Assertion of UDC_GINTSTS.incomplete ISO IN Transfer interrupt with out any of the previously mentioned interrupts indicates that at least 1 periodic IN token was not received by the core, in the current microframe.
 - For ISO IN endpoints, refer to the section Incomplete ISO IN Data Transfers, for more details.
8. Assertion of UDC_DIEPINT n .TxFifoUndrn interrupt indicate that there was an underrun condition for the isochronous transaction in the current microframe. The application may choose ignore this interrupt, as this will eventually result in UDC_GINTSTS.Incomplete isochronous IN interrupt at the end of periodic frame. The application can also choose to service this interrupt. If they choose to do so, then they can save some time in re-enabling the endpoint for the next microframe. In response to this interrupt,
 - Disable the endpoint (Check section Disabling IN Endpoint in Dedicated Tx FIFO config).
 - Application reads Endpoint DIEPSI n register to see how much data is transferred.
 - Re-enable the endpoint for the next microframe.

Incomplete Isochronous IN Data Transfers

This section describes what the application must do on an incomplete isochronous IN data transfer.

Internal Data Flow

1. An isochronous IN transfer is treated as incomplete in one of the following conditions.
 - The core receives a corrupted isochronous IN token on at least one isochronous IN endpoint. In this case, the application detects a UDC_GINTSTS.incomplete Isochronous IN Transfer interrupt.

- The DMA is slow to write the complete data payload to the transmit FIFO and an IN token is received before the complete data payload is written to the FIFO. In this case, the application detects a UDC_DIEPINTn.IN Tkn Rcvd When TxFIFO Empty interrupt. The application can ignore this interrupt, as it eventually results in a UDC_GINTSTS.incomplete Isochronous IN Transfer interrupt at the end of periodic frame.
 - The core transmits a zero-length data packet on the USB in response to the received IN token.
 - If threshold setting is enabled and there was an underrun condition, the core also generates a DINEPINTn.TxfifoUndrn interrupt. The application can ignore this interrupt, which eventually results in a UDC_GINTSTS.incomplete ISO IN Transfer interrupt.
2. The application must set the NAK bit and the disable bit for the endpoint. In DMA mode, the core automatically stops fetching the data payload when the endpoint disable bit is set.
 3. The core disables the endpoint, clears the disable bit, and asserts the Endpoint Disable interrupt for the endpoint.

Application Programming Sequence

1. The application can ignore the UDC_DIEPINTn.IN Tkn Rcvd When TxFIFO empty interrupt on any isochronous IN endpoint, as it eventually results in a UDC_GINTSTS.incomplete Isochronous IN Transfer interrupt. The application can also ignore UDC_DIEPINTn.TxfifoUndrn interrupt when threshold setting is enabled.
2. Assertion of the UDC_GINTSTS.incomplete Isochronous IN Transfer interrupt indicates an incomplete isochronous IN transfer on at least one of the isochronous IN endpoints.
3. The application must read the Endpoint Control register for all isochronous IN endpoints to detect endpoints with incomplete IN data transfers.
4. In both modes of operation, program the following fields in the UDC_DIEPCTLn register to disable the endpoint. See “Disabling a Non-periodic IN Endpoint: Shared FIFO” on page 360 for more details.
 - UDC_DIEPCTLn.SetNAK = 1
 - UDC_DIEPCTLn.Endpoint Disable = 1
5. The UDC_DIEPINTn.Endpoint Disabled interrupt’s assertion indicates that the core has disabled the endpoint.
 - At this point, the application must flush the data in the associated transmit FIFO or overwrite the existing data in the FIFO by enabling the endpoint for a new transfer in the next microframe. To flush the data, the application must use the UDC_GRSTCTL register.

Stalling Non-Isochronous IN Endpoints

This section describes how the application can stall a non-isochronous endpoint.

Application Programming Sequence

1. Disable the IN endpoint to be stalled and set the Stall bit.
 - UDC_DIEPCTLn.Endpoint Disable = 1, when the endpoint is already enabled
 - UDC_DIEPCTLn.STALL = 1
 - The Stall bit always takes precedence over the NAK bit
2. Assertion of the UDC_DIEPINTn.Endpoint Disabled interrupt indicates to the application that the core has disabled the specified endpoint.

3. The application must flush the Non-periodic or Periodic Transmit FIFO, depending on the endpoint type. In case of a non-periodic endpoint, the application must re-enable the other non-periodic endpoints, which do not need to be stalled, to transmit data.
4. Whenever the application is ready to end the STALL handshake for the endpoint, the UDC_DIEPCTLn.STALL bit must be cleared.
5. If the application sets or clears a STALL for an endpoint due to a SetFeature.Endpoint Halt command or ClearFeature.Endpoint Halt command, the Stall bit must be set or cleared before the application sets up the Status stage transfer on the control endpoint.

Special Case: Stalling the Control OUT Endpoint

The core must stall IN/OUT tokens if, during the Data stage of a control transfer, the host sends more IN/OUT tokens than are specified in the SETUP packet. In this case, the application must enable UDC_DIEPINTn.INTknTXFEmp and UDC_DOEPINTn.OUTTknEPdis interrupts during the Data stage of the control transfer, after the core has transferred the amount of data specified in the SETUP packet. Then, when the application receives this interrupt, it must set the STALL bit in the corresponding endpoint control register, and clear this interrupt.

Examples

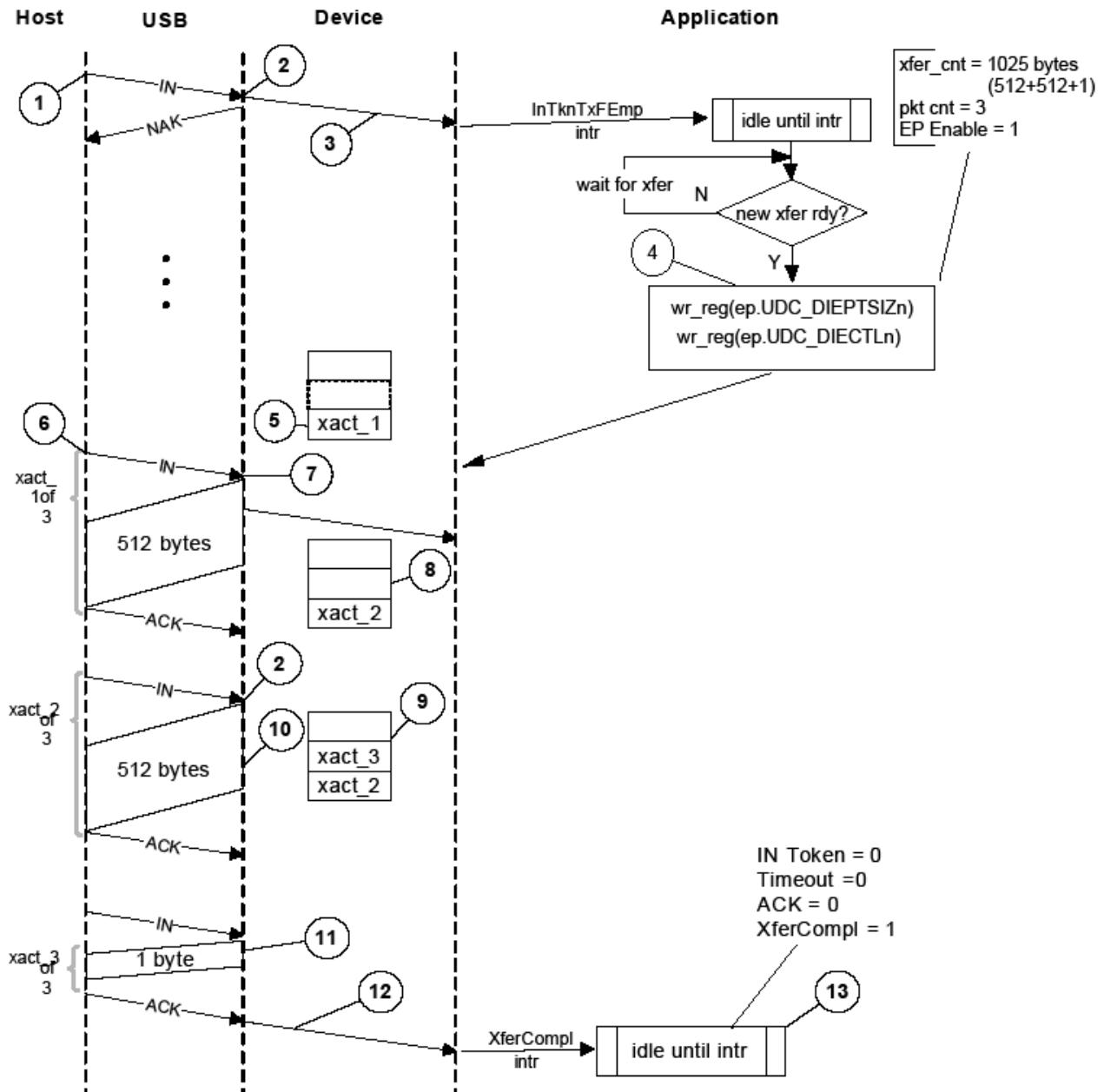
See the following sections for examples:

- See “DMA Mode Bulk IN Transfer” on page 775.“
- See “Bulk IN Stall” on page 777.
- See “Bulk IN DMA mode with Thresholding” on page 779.

DMA Mode Bulk IN Transfer

These notes refer to [Figure 18-9](#).

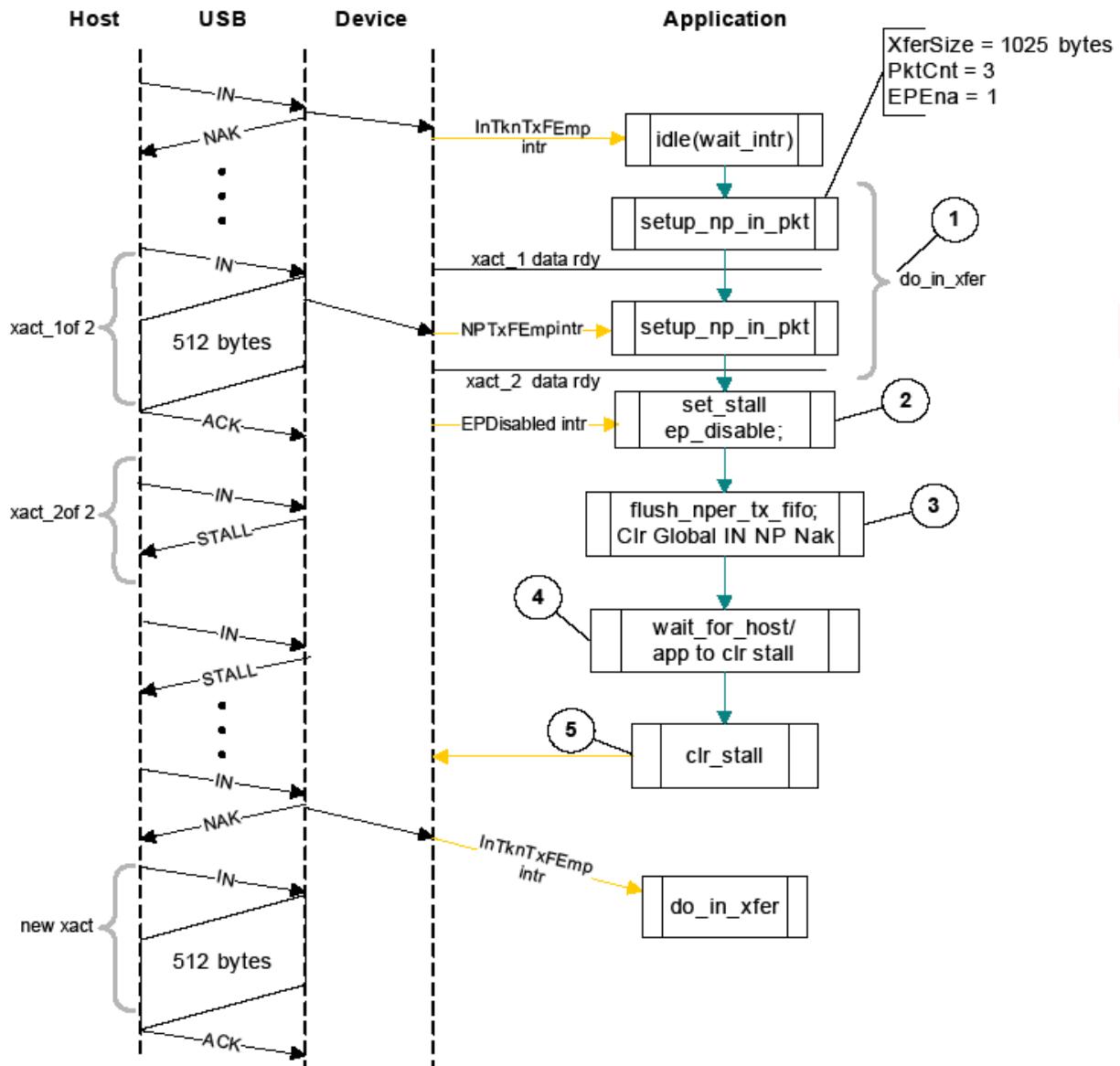
1. The host attempts to read data (IN token) from an endpoint.
2. On receiving the IN token on the USB, the core returns a NAK handshake, because no data is available in the transmit FIFO.
3. To indicate to the application that there was no data to send, the core generates a UDC_DIEPINTn.IN Token Rcvd When TxFIFO Empty interrupt.
4. When data is ready, the application sets up the UDC_DIEPTSIZn register with the TransferSize and Packet Count fields and enables the endpoint.
5. On seeing the endpoint enabled, the internal DMA engine starts fetching one maximum packet size or less of data to the TxFIFO.bb
6. The host reattempts the IN token.
7. Because data is now ready in the TxFIFO, the core now responds with the data and the host ACKs it.
8. The internal DMA engine now starts fetching the second maximum packet size after it sees maximum packet size amount of space.
9. The internal DMA engine fetches the third packet of 1 byte size after it sees the space available. The data for the second packet is being sent on the bus.
10. The second data packet is sent to the host.
11. The last short packet is sent to the host.
12. Because the last packet is sent and XferSize is now zero, the intended transfer is complete. The device core generates a UDC_DIEPINTn.XferCompl interrupt.
13. The application processes the interrupt and uses the setting of the UDC_DIEPINTn.XferCompl interrupt bit to determine that the intended transfer is complete.

Figure 18-9. DMA Mode Bulk IN Transfer

Bulk IN Stall

These notes refer to [Figure 18-10](#).

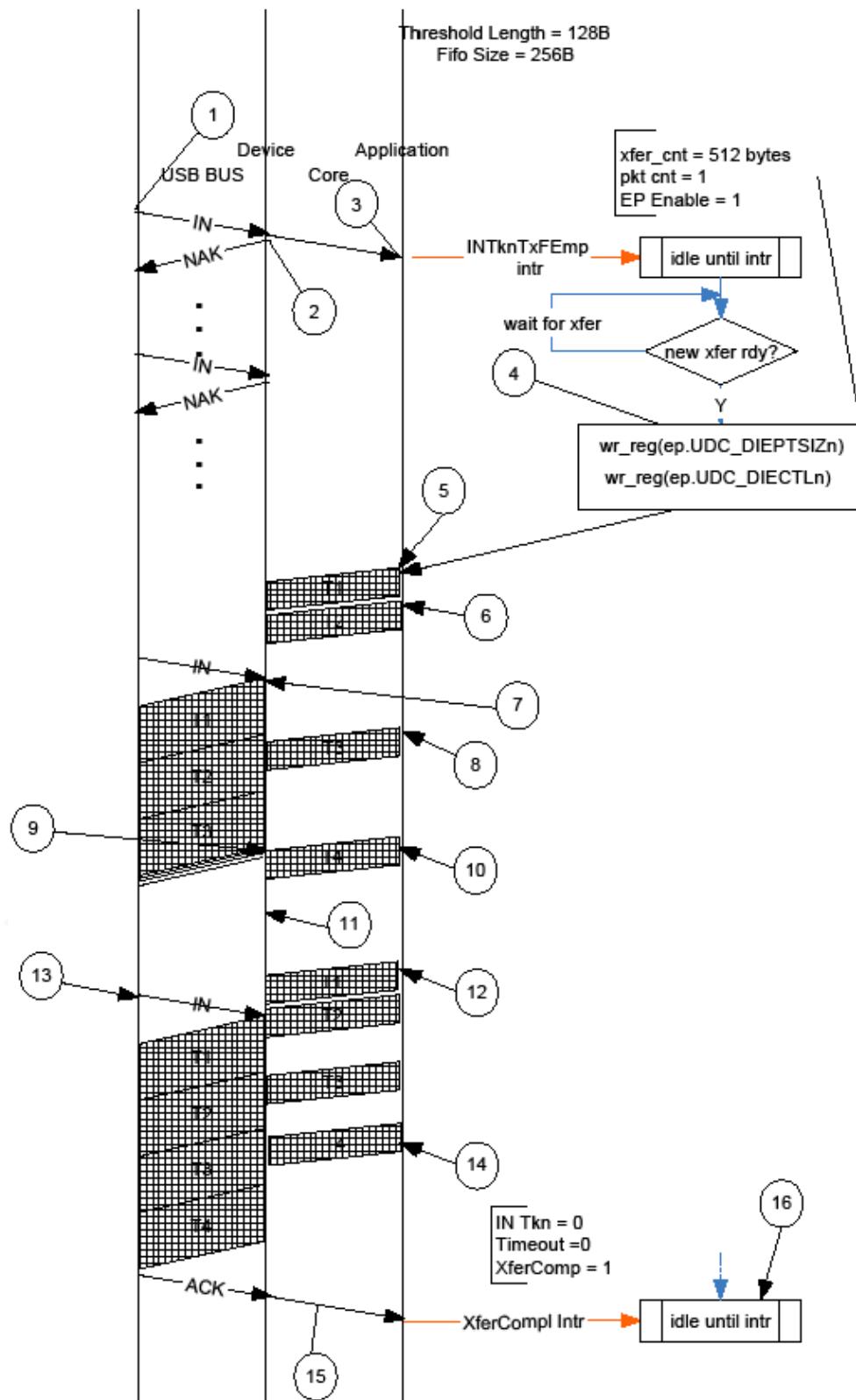
1. The application has scheduled an IN transfer on receiving the UDC_DIEPINTn.InTknRcvd When TxFIFO Empty interrupt.
2. When the transfer is in progress, the application must force a STALL on the endpoint. This could be because the application has received a SetFeature.Endpoint Halt command. The application sets the Stall bit, disables the endpoint and waits for the UDC_DIEPINTn.Endpoint Disabled interrupt. This generates STALL handshakes for the endpoint on the USB.
3. On receiving the interrupt, the application flushes the Non-periodic Transmit FIFO and clears the UDC_DCTL.GlobalINNPNAK bit.
4. On receiving the ClearFeature.Endpoint Halt command, the application clears the Stall bit.
5. The endpoint behaves normally and the application can re-enable the endpoint for new transfers.

Figure 18-10. Bulk IN Stall

Bulk IN DMA mode with Thresholding

These notes refer to [Figure 18-11](#).

1. The host attempts to read data(IN token) from an endpoint
2. On receiving the IN token on the USB bus, the core sends back a NAK handshake because no data is available in the Transmit FIFO
3. To indicate to the application that there was no data to send, the core generates a UDC_DIEPINTn.IN Token Rcvd When TxF Empty interrupt.
4. When data is ready, the application sets up the UDC_DIEPTSIz register with the TransferSize and Packet Count fields and enables the endpoint.
5. On seeing the endpoint enabled, the internal DMA engine start fetching the first threshold amount of data.
6. DMA engine fetching the second threshold.
7. The host re-attempts the IN token, and core start sending the data because it sees at least threshold amount of data in the FIFO.
8. The DMA engine starts fetching the third threshold after it sees threshold amount of space.
9. The MAC sees an underrun condition because of FIFO being empty in the middle of the packet, stops the packet and corrupt the CRC.
10. DMA engine starts to fetch the last threshold for that packet but it is late and will eventually result in underrun.
11. No handshake response from the host, because the packet was corrupted. Core rewinds the pointers and flush the FIFO.
12. The core starts to re-fetch the packet, staring with the first threshold.
13. The host re-attempts the IN token, and the core starts sending the data, since the core detects at least the threshold amount of data in the FIFO.
14. The core fetches the last threshold in time.
15. The core receives the handshake from host and Because the XferSize is now zero, the intended transfer is complete. Device core generates a UDC_DIEPINTn.XferCompl interrupt.
16. The application processes the interrupt and uses the setting of UDC_DIEPINTn.XferCompl interrupt bit to determine that the intended transfer is complete.

Figure 18-11. Bulk IN DMA mode with Thresholding

18.5.5.3 Control Transfers

This section describes the various types of control transfers.

Control Write Transfers (SETUP, Data OUT, Status IN)

This section describes control write transfers.

Application Programming Sequence

1. Assertion of the UDC_DOEPINTn.SETUP Packet interrupt indicates that a valid SETUP packet has been transferred to the application. See “[SETUP Transactions](#)” on page [753](#) for more details. At the end of the Setup stage, the application must reprogram the UDC_DOEPTSIZn.SUPCn field to 3 to receive the next SETUP packet.
2. If the last SETUP packet received before the assertion of the SETUP interrupt indicates a data OUT phase, program the core to perform a control OUT transfer as explained in “[Generic Non-Isochronous OUT Data Transfers without Threshold Setting](#)” on page [757](#).
3. In DMA mode, the application must reprogram the UDC_DOEPDMAAn register to receive a control OUT data packet to a different memory location.
4. In a single OUT data transfer on control endpoint 0, the application can receive up to 64 bytes. If the application is expecting more than 64 bytes in the Data OUT stage, the application must re-enable the endpoint to receive another 64 bytes, and must continue to do so until it has received all the data in the Data stage.
5. Assertion of the UDC_DOEPINTn.Transfer Compl interrupt on the last data OUT transfer indicates the completion of the data OUT phase of the control transfer.
6. On completion of the data OUT phase, the application must do the following.
 - To transfer a new SETUP packet in DMA mode, the application must re-enable the control OUT endpoint as explained in section in section “[SETUP Transactions](#)” on page [753](#).
 - UDC_DOEPCTLn.EPEna = 1'b1
 - To execute the received Setup command, the application must program the required registers in the core. This step is optional, based on the type of Setup command received.
7. For the status IN phase, the application must program the core as described in “[Generic Non-periodic IN Data Transfers without Threshold Setting](#)” on page [767](#) to perform a data IN transfer.
8. Assertion of the UDC_DIEPINTn.Transfer Compl interrupt indicates completion of the status IN phase of the control transfer.
9. The previous step must be repeated until the UDC_DIEPINTn.Transfer Compl interrupt is detected on the endpoint, marking the completion of the control write transfer.

Control Read Transfers (SETUP, Data IN, Status OUT)

This section describes control write transfers.

Application Programming Sequence

1. Assertion of the UDC_DOEPINTn.SETUP Packet interrupt indicates that a valid SETUP packet has been transferred to the application. See “[SETUP Transactions](#)” on page [753](#) for more details. At the end of the Setup stage, the application must reprogram the UDC_DOEPTSIzn.SUPCn field to 3 to receive the next SETUP packet.
2. If the last SETUP packet received before the assertion of the SETUP interrupt indicates a data IN phase, program the core to perform a control IN transfer as explained in “[Generic Non-periodic IN Data Transfers without Threshold Setting](#)” on page [767](#).
3. On a single IN data transfer on control endpoint 0, the application can transmit up to 64 bytes. To transmit more than 64 bytes in the Data IN stage, the application must re-enable the endpoint to transmit another 64 bytes, and must continue to do so, until it has transmitted all the data in the Data stage.
4. The previous step must be repeated until the UDC_DIEPINTn.Transfer Compl interrupt is detected for every IN transfer on the endpoint.
5. The UDC_DIEPINTn.Transfer Compl interrupt on the last IN data transfer marks the completion of the control transfer’s Data stage.
6. To perform a data OUT transfer in the status OUT phase, the application must program the core as described in “[SETUP and OUT Data Transfers](#)” on page [753](#).
 - The application must program the UDC_DCFG.NZStsOUTHShk handshake field to a proper setting before transmitting an data OUT transfer for the Status stage.
 - In DMA mode, the application must reprogram the DOEPDMDAn register to receive the control OUT data packet to a different memory location.
7. Assertion of the UDC_DOEPINTn.Transfer Compl interrupt indicates completion of the status OUT phase of the control transfer. This marks the successful completion of the control read transfer.
 - To transfer a new SETUP packet in DMA mode, the application must re-enable the control OUT endpoint as explained in “[SETUP Transactions](#)” on page [753](#).
 - UDC_DOEPCTLn.EPEna = 1'b1

Two-Stage Control Transfers (SETUP/Status IN)

This section describes two-stage control transfers.

Application Programming Sequence

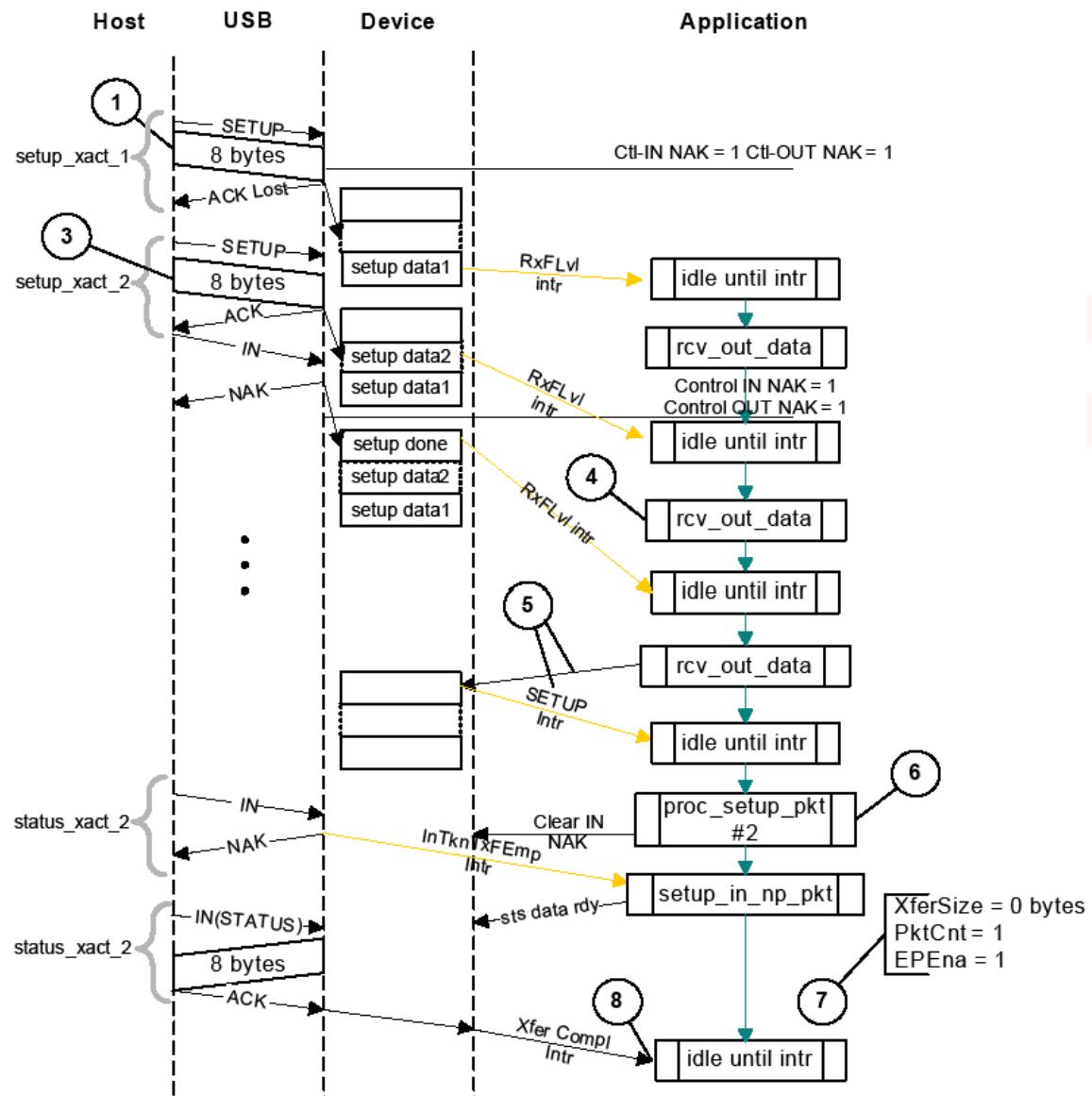
1. Assertion of the UDC_DOEPINTn.SetUp interrupt indicates that a valid SETUP packet has been transferred to the application. See “[SETUP Transactions](#)” on page [753](#) for more detail. To receive the next SETUP packet, the application must reprogram the UDC_DOEPTSIzn.SUPCn field to 3 at the end of the Setup stage.
2. Decode the last SETUP packet received before the assertion of the SETUP interrupt. If the packet indicates a two-stage control command, the application must do the following.
 - To transfer a new SETUP packet in DMA mode, the application must re-enable the control OUT endpoint. See “[SETUP Transactions](#)” on page [753](#) for details.
 - UDC_DOEPCTLn.EPEna = 1'b1
 - Depending on the type of Setup command received, the application can be required to program registers in the core to execute the received Setup command.

3. For the status IN phase, the application must program the core described in “[Generic Non-periodic IN Data Transfers without Threshold Setting](#)” on page [767](#) to perform a data IN transfer.
4. Assertion of the UDC_DIEPINTn.Transfer Compl interrupt indicates the completion of the status IN phase of the control transfer.
5. The previous step must be repeated until the UDC_DIEPINTn.Transfer Compl interrupt is detected on the endpoint, marking the completion of the two-stage control transfer.

Example: Two-Stage Control Transfer

These notes refer to [Figure 18-12](#).

1. SETUP packet #1 is received on the USB and is written to the receive FIFO, and the core responds with an ACK handshake. This handshake is lost and the host detects a timeout.
2. SETUP packet #2 on the USB is written to the receive FIFO, and the core responds with an ACK handshake.
3. The SETUP packet in the receive FIFO sends the application the UDC_GINTSTS.RxFLvl interrupt and the application empties the receive FIFO.
4. After the second SETUP packet, the host sends a control IN token for the status phase. The core issues a NAK response to this token, and writes a Setup Stage Done entry to the receive FIFO. This entry results in a UDC_GINTSTS.RxFLvl interrupt to the application, which empties the receive FIFO. After reading out the Setup Stage Done DWORD, the core asserts the UDC_DOEPINTn.SetUp packet interrupt to the application.
5. On this interrupt, the application processes SETUP Packet #2, decodes it to be a two-stage control command, and clears the control IN NAK bit.
 - UDC_DIEPCTLn.CNAK = 1
6. When the application clears the IN NAK bit, the core interrupts the application with a UDC_DIEPINTn.INTknTXFEMp. On this interrupt, the application enables the control IN endpoint with a UDC_DIEPTSIzn.XferSize of 0 and a UDC_DIEPTSIzn.PktCnt of 1. This results in a zero-length data packet for the status IN token on the USB.
7. At the end of the status IN phase, the core interrupts the application with a UDC_DIEPINTn.XferCompl interrupt.

Figure 18-12. Two-Stage Control Transfer

18.5.6 Device Programming Model — Handling Babble Conditions

If the USB device controller receives a packet that is larger than the maximum packet size for that endpoint, the core stops writing data to the Rx buffer and waits for the end of packet (EOP). When the core detects the EOP, it flushes the packet in the Rx buffer and does not send any response to the host.

If the core continues to receive data at the EOF2 (the end of frame 2, which is very close to SOF), the core generates an early_suspend interrupt (UDC_GINTSTS.ErlySusp). On receiving this interrupt, the application must check the erratic_error status bit (UDC_DSTS.ErrticErr). If this bit is set, the application must take it as a long babble and perform a soft reset.

18.5.7 Device Programming Model -- Partial Power-Down and Clock Gating

18.5.7.1 Device Mode Suspend and Resume With Clock Gating

Sequence of operations:

1. The core detects a USB suspend and generates a Suspend Detected interrupt.
2. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register, the core asserts the suspend_n signal to the PHY, and the PHY clock stops. The application sets the Gate hclk bit in the Power and Clock Gating Control register, and the core gates the hclk (hclk_ctl) to AHB-domain modules other than the BIU.
3. The core remains in Suspend mode.
4. The Resume signaling from the host is detected. The core de-asserts the suspend_n signal to the PHY to generate the PHY clock. A Resume Detected interrupt is generated.
5. The application clears the Gate hclk bit and the Stop PHY Clock bit.
6. The host finishes Resume signaling.
7. The core is in normal operating mode.

18.5.7.2 Device Mode Suspend and Remote Wakeup With Clock Gating

Sequence of operations:

1. The core detects a USB suspend and generates a Suspend Detected interrupt.
2. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register, the core asserts the suspend_n signal to the PHY, and the PHY clock stops. The application sets the Gate hclk bit in the Power and Clock Gating Control register, the core gates the hclk (hclk_ctl) to AHB-domain modules other than the BIU.
3. The core remains in Suspend mode.
4. The application clears the Gate hclk bit and the Stop PHY Clock bit.
5. The application sets the Remote Wakeup bit in the Device Control register, the core starts driving Remote Wakeup signaling.
6. The host drives Resume signaling.
7. The core is in normal operating mode.

18.5.7.3 Device Mode Session End and Start With Clock Gating

Sequence of operations:

1. The core detects a USB suspend, and generates a Suspend Detected interrupt. The host turns off VBUS.
2. The application sets the Stop PHY Clock bit in the Power and Clock Gating Control register, the core asserts the suspend_n signal to the PHY, and the PHY clock stops. The

application sets the Gate hclk bit in the Power and Clock Gating Control register, and the core gates the hclk (hclk_ctl) to AHB-domain modules other than the BIU.

3. The core remains in Low-Power mode.
4. The new session is detected (bsessvld is high). The core de-asserts the suspend_n signal to the PHY to generate the PHY clock. A New Session Detected interrupt is generated.
5. The application clears the Gate hclk and Stop PHY Clock bits.
6. The core detects USB reset.
7. The core is in normal operating mode

18.5.8 Miscellaneous Topics

18.5.8.1 Data FIFO RAM Allocation

If Dynamic FIFO Sizing is enabled in the core, the External RAM must be allocated among different FIFOs in the core before any transactions can start. The application must follow this procedure every time it changes core FIFO RAM allocation.

Device Mode

Dedicated Tx FIFO Operation

When allocating data RAM for FIFOs in Device mode when dedicated TX FIFO is used, keep in mind these factors:

1. 1Receive FIFO RAM Allocation: Receive FIFO RAM allocation is the same as in Shared FIFO operation.
2. 2Transmit FIFO RAM Allocation:
 - The minimum RAM space required for each IN Endpoint Transmit FIFO is the maximum packet size for that particular IN endpoint.
 - The more space allocated in the transmit IN Endpoint FIFO results in a better performance on the USB and can hide the latencies on the AHB.

Table 18-3. Transmit FIFO RAM Allocation

FIFO Name	Data RAM Size
Receive Data FIFO	rx_fifo_size. This must include RAM for SETUP packets, OUT endpoint control information and data OUT packets, as mentioned earlier.
non-periodic Transmit FIFO	tx_fifo_size[0]
Periodic Transmit FIFO 1	tx_fifo_size[1]
Periodic Transmit FIFO 2	tx_fifo_size[2]
.....
Periodic Transmit FIFO <i>i</i>	tx_fifo_size[<i>i</i>]

With this information at hand, the following registers must be programmed as follows:

1. 1Receive FIFO Size Register (UDC_GRXFSIZ) UDC_GRXFSIZ.Receive FIFO Depth = rx_fifo_size;
2. 2Device IN Endpoint Transmit FIFO0 Size Register (UDC_GNPTXFSIZ)
 - UDC_GNPTXFSIZ.non-periodic Transmit FIFO Depth = tx_fifo_size[0];
 - UDC_GNPTXFSIZ.non-periodic Transmit RAM Start Address = rx_fifo_size;

3. 3Device IN Endpoint Transmit FIFO#1 Size Register (DIEPTXF1)
 - DIEPTXF1.Transmit RAM Start Address = UDC_GNPTXFSIZ.FIFO0 Transmit RAM Start Address + tx_fifo_size[0];
4. 4. Device IN Endpoint Transmit FIFO#2 Size Register (DIEPTXF2)
 - DIEPTXF2.Transmit RAM Start Address = DIEPTXF1.Transmit RAM Start Address + tx_fifo_size[1];
5. 5. Device IN Endpoint Transmit FIFO#i Size Register (DIEPTXF*i*)
 - DIEPTXF*m*.Transmit RAM Start Address = DIEPTXF*i*-1.Transmit RAM Start Address + tx_fifo_size[i-1];
6. 6. The transmit FIFOs and receive FIFO must be flushed after the RAM allocation is done, for the proper functioning of the FIFOs.
 - UDC_GRSTCTL.TxFNum = 5'h10
 - UDC_GRSTCTL.TxFFLush = 1'b1
 - UDC_GRSTCTL.RxFFLush = 1'b1
 - The application has to wait until the TxFFLush bit and the RxFFLush bits are cleared, before performing any operation on the core.

Dedicated Tx FIFO operation with Threshold Setting

1. Receive FIFO RAM allocation
 - RAM for Setup Packets: $7*n + 6$ locations have to be Reserved in the receive FIFO, to receive up to “n” setup packets on control endpoints, where “n” is the number of control endpoints supported by the device core. These locations Reserved for Setup Packets are not used by the core, to write any other data.
 - 1 location for Global OUT NAK
 - It is recommended to have an Rx FIFO space for two thresholds. Along with each received threshold, a status information is also written in to the FIFO. With the last threshold of a packet, two status DWORDs are written into the FIFO. With the last packet of a transfer, transfer complete status information is written into the FIFO. So worst case, a minimum of $2*(Rx_threshold\ length/4 + 4)$ space is needed to be allocated to receive a packet.
2. Transmit FIFO RAM Allocation:
 - The minimum RAM space required for each IN Endpoint Transmit FIFO is $\min(2*Tx_threshold_length, endpoint_max_pkt_size)$.
 - The more space allocated in the transmit IN Endpoint FIFO results in a better performance on the USB and can hide the latencies on the AHB.

If high AHB latencies results in underrun error very often, then there is a possibility that the Host might disable this endpoint because of Errors in the packet (typically when there is error in 3 consecutive packets). So threshold setting must be enabled only when the AHB latency is not very high.

18.5.8.2

Dynamic FIFO Allocation

When using Dynamic FIFO allocation in the core, the application can change the RAM allocation for each FIFO during the operation of the core.

In Device mode, before changing FIFO data RAM allocation, the application must determine the following.

- All IN and OUT endpoints are disabled
- NAK mode is enabled in the core on all IN endpoints
- Global OUT NAK mode is enabled in the core

- All FIFOs are empty

Once these conditions are met, the application can reallocate FIFO data RAM as explained in “[Data FIFO RAM Allocation](#)” on page [786](#). When NAK mode is enabled in the core, the core responds with a NAK handshake on all tokens received on the USB, except for SETUP packets.

After the reallocating the FIFO data RAM, the application must flush all FIFOs in the core using the UDC_GRSTCTL.TxFIFO Flush and UDC_GRSTCTL.RxFIFO Flush fields. Flushing is required to reset the pointers in the FIFOs for proper FIFO operation after reallocation.

NETLOGIC
CONFIDENTIAL

18.6 USB Interface Registers

18.6.1 Register Fields

The following terms are used in the register descriptions to describe access to a particular register field.

Table 18-4. Register Field Access

Type Code	Meaning	Description
RO	Read Only	The field contains a static value and cannot be written.
RW	Read/Write	The field can be modified by software. Note that some fields, where indicated, are modifiable only indirectly, by writing to a shadow register then performing a soft reset.
RC	Read/Clear	The field can be read to give status information. Bits may be cleared by writing a 1 to them; writes of '0' are ignored.

18.6.2 Access to Registers

All registers are directly addressed within their address block and are on DWORD boundaries (4 bytes) from the USB base address register (BAR) for XLS_IO_DEV_USB. See [Chapter 8, "Memory and I/O Access"](#) for details.

There are three key concerns for register access:

1. Setup for accessing the USB Interface block within the XLS addressing space
2. Setup for register access; can only be done after configuring interface for either Device Mode or Host Mode operation
3. Accessing registers for the Device Controller, Host Controller, or USB Interface General registers within the USB Interface space.

Setup items 1 and 2 are covered below. For item 3, refer to the Device Register or Host Register description sections.

18.6.2.1 Accessing USB Controller's Configuration Registers

Software can access the USB Controller's own configuration registers through the XLS Peripheral & I/O Configuration region of memory (see Section 8.4, [Table 8-1](#)).

The XLS_IO_BAR register (Register #25) in the System Bridge Controller (SBC) registers provides the base address for accessing the Peripheral & I/O Configuration Region of memory. It is a software-programmable register. Out of reset, the XLS_IO_BAR is enabled and initialized to a default base address (see Section 8.7.1).

The USB controller has two configuration register regions, in which all USB registers reside:

- XLS_IO_DEV_USB_A at region offset 0x24000 from XLS_IO_BAR base address.
- XLS_IO_DEV_USB_B at region offset 0x25000 from XLS_IO_BAR base address.

Within a register region, there is 4KB of space to provide access for up to 1024 registers. See Section 18.7 for the list & descriptions of the USB Controller registers available.

Section 8.5 illustrates how the three different address elements are put together to create the physical address to the XLS Peripheral & I/O Configuration Region of memory that is sent to the System Bridge Controller (SBC). The fields are shown in [Table 18-5](#).

Table 18-5. Physical Address Fields

Physical Address Field (A[x:y])	Description
Address bits [39:20]	Base Address
Address bits [19:12]	Region Offset
Address bits [11:0]	Register Address Offset

Since the CPU is a 64-bit machine, the software generated address for the XLS Peripheral & I/O Configuration Region is a 64-bit address that will be translated by the MMU into a 40-bit physical address sent to SBC.

When SBC gets a memory transaction that matches the memory region for the USB configuration registers, it will forward that request to the USB controller which will perform the required configuration register access.

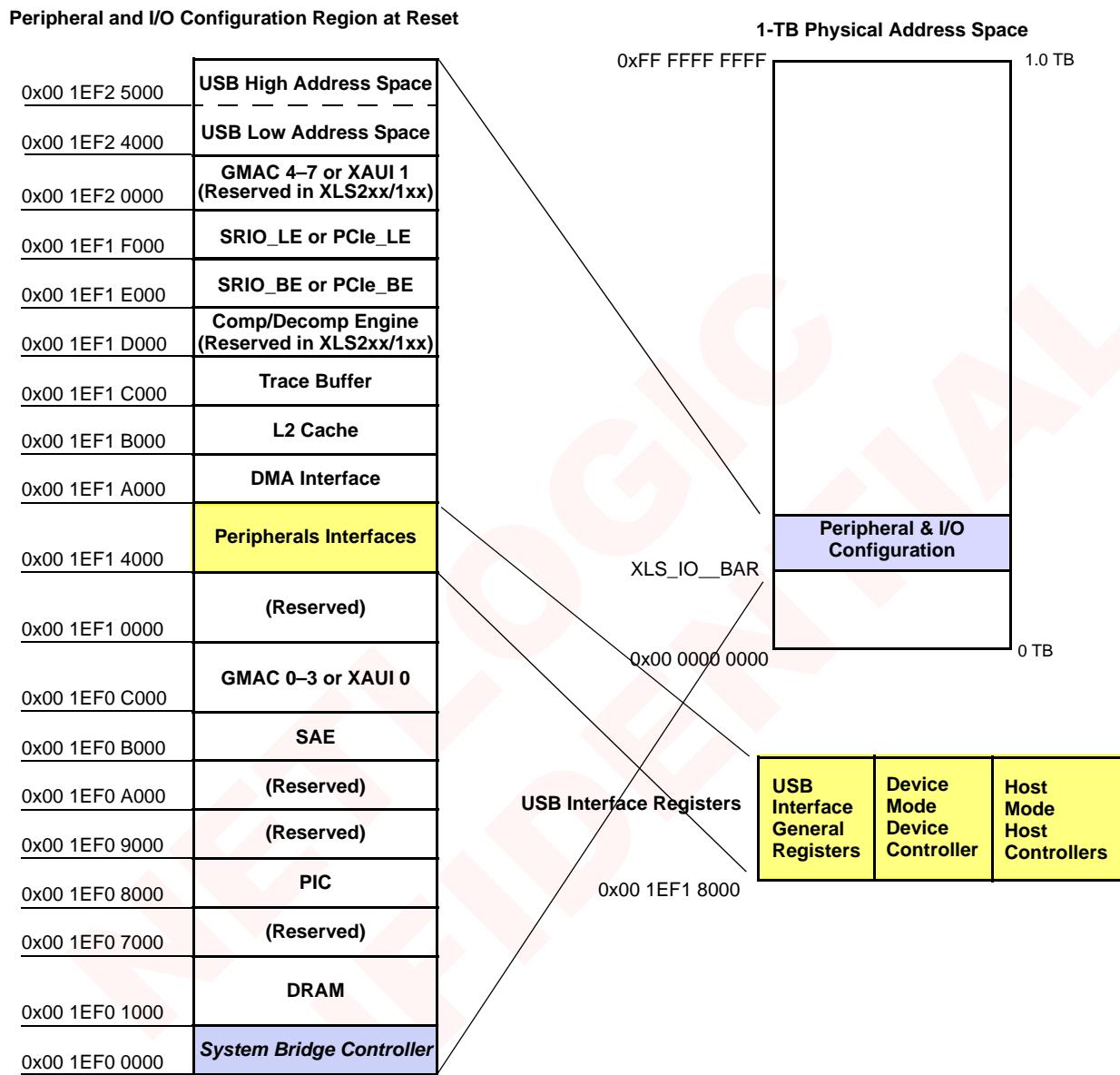
See [Figure 18-13](#) for a diagram showing the position of the USB Interface in the XLS memory space.

18.6.2.2

General Setup for Register Access

Before the registers in the USB Interface can be first accessed, the interface must be configured for operation in either Device Mode or in Host Mode. This is done through a bit field in the GPIO registers. Refer to the chapter on GPIO for register information on this.

After this is done, the USB Interface General Registers or Device Controller or Host Controller registers can be accessed.

Figure 18-13. Location of the USB Interface Within the XLS Addressing Space

18.7 USB Interface General Registers

These registers control aspects of the entire USB interface that do not lie within the Device Controller or Host Control blocks.

18.7.1 Accessing the USB Interface General Registers

To access these registers, the USB Interface must first have been configured for either Device Mode or Host Mode operation.

All addresses are on DWORD boundaries (4 bytes) from the USB base address register (BAR) for XLS_IO_DEV_USB. See [Chapter 8, “Memory and I/O Access”](#) for details.

18.7.2 USB Interface General Registers Summary

Table 18-6. Summary of USB Interface General Registers

Byte Address Offset	Name	Acronym
0x000	USB_GEN_CTRL1	UG_CTL1
0x004	USB_GEN_CTRL2	UG_CTL2
0x008	USB_GEN_CTRL3	UG_CTL3
0x00C	USB_IOBM_TIMER	UG_IOBM_TIMER
0x010	USB_VBUS_TIMER	UG_VBUS_TIMER
0x014	USB_BYTESWAP_EN	BYTESWAP_EN
0x018 to 0x03C	Reserved	
0x040	USB_COHERENT_MEM_BASE	UG_COH_MEMBASE
0x044	USB_COHERENT_MEM_LIMIT	UG_COH_MEMLIM
0x048	USB_L2ALLOC_MEM_BASE	UG_L2ALLOC_MEMBASE
0x04C	USB_L2ALLOC_MEM_LIMIT	UG_L2ALLOC_MEMLIM
0x050	USB_READEX_MEM_BASE	UG_READEX_MEMBASE
0x054	USB_READEX_MEM_LIMIT	UG_READEX_MEMLIM
0x058 to 0x07C	Reserved	
0x080 to 0x0BC	Reserved	
0x0C0	USB_PHY_STATUS	UG_PHY_STAT
0x0C4	USB_INTERRUPT_STATUS	UG_INT_STATUS
0x0C8	USB_INTERRUPT_ENABLE	UG_INT_EN
0x0CC to 0x0EC	Reserved	

18.7.3 USB Interface General Registers Descriptions**18.7.4 USB_GEN_CTRL1****USB Register Address Offset: 0x000**

Bits	Field Name	Field Description	R/W	Reset
31:2	<i>Reserved</i>	<i>Reserved</i>		
1	HOST_RST_N	Resets the Host Controller 0: reset 1: normal operation	RW	0x0
0	DEVICE_RST_N	Resets the Device Controller 0: normal operation 1: reset	RW	0x0

18.7.5 USB_GEN_CTRL2**USB Register Address Offset: 0x004**

Bits	Field Name	Field Description	R/W	Reset
31:20	<i>Reserved</i>			
19:18	TX_TUNE_1	Port_1 Transmitter Tuning for High-Speed Operation. 00: ~-4.5% 01: Design default 10: ~+4.5% 11: ~+9% = Recommended Operating setting This bus is a strapping option that must be set prior to a power-on reset and remain static during normal operation.	RW	0x1
17:16	TX_TUNE_0	Port_0 Transmitter Tuning for High-Speed Operation. Set to '11' for Recommended Operating condition	RW	0x1
15	<i>Reserved</i>			0x0
14	WEAK_PULLDOWN_EN	500-kΩ Pull-Down Resistor on D+ and D- Enable. 1: Enabled 0: Disabled In normal operation, weak_pulldown_en must be set high.	RW	0x0
13	DP_PULLUP_ESD	D+ Pull-Up Resistor Enable. 1: Enabled 0: Disabled In normal operations (esd_test_mode = '0'), this signal is don't-care input.	RW	0x0
12	ESD_TEST_MODE	D+ Pull-Up Resistor Control Select. 1: The dp_pullup_esd signal controls the pull-up resistor on D+. 0: The internal D+ pull-up circuit controls the pull-up resistor on D+. In normal operation, esd_test_mode must be set low.	RW	0x0
11	TX_BIT_STUFF_EN_H_1	Port_1 High-Byte Transmit Bit-Stuffing Enable. 1: Enabled 0: Disabled This controller signal control bit stuffing on data_inh[7:0] when opmode[1:0] = 2'b11.	RW	0x0
10	TX_BIT_STUFF_EN_H_0	Port_0 High-Byte Transmit Bit-Stuffing Enable.	RW	0x0

Bits	Field Name	Field Description	R/W	Reset
9	TX_BIT_STUFF_EN_1	Port_1 Low-Byte Transmit Bit-Stuffing Enable. 1: Enabled 0: Disabled This controller signal control bit stuffing on data_in[7:0] when opmode[1:0]=2'b11.	RW	0x0
8	TX_BIT_STUFF_EN_0	Port_0 Low-Byte Transmit Bit-Stuffing Enable.	RW	0x0
7:6	<i>Reserved</i>			0x0
5	LOOPBACK_ENB_1	Port_1 Loopback Test Enable. 1: Enabled 0: Disabled The enable places the USB 2 PHY in loopback mode, which enables the receive and transmit logic concurrently.	RW	0x0
4	LOOPBACK_ENB_0	Port_0 Loopback Test Enable.	RW	0x0
3	DEVICE_DETECT_VBUS	VBUS detected (Device mode only).	RW	0x0
2	PHY_PORT_RST_N_1	Resets Port_1 of the PHY. 1: normal operation 0: reset	RW	0x0
1	PHY_PORT_RST_N_0	Resets Port_0 of the PHY. 1: normal operation 0: reset	RW	0x0
0	PHY_RST_N	Resets the PHY. 1: normal operation 0: reset	RW	0x0

18.7.6 USB_GEN_CTRL3

USB Register Address Offset: 0x008

Bits	Field Name	Field Description	R/W	Reset
31:11	<i>Reserved</i>			
10:8	PREFETCH_SIZE	The pre-fetch size for a memory read transfer between USB Interface and DI station. Valid value ranges is from 1 to 4.	RW	0x4
7	<i>Reserved</i>			
6:1	DEV_UPPERADDR	This field is used to determine which of the 64 4KB Device controller address spaces the register access is for. (Device Mode only)	RW	0x00
0	USB_FLUSH	Flushes the USB Interface; necessary when resetting the USB Interface separately from an XLS system reset.	RW	0x0

18.7.7 USB_IOBM_TIMER

USB Register Address Offset: 0x00C

Bits	Field Name	Field Description	R/W	Reset
31:0	IOBM_REQUEST_TIMER	Number of IodClk cycles for an unaccepted IOB master request to timeout	RW	0xFFFFFFFF

18.7.8 USB_VBUS_TIMER**USB Register Address Offset: 0x010**

Bits	Field Name	Field Description	R/W	Reset
31:0	VBUS_TIMER	Number of IodClk cycles for a VBUS value to be valid. Default is 1ms with IodClk of 66.67MHz	RW	0x000101D0

18.7.9 USB_BYTESWAP_EN**USB Register Address Offset: 0x014**

Bits	Field Name	Field Description	R/W	Reset
31:0	USB_BYTESWAP_EN	Byte swapping is enabled when the register is not all '0's. When the register is not all '0's, the software is in Little Endian format instead of Big Endian format.	RW	0x00000000

18.7.10 USB_COHERENT_MEM_BASE**USB Register Address Offset: 0x040**

Bits	Field Name	Field Description	R/W	Reset
31:0	USB_COHERENT_MEM_BASE	USB request memory space base for coherency; the memory base addresses minimum memory size of 32B. Programming this base address and also the USB_COHERENT_MEM_LIMIT register together defines a low (base) and high (limit) address range. See the description of the USB_COHERENT_MEM_LIMIT register for a description of how this range affects USB accesses.	RW	0xFFFFFFFF

18.7.11 USB_COHERENT_MEM_LIMIT

USB Register Address Offset: 0x044

Bits	Field Name	Field Description	R/W	Reset
31:0	USB_COHERENT_MEM_LIMIT	<p>USB request memory space limit for coherency; the memory limit addresses minimum memory size of 32B.</p> <p>Programming this limit address and also the USB_COHERENT_MEM_BASE register together defines a low (base) and high (limit) address range.</p> <p>If the address of an incoming USB memory read transaction falls between the L2 Allocation memory base and limit defined by this range, and if the read data is not present in the L2 Cache, then the read data is stored in the L2 Cache.</p> <p>If the address of an incoming USB memory write transaction falls between the L2 Allocation memory base and limit defined by this range, the write data goes into the L2 cache.</p>	RW	0x00000000

18.7.12 USB_L2ALLOC_MEM_BASE

USB Register Address Offset: 0x048

Bits	Field Name	Field Description	R/W	Reset
31:0	USB_L2ALLOC_MEM_BASE	<p>USB request memory space base for L2 allocation; the memory base addresses minimum memory size of 32B.</p> <p>Programming this base address and also the USB_L2ALLOC_MEM_LIMIT register together defines a low (base) and high (limit) address range. See the description of the USB_L2ALLOC_MEM_LIMIT register for a description of how this range affects USB accesses.</p>	RW	0xFFFFFFFF

18.7.13 USB_L2ALLOC_MEM_LIMIT**USB Register Address Offset: 0x04C**

Bits	Field Name	Field Description	R/W	Reset
31:0	USB_L2ALLOC_MEM_LIMIT	<p>USB request memory space limit for L2 allocation; the memory limit addresses minimum memory size of 32B.</p> <p>Programming this limit address and also the USB_L2ALLOC_MEM_BASE register together defines a low (base) and high (limit) address range.</p> <p>If the address of an incoming USB memory read transaction falls between the L2 Allocation memory base and limit defined by this range, and if the read data is not present in the L2 Cache, then the read data is stored in the L2 Cache.</p> <p>If the address of an incoming USB memory write transaction falls between the L2 Allocation memory base and limit defined by this range, the write data goes into the L2 cache.</p>	RW	0x00000000

18.7.14 USB_READEX_MEM_BASE**USB Register Address Offset: 0x050**

Bits	Field Name	Field Description	R/W	Reset
31:0	USB_READEX_MEM_BASE	<p>USB request memory space base for read exclusive; the memory base addresses minimum memory size of 32B.</p> <p>Programming this base address and also the USB_READEX_MEM_LIMIT register together defines a low (base) and high (limit) address range. See the description of the USB_READEX_MEM_LIMIT register for a description of how this range affects USB accesses.</p>	RW	0xFFFFFFFF

18.7.15 USB_READEX_MEM_LIMIT

USB Register Address Offset: 0x054

Bits	Field Name	Field Description	R/W	Reset
31:0	USB_READEX_MEM_LIMIT	<p>USB request memory space limit for read exclusive; the memory limit addresses minimum memory size of 32B.</p> <p>Programming this limit address and also the USB_READEX_MEM_BASE register together defines a low (base) and high (limit) address range.</p> <p>If the address of an incoming USB memory read transaction falls between the Read Exclusive memory base and limit defined by this range, and if the read data is present in the L2 Cache, then the read data is flushed from the L2 Cache.</p> <p>The READEX range has no impact on write transactions.</p>	RW	0x00000000

18.7.16 USB_PHY_STATUS

USB Register Address Offset: 0x0C0

Reset value: 0x00000000

Bits	Field Name	Field Description	R/W	Reset
31:1	<i>Reserved</i>			
0	VBUS	USB VBUS status.	RO	0x0

18.7.17 USB_INTERRUPT_STATUS

USB Register Address Offset: 0x0C4

Bits	Field Name	Field Description	R/W	Reset
31:6	<i>Reserved</i>			
5	FORCE	USB force interrupt	RW	0x0
4	PHY	USB PHY interrupt	RW1C	0x0
3	DEVICE	USB Device Controller interrupt.	RO	0x0
2	EHCI	USB Host EHCI Controller interrupt.	RO	0x0
1	OHCI_1	USB Host OHCI Controller 1 interrupt.	RO	0x0
0	OHCI_0	USB Host OHCI Controller 0 interrupt.	RO	0x0

18.7.18 USB_INTERRUPT_ENABLE

USB Register Address Offset: 0x0C8

Bits	Field Name	Field Description	R/W	Reset
31:4	Reserved			
5	FORCE	USB force interrupt enable. 0: disabled 1: enabled	RW	0x0
4	PHY	USB PHY interrupt enable. 0: disabled 1: enabled	RW	0x0
3	DEVICE	USB Device Controller interrupt enable. 0: disabled 1: enabled	RW	0x0
2	EHCI	USB Host EHCI Controller interrupt enable. 0: disabled 1: enabled	RW	0x0
1	OHCI_1	USB Host OHCI Controller 1 interrupt enable. 0: disabled 1: enabled	RW	0x0
0	OHCI_0	USB Host OHCI Controller 0 interrupt enable 0: disabled 1: enabled	RW	0x0

18.8 Device Controller Registers

Device Controller registers are 32 bits wide, and the addresses are 32-bit block aligned.

18.8.1 Accessing the USB Device Controller Registers

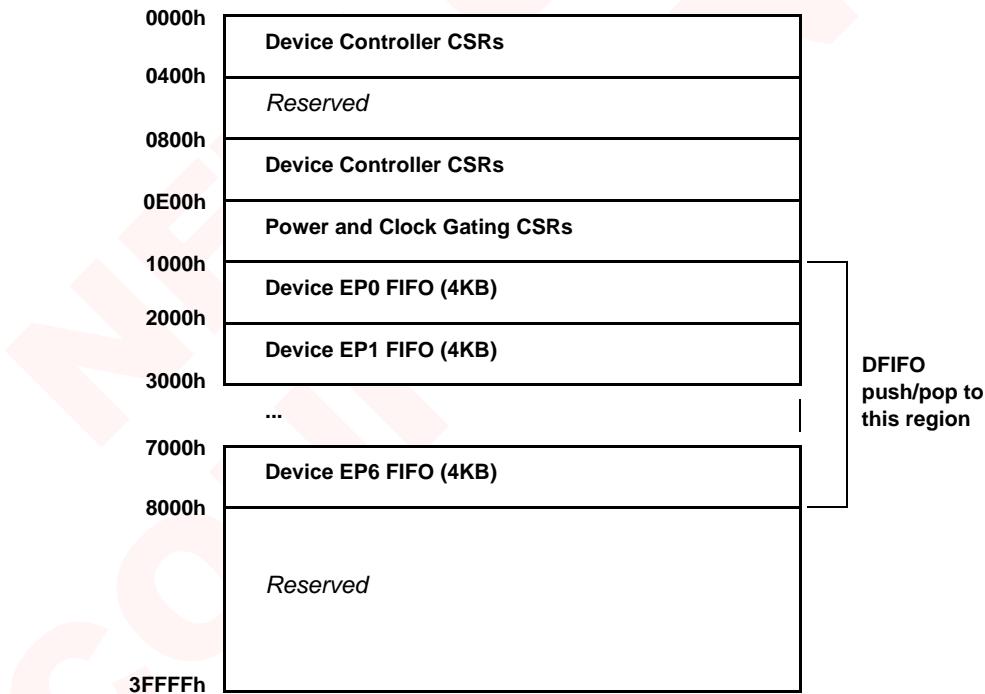
To access these registers, the USB Interface must first have been configured for Device Mode operation.

All addresses are on DWORD boundaries (4 bytes) from the USB base address register (BAR) for XLS_IO_DEV_USB. See [Chapter 8, “Memory and I/O Access”](#) for details. The upper 6 bits of the register access are set by the DEV_UPPERADDR field of the USB_GEN_CTRL3 register. [Table 18-7](#) shows how to address the lower 12 bits of the USB Device Control register.

Table 18-7. Register Address Fields for Accessing USB Interface in Device Mode

Field	Bits	Description
Reserved	39:13	
REGSPACE	12	Register space selector: 0: Access Device Controller registers 1: Access USB Interface General registers
REGADDR	11:0	Register address

Figure 18-14. USB Device Controller Address Block Map



The registers in each block are summarized in the following tables, followed by register descriptions.

18.8.2 Device Controller Control and Status Registers Summary

Table 18-8. USB Device Controller CSR Summary

Offset Address	Register Name	Acronym
008h	UDC Configuration A (UDC_GAHBCFG)	UDC_GAHBCFG
00Ch	UDC Configuration B (UDC_GUSBCFG)	UDC_GUSBCFG
010h	UDC Reset	UDC_GRSTCTL
014h	UDC Interrupt Register	UDC_GINTSTS
018	UDC Interrupt Mask Register	UDC_GINTMSK
020h	UDC Receive Status Read /Pop Register (Read Only)	UDC_GRXSTSP
024h	UDC Receive FIFO Size Register	UDC_GRXFSIZ
028h	UDC Non-periodic Transmit FIFO Size Register	UDC_GNPTXFSIZ
02Ch-038h	Reserved	
03Ch	UDC User ID Register (UDC_GUID)	UDC_GUID
040h	UDC ID Register (Read Only)	UDC_GSNPSID
044h	UDC User HW Config1 Register (Read Only)	UDC_GHWCFG1
048h	UDC User HW Config2 Register (Read Only)	UDC_GHWCFG2
04Ch	UDC User HW Config3 Register (Read Only)	UDC_GHWCFG3
050h	UDC User HW Config4 Register (Read Only)	UDC_GHWCFG4
054h-0FFh	Reserved	
100h	Reserved	
104h-10Ch	UDC Device IN Endpoint Transmit FIFO-n Size Register (UDC_DPTXFSIZn)	UDC_DPTXFSIZn
110h-7FCh	Reserved	
800h	UDC Configuration C	UDC_DCFG
804h	UDC Control Register	UDC_DCTL
808h	UDC Status Register (Read Only)	UDC_DSTS
80Ch	Reserved	Reserved
810h	UDC Device IN Endpoint Common Interrupt Mask Register (UDC_DIEPMSK)	UDC_DIEPMSK
814h	UDC Device OUT Endpoint Common Interrupt Mask register	UDC_DOEPMASK
818h	UDC Device All Endpoints Interrupt Register	UDC_DAINT
81Ch	UDC Device All Endpoints Interrupt Mask Register	UDC_DAINTMSK
820h-82Ch	Reserved	
830h	Device Threshold Control Register	UDC_DTKNQR3
834h	Device IN Endpoint FIFO Empty Interrupt Mask Register	UDC_DTKNQR4
838h-8FCh	Reserved	
900h	Device Control IN Endpoint 0 Control Register	UDC_DIEPCTLn
904h	Reserved	
908h	Device IN Endpoint 0 Interrupt Register	UDC_DIEPINTn
90Ch	Reserved	
910h	Device IN Endpoint 0 Transfer Size Register	UDC_DIEPTSIZn
914h	Device IN Endpoint 0 DMA Address Register	UDC_DIEPDMAAn
918h	Device IN Endpoint Transmit FIFO Status Register	UDC_DTXFSTSAn
91Ch	Reserved	
920h-93Ch	Device IN Endpoint 1 Registers	
940h-95Ch	Device IN Endpoint 2 Registers	
...	...	
9A0-9BCh	Device IN Endpoint 5 Registers	

Table 18-8. USB Device Controller CSR Summary

Offset Address	Register Name	Acronym
9C0h–90Ch	Device IN Endpoint 6 Registers	
9E0h–AFCh	<i>Reserved</i>	
B00h	Device Control OUT Endpoint 0 Control Register	UDC_DOEPCTLn
B04h	<i>Reserved</i>	
B08h	Device OUT Endpoint 0 Interrupt Register	UDC_DOEPINTn
B0Ch	<i>Reserved</i>	
B10h	Device OUT Endpoint 0 Transfer Size Register	UDC_DOEPTSIzn
B14h	Device OUT Endpoint 0 DMA Address Register	UDC_DOEPDMAAn
B14h–B1Ch	<i>Reserved</i>	
B20h–B3Ch	Device OUT Endpoint 1 Registers	
B40h–B5Ch	Device OUT Endpoint 2 Registers	
...
BA0h–BBCh	Device OUT Endpoint 5 Registers	
BC0h–BDCh	Device OUT Endpoint 6 Registers	
BE0h–BFCh	<i>Reserved</i>	
CFDh–DFFh	<i>Reserved</i>	
E00h–FFFh	Power and Clock Gating Registers	
E00h	Power and Clock Gating Control Register	UDC_PCGCR
E05h–FFFh	<i>Reserved</i>	

18.8.3 Register Fields

The following terms are used in the device controller register descriptions to describe access to a particular register field.

Table 18-9. Register Field Access

Type Code	Meaning	Description
RO	Read Only	Register field can only be read by the application. Writes to read-only fields have no effect.
WO	Write Only	Register field can only be written by the application.
RW	Read and Write	Register field can be read and written by the application. The application can set this field by writing 1'b1 and can clear it by writing 1'b0.
R_W_SC	Read, Write, and Self Clear	Register field can be read and written by the application (Read and Write), and is cleared to 1'b0 by the core (Self Clear). The conditions under which the core clears this field are explained in detail in the field's description.
R_W_SS_SC	Read, Write, Self Set, and Self Clear	Register field can be read and written by the application (Read and Write), set to 1'b1 by the core on certain USB events (Self Set), and cleared to 1'b0 by the core (Self Clear). The conditions under which the core sets and clears this field are explained in the field's description. (Only the Port Resume bit of the Host Port Control and Status register, HPRT.PrtRes, uses this access type).
R_SS_WC	Read, Self Set, and Write Clear	Register field can be read by the application (Read), can be set to 1'b1 by the core on a certain internal or USB or AHB event (Self Set), and can be cleared to 1'b0 by the application with a register write of 1'b1 (Write Clear). A register write of 1'b0 has no effect on this field. The conditions under which the core sets this field are explained in detail in the field's description. (For example, interrupt bits.)
R_WS_SC	Read, Write Set, and Self Clear	Register field can be read by the application (Read), can be set to 1'b1 by the application with a register write of 1'b1 (Write Set), and is cleared to 1'b0 by the core (Self Clear). The application cannot clear this type of field, and a register write of 1'b0 to this bit has no effect on this field. The conditions under which the core clears this field are explained in detail in the field's description. (For example, reset signals)
R_SS_SC_WC	Read, Self Set, and Self Clear or Write Clear	Register field can be read by the application (Read), can be set to 1'b1 by the core on certain internal or USB or AHB events (Self Set), and can be cleared to 1'b0 either by the core itself (Self Clear) or by the application with a register write of 1'b1 (Write Clear). A register write of 1'b0 to this bit has no effect on this field. The conditions under which the core sets or clears this field are explained in the field's description. (Only the Port Enable bit of the Host Port Control and Status register, HPRT.PrtEna, and the VStatus Done bit of the PHY Vendor Control register, GPVNDCTL.VStsDone, use this access type.)

18.8.4**USB Device Controller Data FIFO (DFIFO) Register Summary**

These registers are used to read or write the FIFO space for a specific endpoint or a channel, in a given direction. If a channel is of type IN, the FIFO can only be read on the channel. Similarly, if a channel is of type OUT, the FIFO can only be written on the channel.

Table 18-10. Data FIFO (DFIFO) Access Register Map

Address Range	FIFO Access Register Section	Access
1000h–1FFCh	Device IN Endpoint 0: DFIFO Write Access Device OUT Endpoint 0: DFIFO Read Access	WO/RO
2000h–2FFCh	Device IN Endpoint 1: DFIFO Write Access Device OUT Endpoint 1: DFIFO Read Access	WO/RO
...
F000h–FFFCh	Device IN Endpoint 14: DFIFO Write Access Device OUT Endpoint 14: DFIFO Read Access	WO/RO
10000h–10FFCh	Device IN Endpoint 15: DFIFO Write Access Device OUT Endpoint 15: DFIFO Read Access	WO/RO

18.9 Device Controller Register Descriptions

18.9.1 UDC Configuration A (UDC_GAHBCFG)

This register can be used to configure the core after power-on or a change in mode of operation. This register mainly contains AHB system-related configuration parameters. Do not change this register after the initial programming. The application must program this register before starting any transactions on either the AHB or the USB.

USB Register Address Offset: 008h

Bits	Field Name	Field Description	R/W	Reset
31:6	Reserved			
5	DMA Enable (DMAEn)	<ul style="list-style-type: none"> • 1'b0: Core operates in Slave mode • 1'b1: Core operates in a DMA mode 	RW	1'b1
4:1	Burst Length/Type (HBstLen)	<p>This field is used in both External and Internal DMA modes. In External DMA mode, these bits appear on dma_burst[3:0] ports, which can be used by an external wrapper to interface the External DMA Controller interface to DW_ahb_dmac or ARM PrimeCell.</p> <p>Internal DMA Mode—AHB Master burst type:</p> <ul style="list-style-type: none"> • 4'b0000 Single • 4'b0001 INCR • 4'b0011 INCR4 • 4'b0101 INCR8 • 4'b0111 INCR16 • Others: Reserved 	RW	4'b0
0	Global Interrupt Mask (GblIntrMsk)	<p>The application uses this bit to mask or unmask the interrupt line assertion to itself. Irrespective of this bit's setting, the interrupt status registers are updated by the core.</p> <ul style="list-style-type: none"> • 1'b0: Mask the interrupt assertion to the application. • 1'b1: Unmask the interrupt assertion to the application. 	RW	1'b0

18.9.2 UDC Configuration B (UDC_GUSBCFG)

This register can be used to configure the core after power-on or a changing to Host mode or Device mode. It contains USB and USB-PHY related configuration parameters. The application must program this register before starting any transactions on either the AHB or the USB. Do not make changes to this register after the initial programming.

USB Register Address Offset: 00Ch

Bits	Field Name	Field Description	R/W	Reset
31	Corrupt Tx packet	This bit is for debug purposes only. Never set this bit to 1.	RW	1'b0
30:16	<i>Reserved</i>			
15	PHY Low-Power Clock Select (PhyPwrClkSel)	Selects either 480-MHz or 48-MHz (low-power) PHY mode. In FS and LS modes, the PHY can usually operate on a 48-MHz clock to save power. <ul style="list-style-type: none"> • 1'b0: 480-MHz Internal PLL clock • 1'b1: 48-MHz External Clock 	RW	1'b0
14	<i>Reserved</i>			1'b0
13:10	USB Turnaround Time (USBTrdTim)	Sets the turnaround time in PHY clocks. Specifies the response time for a MAC request to the Packet FIFO Controller (PFC) to fetch data from the DFIFO (SPRAM). This must be programmed to <ul style="list-style-type: none"> • 4'h5: When the MAC interface is 16-bit UTMI+ • 4'h9: When the MAC interface is 8-bit UTMI+ 	RW	4'h5
9:7	<i>Reserved</i>			
6	USB 2.0 High-Speed PHY or USB 1.1 Full-Speed Serial Transceiver Select (PHYSel)	The application uses this bit to select either a high-speed UTMI+ or ULPI PHY, or a full-speed transceiver. <ul style="list-style-type: none"> • 1'b0: USB 2.0 high-speed UTMI+ or ULPI PHY • 1'b1: USB 1.1 full-speed serial transceiver 	WO	1'b0
5	Full-Speed Serial Interface Select (FSIntf)	The application uses this bit to select either a unidirectional or bidirectional USB 1.1 full-speed serial transceiver interface. <ul style="list-style-type: none"> • 1'b0: 6-pin unidirectional full-speed serial interface • 1'b1: 3-pin bidirectional full-speed serial interface 	WO	1'b0
4	ULPI or UTMI+ Select (ULPI_UTMI_Sel)	The application uses this bit to select either a UTMI+ interface or ULPI Interface. <ul style="list-style-type: none"> • 1'b0: UTMI+ Interface • 1'b1: ULPI Interface 	RO	1'b1

Bits	Field Name	Field Description	R/W	Reset
3	PHY Interface (PHYIf)	The application uses this bit to configure the core to support a UTMI+ PHY with an 8- or 16-bit interface. When a ULPI PHY is chosen, this must be set to 8-bit mode. <ul style="list-style-type: none"> • 1'b0: 8 bits • 1'b1: 16 bits 	RW	1'b0
2:0	HS/FS Timeout Calibration (TOutCal)	The number of PHY clocks that the application programs in this field is added to the high-speed/full-speed inter-packet timeout duration in the core to account for any additional delays introduced by the PHY. This can be required, because the delay introduced by the PHY in generating the linestate condition can vary from one PHY to another. The USB standard timeout value for high-speed operation is 736 to 816 (inclusive) bit times. The USB standard timeout value for full-speed operation is 16 to 18 (inclusive) bit times. The application must program this field based on the speed of enumeration. The number of bit times added per PHY clock are: High-speed operation: <ul style="list-style-type: none"> • One 30-MHz PHY clock = 16 bit times • One 60-MHz PHY clock = 8 bit times Full-speed operation: <ul style="list-style-type: none"> • One 30-MHz PHY clock = 0.4 bit times • One 60-MHz PHY clock = 0.2 bit times • One 48-MHz PHY clock = 0.25 bit times 	RW	3'h0

18.9.3 UDC Reset (UDC_GRSTCTL)

The application uses this register to reset various hardware features inside the core.

USB Register Address Offset: 010h

Bits	Field Name	Field Description	R/W	Reset
31	AHB Master Idle (AHBIdle)	Indicates that the AHB Master State Machine is in the IDLE condition.	RO	1'b1
30	DMA Request Signal (DMAReq)	Indicates that the DMA request is in progress. Used for debug.	RO	1'b0
29:11	Reserved			19'h0
10:6	TxFIFO Number (TxFNum)	<p>This is the FIFO number that must be flushed using the TxFIFO Flush bit. This field must not be changed until the core clears the TxFIFO Flush bit.</p> <p>5'h0:</p> <ul style="list-style-type: none"> • Non-periodic TxFIFO flush in Host mode • Non-periodic TxFIFO flush in device mode when in shared FIFO operation • Tx FIFO 0 flush in device mode when in dedicated FIFO mode <p>5'h1:</p> <ul style="list-style-type: none"> • Periodic TxFIFO flush in Host mode • Periodic TxFIFO 1 flush in Device mode when in shared FIFO operation • TxFIFO 1 flush in device mode when in dedicated FIFO mode <p>5'h2:</p> <ul style="list-style-type: none"> • Periodic TxFIFO 2 flush in Device mode when in shared FIFO operation • TxFIFO 2 flush in device mode when in dedicated FIFO mode <p>...</p> <p>5'hF:</p> <ul style="list-style-type: none"> • - Periodic TxFIFO 15 flush in Device mode when in shared FIFO operation • - TxFIFO 15 flush in device mode when in dedicated FIFO mode <p>5'h10:</p> <ul style="list-style-type: none"> • Flush all the transmit FIFOs in device or host mode. 	RW	5'h0
5	TxFIFO Flush (TxFFlsh)	<p>This bit selectively flushes a single or all transmit FIFOs, but cannot do so if the USB controller is in the midst of a transaction. The application must write this bit only after checking that the USB controller is neither writing to the TxFIFO nor reading from the TxFIFO. Verify using these registers:</p> <ul style="list-style-type: none"> • Read—NAK Effective Interrupt ensures the USB controller is not reading from the FIFO • Write—UDC_GRSTCTL.AHBIdle ensures the USB controller is not writing anything to the FIFO. <p>Flushing is normally recommended when FIFOs are reconfigured or when switching between Shared FIFO and Dedicated Transmit FIFO operation. FIFO flushing is also recommended during device endpoint disable.</p> <p>The application must wait until the USB controller clears this bit before performing any operations. This bit takes eight clocks to clear, using the slower clock of phy_clk or hclk.</p>	R_WS_SC	1'b0

Bits	Field Name	Field Description	R/W	Reset
4	RxFIFO Flush (RxFFlsh)	The application can flush the entire RxFIFO using this bit, but must first ensure that the USB controller is not in the middle of a transaction. The application must only write to this bit after checking that the USB controller is neither reading from the RxFIFO nor writing to the RxFIFO. The application must wait until the bit is cleared before performing any other operations. This bit requires 8 clocks (slowest of PHY or AHB clock) to clear.	R_WS_SC	1'b0
3:2	Reserved			

Bits	Field Name	Field Description	R/W	Reset
1	HClk Soft Reset (HSftRst)	<p>The application uses this bit to flush the control logic in the AHB Clock domain. Only AHB Clock Domain pipelines are reset.</p> <ul style="list-style-type: none"> • FIFOs are not flushed with this bit. • All state machines in the AHB clock domain are reset to the Idle state after terminating the transactions on the AHB, following the protocol. • CSR control bits used by the AHB clock domain state machines are cleared. • To clear this interrupt, status mask bits that control the interrupt status and are generated by the AHB clock domain state machine are cleared. • Because interrupt status bits are not cleared, the application can get the status of any core events that occurred after it set this bit. <p>This is a self-clearing bit that the core clears after all necessary logic is reset in the core. This can take several clocks, depending on the core's current state.</p>	R_WS_SC	1'b0
0	Core Soft Reset (CSftRst)	<p>Resets the hclk and phy_clock domains as follows:</p> <ul style="list-style-type: none"> • Clears the interrupts and all the CSR registers except the following register bits: PCGCCTL.RstPdwnModule PCGCCTL.GateHclk PCGCCTL.PwrClmp PCGCCTL.StopPPhyLPwrClkSelClk UDC_GUSBCFG.PhyLPwrClkSel UDC_GUSBCFG.DDRSel UDC_GUSBCFG.PHYSel UDC_GUSBCFG.FSIntf UDC_GUSBCFG.ULPI_UTMI_Sel UDC_GUSBCFG.PHYIf HCFG.FSLSPclkSel UDC_DCFG.DevSpd GPIO • All module state machines (except the AHB Slave Unit) are reset to the IDLE state, and all the transmit FIFOs and the receive FIFO are flushed. • Any transactions on the AHB Master are terminated as soon as possible, after gracefully completing the last data phase of an AHB transfer. Any transactions on the USB are terminated immediately. <p>The application can write to this bit any time it wants to reset the core. This is a self-clearing bit and the core clears this bit after all the necessary logic is reset in the core, which can take several clocks, depending on the current state of the core. Once this bit is cleared software must wait at least 3 PHY clocks before doing any access to the PHY domain (synchronization delay). Software must also must check that bit 31 of this register is 1 (AHB Master is IDLE) before starting any operation.</p> <p>Typically software reset is used during software development and also when you dynamically change the PHY selection bits in the USB configuration registers listed above. When you change the PHY, the corresponding clock for the PHY is selected and used in the PHY domain. Once a new clock is selected, the PHY domain has to be reset for proper operation.</p>	R_WS_SC	1'b0

18.9.4 UDC Interrupt Register (UDC_GINTSTS)

This register interrupts the application for system-level events in the current mode of operation (Device mode or Host mode).

This register also indicates the current mode of operation. In order to clear the interrupt status bits of type R_SS_WC, the application must write 1'b1 into the bit.

The FIFO status interrupts are read only; once software reads from or writes to the FIFO while servicing these interrupts, FIFO interrupt conditions are cleared automatically.

USB Register Address Offset: 014h

Bits	Field Name	Field Description	R/W	Reset
31	Resume/Remote Wakeup Detected Interrupt (WkUpInt)	Asserted when a resume is detected on the USB.	R_SS_WC	1'b0
30	Reserved			1'b0
29	Disconnect Detected Interrupt (Disconnect)	Asserted when a device disconnect is detected.	R_SS_WC	1'b0
28:23	Reserved			1'b0
22	Data Fetch Suspended (FetSusp)	<p>This interrupt is valid only in DMA mode. This interrupt indicates that the core has stopped fetching data for IN endpoints due to the unavailability of TxFIFO space or Request Queue space. This interrupt is used by the application for an endpoint mismatch algorithm.</p> <p>For example, after detecting an endpoint mismatch, the application:</p> <ul style="list-style-type: none"> • Sets a global non-periodic IN NAK handshake • Disables In endpoints • Flushes the FIFO • Determines the token sequence from the IN Token Sequence Learning Queue • Re-enables the endpoints • Clears the global non-periodic IN NAK handshake <p>If the global non-periodic IN NAK is cleared, the core has not yet fetched data for the IN endpoint, and the IN token is received: the core generates an “IN token received when FIFO empty” interrupt. The core then sends the host a NAK response. To avoid this scenario, the application can check the UDC_GINTSTS.FetSusp interrupt, which ensures that the FIFO is full before clearing a global NAK handshake.</p> <p>Alternatively, the application can mask the “IN token received when FIFO empty” interrupt when clearing a global IN NAK handshake.</p>	R_SS_WC	1'b0
21	Incomplete Isochronous OUT Transfer (incompISOOUT)	The Device mode, the core sets this interrupt to indicate that there is at least one isochronous OUT endpoint on which the transfer is not completed in the current microframe. This interrupt is asserted along with the End of Periodic Frame Interrupt (EOPF) bit in this register.	R_SS_WC	1'b0
20	Incomplete Isochronous IN Transfer (incompISOIN)	The core sets this interrupt to indicate that there is at least one isochronous IN endpoint on which the transfer is not completed in the current microframe. This interrupt is asserted along with the End of Periodic Frame Interrupt (EOPF) bit in this register.	R_SS_WC	1'b0

Bits	Field Name	Field Description	R/W	Reset
19	OUT Endpoints Interrupt (OEPInt)	The core sets this bit to indicate that an interrupt is pending on one of the OUT endpoints of the core (in Device mode). The application must read the Device All Endpoints Interrupt (UDC_DAINT) register to determine the exact number of the OUT endpoint on which the interrupt occurred, and then read the corresponding Device OUT Endpoint-n Interrupt (UDC_DOEPINTn) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding UDC_DOEPINTn register to clear this bit.	RO	1'b0
18	IN Endpoints Interrupt (IEPInt)	The core sets this bit to indicate that an interrupt is pending on one of the IN endpoints of the core (in Device mode). The application must read the Device All Endpoints Interrupt (UDC_DAINT) register to determine the exact number of the IN endpoint on which the interrupt occurred, and then read the corresponding Device IN Endpoint-n Interrupt (UDC_DIEPINTn) register to determine the exact cause of the interrupt. The application must clear the appropriate status bit in the corresponding UDC_DIEPINTn register to clear this bit.	RO	1'b0
17:16	Reserved			
15	End of Periodic Frame Interrupt (EOPF)	Indicates that the period specified in the Periodic Frame Interval field of the Device Configuration register (UDC_DCFG.PerFrInt) has been reached in the current microframe.	R_SS_WC	1'b0
14	Isochronous OUT Packet Dropped Interrupt (ISOOutDrop)	The core sets this bit when it fails to write an isochronous OUT packet into the RxFIFO because the RxFIFO doesn't have enough space to accommodate a maximum packet size packet for the isochronous OUT endpoint.	R_SS_WC	1'b0
13	Enumeration Done (EnumDone)	The core sets this bit to indicate that speed enumeration is complete. The application must read the Device Status (UDC_DSTS) register to obtain the enumerated speed.	R_SS_WC	1'b0
12	USB Reset (USBRst)	The core sets this bit to indicate that a reset is detected on the USB.	R_SS_WC	1'b0
11	USB Suspend (USBSusp)	The core sets this bit to indicate that a suspend was detected on the USB. The core enters the Suspended state when there is no activity on the phy_line_state_i signal for an extended period of time.	R_SS_WC	1'b0
10	Early Suspend (ErlySusp)	The core sets this bit to indicate that an Idle state has been detected on the USB for 3 ms.	R_SS_WC	1'b0
9:8	Reserved			1'b0
7	Global OUT NAK Effective (GOUTNakEff)	Indicates that the Set Global OUT NAK bit in the Device Control register (UDC_DCTL.SGOUTNak), set by the application, has taken effect in the core. This bit can be cleared by writing the Clear Global OUT NAK bit in the Device Control register (UDC_DCTL.CGOUTNak).	RO	1'b0
6	Global IN Non-periodic NAK Effective (GINNakEff)	Indicates that the Set Global Non-periodic IN NAK bit in the Device Control register (UDC_DCTL.SGNPInNak), set by the application, has taken effect in the core. That is, the core has sampled the Global IN NAK bit set by the application. This bit can be cleared by clearing the Clear Global Non-periodic IN NAK bit in the Device Control register (UDC_DCTL.CGNPInNak). This interrupt does not necessarily mean that a NAK handshake is sent out on the USB. The STALL bit takes precedence over the NAK bit.	RO	1'b0
5	Reserved			1'b0

Bits	Field Name	Field Description	R/W	Reset
4	RxFIFO Non-Empty (RxFlvl)	Indicates that there is at least one packet pending to be read from the RxFIFO.	RO	1'b0
3	Start of (micro)Frame (Sof)	In Device mode, in the core sets this bit to indicate that an SOF token has been received on the USB. The application can read the Device Status register to get the current (micro)frame number. This interrupt is seen only when the core is operating at either HS or FS.	R_SS_WC	1'b0
2:1	Reserved			1'b0
0	Current Mode of Operation (CurMod)	Indicates the current mode of operation. • 1'b0: Device mode	RO	1'b0

18.9.5 UDC Interrupt Mask Register (UDC_GINTMSK)

This register works with the Core Interrupt register to interrupt the application. When an interrupt bit is masked, the interrupt associated with that bit is not generated. However, the Core Interrupt (UDC_GINTSTS) register bit corresponding to that interrupt is still set.

- Mask interrupt: 1'b0
- Unmask interrupt: 1'b1

USB Register Address Offset: 018

Bits	Field Name	Field Description	R/W	Reset
31	Resume/Remote Wakeup Detected Interrupt Mask (WkUpIntMsk)		RW	1'b0
30	Session Request/New Session Detected Interrupt Mask (SessReqIntMsk)		RW	1'b0
29	Disconnect Detected Interrupt Mask(DisconnectIntMsk)		RW	1'b0
28:23	<i>Reserved</i>			
22	Data Fetch Suspended Mask (FetSuspMsk)		RW	1'b0
21	Incomplete Isochronous OUT Transfer Mask (incomplISOOUTMsk)		RW	1'b0
20	Incomplete Isochronous IN Transfer Mask(incomplISOINMsk)		RW	1'b0
19	OUTEndpoints Interrupt Mask (OEPIntMsk)		RW	1'b0
18	IN Endpoints Interrupt Mask (INEPIntMsk)		RW	1'b0
17:16	<i>Reserved</i>			
15	End of Periodic Frame Interrupt Mask (EOPFMsk)		RW	1'b0
14	Isochronous OUT Packet Dropped Interrupt Mask (ISOOutDropMsk)		RW	1'b0
13	Enumeration Done Mask (EnumDoneMsk)		RW	1'b0
12	USB Reset Mask (USBRstMsk)		RW	1'b0
11	USB Suspend Mask (USBSuspMsk)		RW	1'b0
10	Early Suspend Mask (ErlySuspMsk)		RW	1'b0
9:8	<i>Reserved</i>			
7	Global OUT NAK Effective Mask (GOUTNakEffMsk)		RW	1'b0
6	Global Non-periodic IN NAK Effective Mask (GINNakEffMsk)		RW	1'b0
5	<i>Reserved</i>			
4	Receive FIFO Non-Empty Mask (RxFLvIMsk)		RW	1'b0
3	Start of (micro)Frame Mask (SofMsk)		RW	1'b0
2:0	<i>Reserved</i>			

18.9.6 UDC Receive Status Read/Pop Register (Read Only) (UDC_GRXSTSP)

A read to the Receive Status Debug Read register returns the contents of the top of the Receive FIFO. A read to the Receive Status Read and Pop register additionally pops the top data entry out of the Rx FIFO. The receive status contents must be interpreted differently in Host and Device modes. The core ignores the receive status pop/read when the receive FIFO is empty and returns a value of 32'h0000_0000. The application must only pop the Receive Status FIFO when the Receive FIFO Non-Empty bit of the Core Interrupt register (UDC_GINTSTS.RxFLvl) is asserted.

USB Register Address Offset for read: 01Ch

USB Register Address Offset for write: 020h

Bits	Field Name	Field Description	R/W	Reset
31:25:	Reserved			7'h0
24:21:	Frame Number (FN)	This is the least significant 4 bits of the (micro) frame number in which the packet is received on the USB. This field is supported only when isochronous OUT endpoints are supported.	RO	4'h0
20:17	Packet Status (PktSts)	Indicates the status of the received packet <ul style="list-style-type: none"> • 4'b0001: Global OUT NAK (triggers an interrupt) • 4'b0010: OUT data packet received • 4'b0011: OUT transfer completed (triggers an interrupt) • 4'b0100: SETUP transaction completed (triggers an interrupt) • 4'b0110: SETUP data packet received • Others: Reserved 	RO	4'h0
16:15	Data PID (DPID)	Indicates the Data PID of the received OUT data packet <ul style="list-style-type: none"> • 2'b00: DATA0 • 2'b10: DATA1 • 2'b01: DATA2 • 2'b11: MDATA 	RO	2'b0
14:4	Byte Count (BCnt)	Indicates the byte count of the received data packet.	RO	11'h0
3:0	Endpoint Number (EPNum)	Indicates the endpoint number to which the current received packet belongs.	RO	4'h0

18.9.7 UDC Receive FIFO Size Register (UDC_GRXFSIZ)

The application can program the RAM size that must be allocated to the RxFIFO.

USB Register Address Offset: 024h

Bits	Field Name	Field Description	R/W	Reset
31:16	Reserved			16'h0
15:0	RxFIFO Depth	This value is in terms of 32-bit words. • Minimum value is 16 • Maximum value is 32,768	RW	16'h600

18.9.8 UDC Non-periodic Transmit FIFO Size Register (UDC_GNPTXFSIZ)

The application can program the RAM size and the memory start address for the Non-periodic TxFIFO.

USB Register Address Offset: 028h

Bits	Field Name	Field Description	R/W	Reset
31:16	IN Endpoint TxFIFO 0 Depth (INEPTx0Dep)	This value is in terms of 32-bit words. Minimum value is 16 Maximum value is 1536	RW	16'h600
15:0	IN Endpoint FIFO0 Transmit RAM Start Address (INEPTx0StAddr)	This field contains the memory start address for IN Endpoint Transmit FIFO# 0.	RW	16'h600

18.9.9 UDC User ID Register (UDC_GUID)

This is a read/write register containing the User ID. It can be used in the following ways:

- To store the version or revision of your system
- To store hardware configurations that are outside the USB device controller
- As a scratch register

USB Register Address Offset: 03Ch

Bits	Field Name	Field Description	R/W	Reset
31:0	User ID (UserID)	Application-programmable ID field.	RW	32'h12345678

18.9.10 UDC ID Register (Read Only) (UDC_GSNPSID)

USB Register Address Offset: 040h

Bits	Field Name	Field Description	R/W	Reset
31:0	User ID (UserIDR)	User ID	RO	32'h4F54260A

18.9.11 UDC User HW Config1 Register (Read Only) (UDC_GHWCFG1)

This register contains the logical endpoint direction(s).

USB Register Address Offset: 044

Bits	Field Name	Field Description	R/W	Reset
31:0	Endpoint Direction (epdir)	<p>Two bits per endpoint represent the direction.</p> <ul style="list-style-type: none"> • 2'b00: BIDIR (IN and OUT) endpoint • 2'b01: IN endpoint • 2'b10: OUT endpoint • 2'b11: Reserved <p>Bits [31:30]: Endpoint 15 direction Bits [29:28]: Endpoint 14 direction ... Bits [3:2]: Endpoint 1 direction Bits[1:0]: Endpoint 0 direction (always BIDIR)</p>	RO	32'h0

18.9.12 UDC User HW Config2 Register (Read Only) (UDC_GHWCFG2)

USB Register Address Offset: 048h

Bits	Field Name	Field Description	R/W	Reset
31:30	Reserved			2'b0
29:26	(TknQDepth)	Range: 0–30	RO	4'h8
25:24	Host Mode Periodic Request Queue Depth (PTxQDepth)	<ul style="list-style-type: none"> 2'b00: 2 2'b01: 4 2'b10: 8 Others: Reserved 	RO	2'b10
23:22	Non-periodic Request Queue Depth (NPTxQDepth)	<ul style="list-style-type: none"> 2'b00: 2 2'b01: 4 2'b10: 8 Others: Reserved 	RO	2'b10
21:20	Reserved			2'b0
19	Dynamic FIFO Sizing Enabled (DynFifoSizing)	<ul style="list-style-type: none"> 1'b0: No 1'b1: Yes 	RO	1'b1
18	Periodic OUT Channels Supported in Host Mode (PerioSupport)	<ul style="list-style-type: none"> 1'b0: No 1'b1: Yes 	RO	1'b0
17:14	Reserved			
13:10	Number of Device Endpoints (NumDevEps)	Indicates the number of device endpoints supported by the core in Device mode in addition to control endpoint 0. The range of this field is 1–15.	RO	4'h6
9:8	Full-Speed PHY Interface Type (FSPhyType)	<ul style="list-style-type: none"> 2'b00: Full-speed interface not supported 2'b01: Dedicated full-speed interface 2'b10: FS pins shared with UTMI+ pins 2'b11: FS pins shared with ULPI pins 	RO	2'b00
7:6	High-Speed PHY Interface Type (HSPhyType)	<ul style="list-style-type: none"> 2'b00: High-Speed interface not supported 2'b01: UTMI+ 2'b10: ULPI 2'b11: UTMI+ and ULPI 	RO	2'b01
5	Point-to-Point (SingPnt)	<ul style="list-style-type: none"> 1'b0: Multi-point application 1'b1: Single-point application 	RO	1'b0
4:3	Architecture	<ul style="list-style-type: none"> 2'b00: Slave-Only 2'b01: External DMA 2'b10: Internal DMA Others: Reserved 	RO	2'b10
2:0	Mode of Operation	<ul style="list-style-type: none"> 3'b101: Device Others: Reserved 	RO	3'b100

18.9.13 UDC User HW Config3 Register (Read Only) (UDC_GHWCFG3)**USB Register Address Offset: 04Ch**

Bits	Field Name	Field Description	R/W	Reset
31:16	DFIFO Depth (DfifoDepth)	This value is in terms of 32-bit words. <ul style="list-style-type: none">• Minimum value is 32• Maximum value is 32,768	RO	16'h600
15:13	<i>Reserved</i>			
12	AHB and PHY Synchronous (AhbPhySync)	Indicates whether AHB and PHY clocks are synchronous to each other. <ul style="list-style-type: none">• 1'b0: No• 1'b1: Yes This bit is tied to 1.	RO	1'b0
11	Reset Style for Clocked always Blocks in RTL (RstType)	<ul style="list-style-type: none">• 1'b0: Asynchronous reset is used in the core• 1'b1: Synchronous reset is used in the core	RO	1'b1
10	<i>Reserved</i>			
9	Vendor Control Interface Support	<ul style="list-style-type: none">• 1'b0: Vendor Control Interface is not available on the core.• 1'b1: Vendor Control Interface is available.	RO	1'b0
8	I2C Selection	<ul style="list-style-type: none">• 1'b0: I2C Interface is not available on the core.• 1'b1: I2C Interface is available on the core.	RO	1'b0
7	<i>Reserved</i>			
6:4	Width of Packet Size Counters (PktSizeWidth)	<ul style="list-style-type: none">• 3'b000: 4 bits• 3'b001: 5 bits• 3'b010: 6 bits• 3'b011: 7 bits• 3'b100: 8 bits• 3'b101: 9 bits• 3'b110: 10 bits• Others: Reserved	RO	3'b110
3:0	Width of Transfer Size Counters (XferSizeWidth)	<ul style="list-style-type: none">• 4'b0000: 11 bits• 4'b0001: 12 bits....• 4'b1000: 19 bits• Others: Reserved	RO	4'b1000

18.9.14 UDC User HW Config4 Register (Read Only) (UDC_GHWCFG4)

USB Register Address Offset: 050h

Bits	Field Name	Field Description	R/W	Reset
31:30	Reserved			2'h0
29:26	Number of Device Mode IN Endpoints Including Control Endpoints (INEps)	Range 0 -15 • 0: 1 IN Endpoint • 1: 2 IN Endpoints • 15 : 16 IN Endpoints	RO	4'b0100
25	DedFifoMode	• 1'b0: Dedicated Transmit FIFO Operation not enabled. • 1'b1: Dedicated Transmit FIFO Operation enabled.	RO	1'b1
24	"session_end" Filter Enabled (SessEndFltr)	• 1'b0: No filter • 1'b1: Filter	RO	1'b1
23	"b_valid" Filter Enabled (BValidFltr)	• 1'b0: No filter • 1'b1: Filter	RO	1'b1
22	"a_valid" Filter Enabled (AValidFltr)	• 1'b0: No filter • 1'b1: Filter	RO	1'b1
21	"vbus_valid" Filter Enabled (VBusValidFltr)	• 1'b0: No filter • 1'b1: Filter	RO	1'b1
20	"iddg" Filter Enable (IddgFltr)	• 1'b0: No filter • 1'b1: Filter	RO	1'b1
19:16	Number of Device Mode Control Endpoints in Addition to Endpoint 0 (NumCtlEps)	Range: 1-15	RO	0
15:14	UTMI+ PHY/ULPI-to-Internal UTMI+ Wrapper Data Width (PhyDataWidth)	When a ULPI PHY is used, an internal wrapper converts ULPI to UTMI+. • 2'b00: 8 bits • 2'b01: 16 bits • 2'b10: 8/16 bits, software selectable • Others: Reserved	RO	2'b10
13:6	Reserved			8'h0
5	Minimum AHB Frequency Less Than 60 MHz (AhbFreq)	• 1'b0: No • 1'b1: Yes	RO	1'b1
4	Enable Power Optimization? (EnablePwrOpt)	• 1'b0: No • 1'b1: Yes	RO	1'b1
3:0	Number of Device Mode Periodic IN Endpoints (NumDevPerioEps)	Range: 0-15	RO	0

18.9.15 UDC Device IN Endpoint Transmit FIFO-n Size Register (UDC_DPTXFSIZn)

This register holds the size and memory start address of IN endpoint TxFIFOs implemented in Device mode. Each FIFO holds the data for one IN endpoint. This register is repeated for instantiated IN endpoint FIFOs 1 to 3. For IN endpoint FIFO 0 use UDC_GNPTXFSIZ register for programming the size and memory start address.

USB Register Address Offset: FIFO_number: $1 \leq n \leq 3$

Offset: $104h + (\text{FIFO_number}-1) * 04h$

Bits	Field Name	Field Description	R/W	Reset
31:16	Device Periodic TxFIFO Size (DPTxFSIZE)	This value is in terms of 32-bit words. <ul style="list-style-type: none">• Minimum value is 4• Maximum value is 1536	RW	16'h600
15:0	IN Endpoint FIFOn Transmit RAM Start Address (INEPnTxFSAddr)	This field contains the memory start address for IN endpoint Transmit FIFOn ($0 < n \leq 3$).	RW	n=1: 16'hC00 n=2: 16'h1200 n=3: 16'h1800

18.9.16 UDC Configuration C (UDC_DCFG)

This register configures the core in Device mode after power-on or after certain control commands or enumeration. Do not make changes to this register after initial programming.

USB Register Address Offset: 800h

Bits	Field Name	Field Description	R/W	Reset
31:13	Reserved			
12:11	Periodic Frame Interval (PerFrInt)	Indicates the time within a (micro)frame at which the application must be notified using the End Of Periodic Frame Interrupt. This can be used to determine if all the isochronous traffic for that (micro)frame is complete. <ul style="list-style-type: none">• 2'b00: 80% of the (micro)frame interval• 2'b01: 85%• 2'b10: 90%• 2'b11: 95%	RW	2'h0
10:4	Device Address (DevAddr)	The application must program this field after every SetAddress control command.	RW	7'h0
3	Reserved			1'b0
2	Non-Zero-Length Status OUT Handshake (NZStsOUTHShk)	The application can use this field to select the handshake the core sends on receiving a nonzero-length data packet during the OUT transaction of a control transfer's Status stage. <ul style="list-style-type: none">• 1'b1: Send a STALL handshake on a nonzero-length status OUT transaction and do not send the received OUT packet to the application.• 1'b0: Send the received OUT packet to the application (zero-length or non-zero-length) and send a handshake based on the NAK and STALL bits for the endpoint in the Device Endpoint Control register.	RW	1'b0
1:0	Device Speed (DevSpd)	Indicates the speed at which the application requires the core to enumerate, or the maximum speed the application can support. However, the actual bus speed is determined only after the chirp sequence is completed, and is based on the speed of the USB host to which the core is connected. <ul style="list-style-type: none">• 2'b00: High speed (USB 2.0 PHY clock is 30 MHz or 60 MHz)• 2'b01: Full speed (USB 2.0 PHY clock is 30 MHz or 60 MHz)• 2'b10: Low speed (USB 1.1 transceiver clock is 6 MHz). If you select 6 MHz LS mode, you must do a soft reset.• 2'b11: Full speed (USB 1.1 transceiver clock is 48 MHz)	RW	2'b0

18.9.17 UDC Control Register (UDC_DCTL)**USB Register Address Offset: 804h**

Bits	Field Name	Field Description	R/W	Reset
31:12	Reserved			20'h0
11	Power-On Programming Done (PWROnPrgDone)	The application uses this bit to indicate that register programming is completed after a wake-up from Power Down mode.	RW	1'b0
10	Clear Global OUT NAK (CGOUTNak)	A write to this field clears the Global OUT NAK.	WO	1'b0
9	Set Global OUT NAK (SGOUTNak)	A write to this field sets the Global OUT NAK. The application uses this bit to send a NAK handshake on all OUT endpoints. The application must set this bit only after making sure that the Global OUT NAK Effective bit in the Core Interrupt Register (UDC_GINTSTS.GOUTNakEff) is cleared.	WO	1'b0
8	Clear Global Non-periodic IN NAK (CGNPIInNak)	A write to this field clears the Global Non-periodic IN NAK.	WO	1'b0
7	Set Global Non-periodic IN NAK (SGNPIInNak)	A write to this field sets the Global Non-periodic IN NAK. The application uses this bit to send a NAK handshake on all non-periodic IN endpoints. The core can also set this bit when a timeout condition is detected on a non-periodic endpoint in shared FIFO operation. The application must set this bit only after making sure that the Global IN NAK Effective bit in the Core Interrupt Register (UDC_GINTSTS.GINNakEff) is cleared.	WO	1'b0
6:4	Test Control (TstCtl)	<ul style="list-style-type: none"> • 3'b000: Test mode disabled • 3'b001: Test_J mode • 3'b010: Test_K mode • 3'b011: Test_SE0_NAK mode • 3'b100: Test_Packet mode • 3'b101: Test_Force_Enable • Others: Reserved 	RW	3'b0
3	Global OUT NAK Status (GOUTNakSts)	<ul style="list-style-type: none"> • 1'b0: A handshake is sent based on the FIFO Status and the NAK and STALL bit settings. • 1'b1: No data is written to the RxFIFO, irrespective of space availability. Sends a NAK handshake on all packets, except on SETUP transactions. All isochronous OUT packets are dropped. 	RO	1'b0
2	Global Non-periodic IN NAK Status (GNPINNakSts)	<ul style="list-style-type: none"> • 1'b0: A handshake is sent out based on the data availability in the transmit FIFO. • 1'b1: A NAK handshake is sent out on all non-periodic IN endpoints, irrespective of the data availability in the transmit FIFO. 	RO	1'b0

Bits	Field Name	Field Description	R/W	Reset
1	Soft Disconnect (SftDiscon)	<p>The application uses this bit to signal the controller to do a soft disconnect. As long as this bit is set, the host does not see that the device is connected, and the device does not receive signals on the USB. The core stays in the disconnected state until the application clears this bit.</p> <p>The minimum duration for which the core must keep this bit set is specified in Table 18-11.</p> <ul style="list-style-type: none"> • 1'b0: Normal operation. When this bit is cleared after a soft disconnect, the core drives the phy_opmode_o signal on the TMI+ to 2'b00, which generates a device connect event to the USB host. When the device is reconnected, the USB host restarts device enumeration. • 1'b1: The core drives the phy_opmode_o signal on the UTMI+ to 2'b01, which generates a device disconnect event to the USB host. 	RW	1'b0
0	Remote Wakeup Signaling (RmtWkUpSig)	When the application sets this bit, the core initiates remote signaling to wake up the USB host. The application must set this bit to instruct the core to exit the Suspend state. As specified in the USB 2.0 specification, the application must clear this bit 1–15 ms after setting it.	RW	1'b0

Table 18-11. Minimum Duration for Soft Disconnect

Operating Speed	Device State	Minimum Duration
High speed	Suspended	1 ms + 2.5 μ s
High speed	Idle	3 ms + 2.5 μ s
High speed	Not Idle or Suspended (Performing transactions)	125 μ s
Full speed/Low speed	Suspended	1 ms + 2.5 μ s
Full speed/Low speed	Idle	2.5 μ s
Full speed/Low speed	Not Idle or Suspended (Performing transactions)	2.5 μ s

18.9.18 UDC Status Register (Read Only) (UDC_DSTS)

This register indicates the status of the core with respect to USB-related events. It must be read on interrupts from Device All Interrupts (UDC_DAIINT) register.

USB Register Address Offset: 808h

Bits	Field Name	Field Description	R/W	Reset
31:22	Reserved			10'b0
21:8	Frame or Microframe Number of the Received SOF (SOFFN)	When the core is operating at high speed, this field contains a microframe number. When the core is operating at full or low speed, this field contains a frame number.	RO	14'b0
7:4	Reserved		RO	4'b0
3	Erratic Error (ErrticErr)	The core sets this bit to report any erratic errors (phy_rxvalid_i/phy_rxvldh_i or phy_rxactive_i is asserted for at least 2 ms, due to PHY error) seen on the UTMI+. Due to erratic errors, the controller goes into Suspended state and an interrupt is generated to the application with Early Suspend bit of the Core Interrupt register (UDC_GINTSTS.ErlySusp). If the early suspend is asserted due to an erratic error, the application can only perform a soft disconnect recover.	RO	1'b0
2:1	Enumerated Speed (EnumSpd)	Indicates the speed at which the controller has come up after speed detection through a chirp sequence. <ul style="list-style-type: none"> • 2'b00: High speed (PHY clock is running at 30 or 60 MHz) • 2'b01: Full speed (PHY clock is running at 30 or 60 MHz) • 2'b10: Low speed (PHY clock is running at 6 MHz) • 2'b11: Full speed (PHY clock is running at 48 MHz) Low speed is not supported for devices using a UTMI+ PHY.	RO	2'b0
0	Suspend Status (SuspSts)	In Device mode, this bit is set as long as a Suspend condition is detected on the USB. The core enters the Suspended state when there is no activity on the phy_line_state_i signal for an extended period of time. The core comes out of the suspend: <ul style="list-style-type: none"> • When there is any activity on the phy_line_state_i signal • When the application writes to the Remote Wakeup Signaling bit in the Device Control register (DCTL.RmtWkUpSig). 	RO	1'b0

18.9.19 UDC Device IN Endpoint Common Interrupt Mask Register (UDC_DIEPMSK)

This register works with each of the Device IN Endpoint Interrupt (UDC_DIEPINTn) registers for all endpoints to generate an interrupt per IN endpoint. The IN endpoint interrupt for a specific status in the UDC_DIEPINTn register can be masked by writing to the corresponding bit in this register. Status bits are masked by default.

USB Register Address Offset: 810h

Bits	Field Name	Field Description	R/W	Reset
31:9	Reserved	Reserved		23'h0
8	TxfifoUndrnMsk	FIFO Underrun Mask	RW	1'b0
7	Reserved	Reserved	RW	1'b0
6	INEPNakEffMsk	IN Endpoint NAK Effective Mask	RW	1'b0
5	INTknEPMisMsk	IN Token received with EP Mismatch Mask	RW	1'b0
4	INTknTXFEmpMsk	IN Token Received When TxFIFO Empty Mask	RW	1'b0
3	TimeOUTMsk	Timeout Condition Mask (Non-isochronous endpoints)	RW	1'b0
2	AHBErrMsk	AHB Error Mask	RW	1'b0
1	EPDisbldMsk	Endpoint Disabled Interrupt Mask	RW	1'b0
0	XferComplMsk	Transfer Completed Interrupt Mask	RW	1'b0

18.9.20 UDC Device OUT Endpoint Common Interrupt Mask (UDC_DOEPMSK)

This register works with each of the Device OUT Endpoint Interrupt (UDC_DOEPINTn) registers for all endpoints to generate an interrupt per OUT endpoint. The OUT endpoint interrupt for a specific status in the UDC_DOEPINTn register can be masked by writing into the corresponding bit in this register. Status bits are masked by default.

- Mask interrupt: 1'b0
- Unmask interrupt: 1'b1

USB Register Address Offset: 814h

Bits	Field Name	Field Description	R/W	Reset
31:09	Reserved			23'h0
8	OutPktErrMsk	OUT Packet Error Mask	RW	1'b0
7	Reserved			1'b0
6	Back2BackSETup	Back-to-Back SETUP Packets Received Mask (Applies to control OUT endpoints only.)	RW	1'b0
5	Reserved			1'b0
4	OUTTknEPdisMsk	OUT Token Received when Endpoint Disabled Mask (Applies to control OUT endpoints only.)	RW	1'b0
3	SetUPMsk	SETUP Phase Done Mask (Applies to control endpoints only.)	RW	1'b0
2	AHBErrMsk	AHB Error	RW	1'b0
1	EPDisbldMsk	Endpoint Disabled Interrupt Mask	RW	1'b0
0	XferComplMsk	Transfer Completed Interrupt Mask	RW	1'b0

18.9.21 UDC Device All Endpoints Interrupt Register (UDC_DAINT)

When a significant event occurs on an endpoint, a Device All Endpoints Interrupt register interrupts the application using the Device OUT Endpoints Interrupt bit or Device IN Endpoints Interrupt bit of the Core Interrupt register (UDC_GINTSTS.OEPInt or UDC_GINTSTS.IEPInt, respectively). This is shown in [Figure 18-4](#). There is one interrupt bit per endpoint, up to a maximum of 16 bits for OUT endpoints and 16 bits for IN endpoints. For a bidirectional endpoint, the corresponding IN and OUT interrupt bits are used. Bits in this register are set and cleared when the application sets and clears bits in the corresponding Device Endpoint-n Interrupt register (UDC_DIEPINTn/UDC_DOEPINTn).

USB Register Address Offset: 818h

Bits	Field Name	Field Description	R/W	Reset
31:16	OUT EP Interrupt Mask Bits (OutEpMsk)	One per OUT Endpoint: Bit 16 for OUT EP 0, bit 31 for OUT EP 15	RO	16'h0
15:0	IN EP Interrupt Mask Bits (InEpMsk)	One bit per IN Endpoint: Bit 0 for IN EP 0, bit 15 for IN EP 15	RO	16'h0

18.9.22 UDC Device All Endpoints Interrupt Mask Register (UDC_DAINTMSK)

The Device Endpoint Interrupt Mask register works with the Device Endpoint Interrupt register to interrupt the application when an event occurs on a device endpoint. However, the Device All Endpoints Interrupt (UDC_DAINT) register bit corresponding to that interrupt is still set.

- Mask Interrupt: 1'b0
- Unmask Interrupt: 1'b1

USB Register Address Offset: 81Ch

Bits	Field Name	Field Description	R/W	Reset
31:16	OUT EP Interrupt Mask Bits (OutEpMsk)	One per OUT Endpoint: Bit 16 for OUT EP 0, bit 31 for OUT EP 15	RW	16'h0
15:0	IN EP Interrupt Mask Bits (InEpMsk)	One bit per IN Endpoint: Bit 0 for IN EP 0, bit 15 for IN EP 15	RW	16'h0

18.9.23 Device Threshold Control Register (UDC_DTKNQR3)

USB Register Address Offset: 830h

Bits	Field Name	Field Description	R/W	Reset
31:28	Reserved		RW	4'b0
27	Arbiter Parking Enable (ArbPrkEn)	This bit controls internal DMA arbiter parking for IN endpoints. When thresholding is enabled and this bit is set to one, then the arbiter parks on the IN endpoint for which there is a token received on the USB. This is done to avoid getting into underrun conditions. By default the parking is enabled.	RW	
26			RW	1'b1
25:17	Receive Threshold Length (RxThrLen)	This field specifies Receive thresholding size in DWORDS. This field also specifies the amount of data received on the USB before the core can start transmitting on the AHB. The threshold length has to be at least eight DWORDS. The recommended value for ThrLen is to be the same as the programmed AHB Burst Length (UDC_GAHBCFG.HBstLen).	RW	9'h8
16	Receive Threshold Enable (RxThrEn)	When this bit is set, the core enables thresholding in the receive direction.	RW	1'b0
15:11			RW	5'b0
10:2	Transmit Threshold Length (TxThrLen)	This field specifies Transmit thresholding size in DWORDS. This field specifies the amount of data in bytes to be in the corresponding endpoint transmit FIFO, before the core can start transmit on the USB. The threshold length has to be at least eight DWORDS. This field controls both isochronous and non-isochronous IN endpoint thresholds. The recommended value for ThrLen is to be the same as the programmed AHB Burst Length (UDC_GAHBCFG.HBstLen).	RW	9'h8
1	ISO IN Endpoints Threshold Enable. (ISOThrEn)	When this bit is set, the core enables thresholding for isochronous IN endpoints.	RW	1'b0
0	Non-ISO IN Endpoints Threshold Enable. (NonISOThrEn)	When this bit is set, the core enables thresholding for Non Isochronous IN endpoints.	RW	1'b0

18.9.24 Device IN Endpoint FIFO Empty Interrupt Mask Register (UDC_DTKNQR4)

This register is used to control the IN endpoint FIFO empty interrupt generation (UDC_DIEPINTn.TxfEmp).

- Mask interrupt: 1'b0
- Unmask interrupt: 1'b1

USB Register Address Offset: 834h

Bits	Field Name	Field Description	R/W	Reset
31:16	Reserved		RO	16'h0
15:0	IN EP Tx FIFO Empty Interrupt Mask Bits (InEpTxfEmpMsk)	These bits acts as mask bits for UDC_DIEPINTn. TxFEmp interrupt One bit per IN Endpoint: Bit 0 for IN EP 0, bit 15 for IN EP 15	RW	16'h0

Device Logical IN Endpoint-Specific Registers

One set of endpoint registers is instantiated per logical endpoint. A logical endpoint is unidirectional: it can be either IN or OUT. To represent a bidirectional endpoint, two logical endpoints are required, one for the IN direction and the other for the OUT direction. This is also true for control endpoints. The registers and register fields described in this section can pertain to IN or OUT endpoints, or both, or specific endpoint types as noted.

18.9.25 Device Control IN Endpoint 0 Control Register (UDC_DIEPCTLn)

This section describes the Control IN Endpoint 0 Control register. Nonzero control endpoints use registers for endpoints 1–15.

USB Register Address Offset: 900h

Bits	Field Name	Field Description	R/W	Reset
31	Endpoint Enable (EPEna)	Indicates that data is ready to be transmitted on the endpoint. The core clears this bit before setting any of the following interrupts on this endpoint: <ul style="list-style-type: none"> • Endpoint Disabled • Transfer Completed 	R_WS_SC	1'b0
30	Endpoint Disable (EPDis)	The application sets this bit to stop transmitting data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled Interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint.	R_WS_SC	1'b0
29:28	Reserved			2'b0
27	Set NAK (SNAK)	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for an endpoint after a SETUP packet is received on that endpoint.	WO	1'b0
26	Clear NAK (CNAK)	A write to this bit clears the NAK bit for the endpoint.	WO	1'b0
25:22	TxFIFO Number (TxNum)	<ul style="list-style-type: none"> • For Shared FIFO operation, this value is always set to '0', indicating that control IN endpoint 0 data is always written in the Non-Periodic Transmit FIFO. • For Dedicated FIFO operation, this value is set to the FIFO number that is assigned to IN Endpoint 0. 	RW	4'h0

Bits	Field Name	Field Description	R/W	Reset
21	STALL Handshake (Stall)	The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority.	R_WS_SC	1'b0
20	Reserved			1'b0
19:18	Endpoint Type (EPType)	Hardcoded to '00' for control.	RO	2'h0
17	NAK Status (NAKsts)	<p>Indicates the following:</p> <ul style="list-style-type: none"> • 1'b0: The core is transmitting non-NAK handshakes based on the FIFO status • 1'b1: The core is transmitting NAK handshakes on this endpoint. <p>When this bit is set, either by the application or core, the core stops transmitting data, even if there is data available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.</p>	RO	1'b0
16	Reserved			1'b0
15	USB Active Endpoint (USBActEP)	This bit is always set to 1, indicating that control endpoint '0' is always active in all configurations and interfaces.	RO	1'b1
14:11	Next Endpoint (NextEp)	<p>Applies to non-periodic IN endpoints only.</p> <p>Indicates the endpoint number to be fetched after the data for the current endpoint is fetched. The core can access this field, even when the Endpoint Enable (EPEna) bit is not set. This field is not valid in Slave mode.</p> <p>Note: This field is valid only for Shared FIFO operations.</p>	RW	4'b0
10:2	Reserved			9'h0
1:0	Maximum Packet Size (MPS)	<p>Applies to IN and OUT endpoints.</p> <p>The application must program this field with the maximum packet size for the current logical endpoint.</p> <ul style="list-style-type: none"> • 2'b00: 64 bytes • 2'b01: 32 bytes • 2'b10: 16 bytes • 2'b11: 8 bytes 	RW	2'h0

18.9.26 Device Control OUT Endpoint 0 Control Register (UDC_DOEPCTL0)

This section describes the Control OUT Endpoint 0 Control register. Nonzero control endpoints use registers for endpoints 1–15.

USB Register Address Offset: B00h

Bits	Field Name	Field Description	Reset	R/W
31	Endpoint Enable (EPEna)	Indicates that the application has allocated the memory to start receiving data from the USB. The core clears this bit before setting any of the following interrupts on this endpoint: <ul style="list-style-type: none">• SETUP Phase Done• Endpoint Disabled• Transfer Completed Note: In DMA mode, this bit must be set for the core to transfer SETUP data packets into memory.	R_WS_SC	1'b0
30	Endpoint Disable (EPDis)	The application cannot disable control OUT endpoint 0.	RO	1'b0
29:28	Reserved			2'b0
27	Set NAK (SNAK)	A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set bit on a Transfer Completed interrupt, or after a SETUP is received on the endpoint.	WO	1'b0
26	Clear NAK (CNAK)	A write to this bit clears the NAK bit for the endpoint.	WO	1'b0
25:22	Reserved			4'h0
21	STALL Handshake (Stall)	The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.	R_WS_SC	1'b0
20	Snoop Mode (Snp)	This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.	RW	1'b0
19:18	Endpoint Type (EPTType)	Hardcoded to 2'b00 for control.	RO	2'h0
17	NAK Status (NAKSts)	Indicates the following: <ul style="list-style-type: none">• 1'b0: The core is transmitting non-NAK handshakes based on the FIFO status.• 1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit, the core stops receiving data, even if there is space in the RxFIFO to accommodate the incoming packet. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.	RO	1'b0
16	Reserved			1'b0
15	USB Active Endpoint (USBActEP)	This bit is always set to 1, indicating that a control endpoint 0 is always active in all configurations and interfaces.	RO	1'b1

Bits	Field Name	Field Description	Reset	R/W
14:2	Reserved			13'h0
1:0	Maximum Packet Size (MPS)	The maximum packet size for control OUT endpoint 0 is the same as what is programmed in control IN Endpoint 0. <ul style="list-style-type: none"> • 2'b00: 64 bytes • 2'b01: 32 bytes • 2'b10: 16 bytes • 2'b11: 8 bytes 	RO	2'h0

18.9.27 Device Endpoint-n Control Register

The application uses this register to control the behavior of each logical endpoint other than endpoint 0.

USB Register Address Offset: Endpoint_number: $1 \leq n \leq 6$

Offset for IN endpoints: 900h + (Endpoint_number * 20h)

Offset for OUT endpoints: B00h + (Endpoint_number * 20h)

Bits	Field Name	Field Description	R/W	Reset
31	Endpoint Enable (EPEna)	Applies to IN and OUT endpoints. For IN endpoints, this bit indicates that data is ready to be transmitted on the endpoint. For OUT endpoints, this bit indicates that the application has allocated the memory to start receiving data from the USB. The core clears this bit before setting any of the following interrupts on this endpoint: <ul style="list-style-type: none"> • SETUP Phase Done (OUT only) • Endpoint Disabled • Transfer Completed Note: For control OUT endpoints in DMA mode, this bit must be set to be able to transfer SETUP data packets in memory.	R_WS_SC	1'b0
30	Endpoint Disable (EPDis)	Applies to IN and OUT endpoints. The application sets this bit to stop transmitting/receiving data on an endpoint, even before the transfer for that endpoint is complete. The application must wait for the Endpoint Disabled interrupt before treating the endpoint as disabled. The core clears this bit before setting the Endpoint Disabled interrupt. The application must set this bit only if Endpoint Enable is already set for this endpoint.	R_WS_SC	1'b0
29	Set DATA1 PID (SetD1PID)	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA1. Set Odd (micro)frame (SetOddFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to odd (micro)frame.	WO	1'b0
28	Set DATA0 PID (SetD0PID)	Applies to interrupt/bulk IN and OUT endpoints only. Writing to this field sets the Endpoint Data PID (DPID) field in this register to DATA0. Set Even (micro)frame (SetEvenFr) Applies to isochronous IN and OUT endpoints only. Writing to this field sets the Even/Odd (micro)frame (EO_FrNum) field to even (micro) frame.	WO	1'b0

Bits	Field Name	Field Description	R/W	Reset
27	Set NAK (SNAK)	Applies to IN and OUT endpoints. A write to this bit sets the NAK bit for the endpoint. Using this bit, the application can control the transmission of NAK handshakes on an endpoint. The core can also set this bit for OUT endpoints on a Transfer Completed interrupt, or after a SETUP is received on the endpoint.	WO	1'b0
26	Clear NAK (CNAK)	Applies to IN and OUT endpoints. A write to this bit clears the NAK bit for the endpoint.	WO	1'b0
25:22	TxFIFO Number (TxFNum)	These bits specify the FIFO number associated with this endpoint. Each active IN endpoint should be programmed to a separate FIFO number.	RW	4'h0
21	STALL Handshake (Stall)	Applies to non-control, non-isochronous IN and OUT endpoints only. The application sets this bit to stall all tokens from the USB host to this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Only the application can clear this bit, never the core. Applies to control endpoints only. The application can only set this bit, and the core clears it, when a SETUP token is received for this endpoint. If a NAK bit, Global Non-periodic IN NAK, or Global OUT NAK is set along with this bit, the STALL bit takes priority. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.	RW	1'b0
20	Snoop Mode (Snp)	Applies to OUT endpoints only. This bit configures the endpoint to Snoop mode. In Snoop mode, the core does not check the correctness of OUT packets before transferring them to application memory.	R_WS_SC	1'b0
19:18	Endpoint Type (EPType)	Applies to IN and OUT endpoints. This is the transfer type supported by this logical endpoint. <ul style="list-style-type: none"> • 2'b00: Control • 2'b01: Isochronous • 2'b10: Bulk • 2'b11: Interrupt 	RW	2'h0
17	NAK Status (NAKSts)	Applies to IN and OUT endpoints. Indicates the following: <ul style="list-style-type: none"> • 1'b0: The core is transmitting non-NAK handshakes based on the FIFO status. • 1'b1: The core is transmitting NAK handshakes on this endpoint. When either the application or the core sets this bit: <ul style="list-style-type: none"> • The core stops receiving any data on an OUT endpoint, even if there is space in the RxFIFO to accommodate the incoming packet. • For non-isochronous IN endpoints: The core stops transmitting any data on an IN endpoint, even if there data is available in the TxFIFO. • For isochronous IN endpoints: The core sends out a zero-length data packet, even if there data is available in the TxFIFO. Irrespective of this bit's setting, the core always responds to SETUP data packets with an ACK handshake.	RO	1'b0

Bits	Field Name	Field Description	R/W	Reset
16	Endpoint Data PID (DPID)	<p>Applies to interrupt/bulk IN and OUT endpoints only.</p> <p>Contains the PID of the packet to be received or transmitted on this endpoint. The application must program the PID of the first packet to be received or transmitted on this endpoint, after the endpoint is activated. Applications use the SetD1PID and SetD0PID fields of this register to program either DATA0 or DATA1 PID.</p> <ul style="list-style-type: none"> • 1'b0: DATA0 • 1'b1: DATA1 <p>Even/Odd (Micro)Frame (EO_FrNum)</p> <p>Applies to isochronous IN and OUT endpoints only.</p> <p>Indicates the (micro)frame number in which the core transmits/receives isochronous data for this endpoint. The application must program the even/odd (micro) frame number in which it intends to transmit/receive isochronous data for this endpoint using the SetEvnFr and SetOddFr fields in this register.</p> <ul style="list-style-type: none"> • 1'b0: Even (micro)frame • 1'b1: Odd (micro)frame 	RO	1'b0
15	USB Active Endpoint (USBActEP)	<p>Applies to IN and OUT endpoints.</p> <p>Indicates whether this endpoint is active in the current configuration and interface. The core clears this bit for all endpoints (other than EP 0) after detecting a USB reset. After receiving the SetConfiguration and SetInterface commands, the application must program endpoint registers accordingly and set this bit.</p>	R_WS_SC	1'b0
14:11	Next Endpoint (NextEp)	<p>Applies to non-periodic IN endpoints only.</p> <p>Indicates the endpoint number to be fetched after the data for the current endpoint is fetched. The core can access this field, even when the Endpoint Enable (EPEna) bit is low. This field is not valid in Slave mode operation.</p> <p>Note: This field is valid only for Shared FIFO operations.</p>	RW	4'b0
10:0	Maximum Packet Size (MPS)	<p>Applies to IN and OUT endpoints.</p> <p>The application must program this field with the maximum packet size for the current logical endpoint. This value is in bytes.</p>	RW	11'h0

18.9.28 Device Endpoint-n Interrupt Register (UDC_DIEPINTn)

This register indicates the status of an endpoint with respect to USB- and AHB-related events. It is shown in Figure 5-2. The application must read this register when the OUT Endpoints Interrupt bit or IN Endpoints Interrupt bit of the Core Interrupt register (UDC_GINTSTS.OEPInt or UDC_GINTSTS.IEPInt, respectively) is set. Before the application can read this register, it must first read the Device All Endpoints Interrupt (UDC_DAINT) register to get the exact endpoint number for the Device Endpoint-n Interrupt register. The application must clear the appropriate bit in this register to clear the corresponding bits in the UDC_DAINT and UDC_GINTSTS registers.

USB Register Address Offset: Endpoint_number: $0 \leq n \leq 6$

Offset for IN endpoints: $908h + (\text{Endpoint_number} * 20h)$

Offset for OUT endpoints: $B08h + (\text{Endpoint_number} * 20h)$

Bits	Field Name	Field Description	R/W	Reset
31:9	Reserved			23'h0
8	Fifo Underrun (TxfifoUndrn)	Applies to IN endpoints Only Core generates this interrupt when it sees a transmit FIFO underrun condition for this endpoint.	RW	1'b0
	OUT Packet Error (OutPktErr)	Applies to OUT endpoints Only This interrupt is asserted when the core sees an overflow or a CRC error for non-ISOC OUT packet.		
7	Transmit FIFO Empty (TxFEmp)	This bit is valid only for IN Endpoints This interrupt is asserted when the TxFIFO for this endpoint is either half or completely empty. The half or completely empty status is determined by the TxFIFO Empty Level bit in the Core AHB Configuration register (UDC_GAHBCFG.NPTxFEmpLvl).	RW	1'b0
6	IN Endpoint NAK Effective (INEPNakEff)	Applies to periodic IN endpoints only. Indicates that the IN endpoint NAK bit set by the application has taken effect in the core. This bit can be cleared when the application clears the IN endpoint NAK by writing to UDC_DIEPCTLn.CNAK. This interrupt indicates that the core has sampled the NAK bit set (either by the application or by the core). This interrupt does not necessarily mean that a NAK handshake is sent on the USB. A STALL bit takes priority over a NAK bit.	RO	1'b0
	Back-to-Back SETUP Packets Received (Back2BackSETup)	Applies to Control OUT endpoints only. This bit indicates that the core has received more than three back-to-back SETUP packets for this particular endpoint. For information about handling this interrupt, see Section 6.5.2.1.3.	RW	1'b0
5	IN Token Received with EP Mismatch (INTknEPMis)	Applies to non-periodic IN endpoints only. Indicates that the data in the top of the non-periodic TxFIFO belongs to an endpoint other than the one for which the IN token was received. This interrupt is asserted on the endpoint for which the IN token was received. For OUT endpoints, this bit is Reserved	R_SS_WC	1'b0

Bits	Field Name	Field Description	R/W	Reset
4	IN Token Received When TxFIFO is Empty (INTknTxFEmp)	Applies to non-periodic IN endpoints only. Indicates that an IN token was received when the associated TxFIFO (periodic/non-periodic) was empty. This interrupt is asserted on the endpoint for which the IN token was received.	R_SS_WC	1'b0
	OUT Token Received When Endpoint Disabled (OUTTknEPdis)	Applies only to control OUT endpoints. Indicates that an OUT token was received when the endpoint was not yet enabled. This interrupt is asserted on the endpoint for which the OUT token was received.		
3	Timeout Condition (TimeOUT)	Applies to non-isochronous IN endpoints in shared FIFO operation only. Indicates that the core has detected a timeout condition on the USB for the last IN token on this endpoint. SETUP Phase Done (SetUp) Applies to control OUT endpoints only. Indicates that the SETUP phase for the control endpoint is complete and no more back-to-back SETUP packets were received for the current control transfer. On this interrupt, the application can decode the received SETUP data packet.	R_SS_WC	1'b0
2	AHB Error (AHBErr)	Applies to IN and OUT endpoints. This is generated only in Internal DMA mode when there is an AHB error during an AHB read/write. The application can read the corresponding endpoint DMA address register to get the error address.	R_SS_WC	1'b0
1	Endpoint Disabled Interrupt (EPDisbld)	Applies to IN and OUT endpoints. This bit indicates that the endpoint is disabled per the application's request.	R_SS_WC	1'b0
0	Transfer Completed Interrupt (XferCompl)	Applies to IN and OUT endpoints. Indicates that the programmed transfer is complete on the AHB as well as on the USB, for this endpoint.	R_SS_WC	1'b0

18.9.29 Device Endpoint 0 Transfer Size (UDC_DIEPTSIZ0/ UDC_DOEPTSIZ0)

Endpoint Enable bit of the Device Control Endpoint 0 Control registers (DIEPCTL0.EPEna/ UDC_DOEPCCTL0.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

Nonzero endpoints use the registers for endpoints 1–15.

USB Register Address Offset: Offset for IN endpoints: 910h

Offset for OUT endpoints: B10h

Table 18-12. Device IN Endpoint 0 Transfer Size Register: UDC_DIEPTSIZ0

Bits	Field Name	Field Description	R/W	Reset
31:20	Reserved			12'h0
19	Packet Count (PktCnt)	Indicates the total number of USB packets that constitute the Transfer Size amount of data for endpoint 0. This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.	RW	1'b0
18:7	Reserved			12'h0
6:0	Transfer Size (XferSize)	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet from the external memory is written to the TxFIFO.	RW	7'h0

Table 18-13. Device OUT Endpoint 0 Transfer Size Register: DOEPTSIZ0

Bits	Field Name	Field Description	R/W	Reset
31	Reserved			1'b0
30:29	SETUP Packet Count (SUPCnt)	This field specifies the number of back-to-back SETUP data packets the endpoint can receive. <ul style="list-style-type: none"> • 2'b01: 1 packet • 2'b10: 2 packets • 2'b11: 3 packets 	RW	2'h0
28:20	Reserved			9'h0
19	Packet Count (PktCnt)	This field is decremented to zero after a packet is written into the RxFIFO.	RW	1'b0
18:7	Reserved			12'h0
8:0	Transfer Size (XferSize)	Indicates the transfer size in bytes for endpoint 0. The core interrupts the application only after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	RW	7'h0

18.9.30 Device Endpoint-n Transfer Size Register (UDC_DIEPTSIZn)

The application must modify this register before enabling the endpoint. Once the endpoint is enabled using Endpoint Enable bit of the Device Endpoint-n Control registers (UDC_DIEPCTLn.EPEna/ UDC_DOEPCTLn.EPEna), the core modifies this register. The application can only read this register once the core has cleared the Endpoint Enable bit.

This register is used only for endpoints other than Endpoint 0.

USB Register Address Offset: Endpoint_number: $1 \leq n \leq 6$

Offset for IN endpoints: $910h + (\text{Endpoint_number} * 20h)$

Offset for OUT endpoints: $B10h + (\text{Endpoint_number} * 20h)$

Bits	Field Name	Field Description	R/W	Reset
31	Reserved			1'b0
30:29	Multi Count (MC)	<p>Applies to IN endpoints only. For periodic IN endpoints, this field indicates the number of packets that must be transmitted per microframe on the USB. The core uses this field to calculate the data PID for isochronous IN endpoints.</p> <ul style="list-style-type: none"> • 2'b01: 1 packet • 2'b10: 2 packets • 2'b11: 3 packets <p>For non-periodic IN endpoints, this field is valid only in Internal DMA mode. It specifies the number of packets the core should fetch for an IN endpoint before it switches to the endpoint pointed to by the Next Endpoint field of the Device Endpoint-n Control register (UDC_DIEPCTLn.NextEp).</p>	RW	2'b0
	Received Data PID (RxDPID)	<p>Applies to isochronous OUT endpoints only. This is the data PID received in the last packet for this endpoint.</p> <ul style="list-style-type: none"> • 2'b00: DATA0 • 2'b01: DATA1 • 2'b10: DATA2 • 2'b11: MDATA 	RO	
	SETUP Packet Count (SUPCn)	<p>Applies to control OUT Endpoints only. This field specifies the number of back-to-back SETUP data packets the endpoint can receive.</p> <ul style="list-style-type: none"> • 2'b01: 1 packet • 2'b10: 2 packets • 2'b11: 3 packets 	RW	

Bits	Field Name	Field Description	R/W	Reset
28:19	Packet Count (PktCnt)	Indicates the total number of USB packets that constitute the Transfer Size amount of data for this endpoint. <ul style="list-style-type: none">• IN Endpoints: This field is decremented every time a packet (maximum size or short packet) is read from the TxFIFO.• OUT Endpoints: This field is decremented every time a packet (maximum size or short packet) is written to the RxFIFO.	RW	10'hA
18:0	Transfer Size (XferSize)	This field contains the transfer size in bytes for the current endpoint. The core only interrupts the application after it has exhausted the transfer size amount of data. The transfer size can be set to the maximum packet size of the endpoint, to be interrupted at the end of each packet. <ul style="list-style-type: none">• IN Endpoints: The core decrements this field every time a packet from the external memory is written to the TxFIFO.• OUT Endpoints: The core decrements this field every time a packet is read from the RxFIFO and written to the external memory.	RW	19'h13

18.9.31 Device Endpoint-n DMA Address Register (UDC_DIEPDMAAn)

USB Register Address Offset: Endpoint_number: $0 \leq n \leq 3$

Offset for IN endpoints: 914h + (Endpoint_number * 20h)

Offset for OUT endpoints: B14h + (Endpoint_number * 20h)

Bits	Field Name	Field Description	R/W	Reset
31:0	DMA Address (DMAAddr)	Holds the start address of the external memory for storing or fetching endpoint data. This register is incremented on every AHB transaction. Note: For control endpoints, this address stores control OUT data packets as well as SETUP transaction data packets. If more than three SETUP packets are received back-to-back, the, the SETUP data packet in the memory is overwritten.	RW	32'h0

18.9.32 Device IN Endpoint Transmit FIFO Status Register (UDC_DTXFSTS n)

This read-only register contains the free space information for the Device IN endpoint TxFIFO.

USB Register Address Offset: Endpoint_number: $0 \leq n \leq 3$

Offset for IN endpoints: 918h + (Endpoint_number * 20h)

Bits	Field Name	Field Description	R/W	Reset
31:16	Reserved			1'b0
15:0	IN Endpoint TxFIFO Space Avail (INEPTxFSpAvail)	Indicates the amount of free space available in the Endpoint TxFIFO. Values are in terms of 32-bit words. <ul style="list-style-type: none">• 16'h0: Endpoint TxFIFO is full• 16'h1: 1 word available• 16'h2: 2 words available• 16'hn: n words available (where $0 \leq n \leq 1536$)• 16'h600: 1536 words available• Others: Reserved	RO	16'h600

18.9.33 Power and Clock Gating Control Register (UDC_PCGCR)

The application can use this register to control the core's power-down and clock gating features.

Because the CSR module is turned off during power-down, this register is implemented in the AHB Slave BIU module.

USB Register Address Offset: E00h

Bits	Field Name	Field Description	R/W	Reset
31:5	Reserved			27'h0
4	PHY Suspended. (PhySuspended)	Indicates that the PHY has been suspended. After the application sets the Stop Pclk bit[0], this bit is updated once the PHY is suspended. Because the UTMI+ PHY suspend is controlled through a port, the UTMI+PHY is suspended immediately after Stop Pclk is set. However, the ULPI PHY takes a few clocks to suspend, because the suspend information is conveyed through the ULPI protocol to the ULPI PHY.	RO	1'b0
3	Reset Power-Down Modules (RstPdwnModule)	This bit is valid only in Partial Power-Down mode. The application sets this bit when the power is turned off. The application clears this bit after the power is turned on and the PHY clock is up.	RW	1'b0
2	Reserved			
1	Gate Hclk (GateHclk)	The application sets this bit to gate hclk to modules other than the AHB Slave and Master and wakeup logic when the USB is suspended or the session is not valid. The application clears this bit when the USB is resumed or a new session starts.	RW	1'b0
0	Stop Pclk (StopPclk)	The application sets this bit to stop the PHY clock (phy_clk) when the USB is suspended, the session is not valid, or the device is disconnected. The application clears this bit when the USB is resumed or a new session starts.	RW	1'b0

18.10 Host Controller Registers

18.10.0.1 Accessing Registers for Interface in Host Mode

When the USB Interface is configured for Host Mode, registers are accessible as follows:

You may access the USB Interface's general control register block or registers in the Host controller blocks, including FIFOs, using address field bits as shown below.:

Table 18-14. Register Address Fields for Accessing Host Mode USB General Register

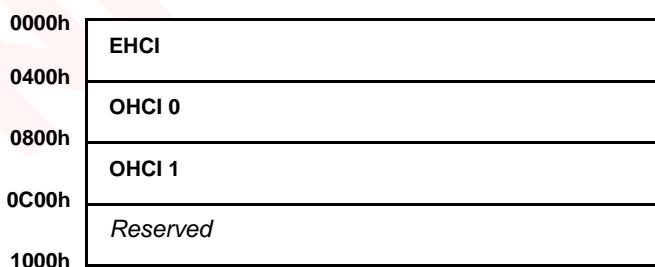
Field	Bits	Description
Reserved	39:13	Reserved
REGSPACE	12	Register block access select: 0: Host Controller registers 1: USB Interface General registers
REGADDR	11:0	Register address in controller

Table 18-15. Register Address Fields for Accessing Host Mode Controller Register

Field	Bits	Description
Reserved	39:13	Reserved
REGSPACE	12	Register block access select: 0: Host Controller registers 1: USB Interface General registers
HOSTSPACE	11:10	Host Controller Access Select: 00: Access EHCI Controller registers 01: Access OHCI 0 Controller registers 10: Access OHCI 1 Controller registers 11: do not use
REGADDR	9:0	Register address in controller

18.10.0.2 USB Host Controller Registers Addressing

Figure 18-15. USB Host Controller Address Block Map



18.11 EHCI Operational Registers (EOR)

The EHCI Capability registers and Operational registers are stored in this block. Refer to EHCI Specification, Revision 1.0 for more information on these registers.

Table 18-16. Capability Registers

Mnemonic	Register Name	Offset from EHCI AHB Slave Start Address ^a	Default Value
HCCAPBASE	Capability Register Length	USBBASE + 00h	32'h01000010
<i>Reserved</i>		USBBASE + 01h	N/A
HCIVERSION	Interface Version Number	USBBASE + 02h	32'h01000000
HCSPARAMS	Structural Parameters	USBBASE + 04h	32'h00002112
HCCPARAMS	Capability Parameters	USBBASE + 08h	32'h00000014 Note: The Isochronous Scheduling Threshold value is set to 1 by default. If Descriptor/Data Prefetch is selected, the value is set to 2
HCSP-PORTROUT	Companion Port Route Description	USBBASE + 0Ch	32'h00000000

a. USBBASE is fixed to the EHCI slave start address (offset = 'h0). USBASE = 0x00.

Table 18-17. Operational Registers

Mnemonic	Register Name	Offset From EHCI Slave Start Address ^a	Default Value
USBCMD	USB Command	USBOPBASE + 00h	32'h0080B000
USBSTS	USB Status	USBOPBASE + 04h	32'h00001000
USBINTR	USB Interrupt Enable	USBOPBASE + 08h	32'h00000000
FRINDEX	USB Frame Index	USBOPBASE + 0ch	32'h00000000
<i>Reserved</i>		USBOPBASE + 10h	
PERIODICLISTBASE	Periodic Frame List Base Address Register	USBOPBASE + 14h	undefined
ASYNCLISTADDR	Synchronous List Address	USBOPBASE + 18h	undefined
CONFIGFLAG	Configured Flag Register	USBOPBASE + 40h	32'h00000000
PORTROUT_1 to PORTROUT_2	Port Status/Control	USBOPBASE + 44h USBOPBASE + 48h	32'h00002000

a. USBOPBASE is fixed to the EHCI slave start address + 'h10 (offset = 'h10). USOBASE = 0x10.

18.12 OHCI PCI Configuration, and Operational Register Maps

Refer to EHCI Specification, Revision 1.0, for more information on these registers.

Table 18-18. Open-HCI-Related PCI Configuration Registers

Offset	Register	Description
05-04	COMMAND	Provides coarse control over a device's ability to generate and respond to PCI cycles
0B-09	CLASS_CODE	Identifies the generic function of the device
13-10	BAR_OHCI	Specifies the base address of a contiguous block in the main memory of the PC host, from which 4 KB of directly mapped addressing spaces are reserved by OpenHCl for the operational registers of the Host Controller

Table 18-19. OHCI Host Controller Operational Register Map

Address Offset	Register Name
0	HcRevision
4	HcControl
8	HcCommandStatus
C	HcInterruptStatus
10	HcInterruptEnable
14	HcInterruptDisable
18	HcHCCA
1C	HcPeriodCurrentED
20	HcControlHeadED
24	HcControlCurrentED
28	HcBulkHeadED
2C	HcBulkCurrentED
30	HcDoneHead
34	HcFmInterval
38	HcFmRemaining
3C	HcFmNumber
40	HcPeriodidStart
44	HcLSThreshold
48	HcRhDescriptorA
4C	HcRhDescriptorB
50	HcRhStatus
54	HcRhPortStatus1
58	HcRhPortStatus2

NETLOGIC
CONFIDENTIAL



Chapter 19 PCI Express™ Interface

19.1 Introduction

This chapter covers the PCI Express (PCIe) implementation in the XLS devices. It provides the functional description, relevant interface capabilities, and programming requirements. It is organized into the following sections:

- Overview of PCIe Interface Configurations
- Theory of Operation
 - Configuration Space Access
 - Extended Configuration Space Access
 - Accessing PCIe Memory Space
 - Accessing PCIe I/O Space
- DMA Transfers
- PCIe Interrupts
- Programming Model
 - Startup/Initialization
- Global PCI Express Interface Registers
- PCIe Configuration Registers: EP Mode
- PCIe Configuration Registers: RC Mode

19.1.1 PCIe Features

- Compliant to PCI Express Base Specification revision 1.1
- 2.5-Gigabits/second/Lane/direction of raw bandwidth
- Type0 configuration space in Endpoint (EP) mode; type 1 configuration space in Root Complex Port (RC) mode
- Maximum payload size of 256 bytes
- Automatic and application-initiated Lane reversal
- Polarity inversion on receive
- Supports ECRC generation and checking
- Supports relaxed ordering
- Supports PCI power management
- Supports PCIe Active-State Power Management (ASPM)
- Supports PCIe Advanced Error Reporting

19.2 Overview of PCIe Interface Configurations

The PCIe controller resources allocated for each XLS device are shown in [Table 19-1](#).

Table 19-1. PCI Express Resources for Each XLS Device

Configuration	XLS6xx	XLS4xx	XLS4xxLite	XLS208	XLS204	XLS108	XLS104
PCI-Express Controllers	4	4	2	4	4	2	2
PCIe lanes per Controller	1 ^a	1 ^a	1 ^b	1	1	1	1

a. Controller 0 can support one x4 lane configuration with the other controller(s) not used.

b. The XLS4xxLite devices can support either two-by-one lane link or one-by-four lanes link

19.2.1 XLS6xx and XLS4xx PCIe Controllers

The XLS6xx and XLS4xx PCIe interface is configurable. It consists of four PCIe controllers with one lane per controller where, optionally, Controller 0 can drive four lanes. All of the controllers can be programmed as RC (Root Complex) and the Controller 0 in each device can be programmed as EP (Endpoint).

The XLS6xx and XLS4xx devices support four Physical layers and four controllers, as shown in [Figure 19-1](#) and [Table 19-2](#). There are four main configurations:

- One x4 EP (Controller 0)
- One x4 RC (Controller 0)
- Four x1 RC (quad controllers)
- Three x1 RC, one x1 EP (quad controllers)

19.2.2 XLS408Lite and XLS404Lite PCIe Controllers

The XLS408Lite and XLS04A devices, have two PCIe controllers with one lane each, but, optionally, Controller_0 can drive four lanes. All of the controllers can be programmed as RC (Root Complex) and the Controller_0 in each device can be programmed as EP (Endpoint).

The XLS408Lite and XLS404Lite devices support two Physical layers distributed among two controllers as shown in [Figure 19-2](#) and [Table 19-3](#). There are four main configurations:

- One x4 EP (Controller 0)
- One x4 RC (Controller 0)
- One x1 EP and one x1 RC (dual controller)
- Two x1 RC (dual controller)

19.2.3 XLS2xx Controllers

The XLS2xx PCIe interface consists of four independent PCIe controllers with one lane each. Each controller can be programmed as RC (Root Complex) and the Controller_0 can be programmed as EP (Endpoint). [Figure 19-3](#) and [Table 19-4](#) illustrate the configurations. There are two configurations:

- Four x1 RC (quad controllers)
- Three x1 RC plus one x1 EP (quad controllers)

19.2.4 XLS1xx Controllers

The XLS1xx PCIe interface consists of two independent PCIe controllers with one lane each. Each controller can be programmed as RC (Root Complex) and the Controller_0 can be programmed as EP (Endpoint). [Figure 19-4](#) and [Table 19-5](#) illustrate the configurations. There are two configurations:

- xxTxx(dual controllers)
- One x1 RC plus one x1 EP (dual controllers)

19.3 Configuring the PCIe Interface

The PCIe controllers and lanes are configured at startup/reset time based on the logic levels of signal pins IO_AD[11:10], see [Section 19.7.1](#) for details. Bits[21:20] of the GPIO Reset Configuration register mirror the IO_AD[11:10] logic levels latched at reset. (See “[GPIO System Control Registers](#)” on page [1141](#). for more details)

Figure 19-1. Internal XLS6xx and XLS4xx PCIe 4x1 and 1x4 Topology

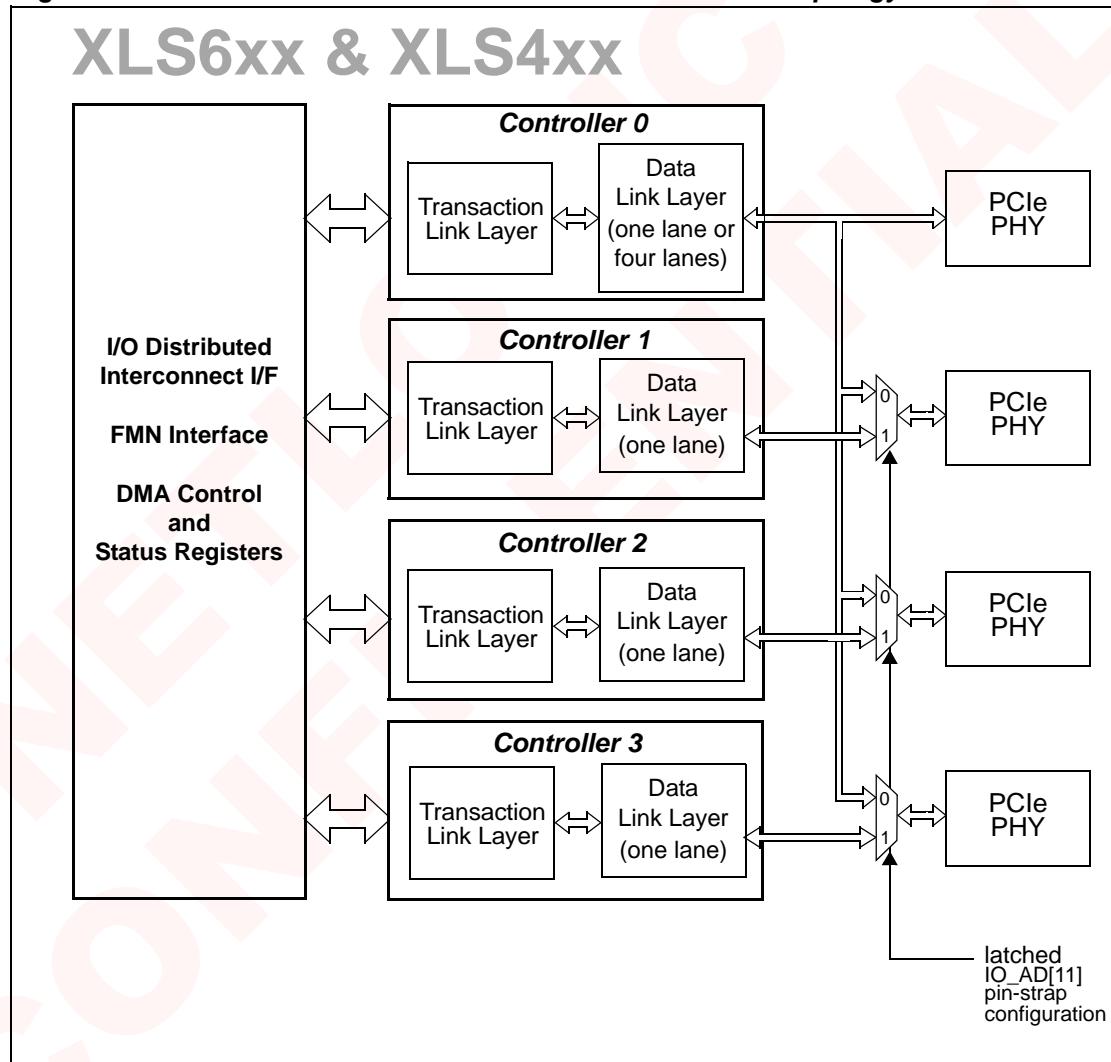


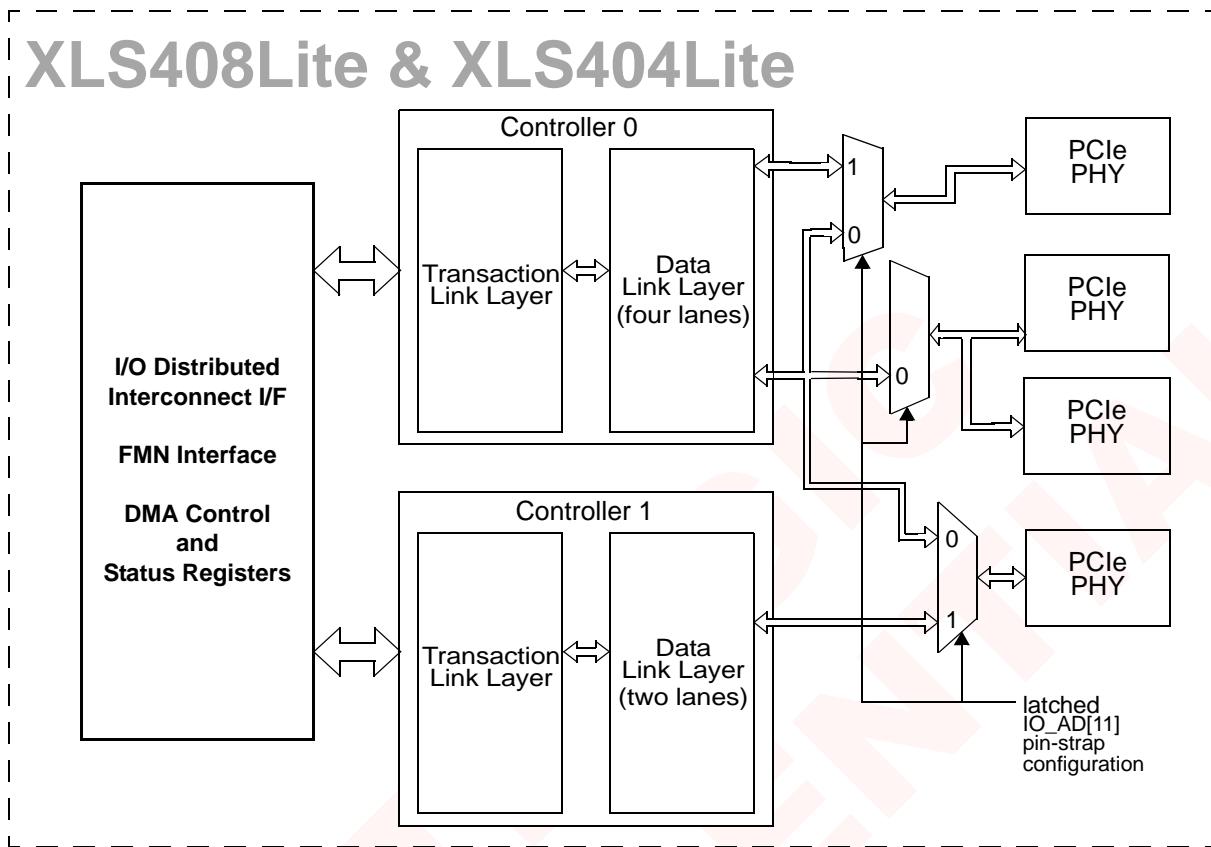
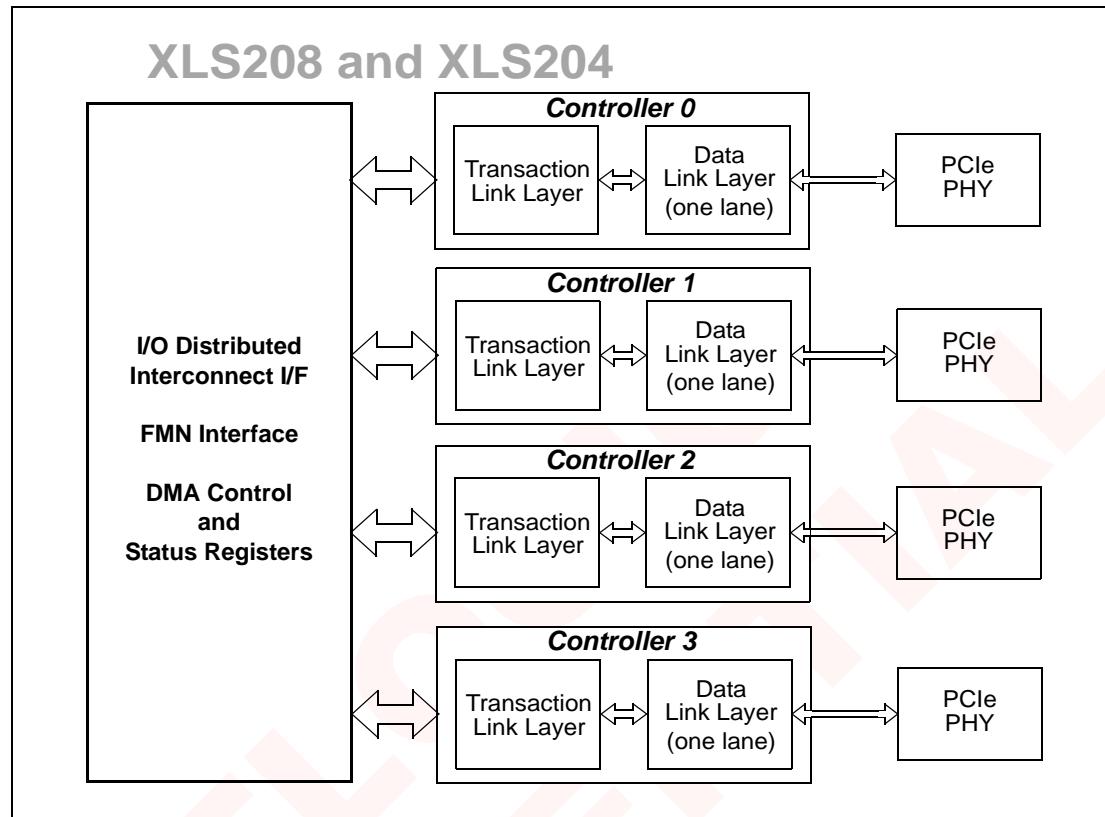
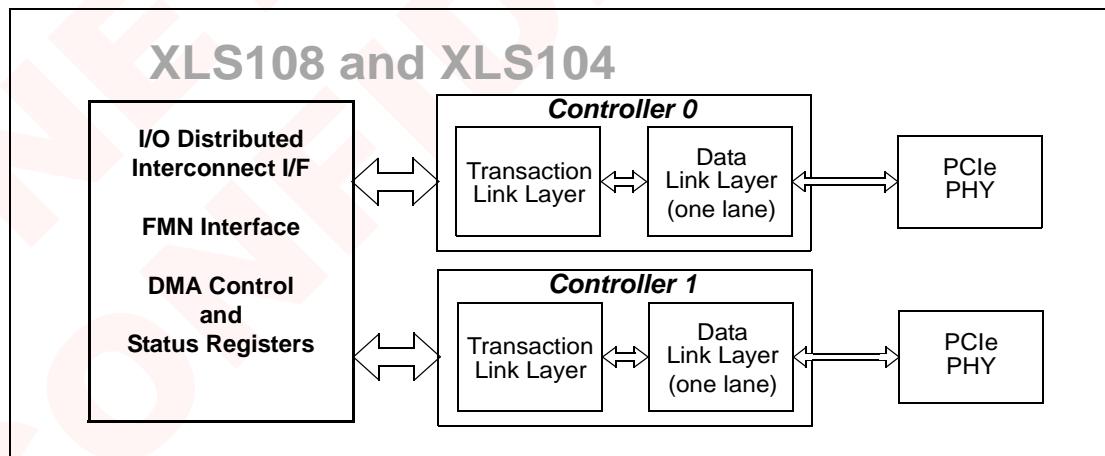
Figure 19-2. Internal XLS408Lite and XLS404Lite PCIe Topology

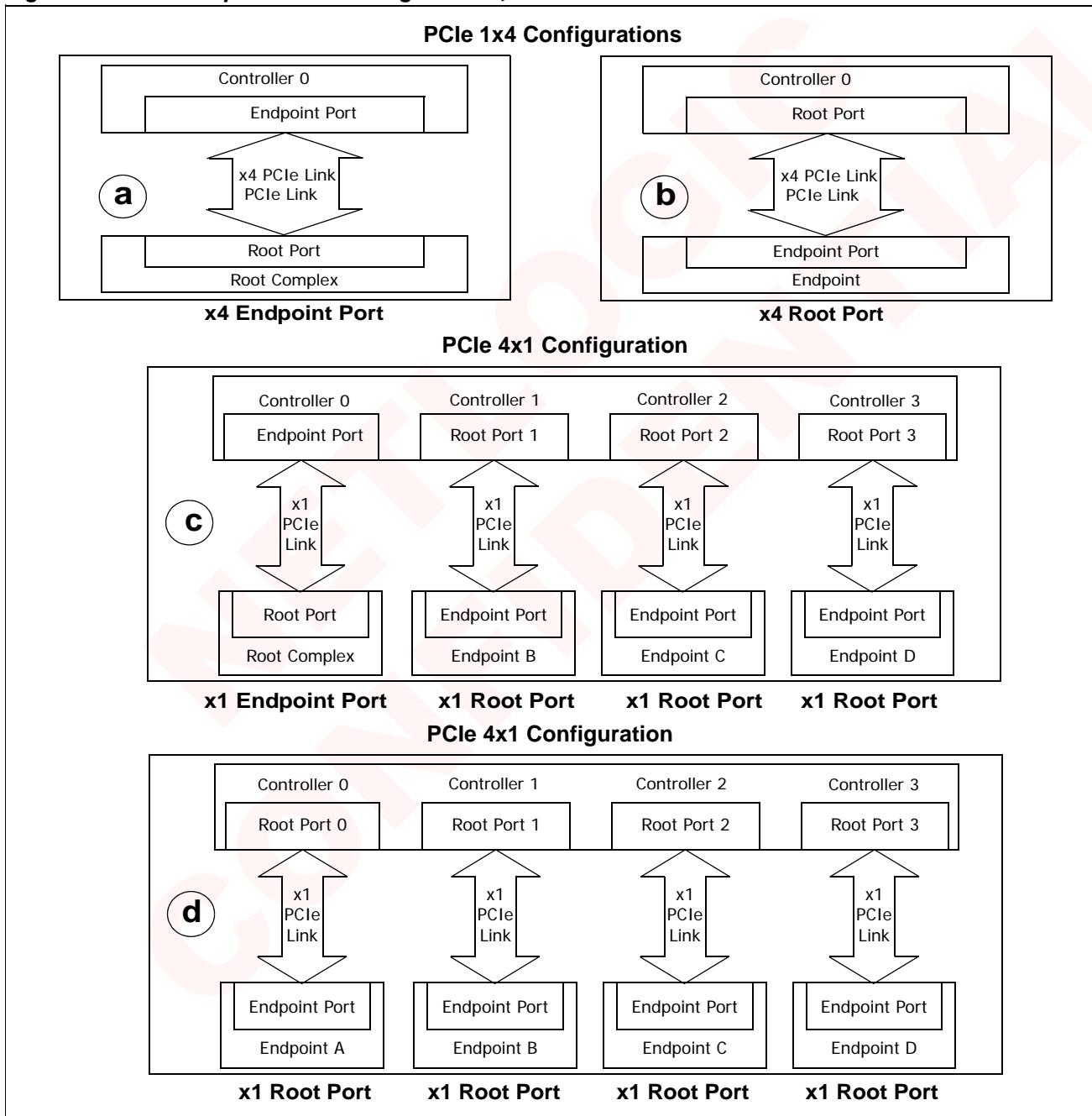
Figure 19-3. Internal XLS208 and XLS204 PCIe 4x1 Topology*Figure 19-4. Internal XLS108 and XLS104 PCIe 2x1 Topology*

19.3.1 Supported PCIe Configurations for XLS6xx and XLS4xx Devices

Table 19-2 and Figure 19-5 show PCI Express configurations supported by the XLS6xx and XLS4xx devices from an external connectivity perspective.

Table 19-2. PCI Express Link Configurations for XLS6xx and XLS4xx Devices

Strapping Pins		Diagram	PCIe Controller_0	PCIe Controller_1	PCIe Controller_2	PCIe Controller_3
IO_AD[11]	IO_AD[10]		Lane 0	Lane 1	Lane 2	Lane 3
Low	Low	(a)	x4 Endpoint Port	-	-	-
Low	High	(b)	x4 Root Port	-	-	-
High	Low	(c)	Endpoint Port	Root Port	Root Port	Root Port
High	High	(d)	Root Port	Root Port	Root Port	Root Port

Figure 19-5. PCI Express Link Configurations, XLS616 and XLS4xx Devices

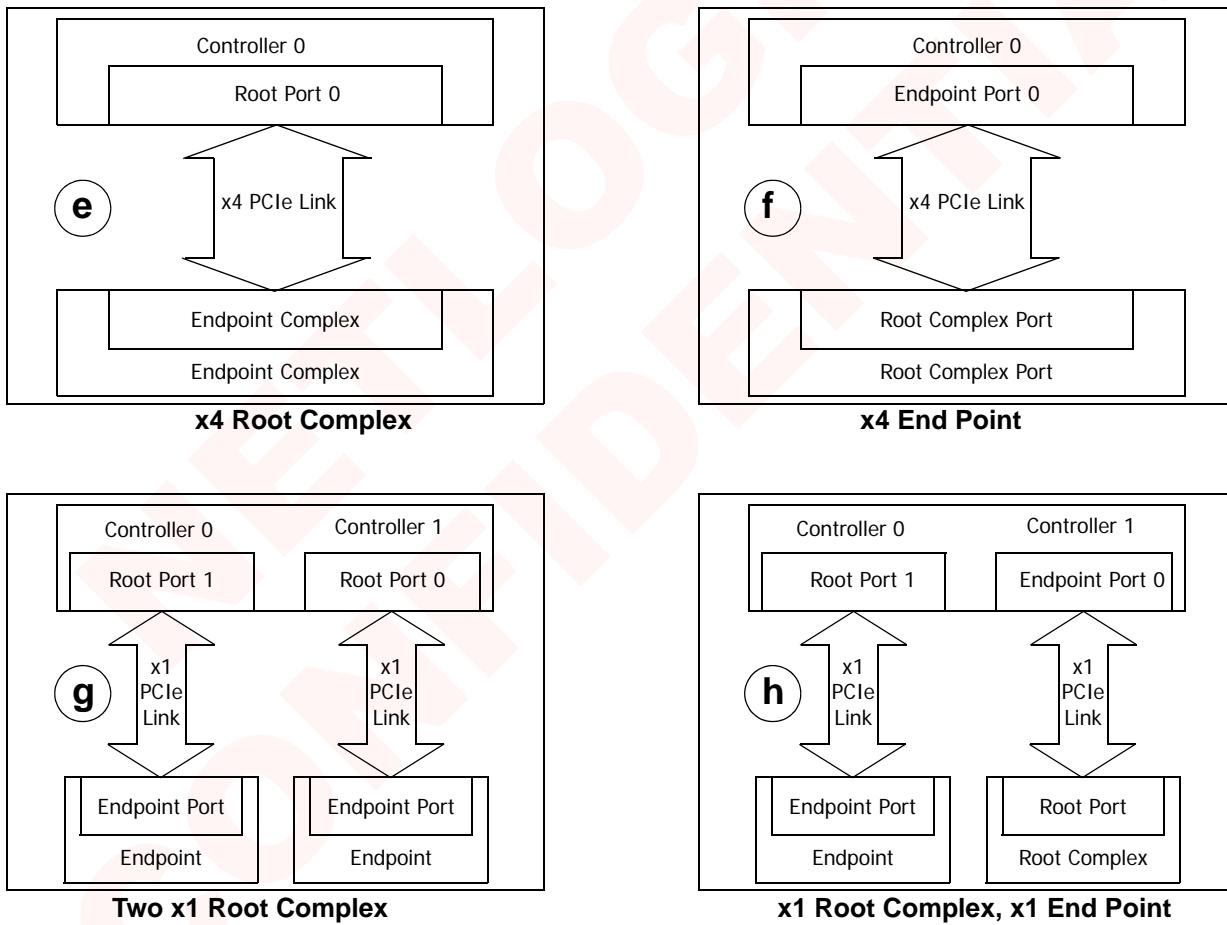
19.3.1.1 Supported PCIe Configurations for XLS408Lite and XLS04A Devices

Table 19-3 and Figure 19-6 show PCI Express configurations supported by the XLS408Lite and XLS404Lite devices from an external connectivity perspective.

Table 19-3. PCI Express Link Configurations, XLS408Lite and XLS404Lite

Strapping Pins		Diagram	PCIe Controller 0	PCIe Controller 1
IO_AD[11]	IO_AD[10]		Link 0	Link 1
Low	Low	(e)	x4 Endpoint Port	—
Low	High		x4 Root Port	
High	Low	(g)	x1 Endpoint Port	x1 Root Port
High	High	(h)	x1 Root Port	x1 Root Port

Figure 19-6. PCI Express Link Configurations, XLS408Lite and XLS404Lite Devices



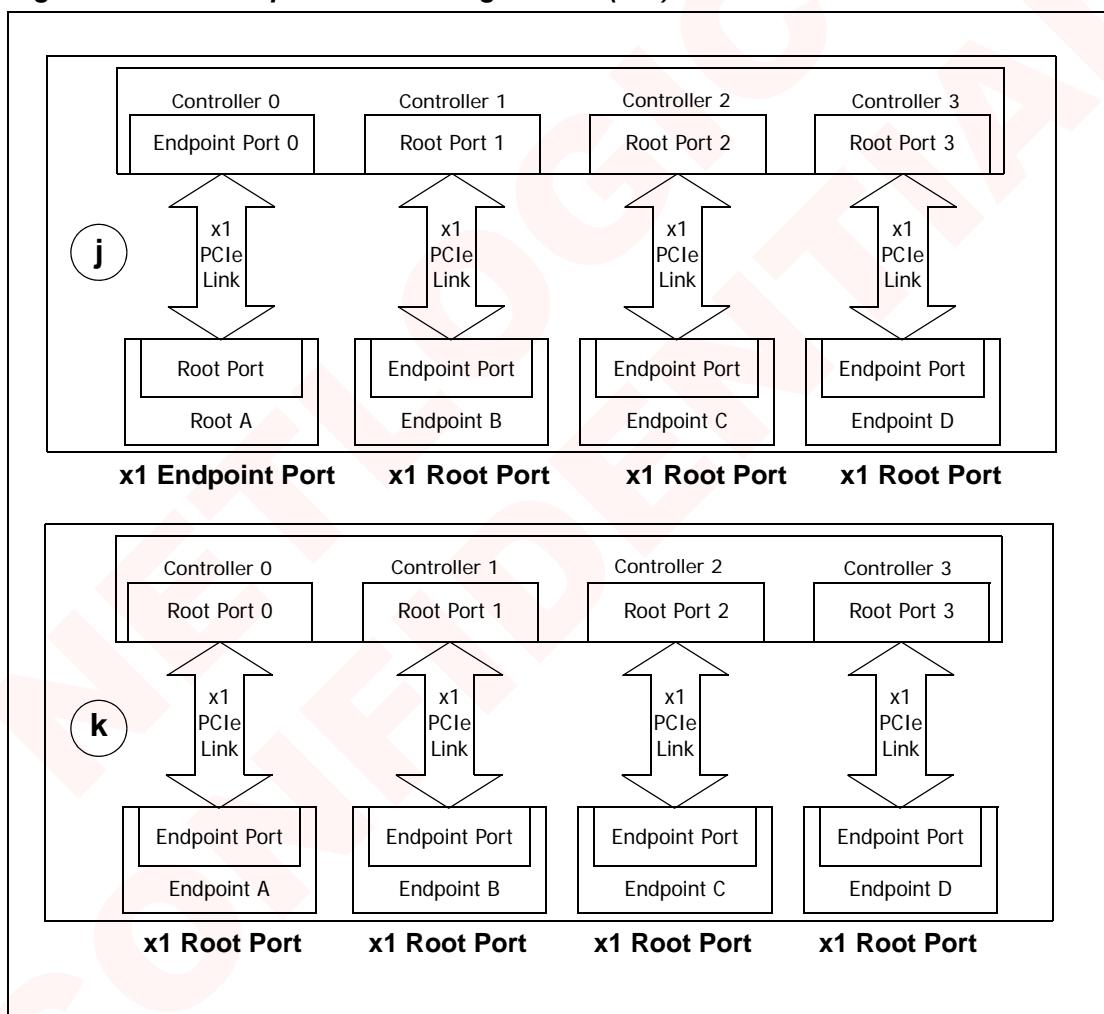
19.3.1.2

Supported PCIe Configurations for XLS2xx Devices

Table 19-4 and Figure 19-7 show PCI Express configurations supported by the XLS2xx devices from an external connectivity perspective.

Table 19-4. PCIe Express Link Configurations, XLS2xx

Strapping Pins		Diagram	PCIe Contr_0	PCIe Contr_1	PCIe Contr_2	PCIe Contr_3
IO_AD[11]	IO_AD[10]		Lane 0	Lane 1	Lane 2	Lane 3
High	Low	(j)	Endpoint Port	Root Port	Root Port	Root Port
High	High	(k)	Root Port	Root Port	Root Port	Root Port

Figure 19-7. PCIe Express Link Configurations (4x1) for XLS2xx Devices

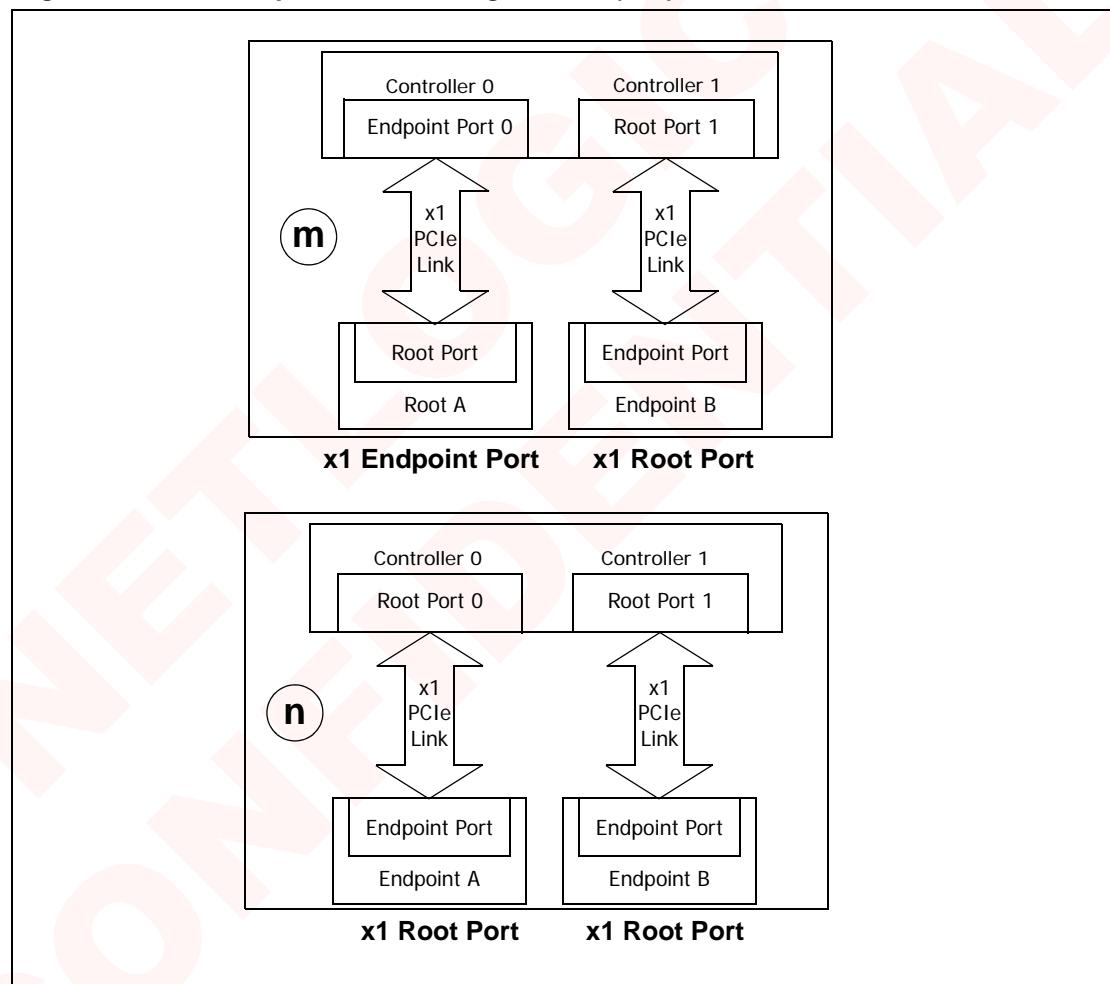
19.3.1.3 Supported PCIe Configurations for XLS1xx Devices

Table 19-5 and Figure 19-8 show PCI Express configurations supported by the XLS1xx devices from an external connectivity perspective.

Table 19-5. PCI Express Link Configurations, XLS1xx

Strapping Pins		Diagram	PCIe Controller_0	PCIe Controller_1
IO_AD[11]	IO_AD[10]		Lane 0	Lane 1
High	Low	(m)	Endpoint Port	Root Port
High	High	(n)	Root Port	Root Port

Figure 19-8. PCI Express Link Configurations (2x1) for XLS1xx Devices



19.4 Theory of Operation

The PCIe interface connects internally to the XLS through a station on the I/O DI ring, and as a station on the Fast Messaging Network ring. I/O DI operation is described in [Chapter 8, “Memory and I/O Access”](#). Accesses are done by one of two methods:

- Direct Access
- Message-based Access

On a direct access, the CPU reads or writes to a defined memory address space range. The internal Memory Bridge detects the read or write request, forwarding the transaction to the PCIe controller, which generates the appropriate PCIe signaling. Each PCIe operation, (e.g. memory read/write, I/O read/write and configuration read/write), has corresponded memory address space and range as defined by the PCIe control BAR (Basic Address Register) section in Chapter 8. Effectively, this set of registers are mapping the CPU memory address space to PCIe operations, that is – accessing this space generates a corresponding PCIe cycle:

- PCIE_CFG_BAR - Defines Configuration transaction space
- PCIE_ECFG_BAR - Defines Extended Configuration transaction space
- PCIE_MEM_BAR - Defines Memory transaction space
- PCIE_IO_BAR - Defines I/O transaction space

Due to the nature of PCIe, direct access consumes many CPU cycles, resulting in reduced efficiency and added latency.

Message-based accesses allow faster, lower-overhead transactions to and from the PCIe devices. This mechanism uses the XLS Fast Messaging Network to send and receive up to 256 Bytes on a single transaction.

Interrupt behavior of the PCIe interface is integrated into the XLS’s versatile interrupt-handling system, which allows effective delivery to threads.

The XLS PCIe external side interface complies with PCI Local Bus Specification Revision 2.2 for PCI-compatible operation, and PCI Express 2.0 Base specification with regard to signals and behavior. This section discusses, in the context of the PCIe interface, how various XLS modules access internal and external addressing spaces, and how to access the XLS’s PCI interface.

The XLS PCIe interface has been augmented to handle BE (Big-Endian) or LE (Little-Endian) data flexibly when accessed properly. See the discussion later in this chapter about register addressing.

19.4.1 Accessing PCIe Controller's Configuration Space

Software can access the PCIe Controller's own configuration header space through the XLS Peripheral & I/O Configuration region of memory (see [Section 8.4, Table 8-1](#)).

The XLS_IO_BAR register (Register #25) in the System Bridge Controller (SBC) provides the base address for accessing the Peripheral & I/O Configuration Region of memory. It is a software-programmable register. Out of reset, the XLS_IO_BAR is enabled and initialized to a default base address (see [Section 8.7.1](#)).

The PCIe controller has two configuration register regions set aside for it – one for big-endian memory address ordering and the other for little-endian ordering.

- XLS_IO_DEV_PCIE_BE at region offset 0x1E000 from XLS_IO_BAR base address
- XLS_IO_DEV_PCIE_LE at region offset 0x1F000 from XLS_IO_BAR base address

Both regions access the same registers. The only difference is in the interpretation of the addressing mode (i.e. big-endian vs little-endian).

Within a register region, there is four KBytes of space to provide access for up to 1024 registers. See [Section 19.9](#) and [Section 19.10](#) for the list and descriptions of the PCIe Configuration registers available.

[Section 8.5](#) illustrates how the three different address elements are put together to create the physical address to the XLS Peripheral & I/O Configuration Region of memory which is sent to the System Bridge Controller (SBC). The fields are shown in [Table 19-6](#):

Table 19-6. Physical Address Fields

Physical Address Field (A[X:Y])	Description
Address bits [39:20]	Base Address
Address bits [19:12]	Region Offset Bit [12] accomplishes the SWAP between big-endian and little-endian access format: When bit [12] = '0': Access is big-endian When bit [12] = '1': Access is little-endian
Address bits [11:0]	Register Address Offset

Since the CPU is a 64-bit machine, the software-generated address for the XLS Peripheral & I/O Configuration Region is a 64-bit address that is translated by the MMU into a 40-bit physical address sent to the System Bridge Controller (SBC).

When the SBC gets a memory transaction that matches the memory region for the PCIe configuration registers, it forwards that request to the PCIe controller, which performs the required configuration register access.

19.4.1.1 Configuration Space Access

The PCIe Controller can be set up to use the Original Configuration Space or the Extended Configuration Space, but not both.

The number of outstanding transactions targeted for the PCIe controller can be set by using the SYS2IO_CREDITS register (#53) in the SBC (see [Section 8.10.1](#)). The field – PCIE_CREDITS – can be increased or decreased from its default value.

Generating PCIe Configuration Transaction is achieved by first configuring the PCIE_CFG_BAR register. Upon access to this memory address space, either for read or write, the I/O-DI interface presents a 40-bit address (translated from 64 bits by the MMU) to the PCIe, which in turn is translated to PCIe configuration address as follows:

39:25	24	23:16	15:11	10:8	7:0
Reserved	Swap ^a	Bus Number	Device Number	Function Number	Register Number

a. Swap rotates byte ordering: '0' = Access as little-endian, '1' = Access as big-endian

Providing the Bus Number and Device Number that identifies the XLS PCIe Controller's own configuration space results in an internal configuration register access. The required read or write command effectively receives or changes the on-chip configuration registers.

19.4.1.2 Extended Configuration Space Access

Similar to PCIe configuration steps, the PCIE_ECFG_BAR register holds the memory address space for PCIe Extended Configuration cycles. Upon accessing this memory address space, the PCIe controller issues a PCIe Extended Configuration transaction. Addresses are translated as follows:

39:29	28	27:20	19:15	14:12	11:8	7:0
Reserved	Swap ^a	Bus Number	Device Number	Function Number	Extended Register Number	Register Number

a. Swap rotates byte ordering: '0' = Access as little-endian, '1' = Access as big-endian

19.4.2 Accessing PCIe Memory Space

Software must configure the PCIE_MEM_BAR register (#66) in the SBC to provide the base address for the PCIe Memory Space within XLS physical memory space.

Since the CPU is a 64-bit machine, the software generated address for PCIe Memory Space is a 64-bit address that is translated by the MMU into a 40-bit physical address sent to the System Bridge Controller (SBC).

The format for the physical address sent to the SBC is shown in [Table 19-7](#):

Table 19-7. Physical Address Fields

Physical Address Field (A[X:Y])	Description
Address bit [39]	Base Address
Address bits [38:24]	Programmable Memory Range
Address bits [23:0]	Memory Space

As shown in [Section 8.12.3, PCIE_MEM_BAR](#) provides an address mask with which to extend the memory space from 24 bits (16 MB) up to 39 bits (512 GB).

When the System Bridge Controller (SBC) gets a memory transaction that matches the memory region for the PCIe Memory Space as specified by [PCIE_MEM_BAR](#), it forwards that request to the PCIe controller which performs the required memory transaction.

19.4.2.1 Memory Read (Root Complex) Transaction

The application initiates a Memory Read Request to the PCIe controller by reading from the memory address range programmed into the PCIE_MEM_BAR register. The PCIe controller forms the read request into a PCIe read transaction, initiates the request over the PCIe bus, and upon receiving valid completion, returns the data to the application. If the PCIe controller did not receive the Completion for the Memory Read Request within the time window, it indicates a Completion time-out to the application client.

19.4.2.2 Memory Write (Root Complex) Transaction

The application initiates a Memory Write Request to the PCIe controller by writing to the memory address range programmed into the PCIE_MEM_BAR register. The PCIe controller forms the request into a PCIe write transaction in a manner similar to that of a Memory Read Request. The difference is that the PCIe controller does not expect to receive a Completion back (posted transaction).

19.4.2.3 Memory Read/Write (Endpoint) Transaction

The requesting component (a Root Complex port) initiates a Memory Read or Write Request to access XLS internal registers or memory. The PCIe controller filters the request based on the Configuration Registers BARS, then the IO-Distributed Interconnect (IO-DI) logic in the XLS fetches the requested data from (on read request), or writes the data to (on write request), the system memory.

19.4.3 Accessing PCIe I/O Space

Software must configure the [PCIE_IO_BAR](#) register (#67) in the SBC to provide the base address for the PCIe I/O Space within XLS' physical memory space.

Since the CPU is a 64-bit machine, the software generated address for PCIe I/O Space is a 64-bit address that is translated by the MMU into a 40-bit physical address sent to the System Bridge Controller (SBC).

The format for the physical address sent to the SBC is shown in [Table 19-8](#):

Table 19-8. Physical Address Fields

Physical Address Field (A[X:Y])	Description
Address bits [39:32]	Base Address
Address bits [31:26]	Programmable I/O Range
Address bits [23:0]	I/O Space

As shown in [Section 8.12.4, PCIE_IO_BAR](#) provides an address mask with which to extend the I/O space from 26 bits (64 MB) up to 32 bits (4 GB).

When the SBC gets a memory transaction that matches the memory region for the PCIe I/O Space as specified by [PCIE_IO_BAR](#), it forwards that request to the PCIe controller which performs the required I/O transaction.

19.4.4 PCIe Configuration Requests

If the PCIe is configured in root complex (RC) mode and no devices are connected to the PCIe bus, a bus error can occur if the primary, secondary, and subordinate bus numbers in the PCI Configuration Space Header (offset 0x18) are configured, followed by the issuance of a configuration request on the PCIe link. If this sequence occurs, a system reset may be required.

To ensure that configuration requests do not cause an error, perform a configuration request using the following sequence:

1. Connect external device(s) to the XLS PCIe bus.
2. Issue a configuration request.
3. Configure the primary, secondary and subordinate numbers into the PCI Configuration Space Header (offset 0x18).

If the external device cannot be guaranteed to be connected before configuring the bus (for example the device may be user-removable), use the following sequence:

1. Create global variables to track whether the PCIe links are up or not.
2. Set up the states of the global variables by examining the XLS link state registers [PCIE_LINK0/1/2/3_STATE](#) corresponding to links 0/1/2/3.
3. When later configuring the PCIe busses, avoid those that show link down.

19.5 DMA Transfers

Message-based PCIe transactions allow fast, low-latency, low-overhead data movement. Data transfer to and from the PCIe uses DMA operations. The DMA engines connect the PCIe controller to the memory subsystem via the FMN and IO-DI rings. Each PCIe controller implements two dedicated buckets with fixed 64 entries per bucket, one bucket for DMA inputs from PCIe bus and the other bucket for DMA outputs to the PCIe bus. The PCIe controller implements two types of accesses, a PCIe access initiated by an XLS transaction, and the second for data transaction initiated by PCIe Endpoints.

19.5.1 XLS to PCIe Bus Transaction

A CPU can initiate either a read or write transaction to the PCIe controller by using the Fast Message Network facilities. Prior to sending messages, the host must have enough credit assignments to the PCIe controller (see [Chapter 12, “Fast Messaging Network”](#) for mode details). The PCIe DMA fetches the requested data from the memory source, formats 256-byte packets automatically, and sends them over the PCIe link. For efficiency, it is recommended that the data should be aligned to 256-byte alignment as well as cache line alignment. (It is not a requirement, but the PCIe DMA controllers perform more efficiently this way.)

19.5.2 PCIe to XLS Subsystem Memory

A PCIe Endpoint (EP) can send/fetch data from the XLS subsystem memory by accessing the memory address space defined by the PCIe Configuration BARS. The PCIe controller buffers the incoming data with an internal buffer (up to 256 Bytes), checks for frame integrity, and performs DMA to the XLS subsystem memory transfers. It is up to the EP device to initiate interrupt messages notifying the host about newly arrived data. (Due to the Full Duplex nature of the PCIe bus, deadlock scenarios are avoided.)

19.5.2.1 PCIe DMA Message Formats

An incoming DMA message is composed of two 64-bit words. The fields of an incoming DMA message are defined in [Table 19-9](#).

Table 19-9. DMA Message Fields

Word	Bits	Field	Description	Word
0	63:61	Reserved		
	60	COHERENT	If the requested transaction data length is less than cache line, 32 bytes, setting this bit forces read-modify-write operation maintaining cache coherency over the whole cache line. Note that due to this process, the transaction takes twice as long to complete.	
	59	L2ALLOC	L2 Allocation allowing the bridge to hold a copy of the data in the L2 cache memory. This option should be avoided when there is no need for further processing to minimize L2 thrashing. If RDX bit is enabled, the operation is marked as "Read Exclusive", that is - after the read operation, the cache line is marked as invalid allowing reuse of this entry.	
	58	RDX	IOB Slave request read exclusive enable	
	57	RETN	Setting this bit instructs the PCIe FMN station controller to send a return message. The destination of the message is specified by the RETID bits.	
	56:50	RETID	Return message target ID	
	49:40	TID	Message tag ID: These 10 bits are used as Message Tag ID and are placed back, with no modifications, on a return message, allowing the recipient to track the return messages.	
	39:0	SRC	DMA source address: 40-bit address of the data to be send over the PCIe	
1	63	TD	PCIe request TD field	These four bits are inserted into the PCIe TLP (Transaction Layer Packet) header.
	62	EP	PCIe request EP field	
	61:60	ATTR	PCIe request ATTR field	
	59:40	BC	DMA data byte count: Number of Bytes to be transmitted	
	39:0	DEST	PCIe destination address	

An outgoing return message is composed of one 64-bit word. The fields of an outgoing return message are defined in [Table 19-10](#).

Table 19-10. Message Out Fields

Bits	Field	Description
63:14	Reserved	
13	FLUSH	Flush is enabled for the message
12	MSGERR	Incoming message error
11	IOBERR	IOB error
10	PCIEERR	PCIe link error
9:0	TID	Message Tag ID

19.6 PCIe Interrupts

The PCIe controller reports interrupts detected by each link. Interrupts are signaled through the “Virtual Wire” concept of the PCIe, that is - in-band messages carry Interrupt information or by assertion of dedicated PCIe Interrupt pins.

19.6.1 Interrupt Setup

The PCIe Controller detects multiple sources of interrupts and reports them to the Programmable Interrupt Controller (PIC) (described in [Chapter 10, “Programmable Interrupt Controller”](#)) as one of five (or three) interrupt requests as shown below:

- For XLS6xx and XLS4xx
 - PCIe Link 0 Interrupt
 - PCIe Link 1 Interrupt
 - PCIe Link 2 Interrupt
 - PCIe Link 3 Interrupt
 - PCIe Invalid Access Interrupt
- For XLS4xxLite
 - PCIe Link 0 Interrupt
 - PCIe Link 1 Interrupt
 - PCIe Invalid Access Interrupt
- For XLS2xx
 - PCIe Link 0 Interrupt
 - PCIe Link 1 Interrupt
 - PCIe Link 2 Interrupt
 - PCIe Link 3 Interrupt
 - PCIe Invalid Access Interrupt
- For XLS1xx
 - PCIe Link 0 Interrupt
 - PCIe Link 1 Interrupt
 - PCIe Invalid Access Interrupt

19.6.2 Message Signaled Interrupts (MSI)

Upon receiving of a PCIe Message Signaled Interrupt or when a PCIe interrupt pin is asserted, the PCIe controller reports the interrupt event to the PIC, where the host can later process the request. MSI interrupts can be masked by using [PCIE_LINK0_MSI_ENABLE](#), [PCIE_LINK1_MSI_ENABLE](#), [PCIE_LINK2_MSI_ENABLE](#), and [PCIE_LINK3_MSI_ENABLE](#) registers on [page 882](#) and [page 891](#).

PCIe Link 0 / Link 1 / Link 2 / Link 3 Interrupts for XLS6xx, XLS4xx and XLS2xx

The PCIe Link 0, Link 1, Link 2, Link 3 interrupts report interrupts detected by each link. If a link receives a PCIe message-based interrupt (MSI) or if a PCIe interrupt pin is asserted, then the link reports the interrupt as a PCIe Link 0 (or Link 1, Link 2, or Link 3) interrupt to the PIC.

- PCIe Controller registers [PCIE_LINK0_MSI_ENABLE](#), [PCIE_LINK1_MSI_ENABLE](#), [PCIE_LINK2_MSI_ENABLE](#), and [PCIE_LINK3_MSI_ENABLE](#) control which MSI interrupt vectors are reported. (See [Section 19.8.3](#))

- PCIe Controller registers [PCIE_LINK0_MSI_STATUS](#), [PCIE_LINK1_MSI_STATUS](#), [PCIE_LINK2_MSI_STATUS](#), and [PCIE_LINK3_MSI_STATUS](#) show which MSI interrupt vectors are pending. (See [Section 19.8.3](#))
- PCIe Controller registers [PCIE_LINK0_INT_ENABLE0/1](#), [PCIE_LINK1_INT_ENABLE0/1](#), [PCIE_LINK2_INT_ENABLE0/1](#), [PCIE_LINK2_INT_ENABLE0/1](#), [PCIE_LINK3_INT_ENABLE0/1](#) control which external PCIe interrupts are reported. (See [Section 19.8.3](#))
- PCIe Controller registers [PCIE_LINK0_INT_STATUS0/1](#), [PCIE_LINK1_INT_STATUS0/1](#), [PCIE_LINK2_INT_STATUS0/1](#), and [PCIE_LINK3_INT_STATUS0/1](#) show which external PCIe interrupts are pending. (See [Section 19.8.3](#))

PCIe Link 0 / Link 1 Interrupts for XLS4xxLite

The PCIe Link 0 and Link 1 interrupts report interrupts detected by each link. If a link receives a PCIe message-based interrupt (MSI) or if a PCIe interrupt pin is asserted, then the link reports the interrupt as a PCIe Link 0 (or Link 1) interrupt to the PIC.

- PCIe Controller registers [PCIE_LINK0_MSI_ENABLE](#) and [PCIE_LINK1_MSI_ENABLE](#), control which MSI interrupt vectors are reported. (See [Section 19.8.4](#) on page 900)
- PCIe Controller registers [PCIE_LINK0_MSI_STATUS](#) and [PCIE_LINK1_MSI_STATUS](#), show which MSI interrupt vectors are pending. (See [Section 19.8.4](#) on page 900)
- PCIe Controller registers [PCIE_LINK0_INT_ENABLE0/1](#) and [PCIE_LINK1_INT_ENABLE0/1](#) control which external PCIe interrupts are reported. (See [Section 19.8.4](#) on page 900)
- PCIe Controller registers [PCIE_LINK0_INT_STATUS0/1](#) and [PCIE_LINK1_INT_STATUS0/1](#) show which external PCIe interrupts are pending. (See [Section 19.8.3](#) on page 864)

PCIe Link 0 / Link 1 Interrupts for XLS1xx

The PCIe Link 0 and Link 1 interrupts for XLS1xx devices report interrupts detected by each link. If a link receives a PCIe message-based interrupt (MSI) or if a PCIe interrupt pin is asserted, then the link reports the interrupt as a PCIe Link 0 (or Link 1) interrupt to the PIC.

- PCIe Controller registers [PCIE_LINK0_MSI_ENABLE](#), and [PCIE_LINK1_MSI_ENABLE](#), control which MSI interrupt vectors are reported. (See [Section 19.8.3](#))
- PCIe Controller registers [PCIE_LINK0_MSI_STATUS](#), and [PCIE_LINK1_MSI_STATUS](#), show which MSI interrupt vectors are pending. (See [Section 19.8.3](#))
- PCIe Controller registers [PCIE_LINK0_INT_ENABLE0/1](#), and [PCIE_LINK1_INT_ENABLE0/1](#) control which external PCIe interrupts are reported. (See [Section 19.8.3](#))
- PCIe Controller registers [PCIE_LINK0_INT_STATUS0/1](#), and [PCIE_LINK1_INT_STATUS0/1](#) show which external PCIe interrupts are pending. (See [Section 19.8.3](#))

19.6.3

PCIe Invalid Access Interrupt

If the PCIe Controller receives a request which does not map to any of its address spaces, then it triggers the PCIe Invalid Request Interrupt.

- PCIe Controller register - [PCIE_IOBM_INT_ENABLE](#) ([Section 19.8](#)) - enables the Invalid Request interrupt.
- PCIe Controller register - [PCIE_IOBM_INT_STATUS](#) ([Section 19.8](#)) - shows where there is a pending Invalid Request interrupt.

19.6.4**Configuring PCIe and CPU for PCIe Interrupts**

Each of these PCIe interrupts is mapped to an entry in the PIC's Interrupt Redirection Table (IRT). Each entry can be configured to send the interrupt to an nCPU with a specific interrupt vector number. Please refer to [Chapter 10, “Programmable Interrupt Controller”](#) for more information on configuring the PIC IRT entries.

Once the user has programmed the PIC to direct a PCIe interrupt to an nCPU, then the user must configure that nCPU to see and respond to the interrupt. To accomplish this, each nCPU provides two Coprocessor 0 registers - the Status register (see [Section 4.2.3.14](#)) and the Extended Interrupt Mask Register (EIMR - see [Section 4.2.3.11](#)).

To enable interrupts from the PIC, the Status register's Interrupt Enable (IE) bit must be set so that the nCPU accepts any interrupt. Next, the desired interrupt vector number must be enabled by setting the corresponding bit in EIMR. For example, if the IRT entry for the PCIe Link 0 interrupt was set for interrupt vector number 20, then bit [20] in the EIMR must be set.

19.6.5**Interface Setup for Startup and Initialization in Device Mode**

The following must be configured and setup for startup and initialization in device mode:

- PCI Configuration Header registers (and any shadow registers) as needed (See [“PCI-Compatible Configuration Header Register Details”](#) on page 928.)
- PCIe interface's Extended Configuration Space Registers as needed (See [“PCI Express Extended Capability Register Maps”](#) on page 926.)

19.6.6**Interface Setup for Startup and Initialization in Host Mode**

The following must be configured and setup for startup and initialization in host mode:

- PCI Configuration Header registers (and any shadow registers) as needed (See [“PCI-Compatible Configuration Header Register Details”](#) on page 928.)
- PCIe interface's Extended Configuration Space Registers as needed (See [“PCI Express Extended Capability Register Maps”](#) on page 926.)

19.7**Programming Model****19.7.1****Startup/Initialization**

The PCIe interface operating mode is determined by the state of Peripherals Bus line IO_AD[11:10] at XLS reset time, as shown in [Table 19-2](#), [Table 19-4](#), and [Table 19-5](#). Bits [21:20] of the GPIO Reset Configuration register mirrors the IO_AD[11:10] logic levels latched at reset. For more information, see [Section 23.3.5 GPIO System Control Registers](#) in this manual and in the XLS Data Book.

19.7.2**Flushing the Interface Buffers**

This section describes how to flush buffers when it is necessary to reset the interface. There are two flush mechanisms used to synchronize the XLS and the PCIe interface:

- Link Flush — This is used when the PCIe interface must be reset but the XLS does not.
- IO DI Station Flush — This is used when XLS and the IO Distributed Interface Station connecting the PCIe interface to the XLS must be reset but not the PCIe interface itself

Link Flush

When the system must reset the PCIe interface but not XLS or IO DI, the following steps must be followed:

1. Put the PCIe Link0/Link1 interface in reset by setting **PCIE_CTRL1[0]** and **PCIE_CTRL1[16]** to '1'
2. Start Link0/Link1 flush by setting **PCIE_CTRL[1:0]** bits to '11'
3. Wait until all requests and responses are flushed out from the IO DI station;
4. Stop Link flush by clearing **PCIE_CTRL[1:0]** bits to '00'
5. Pull the PCIe interface out of reset by clearing **PCIE_CTRL1[0]** and **PCIE_CTRL1[16]**

IO DI Station Flush (XLS Reset)

When the system must reset XLS (EP mode only) but not the PCIe interface, the following steps must be followed:

1. RC put XLS in reset with Type0 Configuration Write to address 0x3FF, bit[0].
2. PCIe asserts reset request to XLS.
3. XLS goes to reset, reset asserted.
4. PCIe interface starts flushing.
5. Wait until all requests and responses are flushed out from the PCIe interface and XLS pulls out of reset.
6. PCIe stops flushing and clears bit[0] of configuration 0x3FF.

Meanwhile, the RC should be polling bit[0] of the configuration register 0x3FF for a value of '0' to indicate XLS reset is done.

Using a mixed mode (**IO_AD[11:10]** = '10'; where one controller is in EP mode and the second controller is in RC mode), when PCIe receives a reset request from the EP, the RC port must be reset. This should be done by software once the XLS reset operation is completed.

19.7.3

Ensure Writes Complete Before Reads

When performing write operations on a PCIe link, in order to avoid reading stale data, it is sometimes necessary to guarantee that all writes have completed prior to issuing a read. One technique sometimes used to accomplish this is to issue a zero-length read on the link. A zero-length read causes prior write transactions to complete without the risk of reading stale data. Unfortunately, the XLS PCIe Interface cannot issue these zero-length read requests on the PCIe link.

The XLS PCIe Interface can accept and will respond to zero length read requests issued by a partner PCIe device.

To ensure completion of prior write operations, XLS software should use an alternate technique. One possible example would be the issuing of a non-zero length "dummy" read (to any address) before performing additional reads. The PCIe protocol ensures that the prior write operation was completed (flush) as a result of the "dummy" read (just as it does with zero-length reads).

19.8 Global PCI Express Interface Registers

The registers in this section apply to both device and root controller modes. The Global PCI Express interface registers differ depending on which XLS family is being used. This section is divided into the following sub sections:

- Summary of XLS6xx, XLS4xx, XLS2xx & XLS1xx PCIE Global Registers
- Summary of the XLS408Lite & XLS404Lite Global PCIE Registers

19.8.1 Field Read/Write Access Codes

The following terms are used in the register descriptions to describe field Read/Write Access type:

Table 19-11. Field Read/Write Access Codes

Type Code	Meaning	Description
RO	Read Only	The field contains a static value and cannot be written.
R/W	Read/Write	The field can be modified by software. Note that some fields, where indicated, are modifiable only indirectly, by writing to a shadow register then performing a soft reset.
RC	Read/Clear	The field can be read to give status information. Bits may be cleared by writing a '1' to them; writes of '0' are ignored.
RW1C	Read-Only Status/ Write-'1'-to- Clear Status Register	Register bits indicate status when read. A set bit indicating a status event may be cleared by writing a '1'. Writing '0' to RW1C bits has no effect. Writing from the application side (if any) requires careful synchronization with host software.
WO	Write only	Any write to this field resets the field to the default value.

19.8.2 Register Address Offsets

In the following sections, register address offsets are given for byte addressing.

19.8.3 Summary of XLS6xx, XLS4xx, XLS2xx & XLS1xx PCIE Global Registers

Table 19-12. Summary of the XLS Families PCI Express Interface Global Registers

Register Offset DWord Addressing	Register Offset Byte Addressing	Name	R/W	Reset Value
0x000	0x000	PCIE_CTRL1	R/W	0x00000000
0x001	0x004	PCIE_CTRL2	R/W	0x00000000
0x002	0x008	PCIE_CTRL3	R/W	0x000C1000
0x003	0x00C	PCIE_CTRL4	R/W	0xAAAA2222
0x004	0x010	PCIE_CTRL	R/W	0x00000000
0x005	0x014	PCIE_IOBM_TIMER	R/W	0xFFFFFFFF
0x006	0x018	PCIE_MSI_CMD	R/W	0x00000000
0x007	0x01C	PCIE_MSI_RESP	RW1C	0x00000000
0x008	0x020	PCIE_DWC_CTRL5	R/W	0x00000000
0x009	0x024	PCIE_DWC_CTRL6	R/W	0x00000000
0x00A-0x00F	0x028-0x03C	Reserved		

Table 19-12. Summary of the XLS Families PCI Express Interface Global Registers (continued)

Register Offset DWord Addressing	Register Offset Byte Addressing	Name	R/W	Reset Value
0x010	0x040	PCIE_IOBM_SWAP_MEM_BASE	R/W	0xFFFFFFFFFF
0x011	0x044	PCIE_IOBM_SWAP_MEM_LIMIT	R/W	0x00000000
0x012	0x048	PCIE_IOBM_SWAP_IO_BASE	R/W	0xFFFFFFFFFF
0x013	0x04C	PCIE_IOBM_SWAP_IO_LIMIT	R/W	0x00000000
0x014	0x050	PCIE_TRGT_COHERENT_MEM_BASE	R/W	0x00000000
0x015	0x054	PCIE_TRGT_COHERENT_MEM_LIMIT	R/W	0xFFFFFFFFFF
0x016	0x058	PCIE_TRGT_L2ALLOC_MEM_BASE	R/W	0xFFFFFFFFFF
0x017	0x05C	PCIE_TRGT_L2ALLOC_MEM_LIMIT	R/W	0x00000000
0x018	0x060	PCIE_TRGT_READEX_MEM_BASE	R/W	0xFFFFFFFFFF
0x019	0x064	PCIE_TRGT_READEX_MEM_LIMIT	R/W	0x00000000
0x01A	0x068	PCIE_EP_MEM_BASE	R/W	0xFFFFFFFFFF
0x01B	0x06C	PCIE_EP_MEM_LIMIT	R/W	0x00000000
0x01C	0x070	PCIE_EP_ADDR_MAP_ENTRY0	R/W	0x00000000
0x01D	0x074	PCIE_EP_ADDR_MAP_ENTRY1	R/W	0x00000000
0x01E	0x078	PCIE_EP_ADDR_MAP_ENTRY2	R/W	0x00000000
0x01F	0x07C	PCIE_EP_ADDR_MAP_ENTRY3	R/W	0x00000000
0x020	0x080	PCIE_LINK0_STATE	RO	0x00000000
0x021	0x084	PCIE_LINK1_STATE	RO	0x00000000
0x022	0x088	PCIE_IOBM_INT_STATUS	RW1C	0x00000000
0x023	0x08C	PCIE_IOBM_INT_ENABLE	R/W	0x00000000
0x024	0x090	PCIE_LINK0_MSI_STATUS	RW1C	0x00000000
0x025	0x094	PCIE_LINK1_MSI_STATUS	RW1C	0x00000000
0x026	0x098	PCIE_LINK0_MSI_ENABLE	R/W	0x00000000
0x027	0x09C	PCIE_LINK1_MSI_ENABLE	R/W	0x00000000
0x028	0x0A0	PCIE_LINK0_INT_STATUS0	RO	0x00000000
0x029	0x0A4	PCIE_LINK1_INT_STATUS0	RO	0x00000000
0x02A	0x0A8	PCIE_LINK0_INT_STATUS1	RW1C	0x00000000
0x02B	0x0AC	PCIE_LINK1_INT_STATUS1	RW1C	0x00000000
0x02C	0x0B0	PCIE_LINK0_INT_ENABLE0	R/W	0x00000000
0x02D	0x0B4	PCIE_LINK1_INT_ENABLE0	R/W	0x00000000
0x02E	0x0B8	PCIE_LINK0_INT_ENABLE1	R/W	0x00000000
0x02F	0x0BC	PCIE_LINK1_INT_ENABLE1	R/W	0x00000000
0x030	0x0C0	PCIE_PHY_CR_CMD	R/W	0x00000000
0x031	0x0C4	PCIE_PHY_CR_WR_DATA	R/W	0x00000000
0x032	0x0C8	PCIE_PHY_CR_RESP	RW1C	0x00000000
0x033	0x0CC	PCIE_PHY_CR_RD_DATA	R/W	0x00000000
0x034	0x0D0	PCIE_IOBM_ERR_CMD	RO	0x00000000
0x035	0x0D4	PCIE_IOBM_ERR_LOWER_ADDR	RO	0x00000000
0x036	0x0D8	PCIE_IOBM_ERR_UPPER_ADDR	RO	0x00000000
0x037	0x0DC	PCIE_IOBM_ERR_BE	RO	0x00000000

Table 19-12. Summary of the XLS Families PCI Express Interface Global Registers (continued)

Register Offset DWord Addressing	Register Offset Byte Addressing	Name	R/W	Reset Value
0x038-0x3F	0x0E0-0x17C	Reserved		
0x060	0x180	PCIE_LINK2_STATE	RO	0x00000000
0x061	0x184	PCIE_LINK3_STATE	RO	0x00000000
0x062-0x063	0x188-0x018C	Reserved		0x00000000
0x064	0x190	PCIE_LINK2_MSI_STATUS	RW1C	0x00000000
0x065	0x194	PCIE_LINK3_MSI_STATUS	RW1C	0x00000000
0x066	0x198	PCIE_LINK2_MSI_ENABLE	R/W	0x00000000
0x067	0x19C	PCIE_LINK3_MSI_ENABLE	R/W	0x00000000
0x068	0x1A0	PCIE_LINK2_INT_STATUS0	RO	0x00000000
0x069	0x1A4	PCIE_LINK3_INT_STATUS0	RO	0x00000000
0x06A	0x1A8	PCIE_LINK2_INT_STATUS1	RW1C	0x00000000
0x06B	0x1AC	PCIE_LINK3_INT_STATUS1	RW1C	0x00000000
0x06C	0x1B0	PCIE_LINK2_INT_ENABLE0	R/W	0x00000000
0x06D	0x1B4	PCIE_LINK3_INT_ENABLE0	R/W	0x00000000
0x06E	0x1B8	PCIE_LINK2_INT_ENABLE1	R/W	0x00000000
0x06F	0x1BC	PCIE_LINK3_INT_ENABLE1	R/W	0x00000000
0x070-0x1D1	0x1C0-0x744	Reserved		
0x1D2	0x748	VC0_POSTED_RX_QUEUE_CTRL	R/W	0x00000000
0x1D3	0x74C-0x7A4	Reserved		
0x1EA	0x7A8	VC0_POSTED_BUFFER_DEPTH	R/W	0x00000000
0x1EB-0x307	0x1A0-0xC1C	Reserved		
0x308	0xC20	PCIE_MSG_TX_THRESHOLD	R/W	0x00000003
0x309-0x31F	0xC21-0xC7C	Reserved		
0x320	0xC80	PCIE_MSG_BUCKET_SIZE_0	R/W	0x00000020
0x321	0xC84	PCIE_MSG_BUCKET_SIZE_1	R/W	0x00000020
0x322	0xC88	PCIE_MSG_BUCKET_SIZE_2	R/W	0x00000020
0x323	0xC8C	PCIE_MSG_BUCKET_SIZE_3	R/W	0x00000020
0x324	0xC90	PCIE_MSG_BUCKET_SIZE_4	R/W	0x00000020
0x325	0xC94	PCIE_MSG_BUCKET_SIZE_5	R/W	0x00000020
0x326	0xC98	PCIE_MSG_BUCKET_SIZE_6	R/W	0x00000020
0x327	0xC9C	PCIE_MSG_BUCKET_SIZE_7	R/W	0x00000020
0x328 – 0x37F	0xCA0-0xDFC	Reserved		
0x380 – 0x3FF	0xE00-0xFFC	PCIE_MSG_CREDIT	R/W	0x00000000

Register Address Offsets

In the following sections, register address offsets are given for byte addressing.

19.8.3.1 PCIE_CTRL1
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x000

Bit#	Name	Description	Attribute	Reset
31:28	Reserved	Reserved		0x0
27	LINK1_APPS_PM_XMT_TURNOFF	Link1 request to generate a PM_TURN_OFF Message. (RC Mode Only)	WO	0x0
26	LINK1_APP_REQ_EXIT_L1	Link1 request to exit ASPM state L1.	R/W	0x0
25	LINK1_APP_REQ_ENTR_L1	Link1 request to enter ASPM state L1.	R/W	0x0
24	LINK1_SYS_AUX_PWR_DET	Indicates that Link1 auxiliary power is present.	R/W	0x0
23:22	Reserved	Reserved		0x0
21	LINK1_APP_INIT_RST	Link1 request to send a Hot Reset to the downstream device. (RC Mode Only)	R/W	0x0
20	Reserved	Reserved		0x0
19	LINK1_TX_LANE_FLIP_EN	Link1 request to perform transmit lane reversal.	R/W	0x0
18	LINK1_RX_LANE_FLIP_EN	Link1 request to perform receive lane reversal.	R/W	0x0
17	LINK1_APP_LTSSM_EN	Enable Link1 LTSSM.	R/W	0x0
16	LINK1_RST_N	Resets the Link1 core.	R/W	0x0
15:12	Reserved	Reserved		0x0
11	LINK0_APPs_PM_XMT_TURNOFF	Link0 request to generate a PM_TURN_OFF Message. (RC Mode Only)	WO	0x0
10	LINK0_APP_REQ_EXIT_L1	Link0 request to exit ASPM state L1.	R/W	0x0
9	LINK0_APP_REQ_ENTR_L1	Link0 request to enter ASPM state L1.	R/W	0x0
8	LINK0_SYS_AUX_PWR_DET	Indicates that Link0 auxiliary power is present.	R/W	0x0
7	LINK0_APPs_PM_XMT_PME	Link0 request to wake up the PMC state machine from the D3 state. (EP Mode Only)	WO	0x0
6	LINK0_OUTBAND_PWRUP_CMD	Link0 request to wake up the PMC state machine from the D3 state. (EP Mode Only)	WO	0x0
5	LINK0_APP_INIT_RST	Link0 request to send a Hot Reset to the downstream device. (RC Mode Only)	R/W	0x0
4	LINK0_APP_REQ_RETRY_EN	Link0 request to defer incoming Configuration Requests until initialization is complete. (EP Mode Only)	R/W	0x0
3	LINK0_TX_LANE_FLIP_EN	Link0 request to perform transmit lane reversal.	R/W	0x0
2	LINK0_RX_LANE_FLIP_EN	Link0 request to perform receive lane reversal.	R/W	0x0
1	LINK0_APP_LTSSM_EN	Enable Link0 LTSSM.	R/W	0x0
0	LINK0_RST_N	Resets the Link0 core.	R/W	0x0

19.8.3.2 PCIE_CTRL2**(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)**

Address Offset: 0x004

Bit#	Name	Description	Attribute	Reset
31:24	<i>Reserved</i>	<i>Reserved</i>		0x00
23	LINK1_EML_INTERLOCK_ENGAGED	Indicates whether the system electromechanical interlock is engaged for Link1.	R/W	0x0
22	LINK1_CMD_CPLED_INT	Indicates that the Hot-Plug controller completed a command for Link1.	R/W	0x0
21	LINK1_PRE_DET_CHGED	Indicates that the state of card present detector has changed for Link1.	R/W	0x0
20	LINK1_MRL_SENSOR_CHGED	Indicates that the state of MRL sensor has changed for Link1.	R/W	0x0
19	LINK1_PWR_FAULT_DET	Indicates the power controller detected a power fault at this slot for Link1.	R/W	0x0
18	LINK1_MRL_SENSOR_STATE	Indicates the state of the manually-operated retention latch (MRL) sensor for Link1: 0: MRL is closed 1: MRL is open	R/W	0x0
17	LINK1_PRE_DET_STATE	Indicates presence of a card in the slot for Link1: 0: Slot is empty 1: Card is present in the slot	R/W	0x0
16	LINK1_ATTEN_BUTTON_PRESSED	Indicates that the system attention button was pressed for Link1. (RC Mode Only)	R/W	0x0
15:8	<i>Reserved</i>	<i>Reserved</i>		0x00
7	LINK0_EML_INTERLOCK_ENGAGED	Indicates whether the system electromechanical interlock is engaged for Link0.	R/W	0x0
6	LINK0_CMD_CPLED_INT	Indicates that the Hot-Plug controller completed a command for Link0.	R/W	0x0
5	LINK0_PRE_DET_CHGED	Indicates that the state of card present detector has changed for Link0.	R/W	0x0
4	LINK0_MRL_SENSOR_CHGED	Indicates that the state of MRL sensor has changed for Link0.	R/W	0x0
3	LINK0_PWR_FAULT_DET	Indicates the power controller detected a power fault at this slot for Link0.	R/W	0x0
2	LINK0_MRL_SENSOR_STATE	Indicates the state of the manually-operated retention latch (MRL) sensor for Link0: 0: MRL is closed 1: MRL is open	R/W	0x0
1	LINK0_PRE_DET_STATE	Indicates presence of a card in the slot for Link0: 0: Slot is empty 1: Card is present in the slot	R/W	0x0
0	LINK0_ATTEN_BUTTON_PRESSED	Indicates that the system attention button was pressed for Link0. (RC Mode Only)	R/W	0x0

19.8.3.3 PCIE_CTRL3
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x008

Bit#	Name	Description	Attribute	Reset
31:21	Reserved	Reserved		0x000
20:16	PHY_TX_LVL	<p>Transmitter output level:</p> <p>5'b00000: 0.892V 5'b00001: 0.901V 5'b00010: 0.911V 5'b00011: 0.920V 5'b00100: 0.929V 5'b00101: 0.938V 5'b00110: 0.948V 5'b00111: 0.957V 5'b01000: 0.966V 5'b01001: 0.076V 5'b01010: 0.985V 5'b01011: 0.994V 5'b01100: 1.003V 5'b01101: 1.013V 5'b01110: 1.022V 5'b01111: 1.031V 5'b10000: 1.041V 5'b10001: 1.050V 5'b10010: 1.059V 5'b10011: 1.069V 5'b10100: 1.078V 5'b10101: 1.087V 5'b10110: 1.096V 5'b10111: 1.106V 5'b11000: 1.115V 5'b11001: 1.124V 5'b11010: 1.134V 5'b11011: 1.143V 5'b11100: 1.152V 5'b11101: 1.161V 5'b11110: 1.171V 5'b11111: 1.180V</p> <p>Note: This field is intended for device debug purposes. For normal operation, the default value should be used.</p>	R/W	0x0C
15:13	Reserved	Reserved		0x0
12:8	PHY_LOS_LVL	<p>Loss-of-signal detection sensitivity. The recommended setting is 16.</p> <p>Note: This field is intended for device debug purposes. For normal operation, the default value should be used.</p>	R/W	0x10
7:5	Reserved	Reserved		0x0
4:0	Reserved	Reserved		0x0

19.8.3.4 PCIE_CTRL4**(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)**

Address Offset: 0x00C

Bit#	Name	Description	Attribute	Reset
31:28	PHY3_TX_BOOST	Transmitter boost control for lane3. Programmed boost value is: boost = -20log(1-(phy3_tx_boost[3:0]+0.5)/32)dB, except setting tx_boost[3:0] to '0' truly produces 0 db of boost. This produces results up to 5.75 dB in steps of ~0.37dB. May transition asynchronously. Changes while the transmitter is on may briefly cause undesired transmit waveforms.	R/W	0xA
27:24	PHY2_TX_BOOST	Transmitter boost control for lane2.	R/W	0xA
23:20	PHY1_TX_BOOST	Transmitter boost control for lane1.	R/W	0xA
19:16	PHY0_TX_BOOST	Transmitter boost control for lane0.	R/W	0xA
15	Reserved	Reserved		0x0
14:12	PHY3_RX_EQ_VAL	Receiver equalization boost control for lane3. Internal linear equalizer boost is ~(phy3_rx_eq_val+1)*0.5dB. May transition asynchronously. Transitions may cause bit error as the boost changes values. 3'b000: 0.5dB 3'b001: 1dB 3'b010: 1.5dB 3'b011: 2dB 3'b100: 2.5dB 3'b101: 3dB 3'b110: 3.5dB 3'b111: 4dB	R/W	0x2
11	Reserved	Reserved		0x0
10:8	PHY2_RX_EQ_VAL	Receiver equalization boost control for lane2.	R/W	0x2
7	Reserved	Reserved		0x0
6:4	PHY1_RX_EQ_VAL	Receiver equalization boost control for lane1.	R/W	0x2
3	Reserved	Reserved		0x0
2:0	PHY0_RX_EQ_VAL	Receiver equalization boost control for lane0.	R/W	0x2

19.8.3.5 PCIE_CTRL

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x010

Bit#	Name	Description	Attribute	Reset
31	TRGT_TIMEOUT_EN	Target interface timeout enable; when this bit is set, the timeout for the target interface completion is enabled. Internal use only; show as Reserved for external documentation.	R/W	0x0
30:11	Reserved			0x00000
10	LINK3_CFG_BAR_MASK_EN	Enables Link3 configuration of the BAR masks.	R/W	0x0
9	LINK2_CFG_BAR_MASK_EN	Enables Link2 configuration of the BAR masks.	R/W	0x0
8	LINK3_FLUSH	Request to flush all Link3 messages and IOB master requests and return error responses. The flush is used when the PCIE core needs reset but Condor does not.	R/W	0x0
7	LINK2_FLUSH	Request to flush all Link2 messages and IOB master requests and return error responses. The flush is used when the PCIE core needs reset but Condor does not.	R/W	0x0
6	INT	Legacy interrupt pin; when the bit goes from low to high, the PCIe core generates an ASSERT_INTX Message; when the bit goes from high to low, the PCIe core generates an DEASSERT_INTX Message. (EP Mode Only)	R/W	0x0
5	LINK1_CFG_BAR_MASK_EN	Enables Link1 configuration of the BAR masks.	R/W	0x0
4	LINK0_CFG_BAR_MASK_EN	Enables Link0 configuration of the BAR masks.	R/W	0x0
3	ADDR_MAP_EN	Enables PCI to Condor address mapping. (EP Mode Only)	R/W	0x0
2	LPBK_EN	Enables PHY Rx to Tx parallel data loop-back.	R/W	0x0
1	LINK1_FLUSH	Request to flush all Link1 messages and IOB master requests and return error responses. The flush is used when the PCIE core needs reset but Condor does not.	R/W	0x0
0	LINK0_FLUSH	Request to flush all Link0 messages and IOB master requests and return error responses. The flush is used when the PCIE core needs reset but Condor does not.	R/W	0x0

19.8.3.6 PCIE_IOBM_TIMER

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x014

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_REQUEST_TIMER	Number of IodClk cycles for an unaccepted IOB master request to timeout	R/W	0xFFFFFFFF

19.8.3.7 PCIE_MSI_CMD

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x018

Bit#	Name	Description	Attribute	Reset
31:6	Reserved	Reserved		0x00000000
5	MSI_EN	Enables Link0 MSI. (EP Mode Only)	R/W	0x0
4:0	MSI	Link0 MSI vector. When the MSI_EN is asserted and when MSI_DONE is de-asserted, an MSI request is generated with MSI vector. (EP Mode Only)	R/W	0x00

19.8.3.8 PCIE_MSI_RESP

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x01C

Bit#	Name	Description	Attribute	Reset
31:1	Reserved	Reserved		0x00000000
0	MSI_DONE	Indicates Link0 MSI request is done. (EP Mode Only)	RW1C	0x0

19.8.3.9 PCIE_DWC_CTRL5
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x020

Bit#	Name	Description	Attribute	Reset
31:28	<i>Reserved</i>			0x0
27	LINK3_APPS_PM_XMT_TURNOFF	Link3 request to generate a PM_TURN_OFF Message. (RC Mode Only)	WO	0x0
26	LINK3_APP_REQ_EXIT_L1	Link3 request to exit ASPM state L1.	R/W	0x0
25	LINK3_APP_REQ_ENTR_L1	Link3 request to enter ASPM state L1.	R/W	0x0
24	LINK3_SYS_AUX_PWR_DET	Indicates that Link3 auxiliary power is present.	R/W	0x0
23:22	<i>Reserved</i>			0x0000
21	LINK3_APP_INIT_RST	Link3 request to send a Hot Reset to the downstream device. (RC Mode Only)	R/W	0x0
20	<i>Reserved</i>			0x0
19	LINK3_TX_LANE_FLIP_EN	Link3 request to perform transmit lane reversal.	R/W	0x0
18	LINK3_RX_LANE_FLIP_EN	Link3 request to perform receive lane reversal.	R/W	0x0
17	LINK3_APP_LTSSM_EN	Enable Link3 LTSSM.	R/W	0x0
16	LINK3_RST_N	Resets the Link3 core.	R/W	0x0
15:12	<i>Reserved</i>			0x0
11	LINK2_APPs_PM_XMT_TURNOFF	Link2 request to generate a PM_TURN_OFF Message. (RC Mode Only)	WO	0x0
10	LINK2_APP_REQ_EXIT_L1	Link2 request to exit ASPM state L1.	R/W	0x0
9	LINK2_APP_REQ_ENTR_L1	Link2 request to enter ASPM state L1.	R/W	0x0
8	LINK2_SYS_AUX_PWR_DET	Indicates that Link2 auxiliary power is present.	R/W	0x0
7:6	<i>Reserved</i>			0x0
5	LINK2_APP_INIT_RST	Link2 request to send a Hot Reset to the downstream device. (RC Mode Only)	R/W	0x0
4	<i>Reserved</i>			0x0
3	LINK2_TX_LANE_FLIP_EN	Link2 request to perform transmit lane reversal.	R/W	0x0
2	LINK2_RX_LANE_FLIP_EN	Link2 request to perform receive lane reversal.	R/W	0x0
1	LINK2_APP_LTSSM_EN	Enable Link2 LTSSM.	R/W	0x0
0	LINK2_RST_N	Resets the Link2 core.	R/W	0x0

19.8.3.10 PCIE_DWC_CTRL6

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x024

Bit#	Name	Description	Attribute	Reset
31:24	<i>Reserved</i>			0x00
23	LINK3_EML_INTERLOCK_ENGAGED	Indicates whether the system electromechanical interlock is engaged for Link3.	R/W	0x0
22	LINK3_CMD_CPLED_INT	Indicates that the Hot-Plug controller completed a command for Link3.	R/W	0x0
21	LINK3_PRE_DET_CHGED	Indicates that the state of card present detector has changed for Link3.	R/W	0x0
20	LINK3_MRL_SENSOR_CHGED	Indicates that the state of MRL sensor has changed for Link3.	R/W	0x0
19	LINK3_PWR_FAULT_DET	Indicates the power controller detected a power fault at this slot for Link3.	R/W	0x0
18	LINK3_MRL_SENSOR_STATE	Indicates the state of the manually-operated retention latch (MRL) sensor for Link3: 0: MRL is closed 1: MRL is open	R/W	0x0
17	LINK3_PRE_DET_STATE	Indicates presence of a card in the slot for Link3: 0: Slot is empty 1: Card is present in the slot	R/W	0x0
16	LINK3_ATTN_BUTTON_PRESSED	Indicates that the system attention button was pressed for Link3. (RC Mode Only)	R/W	0x0
15:8	<i>Reserved</i>			0x00
7	LINK2_EML_INTERLOCK_ENGAGED	Indicates whether the system electromechanical interlock is engaged for Link2.	R/W	0x0
6	LINK2_CMD_CPLED_INT	Indicates that the Hot-Plug controller completed a command for Link2.	R/W	0x0
5	LINK2_PRE_DET_CHGED	Indicates that the state of card present detector has changed for Link2.	R/W	0x0
4	LINK2_MRL_SENSOR_CHGED	Indicates that the state of MRL sensor has changed for Link2.	R/W	0x0
3	LINK2_PWR_FAULT_DET	Indicates the power controller detected a power fault at this slot for Link2.	R/W	0x0
2	LINK2_MRL_SENSOR_STATE	Indicates the state of the manually-operated retention latch (MRL) sensor for Link2: 0: MRL is closed 1: MRL is open	R/W	0x0
1	LINK2_PRE_DET_STATE	Indicates presence of a card in the slot for Link2: 0: Slot is empty 1: Card is present in the slot	R/W	0x0
0	LINK2_ATTN_BUTTON_PRESSED	Indicates that the system attention button was pressed for Link2. (RC Mode Only)	R/W	0x0

19.8.3.11 PCIE_IOBM_SWAP_MEM_BASE
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x040

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_SWAP_MEM_BASE	IOB Master request memory space base for byte swapping; the memory base addresses minimum memory size of 256B.	R/W	0xFFFFFFFF

19.8.3.12 PCIE_IOBM_SWAP_MEM_LIMIT
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x044

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_SWAP_MEM_LIMIT	IOB Master request memory space limit for byte swapping; the memory limit addresses minimum memory size of 256B.	R/W	0x00000000

19.8.3.13 PCIE_IOBM_SWAP_IO_BASE
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x048

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_SWAP_IO_BASE	IOB Master request IO space base for byte swapping; the IO base addresses minimum memory size of 32B.	R/W	0xFFFFFFFF

19.8.3.14 PCIE_IOBM_SWAP_IO_LIMIT
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x04C

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_SWAP_IO_LIMIT	IOB Master request IO space limit for byte swapping; the IO limit addresses minimum memory size of 32B.	R/W	0x00000000

19.8.3.15 PCIE_TRGT_COHERENT_MEM_BASE
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x050

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_TRGT_COHERENT_MEM_BASE	PCIE link request memory space base for coherency; the memory base addresses minimum memory size of 256B.	R/W	0xFFFFFFFF

19.8.3.16 PCIE_TRGT_COHERENT_MEM_LIMIT
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x054

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_TRGT_COHERENT_MEM_LIMIT	PCIE link request memory space limit for coherency; the memory limit addresses minimum memory size of 256B.	R/W	0xFFFFFFFF

19.8.3.17 PCIE_TRGT_L2ALLOC_MEM_BASE
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x058

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_TRGT_L2ALLOC_MEM_BASE	PCIE link request memory space base for L2 Allocation; the memory base addresses a minimum memory size of 256B. Programming this base address and also the PCIE_TRGT_L2ALLOC_MEM_LIMIT register together defines a low (base) and high (limit) address range. See the description of the PCIE_TRGT_L2ALLOC_MEM_LIMIT register for a description of how this range affects PCIe accesses.	R/W	0xFFFFFFFF

19.8.3.18 PCIE_TRGT_L2ALLOC_MEM_LIMIT
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x05C

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_TRGT_L2ALLOC_MEM_LIMIT	<p>PCIE link request memory space limit for L2 allocation; the memory limit addresses a minimum memory size of 256B.</p> <p>Programming this limit address and also the PCIE_TRGT_L2ALLOC_MEM_BASE register together defines a low (base) and high (limit) address range.</p> <p>If the address of an incoming PCIE memory read transaction falls between the L2 Allocation memory base and limit defined by this range, and if the read data is not present in the L2 Cache, then the read data is stored in the L2 Cache.</p> <p>If the address of an incoming PCIE memory write transaction falls between the L2 Allocation memory base and limit defined by this range, the write data goes into the L2 cache.</p>	R/W	0x00000000

19.8.3.19 PCIE_TRGT_READEX_MEM_BASE
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x060

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_TRGT_READEX_MEM_BASE	<p>PCIE link request memory space base for Read Exclusive; the memory base addresses a minimum memory size of 256B.</p> <p>Programming this base address and also the PCIE_TRGT_READEX_MEM_LIMIT register together defines a low (base) and high (limit) address range. See the description of the PCIE_TRGT_READEX_MEM_LIMIT register for a description of how this range affects PCIe accesses.</p>	R/W	0xFFFFFFFF

19.8.3.20 PCIE_TRGT_READEX_MEM_LIMIT**(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)**

Address Offset: 0x064

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_TRGT_READEX_MEM_LIMIT	<p>PCIE link request memory space limit for Read Exclusive; the memory limit addresses a minimum memory size of 256B.</p> <p>Programming this limit address and also the PCIE_TRGT_READEX_MEM_BASE register together defines a low (base) and high (limit) address range.</p> <p>If the address of an incoming PCIE memory read transaction falls between the Read Exclusive memory base and limit defined by this range, and if the read data is present in the L2 Cache, then the read data is flushed from the L2 Cache.</p> <p>The READEX range has no impact on write transactions.</p>	R/W	0x00000000

19.8.3.21 PCIE_EP_MEM_BASE**(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)**

Address Offset: 0x068

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_EP_MEM_BASE	<p>PCIE Endpoint (EP) link memory space base; the memory base addresses a minimum memory size of 256B. (EP Mode Only)</p> <p>Programming this base address and also the PCIE_EP_MEM_LIMIT register together defines a low (base) and high (limit) address range. See the description of the PCIE_EP_MEM_LIMIT register for a description of how this range affects PCIe accesses.</p>	R/W	0xFFFFFFFF

19.8.3.22 PCIE_EP_MEM_LIMIT
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x06C

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_EP_MEM_LIMIT	<p>PCIE Endpoint (EP) link memory space limit; the memory limit addresses a minimum memory size of 256B. (EP Mode Only)</p> <p>Programming this limit address and also the PCIE_EP_MEM_BASE register together defines a low (base) and high (limit) address range.</p> <p>If the address of a CPU PCIE memory request falls between the EndPoint (EP) memory base and limit defined by this range, then the memory request should be sent from the Endpoint (EP) port.</p> <p>The EP base and limit registers are necessary because the XLS has multiple PCIe ports. Endpoint (EP) ports have BARs for incoming memory requests from the link, but no BARs for outgoing memory requests.</p> <p>For Root Complex (RC) ports, the memory base and limit are built into the type1 configuration header; thus PCIE_EP_MEM_BASE and PCIE_EP_MEM_LIMIT registers have no bearing on PCIe RC ports.</p>	R/W	0x00000000

19.8.3.23 PCIE_EP_ADDR_MAP_ENTRY0
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x070

Bit#	Name	Description	Attribute	Reset
31:15	IOBM_EP_ADDR_MAP_ENTRY0	PCIE EP link memory request address map entry 0. When ADDR_MAP_EN is set and when PCIE address [24:23] == 2'b00, the address map entry 0 is used as XLS address [39:23]. (EP Mode Only)	R/W	0x00000
14:0	Reserved			0x0000

19.8.3.24 PCIE_EP_ADDR_MAP_ENTRY1

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x074

Bit#	Name	Description	Attribute	Reset
31:15	IOBM_EP_ADDR_MAP_ENTRY1	PCIE EP link memory request address map entry 1. When ADDR_MAP_EN is set and when PCIE address [24:23] == 2'b01, the address map entry 1 is used as XLS address [39:23]. (EP Mode Only)	R/W	0x00000
14:0	<i>Reserved</i>			0x0000

19.8.3.25 PCIE_EP_ADDR_MAP_ENTRY2

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x078

Bit#	Name	Description	Attribute	Reset
31:15	IOBM_EP_ADDR_MAP_ENTRY0	PCIE EP link memory request address map entry 0. When ADDR_MAP_EN is set and when PCIE address [24:23] == 2'b10, the address map entry 2 is used as XLS address [39:23]. (EP Mode Only)	R/W	0x00000
14:0	<i>Reserved</i>			0x0000

19.8.3.26 PCIE_EP_ADDR_MAP_ENTRY3

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x07C

Bit#	Name	Description	Attribute	Reset
31:15	IOBM_EP_ADDR_MAP_ENTRY1	PCIE EP link memory request address map entry 1. When ADDR_MAP_EN is set and when PCIE address [24:23] == 2'b11, the address map entry 3 is used as XLS address [39:23]. (EP Mode Only)	R/W	0x00000
14:0	<i>Reserved</i>			0x0000

19.8.3.27 PCIE_LINK0_STATE
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x080

Bit#	Name	Description	Attribute	Reset
31:23	Reserved	Reserved		0x0
22:18	LTSSM_STATE	Link0 current LTSSM state.	RO	0x00
17:15	PM_STATE	Link0 current power state.	RO	0x0
14	RDLH_LINK_UP	Link0 data link layer up.	RO	0x0
13	XMLH_LINK_UP	Link0 PHY link up.	RO	0x0
12:8	DEVICE_NUMBER	Link0 PCIE device number.	RO	0x00
7:0	BUS_NUMBER	Link0 PCIE bus number.	RO	0x00

19.8.3.28 PCIE_LINK1_STATE
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x084

Bit#	Name	Description	Attribute	Reset
31:23	Reserved	Reserved		0x00
22:18	LTSSM_STATE	Link1 current LTSSM state.	RO	0x00
17:15	PM_STATE	Link1 current power state.	RO	0x0
14	RDLH_LINK_UP	Link1 data link layer up.	RO	0x0
13	XMLH_LINK_UP	Link1 PHY link up.	RO	0x0
12:8	DEVICE_NUMBER	Link1 PCIE device number.	RO	0x00
7:0	BUS_NUMBER	Link1 PCIE bus number.	RO	0x00

19.8.3.29 PCIE_IOBM_INT_STATUS
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0088

Bit#	Name	Description	Attribute	Reset
31:30	Reserved	Reserved		0x00000000
0	IOBM_INT_STATUS	IOB Master request interrupt.	RW1C	0x0

19.8.3.30 PCIE_IOBM_INT_ENABLE

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x008C

Bit#	Name	Description	Attribute	Reset
31:30	Reserved	Reserved		0x00000000
0	IOBM_INT_ENABLE	IOB Master request interrupt enable.	R/W	0x0

19.8.3.31 PCIE_LINK0_MSI_STATUS

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0090

Bit#	Name	Description	Attribute	Reset
31:0	MSI_STATUS	PCIE Link0 MSI status. When a MSI is received on PCIE link0, the corresponding bit is set for the MSI vector; bit [0] is for vector 1, bit [1] is for vector 2 and so on. (RC Mode Only)	RW1C	0x00000000

19.8.3.32 PCIE_LINK1_MSI_STATUS

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0094

Bit#	Name	Description	Attribute	Reset
31:0	MSI_STATUS	PCIE Link1 MSI status. When a MSI is received on PCIE link1, the corresponding bit is set for the MSI vector; bit [0] is for vector 1, bit [1] is for vector 2 and so on. (RC Mode Only)	RW1C	0x00000000

19.8.3.33 PCIE_LINK0_MSI_ENABLE

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0098

Bit#	Name	Description	Attribute	Reset
31:0	MSI_ENABLE	PCIE Link0 MSI enables. (RC Mode Only)	R/W	0x00000000

19.8.3.34 PCIE_LINK1_MSI_ENABLE

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x009C

Bit#	Name	Description	Attribute	Reset
31:0	MSI_ENABLE	PCIE Link1 MSI enables. (RC Mode Only)	R/W	0x00000000

19.8.3.35 PCIE_LINK0_INT_STATUS0
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0A0

Bit#	Name	Description	Attribute	Reset
31:7	Reserved	Reserved		0x00000000
6	HP_INT	Link0 hot-plug interrupt. This bit is set if the Hot-Plug interrupts are enabled in the Slot Control register, any bit in the Slot Status register is set to '1' and the INTX Assertion Disable bit in the command register is '0'.	RO	0x0
5	PME_INT	Link0 PME interrupt. The bit is set when the PME status bit in the Root Status register is set to '1', the Interrupt Enable bit in the Root Control register is set to '1', and the INTX Assertion Disable bit in the Command register is '0'. (RC Mode Only)	RO	0x0
4	AER_RC_ERR_INT	Link0 advanced error interrupt. The bit is set when an error condition causes a bit to be set in the Root Error Status register and the associated error message reporting enable bit is set in the Root Error Command register. (RC Mode Only)	RO	0x0
3	INTD	Link0 INTD. (RC Mode Only)	RO	0x0
2	INTC	Link0 INTC. (RC Mode Only)	RO	0x0
1	INTB	Link0 INTB. (RC Mode Only)	RO	0x0
0	INTA	Link0 INTA. (RC Mode Only)	RO	0x0

19.8.3.36 PCIE_LINK1_INT_STATUS0
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0A4

Bit#	Name	Description	Attribute	Reset
31:5	Reserved	Reserved		0x00000000
6	HP_INT	Link1 hot-plug interrupt. This bit is set if the Hot-Plug interrupts are enabled in the Slot Control register, any bit in the Slot Status register is set to '1' and the INTX Assertion Disable bit in the command register is '0'.	RO	0x0
5	PME_INT	Link2 PME interrupt. The bit is set when the PME status bit in the Root Status register is set to '1', the Interrupt Enable bit in the Root Control register is set to '1', and the INTX Assertion Disable bit in the Command register is '0'. (RC Mode Only)	RO	0x0
4	AER_RC_ERR_INT	Link1 advanced error interrupt. The bit is set when an error condition causes a bit to be set in the Root Error Status register and the associated error message reporting enable bit is set in the Root Error Command register. (RC Mode Only)	RO	0x0
3	INTD	Link1 INTD. (RC Mode Only)	RO	0x0
2	INTC	Link1 INTC. (RC Mode Only)	RO	0x0
1	INTB	Link1 INTB. (RC Mode Only)	RO	0x0
0	INTA	Link1 INTA. (RC Mode Only)	RO	0x0

19.8.3.37 PCIE_LINK0_INT_STATUS1**(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)****Address Offset: 0x0A8**

Bit#	Name	Description	Attribute	Reset
31:12	Reserved	Reserved		0x0000000
11	PM_TURNOFF	Link0 PME Turnoff message received interrupt. (EP Mode Only)	RW1C	0x0
10	PM_TO_ACK	Link0 PME Turnoff Acknowledge message received interrupt. (RC Mode Only)	RW1C	0x0
9	PM_PME	Link0 PME Message received interrupt. (RC Mode Only)	RW1C	0x0
8	ERR_FATAL	Link0 fatal error message received interrupt. (RC Mode Only)	RW1C	0x0
7	ERR_NONFATAL	Link0 non-fatal error message received interrupt. (RC Mode Only)	RW1C	0x0
6	ERR_COR	Link0 correctable error message received interrupt. (RC Mode Only)	RW1C	0x0
5	PME_MSI	Link0 PME MSI. The bit is set when the PME status bit in the Root Status register is set to '1', the Interrupt Enable bit in the Root Control register is set to '1', and MSI is enabled. (RC Mode Only)	RW1C	0x0
4	AER_RC_ERR_MSI	Link0 advanced error MSI. The bit is set when an error condition causes a bit to be set in the Root Error Status register, the associated error message reporting enable bit is set in the Root Error Command register, and MSI is enabled (RC Mode Only)	RW1C	0x0
3	HP_INT	Link0 hot-plug MSI. This bit is set if the Hot-Plug interrupts are enabled in the Slot Control register, if any bit in the Slot Status register transitions from '0' to '1', and MSI is enabled.	RW1C	0x0
2	HP_PME	Link0 hot-plug PME wake generation. This bit is set if the PME Enable bit in the Power Management Control and Status register is set to '1', and if any bit in the Slot Status register transitions from '0' to '1' and the associated event notification is enabled in the Slot Control register.	RW1C	0x0
1	SYS_ERR	Link0 system error. The bit is set when any device in the hierarchy reports any of the following errors and the associated enable bit is set in the Root Control register: ERR_COR, ERR_FATAL, ERR_NONFATAL. It is also set when an internal error is detected. (RC Mode Only)	RW1C	0x0
0	RST_REQ	Link0 reset request. Reset request due to Link down status.	RW1C	0x0

19.8.3.38 PCIE_LINK1_INT_STAxxTUS1
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0AC

Bit#	Name	Description	Attribute	Reset
31:12	Reserved	Reserved		0x0000000
11	PM_TURNOFF	Link1 PME Turnoff message received interrupt. (EP Mode Only)	RW1C	0x0
10	PM_TO_ACK	Link1 PME Turnoff Acknowledge message received interrupt. (RC Mode Only)	RW1C	0x0
9	PM_PME	Link1 PME Message received interrupt. (RC Mode Only)	RW1C	0x0
8	ERR_FATAL	Link1 fatal error message received interrupt. (RC Mode Only)	RW1C	0x0
7	ERR_NONFATAL	Link1 non-fatal error message received interrupt. (RC Mode Only)	RW1C	0x0
6	ERR_COR	Link1 correctable error message received interrupt. (RC Mode Only)	RW1C	0x0
5	PME_MSI	Link1 PME MSI. The bit is set when the PME status bit in the Root Status register is set to '1', the Interrupt Enable bit in the Root Control register is set to '1', and MSI is enabled. (RC Mode Only)	RW1C	0x0
4	AER_RC_ERR_MSI	Link1 advanced error MSI. The bit is set when an error condition causes a bit to be set in the Root Error Status register, the associated error message reporting enable bit is set in the Root Error Command register, and MSI is enabled (RC Mode Only)	RW1C	0x0
3	HP_INT	Link1 hot-plug MSI. This bit is set if the Hot-Plug interrupts are enabled in the Slot Control register, if any bit in the Slot Status register transitions from '0' to '1', and MSI is enabled.	RW1C	0x0
2	HP_PME	Link1 hot-plug PME wake generation. This bit is set if the PME Enable bit in the Power Management Control and Status register is set to '1', and if any bit in the Slot Status register transitions from '0' to '1' and the associated event notification is enabled in the Slot Control register.	RW1C	0x0
1	SYS_ERR	Link1 system error. The bit is set when any device in the hierarchy reports any of the following errors and the associated enable bit is set in the Root Control register: ERR_COR, ERR_FATAL, ERR_NONFATAL. It is also set when an internal error is detected. (RC Mode Only)	RW1C	0x0
0	RST_REQ	Link1 reset request. Reset request due to Link down status.	RW1C	0x0

19.8.3.39 PCIE_LINK0_INT_ENABLE0

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0B0

Bit#	Name	Description	Attribute	Reset
31:7	Reserved	Reserved		0x00000000
6	HP_INT	Link0 hot-plug interrupt enable.	R/W	0x0
5	PME_INT	Link0 PME interrupt enable. (RC Mode Only)	R/W	0x0
4	AER_RC_ERR_INT	Link0 advanced error interrupt enable. (RC Mode Only)	R/W	0x0
3	INTD	Link0 INTD. (RC Mode Only)	R/W	0x0
2	INTC	Link0 INTC. (RC Mode Only)	R/W	0x0
1	INTB	Link0 INTB. (RC Mode Only)	R/W	0x0
0	INTA	Link0 INTA. (RC Mode Only)	R/W	0x0

19.8.3.40 PCIE_LINK1_INT_ENABLE0

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0B4

Bit#	Name	Description	Attribute	Reset
31:7	Reserved	Reserved		0x00000000
6	HP_INT	Link1 hot-plug interrupt enable.	R/W	0x0
5	PME_INT	Link1 PME interrupt enable. (RC Mode Only)	R/W	0x0
4	AER_RC_ERR_INT	Link1 advanced error interrupt enable. (RC Mode Only)	R/W	0x0
3	INTD	Link1 INTD. (RC Mode Only)	R/W	0x0
2	INTC	Link1 INTC. (RC Mode Only)	R/W	0x0
1	INTB	Link1 INTB. (RC Mode Only)	R/W	0x0
0	INTA	Link1 INTA. (RC Mode Only)	R/W	0x0

19.8.3.41 PCIE_LINK0_INT_ENABLE1
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0B8

Bit#	Name	Description	Attribute	Reset
31:12	Reserved	Reserved		0x0000000
11	PM_TURNOFF	Link0 PME Turnoff message received interrupt enable. (EP Mode Only)	R/W	0x0
10	PM_TO_ACK	Link0 PME Turnoff Acknowledge message received interrupt enable. (RC Mode Only)	R/W	0x0
9	PM_PME	Link0 PME Message received interrupt enable. (RC Mode Only)	R/W	0x0
8	ERR_FATAL	Link0 fatal error message received interrupt enable. (RC Mode Only)	R/W	0x0
7	ERR_NONFATAL	Link0 non-fatal error message received interrupt enable. (RC Mode Only)	R/W	0x0
6	ERR_COR	Link0 correctable error message received interrupt enable. (RC Mode Only)	R/W	0x0
5	PME_MSI	Link0 PME MSI enable. (RC Mode Only)	R/W	0x0
4	AER_RC_ERR_MSI	Link0 advanced error MSI enable. (RC Mode Only)	R/W	0x0
3	HP_MSI	Link0 hot-plug MSI enable.	R/W	0x0
2	HP_PME	Link0 hot-plug PME wake generation enable.	R/W	0x0
1	SYS_ERR	Link0 system error interrupt enable. (RC Mode Only)	R/W	0x0
0	RST_REQ	Link0 reset request enable.	R/W	0x0

19.8.3.42 PCIE_LINK1_INT_ENABLE1

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0BC

Bit#	Name	Description	Attribute	Reset
31:12	Reserved	Reserved		0x0000000
11	PM_TURNOFF	Link1 PME Turnoff message received interrupt enable. (EP Mode Only)	R/W	0x0
10	PM_TO_ACK	Link1 PME Turnoff Acknowledge message received interrupt enable. (RC Mode Only)	R/W	0x0
9	PM_PME	Link1 PME Message received interrupt enable. (RC Mode Only)	R/W	0x0
8	ERR_FATAL	Link1 fatal error message received interrupt enable. (RC Mode Only)	R/W	0x0
7	ERR_NONFATAL	Link1 non-fatal error message received interrupt enable. (RC Mode Only)	R/W	0x0
6	ERR_COR	Link1 correctable error message received interrupt enable. (RC Mode Only)	R/W	0x0
5	PME_MSI	Link1 PME MSI enable. (RC Mode Only)	R/W	0x0
4	AER_RC_ERR_MSI	Link1 advanced error MSI enable. (RC Mode Only)	R/W	0x0
3	HP_INT	Link1 hot-plug MSI enable.	R/W	0x0
2	HP_PME	Link1 hot-plug PME wake generation enable.	R/W	0x0
1	SYS_ERR	Link1 system error interrupt enable. (RC Mode Only)	R/W	0x0
0	RST_REQ	Link1 reset request enable.	R/W	0x0

19.8.3.43 PCIE_PHY_CR_CMD

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0C0

Bit#	Name	Description	Attribute	Reset
31:17	Reserved	Reserved		0x0000
16	CR_CMD	PHY control register command: 0: read 1: write	R/W	0x0
15:0	CR_ADDR	PHY control register address.	R/W	0x0000

19.8.3.44 PCIE_PHY_CR_WR_DATA
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0C4

Bit#	Name	Description	Attribute	Reset
31:16	Reserved	Reserved		0x0000
15:0	CR_WR_DATA	PHY control register write data.	R/W	0x0000

19.8.3.45 PCIE_PHY_CR_RESP
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0C8

Bit#	Name	Description	Attribute	Reset
31:1	Reserved	Reserved		0x00000000
0	CR_CMD_DONE	Indicates PHY control command is done.	RW1C	0x0

19.8.3.46 PCIE_PHY_CR_RD_DATA
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0CC

Bit#	Name	Description	Attribute	Reset
31:16	Reserved	Reserved		0x0000
15:0	CR_RD_DATA	PHY control register read data.	R/W	0x0000

19.8.3.47 PCIE_IOBM_ERR_CMD
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0D0

Bit#	Name	Description	Attribute	Reset
31:7	Reserved	Reserved		0x00000000
6	MULTIPLE	IOB master multiple error occurred.	RO	0x0
5:3	ADDRSPACE	IOB master error request address space.	RO	0x0
2:0	CMD	IOB master error request command.	RO	0x0

19.8.3.48 PCIE_IOBM_ERR_LOWER_ADDR
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0D4

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_ERR_LOWER_ADDR	IOB master request first error address bits 36:5	RO	0x00000000

19.8.3.49 PCIE_IOBM_ERR_UPPER_ADDR

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0D8

Bit#	Name	Description	Attribute	Reset
31:3	Reservedxx	Reserved		0x00000000
2:0	IOBM_ERR_UPPER_ADDR	IOBM master request first error upper address bits [39:37]	RO	0x0

19.8.3.50 PCIE_IOBM_ERR_BE

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0DC

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_ERR_BYTE_ENABLE	IOB master request first error byte enable.	RO	0x00000000

19.8.3.51 Reserved

Address Offset: 0x0E0 through 0x17C

Bit#	Name	Description	Attribute	Reset
31:0	Reserved	Reserved	R/W	0x00

19.8.3.52 PCIE_LINK2_STATE

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x180

Bit#	Name	Description	Attribute	Reset
31:23	Reserved			0x0
22:18	LTSSM_STATE	Link2 current LTSSM state.	RO	0x00
17:15	PM_STATE	Link2 current power state.	RO	0x0
14	RDLH_LINK_UP	Link2 data link layer up.	RO	0x0
13	XMLH_LINK_UP	Link2 PHY link up.	RO	0x0
12:8	DEVICE_NUMBER	Link2 PCIE device number.	RO	0x00
7:0	BUS_NUMBER	Link2 PCIE bus number.	RO	0x00

19.8.3.53 PCIE_LINK3_STATE
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x184

Bit#	Name	Description	Attribute	Reset
31:23	Reserved			0x00
22:18	LTSSM_STATE	Link3 current LTSSM state.	RO	0x00
17:15	PM_STATE	Link3 current power state.	RO	0x0
14	RDLH_LINK_UP	Link3 data link layer up.	RO	0x0
13	XMLH_LINK_UP	Link3 PHY link up.	RO	0x0
12:8	DEVICE_NUMBER	Link3 PCIE device number.	RO	0x00
7:0	BUS_NUMBER	Link3 PCIE bus number.	RO	0x00

19.8.3.54 PCIE_LINK2_MSI_STATUS
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x190

Bit#	Name	Description	Attribute	Reset
31:0	MSI_STATUS	PCIE Link0 MSI status. When a MSI is received on PCIE link2, the corresponding bit is set for the MSI vector; bit [0] is for vector 1, bit [1] is for vector 2 and so on. (RC Mode Only)	RW1C	0x00000000

19.8.3.55 PCIE_LINK3_MSI_STATUS
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x194

Bit#	Name	Description	Attribute	Reset
31:0	MSI_STATUS	PCIE Link1 MSI status. When a MSI is received on PCIE link3, the corresponding bit is set for the MSI vector; bit [0] is for vector 1, bit [1] is for vector 2 and so on. (RC Mode Only)	RW1C	0x00000000

19.8.3.56 PCIE_LINK2_MSI_ENABLE
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x0198

Bit#	Name	Description	Attribute	Reset
31:0	MSI_ENABLE	PCIE Link2 MSI enables. (RC Mode Only)	R/W	0x00000000

19.8.3.57 PCIE_LINK3_MSI_ENABLE
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x019C

Bit#	Name	Description	Attribute	Reset
31:0	MSI_ENABLE	PCIE Link3 MSI enables. (RC Mode Only)	R/W	0x00000000

19.8.3.58 PCIE_LINK2_INT_STATUS0

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x1A0

Bit#	Name	Description	Attribute	Reset
31:7	Reserved			0x0000000
6	HP_INT	Link2 hot-plug interrupt. This bit is set if the Hot-Plug interrupts are enabled in the Slot Control register, any bit in the Slot Status register is set to '1' and the the INTX Assertion Disable bit in the command register is '0'.	RO	0x0
5	PME_INT	Link2 PME interrupt. The bit is set when the PME status bit in the Root Status register is set to '1', the Interrupt Enable bit in the Root Control register is set to '1', and the INTX Assertion Disable bit in the Command register is '0'. (RC Mode Only)	RO	0x0
4	AER_RC_ERR_INT	Link2 advanced error interrupt. The bit is set when an error condition causes a bit to be set in the Root Error Status register and the associated error message reporting enable bit is set in the Root Error Command register. (RC Mode Only)	RO	0x0
3	INTD	Link2 INTD. (RC Mode Only)	RO	0x0
2	INTC	Link2 INTC. (RC Mode Only)	RO	0x0
1	INTB	Link2 INTB. (RC Mode Only)	RO	0x0
0	INTA	Link2 INTA. (RC Mode Only)	RO	0x0

19.8.3.59 PCIE_LINK3_INT_STATUS0

(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x1A4

Bit#	Name	Description	Attribute	Reset
31:7	Reserved			0x0000000
6	HP_INT	Link3 hot-plug interrupt. This bit is set if the Hot-Plug interrupts are enabled in the Slot Control register, any bit in the Slot Status register is set to '1' and the the INTX Assertion Disable bit in the command register is '0'.	RO	0x0
5	PME_INT	Link3 PME interrupt. The bit is set when the PME status bit in the Root Status register is set to '1', the Interrupt Enable bit in the Root Control register is set to '1', and the INTX Assertion Disable bit in the Command register is '0'. (RC Mode Only)	RO	0x0
4	AER_RC_ERR_INT	Link3 advanced error interrupt. The bit is set when an error condition causes a bit to be set in the Root Error Status register and the associated error message reporting enable bit is set in the Root Error Command register. (RC Mode Only)	RO	0x0
3	INTD	Link3 INTD. (RC Mode Only)	RO	0x0
2	INTC	Link3 INTC. (RC Mode Only)	RO	0x0
1	INTB	Link3 INTB. (RC Mode Only)	RO	0x0
0	INTA	Link3 INTA. (RC Mode Only)	RO	0x0

19.8.3.60 PCIE_LINK2_INT_STATUS1
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x1A8

Bit#	Name	Description	Attribute	Reset
31:12	Reserved			0x00000000
11	PM_TURNOFF	Link2 PME Turnoff message received interrupt. (EP Mode Only)	RW1C	0x0
10	PM_TO_ACK	Link2 PME Turnoff Acknowledge message received interrupt. (RC Mode Only)	RW1C	0x0
9	PM_PME	Link2 PME Message received interrupt. (RC Mode Only)	RW1C	0x0
8	ERR_FATAL	Link2 fatal error message received interrupt. (RC Mode Only)	RW1C	0x0
7	ERR_NONFATAL	Link2 non-fatal error message received interrupt. (RC Mode Only)	RW1C	0x0
6	ERR_COR	Link2 correctable error message received interrupt. (RC Mode Only)	RW1C	0x0
5	PME_MSI	Link2 PME MSI. The bit is set when the PME status bit in the Root Status register is set to '1', the Interrupt Enable bit in the Root Control register is set to '1', and MSI is enabled. (RC Mode Only)	RW1C	0x0
4	AER_RC_ERR_MSI	Link2 advanced error MSI. The bit is set when an error condition causes a bit to be set in the Root Error Status register, the associated error message reporting enable bit is set in the Root Error Command register, and MSI is enabled (RC Mode Only)	RW1C	0x0
3	HP_MSI	Link2 hot-plug MSI. This bit is set if the Hot-Plug interrupts are enabled in the Slot Control register, if any bit in the Slot Status register transitions from '0' to '1', and MSI is enabled.	RW1C	0x0
2	HP_PME	Link2 hot-plug PME wake generation. This bit is set if the PME Enable bit in the Power Management Control and Status register is set to '1', and if any bit in the Slot Status register transitions from '0' to '1' and the associated event notification is enabled in the Slot Control register.	RW1C	0x0
1	SYS_ERR	Link2 system error. The bit is set when any device in the hierarchy reports any of the following errors and the associated enable bit is set in the Root Control register: ERR_COR, ERR_FATAL, ERR_NONFATAL. It is also set when an internal error is detected. (RC Mode Only)	RW1C	0x0
0	RST_REQ	Link2 reset request. Reset request due to Link down status.	RW1C	0x0

19.8.3.61 PCIE_LINK3_INT_STATUS1**(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)**

Address Offset: 0x1AC

Bit#	Name	Description	Attribute	Reset
31:12	Reserved			0x00000000
11	PM_TURNOFF	Link3 PME Turnoff message received interrupt. (EP Mode Only)	RW1C	0x0
10	PM_TO_ACK	Link3 PME Turnoff Acknowledge message received interrupt. (RC Mode Only)	RW1C	0x0
9	PM_PME	Link3 PME Message received interrupt. (RC Mode Only)	RW1C	0x0
8	ERR_FATAL	Link3 fatal error message received interrupt. (RC Mode Only)	RW1C	0x0
7	ERR_NONFATAL	Link3 non-fatal error message received interrupt. (RC Mode Only)	RW1C	0x0
6	ERR_COR	Link3 correctable error message received interrupt. (RC Mode Only)	RW1C	0x0
5	PME_MSI	Link3 PME MSI. The bit is set when the PME status bit in the Root Status register is set to '1', the Interrupt Enable bit in the Root Control register is set to '1', and MSI is enabled. (RC Mode Only)	RW1C	0x0
4	AER_RC_ERR_MSI	Link3 advanced error MSI. The bit is set when an error condition causes a bit to be set in the Root Error Status register, the associated error message reporting enable bit is set in the Root Error Command register, and MSI is enabled (RC Mode Only)	RW1C	0x0
3	HP_MSI	Link3 hot-plug MSI. This bit is set if the Hot-Plug interrupts are enabled in the Slot Control register, if any bit in the Slot Status register transitions from '0' to '1', and MSI is enabled.	RW1C	0x0
2	HP_PME	Link3 hot-plug PME wake generation. This bit is set if the PME Enable bit in the Power Management Control and Status register is set to '1', and if any bit in the Slot Status register transitions from '0' to '1' and the associated event notification is enabled in the Slot Control register.	RW1C	0x0
1	SYS_ERR	Link3 system error. The bit is set when any device in the hierarchy reports any of the following errors and the associated enable bit is set in the Root Control register: ERR_COR, ERR_FATAL, ERR_NONFATAL. It is also set when an internal error is detected. (RC Mode Only)	RW1C	0x0
0	RST_REQ	Link3 reset request. Reset request due to Link down status.	RW1C	0x0

19.8.3.62 PCIE_LINK2_INT_ENABLE0
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x1B0

Bit#	Name	Description	Attribute	Reset
31:7	<i>Reserved</i>			0x00000000
6	HP_INT	Link2 hot-plug interrupt enable.	R/W	0x0
5	PME_INT	Link2 PME interrupt enable. (RC Mode Only)	R/W	0x0
4	AER_RC_ERR_INT	Link2 advanced error interrupt enable. (RC Mode Only)	R/W	0x0
3	INTD	Link2 INTD. (RC Mode Only)	R/W	0x0
2	INTC	Link2 INTC. (RC Mode Only)	R/W	0x0
1	INTB	Link2 INTB. (RC Mode Only)	R/W	0x0
0	INTA	Link2 INTA. (RC Mode Only)	R/W	0x0

19.8.3.63 PCIE_LINK3_INT_ENABLE0
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x1B4

Bit#	Name	Description	Attribute	Reset
31:7	<i>Reserved</i>			0x00000000
6	HP_INT	Link3 hot-plug interrupt enable.	R/W	0x0
5	PME_INT	Link3 PME interrupt enable. (RC Mode Only)	R/W	0x0
4	AER_RC_ERR_INT	Link3 advanced error interrupt enable. (RC Mode Only)	R/W	0x0
3	INTD	Link3 INTD. (RC Mode Only)	R/W	0x0
2	INTC	Link3 INTC. (RC Mode Only)	R/W	0x0
1	INTB	Link3 INTB. (RC Mode Only)	R/W	0x0
0	INTA	Link3 INTA. (RC Mode Only)	R/W	0x0

19.8.3.64 PCIE_LINK2_INT_ENABLE1
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0x1B8

Bit#	Name	Description	Attribute	Reset
31:12	<i>Reserved</i>			0x00000000
11	PM_TURNOFF	Link2 PME Turnoff message received interrupt enable. (EP Mode Only)	R/W	0x0
10	PM_TO_ACK	Link2 PME Turnoff Acknowledge message received interrupt enable. (RC Mode Only)	R/W	0x0
9	PM_PME	Link2 PME Message received interrupt enable. (RC Mode Only)	R/W	0x0
8	ERR_FATAL	Link2 fatal error message received interrupt enable. (RC Mode Only)	R/W	0x0

Bit#	Name	Description	Attribute	Reset
7	ERR_NONFATAL	Link2 non-fatal error message received interrupt enable. (RC Mode Only)	R/W	0x0
6	ERR_COR	Link2 correctable error message received interrupt enable. (RC Mode Only)	R/W	0x0
5	PME_MSI	Link2 PME MSI enable. (RC Mode Only)	R/W	0x0
4	AER_RC_ERR_MSI	Link2 advanced error MSI enable. (RC Mode Only)	R/W	0x0
3	HP_MSI	Link2 hot-plug MSI enable.	R/W	0x0
2	HP_PME	Link2 hot-plug PME wake generation enable.	R/W	0x0
1	SYS_ERR	Link2 system error interrupt enable. (RC Mode Only)	R/W	0x0
0	RST_REQ	Link2 reset request enable.	R/W	0x0

**19.8.3.65 PCIE_LINK3_INT_ENABLE1
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)**

Address Offset: 0x1BC

Bit#	Name	Description	Attribute	Reset
31:12	Reserved			0x0000000
11	PM_TURNOFF	Link3 PME Turnoff message received interrupt enable. (EP Mode Only)	R/W	0x0
10	PM_TO_ACK	Link3 PME Turnoff Acknowledge message received interrupt enable. (RC Mode Only)	R/W	0x0
9	PM_PME	Link3 PME Message received interrupt enable. (RC Mode Only)	R/W	0x0
8	ERR_FATAL	Link3 fatal error message received interrupt enable. (RC Mode Only)	R/W	0x0
7	ERR_NONFATAL	Link3 non-fatal error message received interrupt enable. (RC Mode Only)	R/W	0x0
6	ERR_COR	Link3 correctable error message received interrupt enable. (RC Mode Only)	R/W	0x0
5	PME_MSI	Link3 PME MSI enable. (RC Mode Only)	R/W	0x0
4	AER_RC_ERR_MSI	Link3 advanced error MSI enable. (RC Mode Only)	R/W	0x0
3	HP_MSI	Link3 hot-plug MSI enable.	R/W	0x0
2	HP_PME	Link3 hot-plug PME wake generation enable.	R/W	0x0
1	SYS_ERR	Link3 system error interrupt enable. (RC Mode Only)	R/W	0x0
0	RST_REQ	Link3 reset request enable.	R/W	0x0

**19.8.3.66 VC0_POSTED_RX_QUEUE_CTRL
(For XLS6xx, XLS4xx, XLS2xx, and XLS1xx)**

Address Offset: 0x748

This register sets ordering rules and queue mode for TLPs and effectively sets the size of buffers used for packet headers and data.

The reset value of this register should not be changed, except for the XLS4xxLite series (as described in [Section 19.8.4.50, “VC0_POSTED_RX_QUEUE_CTRL”](#)).

**19.8.3.67 VC0_POSTED_BUFFER_DEPTH
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)**

Address Offset: 0x7A8

This register defines the VC0 posted data queue depth and header queue depth.

The reset value in this register should not be changed, except for the XLS4xxLite series (as described in [Section 19.8.4.51, “VC0_POSTED_BUFFER_DEPTH”](#)).

**19.8.3.68 PCIE_MSG_TX_THRESHOLD
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)**

Address Offset: 0xC20

Bit#	Name	Description	Attribute	Reset
31:0	MSG_TX_THRESHOLD	Message transmit credit threshold size.	R/W	0x00000003

**19.8.3.69 PCIE_MSG_BUCKET_SIZE_0
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)**

Address Offset: 0xC80

Bit#	Name	Description	Attribute	Reset
31:9	Reserved			0x000000
8:0	MSG_BUCKET_SIZE_0	Size of the message bucket for link0 DMA Load.	R/W	0x020

**19.8.3.70 PCIE_MSG_BUCKET_SIZE_1
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)**

Address Offset: 0xC84

Bit#	Name	Description	Attribute	Reset
31:9	Reserved			0x000000
8:0	MSG_BUCKET_SIZE_1	Size of the message bucket for link0 DMA Store.	R/W	0x020

**19.8.3.71 PCIE_MSG_BUCKET_SIZE_2
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)**

Address Offset: 0xC88

Bit#	Name	Description	Attribute	Reset
31:9	<i>Reserved</i>			0x000000
8:0	MSG_BUCKET_SIZE_2	Size of the message bucket for link1 DMA Load.	R/W	0x020

19.8.3.72 PCIE_MSG_BUCKET_SIZE_3
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0xC8C

Bit#	Name	Description	Attribute	Reset
31:9	<i>Reserved</i>			0x000000
8:0	MSG_BUCKET_SIZE_3	Size of the message bucket for link1 DMA Store.	R/W	0x020

19.8.3.73 PCIE_MSG_BUCKET_SIZE_4
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0xC90

Bit#	Name	Description	Attribute	Reset
31:9	<i>Reserved</i>			0x000000
8:0	MSG_BUCKET_SIZE_4	Size of the message bucket for link2 DMA Load.	R/W	0x020

19.8.3.74 PCIE_MSG_BUCKET_SIZE_5
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0xC94

Bit#	Name	Description	Attribute	Reset
31:9	<i>Reserved</i>			0x000000
8:0	MSG_BUCKET_SIZE_5	Size of the message bucket for link2 DMA Store.	R/W	0x020

19.8.3.75 PCIE_MSG_BUCKET_SIZE_6
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)

Address Offset: 0xC98

Bit#	Name	Description	Attribute	Reset
31:9	<i>Reserved</i>			0x000000
8:0	MSG_BUCKET_SIZE_6	Size of the message bucket for link3 DMA Load.	R/W	0x020

**19.8.3.76 PCIE_MSG_BUCKET_SIZE_7
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)**

Address Offset: 0xC9C

Bit#	Name	Description	Attribute	Reset
31:9	<i>Reserved</i>			0x000000
8:0	MSG_BUCKET_SIZE_7	Size of the message bucket for link3 DMA Store.	R/W	0x020

**19.8.3.77 PCIE_MSG_CREDIT
(For XLS6xx, XLS4xx, XLS2xx and XLS1xx)**

Address Offset: 0xE00-0xFFC

Bit#	Name	Description	Attribute	Reset
31:9	<i>Reserved</i>			0x000000
8:0	MSG_CREDIT	Credit Counter Registers.	R/W	0x000

19.8.4**Summary of the XLS408Lite & XLS404Lite Global PCIE Registers**

This section describes the PCI Express Interface Global Registers for the XLS408Lite and XLS404Lite. For register descriptions for the XLS6xx, XLS4xx, XLS2xx and XLS1xx families, see [page 864](#).

Table 19-13. Summary of XLS408Lite and XLS404Lite PCI Express Interface Global Registers

Register Offset DWord Addressing	Register Offset Byte Addressing	Name	R/W	Reset Value
0x000	0x000	PCIE_CTRL1	WO/R/W	0x00000000
0x001	0x004	PCIE_CTRL2	R/W	0x00000000
0x002	0x008	PCIE_CTRL3	R/W	0x000C1000
0x003	0x00C	PCIE_CTRL4	R/W	0xAAAAA2222
0x004	0x010	PCIE_CTRL	R/W	0x00000000
0x005	0x014	PCIE_IOBM_TIMER	R/W	0xFFFFFFFF
0x006	0x018	PCIE_MSI_CMD	R/W	0x00000000
0x007	0x01C	PCIE_MSI_RESP	RW1C	0x00000000
0x009-0x00F	0x024-0x03C	Reserved		
0x010	0x040	PCIE_IOBM_SWAP_MEM_BASE	R/W	0xFFFFFFFF
0x011	0x044	PCIE_IOBM_SWAP_MEM_LIMIT	R/W	0x00000000
0x012	0x048	PCIE_IOBM_SWAP_IO_BASE	R/W	0xFFFFFFFF
0x013	0x04C	PCIE_IOBM_SWAP_IO_LIMIT	R/W	0x00000000
0x014	0x050	PCIE_TRGT_COHERENT_MEM_BASE	R/W	0xFFFFFFFF
0x015	0x054	PCIE_TRGT_COHERENT_MEM_LIMIT	R/W	0x00000000
0x016	0x058	PCIE_TRGT_L2ALLOC_MEM_BASE	R/W	0xFFFFFFFF
0x017	0x05C	PCIE_TRGT_L2ALLOC_MEM_LIMIT	R/W	0x00000000
0x018	0x060	PCIE_TRGT_READEX_MEM_BASE	R/W	0xFFFFFFFF
0x019	0x064	PCIE_TRGT_READEX_MEM_LIMIT	R/W	0x00000000
0x01A	0x068	PCIE_EP_MEM_BASE	R/W	0xFFFFFFFF
0x01B	0x06C	PCIE_EP_MEM_LIMIT	R/W	0x00000000
0x01C	0x070	PCIE_EP_ADDR_MAP_ENTRY0	R/W	0x00000000
0x01D	0x074	PCIE_EP_ADDR_MAP_ENTRY1	R/W	0x00000000
0x01E	0x078	PCIE_EP_ADDR_MAP_ENTRY2	R/W	0x00000000
0x01F	0x07C	PCIE_EP_ADDR_MAP_ENTRY3	R/W	0x00000000
0x020	0x080	PCIE_LINK0_STATE	RO	0x00000000
0x021	0x084	PCIE_LINK1_STATE	RO	0x00000000
0x022	0x088	PCIE_IOBM_INT_STATUS	RW1C	0x00000000
0x023	0x08C	PCIE_IOBM_INT_ENABLE	R/W	0x00000000
0x024	0x090	PCIE_LINK0_MSI_STATUS	RW1C	0x00000000
0x025	0x094	PCIE_LINK1_MSI_STATUS	RW1C	0x00000000
0x026	0x098	PCIE_LINK0_MSI_ENABLE	R/W	0x00000000
0x027	0x09C	PCIE_LINK1_MSI_ENABLE	R/W	0x00000000
0x028	0x0A0	PCIE_LINK0_INT_STATUS0	RO	0x00000000
0x029	0x0A4	PCIE_LINK1_INT_STATUS0	RO	0x00000000

Table 19-13. Summary of XLS408Lite and XLS404Lite PCI Express Interface Global Registers (continued)

Register Offset DWord Addressing	Register Offset Byte Addressing	Name	R/W	Reset Value
0x02A	0x0A8	PCIE_LINK0_INT_STATUS1	RW1C	0x00000000
0x02B	0x0AC	PCIE_LINK1_INT_STATUS1	RW1C	0x00000000
0x02C	0x0B0	PCIE_LINK0_INT_ENABLE0	R/W	0x00000000
0x02D	0x0B4	PCIE_LINK1_INT_ENABLE0	R/W	0x00000000
0x02E	0x0B8	PCIE_LINK0_INT_ENABLE1	R/W	0x00000000
0x02F	0x0BC	PCIE_LINK1_INT_ENABLE1	R/W	0x00000000
0x030	0x0C0	PCIE_PHY_CR_CMD	R/W	0x00000000
0x031	0x0C4	PCIE_PHY_CR_WR_DATA	R/W	0x00000000
0x032	0x0C8	PCIE_PHY_CR_RESP	RW1C	0x00000000
0x033	0x0CC	PCIE_PHY_CR_RD_DATA	R/W	0x00000000
0x034	0x0D0	PCIE_IOBM_ERR_CMD	RO	0x00000000
0x035	0x0D4	PCIE_IOBM_ERR_LOWER_ADDR	RO	0x00000000
0x036	0x0D8	PCIE_IOBM_ERR_UPPER_ADDR	RO	0x00000000
0x037	0x0DC	PCIE_IOBM_ERR_BE	RO	0x00000000
0x038-0x3F	0xE0-0xC1C	Reserved		
0x308	0xC20	PCIE_MSG_TX_THRESHOLD	R/W	0x00000003
0x309-0x31F	0xC21-0xC7C	Reserved		
0x320	0xC80	PCIE_MSG_BUCKET_SIZE_0	R/W	0x00000040
0x321	0xC84	PCIE_MSG_BUCKET_SIZE_1	R/W	0x00000040
0x322	0xC88	PCIE_MSG_BUCKET_SIZE_2	R/W	0x00000040
0x323	0xC8C	PCIE_MSG_BUCKET_SIZE_3	R/W	0x00000040
0x324 – 0x37F	0xC90-0xDFC	Reserved		
0x380 – 0x3FF	0xE00-0xFFC	PCIE_MSG_CREDIT	R/W	0x00000000

19.8.4.1 PCIE_CTRL1
(For XLS408Lite and XLS404Lite)

Address Offset: 0x000

Bit#	Name	Description	Attribute	Reset
31:28	Reserved	Reserved		0
27	LINK1_APPS_PM_XMT_TURNOFF	Link1 request to generate a PM_TURN_OFF Message. (RC Mode Only)	WO	0
26	LINK1_APP_REQ_EXIT_L1	Link1 request to exit ASPM state L1.	R/W	0
25	LINK1_APP_REQ_ENTR_L1	Link1 request to enter ASPM state L1.	R/W	0
24	LINK1_SYS_AUX_PWR_DET	Indicates that Link1 auxiliary power is present.	R/W	0
23:22	Reserved	Reserved		0
21	LINK1_APP_INIT_RST	Link1 request to send a Hot Reset to the downstream device. (RC Mode Only)	R/W	0
20	Reserved	Reserved		0
19	LINK1_TX_LANE_FLIP_EN	Link1 request to perform transmit lane reversal.	R/W	0
18	LINK1_RX_LANE_FLIP_EN	Link1 request to perform receive lane reversal.	R/W	0
17	LINK1_APP_LTSSM_EN	Enable Link1 LTSSM.	R/W	0
16	LINK1_RST_N	Resets the Link1 core.	R/W	0
15:12	Reserved	Reserved		0
11	LINK0_APPS_PM_XMT_TURNOFF	Link0 request to generate a PM_TURN_OFF Message. (RC Mode Only)	WO	0
10	LINK0_APP_REQ_EXIT_L1	Link0 request to exit ASPM state L1.	R/W	0
9	LINK0_APP_REQ_ENTR_L1	Link0 request to enter ASPM state L1.	R/W	0
8	LINK0_SYS_AUX_PWR_DET	Indicates that Link0 auxiliary power is present.	R/W	0
7	LINK0_APPS_PM_XMT_PME	Link0 request to wake up the PMC state machine from the D3 state. (EP Mode Only)	WO	0
6	LINK0_OUTBAND_PWRUP_CMD	Link0 request to wake up the PMC state machine from the D3 state. (EP Mode Only)	WO	0
5	LINK0_APP_INIT_RST	Link0 request to send a Hot Reset to the downstream device. (RC Mode Only)	R/W	0
4	LINK0_APP_REQ_RETRY_EN	Link0 request to defer incoming Configuration Requests until initialization is complete. (EP Mode Only)	R/W	0
3	LINK0_TX_LANE_FLIP_EN	Link0 request to perform transmit lane reversal.	R/W	0
2	LINK0_RX_LANE_FLIP_EN	Link0 request to perform receive lane reversal.	R/W	0
1	LINK0_APP_LTSSM_EN	Enable Link0 LTSSM.	R/W	0
0	LINK0_RST_N	Resets the Link0 core.	R/W	0

19.8.4.2 PCIE_CTRL2 (For XLS408Lite and XLS404Lite)

Address Offset: 0x004

Bit#	Name	Description	Attribute	Reset
31:24	Reserved	Reserved		00
23	LINK1_EML_INTERLOCK_ENGAGED	Indicates whether the system electro-mechanical interlock is engaged for Link1.	R/W	0
22	LINK1_CMD_CPLED_INT	Indicates that the Hot-Plug controller completed a command for Link1.	R/W	0
21	LINK1_PRE_DET_CHGED	Indicates that the state of card present detector has changed for Link1.	R/W	0
20	LINK1_MRL_SENSOR_CHGED	Indicates that the state of MRL sensor has changed for Link1.	R/W	0
19	LINK1_PWR_FAULT_DET	Indicates the power controller detected a power fault at this slot for Link1.	R/W	0
18	LINK1_MRL_SENSOR_STATE	Indicates the state of the manually-operated retention latch (MRL) sensor for Link1: 0: MRL is closed 1: MRL is open	R/W	0
17	LINK1_PRE_DET_STATE	Indicates presence of a card in the slot for Link1: 0: Slot is empty 1: Card is present in the slot	R/W	0
16	LINK1_ATTN_BUTTON_PRESSED	Indicates that the system attention button was pressed for Link1. (RC Mode Only)	R/W	0
15:8	Reserved	Reserved		00
7	LINK0_EML_INTERLOCK_ENGAGED	Indicates whether the system electromechanical interlock is engaged for Link0.	R/W	0
6	LINK0_CMD_CPLED_INT	Indicates that the Hot-Plug controller completed a command for Link0.	R/W	0
5	LINK0_PRE_DET_CHGED	Indicates that the state of card present detector has changed for Link0.	R/W	0
4	LINK0_MRL_SENSOR_CHGED	Indicates that the state of MRL sensor has changed for Link0.	R/W	0
3	LINK0_PWR_FAULT_DET	Indicates the power controller detected a power fault at this slot for Link0.	R/W	0
2	LINK0_MRL_SENSOR_STATE	Indicates the state of the manually-operated retention latch (MRL) sensor for Link0: 0: MRL is closed 1: MRL is open	R/W	0
1	LINK0_PRE_DET_STATE	Indicates presence of a card in the slot for Link0: 0: Slot is empty 1: Card is present in the slot	R/W	0
0	LINK0_ATTN_BUTTON_PRESSED	Indicates that the system attention button was pressed for Link0. (RC Mode Only)	R/W	0

19.8.4.3 PCIE_CTRL3

(For XLS408Lite and XLS404Lite)

Address Offset: 0x008

Bit#	Name	Description	Attribute	Reset
31:21	Reserved	Reserved		0x000
20:16	PHY_TX_LVL	<p>Transmitter output level:</p> <p>5'b00000: 0.892V 5'b00001: 0.901V 5'b00010: 0.911V 5'b00011: 0.920V 5'b00100: 0.929V 5'b00101: 0.938V 5'b00110: 0.948V 5'b00111: 0.957V 5'b01000: 0.966V 5'b01001: 0.076V 5'b01010: 0.985V 5'b01011: 0.994V 5'b01100: 1.003V 5'b01101: 1.013V 5'b01110: 1.022V 5'b01111: 1.031V 5'b10000: 1.041V 5'b10001: 1.050V 5'b10010: 1.059V 5'b10011: 1.069V 5'b10100: 1.078V 5'b10101: 1.087V 5'b10110: 1.096V 5'b10111: 1.106V 5'b11000: 1.115V 5'b11001: 1.124V 5'b11010: 1.134V 5'b11011: 1.143V 5'b11100: 1.152V 5'b11101: 1.161V 5'b11110: 1.171V 5'b11111: 1.180V</p> <p>Note: This field is intended for device debug purposes. For normal operation, the default value should be used.</p>	R/W	0x0C
15:13	Reserved	Reserved		0x0
12:8	PHY_LOS_LVL	<p>Loss Of Signal detection sensitivity. The default setting is 16.</p> <p>Note: This field is intended for device debug purposes. For normal operation, the default value should be used.</p>	R/W	0x10
7:5	Reserved	Reserved		0x0
4:0	Reserved	Reserved		0x0

19.8.4.4 PCIE_CTRL4
(For XLS408Lite and XLS404Lite)

Address Offset: 0x00C

Bit#	Name	Description	Attribute	Reset
31:28	PHY3_TX_BOOST	Transmitter boost control for lane3. Programmed boost value is: boost = -20log(1-(phy3_tx_boost[3:0]+0.5)/32)dB, except setting tx_boost[3:0] to '0' truly produces 0 db of boost. This produces results up to 5.75 dB in steps of ~0.37dB. May transition asynchronously. Changes while the transmitter is on may briefly cause undesired transmit waveforms.	R/W	0xA
27:24	PHY2_TX_BOOST	Transmitter boost control for lane2.	R/W	0xA
23:20	PHY1_TX_BOOST	Transmitter boost control for lane1.	R/W	0xA
19:16	PHY0_TX_BOOST	Transmitter boost control for lane0.	R/W	0xA
15	Reserved	Reserved		0x0
14:12	PHY3_RX_EQ_VAL	Receiver equalization boost control for lane3. Internal linear equalizer boost is ~(phy3_rx_eq_val+1)*0.5dB. May transition asynchronously. Transitions may cause bit error as the boost changes values. 3'b000: 0.5dB 3'b001: 1dB 3'b010: 1.5dB 3'b011: 2dB 3'b100: 2.5dB 3'b101: 3dB 3'b110: 3.5dB 3'b111: 4dB	R/W	0x2
11	Reserved	Reserved		0x0
10:8	PHY2_RX_EQ_VAL	Receiver equalization boost control for lane2.	R/W	0x2
7	Reserved	Reserved		0x0
6:4	PHY1_RX_EQ_VAL	Receiver equalization boost control for lane1.	R/W	0x2
3	Reserved	Reserved		0x0
2:0	PHY0_RX_EQ_VAL	Receiver equalization boost control for lane0.	R/W	0x2

19.8.4.5 PCIE_CTRL

(For XLS408Lite and XLS404Lite)

Address Offset: 0x010

Bit#	Name	Description	Attribute	Reset
31	TRGT_TIMEOUT_EN	Target interface timeout enable; when this bit is set, the timeout for the target interface completion is enabled. Internal use only; show as Reserved for external documentation.	R/W	0x0
30:7	Reserved	Reserved		0x0000000
6	INT	Legacy interrupt pin; when the bit goes from low to high, the PCIe core generates an ASSERT_INTX Message; when the bit goes from high to low, the PCIe core generates an DEASSERT_INTX Message. (EP Mode Only)	R/W	0x0
5	LINK1_CFG_BAR_MASK_EN	Enables Link1 configuration of the BAR masks.	R/W	0x0
4	LINK0_CFG_BAR_MASK_EN	Enables Link1 configuration of the BAR masks.	R/W	0x0
3	ADDR_MAP_EN	Enables PCI to XLS address mapping. (EP Mode Only)	R/W	0x0
2	LPBK_EN	Enables PHY Rx to Tx parallel data loop-back.	R/W	0x0
1	LINK1_FLUSH	Request to flush all Link1 messages and IOB master requests and return error responses. The flush is used when the PCIe core must be reset but the XLS need not.	R/W	0x0
0	LINK0_FLUSH	Request to flush all Link0 messages and IOB master requests and return error responses. The flush is used when the PCIe core must be reset but the XLS need not.	R/W	0x0

19.8.4.6 PCIE_IOBM_TIMER

(For XLS408Lite and XLS404Lite)

Address Offset: 0x014

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_REQUEST_TIMER	Number of IodClk cycles for an unaccepted IOB master request to timeout	R/W	0xFFFFFFFF

19.8.4.7 PCIE_MSI_CMD
(For XLS408Lite and XLS404Lite)

Address Offset: 0x018

Bit#	Name	Description	Attribute	Reset
31:6	Reserved	Reserved		0x00000000
5	MSI_EN	1: Enables Link0 MSI. (EP Mode Only)	R/W	0x0
4:0	MSI	Link0 MSI vector. When the MSI_EN = '1' and when MSI_DONE = '0', an MSI request is generated with MSI vector. (EP Mode Only)	R/W	0x00

19.8.4.8 PCIE_MSI_RESP
(For XLS408Lite and XLS404Lite)

Address Offset: 0x01C

Bit#	Name	Description	Attribute	Reset
31:1	Reserved	Reserved		0x0
0	MSI_DONE	Indicates Link0 MSI request is done. (EP Mode Only)	RW1C	0

19.8.4.9 PCIE_IOBM_SWAP_MEM_BASE
(For XLS408Lite and XLS404Lite)

Address Offset: 0x040

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_SWAP_MEM_BASE	IOB Master request memory space base for byte swapping; the memory base addresses minimum memory size of 256 Bytes	R/W	0xFFFFFFFF

19.8.4.10 PCIE_IOBM_SWAP_MEM_LIMIT
(For XLS408Lite and XLS404Lite)

Address Offset: 0x044

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_SWAP_MEM_LIMIT	IOB Master request memory space limit for byte swapping; the memory limit addresses minimum memory size of 256 Bytes.	R/W	0x0

**19.8.4.11 PCIE_IOBM_SWAP_IO_BASE
(For XLS408Lite and XLS404Lite)**

Address Offset: 0x048

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_SWAP_IO_BASE	IOB Master request IO space base for byte swapping; the IO base addresses minimum memory size of 32B.	R/W	0xFFFFFFFF

**19.8.4.12 PCIE_IOBM_SWAP_IO_LIMIT
(For XLS408Lite and XLS404Lite)**

Address Offset: 0x04C

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_SWAP_IO_LIMIT	IOB Master request IO space limit for byte swapping; the IO limit addresses minimum memory size of 32B.	R/W	0x0

**19.8.4.13 PCIE_TRGT_COHERENT_MEM_BASE
(For XLS408Lite and XLS404Lite)**

Address Offset: 0x050

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_TRGT_COHERENT_MEM_BASE	PCIe link request memory space base for coherency; the memory base addresses minimum memory size of 256B.	R/W	0xFFFFFFFF

**19.8.4.14 PCIE_TRGT_COHERENT_MEM_LIMIT
(For XLS408Lite and XLS404Lite)**

Address Offset: 0x054

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_TRGT_COHERENT_MEM_LIMIT	PCIe link request memory space limit for coherency; the memory limit addresses minimum memory size of 256B.	R/W	0x0

**19.8.4.15 PCIE_TRGT_L2ALLOC_MEM_BASE
(For XLS408Lite and XLS404Lite)**

Address Offset: 0x058

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_TRGT_L2ALLOC_MEM_BASE	<p>PCIe link request memory space base for L2 allocation; the memory base addresses minimum memory size of 256B.</p> <p>Programming this base address and also the PCIE_TRGT_L2ALLOC_MEM_LIMIT register together defines a low (base) and high (limit) address range. See the description of the PCIE_TRGT_L2ALLOC_MEM_LIMIT register for a description of how this range affects PCIe accesses.</p>	R/W	0xFFFFFFFF

**19.8.4.16 PCIE_TRGT_L2ALLOC_MEM_LIMIT
(For XLS408Lite and XLS404Lite)**

Address Offset: 0x05C

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_TRGT_L2ALLOC_MEM_LIMIT	<p>PCIe link request memory space limit for L2 allocation; the memory limit addresses minimum memory size of 256B.</p> <p>Programming this limit address and also the PCIE_TRGT_L2ALLOC_MEM_BASE register together defines a low (base) and high (limit) address range.</p> <p>If the address of an incoming PCIe memory read transaction falls between the L2 Allocation memory base and limit defined by this range, and if the read data is not present in the L2 Cache, then the read data is stored in the L2 Cache.</p> <p>If the address of an incoming PCIe memory write transaction falls between the L2 Allocation memory base and limit defined by this range, the write data goes into the L2 cache.</p>	R/W	0x00000000

19.8.4.17 PCIE_TRGT_READEX_MEM_BASE
(For XLS408Lite and XLS404Lite)

Address Offset: 0x060

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_TRGT_READEX_MEM_BASE	<p>PCIe link request memory space base for read exclusive; the memory base addresses minimum memory size of 256B.</p> <p>Programming this base address and also the PCIE_TRGT_READEX_MEM_LIMIT register together defines a low (base) and high (limit) address range. See the description of the PCIE_TRGT_READEX_MEM_LIMIT register for a description of how this range affects PCIe accesses.</p>	R/W	0xFFFFFFFF

19.8.4.18 PCIE_TRGT_READEX_MEM_LIMIT
(For XLS408Lite and XLS404Lite)

Address Offset: 0x064

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_TRGT_READEX_MEM_LIMIT	<p>PCIe link request memory space limit for read exclusive; the memory limit addresses minimum memory size of 256B.</p> <p>Programming this limit address and also the PCIE_TRGT_READEX_MEM_BASE register together defines a low (base) and high (limit) address range.</p> <p>If the address of an incoming PCIe memory read transaction falls between the Read Exclusive memory base and limit defined by this range, and if the read data is present in the L2 Cache, then the read data is flushed from the L2 Cache.</p> <p>The READEX range has no impact on write transactions.</p>	R/W	0x00000000

19.8.4.19 PCIE_EP_MEM_BASE
(For XLS408Lite and XLS404Lite)

Address Offset: 0x068

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_EP_MEM_BASE	<p>PCIe EP link memory space base; the memory base addresses minimum memory size of 256B. (EP Mode Only)</p> <p>Programming this base address and also the PCIE_EP_MEM_LIMIT register together defines a low (base) and high (limit) address range. See the description of the PCIE_EP_MEM_LIMIT register for a description of how this range affects PCIe accesses.</p>	R/W	0xFFFFFFFF

19.8.4.20 PCIE_EP_MEM_LIMIT
(For XLS408Lite and XLS404Lite)

Address Offset: 0x06C

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_EP_MEM_LIMIT	<p>PCIe EP link memory space limit; the memory limit addresses minimum memory size of 256B. (EP Mode Only)</p> <p>Programming this limit address and also the PCIE_EP_MEM_BASE register together defines a low (base) and high (limit) address range.</p> <p>If the address of a CPU PCIe memory request falls between the EndPoint (EP) memory base and limit defined by this range, then the memory request should be sent from the Endpoint (EP) port.</p> <p>The EP base and limit registers are necessary because the XLS has multiple PCIe ports. Endpoint (EP) ports have BARs for incoming memory requests from the link, but no BARs for outgoing memory requests.</p> <p>For Root Complex (RC) ports, the memory base and limit are built into the type1 configuration header; thus PCIE_EP_MEM_BASE and PCIE_EP_MEM_LIMIT registers have no bearing on PCIe RC ports.</p>	R/W	0x00000000

19.8.4.21 PCIE_EP_ADDR_MAP_ENTRY0
(For XLS408Lite and XLS404Lite)

Address Offset: 0x070

Bit#	Name	Description	Attribute	Reset
31:15	IOBM_EP_ADDR_MAP_ENTRY0	PCIe EP link memory request address map entry 0. When ADDR_MAP_EN is set and when PCIe address [24:23] == 2'b00, the address map entry 0 is used as XLS address [39:23]. (EP Mode Only)	R/W	0x00000
14:0	Reserved			0x0000

19.8.4.22 PCIE_EP_ADDR_MAP_ENTRY1
(For XLS408Lite and XLS404Lite)

Address Offset: 0x074

Bit#	Name	Description	Attribute	Reset
31:15	IOBM_EP_ADDR_MAP_ENTRY1	PCIe EP link memory request address map entry 1. When ADDR_MAP_EN is set and when PCIe address [24:23] = '01', the address map entry 1 is used as XLS address [39:23]. (EP Mode Only)	R/W	0x00000
14:0	Reserved			0x0000

19.8.4.23 PCIE_EP_ADDR_MAP_ENTRY2
(For XLS408Lite and XLS404Lite)

Address Offset: 0x078

Bit#	Name	Description	Attribute	Reset
31:15	IOBM_EP_ADDR_MAP_ENTRY0	PCIe EP link memory request address map entry 0. When ADDR_MAP_EN is set and when PCIe address [24:23] = '10', the address map entry 2 is used as XLS address [39:23]. (EP Mode Only)	R/W	0x00000
14:0	Reserved			0x0000

**19.8.4.24 PCIE_EP_ADDR_MAP_ENTRY3
(For XLS408Lite and XLS404Lite)**

Address Offset: 0x07C

Bit#	Name	Description	Attribute	Reset
31:15	IOBM_EP_ADDR_MAP_ENTRY1	PCIe EP link memory request address map entry 1. When ADDR_MAP_EN is set and when PCIe address [24:23] = '11', the address map entry 3 is used as XLS address [39:23]. (EP Mode Only)	R/W	0x00000
14:0	Reserved			0x0000

**19.8.4.25 PCIE_LINK0_STATE
(For XLS408Lite and XLS404Lite)**

Address Offset: 0x080

Bit#	Name	Description	Attribute	Reset
31:23	Reserved	Reserved		0x00
22:18	LTSSM_STATE	Link0 current LTSSM state.	RO	0x00
17:15	PM_STATE	Link0 current power state.	RO	0x0
14	RDLH_LINK_UP	Link0 data link layer up.	RO	0x0
13	XMLH_LINK_UP	Link0 PHY link up.	RO	0x0
12:8	DEVICE_NUMBER	Link0 PCIe device number.	RO	0x00
7:0	BUS_NUMBER	Link0 PCIe bus number.	RO	0x00

**19.8.4.26 PCIE_LINK1_STATE
(For XLS408Lite and XLS404Lite)**

Address Offset: 0x084

Bit#	Name	Description	Attribute	Reset
31:23	Reserved	Reserved		0x00
22:18	LTSSM_STATE	Link1 current LTSSM state.	RO	0x00
17:15	PM_STATE	Link1 current power state.	RO	0x0
14	RDLH_LINK_UP	Link1 data link layer up.	RO	0x0
13	XMLH_LINK_UP	Link1 PHY link up.	RO	0x0
12:8	DEVICE_NUMBER	Link1 PCIe device number.	RO	0x00
7:0	BUS_NUMBER	Link1 PCIe bus number.	RO	0x00

19.8.4.27 PCIE_IOBM_INT_STATUS

Address Offset: 0x088

Bit#	Name	Description	Attribute	Reset
31:30	Reserved	Reserved		0x00000000
0	IOBM_INT_STATUS	IOB Master request interrupt.	RW1C	0x0

19.8.4.28 PCIE_IOBM_INT_ENABLE

Address Offset: 0x08C

Bit#	Name	Description	Attribute	Reset
31:30	Reserved	Reserved		0x00000000
0	IOBM_INT_ENABLE	IOB Master request interrupt enable.	R/W	0x0

19.8.4.29 PCIE_LINK0_MSI_STATUS

(For XLS408Lite and XLS404Lite)

Address Offset: 0x090

Bit#	Name	Description	Attribute	Reset
31:0	MSI_STATUS	PCIe Link0 MSI status. When a MSI is received on PCIe link0, the corresponding bit is set for the MSI vector; bit [0] is for vector 1, bit [1] is for vector 2 and so on. (RC Mode Only)	RW1C	0x00000000

19.8.4.30 PCIE_LINK1_MSI_STATUS

(For XLS408Lite and XLS404Lite)

Address Offset: 0x094

Bit#	Name	Description	Attribute	Reset
31:0	MSI_STATUS	PCIe Link1 MSI status. When a MSI is received on PCIe link1, the corresponding bit is set for the MSI vector; bit [0] is for vector 1, bit [1] is for vector 2 and so on. (RC Mode Only)	RW1C	0x00000000

19.8.4.31 PCIE_LINK0_MSI_ENABLE

(For XLS408Lite and XLS404Lite)

Address Offset: 0x098

Bit#	Name	Description	Attribute	Reset
31:0	MSI_ENABLE	PCIe Link0 MSI enables. (RC Mode Only)	R/W	0x00000000

19.8.4.32 PCIE_LINK1_MSI_ENABLE
(For XLS408Lite and XLS404Lite)

Address Offset: 0x09C

Bit#	Name	Description	Attribute	Reset
31:0	MSI_ENABLE	PCIe Link1 MSI enables. (RC Mode Only)	R/W	0x00000000

19.8.4.33 PCIE_LINK0_INT_STATUS0
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0A0

Bit#	Name	Description	Attribute	Reset
31:6	Reserved	Reserved		0x00000000
5	PME_INT	Link0 PME interrupt. The bit is set when the PME status bit in the Root Status register is set to '1', the Interrupt Enable bit in the Root Control register is set to '1', and the INTX Assertion Disable bit in the Command register is '0'. (RC Mode Only)	RO	0x0
4	AER_RC_ERR_INT	Link0 advanced error interrupt. The bit is set when an error condition causes a bit to be set in the Root Error Status register and the associated error message reporting enable bit is set in the Root Error Command register. (RC Mode Only)	RO	0x0
3	INTD	Link0 INTD. (RC Mode Only)	RO	0x0
2	INTC	Link0 INTC. (RC Mode Only)	RO	0x0
1	INTB	Link0 INTB. (RC Mode Only)	RO	0x0
0	INTA	Link0 INTA. (RC Mode Only)	RO	0x0

19.8.4.34 PCIE_LINK1_INT_STATUS0
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0A4

Bit#	Name	Description	Attribute	Reset
31:6	Reserved	Reserved		0x0000000
5	PME_INT	Link1 PME interrupt. The bit is set when the PME status bit in the Root Status register is set to '1', the Interrupt Enable bit in the Root Control register is set to '1', and the INTX Assertion Disable bit in the Command register is '0'. (RC Mode Only)	RO	0x0
4	AER_RC_ERR_INT	Link1 advanced error interrupt. The bit is set when an error condition causes a bit to be set in the Root Error Status register and the associated error message reporting enable bit is set in the Root Error Command register. (RC Mode Only)	RO	0x0
3	INTD	Link1 INTD. (RC Mode Only)	RO	0x0
2	INTC	Link1 INTC. (RC Mode Only)	RO	0x0
1	INTB	Link1 INTB. (RC Mode Only)	RO	0x0
0	INTA	Link1 INTA. (RC Mode Only)	RO	0x0

19.8.4.35 PCIE_LINK0_INT_STATUS1
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0A8

Bit#	Name	Description	Attribute	Reset
31:4	Reserved	Reserved		0x0000000
3	HP_INT	Link0 hot-plug interrupt. This bit is set if the Hot-Plug interrupts are enabled in the Slot Control register, and if any bit in the Slot Status register transitions from '0' to '1' and the associated event notification is enabled in the Slot Control register.	RW1C	0x0
2	HP_PME	Link0 hot-plug PME wake generation. This bit is set if the PME Enable bit in the Power Management Control and Status register is set to 1, and if any bit in the Slot Status register transitions from '0' to '1' and the associated event notification is enabled in the Slot Control register.	RW1C	0x0
1	SYS_ERR	Link0 system error. The bit is set when any device in the hierarchy reports any of the following errors and the associated enable bit is set in the Root Control register: ERR_COR, ERR_FATAL, ERR_NONFATAL. It is also set when an internal error is detected. (RC Mode Only)	RW1C	0x0
0	RST_REQ	Link0 reset request. Reset request due to Link down status.	RW1C	0x0

19.8.4.36 PCIE_LINK1_INT_STATUS1
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0AC

Bit#	Name	Description	Attribute	Reset
31:4	Reserved	Reserved		0x00000000
3	HP_INT	Link1 hot-plug interrupt. This bit is set if the Hot-Plug interrupts are enabled in the Slot Control register, and if any bit in the Slot Status register transitions from '0' to '1' and the associated event notification is enabled in the Slot Control register.	RW1C	0x0
2	HP_PME	Link1 hot-plug PME wake generation. This bit is set if the PME Enable bit in the Power Management Control and Status register is set to '1', and if any bit in the Slot Status register transitions from '0' to '1' and the associated event notification is enabled in the Slot Control register.	RW1C	0x0
1	SYS_ERR	Link1 system error. The bit is set when any device in the hierarchy reports any of the following errors and the associated enable bit is set in the Root Control register: ERR_COR, ERR_FATAL, ERR_NONFATAL. It is also set when an internal error is detected. (RC Mode Only)	RW1C	0x0
0	RST_REQ	Link1 reset request. Reset request due to Link down status.	RW1C	0x0

19.8.4.37 PCIE_LINK0_INT_ENABLE0
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0B0

Bit#	Name	Description	Attribute	Reset
31:6	Reserved	Reserved		0x00000000
5	PME_INT	Link0 PME interrupt enable. (RC Mode Only)	R/W	0x0
4	AER_RC_ERR_INT	Link0 advanced error interrupt enable. (RC Mode Only)	R/W	0x0
3	INTD	Link0 INTD enable. (RC Mode Only)	R/W	0x0
2	INTC	Link0 INTC enable. (RC Mode Only)	R/W	0x0
1	INTB	Link0 INTB enable. (RC Mode Only)	R/W	0x0
0	INTA	Link0 INTA enable. (RC Mode Only)	R/W	0x0

19.8.4.38 PCIE_LINK1_INT_ENABLE0
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0B4

Bit#	Name	Description	Attribute	Reset
31:6	Reserved	Reserved		0x00000000
5	PME_INT	Link1 PME interrupt enable. (RC Mode Only)	R/W	0x0
4	AER_RC_ERR_INT	Link1 advanced error interrupt enable. (RC Mode Only)	R/W	0x0
3	INTD	Link1 INTD. (RC Mode Only)	R/W	0x0
2	INTC	Link1 INTC. (RC Mode Only)	R/W	0x0
1	INTB	Link1 INTB. (RC Mode Only)	R/W	0x0
0	INTA	Link1 INTA. (RC Mode Only)	R/W	0x0

19.8.4.39 PCIE_LINK0_INT_ENABLE1
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0B8

Bit#	Name	Description	Attribute	Reset
31:4	Reserved	Reserved		0x00000000
3	HP_INT	Link0 hot-plug interrupt enable.	R/W	0x0
2	HP_PME	Link0 hot-plug PME wake generation enable.	R/W	0x0
1	SYS_ERR	Link0 system error enable. (RC Mode Only)	R/W	0x0
0	RST_REQ	Link0 reset request enable.	R/W	0x0

19.8.4.40 PCIE_LINK1_INT_ENABLE1
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0BC

Bit#	Name	Description	Attribute	Reset
31:4	Reserved	Reserved		0x00000000
3	HP_INT	Link1 hot-plug interrupt enable.	R/W	0x0
2	HP_PME	Link1 hot-plug PME wake generation enable.	R/W	0x0
1	SYS_ERR	Link1 system error enable. (RC Mode Only)	R/W	0x0
0	RST_REQ	Link1 reset request enable.	R/W	0x0

19.8.4.41 PCIE_PHY_CR_CMD
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0C0

Bit#	Name	Description	Attribute	Reset
31:17	Reserved	Reserved		0x0000
16	CR_CMD	PHY control register command: 0: read 1: write	R/W	0x0
15:0	CR_ADDR	PHY control register address.	R/W	0x0000

19.8.4.42 PCIE_PHY_CR_WR_DATA
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0C4

Bit#	Name	Description	Attribute	Reset
31:16	Reserved	Reserved		0x0000
15:0	CR_WR_DATA	PHY control register write data.	R/W	0x0000

19.8.4.43 PCIE_PHY_CR_RESP
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0C8

Bit#	Name	Description	Attribute	Reset
31:1	Reserved	Reserved		0x00000000
0	CR_CMD_DONE	Indicates PHY control command is done.	RW1C	0x0

19.8.4.44 PCIE_PHY_CR_RD_DATA
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0CC

Bit#	Name	Description	Attribute	Reset
31:16	Reserved	Reserved		0x0000
15:0	CR_RD_DATA	PHY control register read data.	R/W	0x0000

19.8.4.45 PCIE_IOBM_ERR_CMD
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0D0

Bit#	Name	Description	Attribute	Reset
31:7	Reserved	Reserved		0x00000000
6	MULTIPLE	IOB master multiple error occurred.	RO	0x0
5:3	ADDRSPACE	IOB master error request address space.	RO	0x0
2:0	CMD	IOB master error request command.	RO	0x0

19.8.4.46 PCIE_IOBM_ERR_LOWER_ADDR
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0D4

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_ERR_LOWER_ADDR	IOB master request first error address bits 36:5	RO	0x00000000

19.8.4.47 PCIE_IOBM_ERR_UPPER_ADDR
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0D8

Bit#	Name	Description	Attribute	Reset
31:3	Reserved	Reserved		0x00000000
2:0	IOBM_ERR_UPPER_ADDR	IOBM master request first error upper address bits [39:37]	RO	0x0

19.8.4.48 PCIE_IOBM_ERR_BE
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0DC

Bit#	Name	Description	Attribute	Reset
31:0	IOBM_ERR_BYTE_ENABLE	IOB master request first error byte enable.	RO	0x00000000

19.8.4.49 Reserved
(For XLS408Lite and XLS404Lite)

Address Offset: 0x0E0 through 0xC1C

Bit#	Name	Description	Attribute	Reset
31:0	Reserved	Reserved	R/W	0x00

19.8.4.50 VC0_POSTED_RX_QUEUE_CTRL
(For 408Lite and XLS404Lite)

Address Offset: 0x748

This register sets ordering rules and queue mode for TLPs and effectively sets the size of buffers used for packet headers and data. The value that must be written to this register depends on the number of ports configured for the link.

After reset, software should write one of the following values to the register, depending on the number of ports per link:

	Value
Four ports per link:	0x00018074
One or two ports per link:	0x40218036

The fields in this register are sticky read-only, meaning that the register can be written once after reset and thereafter cannot be altered. The register is not initialized or modified after a hot reset.

19.8.4.51 VC0_POSTED_BUFFER_DEPTH
(For XLS408Lite and XLS404Lite)

Address Offset: 0x7A8

This register defines the VC0 posted data queue depth and header queue depth. The value that must be written to this register depends on the number of ports configured for the link.

After reset, software should write one of the following values to the register, depending on the number of ports per link:

	Value
Four ports per link:	0x001901D1
One or two ports per link:	0x001900D9

The fields in this register are sticky read-only, meaning that the register can be written once after reset and thereafter cannot be altered. The register is not initialized or modified after a hot reset.

19.8.4.52 PCIE_MSG_TX_THRESHOLD
(For XLS408Lite and XLS404Lite)

Address Offset: 0xC20

Bit#	Name	Description	Attribute	Reset
31:0	MSG_TX_THRESHOLD	Message transmit credit threshold size.	R/W	0x00000003

19.8.4.53 PCIE_MSG_BUCKET_SIZE_0
(For XLS408Lite and XLS404Lite)

Address Offset: 0xC80

Bit#	Name	Description	Attribute	Reset
31:9	Reserved			0x000000
8:0	MSG_BUCKET_SIZE_0	Size of the message bucket for link0 DMA Load.	R/W	0x20

19.8.4.54 PCIE_MSG_BUCKET_SIZE_1
(For XLS408Lite and XLS404Lite)

Address Offset: 0xC84

Bit#	Name	Description	Attribute	Reset
31:9	Reserved			0x000000
8:0	MSG_BUCKET_SIZE_1	Size of the message bucket for link0 DMA Store.	R/W	0x20

19.8.4.55 PCIE_MSG_BUCKET_SIZE_2
(For XLS408Lite and XLS404Lite)

Address Offset: 0xC88

Bit#	Name	Description	Attribute	Reset
31:9	Reserved			0x000000
8:0	MSG_BUCKET_SIZE_2	Size of the message bucket for link1 DMA Load.	R/W	0x20

19.8.4.56 PCIE_MSG_BUCKET_SIZE_3
(For XLS408Lite and XLS404Lite)

Address Offset: 0xC8C

Bit#	Name	Description	Attribute	Reset
31:9	Reserved			0x000000
8:0	MSG_BUCKET_SIZE_3	Size of the message bucket for link1 DMA Store.	R/W	0x20

19.8.4.57 PCIE_MSG_CREDIT
(For XLS408Lite and XLS404Lite)

Address Offset: 0xE00-0xFFC

Bit#	Name	Description	Attribute	Reset
31:9	<i>Reserved</i>			0x000000
8:0	MSG_CREDIT	Credit Counter Registers.	R/W	0x000

NETLOGIC
CONFIDENTIAL

19.9 PCIe Configuration Registers: EP Mode

This section describes the interface's PCIe Configuration registers in EP (End Point) mode. The PCIe Configuration registers in RC mode is shown on [page 953](#). All reset values are in hex unless otherwise noted.

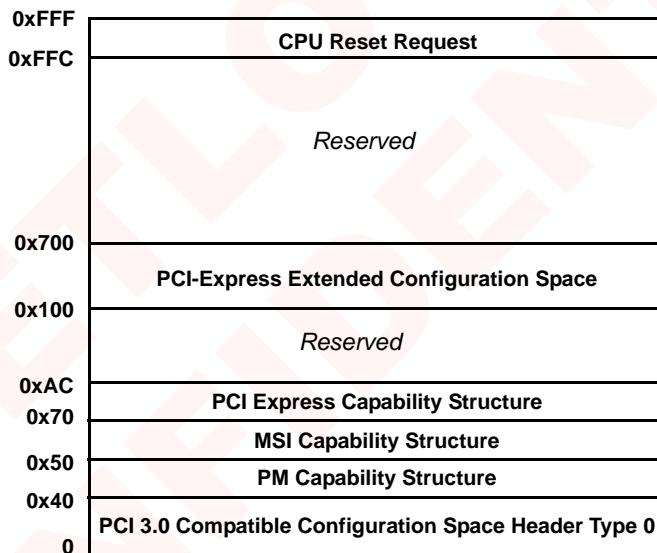
19.9.1 Register Space Layout

The PCIe core has 4096 bytes of PCI Express configuration space. The PCI Express configuration space is divided into:

- 256 bytes of PCI 3.0 Compatible Configuration Space Header (type 0 in EP mode)
- PCI Capabilities structures, which start at offset 0x40
- PCI Express Extended Configuration Space, which starts at offset 0x100
- Port Logic registers (vendor-specific registers), which start at offset 0x700. The Port Logic registers have specific pre-defined usages, mostly for test purposes. The usage of the Port Logic registers is the same in both EP mode and RC mode.

[Figure 19-9](#) shows the EP mode layout of the PCIe core configuration space.

Figure 19-9. EP Mode PCIe core Configuration Space Layout



19.9.2**Register Maps**

Configuration registers are in structures (groups) identified by a Capability ID. Groups are linked together as in PCI. Register locations within a group are specified, but the starting location of each group must be found by traversing the linked list. There are two linked lists of register groups: PCI-compatible base registers and PCI Express Enhanced Capability registers. PCI-compatible base register groups begin at configuration address 0x00. PCI Express Enhanced Capability register groups begin at address 0x100.

19.9.2.1**Configuration Space Header – Type 0**

[Table 19-14](#) shows the configuration field register definitions for PCI Express Type 0 Configuration Space header. Most PCI-compatible register fields have the same software interpretation in PCI 3.0 and PCI Express.

Table 19-14. PCI Configuration Space Header - Type 0

Byte Address Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x00	Device ID		Vendor ID	
0x04	Status Register		Command Register	
0x08	Class Code		Revision ID	
0x0C	BIST(0x00)	Header Type	Latency Timer	Cache Line Size
0x10	Base Address Register 0			
0x14	Base Address Register 1			
0x18	Base Address Register 2			
0x1C	Base Address Register 3			
0x20	Base Address Register 4			
0x24	Base Address Register 5			
0x28	CardBus CIS Pointer			
0x2C	Subsystem ID		Subsystem Vendor ID	
0x30	Expansion ROM Base Address			
0x34	Reserved			CapPtr
0x38	Reserved			
0x3C	Max_Latency ^a	Min_Grant ¹	Interrupt Pin	Interrupt Line

- a. The Max_Latency and Min_Grant registers do not apply to PCI Express and are read-only registers with values hardwired to 0x00.

19.9.2.2**Capability Structures Register Maps**

The Capability Pointer register in the PCI-compatible header register points to the next item in the linked list of capabilities, which, by default, is the PCI Power-Management capabilities register space.

[Table 19-15](#) lists the capabilities supported by the PCIe core and their respective default address offsets and next capability pointers.

Table 19-15. Configuration Structure: Starting Addresses and Next Capability Pointers

Start Address Offset	Item	Next Pointer
0x00	PCI-Compatible Header (Type 0)	0x40
0x40	PCI Power Management	0x50
0x50	Message Signaled Interrupt (MSI)	0x70
0x70	PCI Express Capabilities	0x00

The following tables show the PCI capability structures.

Table 19-16. Power Management Capability Structure

Byte Address Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x40	Power Management Capabilities (PMC)		Next Capability Pointer ('PM_NEXT_PTR)	Capability ID (0x01)
+0x4	Data	PMCSR_BSE Bridge Extensions	Power Management Control Status Register (PMCSR)	

Table 19-17. MSI Capability Structure

Byte Address Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x50	Message Control Register		Next Capability Pointer ('MSI_NEXT_PT)	Capability ID (0x05)
+0x4	MSI Lower 32-bit Address Register			
+0x8	MSI Upper 32-bit Address Register			
+0xC			MSI Data	

Table 19-18. PCI Express Capability Structure

Byte Address Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x70	PCI Express Capabilities Register		Next Capability Pointer ('PCIE_NEXT_PT)	Capability ID (0x10)
+0x4	Device Capabilities			
+0x8	Device Status		Device Control	
+0xC	Link Capabilities			
+0x10	Link Status		Link Control	
+0x14	Slot Capabilities ^a			
+0x18	Slot Status		Slot Control	

a. Slot Capabilities is provided for reference, but is not required for Endpoint device applications.

19.9.2.3**PCI Express Extended Capability Register Maps**

The PCI Express Extended Capabilities registers are located in device configuration space at offsets 0x100 or higher. As with PCI Capabilities, the PCI Express Extended Capability structures are allocated using a linked list with a similar method and format to those of PCI.

The Advanced Error Reporting Capability and Virtual Channel Capability are optional extended capabilities that may be implemented by PCI Express devices supporting advanced error control and reporting and multiple VCs, respectively. The Advanced Error Reporting Capability is required when the device supports ECRC generation/checking. The Virtual Channel Capability is required for any device that supports multiple VCs and/or multiple Traffic Classes (TCs).

The following tables outline the PCI Express Extended Capabilities structures.

Table 19-19. Advanced Error Reporting Capability Structure

Byte Address Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x010	PCI Express Enhanced Capability Header			
+0x4	Uncorrectable Error Status Register			
+0x8	Uncorrectable Error Mask Register			
+0xC	Uncorrectable Error Severity Register			
+0x10	Correctable Error Status Register			
+0x14	Correctable Error Mask Register			
+0x18	Advanced Error Capabilities and Control Register			
+0x1C through +0x28	Header Log Registers			

Table 19-20. Device Serial Number Capability Register

Byte Address Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x140	Device Serial Number Enhanced Capability Header			
+0x4	Serial Number Register (1st DWord)			
+0x8	Serial Number Register (2nd DWord)			

19.9.3 Accessing Configuration Registers

Host software accesses the configuration registers through PCI Express Configuration Requests.

The Attribute (ATTR) column in each register description indicates the read/write access for the register or bit. [Table 19-21](#) defines the read/write attribute abbreviations that are used in the register and bit descriptions throughout this chapter.

Table 19-21. Configuration Register Bit-Field Types

Attribute	Meaning	Description
Hwinit	Hardware Initialized	HwInit bits are controlled by PCIe core hardware and are read-only by host system software.
RO	Read Only	Register bits are read-only and should not be altered by user or application software.
RW	Read-Write	Register bits are read-write and may be read and written normally by the host and the application. Writing from the application side (if any) requires careful synchronization with the host software.
RW1C	Read-Only Status/ Write-'1'-to- Clear Status Register	Register bits indicate status when read. A set bit indicating a status event may be cleared by writing a '1'. Writing '0' to RW1C bits has no effect. Writing from the application side (if any) requires careful synchronization with host software.
ROS	Sticky Read-Only	Registers are read-only and cannot be altered by host or application software, except as noted. Registers are not initialized or modified by a hot reset. A few bits designated as very sticky are not cleared by any type of PCIe core reset when auxiliary power is supplied and enabled.
RWS	Sticky Read-Write	Registers are read-write and may be either set or cleared by host or application software to the desired state. Bits are not initialized or modified by a hot reset. A few bits designated very sticky are not cleared by any type of PCIe core reset when auxiliary power is supplied and enabled.
RW1CS	Sticky Read-Only Status/Write-1-to- Clear Status	Writing a '1' clears status for the host and the application, except as noted. Registers indicate status when read. A set bit indicating a status event may be cleared by writing a '1'. Writing '0' to RW1CS bits has no effect. Bits are not initialized or modified by a hot reset. A few bits designated very sticky are not cleared by any type of PCIe core reset when auxiliary power is supplied and enabled.
RsvdP	Reserved and PReserved	Reserved for future RW implementations: software must preserve the value read when writing to other bits in the same register.
RsvdZ	Reserved and Zero	Reserved for future RW1C implementations: software must write '0' to these bits when writing to other bits in the same register.

19.9.4 Register Default Reset Values

You can select the default reset values of several of the configuration registers through design configuration parameters. The register descriptions in this chapter indicate the default reset value of the register, either as a specific numerical value or as the name of the configuration parameter that sets the default reset value.

19.9.5 PCI-Compatible Configuration Header Register Details

19.9.5.1 Device ID and Vendor ID Register

The following table defines the Device ID and Vendor ID register bit assignments. The default values of both Device ID and Vendor ID are hardware configuration parameters.

Address Offset: 0x04h

Bit#	Description	Attribute	Reset
15:0	Vendor ID	RO	0x182E
31:16	Device ID	RO	0xABCD

19.9.5.2 Command Register

The following table defines the PCI-compatible Command register bit assignments.

Address Offset: 0x04

Bytes: 0-1

Bit#	Description	Attribute	Reset
0	I/O Space Enable	RW	0
1	Memory Space Enable	RW	0
2	Bus Master Enable	RW	0
3	Special Cycle Enable Not applicable for PCI Express. Must be hardwired to '0'.	RO	0
4	Memory Write and Invalidate Not applicable for PCI Express. Must be hardwired to '0'.	RO	0
5	VGA Palette Snoop Not applicable for PCI Express. Must be hardwired to '0'.	RO	0
6	Parity Error Response	RW	0
7	IDSEL Stepping/Wait Cycle Control Not applicable for PCI Express. Must be hardwired to '0'.	RO	0
8	SERR# Enable	RW	0
9	Fast Back-to-Back Enable Not applicable for PCI Express. Must be hardwired to '0'.	RO	0
10	INTX Assertion Disable	RW	0
15:11	Reserved	RO	0x0

19.9.5.3 Status Register

The following table defines the PCI-compatible Status register bit assignments.

Address Offset: 0x04

Bytes: 2-3

Bit#	Description	Attribute	Reset
2:0	<i>Reserved</i>	RO	0x0
3	INTX Status	RO	0
4	Capabilities List Indicates presence of an extended capability item. Hardwired to '1'.	RO	1
5	66.67 MHz Capable Not applicable for PCI Express. Hardwired to '0'.	RO	0
6	<i>Reserved</i>	RO	0
7	Fast Back-to-Back Capable Not applicable for PCI Express. Hardwired to '0'.	RO	0
8	Master Data Parity Error	RW1C	0
10:9	DEVSEL Timing Not applicable for PCI Express. Hardwired to '0'.	RO	0x0
11	Signaled Target Abort	RW1C	0
12	Received Target Abort	RW1C	0
13	Received Master Abort	RW1C	0
14	Signaled System Error	RW1C	0
15	Detected Parity Error	RW1C	0

19.9.5.4 Revision ID Register

Address Offset: 0x08

Byte: 0

Bit#	Description	Attribute	Reset
7:0	Revision ID	RO	0x1

19.9.5.5 Class Code Register

Address Offset: 0x08

Bytes: 1-3

Bit#	Description	Attribute	Reset
7:0	Programming Interface	RO	0x0
15:8	Subclass Code	RO	0x00
23:16	Base Class Code	RO	0xFF

19.9.5.6 Cache Line Size Register

Address Offset: 0x0C

Byte: 0

Bit#	Description	Attribute	Reset
7:0	Cache Line Size The Cache Line Size register is RW for legacy compatibility purposes and is not applicable to PCI Express device functionality. Writing to the Cache Line Size register does not impact functionality of the PCIe core.	RW	0x00

19.9.5.7 Master Latency Timer Register

Address Offset: 0x0C

Byte: 1

Bit#	Description	Attribute	Reset
7:0	Master Latency Timer Not applicable for PCI Express, hardwired to '0'.	RO	0x00

19.9.5.8 Header Type Register

Address Offset: 0x0C

Byte: 2

Bit#	Description	Attribute	Reset
6:0	Configuration Header Format Hardwired to '0' for type 0.	RO	0x0
7	Multi Function Device	RO	0x0

19.9.5.9 BIST Register

Address Offset: 0x0C

Byte: 3

Bit#	Description	Attribute	Reset
7:0	The BIST register functions are not supported by the PCIe core. All 8 bits of the BIST register are hardwired to '0'.	RO	0x00

19.9.5.10 Base Address Registers

Address Offsets: 0x10-0x24

Base Address Register 0 (Optional)

Offset: 0x10

Bit#	Description	Attribute	Reset
31:4	BAR 0 base address bits (for a 64-bit BAR, the remaining upper address bits are in BAR 1). The BAR 0 Mask value determines which address bits are masked.	RO	0x00000000
3	If BAR 0 is a memory BAR, bit [3] indicates if the memory region is prefetchable: 0: = Non-prefetchable 1: = Prefetchable If BAR 0 is an I/O BAR, bit [3] is the second least significant bit of the base address.	RO	0x1
2:1	If BAR 0 is a memory BAR, bits [2:1] determine the BAR type: 00: = 32-bit BAR 10: = 64-bit BAR If BAR 0 is an I/O BAR, bit [2] the least significant bit of the base address and bit [1] is '0'.	RO	0x2
0	0: = BAR 0 is a memory BAR 1: = BAR 0 is an I/O BAR	RO	0x1

Base Address Register 1 (Optional)

Address Offset: 0x14

Bit#	Description	Attribute	Reset
31:0	If BAR 0 is a 64-bit BAR, BAR 1 contains the upper 32 bits of the BAR 0 base address (bits [63:32]). If BAR 0 is a 32-bit BAR, BAR 1 can be independently programmed as an additional 32-bit BAR or can be excluded from the PCIe core hardware configuration. If programmed as an independent 32-bit BAR, the BAR 1 bit definitions are the same as the BAR 0 bit definitions.	RO	0x0

Base Address Register 2 (Optional)

Address Offset: 0x18

Bit#	Description	Attribute	Reset
31:4	BAR 2 base address bits (for a 64-bit BAR, the remaining upper address bits are in BAR 3). The BAR 2 Mask value determines which address bits are masked.	RO	0x0000000
3	If BAR 2 is a memory BAR, bit [3] indicates if the memory region is prefetchable: 0: = Non-prefetchable 1: = Prefetchable If BAR 2 is an I/O BAR, bit [3] is the second least significant bit of the base address.	RO	0x1
2:1	If BAR 2 is a memory BAR, bits [2:1] determine the BAR type: 00: = 32-bit BAR 10: = 64-bit BAR If BAR 2 is an I/O BAR, bit [2] the least significant bit of the base address and bit [1] is '0'.	RO	0x2
0	0: = BAR 2 is a memory BAR 1: = BAR 2 is an I/O BAR	RO	0x0

Base Address Register 3 (Optional)

Address Offset: 0x1C

Bit#	Description	Attribute	Reset
31:0	If BAR 2 is a 64-bit BAR, BAR 3 contains the upper 32 bits of the BAR 2 base address (bits [63:32]). If BAR 2 is a 32-bit BAR, BAR 3 can be independently programmed as an additional 32-bit BAR or can be excluded from the PCIe core hardware configuration. If programmed as an independent 32-bit BAR, the BAR 3 bit definitions are the same as the BAR 2 bit definitions.	RO	0x0

Base Address Register 4 (Optional)

Address Offset: 0x20

Bit#	Description	Attribute	Reset
31:4	BAR 4 base address bits (for a 64-bit BAR, the remaining upper address bits are in BAR 5). The BAR 4 Mask value determines which address bits are masked.	RO	0x0000000
3	If BAR 4 is a memory BAR, bit [3] indicates if the memory region is prefetchable: 0: = Non-prefetchable 1: = Prefetchable If BAR 4 is an I/O BAR, bit [3] is the second least significant bit of the base address.	RO	0x1
2:1	If BAR 4 is a memory BAR, bits [2:1] determine the BAR type: 00: = 32-bit BAR 10: = 64-bit BAR If BAR 4 is an I/O BAR, bit [2] the least significant bit of the base address and bit [1] is '0'.	RO	0x2
0	0: = BAR 4 is a memory BAR 1: = BAR 4 is an I/O BAR	RO	0x1

Base Address Register 5 (Optional)

Address Offset: 0x24

Bit#	Description	Attribute	Reset
31:0	If BAR 4 is a 64-bit BAR, BAR 5 contains the upper 32 bits of the BAR 4 base address (bits 63:32). If BAR 4 is a 32-bit BAR, BAR 5 can be independently programmed as an additional 32-bit BAR or can be excluded from the PCIe core hardware configuration. If programmed as an independent 32-bit BAR, the BAR 5 bit definitions are the same as the BAR 4 bit definitions.	RO	0x0

19.9.5.11 BAR Mask Registers

BAR 0 Mask Register

Address Offset: 0x10

Bit#	Description	Attribute	Reset
31:1	Indicates which BAR 0 bits to mask (make non-writable) from host software, which, in turn, determines the size of the BAR. For example, writing 0xFFFF to the BAR 0 Mask register claims a 4096-byte BAR by masking bits [11:0] of the BAR from writing by host software. If BAR 0 is a 64-bit BAR, then the BAR 1 Mask register contains the upper bits of the BAR 0 Mask. The BAR 0 Mask register is invisible to host software and not readable from the application.	Application write access only; not readable	0x1FFFFF
0	BAR 0 Enable 0: BAR 0 is disabled 1: BAR 0 is enabled Bit [0] is interpreted as BAR Enable when writing to the BAR Mask register rather than as a mask bit because bit [0] of a BAR is always masked from writing by host software.	Same as bits [31:1]	1

BAR 1 Mask Register

Address Offset: 0x14

Bit#	Description	Attribute	Reset
31:0	If BAR 0 is a 32-bit BAR: <ul style="list-style-type: none">• Bits [31:1]: BAR 1 Mask value, interpreted the same way as BAR 0 Mask.• Bit [0]: BAR 1 Enable ('0' = BAR 1 is disabled; '1' = BAR 1 is enabled). If BAR 0 is a 64-bit BAR: <ul style="list-style-type: none">• Bits [31:0] are the upper bits of the BAR 0 Mask value.	Application write access only	0x0

BAR 2 Mask Register

Address Offset: 0x18

Bit#	Description	Attribute	Reset
31:1	Indicates which BAR 2 bits to mask (make non-writable) from host software, which, in turn, determines the size of the BAR. For example, writing 0xFFFF to the BAR 2 Mask register claims a 4096- byte BAR by masking bits [11:0] of the BAR from writing by host software. If BAR 2 is a 64-bit BAR, then the BAR 3 Mask register contains the upper bits of the BAR 2 Mask. The BAR 2 Mask register is invisible to host software and not readable from the application.	Application write access only	0FFF
0	BAR 2 Enable 0: BAR 2 is disabled 1: BAR 2 is enabled Bit [0] is interpreted as BAR Enable when writing to the BAR Mask register rather than as a mask bit because bit [0] of a BAR is always masked from writing by host software.	Same as bits [31:1]	1

BAR 3 Mask Register

Address Offset: 0x1C

Bit#	Description	Attribute	Reset
31:0	If BAR 2 is a 32-bit BAR: <ul style="list-style-type: none">• Bits [31:1]: BAR 3 Mask value, interpreted the same way as BAR 2 Mask.• Bit [0]: BAR 3 Enable ('0' = BAR 3 is disabled; '1'= BAR 3 is enabled). If BAR 2 is a 64-bit BAR: <ul style="list-style-type: none">• Bits [31:0] are the upper bits of the BAR 2 Mask value.	Application write access only	0x0

BAR 4 Mask Register

Address Offset: 0x20

Bit#	Description	Attribute	Reset
31:1	Indicates which BAR 4 bits to mask (make non-writable) from host software, which, in turn, determines the size of the BAR. For example, writing 0xFFFF to the BAR 4 Mask register claims a 4096-byte BAR by masking bits [11:0] of the BAR from writing by host software. If BAR 4 is a 64-bit BAR, then the BAR 5 Mask register contains the upper bits of the BAR 4 Mask. The BAR 4 Mask register is invisible to host software and not readable from the application.	Application write access only	0xFFFF
0	BAR 4 Enable <ul style="list-style-type: none">0: BAR 4 is disabled1: BAR 4 is enabled Bit [0] is interpreted as BAR Enable when writing to the BAR Mask register rather than as a mask bit because bit [0] of a BAR is always masked from writing by host software.	Same as bits [31:1]	1

BAR 5 Mask Register

Address Offset: 0x24

Bit#	Description	Attribute	Reset
31:0	If BAR 4 is a 32-bit BAR: <ul style="list-style-type: none">• Bits [31:1]: BAR 5 Mask value, interpreted the same way as BAR 5 Mask.• Bit [0]: BAR 5 Enable ('0' = BAR 5 is disabled; '1'= BAR 5 is enabled). If BAR 4 is a 64-bit BAR: <ul style="list-style-type: none">• Bits [31:0] are the upper bits of the BAR 4 Mask value.	Application write access only	0x0

Expansion ROM BAR Mask Register

Address Offset: 0x30

Bit#	Description	Attribute	Reset
31:1	Indicates which Expansion ROM BAR bits to mask (make non-writable) from host software, which, in turn, determines the size of the BAR. For example, writing 0xFFFF to the Expansion ROM BAR Mask register claims a 4096-byte BAR by masking bits [11:0] of the BAR from writing by host software. The maximum value is 0xFFFFFFFF because the maximum space that can be claimed by an Expansion ROM BAR is 16 MB. The Expansion ROM BAR Mask register is invisible to host software and not readable from the application.	Application write access only	0
0	Expansion ROM BAR Enable 0: Expansion ROM BAR is disabled 1: Expansion ROM BAR is enabled	Same as bits [31:1]	0

Example BAR Setup

Figure 19-10 shows an example configuration of the six BARs and their corresponding BAR Mask registers. The example configuration includes:

- One 64-bit memory BAR (non-prefetchable)
- One 32-bit memory BAR (non-prefetchable)
- One 32-bit I/O BAR

Figure 19-10. Example Base Address Register Configuration

BAR Address Registers	
Disabled	
BAR 5 (24h)	Writable Base Address Bits (31:8)
BAR 4 (20h)	Masked Base Addr. Bits (7:2) 01
Disabled	
BAR 3 (1Ch)	Writable Base Address Bits (31:20)
BAR 2 (18h)	Masked Base Addr. Bits (19:4) 0000
Upper Address Bits of BAR 0 (non-masked)	
BAR 1 (14h)	Writable Base Address Bits (31:20)
BAR 0 (10h)	Masked Base Addr. Bits (31:20) 0100

BAR Mask Registers	
BAR 5 Mask (24h)	0x0
BAR 4 Mask (20h)	0x000000FF
BAR 3 Mask (1Ch)	0x0
BAR 2 Mask (18h)	0x000FFFFF
BAR 1 Mask (14h)	0x00000000
BAR 0 Mask (10h)	0x000FFFFF

19.9.5.12 CardBus CIS Pointer Register

Address Offset: 0x28

Bit#	Description	Attribute	Reset
31:0	CardBus CIS Pointer	RO	0x0

19.9.5.13 Subsystem ID and Subsystem Vendor ID Register

Address Offset: 0x2C

Bit#	Description	Attribute	Reset
31:16	Subsystem ID	RO	0x0
15:0	Subsystem Vendor ID	RO	0x0

19.9.5.14 Expansion ROM Base Address Register

Address Offset: 0x30

Bit#	Description	Attribute	Reset
31:11	Expansion ROM Address	RW	0x00000
10:1	Reserved	RO	0x000
0	Expansion ROM Enable	RW	0x0

19.9.5.15 Capability Pointer Register

Address Offset: 0x34

Byte: 0

Bit#	Description	Attribute	Reset
7:0	First Capability Pointer. Points to Power Management Capability structure.	RO	0x40

19.9.5.16 Interrupt Line Register

Address Offset: 0x3C

Byte: 0

Bit#	Description	Attribute	Reset
7:0	Interrupt Line	RW	0xFF

19.9.5.17 Interrupt Pin Register

Address Offset: 0x3C

Byte: 1

Bit#	Description	Attribute	Reset
7:0	Interrupt Pin Identifies the legacy interrupt Message that the device (or device function) uses. Valid values are: 0x00: The device (or function) does not use legacy interrupt 0x01: The device (or function) uses INTA 0x02: The device (or function) uses INTB 0x03: The device (or function) uses INTC 0x04: The device (or function) uses INTD	RO	0x1

19.9.6 PCI Power Management Capability Register Details

The PCIe core implements power management capabilities. The Capability Pointer field in the configuration header points to the PCI Power Management registers as the first extended capability by default.

The extent of the power management implementation in the PCIe core includes:

- Power Management register space
- Link state information (provided to both the application logic and PHY interfaces)
- Power management-ready clock and reset implementation

The following sections describe the PCI Power Management registers implemented in the PCIe core. See the PCI Power Management Specification and the PCI Express Base Specification for more details.

19.9.6.1 Power Management Capability ID Register

Address Offset: 0x40

Bit#	Description	Attribute	Reset
7:0	Power Management Capability ID	RO	0x01

19.9.6.2 Power Management Next Item Pointer

Address Offset: `CFG_PM_CAP + 0x01

Bit#	Description	Attribute	Reset
7:0	Next Capability Pointer Points to the MSI capabilities by default.	RO	0x50

19.9.6.3 Power Management Capabilities Register

Address Offset: `CFG_PM_CAP + 0x02

Bit#	Description	Attribute	Reset
15:11	PME_Support Identifies the power states from which the PCIe core can generate PME Messages. A value of '0' for any bit indicates that the device (or function) is not capable of generating PME Messages while in that power state: <ul style="list-style-type: none">• Bit [11]: If set, PME Messages can be generated from D0• Bit [12]: If set, PME Messages can be generated from D1• Bit [13]: If set, PME Messages can be generated from D2• Bit [14]: If set, PME Messages can be generated from D3hot• Bit [15]: If set, PME Messages can be generated from D3cold	RO	0x1B
10	D2 Support	RO	0x0
9	D1 Support	RO	0x1
8:6	AUX Current	RO	0x7
5	Device Specific Initialization (DSI)	RO	0x0
4	Reserved	RsvdP	0
3	PME Clock, hardwired to '0'	RO	0
2:0	Power Management Specification Version	RO	0x011

19.9.6.4 Power Management Control and Status Register

Address Offset: `CFG_PM_CAP + 0x04

Bit#	Description	Attribute	Reset
31:24	Data register for additional information (not supported)	RO	0x00
23	Bus Power/Clock Control Enable, hardwired to '0'	RO	0x0
22	B2/B3 Support, hardwired to '0'	RO	0x0
21:16	Reserved	RsvdP	0x0
14:13	Data Scale (not supported)	RO	0x0
12:9	Data Select (not supported)	RO	0x0
15	PME Status Indicates if a previously enabled PME event occurred or not.	RW1CS	0x0
8	PME Enable (sticky bit) A value of '1' indicates that the device is enabled to generate PME.	RWS	0x0
7:4	Reserved	RsvdP	0x0
3	No Soft Reset	RO	0
2	Reserved	RsvdP	0x0
1:0	Power State Controls the device power state: 00: D0 01: D1 10: D2 11: D3 The written value is ignored if the specific state is not supported.	RW	0x0

19.9.7 MSI Capability Register Details

The MSI Capability structure is required for all PCI Express devices that are capable of generating interrupts. The definition of the MSI register structure is compatible with the PCI 3.0 specification.

19.9.7.1 MSI Capability ID

Address Offset: 0x50

Bit#	Description	Attribute	Reset
7:0	MSI Capability ID	RO	0x05

19.9.7.2 MSI Next Item Pointer

Address Offset: `CFG_MSI_CAP + 0x01

Bit#	Description	Attribute	Reset
7:0	Next Capability Pointer Points to PCI Express Capabilities	RO	0x70

19.9.7.3 MSI Control Register

Address Offset: `CFG_MSI_CAP + 0x02

Bit#	Description	Attribute	Reset
15:8	<i>Reserved</i>	RO	0x00
7	64-bit Address Capable	RO	1
6:4	Multiple Message Enabled Indicates that multiple Message mode is enabled by system software. The number of Messages enabled must be less than or equal to the Multiple Message Capable value.	RW	0x0
3:1	Multiple Message Capable	RO	0
0	MSI Enabled When set, INTX must be disabled.	RW	0

19.9.7.4 MSI Lower 32 Bits Address Register

Address Offset: `CFG_MSI_CAP + 0x04

Bit#	Description	Attribute	Reset
31:2	Lower 32-bit Address	RW	0x00000000
1:0	<i>Reserved</i>	RO	0x0

19.9.7.5 MSI Upper 32 bits Address Register

Address Offset: `CFG_MSI_CAP + 0x08

Bit#	Description	Attribute	Reset
31:0	Upper 32-bit Address	RW	0x00000000

19.9.7.6 MSI Data Register

Address Offset: `CFG_MSI_CAP + 0x0C

Bit#	Description	Attribute	Reset
31:16	Reserved	RO	0x0000
15:0	MSI Data Pattern assigned by system software, bits [4:0] are Or-ed with MSI_VECTOR to generate 32 MSI Messages per function.	RW	0x0000

19.9.8 PCI Express Capability Register Details

The PCIe core implements the PCI Express Capability Structure as defined in the PCI Express Base Specification except for Root Port registers.

19.9.8.1 PCI Express Capability List Register

Address Offset: `CFG_PCIE_CAP + 0x00

Bit#	Description	Attribute	Reset
15:8	Next Capability Pointer: This read-only pointer completes a linked list “chain” of all supported PCIe features. No other supported features exist in the linked list.	RO	0x00
7:0	PCI Express Capability ID	RO	0x10

19.9.8.2 PCI Express Capabilities Register

Address Offset: `CFG_PCIE_CAP + 0x02

Bit#	Description	Attribute	Reset
15:14	RsvdP	RO	0x0
13:9	Interrupt Message Number (updated by hardware)	RO	0x0
8	Slot Implemented	Hwinit	0x0
7:4	Device Port Type	RO	0
3:0	PCI Express Capability Version	RO	0x1

19.9.8.3 Device Capabilities Register

Address Offset: `CFG_PCIE_CAP + 0x04

Bit#	Description	Attribute	Reset
31:28	Reserved	RsvdP	0x0
27:26	Captured Slot Power Limit Scale From Message from RC, upstream port only.	RO	0x0
25:18	Captured Slot Power Limit Value From Message from RC, upstream port only.	RO	0x00
17:16	Reserved	RsvdP	0x0
15	Role-Based Error Reporting	RO	0x1
14	Undefined for PCI Express 1.1 (Was Power Indicator Present for PCI Express 1.0a)	RO	0x0
13	Undefined for PCI Express 1.1 (Was Attention Indicator Present for PCI Express 1.0a)	RO	0x0
12	Undefined for PCI Express 1.1 (Was Attention Button Present for PCI Express 1.0a)	RO	0x0
11:9	Endpoint L1 Acceptable Latency	RO	0x3
8:6	Endpoint L0s Acceptable Latency	RO	0x4
5	Extended Tag Field Supported	RO	0
4:3	Phantom Function Supported	RO	0x0
2:0	MAX_PAYLOAD_SIZE: The maximum payload size supported.	RO	0x1

19.9.8.4 Device Control Register

Address Offset: `CFG_PCIE_CAP + 0x08

Bit#	Description	Attribute	Reset
15	Reserved	RsvdP	0x0
14:12	MAX_READ_REQUEST_SIZE	RW	0x2
11	Enable No Snoop	RW	1
10	AUX Power PM Enable	RW	0x0
9	Phantom Function Enable	RW	0x0
8	Extended Tag Field Enable	RW	0x0
7:5	MAX_PAYLOAD_SIZE	RW	0x0
4	Enable Relaxed Ordering	RW	0x1
3	Unsupported Request Reporting Enable	RW	0x0
2	Fatal Error Reporting Enable	RW	0x0
1	Non-Fatal Error Reporting Enable	RW	0x0
0	Correctable Error Reporting Enable	RW	0x0

19.9.8.5 Device Status Register

Address Offset: `CFG_PCIE_CAP + 0x0A

Bit#	Description	Attribute	Reset
15:6	Reserved	RsvdZ	0x000
5	Transaction Pending Set to '1' when Non-Posted Requests are not yet completed and clear when they are completed.	RO	0
4	Auxiliary Power Detected Set to '1' if Auxiliary power detected.	RO	0
3	Unsupported Request Detected Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.	RW1C	0
2	Fatal Error Detected Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.	RW1C	0
1	Non-Fatal Error detected Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.	RW1C	0
0	Correctable Error Detected Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.	RW1C	0

19.9.8.6 Link Capabilities Register

Address Offset: `CFG_PCIE_CAP + 0x0C

Bit#	Description	Attribute	Reset
31:24	Port Number	RW	0x0
23:22	Reserved	RsvdP	0x00
21	Link Bandwidth Notification Capability Set to '0' for Root Complex devices and '1' for Endpoint devices.	RO	0x0
20	Data Link Layer Active Reporting Capable Set to '0' for Root Complex devices and '1' for Endpoint devices.	RO	0x0
19	Surprise Down Error Reporting Capable Not supported, hardwired to 0x0.	RO	0x0
18	Clock Power Management	RO	0
17:15	L1 Exit Latency	RO	0x6
14:12	L0s Exit Latency	RO	0x3
11:10	Active State Link PM Support	RO	0x3
9:4	Maximum Link Width	RO	0x4
3:0	Maximum Link Speed Default value is 0x1 for 2.5 Gbps Link.	RO	0x1

19.9.8.7 Link Control Register

Address Offset: `CFG_PCIE_CAP + 0x10

Bit#	Description	Attribute	Reset
15:9	Reserved	RsvdP	0
8	Enable Clock Power Management Hardwired to '0' if Clock Power Management is disabled in the Link Capabilities register.	RW	0
7	Extended Synch	RW	0
6	Common Clock Configuration	RW	0
5	Retrain Link Not applicable for an upstream Port or Endpoint device, hardwired to '0'.	RW	0
4	Link Disable Not applicable for an upstream Port or Endpoint device, hardwired to '0'.	RW	0
3	Read Completion Boundary (RCB) *The parameter CX_RCB_SUPPORT decides whether this bit is RO (if set to '0') or RW (if set to '1').	*	0
2	Reserved	RsvdP	0
1:0	Active State Link PM Control	RW	0

19.9.8.8 Link Status Register

Address Offset: `CFG_PCIE_CAP + 0x12

Bit#	Description	Attribute	Reset
15:14	Reserved	RsvdZ	0x0
13	Data Link Layer Active Not applicable for an upstream Port or Endpoint device, hardwired to '0'.	RO	0x0
12	Slot Clock Configuration Indicates that the component uses the same physical reference clock that the platform provides on the connector. The default value is the value you select during hardware configuration.	RW	1
11	Link Training Not applicable for an upstream Port or Endpoint device, hardwired to '0'.	RO	0
10	Undefined for PCI Express 1.1 (Was Training Error for PCI Express 1.0a)	RO	0x0
9:4	Negotiated Link Width Set automatically by hardware after Link initialization.	RO	0x0
3:0	Link Speed The negotiated Link speed: 2.5 Gbps	RO	0x1

19.9.8.9 Slot Capabilities Register

Address Offset: `CFG_PCIE_CAP + 0x14

Bit#	Description	Attribute	Reset
31:19	Physical Slot Number	RW	0x0000
18	No Command Complete Support	RW	0
17	Electromechanical Interlock Present	RW	0
16:15	Slot Power Limit Scale	RW	0
14:7	Slot Power Limit Value	RW	0
6	Hot-Plug Capable	RW	0
5	Hot-Plug Surprise	RW	0
4	Power Indicator Present	RW	0
3	Attention Indicator Present	RW	0
2	MRL Sensor Present	RW	0
1	Power Controller Present	RW	0
0	Attention Button Present	RW	0

19.9.8.10 Slot Control Register

Address Offset: `CFG_PCIE_CAP + 0x18

Bit#	Description	Attribute	Reset
15:13	Reserved	RsvdP	0x0
12	Data Link Layer State Changed Enable Not applicable for an upstream Port or Endpoint device, hardwired to '0'.	RO	0x0
11	Electromechanical Interlock Control	RW	0
10	Power Controller Control	RW	0
9:8	Power Indicator Control	RW	0x3
7:6	Attention Indicator Control	RW	0x3
5	Hot-Plug Interrupt Enable	RW	0
4	Command Completed Interrupt Enable	RW	0
3	Presence Detect Changed Enable	RW	0
2	MRL Sensor Changed Enable	RW	0
1	Power Fault Detected Enable	RW	0
0	Attention Button Pressed Enable	RW	0

19.9.8.11 Slot Status Register

Address Offset: `CFG_PCIE_CAP + 0x1A

Bit#	Description	Attribute	Reset
15:9	Reserved	RsvdZ	0x0
8	Data Link Layer State Changed Not applicable for an upstream Port or Endpoint device, hardwired to '0'.	RO	0
7	Electromechanical Interlock Status	RO	0
6	Presence Detect State	RO	0
5	MRL Sensor State	RO	0
4	Command Completed	RWC1	0
3	Presence Detect Changed	RWC1	0
2	MRL Sensor Changed	RWC1	0
1	Power Fault Detected	RWC1	0
0	Attention Button Pressed	RWC1	0

19.9.9 PCI Express Extended Capabilities Register Details

The PCIe core implements the following PCI Express Extended Capabilities registers:

- Advanced Error Reporting Capability register set
- Virtual Channel Capability register set – only exists in the first function of a multifunction device
- Device Serial Number Capability register set – only exists in the first function of a multifunction device

The following sections describe the individual registers in each group. For register maps, see [Section 19.9.2 Register Maps](#).

19.9.9.1 Advanced Error Reporting Capability Registers

PCI Express Enhanced Capability Header

Address Offset: 0x100

Bit#	Description	Attribute	Reset
31:20	Next Capability Offset Points to the Virtual Channel Capability structure by default.	RO	0x140
19:16	Capability Version	RO	0x0
15:0	PCI Express Extended Capability ID Default value is 0x1 for Advanced Error Reporting.	RO	0x0

Uncorrectable Error Status Register

Address Offset: 0x04

Bit#	Description	Attribute	Reset
31:21	Reserved	RsvdZ	0x000
20	Unsupported Request Error Status	RW1CS	0
19	ECRC Error Status	RW1CS	0
18	Malformed TLP Status	RW1CS	0
17	Receiver Overflow Status	RW1CS	0
16	Unexpected Completion Status	RW1CS	0
15	Completer Abort Status	RW1CS	0
14	Completion Timeout Status	RW1CS	0
13	Flow Control Protocol Error Status	RW1CS	0
12	Poisoned TLP Status	RW1CS	0
11:6	Reserved	RsvdZ	0x00
5	Surprise Down Error Status (not supported)	RO	0
4	Data Link Protocol Error Status	RW1CS	0
3:1	Reserved	RsvdZ	0x0
0	Undefined for PCI Express 1.1 (Was Training Error Status for PCI Express 1.0a)	Undefined	0

Uncorrectable Error Mask Register

Address Offset: 0x08

Bit#	Description	Attribute	Reset
31:21	Reserved	RsvdP	0x000
20	Unsupported Request Error Mask	RWS	0
19	ECRC Error Mask	RWS	0
18	Malformed TLP Mask	RWS	0
17	Receiver Overflow Mask	RWS	0
16	Unexpected Completion Mask	RWS	0
15	Completer Abort Mask	RWS	0
14	Completion Timeout Mask	RWS	0
13	Flow Control Protocol Error Mask	RWS	0
12	Poisoned TLP Mask	RWS	0x00
11:6	Reserved	RsvdP	0x0
5	Surprise Down Error Mask (not supported)	RO	0
4	Data Link Protocol Error Mask	RWS	0
3:1	Reserved	RsvdP	0x0
0	Undefined for PCI Express 1.1 (Was Training Error Mask for PCI Express 1.0a)	Undefined	0

Uncorrectable Error Severity Register

Address Offset: 0x0C

Bit#	Description	Attribute	Reset
31:21	Reserved	RsvdP	0x000
20	Unsupported Request Error Severity	RWS	0
19	ECRC Error Severity	RWS	0
18	Malformed TLP Severity	RWS	1
17	Receiver Overflow Severity	RWS	1
16	Unexpected Completion Severity	RWS	0
15	Completer Abort Severity	RWS	0
14	Completion Timeout Severity	RWS	0
13	Flow Control Protocol Error Severity	RWS	1
12	Poisoned TLP Severity	RWS	0
11:6	Reserved	RsvdP	0x00
5	Surprise Down Error Severity (not supported)	RO	1
4	Data Link Protocol Error Severity	RWS	1
3:1	Reserved	RsvdP	0x0
0	Undefined for PCI Express 1.1 (Was Training Error Severity for PCI Express 1.0a)	Undefined	1

Correctable Error Status Register

Address Offset: 0x10

Bit#	Description	Attribute	Reset
31:14	<i>Reserved</i>	RsvdZ	0x00000
13	Advisory Non-Fatal Error Status	RW1CS	0
12	Reply Timer Timeout Status	RW1CS	0
11:9	<i>Reserved</i>	RsvdZ	0x00
8	REPLAY_NUM Rollover Status	RW1CS	0
7	Bad DLLP Status	RW1CS	0
6	Bad TLP Status	RW1CS	0
5:1	<i>Reserved</i>	RsvdZ	0x00
0	Receiver Error Status	RW1CS	0

Correctable Error Mask Register

Address Offset: 0x14

Bit#	Description	Attribute	Reset
31:14	<i>Reserved</i>	RsvdP	0x00000
13	Advisory Non-Fatal Error Mask	RWS	0x1
12	Reply Timer Timeout Mask	RWS	0
11:9	<i>Reserved</i>	RsvdP	0x0
8	REPLAY_NUM Rollover Mask	RWS	0
7	Bad DLLP Mask	RWS	0
6	Bad TLP Mask	RWS	0
5:1	<i>Reserved</i>	RsvdP	0x00
0	Receiver Error Mask	RWS	0

Advanced Capabilities and Control Register

Address Offset: 0x18

Bit#	Description	Attribute	Reset
31:9	<i>Reserved</i>	RsvdP	0x000000
8	ECRC Check Enable	RWS	0
7	ECRC Check Capable	RO	1
6	ECRC Generation Enable	RWS	0
5	ECRC Generation Capability	RO	1
4:0	First Error Pointer	ROS	0x00

Header Log Registers

The Header Log registers collect the header for the TLP corresponding to a detected error. See the PCI Express Base Specification for details. Each of the Header Log registers is type ROS; the default reset value of each Header Log register is 0x00000000.

Address Offset: 0x1C

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x1C	Header Log Register (first DWord)			
0x20	Header Log Register (second DWord)			
0x24	Header Log Register (third DWord)			
0x28	Header Log Register (fourth DWord)			

19.9.9.2 Device Serial Number Capability Register***Device Serial Number Enhanced Capability Header***

Address Offset: 0x140

Bit#	Description	Attribute	Reset
31:20	Points to end of capabilities.	RW	0x0
19:16	Capability Version	RW	0x1
15:0	Extended Capability ID	RW	0x0003

Serial Number Register

Address Offset: `SN_PTR + 0x4/+0x8

Bit#	Description	Attribute	Reset
	Serial Number register (1st DWord)	RO	0x4
	Serial Number register (2nd DWord)	RO	0x8

19.10 PCIe Configuration Registers: RC Mode

This section describes the interface's PCIe Configuration registers in RC mode.

All reset values are in *hex* unless otherwise noted.

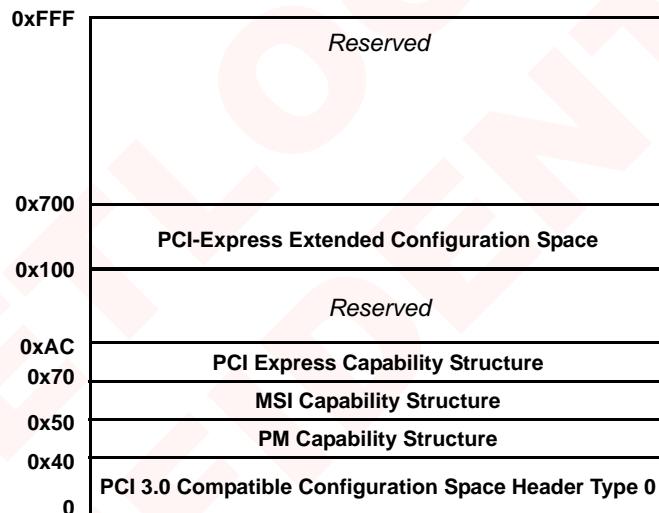
19.10.1 Register Space Layout

The PCIe core has 4096 bytes of PCI Express configuration space. The PCI Express configuration space is divided into:

- 256 bytes of PCI 3.0 Compatible Configuration Space Header (type 1 in RC mode)
- PCI Capabilities structures, which start at offset 0x40
- PCI Express Extended Configuration Space, which starts at offset 0x100
- Port Logic registers (vendor-specific registers), which start at offset 0x700. The Port Logic registers have specific pre-defined usages, mostly for test purposes, and can optionally be removed from the PCIe core hardware configuration.

The following figure shows the RC mode layout of the PCIe core configuration space.

Figure 19-11. RC Mode PCIe core Configuration Space Layout



19.10.2 Register Maps

Configuration registers are in structures (groups) identified by a Capability ID. Groups are linked together as in PCI. Register locations within a group are specified, but the starting location of each group must be found by traversing the linked list. There are two linked lists of register groups: PCI-compatible base registers and PCI-Express Enhanced Capability registers. PCI-compatible base register groups begin at configuration address 0x00. PCI Express Enhanced Capability register groups begin at address 0x100.

19.10.2.1 PCI Configuration Space Header — Type 1

Table 19-22 shows the configuration header field definitions for the PCI Express Type 1 Configuration Space Header. Most PCI-compatible header fields have the same software interpretation in PCI 3.0 and PCI Express.

Table 19-22. PCI Configuration Space Header - Type 1

Byte Address Offset	Byte 3	Byte 2	Byte 1	Byte 0		
0x00	Device ID			Device ID		
0x04	Status Register			Status Register		
0x08	Class Code			Revision ID		
0x0C	BIST(0x00)	Header Type	Latency Timer	Cache Line Size		
0x10	Base Address Register 0					
0x14	Base Address Register 1					
0x18	Secondary Latency Timer	Subordinate Bus Number	Secondary Bus Number	Primary Bus Number		
0x1C	Secondary Status		I/O Limit	I/O Base		
0x20	Memory Limit		Memory Base			
0x24	Prefetchable Memory Limit		Prefetchable Memory Base			
0x28	Prefetchable Base Upper 32 Bits					
0x2C	Prefetchable Limit 32 Upper Bits					
0x30	I/O Limit Upper 32 Bits		I/O Base Upper 32 Bits			
0x34	Reserved			CapPtr		
0x38	Expansion ROM Base Address					
0x3C	Bridge Control		Interrupt Pin	Interrupt Line		

19.10.2.2 Capability Structures Register Maps

Table 19-23. Configuration Structure: Starting Addresses and Next Capability Pointers

Start Address Offset	Item	Next Pointer
0x00	PCI-Compatible Header (Type 1)	0x40
0x40	PCI Power Management	0x50
0x50	Message Signaled Interrupt (MSI)	0x70
0x70	PCI Express Capabilities	0x00

The following tables show the PCI capability structures.

Table 19-24. Power Management Capability Structure

Byte Address Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x40	Power Management Capabilities (PMC)		Next Capability Pointer ('PM_NEXT_PTR)	Capability ID (0x01)
+0x4	Data	PMCSR_BSE Bridge Extensions	Power Management Control Status Register (PMCSR)	

Table 19-25. MSI Capability Structure

Byte Address Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x50	Message Control Register		Next Capability Pointer (`MSI_NEXT_PT)	Capability ID (0x05)
+0x4		MSI Lower 32-bit Address Register		
+0x8		MSI Upper 32-bit Address Register		
+0xC				MSI Data

Table 19-26. PCI Express Capability Structure

Byte Address Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x70	PCI Express Capabilities Register		Next Capability Pointer (`PCIE_NEXT_PT)	Capability ID (0x10)
+0x4		Device Capabilities		
+0x8	Device Status		Device Control	
+0xC		Link Capabilities		
+0x10	Link Status		Link Control	
+0x14		Slot Capabilities		
+0x18	Slot Status		Slot Control	
+0x1C	Root Capabilities		Root Capabilities	
+0x20		Root Status		

19.10.2.3**PCI Express Extended Capability Register Maps**

The PCI Express Extended Capabilities registers are located in device configuration space at offsets 0x100 or higher. As with PCI Capabilities, the PCI Express Extended Capability structures are allocated using a linked list with a similar method and format to those of PCI.

The following tables outline the PCI Express Extended Capabilities structures.

Table 19-27. Advanced Error Reporting Capability Structure

Byte Address Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x010	PCI Express Enhanced Capability Header			
+0x4	Uncorrectable Error Status Register			
+0x8	Uncorrectable Error Mask Register			
+0xC	Uncorrectable Error Severity Register			
+0x10	Correctable Error Status Register			
+0x14	Correctable Error Mask Register			
+0x18	Advanced Error Capabilities and Control Register			
+0x1C through +0x28	Header Log Registers			
+0x2C	Root Error Command Register			
+0x30	Root Error Status Register			
+0x34	Error Source Identification Register			

Table 19-28. Device Serial Number Capability Register

Byte Address Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x140	Device Serial Number Enhanced Capability Header			
+0x4	Serial Number Register (1st DWord)			
+0x8	Serial Number Register (2nd DWord)			

19.10.3 Accessing Configuration Registers

The application software can access the configuration space of the PCIe ports by reading and writing to the PCIE_CFG_BAR and PCIE_ECFG_BAR. Bus Number, Function Number, and Device Number of the address must be set to specific values.

In the access request, the Bus Number and Function Number of the address must be 0x0.

For XLS6xx, XLS4xx and XLS2xx devices, the Device Number of the address is:

- 0x0 for Link 0 in Root Complex
- 0x1 for Link 1
- 0x2 for Link 2
- 0x3 for Link 3
- 0x5 for Link 0 in EndPoint

For XLS4xxLite devices, the Device Number of the address is:

- 0 for Link 0 in Root Complex
- 3 for Link 0 in EndPoint
- 1 for Link 1

For XLS1xx devices, the Device Number of the address is:

- 0x0 for Link 0 in Root Complex
- 0x1 for Link 1
- 0x5 for Link 0 in EndPoint

In the descriptions of configuration space registers in this chapter, the Attribute column indicates the read/write access for the register or bit. [Table 19-29](#) defines the read/write attribute abbreviations used. The definitions match the PCI Express Base Specification.

Table 19-29. Configuration Register Bit-Field Types

Attribute	Meaning	Description
Hwinit	Hardware Initialized	HwInit bits are controlled by PCIe core hardware and are read-only by host system software.
RO	Read Only	Register bits are read-only and should not be altered by user or application software.
RW	Read-Write	Register bits are read-write and may be read and written normally by the host and the application. Writing from the application side (if any) requires careful synchronization with the host software.
RW1C	Read-Only Status/Write-1-to-Clear Status Register	Register bits indicate status when read. A set bit indicating a status event may be cleared by writing a '1'. Writing '0' to RW1C bits has no effect. Writing from the application side (if any) requires careful synchronization with host software.
ROS	Sticky Read-Only	Registers are read-only and cannot be altered by host or application software, except as noted. Registers are not initialized or modified by a hot reset. A few bits designated as very sticky are not cleared by any type of PCIe core reset when auxiliary power is supplied and enabled.
RWS	Sticky Read-Write	Registers are read-write and may be either set or cleared by host or application software to the desired state. Bits are not initialized or modified by a hot reset. A few bits designated very sticky are not cleared by any type of PCIe core reset when auxiliary power is supplied and enabled.
RW1CS	Sticky Read-Only Status/Write-1-to-Clear Status	Writing a '1' clears status for the host and the application, except as noted. Registers indicate status when read. A set bit indicating a status event may be cleared by writing a '1'. Writing '0' to RW1CS bits has no effect. Bits are not initialized or modified by a hot reset. A few bits designated very sticky are not cleared by any type of PCIe core reset when auxiliary power is supplied and enabled.
RsvdP	Reserved and PReserved	Reserved for future RW implementations: software must preserve the value read when writing to other bits in the same register.
RsvdZ	Reserved and Zero	Reserved for future RW1C implementations: software must write '0' to these bits when writing to other bits in the same register.

19.10.4

Register Default Values

You can select the default reset values of several of the configuration registers through design configuration parameters. The register descriptions in this chapter indicate the default reset value of the register, either as a specific numerical value or as the name of the configuration parameter that sets the default reset value.

19.10.5 PCI-Compatible Configuration Header Register Details

19.10.5.1 Device ID and Vendor ID Register

Address Offset: 0x04h

The following table defines the Device ID and Vendor ID register bit assignments. The default values of both Device ID and Vendor ID are hardware configuration parameters.

Bit#	Description	Attribute	Reset
15:0	Vendor ID	RO	0x182E
31:16	Device ID -- provided by each device type	RO	0xABCD

19.10.5.2 Command Register

The following table defines the PCI-compatible Command register bit assignments.

Address Offset: 0x04h

Bytes: 0-1

Bit#	Description	Attribute	Reset
15:11	Reserved	RO	0x0
10	INTX Assertion Disable	RW	0
9	Fast Back-to-Back Enable Not applicable for PCI Express. Must be hardwired to '0'.	RO	0
8	SERR# Enable	RW	0
7	IDSEL Stepping/Wait Cycle Control Not applicable for PCI Express. Must be hardwired to '0'.	RO	0
6	Parity Error Response	RW	0
5	VGA Palette Snoop Not applicable for PCI Express. Must be hardwired to '0'.	RO	0
4	Memory Write and Invalidate Not applicable for PCI Express. Must be hardwired to '0'.	RO	0
3	Special Cycle Enable Not applicable for PCI Express. Must be hardwired to '0'.	RO	0
2	Bus Master Enable	RW	0
1	Memory Space Enable	RW	0
0	I/O Space Enable	RW	0

19.10.5.3 Status Register

The following table defines the PCI-compatible Status register bit assignments.

Address Offset: 0x04h

Bytes: 2-3

Bit#	Description	Attribute	Reset
15	Detected Parity Error	RW1C	0
14	Signaled System Error	RW1C	0
13	Received Master Abort	RW1C	0
12	Received Target Abort	RW1C	0
11	Signaled Target Abort	RW1C	0
10:9	DEVSEL Timing Not applicable for PCI Express. Hardwired to '0'.	RO	0x0
8	Master Data Parity Error	RW1C	0
7	Fast Back-to-Back Capable Not applicable for PCI Express. Hardwired to '0'.	RO	0
6	<i>Reserved</i>	RO	0
5	66.67 MHz Capable Not applicable for PCI Express. Hardwired to '0'.	RO	0
4	Capabilities List Indicates presence of an extended capability item. Hardwired to '1'.	RO	1
3	INTX Status	RO	0
2:0	<i>Reserved</i>	RO	0x0

19.10.5.4 Revision ID Register

Address Offset: 0x08

Byte: 0

Bit#	Description	Attribute	Reset
7:0	Revision ID	RO	0x01

19.10.5.5 Class Code Register

Address Offset: 0x08

Bytes: 1-3

Bit#	Description	Attribute	Reset
23:16	Base Class Code	RO	0x06
15:8	Subclass Code	RO	0x04
7:0	Programming Interface	RO	0x00

19.10.5.6 Cache Line Size Register

Address Offset: 0x0C

Byte: 0

Bit#	Description	Attribute	Reset
7:0	Cache Line Size The Cache Line Size register is RW for legacy compatibility purposes and is not applicable to PCI Express device functionality. Writing to the Cache Line Size register does not impact functionality of the PCIe core.	RW	0x00

19.10.5.7 Master Latency Timer Register

Address Offset: 0x0C

Byte: 1

Bit#	Description	Attribute	Reset
7:0	Master Latency Timer Not applicable for PCI Express, hardwired to '0'.	RO	0x00

19.10.5.8 Header Type Register

Address Offset: 0x0C

Byte: 2

Bit#	Description	Attribute	Reset
6:0	Configuration Header Format Hardwired to 0x01.	RO	0x01
7	Multi Function Device	RO	0x0

19.10.5.9 BIST Register

Address Offset: 0x0C

Byte: 3

Bit#	Description	Attribute	Reset
7:0	The BIST register functions are not supported by the PCIe core. All 8 bits of the BIST register are hardwired to '0'.	RO	0x00

19.10.5.10 Base Address Registers

Address Offset: 0x10–0x14

Base Address Register 0Address 0x10 (if included in the PCIe core hardware configuration)
Offset:

Bit#	Description	Attribute	Reset
31:4	BAR 0 base address bits (for a 64-bit BAR, the remaining upper address bits are in BAR 1). The BAR 0 Mask value determines which address bits are masked.	RW	0x0000000
3	If BAR 0 is a memory BAR, bit [3] indicates if the memory region is prefetchable: ‘0’ = Non-prefetchable ‘1’ = Prefetchable If BAR 0 is an I/O BAR, bit [3] is the second least significant bit of the base address.	RW	0x1
2:1	If BAR 0 is a memory BAR, bits [2:1] determine the BAR type: ‘00’ = 32-bit BAR ‘10’ = 64-bit BAR If BAR 0 is an I/O BAR, bit [2] the least significant bit of the base address and bit [1] is ‘0’.	RW	0x2
0	‘0’ = BAR 0 is a memory BAR ‘1’ = BAR 0 is an I/O BAR	RW	0x0

Base Address Register 1

Address Offset: 0x14

Bit#	Description	Attribute	Reset
31:0	If BAR 0 is a 64-bit BAR, BAR 1 contains the upper 32 bits of the BAR 0 base address (bits [63:32]). If BAR 0 is a 32-bit BAR, BAR 1 can be independently programmed as an additional 32-bit BAR or can be excluded from the PCIe core hardware configuration. If programmed as an independent 32-bit BAR, the BAR 1 bit definitions are the same as the BAR 0 bit definitions.	RO	0x0

19.10.5.11 BAR Mask Registers

BAR 0 Mask Register

Address Offset: 0x10

Bit#	Description	Attribute	Reset
31:1	Indicates which BAR 0 bits to mask (make non-writable) from host software, which, in turn, determines the size of the BAR. For example, writing 0xFFFF to the BAR 0 Mask register claims a 4096-byte BAR by masking bits [11:0] of the BAR from writing by host software. If BAR 0 is a 64-bit BAR, then the BAR 1 Mask register contains the upper bits of the BAR 0 Mask. The BAR 0 Mask register is invisible to host software and not readable from the application.	Application write access only; not readable	0x1FFFFF
0	BAR 0 Enable ‘0’ = BAR 0 is disabled ‘1’ = BAR 0 is enabled Bit [0] is interpreted as BAR Enable when writing to the BAR Mask register rather than as a mask bit because bit [0] of a BAR is always masked from writing by host software.	Same as bits [31:1]	1

BAR 1 Mask Register

Address Offset: 0x14

Bit#	Description	Attribute	Reset
31:0	If BAR 0 is a 32-bit BAR: <ul style="list-style-type: none">• Bits [31:1]: BAR 1 Mask value, interpreted the same way as BAR 0 Mask.• Bit [0]: BAR 1 Enable ('0' = BAR 1 is disabled; '1' = BAR 1 is enabled). If BAR 0 is a 64-bit BAR: <ul style="list-style-type: none">• Bits [31:0] are the upper bits of the BAR 0 Mask value.	Application write access only	0x0

Expansion ROM BAR Mask Register

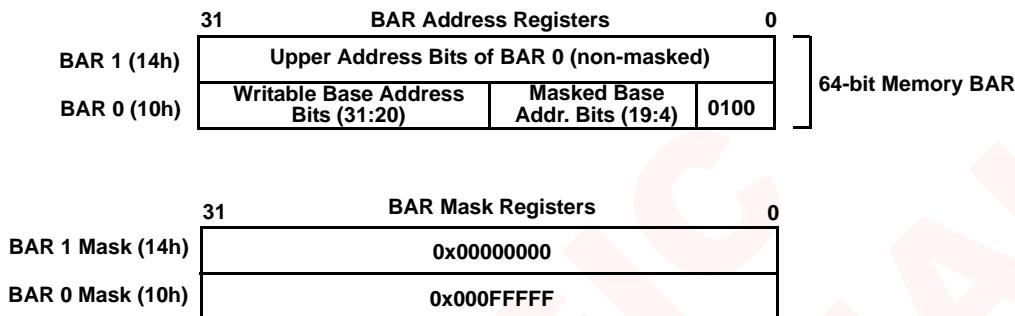
Address Offset: 0x38

Bit#	Description	Attribute	Reset
31:1	Indicates which Expansion ROM BAR bits to mask (make non-writable) from host software, which, in turn, determines the size of the BAR. For example, writing 0xFFFF to the Expansion ROM BAR Mask register claims a 4096-byte BAR by masking bits [11:0] of the BAR from writing by host software. The maximum value is 0xFFFFFFFF because the maximum space that can be claimed by an Expansion ROM BAR is 16 MB. The Expansion ROM BAR Mask register is invisible to host software and not readable from the application.	Application write access only	0
0	Expansion ROM BAR Enable ‘0’ = Expansion ROM BAR is disabled ‘1’ = Expansion ROM BAR is enabled	Same as bits [31:1]	0

Example BAR Setup

Figure 19-12 shows an example configuration of the BARs and their corresponding BAR Mask registers. The example configuration includes one 64-bit memory BAR (non-prefetchable).

Figure 19-12. Example Base Address Register Configuration



19.10.5.12 Bus Number Registers

Address
Offset: 0x18

Bit#	Description	Attribute	Reset
31:24	Secondary Latency Timer Not applicable to PCI Express, hardwired to 0x00.	R0	0x00
23:16	Subordinate Bus Number	RW	0x00
15:8	Secondary Bus Number	RW	0x00
7:0	Primary Bus Number	RW	0x00

19.10.5.13 I/O Base and I/O Limit Register

Address Offset: 0x1C

Bytes: 0-1

Bit#	Description	Attribute	Reset
15:12	I/O Space Limit	RW	0x00
11:9	Reserved	RO	0x00
8	32-Bit I/O Limit <i>'0' = Reserved; do not clear this bit to '0'. '1' = 32-bit I/O addressing This bit should be set to '1'.</i>	RO	0x1
7:4	I/O Space Base	RW	0x00
3:1	Reserved	RO	0x00
0	32-Bit I/O Base <i>'0' = Reserved; do not clear this bit to '0'. '1' = 32-bit I/O addressing This bit should be set to '1'.</i>	RO	0x1

19.10.5.14 Secondary Status Register

Address Offset: 0x1C

Bytes: 2-3

Bit#	Description	Attribute	Reset
15	Detected Parity Error	RW1C	0
14	Received System Error	RW1C	0
13	Received Master Abort	RW1C	0
12	Received Target Abort	RW1C	0
11	Signaled Target Abort	RW1C	0
10:9	DEVSEL Timing Not applicable to PCI Express, hardwired to '0'.	RO	0x0
8	Master Data Parity Error	RW1C	0
7	Fast Back-to-Back Capable Not applicable to PCI Express, hardwired to '0'.	RO	0
6	Reserved	RO	0
5	66.67 MHz Capable Not applicable to PCI Express, hardwired to '0'.	RO	0
4:0	Reserved	RO	0x0

19.10.5.15 Memory Base and Memory Limit Register

Address Offset: 0x20

Bit#	Description	Attribute	Reset
31:20	Memory Limit Address	RW	0x00
19:16	<i>Reserved</i>	RO	0x00
15:4	Memory Base Address	RW	0x00
3:0	<i>Reserved</i>	RO	0x00

19.10.5.16 Prefetchable Memory Base and Limit Register

Address Offset: 0x24

Bit#	Description	Attribute	Reset
32:20	Upper 12 bits of 32-bit Prefetchable Memory End Address	RW	0x000
19:17	<i>Reserved</i>	RO	0x0
16	64-Bit Memory Addressing ‘0’ = 16-bit memory addressing ‘1’ = 32-bit memory addressing	RO	1
15:4	Upper 12 bits of 32-bit Prefetchable Memory Start Address	RW	0x000
3:1	<i>Reserved</i>	RO	0x0
0	64-Bit Memory Addressing ‘0’ = 16-bit memory addressing ‘1’ = 32-bit memory addressing	RO	1

19.10.5.17 Prefetchable Base Upper 32 Bits Register

Address Offset: 0x28

Bit#	Description	Attribute	Reset
31:0	Upper 32 Bits of Base Address of Prefetchable Memory Space Used only when 64-bit prefetchable memory addressing is enabled.	RW	0x00000000

19.10.5.18 Prefetchable Limit Upper 32 Bits Register

Address Offset: 0x2C

Bit#	Description	Attribute	Reset
31:0	Upper 32 Bits of Base Address of Prefetchable Memory Space Used only when 64-bit prefetchable memory addressing is enabled.	RW	0x00000000

19.10.5.19 I/O Base and Limit Upper 16 Bits Register

Address Offset: 0x30

Bit#	Description	Attribute	Reset
31:16	Upper 16 Bits of I/O Limit (if 32-bit I/O decoding is supported for devices on the secondary side)	RW	0x0000
15:0	Upper 16 Bits of I/O Base (if 32-bit I/O decoding is supported for devices on the secondary side)	RW	0x0000

19.10.5.20 Capability Pointer Register

Address Offset: 0x34

Byte: 0

Bit#	Description	Attribute	Reset
7:0	First Capability Pointer. Points to Power Management Capability structure	RO	0x40

19.10.5.21 Expansion ROM Base Address Register

Address Offset: 0x38

Bit#	Description	Attribute	Reset
31:11	Expansion ROM Address	RW	0x0000
10:1	Reserved	RO	0x000
0	Expansion ROM Enable	RW	0x0

19.10.5.22 Interrupt Line Register

Address Offset: 0x3C

Byte: 0

Bit#	Description	Attribute	Reset
7:0	Interrupt Line	RW	0xFF

19.10.5.23 Interrupt Pin Register**Address Offset:** 0x3C**Byte:** 1

The Interrupt Pin register exists in the PCIe core; however, the PCIe core does not generate ASSERT_INTX or DEASSERT_INTX Messages.

Bit#	Description	Attribute	Reset
7:0	Interrupt Pin Identifies the legacy interrupt Message that the device (or device function) uses. Valid values are: 0x00: The device (or function) does not use legacy interrupt 0x01: The device (or function) uses INTA 0x02: The device (or function) uses INTB 0x03: The device (or function) uses INTC 0x04: The device (or function) uses INTD	RO	0x1

19.10.5.24 Bridge Control Register**Address Offset:** 0x3C**Bytes:** 2-3

Bit#	Description	Attribute	Reset
15:12	<i>Reserved</i>	RO	0x0
11	Discard Timer SERR Enable Status Not applicable to PCI Express, hardwired to '0'.	RO	0
10	Discard Timer Status Not applicable to PCI Express, hardwired to '0'.	RO	0
9	Secondary Discard Timer Not applicable to PCI Express, hardwired to '0'.	RO	0
8	Primary Discard Timer Not applicable to PCI Express, hardwired to '0'.	RO	0
7	Fast Back-to-Back Transactions Enable Not applicable to PCI Express, hardwired to '0'.	RO	0
6	Secondary Bus Reset	RW	0
5	Master Abort Mode Not applicable to PCI Express, hardwired to '0'.	RO	0
4	VGA 16-Bit Decode	RW	0
3	VGA Enable	RW	0
2	ISA Enable	RW	0
1	SERR Enable	RW	0
0	Parity Error Response Enable	RW	0

19.10.6 PCI Power Management Capability Register Details

The PCIe core implements power management capabilities. The Capability Pointer field in the configuration header points to the PCI Power Management registers as the first extended capability by default.

The extent of the power management implementation in the PCIe core includes:

- Power Management register space
- Link state information (provided to both the application logic and PHY interfaces)
- Power management-ready clock and reset implementation

The following sections describe the PCI Power Management registers implemented in the PCIe core. See the PCI Power Management Specification and the PCI Express Base Specification for more details.

19.10.6.1 Power Management Capability ID Register

Address Offset: 0x40

Bit#	Description	Attribute	Reset
7:0	Power Management Capability ID	RO	0x01

19.10.6.2 Power Management Next Item Pointer

Address Offset: `CFG_PM_CAP + 0x01

Bit#	Description	Attribute	Reset
7:0	Next Capability Pointer Points to the MSI capabilities	RO	0x50

19.10.6.3 Power Management Capabilities Register

Address Offset: `CFG_PM_CAP + 0x02

Bit#	Description	Attribute	Reset
15:11	PME_Support Identifies the power states from which the PCIe core can generate PME Messages. A value of '0' for any bit indicates that the device (or function) is not capable of generating PME Messages while in that power state: <ul style="list-style-type: none">• Bit [11]: If set, PME Messages can be generated from D0• Bit [12]: If set, PME Messages can be generated from D1• Bit [13]: If set, PME Messages can be generated from D2• Bit [14]: If set, PME Messages can be generated from D3hot• Bit [15]: If set, PME Messages can be generated from D3cold	RO	0x1B
10	D2 Support	RO	0
9	D1 Support	RO	1
8:6	AUX Current	RO	0x7
5	Device Specific Initialization (DSI)	RO	0
4	<i>Reserved</i>	RsvdP	0
3	PME Clock, hardwired to '0'	RO	0
2:0	Power Management Specification Version	RO	0x011

19.10.6.4 Power Management Control and Status Register

Address Offset: `CFG_PM_CAP + 0x04

Bit#	Description	Attribute	Reset
31:24	Data register for additional information (not supported)	RO	0x00
23	Bus Power/Clock Control Enable, hardwired to '0'	RO	0x0
22	B2/B3 Support, hardwired to '0'	RO	0x0
21:16	<i>Reserved</i>	RsvdP	0x0
15	PME Status Indicates if a previously enabled PME event occurred or not.	RW1CS	0x0
14:13	Data Scale (not supported)	RO	0x0
12:9	Data Select (not supported)	RO	0x0
8	PME Enable (sticky bit) A value of 1 indicates that the device is enabled to generate PME.	RWS	0x0
7:4	<i>Reserved</i>	RsvdP	0x0
3	No Soft Reset	RO	0
2	<i>Reserved</i>	RsvdP	0x0
1:0	Power State Controls the device power state: 00b: D0 01b: D1 10b: D2 11b: D3 The written value is ignored if the specific state is not supported.	RW	0x0

19.10.7 MSI Capability Register Details

19.10.7.1 MSI Capability ID

Address Offset: 0x50

Bit#	Description	Attribute	Reset
7:0	MSI Capability ID	RO	0x05

19.10.7.2 MSI Next Item Pointer

Address Offset: `CFG_MSI_CAP + 0x01

Bit#	Description	Attribute	Reset
7:0	Next Capability Pointer Points to PCI Express Capabilities	RO	0x70

19.10.7.3 MSI Control Register

Address Offset: `CFG_MSI_CAP + 0x02

Bit#	Description	Attribute	Reset
15:8	Reserved	RO	0x00
7	64-bit Address Capable	RO	1
6:4	Multiple Message Enabled Indicates that multiple Message mode is enabled by system software. The number of Messages enabled must be less than or equal to the Multiple Message Capable value.	RW	0x0
3:1	Multiple Message Capable	RO	0
0	MSI Enabled When set, INTX must be disabled.	RW	0

19.10.7.4 MSI Lower 32 Bits Address Register

Address Offset: `CFG_MSI_CAP + 0x04

Bit#	Description	Attribute	Reset
31:2	Lower 32-bit Address	RW	0x00000000
1:0	Reserved	RO	0x0

19.10.7.5 MSI Upper 32 bits Address Register

Address Offset: `CFG_MSI_CAP + 0x08

Bit#	Description	Attribute	Reset
31:0	Upper 32-bit Address	RW	0x00000000

19.10.7.6 MSI Data Register

Address Offset: `CFG_MSI_CAP + 0x0C

Bit#	Description	Attribute	Reset
31:16	Reserved	RO	0x0000
15:0	MSI Data Pattern assigned by system software, bits [4:0] are Or-ed with MSI_VECTOR to generate 32 MSI Messages per function.	RW	0x0000

19.10.8 PCI Express Capability Register Details

The PCIe core implements the PCI Express Capability Structure as defined in the PCI Express Base Specification.

19.10.8.1 PCI Express Capability List Register

Address Offset: `CFG_PCIE_CAP

Bit#	Description	Attribute	Reset
15:8	Next Capability Pointer	RO	0x00
7:0	PCI Express Capability ID	RO	0x10

19.10.8.2 PCI Express Capabilities Register

Address Offset: `CFG_PCIE_CAP + 0x02

Bit#	Description	Attribute	Reset
15:14	RsvdP	RO	0x0
13:9	Interrupt Message Number (updated by hardware)	RO	0
8	Slot Implemented	Hwinit	0
7:4	Device Port Type	RO	0100b
3:0	PCI Express Capability Version	RO	0x1

19.10.8.3 Device Capabilities Register

Address Offset: `CFG_PCIE_CAP + 0x04

Bit#	Description	Attribute	Reset
31:28	Reserved	RsvdP	0x0
27:26	Captured Slot Power Limit Scale Not applicable for RC Port, upstream port only.	RO	0x0
25:18	Captured Slot Power Limit Value Not applicable for RC Port, upstream port only.	RO	0x00
17:16	Reserved	RsvdP	0x0
15	Role-Based Error Reporting	RO	0x1
14	Undefined for PCI Express 1.1 (Was Power Indicator Present for PCI Express 1.0a)	RO	0x0
13	Undefined for PCI Express 1.1 (Was Attention Indicator Present for PCI Express 1.0a)	RO	0x0
12	Undefined for PCI Express 1.1 (Was Attention Button Present for PCI Express 1.0a)	RO	0x0
11:9	Endpoint L1 Acceptable Latency Must be 0x0 for non-Endpoint devices.	RO	0x0
8:6	Endpoint L0s Acceptable Latency Must be 0x0 for non-Endpoint devices.	RO	0x0
5	Extended Tag Field Supported	RO	0
4:3	Phantom Function Supported	RO	0x0
2:0	MAX_PAYLOAD_SIZE (256 bytes) Supported	RO	0x1

19.10.8.4 Device Control Register

Address Offset: `CFG_PCIE_CAP + 0x08

Bit#	Description	Attribute	Reset
15	Reserved	RsvdP	0x0
14:12	MAX_READ_REQUEST_SIZE	RW	0x2
11	Enable No Snoop	RW	1
10	AUX Power PM Enable	RWS	0x0
9	Phantom Function Enable	RW	0x0
8	Extended Tag Field Enable	RW	0x0
7:5	MAX_PAYLOAD_SIZE	RW	0x0
4	Enable Relaxed Ordering	RW	0x1
3	Unsupported Request Reporting Enable	RW	0x0
2	Fatal Error Reporting Enable	RW	0x0
1	Non-Fatal Error Reporting Enable	RW	0x0
0	Correctable Error Reporting Enable	RW	0x0

19.10.8.5 Device Status Register

Address Offset: `CFG_PCIE_CAP + 0x0A

Bit#	Description	Attribute	Reset
15:6	Reserved	RsvdZ	0x000
5	Transaction Pending Hard-wired to '0'.	RO	0
4	Auxiliary Power Detected Set to 1 if Auxiliary power detected.	RO	0
3	Unsupported Request Detected Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.	RW1C	0
2	Fatal Error Detected Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.	RW1C	0
1	Non-Fatal Error detected Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.	RW1C	0
0	Correctable Error Detected Errors are logged in this register regardless of whether error reporting is enabled in the Device Control register.	RW1C	0

19.10.8.6 Link Capabilities Register

Address Offset: `CFG_PCIE_CAP + 0x0C

Bit#	Description	Attribute	Reset
31:24	Port Number	RW	0x0
23:22	Reserved	RsvdP	0x00
21	Link Bandwidth Notification Capability Set to '0' for Root Complex devices and '1' for Endpoint devices.	RO	0x0
20	Data Link Layer Active Reporting Capable Set to '0' for Root Complex devices and '1' for Endpoint devices.	RO	0x1
19	Surprise Down Error Reporting Capable Not supported, hardwired to 0x0.	RO	0x0
18	Clock Power Management	RO	0
17:15	L1 Exit Latency	RO	0x6
14:12	L0s Exit Latency	RO	0x3
11:10	Active State Link PM Support	RO	0x3
9:4	Maximum Link Width	RO	0x4
3:0	Maximum Link Speed Default value is 0x1 for 2.5 Gbps Link.	RO	0x1

19.10.8.7 Link Control Register

Address Offset: `CFG_PCIE_CAP + 0x10

Bit#	Description	Attribute	Reset
15:9	Reserved	RsvdP	0x00
8	Enable Clock Power Management Hardwired to '0' if Clock Power Management is disabled in the Link Capabilities register.	RW	0x0
7	Extended Synch	RW	0x0
6	Common Clock Configuration	RW	0x0
5	Retrain Link	RW	0x0
4	Link Disable	RW	0x0
3	Read Completion Boundary (RCB)	RO	0
2	Reserved	RsvdP	0x0
1:0	Active State Link PM Control	RW	0x0

19.10.8.8 Link Status Register

Address Offset: `CFG_PCIE_CAP + 0x12

Bit#	Description	Attribute	Reset
15:14	Reserved	RsvdZ	0x0
13	Data Link Layer Active	RO	0x0
12	Slot Clock Configuration Indicates that the component uses the same physical reference clock that the platform provides on the connector.	RW	1
11	Link Training	RO	0
10	Undefined for PCI Express 1.1 (Was Training Error for PCI Express 1.0a)	RO	0x0
9:4	Negotiated Link Width Set automatically by hardware after Link initialization.	RO	0x00
3:0	Link Speed The negotiated Link speed: 2.5 Gbps	RO	0x1

19.10.8.9 Slot Capabilities Register

Address Offset: `CFG_PCIE_CAP + 0x14

Bit#	Description	Attribute	Reset
31:19	Physical Slot Number	RW	0
18	No Command Complete Support	RW	0x0
17	Electromechanical Interlock Present	RW	0x0
16:15	Slot Power Limit Scale	RW	0
14:7	Slot Power Limit Value	RW	0
6	Hot-Plug Capable	RW	0
5	Hot-Plug Surprise	RW	0
4	Power Indicator Present	RW	0
3	Attention Indicator Present	RW	0
2	MRL Sensor Present	RW	0
1	Power Controller Present	RW	0
0	Attention Button Present	RW	0

19.10.8.10 Slot Control Register

Address Offset: `CFG_PCIE_CAP + 0x18

Bit#	Description	Attribute	Reset
15:13	Reserved	RsvdP	0x0
12	Data Link Layer State Changed Enable	RW	0x0
11	Electromechanical Interlock Control	RW	0x0
10	Power Controller Control	RW	0x0
9:8	Power Indicator Control	RW	0x3
7:6	Attention Indicator Control	RW	0x3
5	Hot-Plug Interrupt Enable	RW	0x0
4	Command Completed Interrupt Enable	RW	0x0
3	Presence Detect Changed Enable	RW	0x0
2	MRL Sensor Changed Enable	RW	0x0
1	Power Fault Detected Enable	RW	0x0
0	Attention Button Pressed Enable	RW	0x0

19.10.8.11 Slot Status Register

Address Offset: `CFG_PCIE_CAP + 0x1A

Bit#	Description	Attribute	Reset
15:9	Reserved	RsvdZ	0x0
8	Data Link Layer State Changed	RW1C	0x0
7	Electromechanical Interlock Status	RO	0x0
6	Presence Detect State	RO	0x0
5	MRL Sensor State	RO	0x0
4	Command Completed	RWC1	0x0
3	Presence Detect Changed	RWC1	0x0
2	MRL Sensor Changed	RWC1	0x0
1	Power Fault Detected	RWC1	0x0
0	Attention Button Pressed	RWC1	0x0

19.10.8.12 Root Control Register

Address Offset: `CFG_PCIE_CAP + 0x1C

Bit#	Description	Attribute	Reset
15:5	Reserved	RsvdP	0x0
4	CRS Software Visibility Enable Not supported, hardwired to 0x0.	RO	0x0
3	PME Interrupt Enable	RW	0x0
2	System Error on Fatal Error Enable	RW	0x0
1	System Error on Non-fatal Error Enable	RW	0x0
0	System Error on Correctable Error Enable	RW	0x0

19.10.8.13 Root Capabilities Register

Address Offset: `CFG_PCIE_CAP + 0x1E

Bit#	Description	Attribute	Reset
0	CRS Software Visibility Not supported, hardwired to 0x0.	RO	0x0

19.10.8.14 Root Status Register

Address Offset: `CFG_PCIE_CAP + 0x20

Bit#	Description	Attribute	Reset
31:18	Reserved	RsvdZ	0x0
17	PME Pending	RO	0x0
16	PME Status	RW1C	0x0
15:0	PME Requester ID	RO	0x0

19.10.9 PCI Express Extended Capabilities Register Details

The PCIe core implements the following PCI Express Extended Capabilities registers:

- Advanced Error Reporting Capability register set
- Device Serial Number Capability register set

The following sections describe the individual registers in each group. For register maps, see [Section 19.9.2 Register Maps](#).

19.10.9.1 Advanced Error Reporting Capability Registers***PCI Express Enhanced Capability Header***

Address Offset: 0x100

Bit#	Description	Attribute	Reset
31:20	Next Capability Offset	RO	0x140
19:16	Capability Version	RO	0x1
15:0	PCI Express Extended Capability ID	RO	0x1

Uncorrectable Error Status Register

Address Offset: 0x100 + 0x04

Bit#	Description	Attribute	Reset
31:21	Reserved	RsvdZ	0x000
20	Unsupported Request Error Status	RW1CS	0
19	ECRC Error Status	RW1CS	0
18	Malformed TLP Status	RW1CS	0
17	Receiver Overflow Status	RW1CS	0
16	Unexpected Completion Status	RW1CS	0
15	Completer Abort Status	RW1CS	0
14	Completion Timeout Status	RW1CS	0
13	Flow Control Protocol Error Status	RW1CS	0
12	Poisoned TLP Status	RW1CS	0
11:6	Reserved	RsvdZ	0x0
5	Surprise Down Error Status (not supported)	RO	0
4	Data Link Protocol Error Status	RW1CS	0
3:1	Reserved	RsvdZ	0x0
0	Undefined for PCI Express 1.1 (Was Training Error Status for PCI Express 1.0a)	Undefined	0

Uncorrectable Error Mask Register

Address Offset: 0x100 + 0x08

Bit#	Description	Attribute	Reset
31:21	Reserved	RsvdP	0x000
20	Unsupported Request Error Mask	RWS	0
19	ECRC Error Mask	RWS	0
18	Malformed TLP Mask	RWS	0
17	Receiver Overflow Mask	RWS	0
16	Unexpected Completion Mask	RWS	0
15	Completer Abort Mask	RWS	0
14	Completion Timeout Mask	RWS	0
13	Flow Control Protocol Error Mask	RWS	0
12	Poisoned TLP Mask	RWS	0
11:6	Reserved	RsvdP	0x00
5	Surprise Down Error Mask (not supported)	RO	0x0
4	Data Link Protocol Error Mask	RWS	0
3:1	Reserved	RsvdP	0x0
0	Undefined for PCI Express 1.1 (Was Training Error Mask for PCI Express 1.0a)	Undefined	0

Uncorrectable Error Severity Register

Address Offset: 0x100 + 0x0C

Bit#	Description	Attribute	Reset
31:21	<i>Reserved</i>	RsvdP	0x000
20	Unsupported Request Error Severity	RWS	0
19	ECRC Error Severity	RWS	0
18	Malformed TLP Severity	RWS	0x1
17	Receiver Overflow Severity	RWS	0x1
16	Unexpected Completion Severity	RWS	0
15	Completer Abort Severity	RWS	0
14	Completion Timeout Severity	RWS	0
13	Flow Control Protocol Error Severity	RWS	0x1
12	Poisoned TLP Severity	RWS	0
11:6	<i>Reserved</i>	RsvdP	0x00
5	Surprise Down Error Severity (not supported)	RO	0x1
4	Data Link Protocol Error Severity	RWS	0x1
3:1	<i>Reserved</i>	RsvdP	0x0
0	Undefined for PCI Express 1.1 (Was Training Error Severity for PCI Express 1.0a)	Undefined	0x1

Correctable Error Status Register

Address Offset: 0x100 + 0x10

Bit#	Description	Attribute	Reset
31:14	<i>Reserved</i>	RsvdZ	0x00000
13	Advisory Non-Fatal Error Status	RW1CS	0
12	Reply Timer Timeout Status	RW1CS	0
11:9	<i>Reserved</i>	RsvdZ	0x0
8	REPLAY_NUM Rollover Status	RW1CS	0
7	Bad DLLP Status	RW1CS	0
6	Bad TLP Status	RW1CS	0
5:1	<i>Reserved</i>	RsvdZ	0x00
0	Receiver Error Status	RW1CS	0

Correctable Error Mask Register

Address Offset: 0x100 + 0x14

Bit#	Description	Attribute	Reset
31:14	Reserved	RsvdP	0x00000
13	Advisory Non-Fatal Error Mask	RWS	0x1
12	Reply Timer Timeout Mask	RWS	0
11:9	Reserved	RsvdP	0x0
8	REPLAY_NUM Rollover Mask	RWS	0
7	Bad DLLP Mask	RWS	0
6	Bad TLP Mask	RWS	0
5:1	Reserved	RsvdP	0x00
0	Receiver Error Mask	RWS	0

Advanced Capabilities and Control Register

Address Offset: 0x100 + 0x18

Bit#	Description	Attribute	Reset
31:9	Reserved	RsvdP	0x000000
8	ECRC Check Enable	RWS	0
7	ECRC Check Capable	RO	1
6	ECRC Generation Enable	RWS	0
5	ECRC Generation Capability	RO	1
4:0	First Error Pointer	ROS	0x00

19.10.9.2 Header Log Registers**Address Offset:** **0x100 + 0x1C**

The Header Log registers collect the header for the TLP corresponding to a detected error. See the PCI Express Base Specification for details. Each of the Header Log registers is type ROS; the default reset value of each Header Log register is 0x00000000.

Byte Offset	Byte 3	Byte 2	Byte 1	Byte 0
0x1C	Header Log Register (first DWord)			
0x20	Header Log Register (second DWord)			
0x24	Header Log Register (third DWord)			
0x28	Header Log Register (fourth DWord)			

Root Error Command Register**Address Offset:** **0x100 + 0x2C**

Bit#	Description	Attribute	Reset
31:3	<i>Reserved</i>	RsvdP	0x00000000
2	Fatal Error Reporting Enable	RW	0x0
1	Non-Fatal Error Reporting Enable	RW	0x0
0	Correctable Error Reporting Enable	RW	0x0

Root Error Status Register**Address Offset:** **0x100 + 0x30**

Bit#	Description	Attribute	Reset
31:27	Advanced Error Interrupt Message Number	RO	0x00
26:7	<i>Reserved</i>	RsvdZ	0x00000
6	Fatal Error Messages Received	RW1CS	0
5	Non-Fatal Error Messages Received	RW1CS	0
4	First Uncorrectable Fatal	RW1CS	0
3	Multiple ERR_FATAL/NONFATAL Received	RW1CS	0
2	ERR_FATAL/NONFATAL Received	RW1CS	0
1	Multiple ERR_COR Received	RW1CS	0
0	ERR_COR Received	RW1CS	0

Error Source Identification Register

Address Offset: 0x100 + 0x34

Bit#	Description	Attribute	Reset
31:16	ERR_FATAL/NONFATAL Source Identification	ROS	0x0000
15:0	ERR_COR Source Identification	ROS	0x0000

19.10.9.3 Device Serial Number Capability Registers**Device Serial Number Enhanced Capability Header**

Address Offset: 0x140

Bit#	Description	Attribute	Reset
31:20	Points to end of capabilities by default	Hwinit	0x000
19:16	Capability Version	Hwinit	0x1
15:0	Extended Capability ID	Hwinit	0x0003

Serial Number Registers

Address Offset: `SN_PTR + 0x4/+0x8

Offset	Description	Attribute	Reset
0x4	Serial Number register (1st DWord)	RO	0
0x8	Serial Number register (2nd DWord)	RO	0



Chapter 20 Serial-RapidIO Interface

20.1 Introduction¹

This chapter covers the optional Serial-RapidIO (SRIO) Interface implementation. It is only available in the XLS616, XLS608, XLS416, XLS408 and XLS404 devices which support an optional 1x4 or 4x1 SRIO Interface, which shares the pins/pads with the PCIe interface.

20.2 Overview of Serial-RapidIO (SRIO) Interface Capabilities

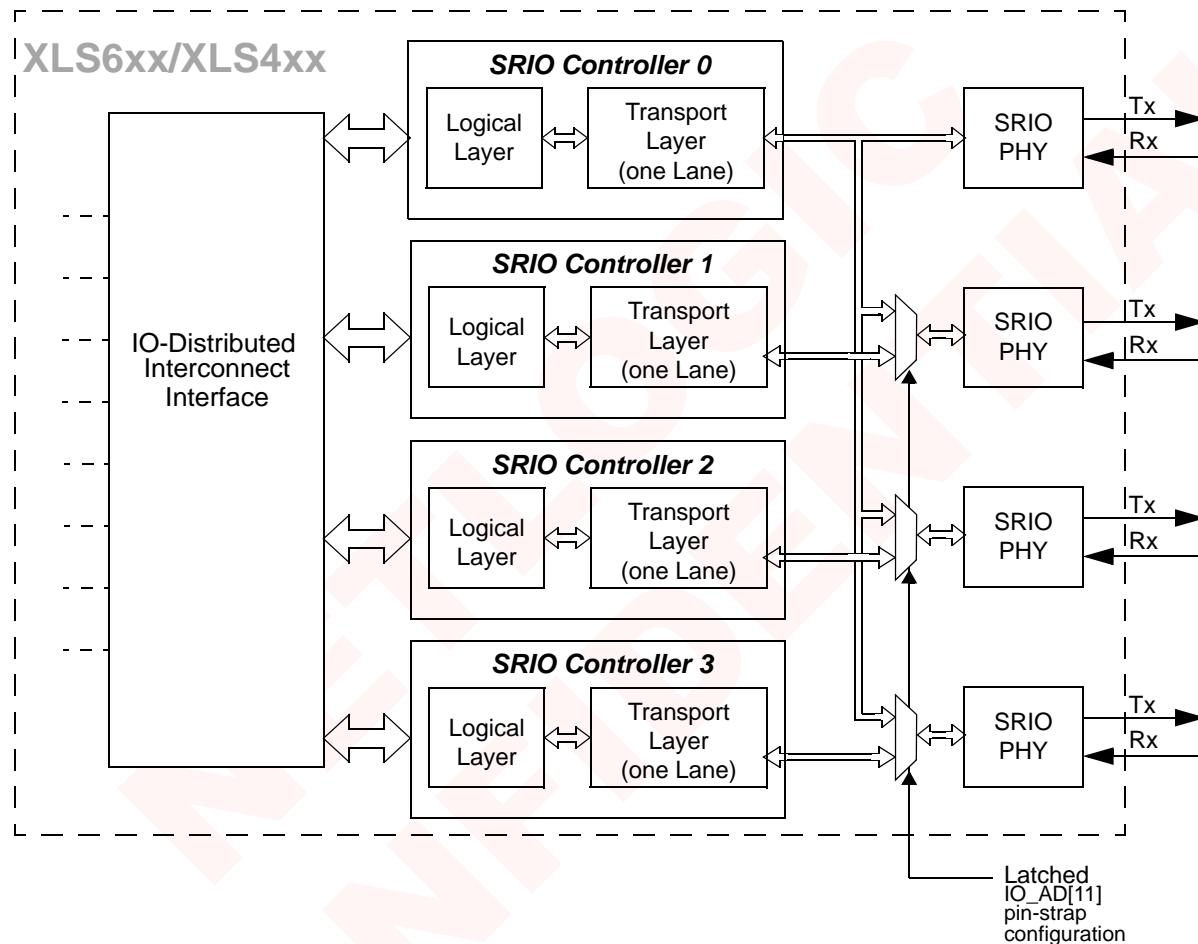
The XLS6xx and XLS4xx Serial-RapidIO (SRIO) Interface is configured as one-controller with four I/O lanes (1x4) - (see [Figure 20-1](#)), or four Controllers with one lane each (4x1). The XLS SRIO interface conforms to the SRIO Interconnect Standard, Revision 1.3. Some of the supported features of the RapidIO standard are:

- Serial RapidIO interface capable of supporting full duplex, data rates up to 10-Gbps in each direction
- Operates at 1.25, 2.5 or 3.125-Gbaud (1.0, 2.0 or 2.5 Gbps) per lane
- Supports Small (8-bit) and Large (16-bit) device ID's
- Provides an efficient RapidIO interface for high-bandwidth, variable-sized packet applications
- Contains enhanced DMA engines to transfer data using queue memory structures to reduce overhead on host processor
- Supports Input/Output, Message, Doorbell, Maintenance and Port-write Logical Layer functionality
- Message support for up to 64 Mailboxes, with 4 Letters per Mailbox
- Four freelists supported
- Status Queues and Free-lists are not shared
- Multi-segment and Single-segment Message support
- Multiple concurrent transactions Automatic retry on up to 32 transactions
- Internal Doorbell and Port-write FIFOs
- Transaction types, such as DMA-style read and writes, allow efficient I/O systems
- RapidIO supports a variety of data sizes within the packet formats to improve the interconnect efficiency by allowing larger non-coherent data quantities to be encapsulated within a single packet.
- System sizes from very small to very large are supported in the same or compatible packet formats
- Read-modify-write atomic operations are useful for synchronization between processors or other system elements.
- The RapidIO architecture presently supports 34-bit local addresses.

1. The functional and register descriptions in this chapter are Copyright 2008, Jennic Ltd. and are reproduced herein by NetLogic under express license rights from Jennic Ltd. All rights reserved.

- RapidIO packet definition is independent of the width of the physical interface to other devices on the interconnect fabric.
- The protocols and packet formats are independent of the physical interconnect topology. The protocols work whether the physical fabric is a point-to-point ring, a bus, a switched multi-dimensional network, a duplex serial connection, and so forth.
- RapidIO is not dependent on the bandwidth or latency of the physical fabric.
- The protocols handle out-of-order packet transmission and reception.

Figure 20-1. Internal XLS6xx and XLS4xx SRIO 4x1 and 1x4 Topology



20.2.1 SRIO Selection and Configuration

All SRIO interfaces share pins with the PCI Express interface. Either PCIe or SRIO is available, selected via pin-straps at power-on reset time. Selection is accomplished via the IO_AD[5:4] external signals, as shown in Table 20-1.

Table 20-1. Pinstrap Selection of PCIe or SRIO for the XLS6xx and XLS4xx

IO_AD5	IO_AD4	Selection of PCIe or SRIO	
0	0	PCI-Express is selected (SRIO is unavailable)	
0	1	1.25 Gbaud SRIO	SRIO is selected and PCIe is unavailable. The SRIO operating frequency is also selected.
1	0	2.50 Gbaud SRIO	SRIO port configuration is selected by IO_AD[11:10] as shown in Table 20-2
1	1	3.125 Gbaud SRIO	

Bits [27:26] of the GPIO Power On Reset Configuration Register provides a read-only mirror of the power-on reset logic levels of pins IO_AD[5:4].

The SRIO Lane configuration is determined via pin-strap at power-on reset time. Pins IO_AD10 and IO_AD11 are strapped as shown in [Table 20-2](#) to select the desired lane configuration, either 4x1 or 1x4.

Table 20-2. SRIO Pinstrap Lane Configuration for the XLS6xx and XLS4xx Devices

IO_AD11	IO_AD10	SRIO Lane Configuration
0	0	When SRIO selected, it is configured as one Controller with four Lanes
0	1	When SRIO selected, it is configured as one Controller with four Lanes
1	0	When SRIO selected, it is configured as four Controllers with one Lane each
1	1	When SRIO selected, it is configured as four Controllers with one Lane each

Bits [21:20] of the GPIO Power On Reset Configuration Register provides a read-only mirror of the power-on reset logic levels of pins IO_AD[11:10].

20.3

SRIo Theory of Operation

The SRIO interface connects to the core through a station on the IO DI ring. I/O DI operation is described in [Chapter 8, “Memory and I/O Access”](#). Accesses are done by direct access.

On a direct access, the CPU reads or writes to a defined memory address space range. The internal Memory Bridge detects the read or write request, forwarding the transaction to the SRIO controller, which generates the appropriate SRIO signaling. Each SRIO operation, (e.g. memory read/write, I/O read/write and configuration read/write), has corresponding memory address space and range as defined by the SRIO control BAR (Basic Address Register) section in Chapter 8. Effectively, this set of registers are mapping the CPU memory address space to SRIO operations, that is – accessing this space generates a corresponding SRIO cycle:

- SRIO_IO_BAR – Defines I/O transaction space
- SRIO_MEM_BAR – Defines Memory transaction space

20.3.1

Accessing SRIO Interface Configuration Registers

Software can access the SRIO Interface registers through the XLS Peripheral & I/O Configuration region of memory (see [Section 8.4, Table 8-1](#)).

The XLS_IO_BAR register (Register #25) in the System Bridge Controller (SBC) provides the base address for accessing the Peripheral & I/O Configuration Region of memory. It is a software-programmable register. Out of reset, the XLS_IO_BAR is enabled and initialized to a default base address (see [Section 8.7.1](#)).

The SRIO controller has two configuration register regions set aside for it – one for big-endian memory address ordering and the other for little-endian ordering.

- XLS_IO_DEV_SRIO_BE at region offset 0x1E000 from XLS_IO_BAR base address

- XLS_IO_DEV_SRIO_LE at region offset 0x1F000 from XLS_IO_BAR base address

Both regions access the same registers. The only difference is in the interpretation of the addressing mode (i.e. big-endian vs little-endian).

Within a register region, there are four KBytes of space to provide access for up to 1024 registers. See [Serial RapidIO \(SRIO\) Interface Registers](#) for the list and descriptions of the SRIO I/O Interface Configuration registers available.

[Section 8.5 on page 228](#) illustrates how the three different address elements are put together to create the physical address to the XLS Peripheral & I/O Configuration Region of memory which is sent to the System Bridge Controller (SBC). The fields are shown in [Table 20-3](#):

Table 20-3. Physical Address Fields for Interface Registers

Physical Address Field (A[X:Y])	Description
Address bits [39:20]	I/O Base Address
Address bits [19:12]	Region Offset: Bit [12] accomplishes the SWAP between big-endian and little-endian access format: When bit [12] = '0': Access is big-endian When bit [12] = '1': Access is little-endian
Address bits [11:0]	Register Address Offset

20.3.2

Accessing SRIO Configuration and Control Registers

The SRIO configuration and control registers are memory mapped into the PCIe space. Either the PCIe or the SRIO controllers have access to this register space as selected by pin-strapping options at power-up reset time. See [Section 8.7 on page 241](#) for a description of the memory organization. The memory address decoding for these registers is shown in [Table 20-4](#)

Since the CPU is a 64-bit machine, the software-generated address for the XLS Peripheral & I/O Configuration Region is a 64-bit address that is translated by the MMU into a 40-bit physical address sent to the System Bridge Controller (SBC). When the SBC gets a memory transaction that matches the memory region for the PCIe/SRIO configuration registers, it forwards that request to the SRIO controller, which performs the required SBC register access.

Table 20-4. Physical Address Fields for Configuration and Control Registers

Physical Address Field (A[X:Y])	Description
Address bits [39:27]	I/O Base Address
Address bits [26]	'0' = Access is big-endian '1' = Access is little-endian
Address bits [25:24]	Link number
Address bits [23:0]	Register Address Offset

20.3.3

Resetting the SRIO Controllers

When a SRIO reset device command is received on the link, it is reported by setting RST bit in the [GEN_STAT](#) register. If the [GEN_IER](#) mask is set, an interrupt is generated to the host. The host can then reset the SRIO controller, either by toggling H_RST_N bit in [SRIO_CTRL](#) or triggering a software reset - SRST bit in the [GEN_CTRL](#) register.

If RSTEN bit in the [GEN_CTRL](#) register is set, a reset-device command on the link will automatically reset all blocks in the controller.

If the reset is applied to the controller using any of these three methods, software must wait 10 ms before starting to reconfigure the controller.

20.4 Programming Model

20.4.1 Startup/Initialization

The XLS616, XLS608, XLS416, XLS408 and XLS404 SRIO interface shares pins with the PCI Express interface. Either PCIe or SRIO is available, selected via pin-strapping at power-on reset time. Selection is accomplished as described in [Section 20.2.1 on page 984](#).

20.4.2 Functionality Overview

The SRIO interface supports the following transactions.

Table 20-5. XLS6xx and XLS4xx Supported SRIO Transactions

Packet Format Type	Transaction Type	Source Processing Generated	Destination Processing Action on Receipt
Type 2	NREAD	Yes. In response to a queued transfer request from host.	DMA Engine issues host bus read from specified address. Resulting data then attached to generated RESPONSE packet.
	ATOMIC	No.	Ignored.
Type 5	NWRITE	Yes. In response to a queued transfer request from host.	DMA Engine writes supplied data to specified address over host bus.
	NWRITE_R	Yes. In response to a queued transfer request from host.	DMA Engine writes supplied data to specified address over host bus. RESPONSE packet generated when complete.
	ATOMIC	No.	Ignored.
Type 6	SWRITE	Yes. In response to a queued transfer request from host.	DMA Engine writes supplied data to specified address over host bus.
Type 8	MAINTENANCE	Yes. In response to a queued transfer request from host, and when a received maintenance request requires a response.	Specified register is written or read as requested and a MAINTENANCE response packet generated.
	PORT-WRITE	Yes. In response to a queued transfer request from host.	Supplied information written to internal host-accessible registers.
Type 10	DOORBELL	Yes. In response to a queued transfer request from host.	Supplied information written to internal host-accessible FIFO and RESPONSE packet generated.
Type 11	MESSAGE	Yes. In response to a queued transfer request from host.	Message segments reassembled in locally assigned buffer in host memory (via DMA engine), and released to host when complete.
Type 13	RESPONSE	Yes. Automatically following receipt of NREAD, NWRITE_R, DOORBELL and MESSAGE transactions.	Matched up with generated request packets where a RESPONSE is expected
Others		No.	Ignored.

Based around a series of shared memory queue structures, this SRIO controller takes some of the processing burden away from the host processor. RapidIO transfer requests are handed off by the host through these data structures, which are accessed via DMA and processed. The completion of requests is reported back to the host. RapidIO incoming requests are similarly handled by DMA operations.

20.4.3 Source Processing

The host controls the generation of all RapidIO requests produced by the device. It does this by placing entries on a Transaction Queue. The entries have a common structure allowing different transfer request types to be mixed on a queue. Each entry specifies the type of transaction to perform along with all the information required for the transaction (e.g. destination ID, transfer size, a pointer into local memory where the payload is located for write, or where the requested data is written for reads etc.). In store and forward applications, having the transfer data linked by pointers from the Transaction Queue entry can prevent the host from having to move large blocks of data from buffer to buffer, saving valuable bus-cycles. The host can queue up to two transactions and then return to other processing leaving the SRIO controller to complete the operations autonomously.

Once a transaction is complete, an entry is normally written onto a Status Queue (this write can be suppressed if desired). The Status Queue entry provides the host with a record and status of the transactions that have been completed. Also returned are pointers to any payload/message buffers so that they may be retrieved (e.g., data returned following a NREAD request or a spent buffer from a NWRITE or MESSAGE etc.). If any timeout, error response or local DMA problems are encountered, these are also logged in the Status Queue entry.

Interrupts to the host may be generated on specific transactions completing or certain queues events.

20.4.3.1 Transaction Queue Processing

Two Transaction Queues are supported (numbered TQ0, TQ1). The user can also choose between the arbitration schemes of how the two Transaction Queues are handled: equal priority or single high priority. These features allow flexibility in how the queues are used, e.g., two hosts can have a Transaction Queue each, avoiding contention when entries are added, or higher-priority requests can be assigned to the high-priority queue. Note that these priorities represent queue priority, not necessarily the priority used within the RapidIO packet (which is assigned by a field within the queue entry).

With the high-priority scheme, TQ0 is always treated as the highest priority queue. Anything added to this queue will be processed at the next available opportunity. Whenever there is nothing in the high-priority TQ0, the packets in the other queue are processed.

Note that completion of a request occurs when all the required RapidIO request packets have been generated; the SRIO controller does not wait for any RapidIO response packets to return before continuing.

20.4.3.2 Overview of Transaction Entry Processing

Table 20-6 shows the limits on the size of transfer (PDU) that can be requested for each transaction type. Since the maximum size operation possible using a single RapidIO request is 256 bytes, the SRIO controller automatically manages the segmentation of larger PDUs into multiple parts as required (up to a maximum of 16). Segmentation may also occur if the offset address is not on a double-word (8-byte) boundary, depending upon the requested transfer size.

Table 20-6. Supported PDU Sizes for Generated Requests

Transaction	Supported PDU Sizes	Offset Address Requirement at Destination	Response Tracking Context Required?
NREAD	1 byte-4K bytes	No restrictions	Yes
NWRITE	1 byte-4K bytes	No restrictions	No
NWRITE_R	1 byte-4K bytes	No restrictions	Yes

Table 20-6. Supported PDU Sizes for Generated Requests (continued)

Transaction	Supported PDU Sizes	Offset Address Requirement at Destination	Response Tracking Context Required?
SWRITE	1 byte-4K bytes	No restrictions	No
MAINTENANCE read	4, 8, 16, 32 and 64 bytes only	4-byte transactions must be word aligned, others must be double-word aligned	Yes
MAINTENANCE Port-write	4, 8, 16, 24, 32, 40, 48, 56 and 64 bytes only	N/A	No
MESSAGE	8 bytes-256 (if MSM option set 4K bytes ^a) in increments of 8 bytes	N/A	Yes
DOORBELL	N/A (no payload, but 2 bytes of information in packet header)	N/A	Yes

- a. Maximum PDU size for MESSAGE limited to 256 bytes if only single segment messages used. Multi-segment Messages are available on Mailboxes 0-3.

Any generated RapidIO request that results in a response needs tracking so that the returning response can be related back to the appropriate request. In addition, any transaction that is segmented into many requests needs tracking so that it is known when all the expected responses have been returned and the Status Queue entry can be generated. The SRIO controller handles these tasks automatically by controlling the generation of Transaction Identities (TIDs) for non-message requests, and Letter number (0-3) for message requests (which form a field in the generated RapidIO requests), and the use of transaction contexts and checkerboards (which allow expected returning Responses to be checked off). In this way, multiple in-flight transactions can be handled (up to 16 simultaneous MESSAGE RapidIO requests, and up to 16 other response-generating requests). Note that in order to achieve a large number of simultaneous outstanding MESSAGE requests, the user must ensure that no more than 4 transmit requests for each Mailbox are in progress at any time (otherwise lack of available Letters for the Mailbox will cause temporary blocking of that Transaction Queue).

Before accepting a Transaction Queue request that needs a tracking context, the availability of a context and checkerboard (if needed) is checked. If available, the transaction can proceed. If all are in use, that Transaction Queue request is deferred and a re-arbitration of the other Transaction Queues is performed to find if a transaction on another queue can proceed. If no further progress can be made on any Transaction Queue, or there are no further requests on any queue, the SRIO controller waits until a context becomes available or a new entry is added to one of the empty Transaction Queues.

Since the Buffer Address may not match the alignment required for the RapidIO request being generated, some byte shifting of data may be performed when transferring the payload data.

20.4.4 Destination Processing

20.4.4.1 I/O Transactions

As a RapidIO destination, incoming Input/Output requests are converted into host DMA accesses. On completion of NREAD and NWRITE_R transactions, a RESPONSE packet is automatically generated and interleaved with any source traffic for transmission.

20.4.4.2 Maintenance and Port-Write Transactions

Received Maintenance Transactions are dealt with internally in the core, with no visibility on the host interface.

Port-writes received are processed and held in the Port-write FIFO, which is capable of storing a single 64-byte Port-write or several smaller transfers. Access to the FIFO by the host interface is via registers.

20.4.4.3

MESSAGES

Received MESSAGE transactions are reassembled in shared memory before being presented to the host.

Four mailboxes are supported. Each Mailbox has the ability to receive simultaneously up to 4 Letters. All mailboxes have the ability to receive multi-segment MESSAGES (each up to 4096 bytes).

Buffers must be provided by the host in shared memory in which to write received messages. The addresses of these are located on Free-Lists maintained by the host. All buffers on each Free-List must be of the same size.

The four Free-Lists are available as one Free-List per Mailbox.

Free-List buffers are allocated from the Free-List associated with the Mailbox on which the MESSAGE is received. The size of buffers on Free-List 0 to Free-List 3 must be defined as being large enough to support the largest single or multi-segment MESSAGE expected.

Segments for multi-segment MESSAGES may be received in any order. The address within the allocated buffer to which data is written is determined by the segment size and segment number from the packet header. As segments are received, a note is made against the number of segments expected. Once all segments have been received, the MESSAGE is released to the host through an entry on the appropriate Mailbox Queue.

Multiple Mailbox Queues are numbers MQ0, MQ1, MQ2 and MQ3. Messages received on Mailbox 0 are notified on MQ0, Mailbox 1 on MQ1 etc. The last Mailbox Queue MQ3 is shared for the notification of all messages received on supported Mailboxes.

Each MESSAGE segment received will automatically trigger an appropriate RESPONSE to be generated back to the source.

20.4.4.4

DOORBELLS

DOORBELL messages are terminated by the SRIO controller and presented to the host through an internal Doorbell FIFO. The DOORBELL user bytes and source ID are accessible to the host through registers via the host bus.

Each DOORBELL received will automatically trigger an appropriate RESPONSE to be generated back to the source.

20.4.5

Data Structures

This section describes in detail how source and destination processing is controlled by the host processor. In particular, it describes the format of the various data structures which the host needs to form in its local memory in order to initiate transactions. Note that they are shown in Little Endian format, and each includes a 31:0 bit label which correspond to the bit labels in memory, and hence the expected position on the bus when transactions are read by the SRIO controller.

20.4.5.1

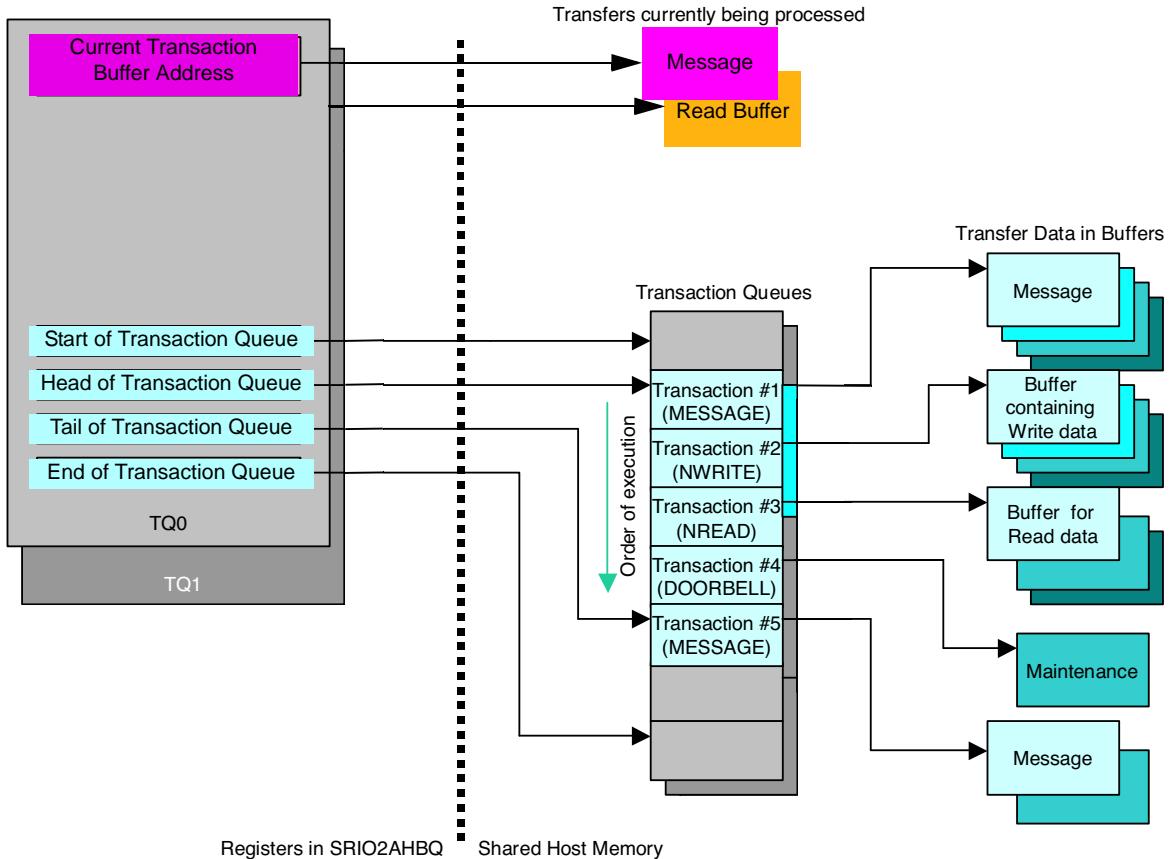
Source Data Structures

Transaction Queues

The Transaction Queues provide the hand-off of RapidIO transaction requests between the host and the SRIO controller. Each Transaction Queue is a circular structure which may be

sited anywhere in shared host memory. Each is managed independently through four internal pointers; the Start and End pointers define the bounds of the circular queue, and the Head and Tail pointers indicate the locations where entries are removed (by the SRIO controller) and added (by the host) respectively. For full details of how the host configures the queue and subsequently adds entries, see below.

Figure 20-2. Transaction Queues

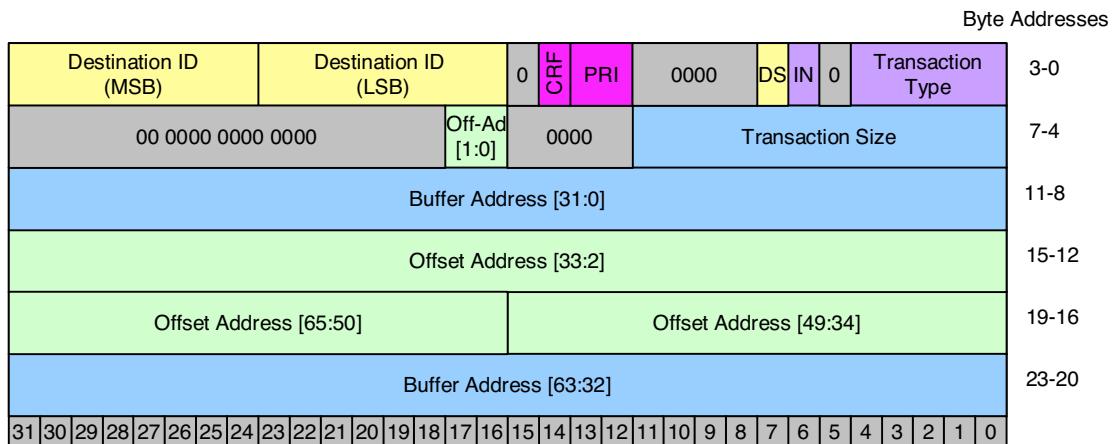


Transaction Queues Updates

The size of all Transaction Queue entries is fixed at 6-words. This provides enough space to define all the parameters for each transaction, including the ability to accommodate extended addressing modes and 64-bit buffer pointers. However, there are several different formats of entry depending upon the transaction type. [Table 20-7](#) shows all the transaction types supported together with their transaction code and queue entry type. The transaction code is in the first word of each Transaction Queue entry, and hence defines the format of the remaining 5-words. The Input/Output Transaction Queue Entry Format is shown in [Figure 20-3](#).

Table 20-7. Supported Transaction Requests

Transaction Type	Transaction Type Coding (in binary)	Queue Entry Type
NREAD	00001	Input/Output
NWRITE	00010	Input/Output
NWRITE_R	00100	Input/Output
SWRITE	00110	Input/Output
MAINTENANCE Read	01101	Maintenance
MAINTENANCE Write	01110	Maintenance
MAINTENANCE Port-write	01111	Port-write
MESSAGE	10000	Message
DOORBELL	10001	Doorbell
Reserved	all other codes	Reserved

Figure 20-3. Input/Output Transaction Queue Entry (Little Endian)

- **Transaction Type** – Indicates the Input/Output transaction type required.
- **DS** – Destination ID Size – indicates if the Source/Destination ID used in the RapidIO packet header should be 8 or 16-bit.

DS	Size
0	8 bit
1	16 bit

- **Destination ID** – This value indicates where the transaction is destined. If DS = 0, indicating 8-bit device addressing, only the LSB is valid. With DS = '1' both the MSB and LSB make up the 16-bit Destination ID.
- **CRF** – Critical Request Flow bit for the packet header. It is used to differentiate between flows of the same priority. If not supported in the system this is set to '0'.

- **PRI** – Transaction Priority – defines the PRI value inserted into the RapidIO packet header and the transmit queue it is scheduled to the Serial Physical Interface.

PRI	Priority	Flow (Non CRF mode)	CRF	Flow (CRF mode)
00	lowest	A	0	A
			1	B
01	next	B	0	C
			1	D
10	next		0	E
		C or Higher	1	F or Higher
11	highest	(Reserved for RESPONSE)	-	(Reserved for RESPONSE)

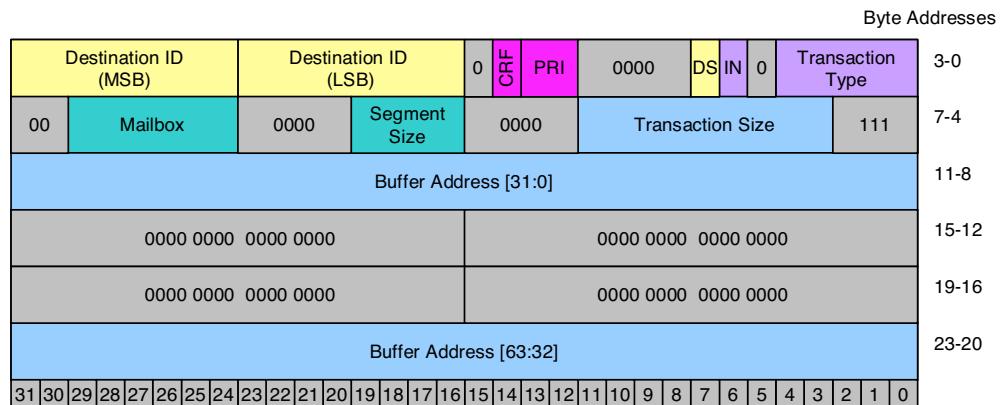
- **Transaction Size** – Indicates the number of bytes to be transferred.

Transaction Size Code	Number of Bytes
0	1
...	...
4095	4096

- **IN** – Interrupt flag – If an entry is read with this flag set to ‘1’, an interrupt to the host is generated to indicate the entry has been reached, processing started and the queue entry removed (respective head pointer incremented).
- **Offset Address** – This sets the offset address at the destination for the transaction. This can be 34, 50 or 66 bits depending upon the setting of the PE_LL_CSR register. Accesses may be from any Offset Address, including non double-word aligned addresses. Note that the least-significant 2 bits of address are separate so that the offset address in word 3 is word-aligned. For 32-bit address host systems, bits [65:34] need not be set (they are ignored by the SRIO controller).

Configured Offset Addressing Size	RapidIO-Style Address Bus Bits	Offset Address bits used from Transaction Entry
34	[0:33]	[33:0]
50	[0:49]	[49:0]
66	[0:65]	[65:0]

- **Buffer Address** – Pointer to the buffer, situated in host memory, allocated for the transaction. For requests which require a payload, the buffer holds the data to be sent. For requests which fetch data, the buffer indicates where the retrieved data should be placed once the response is received. Read buffers are assumed to be large enough to hold the data being requested. Buffers may be situated anywhere in shared memory and there is no restriction to align the buffers to any boundary. It should be noted, however, that having buffers aligned to a double-word boundary may offer some transfer efficiencies across the RapidIO fabric. For 32-bit address host systems, bits [63:32] need not be set (they are ignored by the SRIO controller)

Figure 20-4. MESSAGE Transaction Queue Entry (Little Endian)

- **Transaction Type** – Indicates a Message transaction.
- **Transaction Size** – Indicates the number of double-words to be transferred. MESSAGE transactions can only transfer whole double-words so the Transaction-Size is defined in whole double-words. To achieve this, the host must write ‘1’s to bits [2:0] of byte 4 (these bits form part of the transaction size field for other transaction types). The size must be 256 bytes or less if mailbox 4-63 is selected.

Transaction Size Code	Number of Double-words	Number of Bytes
0	1	8
...
511	512	4096

- **Segment Size** – Defines the standard message packet payload size (SSIZE) and thus the size of each segment generated for the PDU, with the exception of the last segment. The user must ensure that the transaction size/segment size combination chosen are such that the number of segments inferred is less than or equal to 16.

Segment Size Code	Size of each generated	Segment
(in binary)	except the last	
0000-1000	Reserved	
1001	8 bytes	
1010	16 bytes	
1011	32 bytes	
1100	64 bytes	
1101	128 bytes	
1110	256 bytes	
1111	Reserved	

- **Mailbox** – This defines the Mailbox to be used for the MESSAGE transfer.

Mailbox	Use
0	Can be used to transfer multi-segment or single-segment MESSAGES
1	
2	
3	

All other fields as previously defined.

- DOORBELL Transaction Queue Entry Format

Figure 20-5. DOORBELL Transaction Queue Entry Format (LittleEndian)

Byte Addresses																															
Destination ID (MSB)		Destination Address (LSB)		0	CRE	PRI	0000	DS	IN	0	Transaction Type																				
Info Byte (MSB)		Info Byte (LSB)		0000 0000 0000 0000 0000 0000 0000 0000																											
0000 0000 0000 0000		0000 0000 0000 0000		0000 0000 0000 0000 0000 0000 0000 0000																											
0000 0000 0000 0000		0000 0000 0000 0000		0000 0000 0000 0000 0000 0000 0000 0000																											
0000 0000 0000 0000		0000 0000 0000 0000		0000 0000 0000 0000 0000 0000 0000 0000																											
0000 0000 0000 0000		0000 0000 0000 0000		0000 0000 0000 0000 0000 0000 0000 0000																											
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- **Transaction Type** – Indicates a Doorbell transaction.
- **Info-Byte1/Info-Byte2** – The two Information bytes that form the payload for a DOORBELL Message.

All other fields as previously defined.

Maintenance Transaction Queue Entry Format

Figure 20-6. MAINTENANCE Transaction Queue Entry (LittleEndian)

Byte Addresses																															
Destination ID (MSB)		Destination ID (LSB)		0	CRE	PRI	0000	DS	IN	0	Transaction Type																				
00 0000 0000 0000				00	0000		00 0000		Transaction Size	11																					
Buffer Address [31:0]																															
00	Hop Count		Register Offset Address [23:2]																												
0000 0000 0000 0000				0000 0000 0000 0000																											
Buffer Address [63:32]																															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

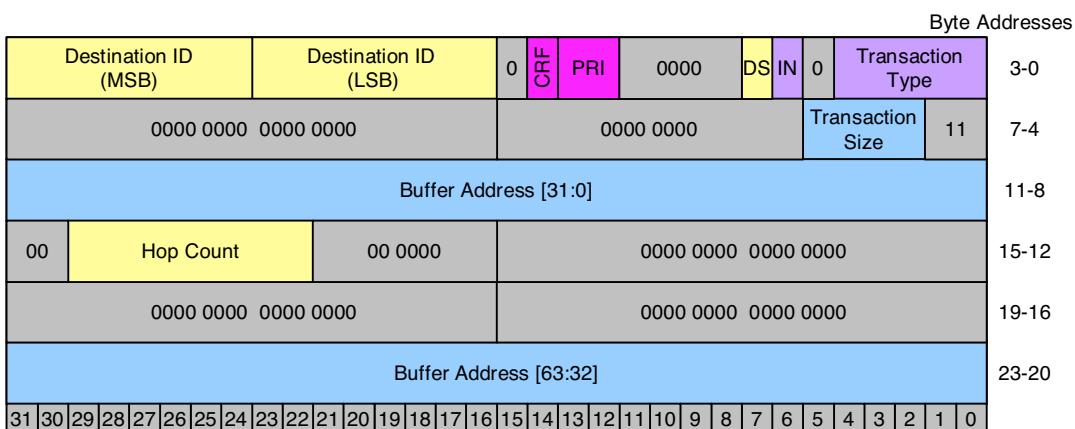
- **Transaction Type** – Indicates a Maintenance transaction (read or write).
- **Transaction Size** – Indicates the number of words to be transferred. MAINTENANCE transactions can only transfer whole words, so the Transaction-Size is defined in whole words. To achieve this, the host must write ‘1’s to bits [1:0] of byte 4 (these form part of the transaction size field for other transaction types). Only a limited number of transaction sizes are allowed, these being the ones which map into a single legal RapidIO MAINTENANCE packet, and there are limitations on the address for certain sizes. The [Table 20-8](#) lists all the legal transaction sizes and register offset address. All are valid for Maintenance Writes, but only some for Maintenance Reads. Any other size/address combinations result in a Status Queue entry being made with a Status Code of “Transfer Size Error” (no MAINTENANCE packet being generated).

Table 20-8. Legal Transaction Sizes and Register Offset Address

Transaction Size Code	Number of Words	Number of Bytes	Allowed value(s) of Register Offset Address bit [2]	Valid for Reads?
0	1	4	0 or 1	Yes
1	2	8	0	Yes
3	4	16	0	Yes
5	6	24	0	No
7	8	32	0	Yes
9	10	40	0	No
11	12	48	0	No
13	14	56	0	No
15	16	64	0	Yes

- **Hop Count** – The Hop Count value for the MAINTENANCE transaction.
- **Register Offset Address** – This sets the word offset address at the destination for the transaction. All other fields as previously defined.

Port-write Transaction Queue Entry Format

Figure 20-7. Port-Write Transaction Queue Entry (Little Endian)

- **Transaction Type** – Indicates Port-write Transaction.
- **Transaction Size** – Indicates the number of words to be transferred. Port-Write transactions can only transfer whole words so the Transaction-Size is defined in whole

words. To achieve this, the host must write '1's to bits [1:0] of byte 4 (these form part of the transaction size field for other transaction types). Only certain sizes of transaction are allowed as follows:

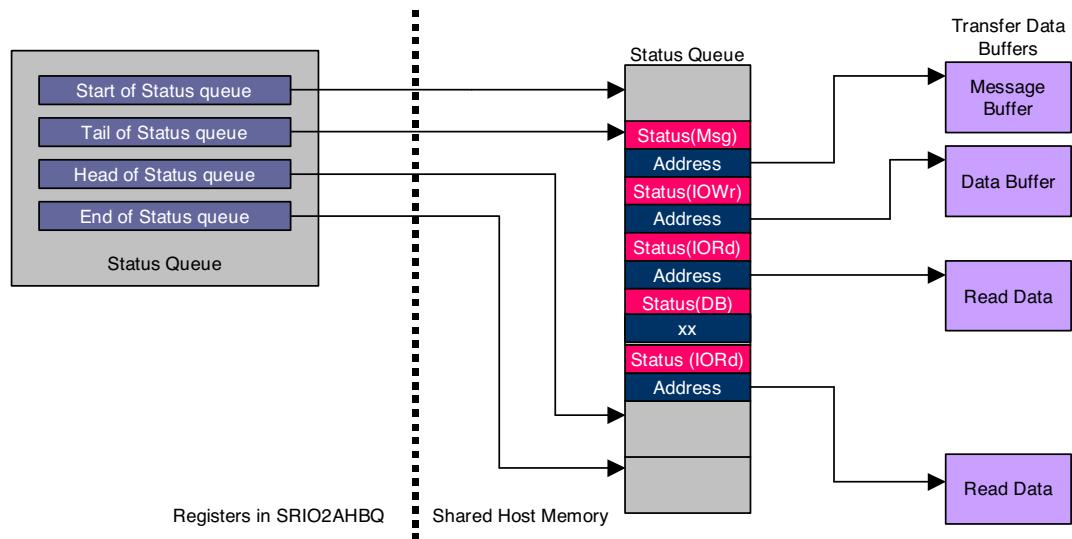
Transaction Size Code	Number of Words	Number of Bytes
0	1	4
1	2	8
3	4	16
5	6	24
7	8	32
9	10	40
11	12	48
13	14	56
15	16	64

All other fields as previously defined.

Status Queue

A Status Queue is circular and managed in a similar way to the Transaction Queue. The SRIO controller adds entries to the tail of the queue, whilst the host reads entries from the head of the Queue.

Figure 20-8. Status Queue



The Status Queue reports to the host progress of transfers from the Transaction Queues and provide the relevant information for the associated buffers to be retrieved.

- Each Transaction Queue can have its own corresponding Status Queue.
- Each entry on the Status Queue is 8-Words in length and is formatted as shown in [Figure 20-9](#), but the Bytes 16-31 are Reserved

Figure 20-9. Status Queue Entry (Little Endian)

Byte Addresses									
Destination ID (MSB)	Destination ID (LSB)	Status	Transaction Queue ID	DS	00	Transaction Type			
0000 0000	Mailbox	LT	0000	Transaction Size					
Buffer Address[31:0]									
Buffer Address[63:32]									
31	30	29	28	27	26	25			
24	23	22	21	20	19	18			
17	16	15	14	13	12	11			
10	9	8	7	6	5	4			
3	2	1	0						

- **Transaction Queue ID** – Indicates from which transaction queue this status message relates.
- **Transaction Type** – Reports the type of transaction completed.
- **Status** – Reports the status of the completed transaction.

Status Code (in binary)	Status
0000	Transferred OK
0001	Response packet had format error, or contained ERROR status.
0010	Response Time-Out occurred
0011	Response payload had different length to that requested.
0100	MESSAGE/DOORBELL RETRY limit reached.
0101	Unknown Transaction Type
0110	Unable to process non-MAINTENANCE transaction because transmission only enabled for MAINTENANCE packets
0111	DMA Error encountered during payload read or write
1000	DMA Error encountered during transaction queue entry read
1001	Transaction specified a size which is unsupported for the transaction type
1010-1111	Reserved

- **LT** – Valid only for MESSAGE transactions, reports the Letter value used (0-3).
- **Mailbox** – Valid only for MESSAGE transactions, reports the destination Mailbox used (0-3).
- **Transaction Size** – Reports the number of bytes transferred for the transaction.

Transaction Size Code	Number of Bytes
0	1
...	...
4095	4096

- **Buffer Address** – Pointer to the buffer, situated in host memory, used for the transaction. For write transactions (NWRITE, NWRITE_R, SWRITE, MESSAGE, MAINTENANCE Write) the buffer holds the data that was transferred. For read transactions (NREAD, MAINTENANCE Read), the Buffer Address points to the buffer where the retrieved data

has been placed. For 32-bit address host systems, the upper 32 bits may not be written by the SRIO controller.

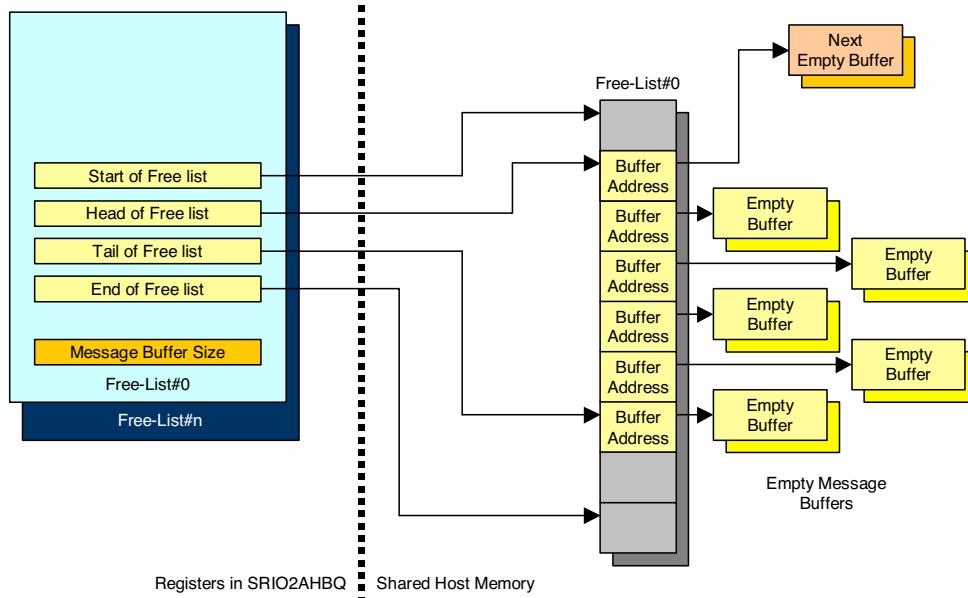
All other fields as previously defined.

20.4.5.2 Destination Data Structures

Free-Lists

The Free-lists provide the mechanism for the host to define pools of buffers into which the SRIO controller writes received MESSAGES. The free-lists are circular queues sited in shared memory. They are defined and controlled as with other queues through internal pointers. Each entry on a free-list contains a pointer to an associated buffer. Free-buffers are removed from the head of the queue by SRIO controller, and the head pointer updated, whilst the host must return empty buffers to the tail, updating the tail pointer.

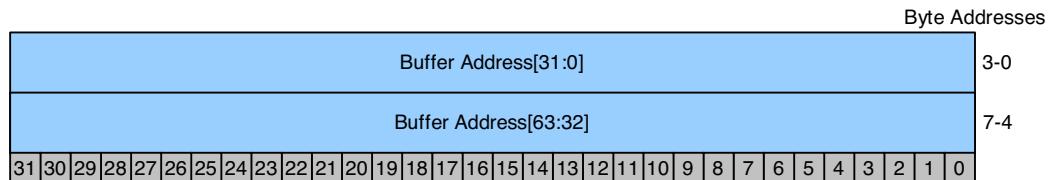
Figure 20-10. Free-Lists



The Free-List are allocated one Free-List per Mailbox. Buffers are allocated from the Free-List corresponding to the mailbox on which the MESSAGE is received. The maximum number of mailboxes available is limited to 4.

All the buffers on a Free-List are of equal size and defined in the Free-List Buffer Size Registers. Each entry of the Free-List is two words and contains a pointer to an empty buffer that may be used by the SRIO controller.

Figure 20-11. Free-List Entry (LittleEndian)



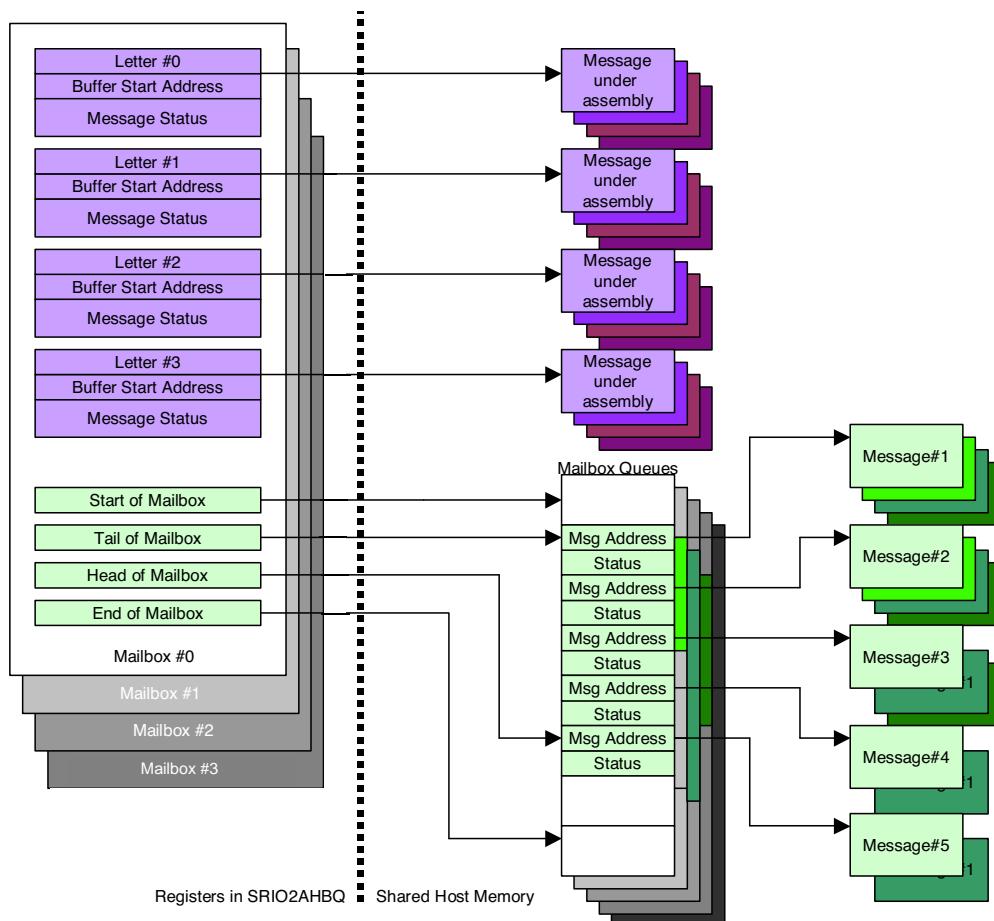
- **Buffer Address** – Pointer to the free buffer, situated in host memory. Buffers are always double-word aligned. For 32-bit address host systems, bits [63:32] need not be set (they are ignored by the SRIO controller).

Mailbox Queues

Mailbox Queues (limited to four) are used by the SRIO controller to inform the host of the receipt of MESSAGES.

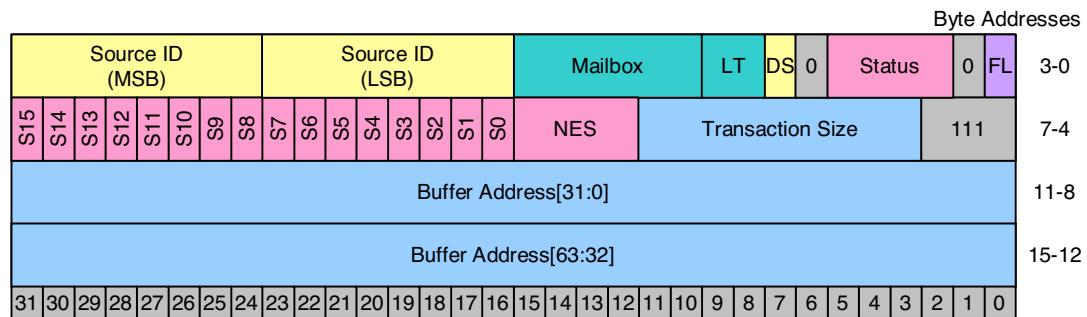
Each entry on the mailbox queue consists of the pointer to the start of the buffer, MESSAGE PDU length and relevant status information.

Figure 20-12. Mailbox Queues (Limited to four) for Received MESSAGES



Four separate Mailbox queues are supported, one corresponding to each Mailbox configured and a final one shared across all of the remaining mailboxes. Each entry on a Mailbox Queue is 8-words, but bytes 16-31 are Reserved.

Figure 20-13 shows the format of each Mailbox Queue entry.

Figure 20-13. Mailbox Queue Entry (Little Endian)

- **Status** – Reports the status of the released MESSAGE transaction. The user should discard the PDU if an error condition is indicated, but the buffer should be reclaimed/reused as normal.

Status Code	Status
0000	Complete MESSAGE transaction received OK
0001	Reserved Segment Size field (SSIZE) coding received
0010	For a multiple-segment message operation, the packet segment number (msgseg) was larger than the total number of segments expected (msglen)
0011	On a packet that was not the last of a transaction, the received payload length was not equal to the indicated Segment Size (SSIZE).
0100	Mailbox number greater than maximum supported number (as set by HMA field in MQ_CTRL3 register)
0101	Buffer size on applicable free list too small for complete MESSAGE transaction.
0110	Timeout occurred during reception of multi-segment MESSAGE
0111	DMA write of payload for a segment failed to complete successfully
1000	DMA read of Free-List entry failed to complete successfully (the Buffer Address in this entry is invalid)
Others	Reserved

- **S0-S15** - Sixteen flags that indicate the outstanding expected segments not received for a prematurely released PDU (applicable for all Errored Status conditions).

Sn (n = 0-15)	
0	Received/not applicable
1	Segment outstanding or errored

- **NES** -Reports the number of expected MESSAGE segments making up the PDU (0 = 1 segment, 15 = 16 segments).
- **LT** -Indicates the Letter number on which the MESSAGE was received.
- **Mailbox** – Indicates the Mailbox on which the MESSAGE was received.

- **Transaction Size** - Reports the number of double-words in the PDU.

Transaction Size Code	Number of Double-words	Number of Bytes
0	1	8
...
511	512	4096

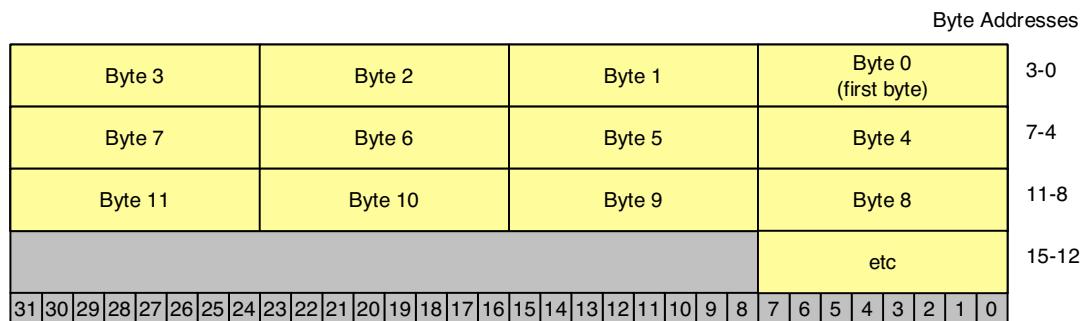
- **Buffer Address** - Pointer to the buffer used to store the PDU. For 32-bit address host systems, the upper 32 bits may not be written by the SRIO controller.

All other fields as previously defined.

20.4.5.3 Payload Data Structure

Payload data is read from or written to Buffers in host memory, the addresses of these Buffers being allocated by the Host. An example structure is shown in the [Figure 20-14](#). However, note that payloads do not always have to be aligned to word or double-word boundaries (any restrictions are noted in each of the preceding sections).

Figure 20-14. Payload Data Structure (Little Endian)



20.5 Serial RapidIO (SRIOD) Interface Registers

This section describes the SRIOD Interface Global Registers for the XLS6xx and XLS4xx devices. A summary of the SRIOD Interface Global registers is shown in [Table 20-9](#).

Table 20-9. Summary of XLS6xx and XLS4xx SRIOD Interface Global Registers

Register Offset DWord Addressing	Register Offset Byte Addressing	Name	R/W	Reset Value
0x000	0x000	SRIOD_CTRL	R/W	
0x001	0x004	SRIOD_PHY_CTRL0	R/W	
0x002	0x008	SRIOD_PHY_CTRL1	R/W	
0x003	0x00C	SRIOD_PHY0_CTRL	R/W	
0x004	0x010	SRIOD_PHY1_CTRL	R/W	
0x005	0x014	SRIOD_PHY2_CTRL	R/W	
0x006	0x018	SRIOD_PHY3_CTRL	R/W	
0x008	0x01C	SRIOD_COHERENT_MEM_BASE	R/W	
0x009	0x020	SRIOD_COHERENT_MEM_LIMIT	R/W	
0x010	0x024	SRIOD_REG_L2ALLOC_MEM_BASE	R/W	
0x011	0x028	SRIOD_REG_L2ALLOC_MEM_LIMIT	R/W	
0x012	0x02C	SRIOD_REG_READEX_MEM_BASE	R/W	
0x013	0x048	SRIOD_REG_READEX_MEM_LIMIT	R/W	
0x016	0x04C	SRIOD_REG_PHY_CR_CMD	R/W	
0x017	0x050	SRIOD_REG_PHY_CR_WR_DATA	R/W	
0x018	0x054	SRIOD_REG_PHY_CR_RESP	R/W	
0x019	0x058	SRIOD_REG_PHY_CR_RD_DATA	R/W	

20.5.1 Field Modifiability Codes

The following terms are used in the register descriptions to describe field modifiability type:

Table 20-10. Field Modifiability Codes

Type Code	Meaning	Description
RO	Read Only	The field contains a static value and cannot be written.
R/W	Read/Write	The field can be modified by software. Note that some fields, where indicated, are modifiable only indirectly, by writing to a shadow register then performing a soft reset.
RC	Read/Clear	The field can be read to give status information. Bits may be cleared by writing a '1' to them; writes of '0' are ignored.
RW1C	Read-Only Status/ Write-'1'-to-Clear Status Register	Register bits indicate status when read. A set bit indicating a status event may be cleared by writing a '1'. Writing '0' to RW1C bits has no effect. Writing from the application side (if any) requires careful synchronization with host software.
WO	Write only	Any write to this field resets the field to the default value.

20.5.1.1 Register Address Offsets

In the following sections, register address offsets are given for byte addressing.

20.5.2 SRIO_CTRL

Address Offset: 0x000

Bit#	Name	Description	Attribute	Reset
3:0	SRIO_REG_H_RST_N	One bit for each SRIO Controller, bit[0] for Controller_0 1 = Reset each SRIO Controller	R/W	0
7:4	SRIO_REG_LOOP_EN		R/W	0
11:8	SRIO_REG_CFG_FLUSH		R/W	0
15:12	Reserved	Reserved	R/W	0
17:16	SRIO_REG_DVT0		R/W	0
19:18	SRIO_REG_DVT1		R/W	0
21:20	SRIO_REG_DVT2		R/W	0
23:22	SRIO_REG_DVT3		R/W	0
27:24	SRIO_REG_FLUSH		R/W	0
28	SRIO_REG_PEFETCH_TQ_DISABLE	'1' = Disable the Prefetch to Transaction Queue_0	R/W	0
29	SRIO_REG_PEFETCH_FL_DISABLE	'1' = Disable the Prefetch to Free-List Queue	R/W	0
30	SRIO_REG_32B_WRITE	'0' = Writes are not aligned to 32 bytes '1' = Enable all the Writes to be 32-byte aligned	R/W	0
31	Reserved	Reserved	R/W	0

20.5.3 SRIO_PHY_CTRL0

Address Offset: 0x001

Bit#	Name	Description	Attribute	Reset
0	SRIO_REG_PHY_VP_IS_1P2		R/W	0
5:1	SRIO_REG_PHY_ACJT_LVL		R/W	01000
6	SRIO_REG_PHY_RTUNE_DO_TUNE		R/W	0
11:7	SRIO_REG_PHY_LOS_LVL		R/W	10011
16:12	SRIO_REG_PHY_TX_LVL		R/W	11111
20:17	SRIO_REG_PHY_TX_CLK_ALIGN		R/W	0000
31:21	Reserved	Reserved	R/W	0

20.5.4 SRIO_PHY_CTRL1

Address Offset: 0x002

Bit#	Name	Description	Attribute	Reset
4:0	SRIO_REG_PHY_MPLL_NCY	1.25 Gbaud = '00001' 2.50 Gbaud = '00100' 3.125 Gbaud = '00101'	R/W	See Desc
6:5	SRIO_REG_PHY_MPLL_NCY5	1.25 Gbaud = '10' 2.50 Gbaud = '00' 3.125 Gbaud = '01'	R/W	See Desc
8:7	SRIO_REG_PHY_MPLL_PRESCALE		R/W	10
9	SRIO_REG_PHY_MPLL_SS_EN		R/W	0
12:10	SRIO_REG_PHY_MPLL_CKO_WORD _CON		R/W	001
15:13	SRIO_REG_PHY_MPLL_INT_CTL		R/W	000
18:16	SRIO_REG_PHY_MPLL_PROP_CTL		R/W	111
22:19	SRIO_REG_PHY_HALF_RATE		R/W	0000
31:21	Reserved	Reserved	R/W	0

20.5.5 SRIO_PHY0_CTRL

Address Offset: 0x003

Bit#	Name	Description	Attribute	Reset
0	SRIO_REG_PHY0_TX_CALC		R/W	0
3:1	SRIO_REG_PHY0_TX_ATTEN		R/W	000
5:4	SRIO_REG_PHY0_TX_EDGERATE		R/W	00
9:6	SRIO_REG_PHY0_TX_BOOST		R/W	1001
10	SRIO_REG_PHY0_RX_TERM_EN		R/W	1
11	SRIO_REG_PHY0_RX_ALIGN_EN		R/W	0
14:12	SRIO_REG_PHY0_RX_EQ_VAL		R/W	101
17:15	SRIO_REG_PHY0_RX_DPLL_MODE		R/W	001
18	SRIO_REG_PHY0_RX_DPLL_RESET		R/W	0
20:19	SRIO_REG_PHY0_RX_LOS_CTL		R/W	11
31:21	Reserved	Reserved	R/W	0

20.5.6 SRIO_PHY1_CTRL

Address Offset: 0x004

Bit#	Name	Description	Attribute	Reset
0	SRIO_REG_PHY1_TX_CALC		R/W	0
3:1	SRIO_REG_PHY1_TX_ATTEN		R/W	000

Bit#	Name	Description	Attribute	Reset
5:4	SRIO_REG_PHY1_TX_EDGERATE		R/W	00
9:6	SRIO_REG_PHY1_TX_BOOST		R/W	1001
10	SRIO_REG_PHY1_RX_TERM_EN		R/W	1
11	SRIO_REG_PHY1_RX_ALIGN_EN		R/W	0
14:12	SRIO_REG_PHY1_RX_EQ_VAL		R/W	101
17:15	SRIO_REG_PHY1_RX_DPLL_MODE		R/W	001
18	SRIO_REG_PHY1_RX_DPLL_RESET		R/W	0
20:19	SRIO_REG_PHY1_RX_LOS_CTL		R/W	11
31:21	Reserved	Reserved	R/W	0

20.5.7 SRIO_PHY2_CTRL

Address Offset: 0x005

Bit#	Name	Description	Attribute	Reset
0	SRIO_REG_PHY2_TX_CALC		R/W	0
3:1	SRIO_REG_PHY2_TX_ATTEN		R/W	000
5:4	SRIO_REG_PHY2_TX_EDGERATE		R/W	00
9:6	SRIO_REG_PHY2_TX_BOOST		R/W	1001
10	SRIO_REG_PHY2_RX_TERM_EN		R/W	1
11	SRIO_REG_PHY2_RX_ALIGN_EN		R/W	0
14:12	SRIO_REG_PHY2_RX_EQ_VAL		R/W	101
17:15	SRIO_REG_PHY2_RX_DPLL_MODE		R/W	001
18	SRIO_REG_PHY2_RX_DPLL_RESET		R/W	0
20:19	SRIO_REG_PHY2_RX_LOS_CTL		R/W	11
31:21	Reserved	Reserved	R/W	0

20.5.8 SRIO_PHY3_CTRL

Address Offset: 0x006

Bit#	Name	Description	Attribute	Reset
0	SRIO_REG_PHY3_TX_CALC		R/W	0
3:1	SRIO_REG_PHY3_TX_ATTEN		R/W	000
5:4	SRIO_REG_PHY3_TX_EDGERATE		R/W	00
9:6	SRIO_REG_PHY3_TX_BOOST		R/W	1001
10	SRIO_REG_PHY3_RX_TERM_EN		R/W	1
11	SRIO_REG_PHY3_RX_ALIGN_EN		R/W	0
14:12	SRIO_REG_PHY3_RX_EQ_VAL		R/W	101
17:15	SRIO_REG_PHY3_RX_DPLL_MODE		R/W	001
18	SRIO_REG_PHY3_RX_DPLL_RESET		R/W	0

Bit#	Name	Description	Attribute	Reset
20:19	SRIO_REG_PHY3_RX_LOS_CTL		R/W	11
31:21	Reserved	Reserved	R/W	0

20.5.9 SRIO_COHERENT_MEM_BASE

Address Offset: 0x008

Bit#	Name	Description	Attribute	Reset
31:0	SRIO_REG_COHERENT_MEM_BASE		R/W	0

20.5.10 SRIO_COHERENT_MEM_LIMIT

Address Offset: 0x009

Bit#	Name	Description	Attribute	Reset
31:0	SRIO_REG_COHERENT_MEM_LIMIT		R/W	0

20.5.11 SRIO_REG_L2ALLOC_MEM_BASE

Address Offset: 0x0010

Bit#	Name	Description	Attribute	Reset
31:0	SRIO_REG_L2ALLOC_MEM_BASE		R/W	0

20.5.12 SRIO_REG_L2ALLOC_MEM_LIMIT

Address Offset: 0x0010

Bit#	Name	Description	Attribute	Reset
31:0	SRIO_REG_L2ALLOC_MEM_LIMIT		R/W	0

20.5.13 SRIO_REG_READEX_MEM_BASE

Address Offset: 0x0012

Bit#	Name	Description	Attribute	Reset
31:0	SRIO_REG_READEX_MEM_BASE		R/W	0

20.5.14 SRIO_REG_READEX_MEM_LIMIT

Address Offset: 0x0013

Bit#	Name	Description	Attribute	Reset
31:0	SRIO_REG_READEX_MEM_LIMIT		R/W	0

20.5.15 SRIO_REG_PHY_CR_CMD

Address Offset: 0x0016

Bit#	Name	Description	Attribute	Reset
15:0	SRIO_REG_PHY_CR_ADDR		R/W	
16	SRIO_REG_PHY_CR_CMD		R/W	
31:17	Reserved	Reserved	R/W	0

20.5.16 SRIO_REG_PHY_CR_WR_DATA

Address Offset: 0x0017

Bit#	Name	Description	Attribute	Reset
15:0	SRIO_REG_PHY_CR_WR_DATA		R/W	
31:16	Reserved	Reserved	R/W	0

20.5.17 SRIO_REG_PHY_CR_RESP

Address Offset: 0x0018

Bit#	Name	Description	Attribute	Reset
0	SRIO_REG_PHY_CR_RESP		R/W	
31:2	Reserved	Reserved	R/W	0

20.5.18 SRIO_REG_PHY_CR_RD_DATA

Address Offset: 0x0019

Bit#	Name	Description	Attribute	Reset
15:0	SRIO_REG_PHY_CR_RD_DATA		R/W	
31:16	Reserved	Reserved	R/W	0

20.6 SRIo Programming Register Space

All registers have bits numbered in accordance with the RapidIO standards (reference [1]). This defines the bit ordering to be in big-endian format where bit[0] is the most-significant bit in a byte/word. When accessing these registers from the slave Generic Host Interface it should be noted that the register bits are connected as shown below. However, note that no twisting of bytes occurs (the bits are just labelled in reverse).

Table 20-11.

m s b	MSByte							i s b	Top row is RapidIO (Big Endian) bit nomenclature																						
0	1	2	3	4	5	6	7	8	9	1	1	1	1	1	1	1	1	1	1	1	1	2	2	2	2	2	2	2	2	3	3
Bottom Row is generic Host Interface bit labels																															
1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0	9	8	7	6	5	4	3	2	1	0

Table 20-12. Summary of XLS6xx and XLS4xx SRIo Interface Global Registers

Register Offset Byte Addressing	Name	R/W	Reset Value
RapidIO Capability Space Registers			
0x0000	DEVICE_IDENTITY_CAR (D_ID_CAR)	R	-
0x0004	DEVICE_INFORMATION_CAR (D_IF_CAR)	R	-
0x0018	SOURCE_OPERATIONS_CAR (S_OP_CAR)	R	0x0000_FC04
0x001C	DESTINATION_OPERATION_CAR (D_OP_CAR)	R	0x0000_FC04
Command and Status Space Registers			
0x004C	Processing Element Logical Layer Control CSR (PE_LL_CSR)	R/W	0X0000_0001
0x0058	Local Configuration Space High Base Address CSR (LCS_UBA_CSR)	R/W	0X0000_0000
0x005C	Local Configuration Space Base Address CSR (LCS_LBA_CSR)	R/W	0X0000_0000
0x0060	Base Device ID CSR (BD_ID_CSR)	R/W	0X0000_0000
0x0068	Host Base Device ID Lock CSR (HBD_ID_CSR)	R/W	0X0000_FFFF
0x006C	Component TAG CSR (CP_TAG_CSR)	R/W	0X0000_0000
Extended Features Registers			
0x0100	1x/4x LP-Serial Port maintenance Block Header (PM_BLK_HDR)	R	0x0000_0002
0x0120	Port Link Time-out Control CSR (PL_TO_CSR)	R/W	0xFFFF_FF00
0x0124	Port Response Time-out Control CSR (PR_TO_CSR)	R/W	0xFFFF_FF00
0x013C	Port General Control CSR (PG_CTRL_CSR)	R/W	0x0000_0000
0x0140	Port 0 Link Maintenance Request CSR (P0_LMRQ_CSR)	R/W	0x0000_0000
0x0144	Port 0 Link Maintenance Response CSR (P0_LMRS_CSR)	R	0x0000_0000
0x0148	Port 0 Local AckID CSR (P0_LAID_CSR)	R	0x0000_0000
0x0158	Port 0 Error and Status CSR (P0_EAS_CSR)	Mixed	0x0000_0001
0x015C	Port 0 Control CSR (P0_CTRL_CSR)	Mixed	0x0000_0001

Note: Slave Generic Host Interface accesses to on-chip registers will never result in a Retry. Read accesses will always result in about five wait states inserted before the data is returned, whilst writes result in a single wait states.

20.6.1 DEVICE_IDENTITY_CAR

Identifies the vendor that manufactured the device and uniquely identifies the type of device.

Address Offset: 0x0000

Bit#	Name	Description	Attribute	Reset
15:0	DID	Device identifier	R	
31:16	DVID	Device vendor identifier	R	

20.6.2 DEVICE_INFORMATION_CAR

Identifies the revision of the device.

Address Offset: 0x0004

Bit#	Name	Description	Attribute	Reset
31:0	DRL	Device revision level	R	

20.6.3 SOURCE_OPERATIONS_CAR

Defines the set of RapidIO operations (in addition to maintenance reads and writes) that can be issued by this device. A '0' means not supported; a '1' means supported.

Address Offset: 0x0018

Bit#	Name	Description	Attribute	Reset
15:0	Reserved	Reserved	R	0
16	NRD	Generate NREAD	R	1
17	NWR	Generate NWRITE	R	1
18	SWR	Generate SWRITE	R	1
19	NWRR	Generate NWRITE_R	R	1
20	MESS	Generate MESSAGE	R	1
21	DBELL	Generate DOORBELL	R	1
22	ACAS	Generate ATOMIC compare and swap	R	0
23	ATAS	Generate ATOMIC test and swap	R	0
24	AINC	Generate ATOMIC increment	R	0
25	ADEC	Generate ATOMIC decrement	R	0
26	ASET	Generate ATOMIC set	R	0
27	ACLR	Generate ATOMIC clear	R	0
28	ASWAP	Generate ATOMIC swap	R	0
29	PW	Generate Maintenance Port-write	R	1
31:30	Reserved	Reserved	R	00

20.6.4 DESTINATION_OPERATION_CAR

Defines the set of operations (in addition to maintenance reads and writes) that are supported by this device. A '0' means not supported; a '1' means supported.

Address Offset: 0x001C

Bit#	Name	Description	Attribute	Reset
15:0	Reserved	Reserved	R	0
16	NRD	Accept NREAD	R	1
17	NWR	Accept NWRITE	R	1
18	SWR	Accept SWRITE	R	1
19	NWRR	Accept NWRITE_R	R	1
20	MESS	Accept MESSAGE	R	1
21	DBELL	Accept DOORBELL	R	1
22	ACAS	Accept ATOMIC compare and swap	R	0
23	ATAS	Accept ATOMIC test and swap	R	0
24	AINC	Accept ATOMIC increment	R	0
25	ADEC	Accept ATOMIC decrement	R	0
26	ASET	Accept ATOMIC set	R	0
27	ACLR	Accept ATOMIC clear	R	0
28	ASWAP	Accept ATOMIC swap	R	0
29	PW	Accept Maintenance Port-write	R	1
31:30	Reserved	Reserved	R	00

20.6.5 PE_LL_CSR

Processing Element Logical Layer Control CSR.

Address Offset: 0x004C

Bit#	Name	Description	Attribute	Reset
28:0	Reserved	Reserved	R	00
31:29	EAC	Controls the number of address bits generated by the PE as source and processed by the PE as the target of an operation. Whether values other than 34-bit are allowed are controlled by the Extended Addressing settings in the Processing Element Feature Capability register: 100 = Generates / expects 66-bit offset addresses 010 = Generates / expects 50-bit offset addresses 001 = Generates / expects 34-bit offset addresses others – Reserved	R	001

20.6.6 LCS_UBA_CSR

Local Configuration Space High Base Address CSR: Specifies the most significant bytes of a local physical address offset for the processing elements configuration spaces if the local address space is greater than 32-bits.

Address Offset: 0x0058

Bit#	Name	Description	Attribute	Reset
31:0	Reserved	Reserved for a 34-bit local Physical Address	R/W	0x0000

20.6.7 LCS_LBA_CSR

Local Configuration Space Base Address CSR. Specifies the local physical address offset for the processing element's configuration registers register space, causing the register space to be physically mapped in the processing element. This register allows configuration and maintenance of processing elements through regular read and write operations rather than maintenance configuration operations. The double-word offset is right justified in the register.

Address Offset: 0x005C

Bit#	Name	Description	Attribute	Reset
0	Reserved	Reserved for a 34-bit local Physical Address	R/W	0
31:1	LCSBA	Bits 0-30 of a 34-bit local Physical Address	R/W	0x0000_0000

20.6.8 BD_ID_CSR

Base Device ID CSR.

Address Offset: 0x0060

Bit#	Name	Description	Attribute	Reset
7:0	Reserved	Reserved	R/W	0x0000
15:8	SDID	Base ID of the device in a Small common transport system. The default value depends upon the setting of the DVT input pins as follows: 0 = 0x00 (host device, low priority) 1 = 0x01 (host device, high priority) 2 = 0xFE (agent device with boot ROM) 3 = 0xFF (non-boot agent device)	R/W	f(DVT)
31:16	LDID	Base ID of the device in a Large common transport system. The default value depends upon the setting of the DVT input pins as follows: 0 = 0x0000 (host device, low priority) 1 = 0x0001 (host device, high priority) 2 = 0xFFFFE (agent device with boot ROM) 3 = 0xFFFF (non-boot agent device)	R/W	f(DVT)

20.6.9 HBD_ID_CSR

Host-Base Device ID Lock CSR: Contains the base device ID for the PE responsible for initializing this PE. This register is a write once/reset-able field that provides a lock function. Once written, all further writes are ignored, except in the case where the written value matches the value contained in the register, in which case the register is returned to its default value of 0xFFFF. Once written, a PE must read back the value to verify that it owns the lock before attempting to initialize this PE.

Address Offset: 0x0068

Bit#	Name	Description	Attribute	Reset
15:0	Reserved	Reserved	R/W	0x0
31:1	HBDID	Host Base Device ID for the PE that initializing this PE.	R/W	0xFFFF

20.6.10 CP_TAG_CSR

Component TAG CSR. Can be used by software as a general register for any function.

Address Offset: 0x006C

Bit#	Name	Description	Attribute	Reset
31:0	CTAG	Component TAG for the processing element.	R/W	0x0

20.6.11 PM_BLK_HDO

1x/4x *LP-Serial Port-Maintenance Block Header*.

Address Offset: 0x0100

Bit#	Name	Description	Attribute	Reset
15:0	EF_PTR	Extended Features: Hardwired to indicate “no more blocks”.	R/W	0x0000
31:16	EF_ID	Extended Features Identity. Hardwired to indicate “Software Assisted generic endpoint”	R/W	0x0002

20.6.12 PL_TO_CSR

Port-Link Time-out Control CSR. The port link time-out control register contains the time-out value for all ports on the device. This time-out is for link events such as sending a packet to receiving the corresponding acknowledge, and sending a link request to receiving the corresponding link-response. The reset value is the maximum time-out interval, and represents between 3 and 5 seconds.

Address Offset: 0x0120

Bit#	Name	Description	Attribute	Reset
23:0	TO	Time-out interval value	R/W	0xFFFFFFFF
31:24	Reserved	Reserved	R/W	0x0

20.6.13 PR_TO_CSR

Port-Response Time-out Control CSR. The port-response time-out control register contains the time-out timer count for all ports on the device. This time-out is used in Source Processing from sending a requestM packet to receiving a corresponding response packet, and in Destination Processing to time gaps between receiving different segments of a multi-segment message. The reset value is the maximum time-out interval, and represents between 3 and 5 seconds (the actual time is a function of the Logical Layer clock frequency and the value programmed into the Port Response Timeout Pre-scale register).

Address Offset: 0x0124

Bit#	Name	Description	Attribute	Reset
23:0	TO	Time out interval value. With Pre-scale register set correctly, timeouts are handled in blocks of 61.44 us. The variations shown below come about due to where within the current timeout block any new request is issued. Although the lower 8-bits of this register field are implemented, they are unused and have been set at 0x00 for clarity in these examples. 0x0000000 = Timeout in 0-61 us (i.e., the next 61 us block) 0x000100 = Timeout in 61 us-123 us 0xFFFFF00 = Timeout in 4.03 s	R/W	0xFFFFFFFF
31:24	Reserved	Reserved	R/W	0x0

20.6.14 PG_CTRL_CSR

Port General Control CSR. Default values are set by the DVT input pins.

Address Offset: 0x013C

Bit#	Name	Description	Attribute	Reset
0	HOST	A host device is a device that is responsible for system exploration, initialization, and maintenance. Agents or slave devices are typically initialized by Host devices: 0 = Agent or slave device 1 = Host device	R/W	DVT[1]
1	MSTEN	Master Enable. Controls whether or not a device is allowed to issue requests into the system: 0 = Device cannot issue requests (may only respond to them) 1 = Device can issue requests	R/W	DVT[1]
2	DVRD	Discovered. This device has been located by the processing element responsible for system configuration: 0 = Device has not been previously discovered 1 = Device has been discovered by another processing element	R/W	DVT[1]
31:3	Reserved	Reserved	R/W	0x0

20.6.15 P0_LMRQ_CSR

Port 0 Link Maintenance Request CSR. A write generates a link-request control symbol on the corresponding RapidIO port interface.

Address Offset: 0x0140

Bit#	Name	Description	Attribute	Reset
28:0	Reserved	Reserved	R/W	0x0
31:29	CMD	Command will be sent in the link-request control symbol. If read this returns the last value sent. 0x0 = null (do not send Link-Request) 0x3 = Reset-device 0x4 = Input-Status others = Reserved	R/W	000

20.6.16 P0_LMRS_CSR

Port 0 Link Maintenance Response CSR. Provides the status received in the link-response control symbol.

Address Offset: 0x0144

Bit#	Name	Description	Attribute	Reset
0	RV	Response Valid. If the link-request causes a link-response, a '1' indicates that the link-response has been received and the status fields are valid. If the link request does not cause a link response, a '1' indicates that the link-request has been transmitted. This bit automatically clears on read.	R	0
21:1	Reserved	Reserved	R	0x0
26:22	AIDS	AckID Status field from the link-response control symbol	R	0x00
31:27	LS	Link Status field from the link-response control symbol	R	0x00

20.6.17 P0_LAID_CSR***Port 0 Local AckID Status CSR.***

Note that writing new values to the OBAID and OSAID fields is recommended only when Port Error is set, and should be followed by writing a '1' to the COAID flag.

Address Offset: 0x0148

Bit#	Name	Description	Attribute	Reset
0	COAID	Clear Outstanding AckIDs. Writing a '1' causes all outstanding unacknowledged packets to be discarded. This action is only enabled when Port Error is set.	R/W	0
2:1	Reserved	Reserved	R/W	0x0
7:3	IBAID	Inbound AckID. Input port next expected ackID value.	R/W	0x00
18:8	Reserved	Reserved	R/W	0x0
23:19	OSAID	Outstanding AckID. Output port unacknowledged ackID status. Next expected acknowledge control symbol ackID field that indicates the ackID value expected in the next received acknowledge control symbol.	R/W	0x00
26:24	Reserved	Reserved	R/W	000
31:27	OBAID	Outbound Ack ID. Output port next transmitted ackID value. This value can be overridden but does not cause retransmission of unacknowledged outstanding packets. ^a	R/W	0x00

- a. Note that strictly this is a non-compliance to the RapidIO standard (which calls for a write to this field to force retransmission of outstanding packets). Instead, error recovery by the retransmission of unacknowledged packets due to recoverable errors is handled automatically in hardware via the Output Port Error Recovery State Machine.

20.6.18 P0_EAS_CSR***Port 0 Error and Status CSR.***

Address Offset: 0x0158

Bit#	Name	Description	Attribute	Reset
10:0	Reserved	Reserved	R	0x0
11	ORE	Output Retry Encountered. Output port has encountered a retry condition. This bit is set when bit[13] is set.	RWTC	0
12	OR	Output Retried. Output port has received a packet-retry control symbol and cannot make forward progress. This bit is set when bit[13] is set, and cleared when a packet-accepted or a packet-not-accepted control symbol is received.	R	0
13	ORS	Output port has received a packet-retry control symbol and is in the "Output Retry Stopped" state.	R	0
14	OEE	Output Error Encountered. Output port has encountered (and possibly recovered from) a transmission error. This bit is set when bit[15] is set.	RWTC	0
15	OES	Output is in the "output error stopped" state.	R	0
20:16	Reserved	Reserved	R	0x00
21	IRT	Input port is in the "Input Retry Stopped" state.	R	0

Bit#	Name	Description	Attribute	Reset
22	IEE	Input Error Encountered. Input port has encountered (and possibly recovered from) a transmission error. This bit is set when bit[23] is set.	RWTC	0
23	IES	Input port is in the "Input Error Stopped" state.	R	0
26:24	Reserved	Reserved	R	000
27	PWP	Port-write Pending. Port has generated a Maintenance Port-write transaction (by a request on a Transaction Queue).	RWTC	0
28	Reserved	Reserved	R	000
29	PERR	Port Error. Input or output port has encountered an error from which the hardware was unable to recover.	R	0
30	POK	Port Ok. The input and output ports are initialized and the port is exchanging error-free control symbols with the attached device.	R	0
31	PUI	Port Un-initialized. Input and output ports are not initialized. This bit and bit[30] are mutually exclusive.	R	0

20.6.19 P0_CTRL_CSR

Port 0 Control CSR.

Address Offset: 0x015C

Bit#	Name	Description	Attribute	Reset
1:0	PW	Port Width. Indicates the width of the port implemented in hardware: 0 = Single-lane port 1 = Four-lane port others = Reserved	R	00
4:2	IPW	Initialized Port Width. Indicates the width of the port after initialization: 0 = Single-lane, lane 0 1 = Single-lane, lane 2 2 = Four-lane others = Reserved	R	000
7:5	PWO	Port Width Override. Allows software to override the implemented port width. If overridden, the port will re-initialize to change to the requested size. 0 = No override 2 = Force single-lane, lane 0 3 = Force single-lane, lane 2 others = Reserved	R/W	000
8	PD	Port Disable. When disabled, the port receivers and drivers are incapable of receiving or transmitting any packets or control symbols. 0 = Ports enabled 1 = Ports disabled	R/W	0
9	OPE	Output Port Enable. When disabled, the port is only able to transmit I/O logical Maintenance packets. Control symbols are not affected and are sent normally. 0 = Output port disabled 1 = Output port enabled	R/W	0

Bit#	Name	Description	Attribute	Reset
10	IPE	Input Port Enable. When disabled, the port is only able to receive I/O logical Maintenance packets. Other types of packet generate packet-not accepted control symbols to force an error condition to be signalled by the sending device. Control symbols are not affected and are received and handled normally. 0 = Input port disabled 1 = Input port enabled	R/W	0
11	ECD	Error Checking Disable. This bit controls all RapidIO transmission error checking. The behavior of the device if an error occurs when error checking and recovery is disabled is undefined. 0 = Error checking and recovery is enabled 1 = Error checking and recovery is disabled	R/W	0
12	MEP	Multicast Event Participant. Controls if a multicast event is generated when a Transmit Multicast-event command is issued by the software via the PL_CMD register. 0 = Multicast events not generated 1 = Multicast events generated	R/W	0
30:13	Reserved	Reserved	R	0x0
31	PT	Indicates the Port Type: 0 = Parallel port 1 = Serial port	R	1

20.7 SRI0 Configuration and Control Registers

These SRI0 configuration and control registers are memory mapped into the PCIe space. Either the PCIe or the SRI0 controllers have access to this register space as selected by pin-strapping options at power-up reset time. See [Section 20.2.1 on page 984](#) for details.

See [Section 8.7 on page 241](#) for a description of the memory organization.

Table 20-13. Summary of XLS6xx and XLS4xx SRI0 Implementation Defined Registers

Register Offset Byte Addressing	Name	R/W	Reset Value
General Registers			
0x10000	Master Interrupt Status Register (MISR)	R	0x0000_0000
0x10004	Master Interrupt Enable Register (MIER)	RW	0x0000_0000
0x10010	Status Register (GEN_STAT)	RWTC	0x0000_0000
0x10014	Interrupt Enable Register (GEN_IER)	RW	0x0000_0000
0x10018	Control Register (GEN_CTRL)	RW	0x0000_0000
0x1001C	Port Response Timeout Pre-scale Register (GEN_PRTP)	RW	0x0000_002F
0x10020	Port Response Maximum Retries Register (GEN_PRMR)	RW	0x0000_00FF
0x10030	DMA Error Capture High Start Address (DEC_HSA_CSR)	R	0x0000_0000
0x10034	DMA Error Capture Low Start Address (DEC_LSA_CSR)	R	0x0000_0000
0x10038	DMA Error Capture Information (DEC_INFO_CSR)	R	0x0000_0000
Physical Layer Registers			
0x10100	Transmit Watermark Register (PL_TWM)	RW	0x0403_0200
0x10104	Control Register (PL_CTRL)	RW	0x0000_0101
0x10108	Status Register (PL_STAT)	R	0x0000_0000
0x1010C	Command Register (PL_CMD)	RW	0x0000_0000
RapidIO to Host Address Translation Registers			
0x10200	Lower Mask Register (AT_LMASK)	RW	0x0000_0000
0x10204	Higher Mask Register (AT_HMASK)	RW	0x0000_0000
0x10208	Lower Remap Register (AT_LREMAP)	RW	0x0000_0000
0x1020C	Higher Remap Register (AT_HREMAP)	RW	0x0000_0000
Error Injection Registers			
0x10300	Control Symbol Control Register 1 (EI_CS_CTRL1)	RW	0x0000_0000
0x10304	Control Symbol Control Register 2 (EI_CS_CTRL2)	RW	0x0000_0000
0x10308	Packet Control Register 1 (EI_PKT_CTRL1)	RW	0x0000_0000
0x1030C	Packet Control Register 2 (EI_PKT_CTRL2)	RW	0x0000_0000
Transaction Queue Registers (Per Transaction Queue where n = 0 to NTQ-1)			
0x20000 to 0x20000+0x20*n	Start pointer (TQn_SPTR)	RW	0x0000_0000
0x20004 to 0x20004+0x20*n	End pointer (TQn_EPTR)	RW	0x0000_0000
0x20008 to 0x20008+0x20*n	Head pointer (TQn_HPTR)	R	0x0000_0000
0x2000C to 0x2000C+0x20*n	Tail pointer (TQn_TPTR)	RW	0x0000_0000
0x20010 to 0x20010+0x20*n	Common high-portion of pointer (TQn_UPTR)	RW	0x0000_0000
0x20200	Control Register 1 (TQ_CTRL1)	RW	0x0000_0000
0x20204	Control Register 2 (TQ_CTRL2)	RW	0x0000_0000

Table 20-13. Summary of XLS6xx and XLS4xx SRIO Implementation Defined Registers

Register Offset Byte Addressing	Name	R/W	Reset Value
0x20208	Status Register (TQ_STAT)	Mixed	0x0000_0000
0x2020C	Interrupt Enable Register (TQ_IER)	RW	0x0000_0000
Status Queue Registers (Per Status Queue where n = 0 to NTQ-1)			
0x20400 to 0x20400+0x20*n	Start pointer (SQn_SPTR)	RW	0x0000_0000
0x20404 to 0x20404+0x20*n	End pointer (SQn_EPTR)	RW	0x0000_0000
0x20408 to 0x20400+0x20*n	Head pointer (SQn_HPTR)	RW	0x0000_0000
0x20410 to 0x20410+0x20*n	Tail pointer (SQn_TPTR)	R	0x0000_0000
0x20410 to 0x20410+0x20*n	Common high-portion of pointer (SQn_UPTR)	RW	0x0000_0000
0x20600	Control Register (SQ_CTRL)	RW	0x0000_0000
0x20608	Status Register (SQ_STAT)	R	0x0000_0000
0x2060C	Interrupt Enable Register (SQ_IER)	RW	0x0000_0000
Doorbell FIFO Registers			
0x20800	Control Register (DF_CTRL)	RW	0x0000_0000
0x20804	Information Register (DF_INFO)	R	0x0000_0000
0x20808	Status Register (DF_STAT)	R	0x0000_0000
0x2080C	Interrupt Enable Register (DF_IER)	RW	0x0000_0000
0x20810	Read Port Register (DF_RP)	R	0x0000_0000
Port-write FIFO Registers			
0x20904	Information register (PW_INFO)	R	0x0000_0000
0x20908	Status Register (PW_STAT)	Mixed	0x0000_0000
0x2090C	Interrupt Enable Register (PW_IER)	RW	0x0000_0000
0x20910	Port-write Read Port register (PW_RP)	R	0x0000_0000
Free-List Registers (per Free-List where n = 0 to 15)			
0x20A00 to 0x20A00+0x20*n	Start pointer (FLn_SPTR)	RW	0x0000_0000
0x20A04 to 0x20A04+0x20*n	End pointer (FLn_EPTR)	RW	0x0000_0000
0x20A08 to 0x20A08+0x20*n	Head pointer (FLn_HPTR)	R	0x0000_0000
0x20A0C to 0x20A0C+0x20*n	Tail pointer (FLn_TPTR)	RW	0x0000_0000
0x20A10 to 0x20A10+0x20*n	Common high-portion of pointer (FLn_UPTR)	RW	0x0000_0000
0x20A14 to 0x20A14+0x20*n	Buffer Size Register (FLn_BUF)	RW	0x0000_0000
0x20C00	Control Register (FL_CTRL)	RW	0x0000_0000
0x20C08	Status Register (FL_STAT)	R	0x0000_0000
0x20C0C	Interrupt Enable Register (FL_IER)	RW	0x0000_0000

Table 20-13. Summary of XLS6xx and XLS4xx SRIO Implementation Defined Registers

Register Offset Byte Addressing	Name	R/W	Reset Value
Mailbox Queue Registers (Per Mailbox Queue where n = 0 to 3)			
0x21000 to 0x21000+0x20*n	Start pointer (MQn_SPTR)	RW	0x0000_0000
0x21004 to 0x21004+0x20*n	End pointer (MQn_EPTR)	RW	0x0000_0000
0x21008 to 0x21008+0x20*n	Head pointer (MQn_HPTR)	RW	0x0000_0000
0x2100c to 0x2100c+0x20*n	Tail pointer (MQn_TPTR)	R	0x0000_0000
0x21010 to 0x21010+0x20*n	Common high-portion of pointer (MQ_UPTR)	RW	0x0000_0000
0x21800	Control Register 1 (MQ_CTRL1)	RW	0x0000_0000
0x21804	Control Register 2 (MQ_CTRL2)	RW	0x0000_0000
0x21808	Control Register 3 (MQ_CTRL3)	RW	0x0000_0000
0x21820	Status Register 1 (MQ_STAT1)	R	0x0000_0000
0x21824	Status Register 2 (MQ_STAT2)	R	0x0000_0000
0x21828	Status Register 3 (MQ_STAT3)	R	0x0000_0000
0x2182C	Status Register 4 (MQ_STAT4)	R	0x0000_0000
0x21830	Interrupt Enable Register 1 (MQ_IER1)	RW	0x0000_0000
0x21834	Interrupt Enable Register 2 (MQ_IER2)	RW	0x0000_0000
0x21838	Interrupt Enable Register 3 (MQ_IER3)	RW	0x0000_0000
0x2183C	Interrupt Enable Register 4 (MQ_IER4)	RW	0x0000_0000
Logical/Transport Layer Error Reporting Registers (Based on Error Management Extension specification)			
0x30008	Error Detect CSR (LTL_ERD_CSR)	RW	0x0000_0000
0x3000C	Error Enable CSR (LTL_ERE_CSR)	RW	0x0000_0000
0x30010	High Address Capture CSR (LTL_HAC_CSR)	RW	0x0000_0000
0x30014	Address Capture CSR (LTL_LAC_CSR)	RW	0x0000_0000
0x30018	Device ID Capture CSR (LTL_DID_CSR)	RW	0x0000_0000
0x3001C	Control Capture CSR (LTL_CC_CSR)	RW	0x0000_0000
Port 0 Error Reporting Registers (Based on Error Management Extension specification)			
0x30040	Error Detect CSR (P0_ERD_CSR)	RW	0x0000_0000
0x30044	Error Enable CSR (P0_ERE_CSR)	RW	0x0000_0000
0x30048	Attributes Capture CSR (P0_AC_CSR)	RW	0x0000_0000
0x3004C	Packet Capture 0 CSR (P0_PC0_CSR)	RW	0x0000_0000
0x30050	Packet Capture 1 CSR (P0_PC1_CSR)	RW	0x0000_0000
0x30054	Packet Capture 2 CSR (P0_PC2_CSR)	RW	0x0000_0000
0x30058	Packet Capture 3 CSR (P0_PC3_CSR)	RW	0x0000_0000

20.7.1

Interrupt Generation

The SRIO controller has many potential interrupt sources. In order to minimize the overhead of servicing interrupts, a 2-tier system of registers is implemented so that only two register reads are needed by the host to determine the cause of an interrupt.

The top tier is the Master Interrupt Status Register (MISR). This should be the first read whenever a hardware interrupt is signalled. Each bit in the register indicates which interrupt group(s) are responsible for causing the interrupt (note that one or more bits may be set). Each interrupt group has a separate associated register – the second tier.

The second-tier registers (generally named *_STAT) have bits that mark significant events or conditions within the group. The status bits relating to transitory events (e.g., Multicast Event Received in GEN_STAT) are ‘sticky’ in that they do not automatically clear when the underlying event has cleared. This is to ensure that the host can always determine the cause of an interrupt. These are marked as type RWTC and the host must clear them when required by writing a ‘1’ to them. The status bits relating to conditions that are more static (e.g. status queue not empty) are not sticky. These are marked as type R and automatically clear when the host has performed appropriate actions (e.g., read entries off the status queue). In addition, each second-tier event register has a companion Interrupt Enable Register (named *_IER) to allow the host to only allow selected sources to generate interrupts. When the bit-wise AND of the pair of second tier registers results in no ‘1’s, the corresponding group bit in MISR automatically clears.

20.7.1.1 MISR

Master Interrupt Status Register. Indicates which sources of interrupt are currently active. Used in conjunction with the MIER register to generate the hardware active-low interrupt signal s_intreqn. An interrupt is generated if the bit-wise AND of this register and MIER results in any ‘1’s. Any interrupt must be cleared by clearing the source of the interrupt in the second-tier registers, or clearing the respective bit in the MIER register.

Address Offset: 0x10000

Bit#	Name	Description	Attribute	Reset
0:19		Reserved	R	All 0's
20	GEN	General Interrupt. This bit is set if any bit in the GEN_STAT and corresponding bit in GEN_IER registers are both set.	R	0b0
21	TQ	Transaction Queue Interrupt. This bit is set if any bit in the TQ_STAT and corresponding bit in TQ_IER registers are both set.	R	0b0
22	SQ	Status Queue Interrupt. This bit is set if any bit in the SQ_STAT and corresponding bit in SQ_IER registers are both set.	R	0b0
23	FL	Free-List Interrupt. This bit is set if any bit in the FL_STAT and corresponding bit in FL_IER registers are both set.	R	0b0
24	MQ4	Mailbox Queue Interrupt 4. This bit is set if any bit in the MQ_STAT4 and corresponding bit in MQ_IER4 registers are both set.	R	0b0
25	MQ3	Mailbox Queue Interrupt 3. This bit is set if any bit in the MQ_STAT3 and corresponding bit in MQ_IER3 registers are both set.	R	0b0
26	MQ2	Mailbox Queue Interrupt 2. This bit is set if any bit in the MQ_STAT2 and corresponding bit in MQ_IER2 registers are both set.	R	0b0
27	MQ1	Mailbox Queue Interrupt 1. This bit is set if any bit in the MQ_STAT1 and corresponding bit in MQ_IER1 registers are both set.	R	0b0
28	DF	Doorbell FIFO Interrupt. This bit is set if any bit in the DF_STAT and corresponding bit in DF_IER registers are both set.	R	0b0
29	PW	Port-write Interrupt. This bit is set if any bit in the PW_STAT and corresponding bit in PW_IER registers are both set.	R	0b0

Bit#	Name	Description	Attribute	Reset
30	P0	Port 0 Physical Layer Error Capture Interrupt. This bit is set if any bit in the P0_ERD_CSR and corresponding bit in the P0_ERE_CSR registers are both set.	R	0b0
31	LTL	Logical/Transport Layer Error Capture Interrupt. This bit is set if the LTL registers are locked.	R	0b0

20.7.1.2 MIER

Master Interrupt Enable Register: Set bit to '1' to enable respective group interrupt

Address Offset: 0x10004

Bit#	Name	Description	Attribute	Reset
0:19		Reserved	RW	All 0's
20	GEN	General Interrupt Enable	RW	0b1
21	TQ	Transaction Queue Interrupt Enable	RW	0b1
22	SQ	Status Queue Interrupt Enable	RW	0b1
23	FL	Free-List Interrupt Enable	RW	0b1
24	MQ4	Mailbox Queue Interrupt 4 Enable	RW	0b1
25	MQ3	Mailbox Queue Interrupt 3 Enable	RW	0b1
26	MQ2	Mailbox Queue Interrupt 2 Enable	RW	0b1
27	MQ1	Mailbox Queue Interrupt 1 Enable	RW	0b1
28	DF	Doorbell FIFO Interrupt Enable	RW	0b1
29	PW	Port-write Interrupt Enable	RW	0b1
30	P0	Port 0 Physical Layer Error Capture Interrupt Enable	RW	0b1
31	LTL	Logical/Transport Layer Error Capture Interrupt Enable	RW	0b1

20.7.2 General Registers

20.7.2.1 GEN_STAT

General Status Register: Indicates general events that have occurred. A '1' indicates that the event has been seen (but note that the condition of the device may have changed before this bit is read). To clear bits, perform a write to this register with a '1' in all positions where it is desired to clear the bit.

Address Offset: 0x10010

Bit#	Name	Description	Attribute	Reset
0:16	-	Reserved	RWTC	All 0's
17:22	TOOR	<p>Transaction Processor Out Of Resource events. To aid in diagnosing any problems with throughput, these events indicate if any internal resources temporarily run out, thus causing the Transaction Queue Processor to pause. Each bit indicates that a separate resource has run out:</p> <ul style="list-style-type: none"> [17] Space in Response Checker store (used for all transactions where responses expected). [18] Letters on requested Mailbox (used for Message transactions only) [19] Checker-Boards (used for segmenting transactions where responses expected only) [20] TID's (used for IO transactions where responses expected only) [21] Tracking Context stores (used for all transactions where responses expected) [22] Transmit buffers (used for all transactions) 	RWTC	0b000000
23	SQWE	Status Queue Entry Write Error. This bit is set if a DMA error is flagged during a Status Queue Entry write. This is a serious error which indicates a major system problem. The SRIO controller will attempt to continue, but note that the transfer data buffer associated with the transaction is likely lost.	RWTC	0b0
24	MQWE	Mailbox Queue Entry Write Error. This bit is set if a DMA error is flagged during a Mailbox Queue Entry write. This is a serious error which indicates a major system problem. The SRIO controller will attempt to continue, but note that the buffer associated with the transaction is retained and will be used for the next received message (and hence not lost).	RWTC	0b0
25	MCE	Message Configuration Error. This bit set when a MESSAGE RapidIO packet is received and the required Free List or Mailbox queue is not configured.	RWTC	0b0
26	DEC	DMA Error Captured. This bit is set when the DMA Error Capture registers contain valid information.	RWTC	0b0
27	POK	Port OK. This bit is set whenever the Port OK bit in the Port 0 Error and Status CSR is set.	RWTC	0b0
28	PUI	Port Un-Initialized. This bit is set whenever the Port Un-initialized bit in the Port 0 Error and Status CSR is set.	RWTC	0b0
29	PERR	Port Error. This bit is set whenever the Port Error bit in the Port 0 Error and Status CSR is set. To clear down, the host must first write a '1' to the bit in the Port 0 Error and Status CSR, followed by writing a '1' to this bit.	RWTC	0b0
30	MER	Multicast Event Received	RWTC	0b0
31	RST	Received a reset-device command over the RapidIO Interface	RWTC	0b0

20.7.2.2 GEN_IER

General Interrupt Enable Register: The GEN bit set in the MISR register is set (from which an interrupt can be generated if desired) whenever a bit in this register and the corresponding bit in the GEN_STAT register are both set.

Address Offset: 0x10014

Bit#	Name	Description	Attribute	Reset
0:16		Reserved	RW	All 0's
17:22	TOOR	Transaction Processor Out Of Resource events interrupt enables	RW	0b000000
23	SQWE	Status Queue Entry Write Error interrupt enable	RW	0b0
24	MQWE	Mailbox Queue Entry Write Error interrupt enable	RW	0b0
25	MCE	Message Configuration Error interrupt enable	RW	0b0
26	DEC	DMA Error Captured interrupt enable	RW	0b0
27	POK	Port OK event interrupt enable	RW	0b0
28	PUI	Port Un-initialized event interrupt enable	RW	0b0
29	PERR	Port Error event interrupt enable	RW	0b0
30	MER	Multicast Event Received event interrupt enable	RW	0b0
31	RST	Reset-device received event interrupt enable	RW	0b0

20.7.2.3 GEN_CTRL

General Control Register.

Address Offset: 0x10018

Bit#	Name	Description	Attribute	Reset
0	RCID	When set, disables the ability of receiver logical layer to process transactions. Used for test purposes only.	RW	0b0
1	RCHKD	Response Source Device ID Check Disable. When an expected RapidIO Response packet is received, its Source Device ID should match the Destination Device ID used in the corresponding Request packet. A check is made on this and the Response declared as Unsolicited if a mismatch occurs. However, when the transmit data is looped to the receiver (either internally or externally), this would cause all Responses to be rejected. This bit allows the check to be disabled. 0 = declare Unsolicited Response if Source Device ID mismatch, 1 = accept Response packets even if Source Device ID mismatch – <i>Use for test purposes only.</i>	RW	0b0
2:27		Reserved	RW	All 0's

Bit#	Name	Description	Attribute	Reset
28	SIDSB	Controls if the Source Device ID field (SID) of all generated Response packets is sourced from the local Source Device ID registers, or from the Destination Device ID field (DID) of the corresponding Request packet: 0 = SID sourced from Request packet DID (swapped), 1 = SID sourced from local Device ID registers (substituted)	RW	0b0
29	RSTEN	Reset Enable: Controls if a Reset-Device command received on RapidIO link is processed: 0 = Ignored 1 = Resets all SRIO Controllers	RW	0b0
30	SRST	Writing a '1' to this bit causes all blocks within the SRIO Controller to be reset.	RW	0b0
31	DIDRJ	Controls if the Destination Device ID field (DID) of all incoming packets is compared against the programmed Source Device ID value to determine if it is rejected or not. Note that Maintenance Read requests are always accepted. 0 = All packets accepted regardless of DID (promiscuous mode), 1 = Packet rejected if DID mismatch occurs (unless Maintenance read)	RW	0b0

20.7.2.4 Port Response Timeout Pre-scale Register

The logical layer implements timeouts for the receipt of responses from the transmission of requests. This timeout period is specified in the Port Response Timeout Control CSR (see section 6.3.3). As the design of the logical layer does not require a clock of a specific frequency, this pre-scale register is provided in order that the timeout can be specified in a consistent manner, independent of the logical layer clock frequency.

The value programmed into this pre-scale register represents the number of logical layer clock periods in one tick of the timeout counter. The RapidIO standard defines that a timeout value of 24 bits represents a time interval of between 3 and 5 seconds. Therefore, each tick of a 24-bit timeout counter represents a time period of between 178.8ns and 298.0ns (mid-point=238.5ns).

The following Table 20-14 shows examples of how to calculate the programmed value.

Table 20-14. Logical Layer Response Timeout Pre-scale value

Logical Layer Clock		Number of clock periods in one timeout tick (238.5/A, rounded to nearest integer) 'B'	Code to program into pre-scale register (B - 1)	Maximum timeout value programmable ($2^{24} * A * B$)	Period (ns) 'A'
Frequency (MHz)					
300	3.33	72	71 (0x47)	4.027s	
200	5	48	47 (0x2f)	4.027s	
100	10	24	23 (0x17)	4.027s	

20.7.2.5 GEN_PRTP***Port Response Timeout Pre-Scale.*****Address Offset:** 0x1001C

Bit#	Name	Description	Attribute	Reset
0:23		<i>Reserved</i>	RW	All 0's
24:31	TOPS	Pre-scale value for Port Response Timeout calculations (Default is for 200MHz host clock)	RW	0x2F

20.7.2.6 GEN_PRMR***Port Response Maximum Retries Register.*****Address Offset:** 0x10020

Bit#	Name	Description	Attribute	Reset
0:23		<i>Reserved</i>	RW	All 0's
24:31	MR	Sets the maximum number of Retry responses received by the source before aborting a MESSAGE or DOOBELL request: 0 = 0-retries (abort occurs on first Retry response received), 1 = 1-retry ... 255 = 255-retries	RW	0xFF

20.7.2.7 DEC_HSA_CSR**DMA Error Capture High Start Address CSR.****Address Offset:** 0x10030

Bit#	Name	Description	Attribute	Reset
0:31	HADDR	High-order Address. Most significant 32 bits of start address of errored	R	0x00000000

20.7.2.8 DEC_LSA_CSR**DMA Error Capture Low Start Address CSR.****Address Offset:** 0x10034

Bit#	Name	Description	Attribute	Reset
0:31	LADDR	Low-order Address. Least significant 32 bits of start address of errored transaction (m_haddr[31:0]).	R	0x00000000

20.7.2.9 DEC_INFO_CSR

DMA Error Capture Information Address CSR.

Address Offset: 0x10038

Bit#	Name	Description	Attribute	Reset
0:14	-	Reserved	R	All 0's
15	RNW	Read or Write indication: 0 = Write transaction, 1 = Read transaction	R	0b0
16:23	-	Reserved	R	All 0's
24:31	LEN	Length of errored transaction in bytes: 0 = 1 byte, 1 = 2 bytes ... 255 = 256 bytes	R	0x00

20.7.3 Physical Layer Registers**20.7.3.1 PL_TWM**

Physical Layer Transmit Watermark Register.

Address Offset: 0x10100

Bit#	Name	Description	Attribute	Reset
0:2	-	Reserved	RW	All 0's
3:7	WM0	Watermark Zero ^a	RW	0b00100
8:10		Reserved	RW	All 0's
11:15	WM1	Watermark One ¹	RW	0b00011
16:18		Reserved	RW	All 0's
19:23	WM2	Watermark Two ¹	RW	0b00010
24:26		Reserved	RW	All 0's
27:31	WM3	Watermark Three (non-standard) Will prevent any priority packet being transmitted if the target buffer status is less than WM3.	RW	0b00000

a. Function as defined in Part VI of the RapidIO Interconnect Specification, Revision 1.3

20.7.3.2 PL_CTRL

Physical Layer General Control Register.

Address Offset: 0x10104

Bit#	Name	Description	Attribute	Reset
0:10		Reserved	RW	All 0's
11	LOOP	Loopback enable. Causes the inward-facing loopback to be applied (in exactly the same way as of the LOOP_EN pin is asserted). 0 = normal mode, 1 = loopback mode Use for test purposes only. Note that when loop-backs are applied (either internally or externally), the RCHKD bit in GEN_CTRL CSR must be set else incoming Response packets will be rejected.	RW	0b0
12	LPAR	Lost Packet Accepted Recovery mode. Controls how the loss of a packet-accepted control symbol to the only in-flight packet is handled: 0 = causes port error (from which software recovery is required), 1 = allows auto recovery in hardware	RW	0b0
13	FIM	Fast Init Mode. For test/simulation purposes, scales down counter values used during initialization: 0 = normal, 1 = fast initialization	RW	0b0
14	TOS	Time-Out Scaling. Set the rate of the physical layer clock used to ensure physical layer timeouts are scaled correctly: 0 = 250MHz or 312.5MHz physical layer clock, 1 = 125MHz physical layer clock	RW	0b0
15	SYNC	Controls if Receiver should synchronize to far-end AckID after initialization: 0 = don't sync, 1 = sync	RW	0b0
16:18	-	Reserved	RW	All 0's
19:23	MAXLR	Controls the maximum number of times a Link-Request control symbol is retries before Port Error is declared: 0 = unlimited, 1 = 1 retry allowed ... 31 = 31 retries allowed	RW	0b00001
24:26	-	Reserved	RW	All 0's
27:31	MAXPR	Controls the maximum number of times a packet is retried before Output Error Stopped is declared: 0 = unlimited, 1 = 1 retry allowed ... 31 = 31 retries allowed	RW	0b00000

20.7.3.3 PL_STAT**Physical Layer Status Register.**

Address Offset: 0x10108

Bit#	Name	Description	Attribute	Reset
0:26		Reserved	R	All 0's
27:31	FEBS	Far-End-Buffer-Status received after link initialization. Note: Useful for setting Watermark registers.	R	0b00000

20.7.3.4 PL_CMD**Physical Layer Command Register.**

Address Offset: 0x1010C

Bit#	Name	Description	Attribute	Reset
0:29		Reserved	RW	All 0's
30:31	DOIT	Perform Command: 0 = Null (no command), 1 = Transmit Multicast-event control symbol 2 = Force re-initialization 3 = Reserved	RW	0b00

20.7.4 RapidIO to Host Address Translation Registers

These registers allow the offset address of incoming RapidIO NWRITE, NWRITE_R, SWRITE and NREAD transactions to be re-mapped before being used on the master Generic Host Interface, if desired. This also has the effect of controlling where in host memory incoming transactions are allowed to access, thus protecting all other memory regions from potential corruption. The minimum area allocated in host memory is 1-Mbyte (20 address bits).

The mask registers define which upper address bits are marked for replacement, and the Remap registers define what they are replaced with. The details of the translation depend upon the sizes of the incoming addresses and the required size of the outgoing address as follows:

RapidIO Address Size	Host Address Size	Mask bits used	Remap bits used	Notes
34	32	[32:43]	[32:43]	RapidIO address bits [0:1] ignored. Mask and Remap bits applied to top 12 bits of remaining 32 bits of address.
50	32	[32:43]	[32:43]	RapidIO address bits [0:17] ignored. Mask and Remap bits applied to top 12 bits of remaining 32 bits of address.
66	32	[32:43]	[32:43]	RapidIO address bits [0:33] ignored. Mask and Remap bits applied to top 12 bits of remaining 32 bits of address.
34	64	[30:43]	[0:43]	Mask bits applied to RapidIO address. Remap bits [0:29] appended to RapidIO address to form base 64-bit address, then Remap bits [30:43] used according to Mask.
50	64	[14:43]	[0:43]	Mask bits applied to RapidIO address. Remap bits [0:13] appended to RapidIO address to form base 64-bit address, then Remap bits [14:43] used according to Mask.
66	64	[0:43]	[0:43]	RapidIO address bits [0:1] ignored. Mask and Remap bits applied to top 44 bits of remaining 64 bits of address.

20.7.4.1 AT_LMASK

Address Translation Lower Mask Register.

Address Offset: 0x10200

Bit#	Name	Description	Attribute	Reset
0:11	LMASK	Lower 12-bits of the mask. Forms MASK[32:43]. A '1' indicates that the respective address bit should be replaced by a bit in the Remap register. Bits should only be set in this register if all bits are set in AT_HMASK, and all set bits should be justified to the MSB end of the word.	RW	0x000
12:31		Reserved	RW	All 0's

20.7.4.2 AT_HMASK

Address Translation Higher Mask Register.

Address Offset: 0x10204

Bit#	Name	Description	Attribute	Reset
0:31	HMASK	Higher 32-bits of the mask. Forms MASK[0:31]. A '1' indicates that the respective address bit should be placed by a bit in the Remap register. All set bits should be justified to the MSB end of the word.	RW	0x00000000

20.7.4.3 AT_LREMAP***Address Translation Lower Remap*** Register.

Address Offset: 0x10208

Bit#	Name	Description	Attribute	Reset
0:11	LREMAP	Lower 12-bits of the remap address. Forms REMAP[32:43].	RW	0x00000
12:31		Reserved	RW	All 0's

20.7.4.4 AT_HREMAP***Address Translation Higher Remap*** Register.

Address Offset: 0x1020C

Bit#	Name	Description	Attribute	Reset
0:31	HREMAP	Higher 32-bits of the remap address. Forms REMAP[0:31].	RW	0x00000

20.7.5 Error Injection Registers**20.7.5.1 EI_CS_CTRL1*****Control Symbol Error Injection Control Register 1***

This register, along with the Control Symbol Error Injection Mask Register, can be used to insert errors into Control Symbols before being transmitted. *They are intended for debug purposes only.* Errors can be inserted either once (single-shot mode) or continuously. Errors can be inserted into any Control Symbol(s) or just certain types; for instance, a facility is provided to search for Control Symbols containing certain Stype0 or Stype1 values.

Corruption is achieved by either XORing the 24-bit Control Symbol with a 24-bit mask, or by forcing a new 24-bit value over the existing one. The mask value is contained in the Control Symbol Error Injection Mask Register. Note that the Start of Control Symbol delimiter (SC) cannot be corrupted. Neither can errors be inserted after 8b/10b encoding.

Note that the CRC-5 field can be regenerated after corruption to avoid CRC-5 failures being detected at the other end of the link.

Address Offset: 0x10300

Bit#	Name	Description	Attribute	Reset
0:15		Reserved	RW	All 0's
16	ST1CMP	Stype1 Compare: 0 = Ignore generated stype1 field, 1 = Check generated stype1 field	RW	0b0
17:19	ST1MAT	Stype1 Match. When the Stype1 Compare bit is set, only generated control symbols whose stype1 field matches this value will be errored	RW	0b000
20	ST0CMP	Stype0 Compare: 0 = Ignore generated stype0 field, 1 = Check generated stype0 field	RW	0b0

Bit#	Name	Description	Attribute	Reset
21:23	ST0MAT	Stype0 Match. When the Stype0 Compare bit is set, only generated control symbols whose stype0 field matches this value will be errored	RW	0b000
24:27	-	Reserved	RW	All 0's
28	RPLCE	Controls how Mask value is used: 0 = XOR generated control symbol with mask value, 1 = Replace generated control symbol with mask value	RW	0b0
29	SS	Controls if error insertion is continuous or single shot: 0 = Continually error the targeted control symbols, 1 = Error the first targeted control symbol only. Re-writing a '1' to this bit (after the previous error insertion has occurred) will cause the error to be inserted again.	RW	0b0
30	CRC	Controls if CRC-5 is recalculated after error insertion: 0 = CRC-5 is not recalculated, 1 = CRC-5 is recalculated	RW	0b0
31	EN	Controls insertion of control symbol error: 0 = Disable error Injection, 1 = Enable error injection	RW	0b0

20.7.5.2 EI_CS_CTRL2***Control Symbol Error Injection Control Register 2.***

Error Injection Control register 2 for control symbols.

Address Offset: 0x10304

Bit#	Name	Description	Attribute	Reset
0:7		Reserved	RW	All 0's
8:31	MASK	Control Symbol Mask	RW	0x0000000

20.7.5.3 EI_PKT_CTRL1***Packet Error Injection Control Register 1***

This register, along with the Packet Error Injection Control and Mask Register, can be used to inject errors into packets for debug purposes. Errors can be inserted either once (single-shot mode) or continuously. Errors can be inserted into any packet(s) or only those which successfully match certain search criteria.

The search facility provided allows the user to compare a single numbered byte within a packet (via the "Byte number search" field, where '0' indicates the first byte, '1' indicates the second byte and so on) against a particular value (stored in "Byte Match"). Note that the "Byte compare" field indicates which bits within the byte are used in the comparison. If a match is found then the user is allowed to corrupt a numbered byte (via the "Byte number replace" field) by XORing its value with an 8-bit mask ("Byte Mask").

Note that if the search facility is used ("Byte compare" non-zero), then the byte to be corrupted must either be within the 32-bit word that contained the byte being searched for, or in a subsequent word ("Byte number replace"[8:2] >= "Byte number search"[8:2]).

The final CRC-16 field can (optionally) be regenerated after corruption to avoid CRC-16 failures being detected at the other end of the link. If a mid-packet CRC-16 field is present it is always recalculated – note this may prevent changes within the first 80 bytes of the packet being detected at the final CRC-16 even if that CRC is not regenerated.

Address Offset: 0x10308

Bit#	Name	Description	Attribute	Reset
0:7	BMAT	Byte Match	RW	0x00
8:14		<i>Reserved</i>	RW	All 0's
15:23	BSRCH	Byte number to search on	RW	0x000
24:28	-	<i>Reserved</i>	RW	All 0's
29	SS	Controls if error insertion is continuous or single shot: 0 = Continually error the targeted packets, 1 = Error the first targeted packet only. Re-writing a '1' to this bit (after the previous error insertion has occurred) will cause the error to be inserted again.	RW	0b0
30	CRC	Controls if CRC-16 is recalculated after error insertion: 0 = CRC-16 is not recalculated, 1 = CRC-16 is recalculated	RW	0b0
31	EN	Controls insertion of packet errors: 0 = Disable error Injection, 1 = Enable error injection	RW	0b0

20.7.5.4 EI_PKT_CTRL2***Packet Error-Injection and Mask Control Register 2***

Address Offset: 0x1030C

Bit#	Name	Description	Attribute	Reset
0:7	BCMP	Byte Compare. On a per bit basis controls the comparison of the generated byte located using the BSRCH field, against the BMAT field in the EI_PKT_CTRL1 register: 0 = Do not compare, 1 = Compare	RW	0x00
8:14		<i>Reserved</i>	RW	All 0's
15:23	BNUM	Byte number within a packet to error	RW	0x000
24:31	MASK	Byte mask	RW	0x00

20.7.6 Transaction Queues Registers

There are two Transaction Queues supported. Each Transaction Queue is a circular structure situated in shared host memory and is defined using four pointers, all of which must be aligned to double-word boundaries:

- The Start and End pointers define where the queue is situated in memory. Both indicate the address of the first double-word of a potential queue entry, the Start pointer indicating the start of the circular structure and End pointer the end of it. The gap between them determines the maximum number of entries possible for the queue.
- The Tail and Head pointers define where entries are added (by the host) and removed (by the SRIO controller) respectively. The gap between them (taking into account wraps) determines the number of entries currently on the queue.

A set of five 32-bit registers define the pointers for each queue. All the pointers are 64-bit qualities which share a common upper 32-bits defined by the register TQn_UPTR (this need not be used if using a 32-bit address host). Each then has its own register to define the lower 32-bits. All four pointers are initialized to 0 by the SRIO controller following a device reset. Each queue must then be configured by the host before use, which it does by:

- Writing double-word aligned addresses into both the Start and End pointer registers, ensuring that the End pointer is at a higher address than the Start pointer, and that the minimum number of entries is four. Note that as each entry consists of six words (See “[Transaction Queues](#)” on page 990 for details of format), care is needed in selecting values for these pointers.
- Writing a ‘1’ to the appropriate configuration bit in the TQ_CTRL1 register. This causes the SRIO controller to copy the Start pointer value to both the Head and Tail pointer registers and the queue fill level indicators to begin operating. The Start and End pointers must not be changed once a queue has been configured.

At this point, the queue is empty and ready for the host to add entries. To add entries, the host should write the required 6-word entry to shared host memory starting at the respective Tail pointer address. It should then update the Tail pointer to point to the next legal entry position, taking into account any wrap required when the End pointer is reached. If the updated Tail pointer is equal to the Head pointer, the queue is full and no more entries must be added (otherwise queue corruption will result). Multiple entries can be added at once if desired, but it is the hosts responsibility to ensure that the queue does not overflow.

Per-queue Full flags are implemented (FULL in register TQn_TPTR) to allow the host to differentiate between the full and empty conditions when the Head and Tail pointers are equal. In addition, to aid in managing the queues, a per-queue Not Nearly Full flag is implemented (NNF in register TQ_STAT). The host can set the threshold at which this operates (using NFT in register TQ_CTRL1). Interrupts can be generated if desired when the Not Nearly Full condition is true (indicating more entries can be added).

To cater for situations where multiple hosts may want to add entries to the same Transaction Queue, a LOCK indication bit is implemented in the Tail pointer register. This is automatically set by the SRIO controller whenever the Tail pointer register is read by a host, and cleared whenever the Tail pointer register is written (regardless of whether the LADDR value is changed or not). Thus when a host wants to add entries, it should read the Tail pointer, ensure the LOCK indication bit is clear, add one or more entries to the indicated position in shared host memory, and then write back the updated Tail pointer (with LOCK bit position set to ‘0’). If the queue is empty, it is possible to go straight to the full condition by writing the Tail pointer with an unchanged value.

If the LOCK bit is set when the Tail pointer is read, this indicates another host is currently adding an entry. The locked-out host should regularly poll the register until the LOCK bit is clear. Note that the FULL flag is co-located with LOCK so that when a host can proceed, it can immediately tell if there is space to add an entry or not. Note that no protection of registers occurs when the LOCK bit is active; it is simply a bit to aid the host when adding entries.

If a host reads the Tail pointer register (thus setting the LOCK bit) and wishes to clear the lock without adding any entries, it should write to the Tail pointer register with the LOCK bit position set to ‘1’ (the lock is then cleared without altering the LADDR value).

When enabled by the EN bit in register TQ_CTRL1, the SRIO controller scans all supported and configured transaction queues and arbitrates between them based on priority and status of internal resources. TQ0 is the highest priority queue.

A Status Queue entry is normally written whenever a transaction completes. However, it is possible to suppress these writes on a per-transaction queue basis by setting the respective STQS bits in the TQ_CTRL2 register. This feature may be useful when the user wants to perform a regular set of transactions to or from the same locations in host memory. However, note that for read requests, such suppression means that the SRIO controller will give no indication of when the requested data has arrived.

TQn_SPTR***Transaction Queue n Start Pointer***

Address Offset: 0x20000 or 0x20001

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Forms the lower 32 bits of pointer when 0b000 added as lowest 3 bits (to ensure double-word alignment)	RW	0x00000000
29:31		Reserved	RW	All 0's

TQn_EPTR***Transaction Queue n End Pointer.***

Address Offset: 0x20004 or 0x20005

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Forms the lower 32 bits of pointer when '000' added as lowest 3 bits (to ensure double-word alignment)	RW	0x00000000
29:31		Reserved	RW	All 0's

TQn_HPTR***Transaction Queue n Head Pointer.***

Address Offset: 0x20008 or 0x20009

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Forms the lower 32 bits of pointer when '000' added as lowest 3 bits (to ensure double-word alignment)	R	0x00000000
29:31		Reserved	R	All 0's

TQn_TPTR***Transaction Queue n Tail Pointer.***

Address Offset: 0x2000C or 0x2000D

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Forms the lower 32 bits of pointer when '000' added as lowest 3 bits (to ensure double-word alignment). This value only changes when the written value has the LOCK bit position set to '0'.	RW	0x00000000
29	-	Reserved	RW	0
30	FULL	Indicates if the queue is Full or not (note that this is a read-only bit): 0 = queue is not full, 1 = queue is full	RW	0
31	LOCK	Indicates queue locked by a host. Note that this bit does not act as a simple read/write bit; a read from the register sets this bit, and a write clears it (regardless of the value written): 0 = queue not locked, 1 = queue locked by another host	RW	0

20.7.6.5 TQn_UPTR***Transaction Queue n Upper Pointers.***

Address Offset: 0x20010 or 0x20011

Bit#	Name	Description	Attribute	Reset
0:31	HADDR	Upper 32 bits of all 4 pointers (ignored if using a 32-bit address host).	RW	0x00000000

20.7.6.6 TQ_CTRL1***Transaction Queues Control Register1.***

Address Offset: 0x20200

Bit#	Name	Description	Attribute	Reset
0:13	-	Reserved	RW	All 0's
14:15	CONF	Per queue Configure (bit[15] relates to TQ0, bit[14] to TQ1). Writing a '1' to a bit causes the respective Transaction Queue 'n' to be configured ready for use.	RW	0x0000
16:19	NFT	Nearly Full Threshold. Sets the number of remaining free entries in a queue at which the Not Nearly Full flags (NNF bits in the TQ_STAT register) are asserted and negated: 0 = 0 (i.e., flags become queue not full), 0x1 = 1 0x2 = 2 0x3 = 4 0x4 = 8 others = Reserved	RW	0x0
20	EN	Transaction Queue Enable. Allows the processing of entries on all configured transaction queues: 0 = disabled, 1 = enabled	RW	0b0
21:22	PRIOR	Priority scheme to use for the Transaction Queues: 00 = ramped (TQ0 highest, then TQ1 etc.) 01 = equal (round-robin arbitration used) 10 = TQ0 High priority, TQ1 to TQn equal priority	RW	0b00
23:31	-	Reserved	RW	All 0's

20.7.6.7 TQ_CTRL2***Transaction Queues Control Register2***

Address Offset: 0x20204

Bit#	Name	Description	Attribute	Reset
0:13	-	Reserved	RW	All 0's
14:15	STQS	Per queue Status Queue Entry Suppression (bit[15] relates to TQ0, bit[14] to TQ1). Writing a '1' to a bit causes the respective Transaction Queue 'n' to suppress the generation of any status entries, whenever it completes a transaction.	RW	0x0000
2:31	-	Reserved	RW	All 0's

20.7.6.8 TQ_STAT***Transaction Queues Status Register.***

Address Offset: 0x20208

Bit#	Name	Description	Attribute	Reset
0:13	-	Reserved	RW	All 0's
14:15	INEE	Per queue IN bit Encountered Event (bit[15] relates to TQ0, bit[14] to TQ1). A bit is set when the IN bit is set in an entry on the respective Transaction Queue.	RWTC	0x0000
16:31	NNF	Per-queue Not Nearly Full flags (bit[31] relates to TQ0, bit[30] to TQ1 etc). The threshold at which not nearly full is declared is controlled by the NFT field in the TQ_CTRL1 register: 0 = queue is nearly full (free entries below threshold), 1 = queue is not nearly full Only bits[32-NTQ] to [31] are implemented; any remaining bits are Reserved.	R	0x0000

20.7.6.9 TQ_IER***Transaction Queues Interrupt Enable Register.***

The TQ bit set in the MISR register is set (from which an interrupt can be generated if desired) whenever a bit in this register and the corresponding bit in the TQ_STAT register are both set.

Address Offset: 0x2020C

Bit#	Name	Description	Attribute	Reset
0:15	INEE	Per queue IN bit Encountered Event interrupt enable	RW	0x0000
16:31	NNF	Per queue Not Nearly Full interrupt enable	RW	0x0000

20.7.7 Status Queue

Each Status Queue is a circular structure situated in shared host memory and is defined using four pointers, all of which must be aligned to 32-Byte boundaries:

- The Start and End pointers define where the queue is situated in memory. Both indicate the address of the first 32 Bytes of a potential queue entry; the Start pointer indicating the start of the circular structure and End pointer the end of it. The gap between them determines the maximum number of entries possible for the queue.
- The Tail and Head pointers define where entries are added (by the SRIO controller) and removed (by the host) respectively. The gap between them (taking into account wraps) determines the number of entries currently on the queue.

A set of five 32-bit registers define the pointers for the queue. All the pointers are 64-bit quantities which share a common upper 32-bits defined by the register SQn_UPTR (this need not be used if using a 32-bit address host). Each then has its own register to define the lower 32-bits. All four pointers are initialized to 0 by the SRIO controller following a device reset. Each queue must then be configured by the host before use, which it does by:

- Ensuring that the End pointer is at a higher address than the Start pointer, and that the minimum number of entries (N) is four. Note that as each entry consists of eight words.
- Writing a '1' to the configuration bit in the SQ_CTRL Register. This causes the SRIO controller to copy the Start pointer value to both the Head and Tail pointer registers and the queue fill level indicators to begin operating. The Start and End pointers must not be changed once a queue has been configured.

At this point, the queue is empty and ready for the SRIO controller to add entries. An Empty flag is implemented (EMPTY in register SQn_HPTR) to allow the host to differentiate between the full and empty conditions when the Head and Tail pointers are equal. In addition, to aid in managing the queue, a Not Nearly Empty flag is implemented (NNE in register SQ_STAT). The host can set the threshold at which this operates (using NET in register SQ_CTRL). An interrupt can be generated if desired when the Not Nearly Empty condition is true. This is designed to relieve the burden of processing the Status Queue by allowing the host to program the device so that multiple status entries can be processed on each visit.

To program a Status Queue allocate a memory space for (N) entries aligned to 32-Byte boundary. Shift the start of the memory space to the right by one bit and write it to the SQn_SPTR register. Add $(N - 1) * 32$ to the allocated memory space and shift it to the right by one bit, and write this value to the SQn_EPTR register.

To access the Status-Queue entries:

1. Read the LADDR field of the SQn_SPTR register
2. Left shift the address by one bit
3. Use the resulting address to access the memory

To remove entries, the host should read the 8-word entry in shared host memory starting at the Head pointer address. It should then update the Head pointer by incrementing by one to point to the next legal entry position, taking into account any wrap required when the End pointer is reached. If the updated Head pointer is equal to the Tail pointer, the queue is empty and the Head pointer must not be advanced any further (otherwise queue corruption will result). The host must service the Status Queue periodically or else it could fill. If the queue is allowed to become full, then no more Transactions will be acted upon and Source processing will halt. In order to inform the host that this has occurred, a Full flag is generated (FULL in register SQ_STAT) from which an interrupt can be generated. If the queue is full, it is possible to go straight to the empty condition by writing the Head pointer with an unchanged value.

If the SRIO controller needs to write a status queue entry but the respective status queue is not configured, it sets the Full flag for the respective queue as an indication of the user error. Processing then halts until the queue is configured.

Each Transaction Queue has its own Status Queue. Shared Status Queues are not supported.

20.7.7.1 SQn_SPTR

Status Queue n Start Pointer

Address Offset: 0x20400 or 0x20401

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Forms the lower 32 bits of pointer when 0b000 added as lowest 3 bits (to ensure double-word alignment)	RW	0x00000000
29:31		Reserved	RW	All 0's

20.7.7.2 SQn_EPTR

Status Queue n End pointer

Address Offset: 0x20404 or 0x20405

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Forms the lower 32 bits of pointer when 0b000 added as lowest 3 bits (to ensure double-word alignment)	RW	0x00000000
29:31		Reserved	RW	All 0's

20.7.7.3 SQn_HPTR

Status Queue n Head Pointer

Address Offset: 0x20408 or 0x20409

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Forms the lower 32 bits of pointer when 0b000 added as lowest 3 bits (to ensure double-word alignment)	RW	0x00000000
29	-	Reserved	RW	0
30	EMPTY	Indicates if the queue is Empty or not: 0 = queue is not empty 1 = queue is empty	RW	0
31	-	Reserved	RW	0

20.7.7.4 SQn_TPTR

Status Queue n Tail Pointer

Address Offset: 0x2040C or 0x2040D

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Forms the lower 32 bits of pointer when 0b000 added as lowest 3 bits (to ensure double-word alignment)	R	0x00000000
29:31		Reserved	R	All 0's

20.7.7.5 SQn_UPTR***Status Queue n Upper Pointer***

Address Offset: 0x20410 or 0x20411

Bit#	Name	Description	Attribute	Reset
0:31	HADDR	Upper 32 bits of all 4 pointers (ignored if using a 32-bit address host).	RW	0x00000000

20.7.7.6 SQ_CTRL***Status Queue Control Register***

Address Offset: 0x10600

Bit#	Name	Description	Attribute	Reset
0:13	-	Reserved	RW	All 0's
14:15	CONF	Per queue Configure (bit[15] relates to SQ0, bit[14] to SQ1). Writing a '1' to a bit causes the respective Status Queue 'n' to be configured ready for use.	RW	0
16:19	NET	Nearly Empty Threshold. Sets the number of entries in the queue at which the Not Nearly Empty flag (NNE bit in the SQ_STAT register) is asserted and negated: 0x0 = 0 (i.e., flag becomes queue not empty), 0x1 = 1 0x2 = 2 0x3 = 4 0x4 = 8 others = Reserved	RW	00
20:31	-	Reserved	RW	All 0's

20.7.7.7 SQ_STAT***Status Queue Status Register***

Address Offset: 0x20608

Bit#	Name	Description	Attribute	Reset
0:13	-	Reserved	RW	All 0's
14:15	FULL	Full flag. (bit[15] relates to SQ0, bit[14] to SQ1). Set when the Status Queue is full. This is a serious condition which the host must quickly rectify by reading entries and updating the Head pointer. Also set if any status queue entry wants to be written, but the respective Status Queue is not configured.	R	00
16:29	-	Reserved	RW	All 0's
30:31	NNE	Not Nearly Empty flag. (bit[31] relates to SQ0, bit[14] to SQ1). Indicates when the fill level of the Status Queue is nearly empty. The threshold at which not nearly empty is declared is controlled by the NET field in the SQ_CTRL register: 0 = queue is nearly empty (fill level below threshold), 1 = queue is not nearly empty.	R	00

20.7.7.8 SQ_IER***Status Queue Interrupt Enable Register***

The SQ bit set in the MISR register is set (from which an interrupt can be generated if desired) whenever a bit in this register and the corresponding bit in the SQ_STAT register are both set.

Address Offset: 0x2060C

Bit#	Name	Description	Attribute	Reset
0:15	FULL	Full interrupt enable (bit[15] relates to SQ0, bit[14] to SQ1 etc).	RW	0x0000
16:31	NNE	Not Nearly Empty interrupt enable (bit[31] relates to SQ0, bit[14] to SQ1 etc).	RW	0x0000

20.7.8 DOORBELL FIFO

DOORBELL Messages received by the SRIO controller are processed and held in an internal FIFO which has space for up to eight DOORBELL messages. The Source ID and INFO fields are stored for each message. Messages that are accepted cause a DONE RESPONSE to be generated. The host reads entries off the FIFO by reading the DF_RP register.

To aid in managing the queue, a Not-Nearly-Empty flag is implemented (NNE in register DF_STAT). The host can set the threshold at which this operates (using NET in register DF_CTRL). An interrupt can be generated if desired when the Not Nearly Empty condition is true. This is designed to relieve the burden of processing the FIFO by allowing the host to program the device so that multiple doorbell messages can be processed on each visit if desired. In addition, a count of the number of messages available to be read in the FIFO is made available in the NUM field of the DF_INFO register. If a read of DF_RP is performed when the number of available messages is zero, invalid data will be read.

The host must service the FIFO periodically or else it could fill. If the FIFO is allowed to become full, then incoming DOORBELL messages can not be processed (RETRY RESPONSES being returned to the sender). In order to inform the host that this has occurred, a Full flag is generated (FULL in register DF_STAT) from which an interrupt can be generated.

20.7.8.1 DF_CTRL***Doorbell FIFO Control Register***

Address Offset: 0x20800

Bit#	Name	Description	Attribute	Reset
0:15	-	Reserved	RW	All 0's
16:19	NET	Nearly Empty Threshold. Sets the number of entries in a queue at which the Not Nearly Empty flag (NNE bit in the DF_STAT register) is asserted and negated: 0x0 = 0 (i.e., flag becomes queue not empty) 0x1 = 1 0x2 = 2 0x3 = 4 0x4 = 8 others = Reserved	RW	0x0
20:31	-	Reserved	RW	All 0's

20.7.8.2 DF_INFO***Doorbell FIFO Information Register***

Address Offset: 0x20804

Bit#	Name	Description	Attribute	Reset
0:11	-	<i>Reserved</i>	R	All 0's
12:15	NUM	Number of DOORBELL Messages currently available in the FIFO (0x0 - 0x8, where 0x0 = none).	R	0x0
16:31	-	<i>Reserved</i>	R	All 0's

20.7.8.3 DF_STAT***Doorbell FIFO Status Register***

Address Offset: 0x20808

Bit#	Name	Description	Attribute	Reset
0:14	-	<i>Reserved</i>	R	All 0's
15	FULL	Full flag. Set when the FIFO is full. This is a serious condition which the host must quickly rectify by reading entries.	R	0b0
16:30	-	<i>Reserved</i>	R	All 0's
31	NNE	Not-Nearly-Empty flag. Indicates when the fill level of the FIFO is nearly empty. The threshold at which not nearly empty is declared is controlled by the NET field in the DF_CTRL register: 0 = FIFO is nearly empty (fill level below threshold) 1 = FIFO is not nearly empty	R	0b0

20.7.8.4 DF_IER***Doorbell FIFO Interrupt Enable Register***

The DF bit set in the MISR register is set (from which an interrupt can be generated if desired) whenever a bit in this register and the corresponding bit in the DF_STAT register are both set.

Address Offset: 0x2080C

Bit#	Name	Description	Attribute	Reset
0:14	-	<i>Reserved</i>	RW	All 0's
15	FULL	Full interrupt enable	RW	0b0
16:30	-	<i>Reserved</i>	RW	All 0's
31	NNE	Not-Nearly-Empty interrupt enable	RW	0b0

20.7.8.5 DF_RP***Doorbell FIFO Read Port***

A read of this register causes the entry at the head of the FIFO, if available, to be presented.

Address Offset: 0x20810

Bit#	Name	Description	Attribute	Reset
0:7	HSID	Upper 8 bits of the Source ID from the RapidIO packet. Only valid for 16-bit device addressing; for 8-bit device addressing this is set to 0x00.	R	0x00
8:15	LSID	Lower 8 bits of the Source ID from the RapidIO packet.	R	0x00
16:23	HINFO	Info Byte (MSB).	R	0x00
24:31	LINFO	Info Byte (LSB).	R	0x00

20.7.9 Port-write FIFO

Port-writes received by the SRIO controller are processed and held in an internal FIFO. The Port-write FIFO has storage for a single 64-byte Port-write or several smaller transfers.

The PW_INFO register allows the host to determine the number of separate Port-write transactions in the FIFO, and the size and source ID of the transaction at the head of the FIFO. The Port-write data payload is retrieved by the host repeatedly reading the PW_RP register. The WORDS status information counts down while this is performed in order to indicate the number of remaining words for the current transaction. As the last word is read, the NUM field reduces by 1, and, if any other transactions are queued, the WORDS field changes to indicate the number of words in the next transaction.

To aid in managing the queue, a FIFO Not Empty flag (NE in register PW_STAT) is generated whenever the FIFO contains one or more entries, from which an interrupt can be generated if desired.

If on receiving the Port-write transaction, the SRIO controller determines that the transaction will not fit into the Port-write FIFO, the transaction is dropped and the OFLWE event set in the PW_STAT register. The host should read the FIFO as soon as possible after a transaction arriving to minimize the probability of this happening.

20.7.9.1 PW_INFO***Port-Write Information register***

Address Offset: 0x20904

Bit#	Name	Description	Attribute	Reset
0:7	-	Reserved	R	All 0's
8:11	WORDS	Number of Port-write words remaining for transaction at head of FIFO (0-15, where 0 = one)	R	0x0
12:15	NUM	Number of received Port-write transfers currently available in the FIFO (0-15, where 0 = none)	R	0x0

Bit#	Name	Description	Attribute	Reset
16:23	HSID	Upper 8 bits of the Source ID from the RapidIO packet. Only valid for 16-bit device addressing; for 8-bit device addressing this is set to 0x00.	R	0x00
24:31	LSID	Lower 8 bits of the Source ID from the RapidIO packet.	R	0x00

20.7.9.2 PW_STAT**Port-write FIFO Status Register**

Address Offset: 0x20908

Bit#	Name	Description	Attribute	Reset
0:14	-	Reserved	-	All 0's
15	OFLWE	Overflow Event. Set when a transaction arrives which not fit in the FIFO.	RWTC	0b0
16:30	-	Reserved	-	All 0's
31	NE	Not Empty flag. Set when there are one or more Port-writes available to read.	R	0b0

20.7.9.3 PW_IER**Port-Write FIFO Interrupt Enable Register**

The PW bit set in the MISR register is set (from which an interrupt can be generated if desired) whenever a bit in this register and the corresponding bit in the PW_STAT register are both set.

Address Offset: 0x2090C

Bit#	Name	Description	Attribute	Reset
0:14	-	Reserved	RW	All 0's
15	OFLWE	Overflow Event interrupt enable	RW	0b0
16:30	-	Reserved	RW	All 0's
31	NE	Not Empty interrupt enable	RW	0b0

20.7.9.4 PW_RP**Port-Write FIFO Read Port register**

A read of this register causes the entry at the head of the FIFO, if available, to be presented.

Address Offset: 0x20910

Bit#	Name	Description	Attribute	Reset
0:7	BYTE0	Payload Byte (n)	R	0x00
8:15	BYTE1	Payload Byte (n+1)	R	0x00
16:23	BYTE2	Payload Byte (n+2)	R	0x00
24:31	BYTE3	Payload Byte (n+3)	R	0x00

20.7.10 Free-Lists

The maximum number of Free-List queues supported is equal to 4. In these descriptions, the letter 'n' relates to the queue number, and can range 0-to-3. The number of Free-Lists equals the number of Mailboxes supported (NMQ), which is limited to 4.

Each Free-List queue is a circular structure situated in shared host memory and is defined using four pointers, all of which must be aligned to double-word boundaries:

- The Start and End pointers define where the queue is situated in memory. Both indicate the address of the first double-word of a potential queue entry, the Start pointer indicating the start of the circular structure and End pointer the end of it. The gap between them determines the maximum number of entries possible for the queue.
- The Tail and Head pointers define where entries are added (by the host) and removed (by the SRIO controller) respectively. The gap between them (taking into account wraps) determines the number of entries currently on the queue.

A set of five 32-bit registers define the pointers for each queue. All the pointers are 64-bit qualities which share a common upper 32-bits defined by the register FL_n_UPTR (this need not be used if using a 32-bit address host). Each then has its own register to define the lower 32-bits. All four pointers are initialized to '0' by the SRIO controller following a device reset. Each queue must then be configured by the host before use, which it does by:

- Writing double-word aligned addresses into both the Start and End pointer registers, ensuring that the End pointer is at a higher address than the Start pointer, and that the minimum number of entries is four. Note that as each entry consists of two words.
- Writing the size of buffers which will be placed on the queue into the FL_n_BUF register.
- Writing a '1' to the appropriate configuration bit in the FL_CTRL register. This causes the SRIO controller to copy the Start pointer value to both the Head and Tail pointer registers and the queue fill level indicators to begin operating. The Start and End pointers must not be changed once a queue has been configured.

At this point, the queue is empty and ready for the host to add entries. To add entries, the host should write the required 2-word entry to shared host memory starting at the respective Tail pointer address. It should then update the Tail pointer to point to the next legal entry position, taking into account any wrap required when the End pointer is reached. If the updated Tail pointer is equal to the Head pointer, the queue is full and no more entries must be added (otherwise queue corruption will result). Multiple entries can be added at once if desired, but it is the hosts responsibility to ensure that the queue does not overflow.

Per-queue Full flags are implemented (FULL in register FL_n_TPTR) to allow the host to differentiate between the full and empty conditions when the Head and Tail pointers are equal. In addition, to aid in managing the queues, a per-queue Nearly Empty flag is implemented (NE in register FL_STAT). The host can set the threshold at which this operates (using NET in register FL_CTRL). An interrupt can be generated if desired when the Nearly Empty condition is true.

The host must replenish entries on the Free-Lists periodically or else they could empty. If a queue is allowed to become empty, then incoming MESSAGE transactions cannot be processed (RETRY RESPONSES being returned to the sender). In order to inform the host that this has occurred, an Empty flag is generated (EMPTY in register FL_STAT) from which an interrupt can be generated. Any MESSAGE transactions received before the required Free-List has been configured will be sent an ERROR RESPONSE.

To cater for situations where multiple hosts may want to add entries to the same Free-List, a LOCK indication bit is implemented in the Tail pointer register. This is automatically set by the SRIO controller whenever the Tail pointer is read by the host, and cleared whenever the Tail pointer is written (regardless of whether the LADDR value is changed or not). Thus when a host wants to add entries, it should read the Tail pointer, ensure the LOCK indication bit is clear, add 1 or more entries to the indicated position in shared host memory, and then write back the

updated Tail pointer (with LOCK bit position set to '0'). If the queue is empty, it is possible to go straight to the full condition by writing the Tail pointer with an unchanged value.

If the LOCK bit is set when the Tail pointer is read, this indicates another host is currently adding an entry. The locked-out host should regularly poll the register until the LOCK bit is clear. Note that a FULL flag is co-located with LOCK so that when a host can proceed, it can immediately tell if there is space to add an entry or not. Also note that no protection of registers occurs when the LOCK bit is active; it is simply a bit to aid the host when adding entries.

If a host reads the Tail pointer register (thus setting the LOCK bit) and wishes to clear the lock without adding any entries, it should write to the Tail pointer register with the LOCK bit position set to '1' (the lock is then cleared without altering the LADDR value).

20.7.10.1 FLn_SPTR

Free-List n Start Pointer

Address Offset: 0x20A00 to (0x20A00 + 0x20*n)

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Forms the lower 32 bits of pointer when 0b000 added as lowest 3 bits (to ensure double-word alignment)	RW	0x00000000
29:31	-	Reserved	RW	All 0's

20.7.10.2 FLn_EPTR

Free-List n End Pointer

Address Offset: 0x20A04 to (0x20A04 + 0x20*n)

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Forms the lower 32 bits of pointer when 0b000 added as lowest 3 bits (to ensure double-word alignment)	RW	0x00000000
29:31	-	Reserved	RW	All 0's

20.7.10.3 FLn_HPTR

Free-List n Head pointer

Address Offset: 0x20A08 to (0x20A08 + 0x20*n)

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Forms the lower 32 bits of pointer when 0b000 added as lowest 3 bits (to ensure double-word alignment)		0x00000000
29:31	-	Reserved		All 0's

20.7.10.4 FLn_TPTR***Free-List n Tail Pointer***

Address Offset: 0x20A0C to (0x20A0C + 0x20*n)

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Forms the lower 32 bits of pointer when 0b000 added as lowest 3 bits (to ensure double-word alignment). This value only changes when the written value has the LOCK bit position set to '0'.	RW	0x00000000
29	-	Reserved	RW	0b0
30	FULL	Indicates if the list is Full or not (note that this is a read-only bit): 0 = queue is not full 1 = queue is full	RW	0b0
31	LOCK	Indicates list locked by a host. Note that this bit does not act as a simple read/write bit; a read from the register sets this bit, and a write clears it (regardless of the value written): 0 = queue not locked 1 = queue locked by another host	RW	0b0

20.7.10.5 FLn_UPTR***Free-List n Upper Pointer***

Common High portion of the pointers.

Address Offset: 0x20A10 to (0x20A10 + 0x20*n)

Bit#	Name	Description	Attribute	Reset
0:31	HADDR	Upper 32 bits of all 4 pointers (ignored if using a 32-bit address host).	RW	0x00000000

20.7.10.6 FLn_BUF***Free-List n Buffer Size***

Reception of a MESSAGE which is larger than the buffer size results in an ERROR RESPONSE being returned to the sender, the declaration of a Message Format Error and the generation of an errored status entry on the respective mailbox queue.

Address Offset: 0x20A14 to (0x20A14 + 0x20*n)

Bit#	Name	Description	Attribute	Reset
0:19		Reserved	RW	All 0's
20:28	SIZE	Size of all buffers on Free-List, in double-words:'0' = one double-word (8 bytes) '1' = two double-words (16-bytes) ... 511 = 512 double-words (4096 bytes)	RW	0x000
29:31		Reserved	RW	All 0's

20.7.10.7 FL_CTRL***Free-Lists Control Register***

Address Offset: 0x20C00

Bit#	Name	Description	Attribute	Reset
0:11	-	<i>Reserved</i>	RW	All 0's
12:15	CONF	Per queue Configure (bit[15] relates to FL0, bit[14] to FL1 etc.) Writing a '1' to a bit causes the respective queue to be configured ready for use.	RW	0x0
16:19	NET	Nearly Empty Threshold. Sets the number of entries in a queue at which the Nearly Empty flag (NE bit in the FL_STAT register) is asserted and negated: 0 = 0 (i.e., flag becomes queue empty) 1 = 1 2 = 2 3 = 4 4 = 8 others = <i>Reserved</i>	RW	0x0
20:31	-	<i>Reserved</i>	RW	All 0's

20.7.10.8 FL_STAT***Free-Lists Status Register***

Address Offset: 0x20C08

Bit#	Name	Description	Attribute	Reset
0:11	-	<i>Reserved</i>	RW	All 0's
12:15	EMPTY	Per queue Empty (bit[15] relates to FL0, and bit[14] to FL1, etc.) Set when the respective Free-List is empty. This is a serious condition which the host must quickly rectify by adding new entries.	R	0x0
16:27	-	<i>Reserved</i>	RW	All 0's
28:31	NE	Per queue Nearly Empty flag (bit[31] relates to FL0, and bit[30] to FL1 etc.) The threshold at which nearly empty is declared is controlled by the NET field in the FL_CTRL register: 0 = queue is not nearly empty (fill level above threshold) 1 = queue is nearly empty	R	0x0

20.7.10.9 FL_IER***Free-Lists Interrupt Enable Register***

The FL bit set in the MISR register is set (from which an interrupt can be generated if desired) whenever a bit in this register and the corresponding bit in the FL_STAT register are both set.

Address Offset: 0x20C0C

Bit#	Name	Description	Attribute	Reset
0:11	-	<i>Reserved</i>	RW	All 0's
12:15	EMPTY	Per queue Empty interrupt enables (bit[15] relates to FL0, and bit[14] to FL1 etc).	RW	0x0000
16:27	-	<i>Reserved</i>	RW	All 0's
28:31	NE	Per queue Nearly Empty interrupt enables (bit[31] relates to FL0, and bit[30] to FL1 etc).	RW	0x0000

20.7.11 Mailbox Queues

The number of Mailbox Queues supported is equal to the parameter NMQ. In these descriptions, the letter 'n' relates to the queue number, and can range **n = 0-to-3**.

Each Mailbox Queue is a circular structure situated in shared host memory and is defined using four pointers, all of which must be aligned to 32-Byte boundaries:

- The Start and End pointers define where the queue is situated in memory. Both indicate the address of the first 32-Bytes of a potential queue entry, the Start pointer indicating the start of the circular structure and End pointer the end of it. The gap between them determines the maximum number of entries possible for the queue.
- The Tail and Head pointers define where entries are added (by the SRIO controller) and removed (by the host) respectively. The gap between them (taking into account wraps) determines the number of entries currently on the queue.

A set of five 32-bit registers define the pointers for each queue. All the pointers are 64-bit qualities which share a common upper 32-bits defined by the register MQn_UPTR (this need not be used if using a 32-bit address host). Each then has its own register to define the lower 32-bits. All four pointers are initialized to 0 by the SRIO controller following a device reset. Each queue must then be configured by the host before use, which it does by:

- Writing double-word aligned addresses into both the Start and End pointer registers, ensuring that the End pointer is at a higher address than the Start pointer, and that the minimum number of entries is four. Note that as each entry consists of eight words.
- Writing a '1' to the appropriate configuration bit in the MQ_CTRL register. This causes the SRIO controller to copy the Start pointer value to both the Head and Tail pointer registers and the queue fill level indicators to begin operating. The Start and End pointers must not be changed once a queue has been configured.

At this point, the queue is empty and ready for the SRIO controller to add entries. An Empty flag is implemented per queue (EMPTY in register MQn_HPTR) to allow the host to differentiate between the full and empty conditions when the Head and Tail pointers are equal. In addition, to aid in managing the queue, a Not Nearly Empty flag is implemented (NNE in register MQ_STATx). The host can set the threshold at which this operates (using NET in register MQ_CTRL3). An interrupt can be generated if desired when the Not Nearly Empty condition is true. This is designed to relieve the burden of processing the queues by allowing the host to program the device so that multiple mailbox entries can be processed on each visit.

To program a Mailbox Queue allocate a memory space for (N) entries aligned to 32-Byte boundary. Shift the start of the memory space to the right by one bit and write it to the MQn_SPTR register. Add $(N - 1) * 32$ to the allocated memory space and shift it to the right by one bit, then write this value to the MQn_EPTR register.

To access the Mailbox-Queue entries:

1. Read the LADDR field of the MQn_SPTR register
2. Left shift the address by one bit
3. Use the resulting address to access the memory

To remove entries, the host should read the 8-word entry in shared host memory starting at the Head pointer address. It should then update the Head pointer by incrementing by one to point to the next legal entry position, taking into account any wrap required when the End pointer is reached. If the updated Head pointer is equal to the Tail pointer, the queue is empty and the Head pointer must not be advanced any further (otherwise queue corruption will result). The host must service the Mailbox Queues periodically or else they could fill. If a queue is allowed to become full, then incoming MESSAGE transactions for that mailbox can not be processed (RETRY RESPONSES being returned to the sender). In order to inform the host that this has occurred, a Full flag is generated (FULL in register MQn_STAT) from which an interrupt can be

generated. If the queue is full, it is possible to go straight to the empty condition by writing the Head pointer with an unchanged value.

By default, messages will be accepted for 4 Mailboxes. However, the host can limit which are accepted by altering the HMA value in the MQ_CTRL3 register.

NETLOGIC
CONFIDENTIAL

20.7.11.1 MQn_SPTR***Mailbox Queue n Start Pointer***

Address Offset: 0x21000 to (0x21000 + 0x20*n)

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Forms the lower 32 bits of pointer when 0b000 added as lowest 3 bits (to ensure double-word alignment)	RW	0x00000000
29:31		Reserved	RW	All 0's

20.7.11.2 MQn_EPTR***Mailbox Queue n End Pointer***

Address Offset: 0x21004 to (0x21004 + 0x20*n)

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Forms the lower 32 bits of pointer when 0b000 added as lowest 3 bits (to ensure double-word alignment)	RW	0x00000000
29:31		Reserved	RW	All 0's

20.7.11.3 MQn_HPTR***Mailbox Queue n Head Pointer***

Address Offset: 0x21008 to (0x21008 + 0x20*n)

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Forms the lower 32 bits of pointer when 0b000 added as lowest 3 bits (to ensure double-word alignment)	RW	0x00000000
29	-	Reserved	RW	0b0
30	EMPTY	Indicates if the queue is Empty or not: 0 = queue is not empty 1 = queue is empty	RW	0b0
31	-	Reserved	RW	0b0

20.7.11.4 MQn_TPTR***Mailbox Queue n Tail Pointer***

Address Offset: 0x2100C to (0x2100C + 0x20*n)

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Forms the lower 32 bits of pointer when 0b000 added as lowest 3 bits (to ensure double-word alignment)	R	0x00000000
29:31		Reserved	R	All 0's

20.7.11.5 MQ_UPTR***Mailbox Queue n Upper Pointer***

Common High portion of pointers for Mailbox Queue n.

Address Offset: 0x21010 to (0x21010 + 0x20*n)

Bit#	Name	Description	Attribute	Reset
0:31	HADDR	Upper 32 bits of all 4 pointers (ignored if using a 32-bit address host).	RW	0x00000000

20.7.11.6 MQ_CTRL1***Mailbox Queues Control Register 1***

Address Offset: 0x21800

Bit#	Name	Description	Attribute	Reset
0:27		Reserved	RW	All 0's
28:31	CONF	Per queue Configure (bit 31 relates to MQ0, bit 30 to MQ1 etc.) Writing a '1' to a bit causes the respective Mailbox Queue 'n' to be configured ready for use.	RW	0x0

20.7.11.7 MQ_CTRL2***Mailbox Queues Control Register 2.***

This register is only implemented if NMQ >= 33.

Address Offset: 0x21804

Bit#	Name	Description	Attribute	Reset
0:27		Reserved	RW	All 0's
28:31	CONF	Per queue Configure (bit[31] relates to MQ32, bit[30] to MQ33 etc). Writing a '1' to a bit causes the respective Mailbox Queue 'n' to be configured ready for use.	RW	0x0

20.7.11.8 MQ_CTRL3***Mailbox Queues Control Register 3***

Address Offset: 0x21808

Bit#	Name	Description	Attribute	Reset
0:15	-	Reserved	RW	All 0's
16:19	NET	Nearly Empty Threshold. Sets the number of entries in a queue at which the Not Nearly Empty flag (NNE bit in the MQ_STATx register) is asserted and negated: 0x0 = 0 (i.e., flag becomes queue not empty) 0x1 = 1 0x2 = 2 0x3 = 4 0x4 = 8 others = Reserved	RW	0x0
20:25	HMA	Sets the Highest-numbered Mailbox on which a message will be Accepted. All messages will be accepted which fall in the range 0...HMA. Any falling outside the range will not be accepted and have an ERROR RESPONSE returned for them.	RW	0x3F
26:31	-	Reserved	RW	All 0's

20.7.11.9 MQ_STAT1***Mailbox Queues Status Register 1***

Address Offset: 0x21820

Bit#	Name	Description	Attribute	Reset
0:27	-	Reserved	RW	All 0's
28:31	FULL	Per queue Full flags (bit[31] relates to MQ0, bit[30] to MQ1 etc). Set when the respective queue is full. This is a serious condition which the host must quickly rectify by reading entries and updating the Head pointer.	R	0x0

20.7.11.10 MQ_STAT2***Mailbox Queues Status Register 2***

This register is only implemented if NMQ >= 33.

Address Offset: 0x21824

Bit#	Name	Description	Attribute	Reset
0:27	-	Reserved	RW	All 0's
28:31	FULL	Per queue Full flags (bit[31] relates to MQ32, bit[30] to MQ33 etc). Set when the respective queue is full. This is a serious condition which the host must quickly rectify by reading entries and updating the Head pointer.	R	0x0

20.7.11.11 MQ_STAT3***Mailbox Queues Status Register 3***

Address Offset: 0x21828

Bit#	Name	Description	Attribute	Reset
0:27	-	<i>Reserved</i>	RW	All 0's
28:31	NNE	Per queue Not Nearly Empty flags (bit[31] relates to MQ0, bit[30] to MQ1 etc). The threshold at which not nearly empty is declared is controlled by the NET field in the MQ_CTRL3 register: 0 = queue is nearly empty (fill level below threshold) 1 = queue is not nearly empty Only bits [32-NMQ] to [31] are implemented; any remaining bits are reserved.	R	0x0

20.7.11.12 MQ_STAT4***Mailbox Queues Status Register 4.***

This register is only implemented if NMQ >= 33.

Address Offset: 0x2182C

Bit#	Name	Description	Attribute	Reset
0:31	-	<i>Reserved</i>	RW	All 0's

20.7.11.13 MQ_IER1***Mailbox Queues Interrupt Enable Register 1***

The MQ1 bit set in the MISR register is set (from which an interrupt can be generated if desired) whenever a bit in this register and the corresponding bit in the MQ_STAT1 register are both set.

Address Offset: 0x21830

Bit#	Name	Description	Attribute	Reset
0:27	-	<i>Reserved</i>	RW	All 0's
28:31	FULL	Full interrupt enable	RW	0x0

20.7.11.14 MQ_IER2***Mailbox Queues Interrupt Enable Register 2***

The MQ2 bit set in the MISR register is set (from which an interrupt can be generated if desired) whenever a bit in this register and the corresponding bit in the MQ_STAT2 register are both set. This register is only implemented if NMQ >= 33.

Address Offset: 0x21834

Bit#	Name	Description	Attribute	Reset
0:31	-	<i>Reserved</i>	RW	All 0's

20.7.11.15 MQ_IER3***Mailbox Queues Interrupt Enable Register 3***

The MQ3 bit set in the MISR register is set (from which an interrupt can be generated if desired) whenever a bit in this register and the corresponding bit in the MQ_STAT3 register are both set.

Address Offset: 0x21838

Bit#	Name	Description	Attribute	Reset
0:27	-	<i>Reserved</i>	RW	All 0's
28:31	NNE	Not-Nearly-Empty interrupt enable	RW	0x0

20.7.11.16 MQ_IER4***Mailbox Queues Interrupt Enable Register 4***

The MQ4 bit set in the MISR register is set (from which an interrupt can be generated if desired) whenever a bit in this register and the corresponding bit in the MQ_STAT4 register are both set. This register is only implemented if NMQ >= 33.

Address Offset: 0x2183C

Bit#	Name	Description	Attribute	Reset
0:31	-	<i>Reserved</i>	RW	All 0's

20.7.12**Logical/Transport Layer Error and Capture Registers**

When a logical or transport layer error is detected, the appropriate error bit is set by the hardware in the Error Detect CSR. If the corresponding bit is also set in the Error Enable CSR, many fields of the packet header are stored in the Capture CSRs (not all fields are valid for all errors; see the individual CSR descriptions for details), the Error Detect and Capture CSRs lock, and the LTL bit in the MISR register set (from which an interrupt can be generated if desired). Any subsequent errors are ignored until the lock is released by the software writing all 0's to the Error Detect CSR. An interrupt is generated when the LTL bits in both the MISR and MIER registers are both high. Note that the LTL bit in MISR is an indication of the lock status, and will only clear when the lock is released.

The interrupt should trigger error-handling software to come and read the registers. It can determine which particular error triggered the capture by bitwise ANDing the Error Detect and Error Enable CSRs which will yield a single '1' bit in the appropriate position.

Although all of these registers are writable by software, the software should only normally write to the Error Detect CSR (individual 0's to clear specific error bits, or all 0's to clear all bits and clear any lock) and the Error Enable CSR (to control on which errors a Capture and Lock operation should be performed). Write capability is provided to the other registers to aid verification of software error handling procedures. To use this feature, the software must follow the following procedure:

- Write all 0's to the Error Enable CSR (to prevent the hardware performing any capture operations itself)
- Write all 0's to the Error Detect CSR (to clear any error bits and clear any lock (although the existence of a lock does not prevent the software from writing any of the registers))
- Write the desired values to the Capture CSRs
- Write one or more bits in the Error Enable CSR (corresponding to the error scenario being verified)
- Write one or more bits in the Error Detect CSR (ensuring that a bitwise AND of this CSR with the Error Enable CSR would produce at most a single '1'). If the hardware detects any of these bits correspond to a set bit in the Error Enable CSR, it will lock the registers and set the LTL bit in MISR.

The normal error-handling software can then read and write registers as necessary to complete the error handling procedure and it will be unaware that the hardware did not trigger the event.

20.7.12.1**LTL_ERD_CSR*****Logical/Transport Layer Error Detect CSR register***

Indicates the error(s) detected by the Logical or transport layer. Multiple bits may get set in the register, if simultaneous errors are detected during the same clock cycle that the errors are logged, or if errors occur for which the respective bits in the LTL_ERE_CSR register are not set (so no capture and lock occurs).

This is a plain read/write register. To clear bits, software must write zeros with ones at any bit position it requires to stay at one. Note that this means that the software can write ones to emulate hardware defects events to allow software debug (see the preceding section for more details). However, once a capture and lock has occurred, the lock can only be cleared by writing all 0's to this register.

Note that two of the checks performed on incoming packets are performed before all the others; ITTE (bit 5) and UT (bit 9). None of the other checks are performed if either of these fail.

Address Offset: 0x30008

Bit#	Name	Description	Attribute	Reset
0	IOER	IO Error Response. Received a RESPONSE or MAINTENANCE response with status of 'ERROR'. Packet accepted. This check is only performed if first two checks passed.	RW	0b0
1	MER	Message Error Response. Received a MESSAGE RESPONSE with status of 'ERROR'. Packet accepted. This check is only performed if first two checks passed.	RW	0b0
2	-	Reserved	RW	0b0
3	MFE	Message Format Error. Received MESSAGE packet data payload with an invalid size or segment. Packet is rejected. This check is only performed if first two checks passed. Details of the checks performed are as follows: <ul style="list-style-type: none"> The segment size code (SSIZE) is checked for validity (that it decodes to 8, 16, 32, 64, 128 or 256 bytes). The actual amount of payload is checked to be equal to that indicated by SSIZE, except for the last or only segment of a message where the payload length is checked to be equal to or less than the SSIZE code. The segment number (msgseg) is checked to be not greater than the total number of segments in the message (msglen). 	RW	0b0
4	ITD	Illegal Transaction Decode. Received illegal fields in the request/response packet for a supported transaction. The list below shows what is checked. Note that no packets are deleted as result of a check failure, but are passed on to the respective destinations where a (status = ERROR) RESPONSE packet may be generated if appropriate. Checks are only performed if first two checks passed. <ol style="list-style-type: none"> For NWRITE and NWRITE_R requests, the {wdptr, size} coding is checked for validity, and the actual amount of payload data received is less than or equal to the amount indicated in the coding For MAINTENANCE read and write requests, the {wdptr, size} coding is checked to be 4, 8, 16, 32 or 64 bytes, and for write requests, the actual amount of payload data is less than or equal to the amount indicated in the coding. For transactions where there should be a payload (NWRITE, NWRITE_R, SWRITE, MESSAGE, PORT-WRITE, MAINTENANCE write request, MAINTENANCE read response with status = DONE, RESPONSE with data and status = DONE) a check is made that there is one. For transactions where there should not be a payload (NREAD, DOORBELL, MAINTENANCE read request, MAINTENANCE write response with status = DONE, MAINTENANCE read and write response with status = ERROR, RESPONSE without data, RESPONSE with status = ERROR, RESPONSE with status = RETRY) a check is made that there is not one. For MAINTENANCE response transactions, a check is made of the status field to ensure that it is either DONE or ERROR. For I/O and MESSAGE RESPONSEs, a check is made of the status field to ensure that it is either DONE, RETRY or ERROR. 	RW	0b0
5	ITTE	Illegal Transaction Target Error. Rejected a packet because it contained a destination device ID that is not defined for this end-point and the Device ID Packet Reject control bit (in the General Control Register) is set. This is the first check performed.	RW	0b0
6	MRTO	Message Request Time-out. A required message request has not been received within the specified time-out interval.	RW	0b0
7	RTO	Response Time-out. Required response has not been received within the specified time out interval.	RW	0b0

Bit#	Name	Description	Attribute	Reset
8	UR	Unsolicited Response. An unsolicited/unexpected Response packet was received. This is check is only performed if first two checks passed.	RW	0b0
9	UT	Unsupported Transaction. A transaction is received that is not supported (either because it contains reserved codes in ftype and /or ttype fields, or the Destination Operations CAR shows the transaction as unsupported). Packet is deleted. This is the second check performed , and is only performed if first check passed.	RW	0b0
10:23		Reserved	RW	All 0's
24	RLM	Response Length Mismatch. A read RESPONSE or MAINTENANCE read response is received which is correct in all aspects except that the payload length is not the same as the requested length. Response data is ignored, and host aborted if appropriate.	RW	0b0
25	ERT	Exhausted Retries. A re-triable request (MESSAGE or DOORBELL) has used all its allowed retry attempts.	RW	0b0
26:31		Reserved	RW	All 0's

20.7.12.2 LTL_ERE_CSR

Logical/Transport Layer Error Enable CSR

A capture and lock operation is performed if a bit in this register is ‘1’ when the corresponding bit in the LTL_ERD_CSR register is set (indicating an error). This also causes the LTL bit in the MISR register to be set, from which an interrupt can be generated if desired. The capture registers are abbreviated as follows:

HAC/LAC: Higher and Lower Address Capture

DID: Device Identity (source and destination from the errored packet)

CC: Control Capture (various fields from the errored packet, with two different formats, 1 and 2)

The information to populate all of these registers is not available for all errors. Each error describes which capture registers are used. When a capture register is not relevant to an error, it is loaded with all 0's.

Address Offset: 0x3000C

Bit#	Name	Description	Attribute	Reset
0	IOER	Enable reporting of an I/O error response. DID and CC (format 1) capture registers populated (address information from originating request not available).	RW	0b0
1	MER	Enable reporting of a Message error response. DID and CC (format 1) capture registers populated (address information from originating request not available).	RW	0b0
2	-	Reserved	RW	0b0
3	MFE	Enable reporting of a message format error. DID and CC (format 1) capture registers populated (address information not applicable for Messages).	RW	0b0
4	ITD	Enable reporting of an illegal transaction decode error. All capture registers populated (CC format 1). Note that HAC/LAC may or may not contain valid information depending upon the packet type.	RW	0b0
5	ITTE	Enable reporting of an illegal transaction target error. All capture registers populated (CC format 1). Note that HAC/LAC may or may not contain valid information depending upon the packet type.	RW	0b0

Bit#	Name	Description	Attribute	Reset
6	MRTO	Enable reporting of a Message Request time-out error. DID and CC (format 1) capture registers populated (address information not applicable for Messages).	RW	0b0
7	RTO	Enable reporting of a packet response time out error. DID and CC (format 2) capture registers populated (address information from originating request not available or not applicable).	RW	0b0
8	UR	Enable reporting of unsolicited response error. DID and CC (format 1) capture registers populated (address information not available in response packets).	RW	0b0
9	UT	Enable reporting of unsupported transaction error. All capture registers populated (CC format 1). Note that HAC/LAC may or may not contain valid information depending upon the packet type.	RW	All 0's
10:23		Reserved	RW	0b0
24	RLM	Enable reporting of incorrect length response payload. DID and CC (format 2) capture registers populated (address information from originating request not available).	RW	0b0
25	ERT	Enable reporting of exhausted retries event. DID and CC (format 1) capture registers populated (address information not applicable to re-triable requests).	RW	All 0's
26:31		Reserved	RW	0b0

20.7.12.3 LTL_HAC_CSR

Logical/Transport Layer High Address Capture CSR

Contains address information associated with an errored received request packet. Only valid when indicated by the LTL_ERE_CSR registers bit descriptions (loaded with all zeros otherwise). It is locked when a Logical/Transport error is detected and the corresponding enable bit is set.

Address Offset: 0x30010

Bit#	Name	Description	Attribute	Reset
0:31	HADDR	High-order Address. Near-most significant 32 bits of address. The significance of the field depends upon the addressing mode (unused portions are set to all zeros): 34-bit: HADDR[0:31] unused 50-bit: HADDR[16:31] forms address[2:17] (HADDR[0:15] unused) 66-bit: HADDR[0:31] forms address[2:33]	RW	0x00000000

20.7.12.4 LTL_LAC_CSR***Logical/Transport Layer Address Capture CSR***

Contains address information associated with an errored received request packet. Only valid when indicated by the LTL_ERE_CSR registers bit descriptions (loaded with all zeros otherwise). It is locked when a Logical/Transport error is detected and the corresponding enable bit is set.

Address Offset: 0x30014

Bit#	Name	Description	Attribute	Reset
0:28	LADDR	Low-order Address. Least significant 29 bits of address. The significance of the field depends upon the addressing mode: 34-bit: LADDR[0:28] forms address[2:30] 50-bit: LADDR[0:28] forms address[18:46] 66-bit: LADDR[0:28] forms address[34:62]	RW	0x0000000
29	-	Reserved	RW	0b0
30:31	XADDR	Extended Address (xamsbs). Most significant 2-bits of address. Forms address[0:1] in all addressing modes.	RW	0b00

20.7.12.5 LTL_DID_CSR***Logical/Transport Layer Device ID Capture CSR***

Contains Error information. Only valid when indicated by the LTL_ERE_CSR registers bit descriptions (loaded with all zeros otherwise). It is locked when a Logical/Transport error is detected and the corresponding enable bit is set. The fields are taken directly from the errored packet, when available.

Address Offset: 0x30018

Bit#	Name	Description	Attribute	Reset
0:7	HDID	Most significant byte of the destination ID associated with the error. For a timed-out response error (RTO), not valid.	RW	0x00
8:15	DID	The destination ID associated with the error. For a timed-out response error (RTO), not valid.	RW	0x00
16:23	HSID	Most significant byte of the source ID associated with the error. For a timed-out response error (RTO), contains the ID of the device from which the response was expected.	RW	0x00
24:31	SID	The source ID associated with the error. For a timed-out response error (RTO), contains the ID of the device from which the response was expected.	RW	0x00

20.7.12.6 LTL_CC_CSR***Logical/Transport Layer Control Capture CSR***

Contains Error information. Only valid when indicated by the LTL_ERE_CSR registers bit descriptions (loaded with all zeros otherwise). It is locked when a Logical/Transport error is detected and the corresponding enable bit is set. The fields in this register take on two different formats depending upon the type of error causing the capture (see descriptions in [LTL_ERE_CSR register](#)).

Address Offset: 0x3001C

Bit#	Name	Description	Attribute	Reset
Format 1 (relate to most types of error)				
0:3	FTYPE	Format type of the packet containing the reported error	RW	0x0
4:7	TTYPE	For MESSAGE transaction, the msglen field. For other transactions, the Transaction type of the packet containing the reported error (not valid for SWRITE or DOORBELL transactions).	RW	0x0
8:15	TID	For IO type transactions, the Transaction ID field of the packet containing the reported error. For MESSAGE type transactions, the {letter, mbox, msgseg/xmbox} information.	RW	0x00
16:19	SS	The read size, write size or status field (depending upon the packet type) of the packet containing the reported error (not valid for SWRITE or DOORBELL)	RW	0x0
20:21	TT	Value of the TT field of the packet containing the reported error (not valid if performing a MRTO capture)	RW	0b00
22:24	PRIOR	The priority of the packet containing the reported error (not valid if performing a MRTO capture)	RW	0b000
25	WDPTR	Word pointer of the packet containing the reported error (not valid for DOOBELL, MESSAGE or RESPONSE)	RW	0b0
26:31	TPYD	Code describing length of any payload as determined by the transport layer. If payload detected, [26] = 1 and [27:31] gives number of double-words counted (0 = 1double word etc). If no payload detected, [26] = 0 and [27:31] undefined.	RW	0x00
Format 2 (relate to a timed-out response (RTO), or a response with an unexpected payload length (RLM)):				
0:3	FTYPE	The Format type of the originating request	RW	0x00
4:7	TTYPE	The Transaction type of the originating request	RW	0x0
8:15	TID	For IO type transactions, the Transaction ID field. For MESSAGE type transactions, the {letter, mbox, msgseg/xmbox} information.	RW	0b00
16:20	ELEN	The expected length of the response payload, if any (see EPYD bit): 0 = 1 double word (8 bytes) ... 31 = 32 double words (256 bytes)	RW	0b0
21	SZ	Size of SID field captured in the LTL_DID_CSR register: 0 = 8-bit 1 = 16-bit	RW	All 0's
22	EPYD	Whether a payload was expected on the response or not: 0 = don't expect payload 1= expect payload	RW	0b0
23:24	-	Reserved	RW	0x00

Bit#	Name	Description	Attribute	Reset
25	REQ	Indicates the type of the originating request (and hence the format of TID): 0 = IO (NREAD, NWRITE_R, MAINTENANCE) or DOORBELL 1 = MESSAGE	RW	0x00
26:31	TPYD	Code describing length of any payload as determined by the transport layer. If payload detected, [26] = 1 and [27:31] gives number of double-words counted (0 = 1double word etc). If no payload detected, [26] = 0 and [27:31] undefined.	RW	0x0

20.7.13 Port 0 Error Reporting Registers

When a port 0 physical layer error is detected, the appropriate error bit is set in the Error Detect CSR. If the corresponding bit is also set in the Error Enable CSR, the Attribute Capture CSR is written with details of the error and, if the Info Type is Packet, the four Packet Capture CSRs also written.

All the Capture CSRs are then locked and the P0 bit in the MISR register set (from which an interrupt can be generated if desired). Any subsequent errors continue to be logged in the Error Detect CSR (it is not locked) but do not cause a Capture until the lock is released by the software clearing the Capture Valid Info bit in the Attributes Capture CSR by writing a zero to it (the status of the lock is always indicated by this bit). An interrupt is generated when the P0 bits in both the MISR and MIER registers are both high. Note that once set, the P0 bit in MISR will only clear when the lock is released.

The interrupt should trigger error-handling software to come and read the registers. It can determine which particular error triggered the capture by examining the Error Type field of the Attributes Capture CSR.

Although all of these registers are writable by software, the software should only normally write to the Error Detect CSR, Error Enable CSR and the Capture Valid Info bit of the Attributes Capture CSR. Write capability is provided to the other registers to aid verification of software error handling procedures. To use this feature, the software must follow the following procedure:

- Write all 0's to the Error Enable CSR (to prevent the hardware performing any capture operations itself)
- Write the required values to the Attributes Capture CSR, ensuring the Capture Valid Info bit is set (to lock the hardware)
- Write the desired values to the other Capture CSRs
- Write 1 or more bits in the Error Enable CSR (corresponding to the error scenario being verified)
- Write 1 or more bits in the Error Detect CSR (ensuring that a bitwise AND of this CSR with the Error Enable CSR would produce at least one '1', and one of these correspond to the value written to the Error Type field of the Attributes Capture CSR). If the hardware detects any of these bits correspond to a set bit in the Error Enable CSR, and the Capture Valid Info bit is set, it will set the P0 bit in MISR.

The normal error-handling software can then read and write registers as necessary to complete the error handling procedure and it will be unaware that the hardware did not trigger the event.

20.7.13.1 P0_ERD_CSR***Port 0 Error Detect CSR***

Indicates transmission errors that are detected by the hardware. If an error is detected and the corresponding bit in the Port 0 Error Enable CSR is high, then a Capture and Lock operation is performed. Note that this CSR does not lock. Any errors detected before and after a Capture and Lock operation are recorded in this CSR.

Note that this is a plain read/write register. To clear bits, software must write zeros with ones at any bit position it requires to stay at one. Note that this means that the software can write ones to emulate hardware defects events to allow software debug (see the preceding section for more details).

The Priority column indicates the order in which Packet errors are evaluated (items without a priority are Control-type errors). The priority 1 error is evaluated first, and if it is detected, no other Packet errors are checked. Thus for any declared error, it can be assumed that all the checks for lower-numbered errors have been passed.

Address Offset: 0x30040

Bit#	Name	Description	Priority	Attri.	Reset
0	BC	Bad Cause. Received a packet that was rejected by the PHY for a reason not covered by other error indications in this register.	6	RW	0b0
1:4	CCRC5 CUACK CCRC16 CGT276	These 4 bits largely mirror the packet-type indications given by bits[9, 12, 13 and 14] respectfully, but without the packet capture possibility. Normally, these bits will be set earlier than their equivalent bits (because of a shorter path through the hardware). However, at certain times (e.g. when the Receive Transport Buffer is getting full) only these bits will be set. In addition, only CCRC5 (and not BCRC5) will be set if an /SC/ character is received with a CRC5 error.	-	RW	0x0
5:8		Reserved	-	RW	0x0
9	BCRC5	Received a corrupt control symbol (bad CRC5 value).	2	RW	0b0
10	CSACK	Received an acknowledge control symbol with unexpected ackID (packet_accepted or packet_retry)	-	RW	0b0
11	PNA	Received Packet-Not-Accepted acknowledge control symbol	-	RW	0b0
12	UACK	Received packet with unexpected ackID – out of sequence ackID	3	RW	0b0
13	BCRC16	Received packet with bad CRC16 value.	1	RW	0b0
14	GT276	Received packet exceeds 276 bytes (the maximum allowed size)	5	RW	0b0
15:25	-	Reserved	-	RW	All 0's
26	NOACK	Non-outstanding ackID. Link response received with an ackID that is not outstanding	-	RW	0b0
27	PE	Protocol Error. An unexpected packet or control symbol was received	4	RW	0b0
28	-	Reserved	-	RW	0b0
29	DE	Delineation Error. Received unaligned /SC/ or /PD/ or undefined group code (serial PHY)	-	RW	0b0
30	UACS	Unsolicited Acknowledge Control Symbol. An unexpected acknowledge control symbol was received	-	RW	0b0
31	LTO	Link Time-out. An acknowledge or link-response control symbol is not received within the specified time-out interval.	-	RW	0b0

20.7.13.2 P0_ERE_CSR***Port 0 Error Enable CSR***

This register contains the bits to control when an error condition logged in the P0_ERD_CSR register causes a capture and lock operation. The P0 bit in the MISR register is set (from which an interrupt can be generated if desired) whenever a lock is engaged, and cleared when the lock is released.

Address Offset: 0x30044

Bit#	Name	Description	Attribute	Reset
0	BC	Bad Cause	RW	0b0
1:4	CCRC5 CUACK CCRC16 CGT276	Non-packet-capture versions of BCRC5, UACK, BCRC16 and GT276 bits respectively.	RW	0x0
5:8	-	<i>Reserved</i>	RW	0x0
9	BCRC5	Received Corrupt Control Symbol	RW	0b0
10	CSACK	Received acknowledge control symbol with unexpected ackID	RW	0b0
11	PNA	Rx'd Packet-Not-Accepted control symbol	RW	0b0
12	UACK	Received packet with unexpected ackID	RW	0b0
13	BCRC16	Received packet with bad CRC	RW	0b0
14	GT276	Received packet exceeds 276 bytes	RW	0b0
15:25	-	<i>Reserved</i>	RW	All 0's
26	NOACK	Non-outstanding ackID	RW	0b0
27	PE	Protocol Error	RW	0b0
28	-	<i>Reserved</i>	RW	0b0
29	DE	Delineation Error	RW	0b0
30	UACS	Unsolicited Acknowledge Control Symbol	RW	0b0
31	LTO	Link Time-out	RW	0b0

20.7.13.3 P0_AC_CSR***Port 0 Attributes Capture CSR***

Indicates the type of information contained in all the Port 0 Capture registers. In the case of multiple detected errors during the same clock cycle, one of the errors must reflect the Error type field.

Address Offset: 0x30048

Bit#	Name	Description	Attribute	Reset
0:1	ITYPE	Information Type logged: 0 = Packet (packet capture CSR's valid) 1 = Control Symbol (packet capture CSR's not valid) others = Reserved	RW	0b00
2	-	Reserved	RW	0b0
3:7	ETYPE	Error Type. The encoded value of the bit in the Port 0 Error Detect CSR that describes the error captured in all the Port 0 Capture CSR's.	RW	0x00
8:30		Reserved	RW	All 0's
31	CIV	Capture Information Valid. This bit is set by hardware to indicate that the Capture registers (just this CSR for Control Symbol Info Type, this plus the four Packet Capture CSR's for Packet Info Type) contain valid information and are locked. This means that the hardware will not perform any further captures until software clears this bit (the locks condition does not stop the software from writing these registers).	RW	0b0

20.7.13.4 P0_PC0_CSR***Port 0 Packet Capture 0 CSR***

Contains bytes 0,1,2 & 3 of the packet.

This register holds the first 4-bytes of a captured errored packet. However, note that the first 5-bits of byte 0 are not a true reflection of what was received on the port (the AckID). Instead, these bits have been overwritten with internal information as follows:

- bit[0]: used to indicate that the port has detected an error (so will always be captured as '1').
- bits[1:4] gives the reason why the packet was considered errored (this cause value is decoded to set the relevant bit in the P0_ERD_CSR).

Note that bits[5:7] of byte 0 retain the value of the 3-bit reserved field as received.

Figure 20-15. Packet Capture 0 CSR - Bits 0-15



Address Offset: 0x3004C

Bit#	Name	Description	Attribute	Reset
0:31	DATA	Bytes 0 to 3 of the packet (but note that some data in byte 0 is overwritten by internal usage, see above)	RW	0x00000000

20.7.13.5 P0_PC1_CSR

Port 0 Packet Capture 1 CSR

Contains bytes 4, 5, 6 & 7 of the packet.

Address Offset: 0x30050

Bit#	Name	Description	Attribute	Reset
0:31	DATA	Bytes 4 to 7 of the packet	RW	0x00000000

20.7.13.6 P0_PC2_CSR

Port 0 Packet Capture 2 CSR

Contains bytes 8, 9, 10 & 11 of the packet

Address Offset: 0x30054

Bit#	Name	Description	Attribute	Reset
0:31	DATA	Bytes 8 to 11 of the packet	RW	0x00000000

20.7.13.7 P0_PC3_CSR

Port 0 Packet Capture 2 CSR

Contains bytes 12,13, 14 & 15 of the packet

Address Offset: 0x30058

Bit#	Name	Description	Attribute	Reset
0:31	DATA	Bytes 12 to 15 of the packet.	RW	0x00000000



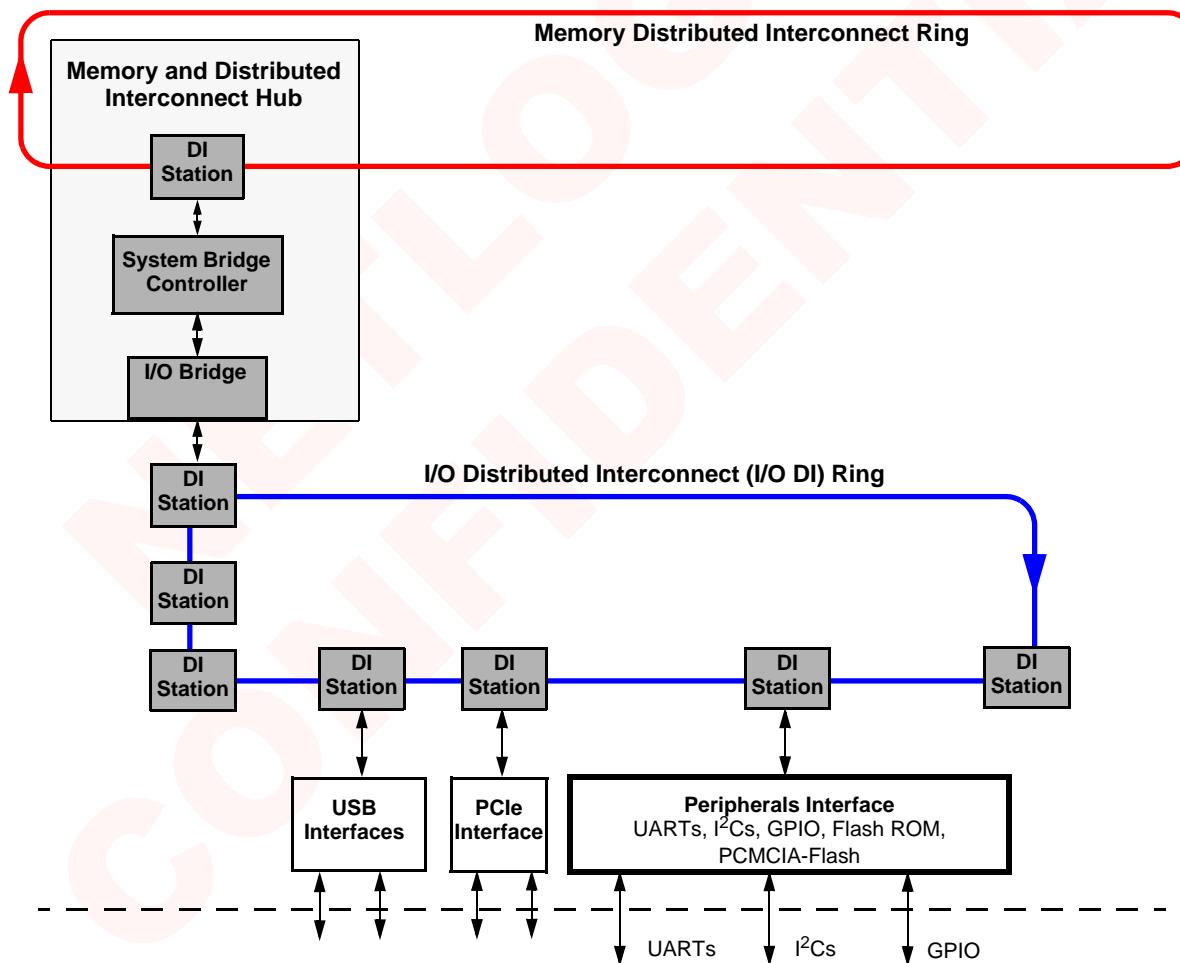
Chapter 21 UARTs, I²C, and Peripherals Interface

21.1 Introduction

This chapter provides information about the two UART ports and the two IIC (I^2C) ports.

The following diagram shows connectivity for the Peripheral Interface inside the XLS.

Figure 21-1. The Peripherals Interface on the I/O Distributed Interconnect Ring



21.2 UART Ports

The two XLS processor UART (Universal Asynchronous Receiver Transmitter) ports 1 and 2 provide serial communication capabilities, which allow communication with the modem or other external devices using RS-232 protocol. Each UART port is separately configured and can be run as an asynchronous link from 1200 baud to 4 Mbaud.

Note that if compatibility with industry standard 16550 UART devices is desired, the following software considerations must be made:

- Compatibility with the 16550 on transmit requires the DTR and RTS bits to be inverted by the hardware when they are transmitted out the UART. The XLS transmits DTR and RTS as they are written in the **UART_MODEM_CTL** register (Modem Control register) without this inversion. These bits must be inverted by software if 16550 compatibility is required.
- Compatibility with the 16550 on receive requires the DCD, RI, DSR, and CTS bits to be inverted by the hardware when they are received at the UART. The XLS does not invert these signals before they are stored in the **UART_MODEM_STS** register (Modem status register) as required for 16550 compatibility.

21.2.1 UART Port Signals

The sixteen signals for the two UART ports are listed for easy reference.

Table 21-1. UART Ports 1 and 2 Signal Descriptions

Signal Name	Signals	I/O	Description
TXD1	1	Out	Transmit Data port 1
RXD1	1	In	Receive Data port 1
RTS1	1	Out	Request To Send port 1
CTS1	1	In	Clear To Send port 1
DTR1	1	Out	Data Terminal Ready port 1
DSR1	1	In	Data Send Ready port 1
DCD1	1	In	Data Carrier Detect port 1
RI1	1	In	Ring Indicator port 1
TXD2	1	Out	Transmit Data port 2
RXD2	1	In	Receive Data port 2
RTS2	1	Out	Request To Send port 2
CTS2	1	In	Clear To Send port 2
DTR2	1	Out	Data Terminal Ready port 2
DSR2	1	In	Data Send Ready port 2
DCD2	1	In	Data Carrier Detect port 2
RI2	1	In	Ring Indicator port 2

21.3 I²C Ports

XLS Processors have two standard I²C™ bus ports 1 and 2 for serial communications with low-bandwidth devices in the system.

21.3.1 I²C Port Signals

Each I²C bus port consists of two bi-directional bus lines; the Serial Data (SDA) line and the Serial Clock (SCL) line. The four signals for both ports are listed for reference.

Table 21-2. I²C Ports 1 and 2 Signal Descriptions

Signal Name	Signal s	I/O	Description
SDA1	1	I/O	Serial Data port 1
SCL1	1	I/O	Serial Clock port 1
SDA2	1	I/O	Serial Data port 2
SCL2	1	I/O	Serial Clock port 2

Both the Serial Data (SDA) line and the Serial Clock (SCL) line are open drain, so they can be hard-wired to other I²C ports. Communications on the I²C bus is initiated when one device pulls the SDA line Low. The address of the device receiving the communications is shifted out on the SDA line one bit per SCL followed by the data. The transmitter puts the data on the SDA line during the Low phase of the clock, while the receiver samples the data on the SDA line during the high phase of the clock. Each byte of data is acknowledged by the recipient. When the transmission is completed, the transmitter asserts the stop condition by releasing the SDA line during the high phase of SCL. The free SDA line floats high, pulled up by an external resistor on the I²C bus.

XLS Processors support I²C devices with up to 256 data words only and drive only one 8-bit data word address following the device address word and acknowledgement.

21.4 Programming Model

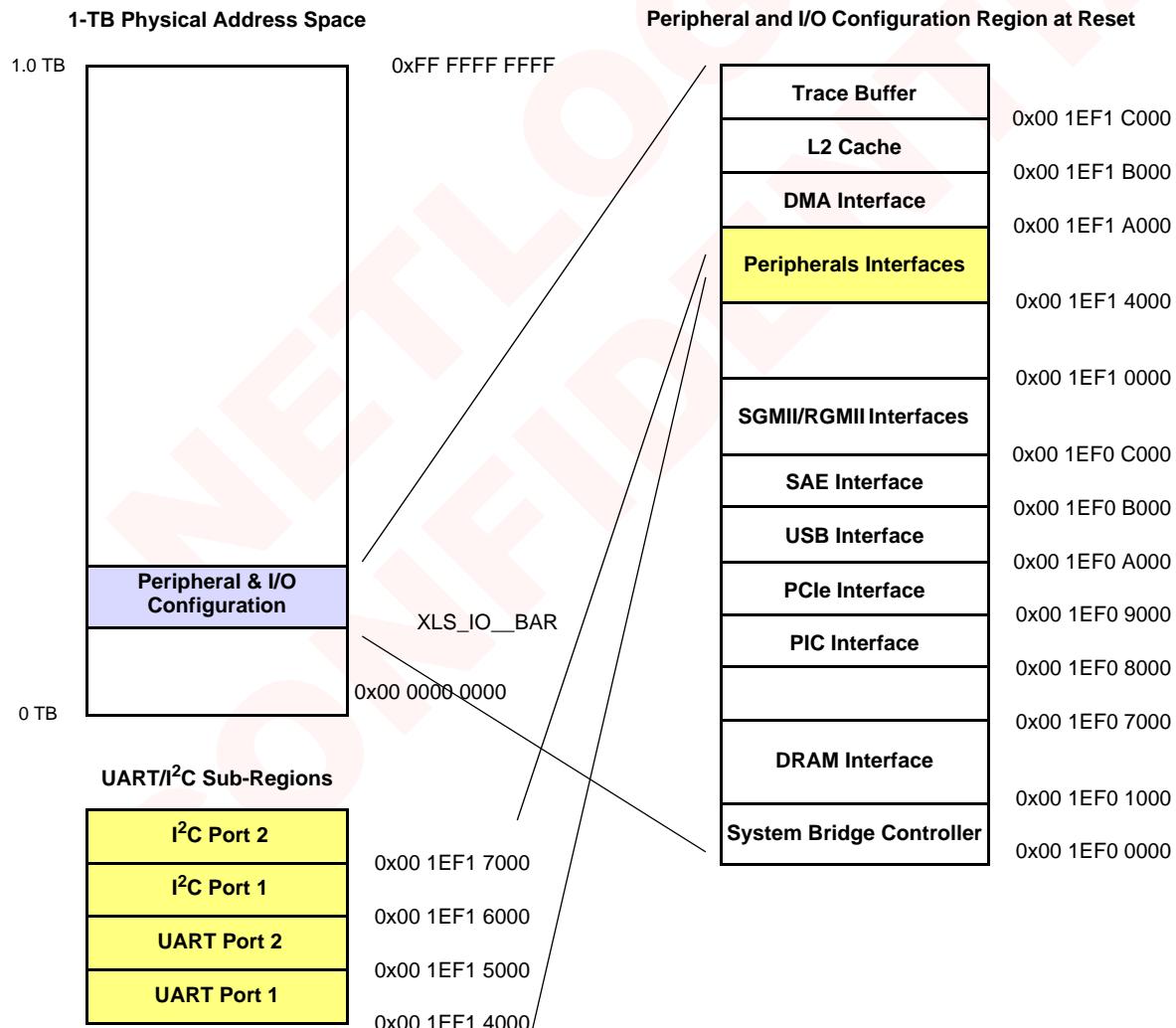
21.4.1 Register Addressing for UARTs and I²C

Operation of the UART/I²C portion of the Peripherals Interface is governed by registers in the four sub-region address spaces as shown in the system memory map in Figure 21-2. The offsets for the sub-region address spaces is given in Table 21-3.

Table 21-3. UART/I²C Sub-Regions in the Peripherals & I/O Configuration Region

Peripheral	Peripheral Device Name	Sub-Region Offset	Sub-Region Offset Address at Reset
UART_1	XLS_IO_DEV_UART_1	0x1 4000	0x00 1EF1 4000
UART_2	XLS_IO_DEV_UART_2	0x1 5000	0x00 1EF1 5000
I ² C_1	XLS_IO_DEV_I2C_1	0x1 6000	0x00 1EF1 6000
I ² C_2	XLS_IO_DEV_I2C_2	0x1 7000	0x00 1EF1 7000

Figure 21-2. Memory Map of the UART/I²C Portion of the Peripheral Interface



21.4.2 UART Port Programming

Both UART ports are identical and programming is the same with the exception that the different sub-region register offsets are used.

21.4.2.1 Reset Initialization

Upon Reset, the UART controller performs the following tasks:

- The receiver and the transmitter FIFOs are cleared
- The receiver and the transmitter shift registers are cleared
- The divisor latch register is set to zero
- The [UART_LINE_CTL](#). register is set for communication of 8 bits of data, no parity and one stop bit
- All interrupts are disabled

21.4.2.2 Initialization Programming

After Reset the following tasks should be programmed:

- Set the [UART_LINE_CTL](#). register with the line control parameters. Set [UART_LINE_CTL](#).[.DIV_ACCESS](#) to allow access to divisor latches
- Set the divisor latches with MSB first and LSB second
- Clear [UART_LINE_CTL](#).[.DIV_ACCESS](#) to disable access to divisor latches
- Set the [RX_FIFO_FULL](#) level in the [UART_FIFO_CTL](#). register. Generally higher level values produce fewer interrupts to the system, so if the system responds fast enough setting it to 14 bytes is recommended.
- Enable the desired interrupts by setting appropriate bits in the [UART_INT_EN](#) register.

21.4.3 I²C Port Programming

The I²C host controller enables the XLS Processor host to access external I²C slave devices using a simple register interface. Software can program the Device Address register [I2C_DEVADDR](#), along with address/data to be written to or read from external I²C slave devices. Then the I²C state machine handles the proper hardware signaling.

Host software configures the I²C controller by programming the [I2C_CONFIG](#), [I2C_HOLD](#) and [I2C_DIVISOR](#) registers. To access an I²C slave device software programs the [I2C_DEVADDR](#), [I2C_ADDR](#) and [I2C_DOUT](#) or [I2C_DIN](#) registers.

Writing the [RWDIR](#) bit to the appropriate value in the [I2C_START](#) register initiates the transfer with the I²C slave device. When a transaction is started, the I²C host controller asserts a start condition on the I²C bus followed by the contents of the [I2C_DEVADDR](#). register, then checks for an acknowledge signal from the slave device. If the proper acknowledge signal is not detected by the I²C host controller the [ACKERR](#) bit is set in the [I2C_STATUS](#) register. If no [ACKERR](#) occurs, then the I²C host controller next transmits the contents of the [I2C_ADDR](#) register and checks for an acknowledge from the I²C slave device. If the transaction is a Write transfer, the I²C host controller next transmits the contents of the [I2C_DOUT](#) register; if the transaction is a Read transfer, the I²C host controller shifts in data from the slave device and places it in the [I2C_DIN](#) register. The I²C host controller then asserts a stop condition.

21.4.3.1 I²C Write Transfers

To initiate a Write to an I²C slave device, the following steps should be executed by software:

1. Configure the **I2C_CONFIG** and **I2C_DIVISOR** registers if necessary
2. Optionally set the **I2C_BYTE_CNT** register with number of bytes to write
3. Write the **I2C_DEVADDR** register
4. Write the **I2C_ADDR** register
5. Write the **I2C_DOUT** register.
6. Write the **I2C_START** register.

When the **I2C_START** register is written, a Start condition is automatically asserted on the I²C bus. When the transfer starts, the **BUSY** bit in the **I2C_STATUS** register is set. The device address in the **I2C_DEVADDR** is serially shifted out, followed by the **RWDIR** bit, the **I2C_ADDR** and the Write data from the **I2C_DOUT** register. When the transaction is completed, the I²C host controller initiates a Stop condition and the **BUSY** bit is cleared.

Note that during a sequential Write, the **SDOEMPTY** bit is set in the **I2C_STATUS** register after every byte is transferred. The state machine then holds I²C bus SCLK signal Low until the software writes the **I2C_DOUT** register. I²C software should monitor the **SDOEMPTY** bit, so new data can be written to the **I2C_DOUT** register as soon as it is available. It is important to keep fresh data in the **I2C_DOUT** register, or stale data can be re-transmitted. The I²C controller repeats this until all bytes specified by the **I2C_BYTE_CNT** register have been transferred. It is not necessary to Write to the **I2C_START** register for each subsequent byte of data.

21.4.3.2 I²C Read Transfers

A commonly used way to Read from I²C devices is for the controller to:

- First perform a Write with no data which issues a device address and the data address. This allows loading the data address and prepares to turn around the SDA bus.
- Then perform a Read operation where only the device address is used as described below.

A Read operation performs the following steps executed by software:

- Configure the **I2C_CONFIG** and **I2C_DIVISOR** registers again if necessary
- Optionally set the **I2C_BYTE_CNT** register with number of bytes to read
- Write the **I2C_DEVADDR** register
- Write the **I2C_START** register.

When the **I2C_START** register is written, a Start condition is automatically asserted on the I²C bus. When the transfer starts, the **BUSY** bit in the **I2C_STATUS** register is set. The device address in **I2C_DEVADDR** will be serially shifted out followed by the **RWDIR** bit. The external device drives the read data on the SD line and the host controller samples the SD line and stores the data in the **I2C_DIN** register. When the data has been received, the I²C host controller completes the transaction by initiating a Stop condition and clearing the **BUSY** bit.

Note that during a sequential Read from an external I²C slave device, the **DATARDY** bit in the **I2C_STATUS** register is set after every data byte is shifted in. The state machine then holds the I²C bus SCLK low until software reads the **I2C_DIN** register. The I²C host controller repeats this until all bytes specified by the **I2C_BYTE_CNT** register have been transferred. It is not necessary to Write to the **I2C_START** register for each subsequent byte of data.

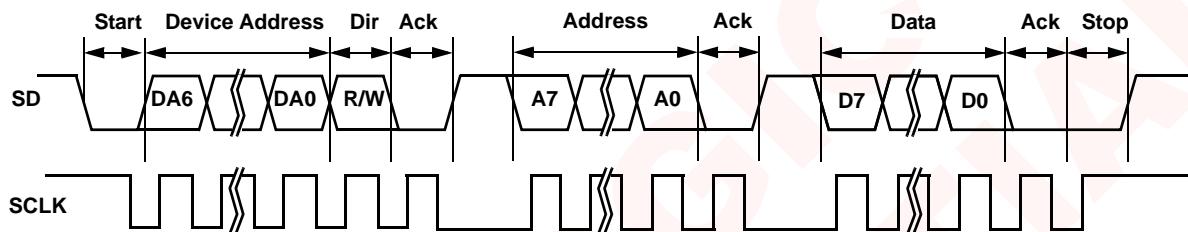
21.4.3.3 Transfer Options

A number of transfer options are available to suit a variety of target devices. Refer to the documentation on each target device to determine the mode necessary to access the devices, and the features supported by the devices.

Normal Transfer

A normal transfer includes a device address, direction bit, address and data. For a normal transfer, **DEVADDIS** is cleared, **ADDRDIS** is cleared, and **NODATA** is cleared.

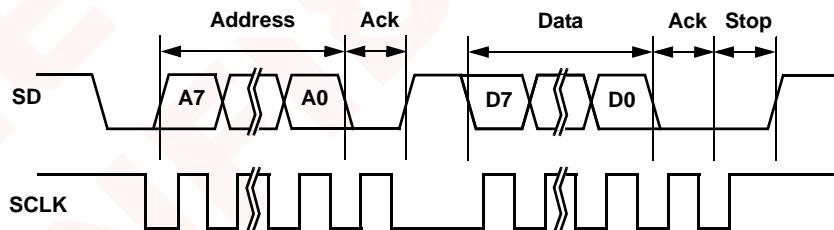
Figure 21-3. I²C Bus Normal Transfer



Device Address Disabled

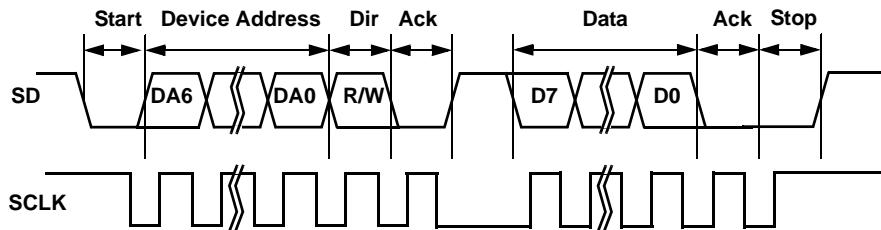
When the device address is disabled, transfer includes an address and data. For this transfer type, **DEVADDIS** is set, **ADDRDIS** may be set or cleared, and **NODATA** is cleared.

Figure 21-4. Device Address Disabled

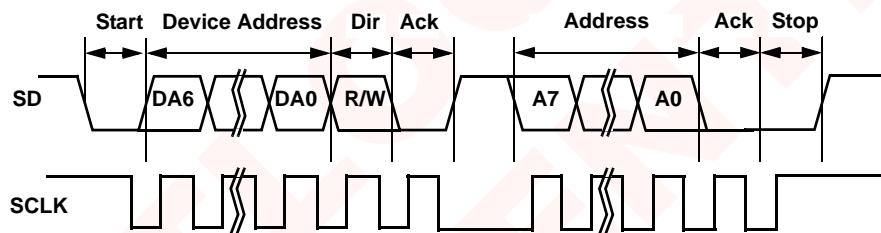


Address Disabled

When the device address is disabled, transfer includes an address and data. For this transfer type, **DEVADDIS** is cleared, **ADDRDIS** is set, and **NODATA** is cleared.

Figure 21-5. Address Disabled**Address Only**

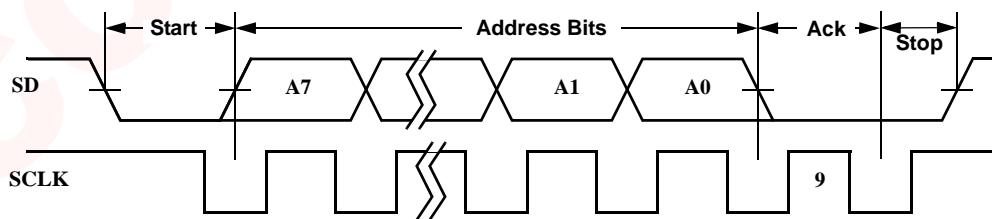
An address-only transfer includes a device address and an address. For this transfer type, **DEVADDIS** is cleared, **ADDRDIS** may be set or cleared, and **NODATA** is set.

Figure 21-6. I²C Bus Address Only Transfer

If supported by the slave device, this transfer type may be used to initiate a random Read: an address-only Write cycle is issued (**DEVADDIS** is cleared, **ADDRDIS** is cleared, **NODATA** is set and **RWDIR** is cleared), followed by a burst Read with no address (**DEVADDIS** is cleared, **ADDRDIS** is set, **NODATA** is cleared and **RWDIR** is set).

Address Only with No Device Address

An address-only transfer may also be issued with only an address. For this transfer type, **DEVADDIS** is set, **ADDRDIS** may be set or cleared, and **NODATA** is set.

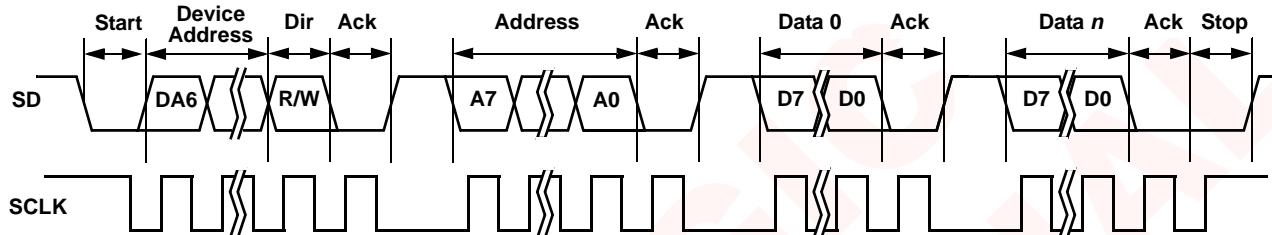
Figure 21-7. I²C Bus Address Only with No Device Address Transfer

Sequential Transfer

Sequential transfers include a device address and/or an address and multiple bytes of data. All of the above transfer types except Address-Only may be extended to burst transfers. For a burst transfer, [I2C_BYTE_CNT](#) is a non-zero value.

In this example, [DEVADDIS](#) is cleared, [ADDRDIS](#) is cleared, [NODATA](#) is cleared and [I2C_BYTE_CNT](#) is ‘n-1’.

Figure 21-8. Sequential Transfer



21.4.3.4 I²C Interrupts

If I²C interrupts are enabled by setting [I2C_CONFIG\[8\]](#) to ‘1’, then an I²C interrupt is generated when any one of the following conditions is true:

- A sequential write is occurring and the data register goes empty during a data phase
- A sequential read is being performed and the data register becomes full during a data phase
- When there is an attempted write to the [I2C_START](#) register while a transfer is already in progress
- When the host controller does not receive a proper acknowledgment from the I²C slave device after the transmission of a device address, an address, or data out
- If the I²C master loses arbitration for the I²C bus in a state other than an IDLE state. When this occurs, the state machine is reset to IDLE.
- On transition from a BUSY to an IDLE phase

A Programmable Interrupt Controller (PIC) must be set up to poll the I²C registers to determine the status and to initiate necessary action. In the PIC, this interrupt generation is configured as a Low-to-High transition. Thus, one interrupt is raised per event, and no clearing of the interrupt is needed. See the actions required to set and clear the interrupts in the [I2C_STATUS](#) register.

21.4.3.5 I²C System Issues

The I²C bus clock is assumed to be running at a slower speed than the system clock. In a typical system, the main system clock will be running at orders of magnitudes faster than the I²C bus clock. As a result, logic is built into the block to synchronize the two clock domains. Because the synchronization logic assumes the I²C bus clock, in order to work properly, the [I2C_DIVISOR](#) register MUST NOT be programmed to a value lower than eight. This I²C host controller also does not support multi-master arbitration.

21.5 UART and I²C Registers

21.5.1 UART and I²C Register Summary

There are twenty-one registers in the UART/I²C Interface sub-region as listed in [Table 21-4](#). They are divided by section for the UART and the I²C ports.

Register address offsets are given for byte addressing.

Table 21-4. Summary of UART and I²C Interface Registers

Register ID	Register Name	Description	R/W	Reset Value
Section 21.5.2, "UART Register Descriptions"				
0x00	UART_RX_DATA	UART Receiver Data register	RO	0x0000 0000
0x00	UART_TX_DATA	UART Transmitter Data register	WO	0x0000 0000
0x01	UART_INT_EN	UART Interrupt Enable register	R/W	0x0000 0000
0x02	UART_INT_ID	UART Interrupt Identification register	RO	0x0000 00C1
0x02	UART_FIFO_CTL.	UART FIFO Control register	WO	0x0000 00C0
0x03	UART_LINE_CTL.	UART Line Control register	R/W	0x0000 0003
0x04	UART_MODEM_CTL	UART Modem Control register	WO	0x0000 0000
0x05	UART_LINE_STS	UART Line Status register	RO	0x0000 0000
0x06	UART_MODEM_STS	UART Modem Status register	RO	0x0000 0000
0x00 ^a	UART_DIVISOR0	UART Clock Divisor LSB register	WO	0x0000 0000
0x01 ^a	UART_DIVISOR1	UART Clock Divisor MSB register	WO	0x0000 0000
Section 21.5.3, "I²C Host Controller Register Descriptions"				
0x00	I2C_CONFIG	I ² C Configuration register	R/W	0x0000 0000
0x01	I2C_DIVISOR	I ² C Clock Divisor register	R/W	0x0000 0000
0x02	I2C_DEVADDR.	I ² C Device Address register	R/W	0x0000 0000
0x03	I2C_ADDR	I ² C Address register	R/W	0x0000 0000
0x04	I2C_DOUT	I ² C Data Out register	R/W	0x0000 0000
0x05	I2C_DIN	I ² C Data In register	R/W	0x0000 0000
0x06	I2C_STATUS	I ² C Status register	R/W	0x0000 0000
0x07	I2C_START	I ² C Start Transfer register	R/W	0x0000 0000
0x08	I2C_BYTE_CNT	I ² C Transfer Byte Count register	R/W	0x0000 0000
0x09	I2C_HOLD	I ² C Hold Time register	R/W	0x0000 0000

a. Accessed with the use of the UART_LINE_CTL.DIV_ACCESS field.

21.5.2 UART Register Descriptions

There are eleven distinct registers at seven different offset addresses for each UART. The sub-region offset for UART 1 is 0x1 4000 and the sub-region offset for UART 2 is 0x1 5000.

21.5.2.1 UART_RX_DATA

UART Register ID: 0x00
UART Register Address Offset: 0x000

This Read Only portion of a Read/Write register contains the Receiver byte input Data. The value after Reset is 0x0000 0000.

		31:8	7:0	
RESERVED		RDATA		
Bits	Field Name	Field Description	R/W	Reset
31:8	RESERVED	RESERVED	RO	0x0000 00
7:0	RDATA	Data value from the Receiver FIFO output	RO	0x00

21.5.2.2 UART_TX_DATA

UART Register ID: 0x00
UART Register Address Offset: 0x000

This Write Only portion of a Read/Write register contains the Transmitter byte output Data. The value after Reset is 0x0000 0000.

		31:8	7:0	
RESERVED		WDATA		
Bits	Field Name	Field Description	R/W	Reset
31:8	RESERVED	RESERVED	RO	0x0000 00
7:0	WDATA	Data value for the Transmitter FIFO output	WO	0x00

21.5.2.3 UART_INT_EN

UART Register ID: 0x01
UART Register Address Offset: 0x004

This Read/Write register Enables UART Interrupts. The value after Reset is 0x0000 0000.

31:4	3	2	1	0
RESERVED	MODEM_STS_INT	RXLINE_STS_INT	TXDATA_EMTPY_INT	RXDATA_INT
Bits	Field Name	Field Description	R/W	Reset
31:4	RESERVED	RESERVED	RO	0x0000 0000
3	MODEM_STS_INT	Modem Status Interrupt: 0: Disabled 1: Enabled	R/W	b0

Bits	Field Name	Field Description	R/W	Reset
2	RXLINE_STS_INT	Receiver Line Status Interrupt: 0: Disabled 1: Enabled	R/W	b0
1	TXDATA_EMTY_INT	Transmitter Data Register Empty Interrupt 0: Disabled 1: Enabled	R/W	b0
0	RXDATA_INT	Receiver Data ready Interrupt: 0: Disabled 1: Enabled	R/W	b0

21.5.2.4 UART_INT_ID

UART Register ID: **0x02**

UART Register Address Offset: **0x008**

This Read Only portion of a Read/Write register Identifies UART Interrupts. The Read value after Reset is 0x0000 00C1.

31:4	RESERVED	3:1	0
	INT_ID	INT_STS	

Bits	Field Name	Field Description	R/W	Reset
31:4	RESERVED	RESERVED	RO	0x0000 00C
3:1	INT_ID	Interrupt Identification: 0: Modem Status 1: Transmitter Data Register Empty 2: Receiver Data Ready 3: Receiver Line Status 4-5: Invalid 6: Time-out Indication 7: Invalid	RO	b000
0	INT_STS	Interrupt pending Status: 0: Interrupt pending 1: No interrupt pending	RO	b1

Table 21-5 lists the possible interrupts along with their priority, their sources and reset control.

Table 21-5. Interrupt Identification Properties

INT_ID Value	Interrupt Identification	Interrupt Source	Interrupt Reset	Priority
0	Modem Status	CTS, DSR, RI or DCD	Reading UART_MODEM_STS	4th
1	Transmitter Data Register Empty	Transmitter data register empty	Writing to the Transmitter Data Register or reading UART_INT_ID	3rd
2	Receiver Data Ready	FIFO Full level reached	FIFO drops below Full level	2nd

Table 21-5. Interrupt Identification Properties

INT_ID Value	Interrupt Identification	Interrupt Source	Interrupt Reset	Priority
3	Receiver Line Status	Parity, Overflow or Framing Errors or Break condition	Reading UART_LINE_STS	1st
6	Time-out Indication	At least one character in the Receiver FIFO, but no character has been input to the FIFO or Read from it for the last four character times	Reading from the Receiver FIFO	2nd

21.5.2.5 UART_FIFO_CTL.**UART Register ID:** **0x02****UART Register Address Offset:** **0x008**

This Write Only portion of a Read/Write register provides UART FIFOs Control. The Write value after Reset is 0x0000 00C0

31:8	7:6	5:3	2	1	0
RESERVED	RX_FIFO_FULL	RESERVED	TX_FIFO_RS	RX_FIFO_RS	RESERVED

Bits	Field Name	Field Description	R/W	Reset
31:8	RESERVED	RESERVED	RO	0x0000 00
7:6	RX_FIFO_FULL	Receiver FIFO Full interrupt level: 00: One byte 01: Four bytes 10: Eight bytes 11: Fourteen bytes	WO	b11
5:3	RESERVED	RESERVED	WO	b00 0
2	TX_FIFO_RS	1: Setting this bit clears the Transmitter FIFO and Resets logic. Output shift register is not cleared and outputting of current character continues.	WO	b0
1	RX_FIFO_RS	1: Setting this bit clears the Receiver FIFO and Resets logic. Input shift register is not cleared and inputting of current character continues.	WO	b0
0	RESERVED	RESERVED	WO	b0

21.5.2.6 UART_LINE_CTL.

UART Register ID: 0x03
UART Register Address Offset: 0x00C

31:8	7	6	5	4	3	2	1:0
RESERVED	DIV_ACCESS	BREAK_CTL	SPARITY_EN	EPARITY_EN	PARITY_EN	STOP_BITS	CHAR_BITS

This Read/Write register provides UART Line Control. The value after Reset is 0x0000 0003

Bits	Field Name	Field Description	R/W	Reset
31:8	RESERVED	RESERVED	RO	0x0000 00
7	DIV_ACCESS	Divisor latch Access: 0: Normal registers are accessed 1: Divisor latches can be accessed	R/W	b0
6	BREAK_CTL	Break Control: 0: Break is disabled 1: Serial output is forced into logical '0' for Break condition	R/W	b0
5	SPARITY_EN	Sticky Parity: 0: Sticky Parity disabled 1: If Parity and Even Parity are enabled, the parity bit is transmitted and checked for a logical '0'. If Parity and Odd Parity are enabled then the parity bit is transmitted and checked for a logical '1'.	R/W	b0
4	EPRORITY_EN	Even Parity Enable: 0: Odd Parity enabled 1: Even Parity enabled	R/W	b0
3	PARITY_EN	Parity Enable: 0: No parity enabled 1: Parity bit is generated on each outgoing character and is checked on each incoming one	R/W	b0
2	STOP_BITS	Number of generated Stop Bits: 0: One stop bit 1: Two stop bits or 1.5 stop bits when 5-bit character length Note that the receiver always checks only the first stop bit.	R/W	b0
1:0	CHAR_BITS	Bits per Character: 00: Five bits 01: Six bits 10: Seven bits 11: Eight bits	R/W	b11

21.5.2.7 UART_MODEM_CTL

UART Register ID: **0x04**
UART Register Address Offset: **0x010**

This Write Only register provides UART Modem Control. The Write value after Reset is 0x0000 0000

31:5	4	3	2	1	0
RESERVED	LOOP_EN	OUT2	OUT1	RTS	DTR

Bits	Field Name	Field Description	R/W	Reset
31:5	RESERVED	RESERVED	RO	b0000 0000 0000 0000 0000 000
4	LOOP_EN	Loopback test mode Enable: 0: Normal operation 1: Loopback mode enabled In loopback mode the serial output signal is set to a logical one. The output of the transmitter shift register is internally connected to the input of the receiver shift register and the following connections are made: DTR -> DSR, RTS -> CTS, Out1 -> RI and Out2 -> DCD.	WO	b0
3	OUT2	Out2 signal. In loopback mode connected to Data Carrier Detect (DCD) signal input.	WO	b0
2	OUT1	Out1 signal. In loopback mode connected to Ring Indicator (RI) signal input.	WO	b0
1	RTS	Request To Send (RTS) control signal: 1: RTS is a logical one 0: RTS is a logical zero	WO	b0
0	DTR	Data Terminal Ready (DTR) control signal: 1: DTR is a logical one 0: DTR is a logical zero	WO	b0

21.5.2.8 UART_LINE_STS

UART Register ID: **0x05**
UART Register Address Offset: **0x014**

This Read Only register provides UART Line Status. The value after Reset is 0x0000 0000.

31:8	7	6	5	4	3	2	1	0
RSVD	ERROR	TX_EMPTY	TX_FIFO_EMPTY	BREAK	FRAME_ERR	PARITY_ERR	OVFL_ERR	DATA_RDY

Bits	Field Name	Field Description	R/W	Reset
31:8	RESERVED	RESERVED	RO	0x0000 00
7	ERROR	0: No parity, framing or break error condition exists 1: At least one parity error, framing error or break condition has been received and in the FIFO. This bit is cleared upon reading from the register.	RO	b0
6	TX_EMPTY	Transmitter Empty status. 0: Transmitter FIFO and shift register are both not empty. 1: Both the transmitter FIFO and transmitter shift register are empty. This bit is cleared when data is being written to the transmitter FIFO.	RO	b0
5	TX_FIFO_EMPTY	Transmitter FIFO is Empty status: 0: Transmitter FIFO not empty 1: Transmitter FIFO is empty. Generates Transmitter Data Register Empty interrupt. This bit is cleared when data is being written to the transmitter FIFO.	RO	b0
4	BREAK	Break condition status: 0: No break condition in the current character 1: A break condition in the current character. Break occurs when line is held at logical '0' for a time of one character (start bit + data + parity + stop bit). Then one zero character enters FIFO and UART waits for a valid start bit to receive next character. This bit is cleared upon reading from the register. Generates Receiver Line Status interrupt.	RO	b0
3	FRAME_ERR	Framing Error status: 0: No framing error in the current character 1: The received character at the top of the FIFO did not have a valid stop bit. The following data maybe corrupt. This bit is cleared upon reading from the register. Generates Receiver Line Status interrupt.	RO	b0
2	PARITY_ERR	Parity Error status: 0: No parity error in the current character 1: The character that is currently at the top of the FIFO has been received with a parity error. This bit is cleared upon reading from the register. Generates a Receiver Line Status interrupt.	RO	b0
1	OVFL_ERR	Overflow Error status: 0: No overflow state 1: When Receiver FIFO is Full and another character is received in the Receiver shift register. If another character starts to arrive, it will overwrite the data in the shift register but the FIFO will remain intact. This bit is cleared upon reading from the register. Generates a Receiver Line Status interrupt.	RO	b0
0	DATA_RDY	Data Ready status: 0: No characters in the Receiver FIFO 1: At least one character has been received and is in the FIFO.	RO	b0

21.5.2.9 UART_MODEM_STS

UART Register ID: **0x06**
UART Register Address Offset: **0x018**

This Read Only register provides UART Modem Status. The value after Reset is 0x0000 0000.

31:8	7	6	5	4	3	2	1	0
RESERVED	DCD	RI	DSR	CTS	DDCD_STS	TERI_STS	DDSR_STS	DCTS_STS

Bits	Field Name	Field Description	R/W	Reset
31:8	RESERVED	RESERVED	RO	0x0000 00
7	DCD	DCD input or equal to Out2 in loopback mode	RO	b0
6	RI	RI input or equal to Out1 in loopback mode	RO	b0
5	DSR	DSR input or equal to DTR in loopback mode	RO	b0
4	CTS	CTS input or equal to RTS in loopback mode.	RO	b0
3	DDCD_STS	Delta Data Carrier Detect (DDCD) Status: 0: DCD line has not changed state 1: DCD line has changed its state Clears itself upon being read.	RO	b0
2	TERI_STS	Trailing Edge of Ring Indicator (TERI) detector Status: 0: RI line has not changed its state from low to high 1: RI line has changed its state from low to high Clears itself upon being read.	RO	b0
1	DDSR_STS	Delta Data Set Ready (DDSR) Status: 0: DSR line has not changed its state 1: DSR line has changed its state Clears itself upon being read.	RO	b0
0	DCTS_STS	Delta Clear To Send (DCTS) Status: 0: CTS line has not changed its state 1: CTS line has changed its state Clears itself upon being read.	RO	b0

21.5.2.10 **UART_DIVISOR0**

UART Register ID: 0x00
UART Register Address Offset: 0x000

This Write Only register contains the lower-order byte of the 16-bit UART clock Divisor. The two bytes form one 16-bit value equal to the system clock frequency divided by 16 times the desired baud rate. The internal counter starts when the LSB is written, so the MSB should be written first and the LSB last.

Note that [UART_LINE_CTL..DIV_ACCESS](#) must be set to write to this register at this offset. The value after Reset is 0x0000 0000.

		31:8	7:0	
		RESERVED	DLATCH0	
Bits	Field Name	Field Description	R/W	Reset
31:8	RESERVED	RESERVED	RO	0x0000 00
7:0	DLATCH0	LSB of the UART clock divisor	WO	0x00

21.5.2.11 **UART_DIVISOR1**

UART Register ID: 0x01
UART Register Address Offset: 0x004

This Write Only register contains the higher-order byte of the 16-bit UART clock Divisor. Note that [UART_LINE_CTL..DIV_ACCESS](#) must be set to Write to this register at this offset. The value after Reset is 0x0000 0000.

		31:8	7:0	
		RESERVED	DLATCH1	
Bits	Field Name	Field Description	R/W	Reset
31:8	RESERVED	RESERVED	RO	0x0000 00
7:0	DLATCH1	MSB of the UART clock divisor	WO	0x00

21.5.3 I²C Host Controller Register Descriptions

There are ten distinct registers at ten different offset addresses for each I²C Port. The sub-region offset for I²C Port 1 is 0x1 6000 and the sub-region offset for I²C Port 2 is 0x1 7000.

21.5.3.1 I²C_CONFIG

I²C Register ID: 0x00
I²C Register Address Offset: 0x000

This Read/Write register Configures the I²C port. The value after Reset is 0x0000 0000

31:8	7:5	4:2	1	0		
RESERVED		ADDRLEN	DEVADLEN	ADDRDIS	DEVADDIS	
Bits	Field Name	Field Description			R/W	Reset
31:9	RESERVED	RESERVED			RO	0x0000 0
8	INTR_EN	Interrupt Enable: I ² C interrupts are enabled by setting this bit. See I²C_STATUS register for the source of the interrupts. 0: Interrupts disabled 1: Interrupts enabled			R/W	b0
7:5	ADDRLEN	Address Length. Selects number of data address bits to be transferred from I²C_ADDR register: 0: One-bit address 1: Two-bit address, etc.			R/W	b000
4:2	DEVADLEN	Device Address Length. Selects number of device address bits to be transferred from I²C_DEVADDR . register: 0: One-bit address 1: Two-bit address, etc. Should be programmed to b110 for compliance with I ² C bus protocol.			R/W	b0 00
1	ADDRDIS	Address Disable: 0: Normal transfers occur with data address followed by read or write data 1: Read or write serial data without transferring the address			R/W	b0
0	DEVADDIS	Device Address Disable: 0: Device address transmitted before the data address 1: Device address not transferred Note: If this bit is set, the ADDRDIS bit is ignored, and a data address is always transmitted. Note: Most I ² C slave devices require a device address to be transmitted so this bit is typically cleared.			R/W	b0

21.5.3.2 I²C_DIVISOR

I²C Register ID: 0x01
I²C Register Address Offset: 0x004

This Read/Write register sets the clock Divisor for the I²C port. The value after Reset is 0x0000 0000.

31:16	15:0
RESERVED	CLKDIV

Bits	Field Name	Field Description	R/W	Reset
31:16	RESERVED	RESERVED	RO	0x0000
15:0	CLKDIV	SCLK Divisor. Value determines the I ² C SCLK signal frequency where: $\text{CLKDIV} = (33 \text{ MHz}/f_{\text{SCLK}}) - 2.5$ Note: Only values of eight and higher are valid. Note: For most systems CLKDIV is large so that the SCLK frequency can be approximated as: $\text{CLKDIV} \cong 33 \text{ MHz}/f_{\text{SCLK}}$	R/W	0x0000

21.5.3.3 I²C_DEVADDR.

I²C Register ID: 0x02
I²C Register Address Offset: 0x008

This Read/Write register sets the sent Device Address for the I²C port which is the address of the external device to be communicated with. The value after Reset is 0x0000 0000

31:7	6:0
RESERVED	DEVADDR

Bits	Field Name	Field Description	R/W	Reset
31:7	RESERVED	RESERVED	RO	b0000 0000 0000 0000 0000 0000 0
6:0	DEVADDR	I ² C Device Address. This value is transmitted as the device address if I²C_CONFIG.DEVADDIS is not set.	R/W	b000 0000

21.5.3.4 I²C_ADDR

I²C Register ID: **0x03**
I²C Register Address Offset: **0x00C**

This Read/Write register sets the transmitted data Address from the I²C port for use by the external device. The value after Reset is 0x0000 0000.

		31:8	7:0		
RESERVED			ADDR		
Bits	Field Name	Field Description		R/W	Reset
31:8	RESERVED	RESERVED		RO	0x0000 00
7:0	ADDR	Address sent to external I ² C slave devices when I²C_CONFIG.ADDRDIS is cleared.		R/W	0x00

21.5.3.5 I²C_DOUT

I²C Register ID: **0x04**
I²C Register Address Offset: **0x010**

This Read/Write register sets the transmitter Output Data for the I²C port. The value after Reset is 0x0000 0000.

		31:8	7:0		
RESERVED			DATAOUT		
Bits	Field Name	Field Description		R/W	Reset
31:8	RESERVED	RESERVED		RO	0x0000 00
7:0	DATAOUT	Data of 8-bits to be written to external I ² C slave devices during a Write transfer		R/W	0x00

21.5.3.6 I²C_DIN

I²C Register ID: **0x05**
I²C Register Address Offset: **0x014**

This Read/Write register contains the received Input Data for the I²C port. The value after Reset is 0x0000 0000.

		31:8	7:0		
RESERVED			DATAIN		
Bits	Field Name	Field Description		R/W	Reset
31:8	RESERVED	RESERVED		RO	0x0000 00
7:0	DATAIN	Data of 8-bits received from the external I ² C slave devices during a Read transaction. I²C_STATUS.DATARDY is set when data is valid in this register.		R/W	0x00

21.5.3.7 I²C_STATUS

I²C Register ID: 0x06
I²C Register Address Offset: 0x018

This Read Only register indicates the I²C port Status. The value after Reset is 0x0000 0000.

31:5	4	3	2	1	0
RESERVED	STARTERR	ACKERR	DATARDY	SDOEMPTY	BUSY

Bits	Field Name	Field Description	R/W	Reset
31:8	RESERVED	RESERVED	RO	b0000 0000 0000 0000 0000 0000
7	INTR	Interrupt: This bit is asserted if I ² C_CONFIG[8] is enabled and any bit of I ² C_STATUS[5:0] is set. This bit clears automatically when the corresponding action is taken as required in Section 21.4.3.4 .	RO	b0
6	RESERVED	RESERVED	RO	b0
5	LOST_ARB	Lost Arbitration for Bus: This bit is set if the I ² C master loses arbitration for the I ² C bus in a state other than an IDLE state. When this occurs, the state machine is reset to IDLE. The I ² C transaction must be re-tried by user software. 0: The I ² C master has not lost arbitration for the bus in a non-IDLE state. 1: The I ² C master has lost arbitration for the bus in a non-IDLE state.	RO	b0
4	STARTERR	Start Error. 0: Cleared when software writes to I ² C_START while BUSY is cleared. 1: Set when the I ² C_START register is written to while a transfer is already in progress. When this occurs the Write is ignored.	RO	b0
3	ACKERR	Acknowledgement Error. 0: Cleared when software writes to the I ² C_START register while BUSY is cleared. 1: Set when the host controller does not receive a proper acknowledge from the I ² C slave device after the transmission of a device address, an address, or data out.	RO	b0
2	DATARDY	Data Ready. Indicates the receiver register contains valid data. 0: Cleared when software reads the DIN register. 1: Set when data is received from an I ² C slave device and is transferred from the interface shift register to the I ² C_DIN register.	RO	b0

Bits	Field Name	Field Description	R/W	Reset
1	SDOEMPTY	SDO Empty. Indicates the transmitter data register is empty. 0: Cleared when the I2C_DOUT register is written to by software. 1: set when transmit data is transferred from the register to the interface shift register. Software may write to I2C_DOUT register when this bit is set.	RO	b0
0	BUSY	Port is Busy. 0: Cleared when it is idle. Software may initiate an I ² C transfer when cleared and should not modify any I ² C host controller registers while it is set. 1: Set when the I ² C interface is active.	RO	b0

21.5.3.8 I2C_START

I²C Register ID: 0x07

I²C Register Address Offset: 0x01C

This Read/Write register Starts I²C transfers and provides control/status for the I²C port. Writing to this register, regardless of the value, starts an I²C transfer. The value after Reset is 0x0000 0000.

31:2			1	0
RESERVED		NODATA	RWDIR	

Bits	Field Name	Field Description	R/W	Reset
31:2	RESERVED	RESERVED	RO	b0000 0000 0000 0000 0000 0000 0000 00
1	NODATA	No Data transfer, address only. I2C_CONFIG.ADDRDIS is ignored if this bit is set for a transaction. 1: Setting this bit initiates an address-only transaction. If I2C_CONFIG.DEVADDIS is cleared the device address, direction, data address and stop condition are transmitted to the I ² C slave device. If DEVADDIS is set, only the data address and stop condition are transmitted. 0: This bit should be cleared for normal I ² C bus accesses.	R/W	b0
0	RWDIR	Read/Write Direction. 0: Clearing this bit initiates a Write transaction. 1: Setting this bit initiates a Read transaction. This bit is shifted out to the I ² C slave device after the device address unless I2C_CONFIG.DEVADDIS is set.	R/W	b0

21.5.3.9 I²C_BYTE_CNT

I²C Register ID: 0x08
I²C Register Address Offset: 0x020

This Read/Write register determines the data transfer Byte Count for the I²C port. The value after Reset is 0x0000 0000.

31:6			5:0	
RESERVED			BYTE_CNT	
Bits	Field Name	Field Description	R/W	Reset
31:6	RESERVED	RESERVED	RO	b0000 0000 0000 0000 0000 0000 00
5:0	BYTE_CNT	Byte Count. The value written to this register plus one indicates the number of data bytes to be written to or read from the external I ² C slave device. If its value is non-zero, multiple sequential Read or Write cycles will be issued with a single address and/or device address. Note that most devices are not capable of sequential transfers.	R/W	b00 0000

21.5.3.10 I²C_HOLD

I²C Register ID: 0x09
I²C Register Address Offset: 0x024

This Read/Write register determines the Hold time limit between a Start and Stop condition for the I²C port. The value after Reset is 0x0000 0000.

31:8			7:0	
RESERVED			HDSTATIM	
Bits	Field Name	Field Description	R/W	Reset
31:10	RESERVED	RESERVED	RO	0x0000 00
9:0	HDSTATIM	Hold Start Time. The value written to this register is used to determine the bus free time between a STOP and START condition. This is the timing parameter t _{BUF} where: $t_{BUF} = 15 \text{ ns} \times (HDSTATIM + 1)$ t _{BUF} is defined as 4.7μs in I ² C Fast mode.	R/W	b00 0000 0000



Chapter 22 Flash/PCMCIA Memory and the Peripherals Interface

22.1 Introduction

This chapter discusses Flash Memory basics, and connection NOR or NAND Flash Memory devices and PCMCIA CompactFlash devices to the Peripheral Input/Output Interface.

22.2 Technology: NOR Flash vs. NAND Flash

There are similarities and differences between NOR and NAND Flash memory devices. Similarities include the following:

- **Data Volatility:** Both NOR and NAND Flash devices are characterized as non-volatile storage.
- **Boot Applications:** Although normally boot Flash devices are NOR devices, both types can be used to boot with proper processor support. (The XLS Processor family has hardware support to allow using a NAND Flash device as a boot device.)
- **Erased vs. Programmed Bits:** In both NOR and NAND Flash devices, an erased bit contains a logical '1'. A logical '0' must be programmed.

Differences include the following:

- **Density:** NAND Flash devices typically have a higher density than NOR Flash devices—greater storage capacity.
- **Direct vs. Indirect Access:** A byte in a NOR Flash device is directly accessible, and the bytes may be accessed in any order for read or write. With NAND Flash, a page is addressed, and then the bytes on that page are accessed (indirect access).
- **Pin Count:** In NOR Flash, because every byte is directly accessible, there are separate non-multiplexed address and data pins. NAND Flash devices multiplex address and data on the same bus, thus requiring fewer pins for comparable density devices.
- **Erasing Data:** NOR Flash data is erased by sectors. NAND Flash data is erased by blocks.
- **Bit Errors:** All bits in a NOR Flash device are functional. NAND Flash devices, unless pre-sorted, are not usually guaranteed to be error free, and an error correcting code is used to recover from errors and maintain data integrity. Also, each NAND device ships with a list of bad blocks, just like a disk drive, and the user's software must manage the bad blocks in the NAND device.
- **Boot Applications:** Traditionally, NOR Flash devices have been used when a boot device is needed, since NOR Flash is directly and linearly addressable, and thus permits random access. NAND Flash devices, since their bytes are not accessed randomly and since their access addresses are not always contiguous, are not used for boot applications unless the device to be booted has special hardware support (as does the XLS Processor Family).
- **Access Speed Limitations:** The main performance disadvantages of NOR Flash are slow write times and slow erase times. The main performance disadvantage of NAND Flash is slow random read access. (However, once the first byte on a page is available

from a NAND Flash device, successive bytes on that page are available much more quickly than bytes read from a NOR Flash device).

22.3

NAND Flash Basics

A NAND Flash is organized as a series of blocks. Individual bytes may be programmed within a block, but a complete block is the smallest entity that can be erased. Erasing a block causes all bits to be set to logical '1'. There is a finite limit to the number of times a NAND block can be erased and programmed. A technique called "wear leveling" is used to ensure that no one block is erased more than others.

Data is stored in a NAND Flash device in a way similar to how data is stored on a magnetic disk drive. Where hard disks use sectors, NAND Flash devices use "pages." A 2GB NAND Flash device would be organized as 2048 blocks, with 64 pages per block. Each page is 2,112 bytes, consisting of 2,048 data bytes and 64 spare bytes (which are used for ECC, wear-leveling, defect management, and other software functions).

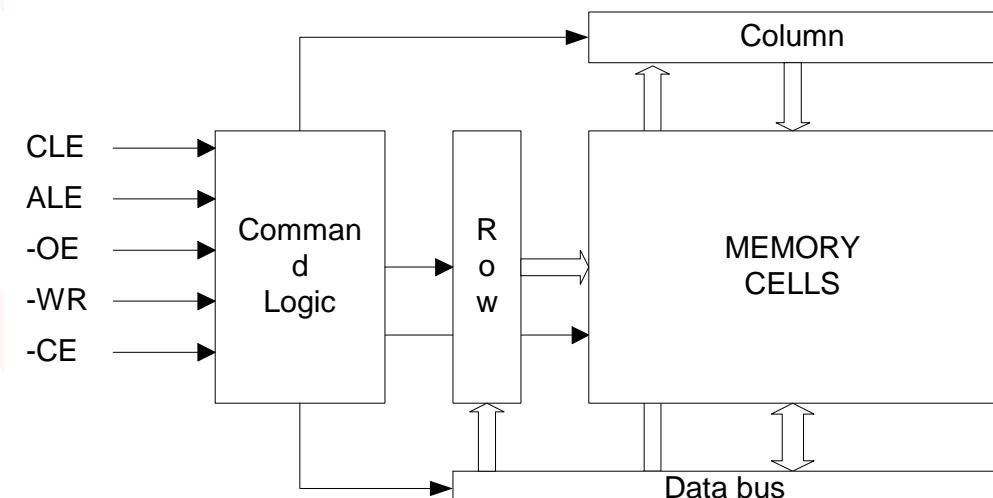
NAND Flash devices are offered with either an 8-bit interface or a 16-bit interface, representing an 8-bit data bus or a 16-bit data bus to the device. For 16-bit devices, NAND commands and addresses use the lower 8 bits, and the upper 8 bits are used only during data transfer.

There are only three basic operations for NAND Flash:

- Read a page
- Program a page
- Erase a block

Accessing NAND Flash is done via indirect access. Unlike NOR Flash, where every byte is randomly accessed and each byte or word has a unique address, NAND Flash—because of its much larger capacity—needs a special protocol. Prior to accessing a byte or word, the address must be loaded into the internal address registers (which can be as large as 34 bits for a 16GB device). This is accomplished by utilizing a command/address interface. [Figure 22-1](#) describes a typical NAND Flash structure.

Figure 22-1. NAND Flash Memory Device Block Diagram



A NAND Flash device accepts operation command inputs on the ALE and CLE input pins, the states of which are decoded to specify the operation requested. The NAND Flash command logic responds to the following two-signal commands:

<u>ALE</u>	<u>CLE</u>	<u>Command</u>
0	0	Access DATA register
0	1	Access CMD register
1	0	Access ADDRS register
1	1	Reserved

22.4 Interfacing FLASH Memory to the XLS

Flash Memory devices are attached externally using signals of the Peripherals I/O Bus and the General-Purpose Input/Output (GPIO) interface. [Figure 22-1](#) shows conceptually how the various types of FLASH Memory devices are attached.

22.4.1 FLASH Memory on the Peripherals Input/Output Bus

The Peripherals Input/Output Bus is a generic 32-bit bus interface supporting 8/16/32-bit devices. The interface can be asynchronous or synchronous (66.67 MHz or 33.33 MHz). It supports NOR/NAND Flash ROM and PCMCIA CompactFlash.

The Peripheral Input/Output bus is used to connect to external devices, which appear in the address space as memory-mapped devices. These devices could be FPGAs with memory-mapped registers, memory devices that present a large array of addressable locations for data or parameter storage, or any other external I/O device that has a memory-mapped interface.

The device connected to the Peripheral I/O interface will typically be a Flash Memory device, used for sizable amounts of non-volatile storage (megabytes or gigabytes). The Flash device can be NOR or NAND Flash used as a boot ROM or for other purposes. (Compact Flash can also be applied, but it connects to the PCMCIA interface.)

The Peripheral I/O bus controller capabilities are as follows:

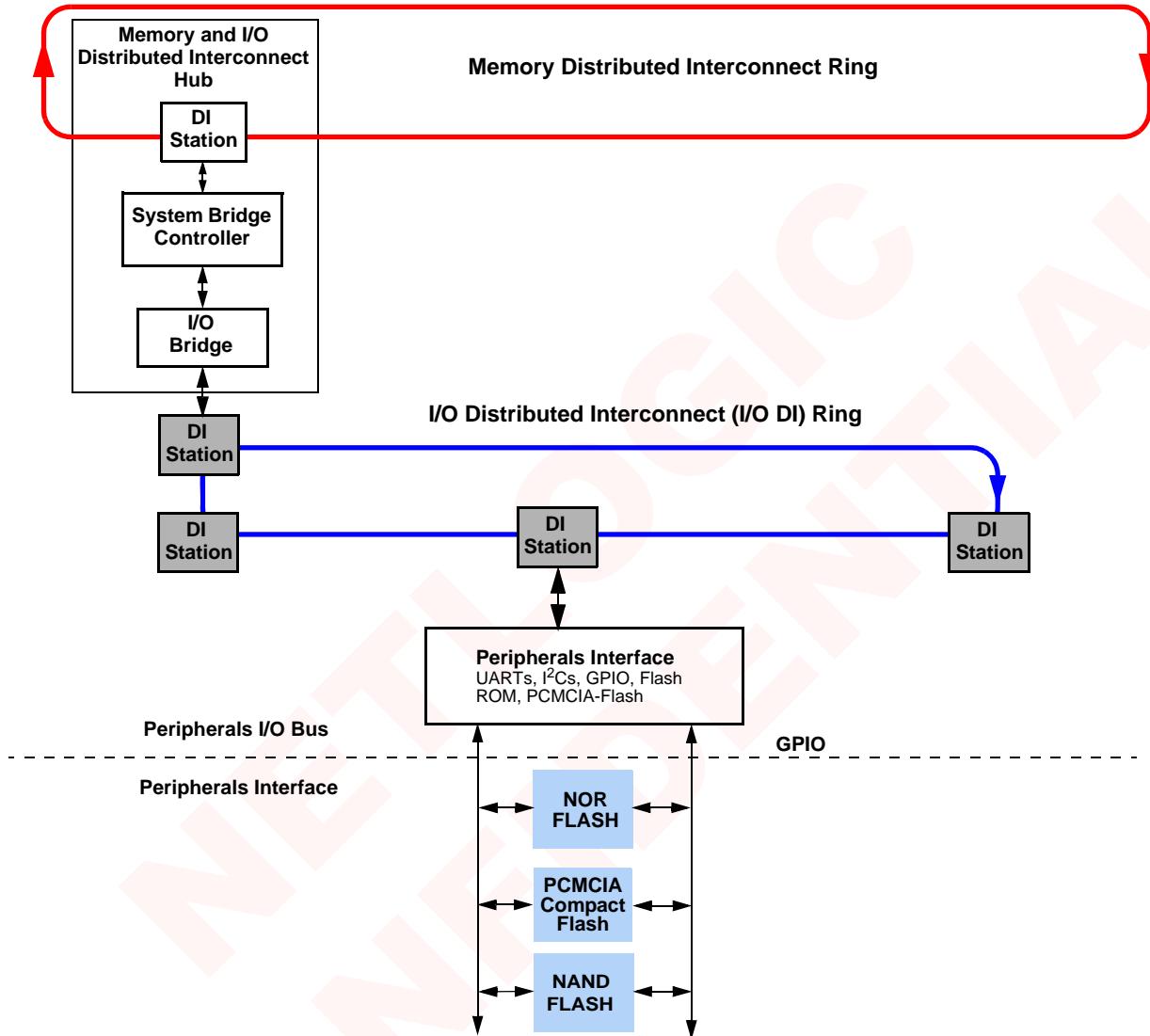
- Supports ten chip-select areas
- Each chip-select area has its own set of base address, address mask, device and timing parameter registers
- Supports multiplexed mode for Address/Data
- Supports burst access in Synchronous mode (using DMA engine)
- Supports the IO_WAIT_L signal for extended bus cycles
- Data parity generation/error detection supported
- Interrupt raised for error conditions like data parity error, illegal address, multiple CS and Wait time-outs

Nine of the ten chip-select areas are used for generic bus access. One chip-select area (CS[0]) is used only for boot flash. One other chip-select area is used for PCMCIA CompactFlash (CS[6]). Any chip-select area from CS[1] to CS[9]) can be used for any non-boot PCMCIA function.

The data width of each chip-select region can be configured as 8-bit, 16-bit, or 32-bit. When access is made to a narrower region, the controller automatically converts the bigger read/write request into multiple accesses.

For each of the ten address areas of the ten chip selects, the functional behavior and timing may be different.

Figure 22-2. Interfacing Flash Memory to the XLS



The various bus activity modes that may be selected for a given chip-select area are shown in **Table 22-1**. Note that some bus modes are possible with multiple Flash device types (NOR, NAND, PCMCIA, CF).

Table 22-1. External Device Connections to Peripheral IO Bus (Bus Functional Modes)

Bus Activity Mode of Attached Device	Connect to these XLS Signals								
	Address (i.e., while ALE = 1)		Data (i.e., while ALE = 0)					Control	
	IO_AD[...]		IO_AD[...] ¹				Parity	Compact FLASH Card Enables	
	[31:28]	[27:0]	[31:24]	[23:16]	[15:8]	[7:0]	IO_ADP[3:0] ⁴	CF_CE2_L ⁵	CF_CE1_L ⁶
Non-multiplexed			Att. device D[7:0]	Attached device Addr[23:0]		Attached device BP3_L			
8-bit Big-Endian Data	Attached device Addr[31:28]	Attached device Addr[27:0]	Att. device D[7:0]			Attached device BP3_L			
16-bit Big-Endian Data			Attached device D[15:0]				Attached device BP[3:2]_L		
32-bit Big-Endian Data			Attached device D[31:0]				Attached device BP[3:0]_L		
8-bit Little-Endian Data ²			Att. device D[7:0]			Attached device BP3_L	Logic HIGH (1)	Logic LOW (0)	
16-bit Little-Endian Data ²			Att. device D[7:0]	Att. device D[15:8]			Attached device BP[3:2]_L	Logic LOW (0)	Logic LOW (0)
Generic Interface	Attached device BE[3:0] ³		Attached device D[31:0]				Attached device BP[3:0]_L		

Note 1. The XLS IO_AD bus always operates in Big-Endian mode.

Note 2. The PCMCIA and Compact FLASH Standards specify Little-Endian operation. Likewise, most FLASH memory devices operate in Little-Endian mode.

Note 3. BE[3:0] are the attached device's Byte Enable signals, relating to bytes Addr[31:24], Addr[23:16], Addr[15:8], and Addr[7:0], respectively.

Note 4. IO_ADP[3:0] are the XLS Byte Parity signals for bytes IO_AD[31:24], IO_AD[23:16], IO_AD[15:8], and IO_AD[7:0], respectively. They share pins with GPIO[13:10]. (See [Section 22.4.2.1](#) for more information).

Note 5. CF_CE2_L is the XLS Compact FLASH Card Enable 2 signal. It shares a pin with GPIO2.

Note 6. CF_CE1_L is the XLS Compact FLASH Card Enable 1 signal. It shares a pin with GPIO1.

The Peripherals I/O Bus is composed of the forty-six signals listed in [Table 22-2](#).

Table 22-2. Peripherals Input/Output Bus Port Signal Descriptions

Signal Name	Signals	I/O	Description
IO_AD[31:0]	32	I/O	Multiplexed Address/Data bus
IO_OE_L	1	Out	Output Enable in Read operation
IO_WAIT_L	1	In	Wait for data in Read operation
IO_ALE	1	Out	Address Latch Enable
IO_WE_L	1	Out	Write Strobe
IO_CS[9:0]_L	10	Out	Chip Select

The IO_AD signals are also used at Reset time to determine some initial reset states for the system configuration.

22.4.2 Flash Memory Devices and GPIO Port Signals

The twenty-five signals of the General-Purpose Input/Output port are programmable and all but seven can serve multiple functions, including generating interrupts to the XLS Cores. Fourteen of the signals are used by the internal PCMCIA controller when it is used with the Peripherals I/O Bus to support PCMCIA. Six of the signals are used to support Boot Flash Memories on the Peripheral I/O Bus. The XLS can boot from NOR, NAND, or linear PC Card Type 1 Flash.

Most GPIO signals can be configured as inputs, outputs or interrupt inputs (edge or level sensitive). Each GPIO pin has its own direction control, interrupt enable and input and output data bit registers. All alternate signal names and their GPIO connections are given in [Table 22-3](#). (See also [Section 23.3 GPIO Registers](#) for details on GPIO configuration.)

Table 22-3. GPIO Port Signals and Flash Memory

Signal Name		Signals	GPIO / Alternate I/O	GPIO / Alternate Description
GPIO	Alternate			
GPIO0	CF_REG_L	1	I/O / O	GPIO (interrupt input) / CompactFlash Register selects attribute memory
GPIO1	CF_CE1_L	1	I/O / O	GPIO (interrupt input) / CompactFlash Card Enable 1
GPIO2	CF_CE2_L	1	I/O / O	GPIO (interrupt input) / CompactFlash Card Enable 2
GPIO3	CF_CD1_L	1	I/O / I	GPIO (interrupt input) / CompactFlash Card Detect 1
GPIO4	CF_CD2_L	1	I/O / I	GPIO (interrupt input) / CompactFlash Card Detect 2
GPIO5	CF_READY	1	I/O / I	GPIO (interrupt input) / CompactFlash Ready if PCMCIA is enabled. Based on how the PCMCIA device/interface is defined, memory mode or I/O mode, this would be either a READY or an IREQ_L signal. Can cause interrupt: see FLASH_INT_STATUS and FLASH_INT_MASK .
GPIO6	CF_WP	1	I/O / I	GPIO (interrupt input) / CompactFlash Write Protect if PCMCIA is enabled. Based on how the PCMCIA device/interface is defined, memory mode or I/O mode, this would be either a Write Protect or an IOIS16_L signal. Can cause interrupt: see FLASH_INT_STATUS and FLASH_INT_MASK .
GPIO7	CF_BVD1	1	I/O / I	GPIO (interrupt input) / CompactFlash Battery Voltage Detect 1
GPIO8	CF_BVD2	1	I/O / I	GPIO (interrupt input) / CompactFlash Battery Voltage Detect 2

Table 22-3. GPIO Port Signals and Flash Memory (continued)

Signal Name		Signals	GPIO / Alternate I/O	GPIO / Alternate Description
GPIO	Alternate			
GPIO9	CF_RESET	1	I/O / O	GPIO (interrupt input) / CompactFlash Reset If PCMCIA is enabled (via IO_AD[17] at POWER_GOOD =1) then this pin is controlled by the FLASH_INT_MASK register bit field (2): RESET. Also note that If PCMCIA_EN =1 and BOOT_SELECT=PCMCIA at POWER_GOOD, then boot will occur from CS6 (i.e., PCMCIA).
GPIO[13:10]	IO_AD[3:0]	4	I/O / I/O	GPIO (interrupt input) / I/O bus Address and Data Parity (see also Section 22.4.2.1)
GPIO14	FLASH_CPUID0	1	O	Flash memory CPU ID 0 always ^a
GPIO15	FLASH_CPUID1	1	O	Flash memory CPU ID 1 always ^a
GPIO16	FLASH_CPUID2	1	O	Flash memory CPU ID 2 always ^a
GPIO17	FLASH_RDY	1	I	Flash memory Ready always. Connected to Ready/Busy status pins of the boot device or other devices connected to the IO Bus. Asserted when the device is busy with Write/Erase operations and unasserted when done. Can cause interrupt: see FLASH_INT_STATUS and FLASH_INT_MASK .
GPIO18	FLASH_ADV	1	O	Flash memory Address Valid always. Connected to ADV pin of AMD flash memories (ONLY AMD memories). Will be asserted during burst transfers (both Read and Write) with AMD memories to indicate that the burst starting address is present on the IO_AD bus.
GPIO19	FLASH_RESET_N	1	O	Flash memory Reset always. Connect to the Reset pin of boot flash or other devices on the IO Bus to restore the device to a known state.
GPIO20	-	1	I/O	General Purpose I/O (interrupt input)
GPIO21	-	1	I/O	General Purpose I/O (interrupt input)
GPIO22	-	1	I/O	GPIO
GPIO23	THERMAL_INTRPT	1	I	Thermal Interrupt always
GPIO24	-	1	I/O	General Purpose I/O (interrupt input)

a. The CPU ID reflects the CPU ID number of the CPU that is currently accessing the flash device.

22.4.2.1 **GPIO[13:10] Dual Definition Restrictions**

GPIO[13:10] have dual definitions:

- Interrupt inputs
- I/O Address and Data Parity

If any of the following conditions are true:

- GPIO Reset Configuration register bit [12] is set by software (selecting Parity)
- IO_AD19 is strapped HIGH at reset time (selecting Parity)
- FLASH_CSDEV_PARM[n] bit [10] is set (for any chip select region n), selecting a Generic Interface

then GPIO[13:10] cannot be used as interrupt inputs. In any of these cases, GPIO[13:10] are defined with their alternate definition: I/O Address and Data Parity.

22.4.3

Initial States for Flash ROM on the Peripherals I/O Bus

Initial states are read in on the Peripherals I/O Bus (IO_AD[31:0]) during a hard reset for the period when the POWER_GOOD signal has been asserted and CORE_RESET_L has not been un-asserted. These initial states are listed in [Table 22-4](#) that apply to Flash ROM choices. They are determined at reset time by the XLS sensing the voltage level of the pin, which can be defined either passively (using static weak pull-up or pull-down resistors, typically 10KOhm), or using active devices such as FPGAs or PLDs.

Table 22-4. Initialization Options Sensed at Reset Time

IO_AD[31:0] Signal	Reset Time Strapping Option	Description
IO_AD20	Endian Select	Data endian format select: 0: Little endian 1: Big endian
IO_AD19	Flash Parity Enable	Parity on Flash bus: 0: Disabled 1: Enabled
IO_AD18	PCMCIA Enable / NAND Boot 5 Addr Phases	PCMCIA Enable / NAND Boot 5 Addr Phases: This signal has multiple functions: PCMCIA Enable: If IO_AD15 (NAND Boot) = 0, this signal enables the PCMCIA controller: 0 = Booting from or accessing PCMCIA devices is disabled. 1 = This signal combines with IO_AD16 to select between enabling boot from PCMCIA and enabling access of PCMCIA devices in a non-boot scenario. NAND Boot, 5 Address Phases: If IO_AD15 (NAND Boot) = 1, this signal combines with IO_AD16 to select between NAND Flash boot on CS0 with 4 or 5 Address Phases. (See "Boot Device Selection Table," Section 8.4 of XLS DataBook, for more details)
IO_AD17	Flash Data Bus Width Select	Flash Data Bus Width Select: This signal has multiple functions: NOR Flash 16-bit / 32-bit Select: If IO_AD15 (NAND Boot) = 0 and IO_AD16 = 0, this signal selects between 16-bit and 32-bit NOR Flash device bus width. 0: 16-bit data, 32-bit address 1: 32-bit data, 32-bit address NAND Flash 16-bit / 8-bit Select: If IO_AD15 (NAND Boot) = 1, this signal selects between 16-bit and 8-bit NAND Flash device bus width. 0: 16-bit data, 4 or 5 phases (i.e., 32 or 40 bits) of addr. 1: 8-bit data, 4 or 5 phases (i.e., 32 or 40 bits) of addr. Don't Care: If IO_AD15 = 0 and IO_AD16 = 1, this signal is a Don't Care. (See "Boot Device Selection Table," Section 8.4 of XLS DataBook, for more details)

Table 22-4. Initialization Options Sensed at Reset Time (continued)

IO_AD[31:0] Signal	Reset Time Strapping Option	Description
IO_AD16	Boot Select	<p>Boot Device Select: This signal selects boot device type, as follows:</p> <p>If IO_AD15 = 0:</p> <ul style="list-style-type: none"> 0: Boot from 16-bit or 32-bit NOR Flash (CS0). (Data bus width depends on IO_AD17. Whether non-boot PCMCIA accesses are allowed depends on IO_AD18.) 1: Depending on IO_AD18, boot from one of these options: <ul style="list-style-type: none"> • IO_AD18 = 1: Boot from PCMCIA Flash (CS6, 16-bit data bus width) • IO_AD18 = 0: Boot from 8-bit NOR Flash (CS0, non-muxed) <p>If IO_AD15 = 1:</p> <ul style="list-style-type: none"> 0: If IO_AD18 = 1, boot from 8-bit or 16-bit NAND Flash, CS0, 5 address phases (i.e., 5-byte address). (Data bus width depends on IO_AD17.) 1: If IO_AD18 = 0, boot from 8-bit or 16-bit NAND Flash, CS0, 4 address phases (i.e., 4-byte address). (Data bus width depends on IO_AD17.) <p>(Other decode options are Reserved.)</p> <p>(See “Boot Device Selection Table,” Section 8.4 of XLS DataBook for more details)</p>
IO_AD15	NAND Boot	<p>NAND Boot Flash Select: This signal selects as follows:</p> <ul style="list-style-type: none"> 0 - Boot from NOR Flash or PCMCIA device 1 - Boot from NAND device (CS0). (Data bus width depends on IO_AD17. Address phase count depends on IO_AD16.) <p>(See “Boot Device Selection Table,” Section 8.4 of XLS DataBook, for more details)</p>
IO_AD12	Flash Frequency	<p>Flash Frequency select:</p> <ul style="list-style-type: none"> 0: 66.67 MHz 1: 33.33 MHz <p>Note that if IO_AD12 is set to 1, then on startup while POWER_GOOD is 0, the core clock (66.67MHz) will be output on GPIO20. Once POWER_GOOD is 1 (asserted), core clock/2 will be output on GPIO20.</p>

22.4.4 XLS NAND Flash Interface

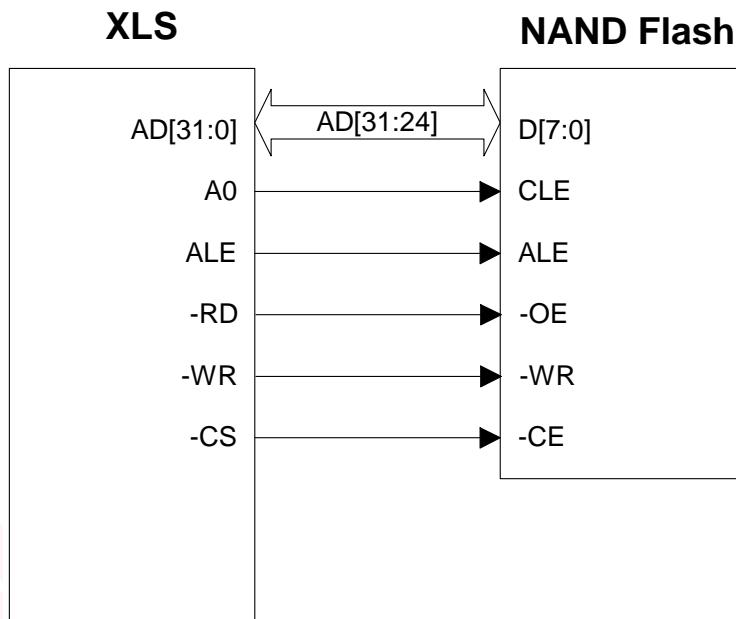
22.4.4.1 Connections

Accessing a NAND Flash device is done via several dedicated registers. Each phase of the transaction,

- Command phase
- Address phase
- Data phase

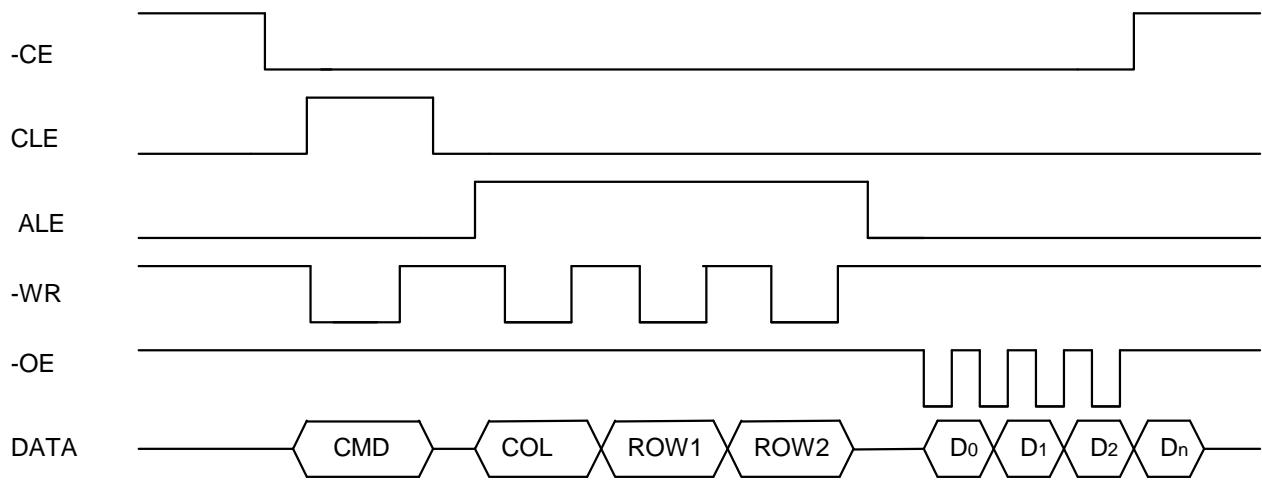
is associated with a specific register that, once written, triggers the transaction. [Figure 22-3](#) describes the interconnection between the XLS and a NAND Flash device.

Figure 22-3. NAND Flash Connection to XLS (8-bit NAND Device)



22.4.4.2 Signal Waveform

Read, write and erase cycles start with the command phase, followed by address phase, and finally the data phase of the transaction. [Figure 22-4](#) illustrates the three phases.

Figure 22-4. NAND Flash Access Diagram (Read Operation Shown)

NAND Flash Command Phase

During the Command phase ($\text{CLE} = '1'$ and $\text{ALE} = '0'$) shown in Figure 22-4, the following events occur:

1. The command byte is placed on the Data bus by the XLS
2. The XLS pulls the WE signal low, then high.

This stores the command into the command register.

NAND Flash Address Phase

During the Address phase ($\text{CLE} = '0'$ and $\text{ALE} = '1'$) of Figure 22-4, the following events occur:

1. The XLS places the first address byte on the data bus
2. The XLS toggles the WE signal

The first address byte (called the column byte in Figure 22-4), is usually set to '0' so that access will start from the beginning of the FLASH Memory's page. The address bytes that follow after column byte (indicated by Row1 and Row2 in Figure 22-4) are used to set the page within a block. (Note that in higher density NAND Flash devices, the address phase is four bytes long rather than the three bytes shown in Figure 22-4.)

NAND Flash Data Phase

During the Data phase ($\text{CLE} = '0'$ and $\text{ALE} = '0'$) of Figure 22-4, the following events occur:

1. Once BUSY goes high, data is available in the data register for read out.
2. The first data byte output is byte 0.
3. Each RE pulse reads out the next byte in the register. Data length on a 2GB NAND Flash device is a burst of 2,112 bytes.
4. After a full burst is read, the column must be programmed to select the next page.

The XLS offers dedicated registers for activating CLE and ALE. Writing the CLE register causes the data written to be placed on the data bus and the CLE signal to be asserted. Writing the ALE register causes the data written to be placed on the data bus and the ALE signal to be asserted.

Data is received or sent by accessing a single memory space programmed into the FLASH_CSBASE_ADDR[n], FLASH_CSADDR_MASK[n], and FLASH_CSDEV_PARAM[n] registers.

Access Code Example:

Assume a 512MByte device with the following characteristics:

- 512 bytes per page (row 1/2 size)
- 16K pages per block (column size)

```
init read_nand_flash_burst(int addr, int burst_size, int *data)
{
    int burst_count = 0;

    *NAND_CLE_REG = 0x01; // Set read page command
    *NAND_ALE_REG = (addr >> 14) & 0xFF; // Set COL
    *NAND_ALE_REG = (addr >> 8) & 0xFF; // Set Row1
    *NAND_ALE_REG = (addr & 0xFF); // Set Row2

    For( ;burst_count < burst_size; burst_count++)
        Data[burst_count] = *NAND_BASE_ADDR;

    Return 0;
}
```

22.4.5 PCMCIA Flash Memory Support

The Peripherals Input/Output Bus in combination with the GPIO port provides a PCMCIA (PC Card) interface. It complies with the 16-bit PC card specification (PCMCIA Release 2.1). It is a 16-bit asynchronous interface supporting both 8-bit/16-bit accesses.

The additional controller capabilities for PCMCIA are as follows:

- Allows card insertion and removal to be detected [CD1_L and CD2_L]
- Supports memory and REG accesses
- PCMCIA interface supported in any of the chip-select areas except CS0
- Can boot from NOR or linear PC Card Type 1 Flash
- Interrupt triggered to signal a card status change
- Software Reset to the PCMCIA device supported
- Generates card enable signals (CE1_L and CE2_L)

22.5 Programming Model

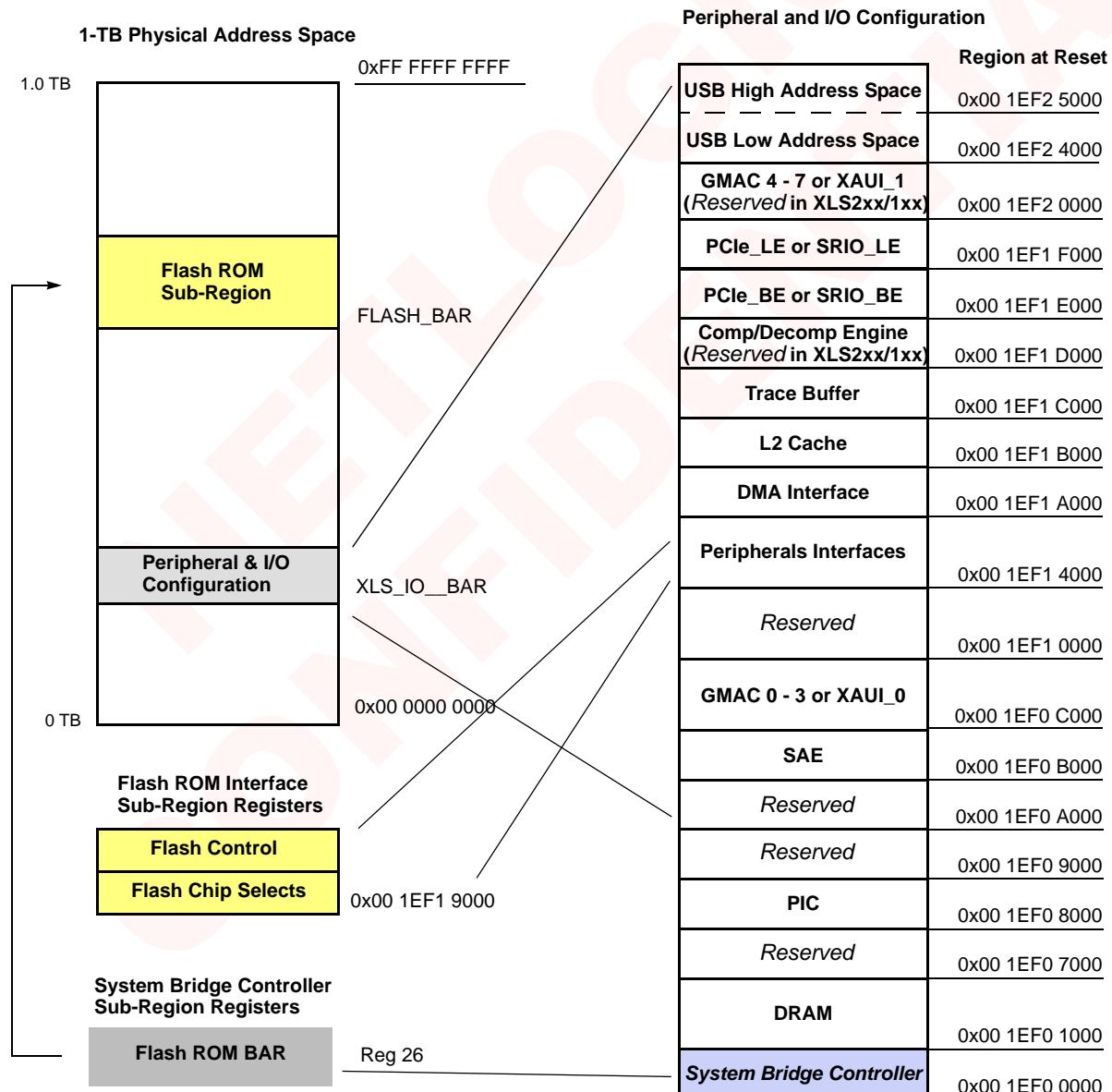
22.5.1 Register Addressing for Flash/PCMCIA

Operation of the Flash/PCMCIA memory portion of the Peripherals Interface is governed by registers in the System Bridge Controller (SBC) in the Memory and I/O Hub, and the Flash ROM sub-region address space as shown in the system memory map in [Figure 22-5](#). The offset for the Flash ROM interface register sub-region address space is given in [Table 22-5](#).

Table 22-5. Flash ROM Registers in the Peripherals & I/O Configuration Region

Peripheral	Peripheral Device Name	Sub-Region Offset	Sub-Region Offset Address at Reset
Flash ROM	XLS_IO_DEV_FLASH	0x1 9000	0x00 1EF1 9000

Figure 22-5. Memory Map of the Flash ROM Interface



22.6 Flash/PCMCIA Memory Registers

The 74 registers in the Flash Interface sub-region as listed in [Table 22-6](#). They are divided by section for chip-select area parameters and general Flash controller functions.

Register IDs are expressed in hexadecimal. Registers are 32 bits wide unless otherwise noted.

Table 22-6. Summary of Flash/PCMCIA Memory Interface Registers

Register ID	Register Name	Description	R/W	Reset Value
Section 22.7.1, “Chip-Select Area Registers”				
FLASH_CSBASE_ADDR[n]				
0x00	FLASH_CSBASE_ADDR0	CS0 Base Address register	R/W	0x0000 00C0
0x01	FLASH_CSBASE_ADDR1	CS1 Base Address register	R/W	0x0000 0000
0x02	FLASH_CSBASE_ADDR2	CS2 Base Address register	R/W	0x0000 0000
0x03	FLASH_CSBASE_ADDR3	CS3 Base Address register	R/W	0x0000 0000
0x04	FLASH_CSBASE_ADDR4	CS4 Base Address register	R/W	0x0000 0000
0x05	FLASH_CSBASE_ADDR5	CS5 Base Address register	R/W	0x0000 0000
0x06	FLASH_CSBASE_ADDR6	CS6 Base Address register	R/W	0x0000 0000
0x07	FLASH_CSBASE_ADDR7	CS7 Base Address register	R/W	0x0000 0000
0x08	FLASH_CSBASE_ADDR8	CS8 Base Address register	R/W	0x0000 0000
0x09	FLASH_CSBASE_ADDR9	CS9 Base Address register	R/W	0x0000 0000
FLASH_CSADDR_MASK[n]				
0x10	FLASH_CSADDR_MASK0	CS0 Address Mask register	R/W	0x0000 003F
0x11	FLASH_CSADDR_MASK1	CS1 Address Mask register	R/W	0x0000 0000
0x12	FLASH_CSADDR_MASK2	CS2 Address Mask register	R/W	0x0000 0000
0x13	FLASH_CSADDR_MASK3	CS3 Address Mask register	R/W	0x0000 0000
0x14	FLASH_CSADDR_MASK4	CS4 Address Mask register	R/W	0x0000 0000
0x15	FLASH_CSADDR_MASK5	CS5 Address Mask register	R/W	0x0000 0000
0x16	FLASH_CSADDR_MASK6	CS6 Address Mask register	R/W	0x0000 0000
0x17	FLASH_CSADDR_MASK7	CS7 Address Mask register	R/W	0x0000 0000
0x18	FLASH_CSADDR_MASK8	CS8 Address Mask register	R/W	0x0000 0000
0x19	FLASH_CSADDR_MASK9	CS9 Address Mask register	R/W	0x0000 0000
FLASH_CSDEV_PARM[n]				
0x20	FLASH_CSDEV_PARM0	CS0 Device Parameter register	R/W	0x0000 00E0
0x21	FLASH_CSDEV_PARM1	CS1 Device Parameter register	R/W	0x0000 00E0
0x22	FLASH_CSDEV_PARM2	CS2 Device Parameter register	R/W	0x0000 00E0
0x23	FLASH_CSDEV_PARM3	CS3 Device Parameter register	R/W	0x0000 00E0
0x24	FLASH_CSDEV_PARM4	CS4 Device Parameter register	R/W	0x0000 00E0
0x25	FLASH_CSDEV_PARM5	CS5 Device Parameter register	R/W	0x0000 00E0
0x26	FLASH_CSDEV_PARM6	CS6 Device Parameter register	R/W	0x0000 00E0
0x27	FLASH_CSDEV_PARM7	CS7 Device Parameter register	R/W	0x0000 00E0
0x28	FLASH_CSDEV_PARM8	CS8 Device Parameter register	R/W	0x0000 00E0
0x29	FLASH_CSDEV_PARM9	CS9 Device Parameter register	R/W	0x0000 00E0

Table 22-6. Summary of Flash/PCMCIA Memory Interface Registers (continued)

Register ID	Register Name	Description	R/W	Reset Value
FLASH_CSTIME_PARMA[n]				
0x30	FLASH_CSTIME_PARMA0	CS0 Timing Parameters A register	R/W	0x4F40 0E22
0x31	FLASH_CSTIME_PARMA1	CS1 Timing Parameters A register	R/W	0x4F40 0E22
0x32	FLASH_CSTIME_PARMA2	CS2 Timing Parameters A register	R/W	0x4F40 0E22
0x33	FLASH_CSTIME_PARMA3	CS3 Timing Parameters A register	R/W	0x4F40 0E22
0x34	FLASH_CSTIME_PARMA4	CS4 Timing Parameters A register	R/W	0x4F40 0E22
0x35	FLASH_CSTIME_PARMA5	CS5 Timing Parameters A register	R/W	0x4F40 0E22
0x36	FLASH_CSTIME_PARMA6	CS6 Timing Parameters A register	R/W	0x4F40 0E22
0x37	FLASH_CSTIME_PARMA7	CS7 Timing Parameters A register	R/W	0x4F40 0E22
0x38	FLASH_CSTIME_PARMA8	CS8 Timing Parameters A register	R/W	0x4F40 0E22
0x39	FLASH_CSTIME_PARMA9	CS9 Timing Parameters A register	R/W	0x4F40 0E22
FLASH_CSTIME_PARMB[n]				
0x40	FLASH_CSTIME_PARMB0	CS0 Timing Parameters B register	R/W	0x0000 83CF
0x41	FLASH_CSTIME_PARMB1	CS1 Timing Parameters B register	R/W	0x0000 83CF
0x42	FLASH_CSTIME_PARMB2	CS2 Timing Parameters B register	R/W	0x0000 83CF
0x43	FLASH_CSTIME_PARMB3	CS3 Timing Parameters B register	R/W	0x0000 83CF
0x44	FLASH_CSTIME_PARMB4	CS4 Timing Parameters B register	R/W	0x0000 83CF
0x45	FLASH_CSTIME_PARMB5	CS5 Timing Parameters B register	R/W	0x0000 83CF
0x46	FLASH_CSTIME_PARMB6	CS6 Timing Parameters B register	R/W	0x0000 83CF
0x47	FLASH_CSTIME_PARMB7	CS7 Timing Parameters B register	R/W	0x0000 83CF
0x48	FLASH_CSTIME_PARMB8	CS8 Timing Parameters B register	R/W	0x0000 83CF
0x49	FLASH_CSTIME_PARMB9	CS9 Timing Parameters B register	R/W	0x0000 83CF
Section 22.7.2, "Flash Memory Controller Registers"				
0x50	FLASH_INT_MASK	Flash Interrupt Mask register	R/W	0x0000 0004
0x60	FLASH_INT_STATUS	Flash Interrupt Status register	R/W	0x0000 0000
0x70	FLASH_ERROR_STATUS	Flash Error Status register	RO	0x0000 0000
0x80	FLASH_ERROR_ADDR	Flash Error Address register	RO	0x0000 0000
Section 22.7.3, "NAND Flash Memory Control Registers"				
CLE[n]				
0x90	CLE0	CS0 8-bit CLE command value register	R/W	0x0000 0000
0x91	CLE1	CS1 8-bit CLE command value register	R/W	0x0000 0000
0x92	CLE2	CS2 8-bit CLE command value register	R/W	0x0000 0000
0x93	CLE3	CS3 8-bit CLE command value register	R/W	0x0000 0000
0x94	CLE4	CS4 8-bit CLE command value register	R/W	0x0000 0000
0x95	CLE5	CS5 8-bit CLE command value register	R/W	0x0000 0000
0x96	CLE6	CS6 8-bit CLE command value register	R/W	0x0000 0000
0x97	CLE7	CS7 8-bit CLE command value register	R/W	0x0000 0000
0x98	CLE8	CS8 8-bit CLE command value register	R/W	0x0000 0000
0x99	CLE9	CS9 8-bit CLE command value register	R/W	0x0000 0000
ALE[n]				
0xA0	ALE0	CS0 8-bit ALE command value register	R/W	0x0000 0000
0xA1	ALE1	CS1 8-bit ALE command value register	R/W	0x0000 0000
0xA2	ALE2	CS2 8-bit ALE command value register	R/W	0x0000 0000
0xA3	ALE3	CS3 8-bit ALE command value register	R/W	0x0000 0000
0xA4	ALE4	CS4 8-bit ALE address phase value register	R/W	0x0000 0000

Table 22-6. Summary of Flash/PCMCIA Memory Interface Registers (continued)

Register ID	Register Name	Description	R/W	Reset Value
0xA5	ALE5	CS5 8-bit ALE address phase value register	R/W	0x0000 0000
0xA6	ALE6	CS6 8-bit ALE address phase value register	R/W	0x0000 0000
0xA7	ALE7	CS7 8-bit ALE address phase value register	R/W	0x0000 0000
0xA8	ALE8	CS8 8-bit ALE address phase value register	R/W	0x0000 0000
0xA9	ALE9	CS9 8-bit ALE address phase value register	R/W	0x0000 0000

NETLOGIC
CONFIDENTIAL

22.7 Flash/PCMCIA Register Descriptions

There are fifty-four registers in the Flash Interface sub-region that functionally divide into two sections for the Chip-Select Areas and for the Flash Memory Controllers.

22.7.1 Chip-Select Area Registers

In each of the ten chip-select Flash memory areas, the area's base address, size, device parameters, and timing parameters can all be defined independently, using the following ten sets of five registers. (In addition, ALE and CLE can be defined independently, using the ten sets of two registers described in [Section 22.7.3](#)).

22.7.1.1 FLASH_CSBASE_ADDR[n]

Register ID: 0x00 + *n*
Address Offset: 0x000 + (*n* x 4)

These Read/Write registers determine the Chip-Select Base addresses for the ten Flash ROM areas in the Peripherals I/O Bus address space. The Area number *n* varies from 0 to 9. The maximum memory size supported is 4 GB and 64 KB is the minimum. The value after Reset is 0x0000 00C0 or 0x0000 0000

31:16		15:0		
Reserved		BASE_ADDR		
Bits Field Name Field Description R/W Reset				
31:16	Reserved	Reserved	RO	0x0000
15:0	BASE_ADDR	Chip-select area Base Address on the Peripherals I/O Bus	R/W	0x00C0, <i>n</i> = 0 0x0000, <i>n</i> = 1 - 9

22.7.1.2 FLASH_CSADDR_MASK[n]Register ID: 0x10 + *n*Address Offset: 0x40 + (*n* x 4)

These Read/Write registers determine the Chip-Select Base Address Masks for the ten Flash ROM areas in the Peripherals I/O Bus address space. The Area number *n* varies from 0 to 9. The maximum memory size supported is 4 GB and 64 KB is the minimum. The value after Reset is 0x0000 003F or 0x0000 0000.

31:16	15:0
<i>Reserved</i>	BASE_AMASK

Bits	Field Name	Field Description	R/W	Reset
31:16	<i>Reserved</i>	<i>Reserved</i>	RO	0x0000
15:0	BASE_AMASK	<p>Chip-select area Address Mask determines area size on the Peripherals I/O Bus:</p> <ul style="list-style-type: none"> 0x0000: 64 KB 0x0001: 128 KB 0x0003: 256 KB 0x0007: 512 KB ----- 0xFFFF: 4 GB <p>BASE_AMASK[15:0] masks the physical address[31:16]. Physical address[15:0] is always masked, which makes the minimum memory space 64 KB.</p> <p>Following are the values of the mask register and the effect on the memory size of that chip select region:</p> <ul style="list-style-type: none"> • 0x0000 masks physical address [15:0] so memory size would be 64 KB • 0x0001 masks Physical address [16:0] so memory size would be 128 KB • 0x0003 masks Physical address [17:0] so memory size would be 256 KB • and so on . . . 	R/W	0x003F, <i>n</i> = 0 0x0000, <i>n</i> = 1 - 9

22.7.1.3 FLASH_CSDEV_PARM[n]

Register ID: 0x20 + *n*
Address Offset: 0x80 + (*n* x 4)

These Read/Write registers determine the Chip-Select Device Parameters for the ten Flash ROM areas in the Peripherals I/O Bus address space. The Area number *n* varies from 0 to 9. The value after Reset is 0x0000 00E0.

FLASH_CSDEV_PARM[n]														
31:16														
REESERVED														
15	14	13	12	11	10	9	8:7	6	5	4	3:1	0		
BOOT FROM NAND	NAND TYPE	ADV_TYPE	PARITY_TYPE	PARITY_EN	GENIF_EN	PCMCIA_EN	DWIDTH	MX_ADDR	WAIT_POL	WAIT_EN	BURST	BURST_EN		
Bits	Field Name		Field Description									R/W	Reset	
31:16	Reserved		(Reserved)									RO	b0000 0000 0000 0000	
15	BOOTFROMNAND		Boot From NAND: Boot from NAND Flash on Chip Select 0. 0: Boot from NOR Flash or linear PC Card Type 1 Flash. 1: Boot from NAND Flash. Note: After boot from NAND Flash is done, SW should clear this bit.									R/W	b0	
14	NAND_EN		NAND Flash Enable: Enables NAND Flash for this region 0: NOR Flash 1: NAND Flash									R/W	b0	
13	ADV_TYPE		Address Valid Sensing Type: The Address Valid sensing method is selected with this bit: 0: Level sensing 1: Pulse sensing (used in Burst Mode for AMD memories)									R/W	b0	
12	PARITY_TYPE		Parity Type: This bit selects between even and odd parity, for use when the Flash Memory pins are connected to a PLD or FPGA which runs the Flash memory protocol: 0: Even Parity 1: Odd Parity									R/W	b0	
11	PARITY_EN		Parity Enable: This bit enables parity checking on the Flash interface, for use when the Flash Memory pins are connected to a PLD or FPGA which runs the Flash memory protocol: 0: Disable Parity checking 1: Enable Parity checking									R/W	b0	
10	GENIF_EN		Generic PLD/FPGA Interface Mode Enable: This bit allows connecting the Flash Memory interface to a PLD or FPGA which runs the Flash memory protocol: 0: Disable generic PLD/FPGA mode 1: Enable generic PLD/FPGA mode Note that if this bit is set to '1', then GPIO[13:10] cannot be used for interrupts. See Section 22.4.2.1 for a discussion of this topic.									R/W	b0	
9	PCMCIA_EN		PCMCIA Interface Mode Enable: This bit allows connecting the Flash Memory interface to a PCMCIA device: 0: Disable PCMCIA mode 1: Enable PCMCIA mode									R/W	b0	

Bits	Field Name	Field Description	R/W	Reset
8:7	DWIDTH	Data Bus Width: The Data Bus width of the connected device: 00 or 11: 8-bit device 01: 16-bit device 10: 32-bit device	R/W	b0 1
6	MX_ADDR	Muxed Address: Enable Multiplexed mode of operation: 0: Address/Data unmultiplexed. AD[31:24] = Data, AD[23:0] = Addr 1: Address/Data are multiplexed. External latch required.	R/W	b1
5	WAIT_POL	Wait Polarity: External WAIT Polarity: 0: Active high polarity 1: Active low polarity	R/W	b1
4	WAIT_EN	Wait Enable: Enables device Wait acknowledge mode: 0: Disable Wait mode. External IO_WAIT_L signal is not used. 1: Enable device Wait mode. External IO_WAIT_L signal is used.	R/W	b0
3:1	BURST	Burst Length: Specifies the Burst mode of operation: 000: Two transfers 001: Four transfers 010: Eight transfers 011: Sixteen transfers 100: Thirty-two transfers Note: The XLS has a limitation of 32 bytes for any burst transfer: • For a 32-bit (4 Byte) wide interface, the maximum burst size is 8 transfers (8 transfers x 4 bytes/transfer = 32Bytes). • For a 16-bit wide interface, the maximum burst size is 16 transfers (16 transfers x 2 bytes/transfer = 32Bytes). • For an 8-bit wide interface, the maximum burst size is 32 transfers (32 transfers x 1 byte/transfer = 32Bytes). Any of the interfaces can also use 2 or 4 transfers.	R/W	b000
0	BURST_EN	Burst Enable: Enables Burst mode operation. Burst mode is used only with synchronous Flash devices. 0: Burst mode disabled 1: Burst mode enabled	R/W	b0

22.7.1.4 FLASH_CSTIME_PARMA[n]

Register ID: 0x30 + n

Address Offset: 0x0C0 + (n x 4)

These Read/Write registers determine the Chip-Select A Timing Parameters for the ten Flash ROM areas in the Peripherals I/O Bus address space. The Area number n varies from 0 to 9. All the timing values are in terms of Flash clock cycles (33.33/66.67 MHz). The register value after Reset is 0x4F40 0E22.

31:28	27:24	23:22	21:19	18:16	15:11	10:6	5:3	2:0
CS_TO_CS	WE_TO_CS	OE_TO_CS	CS_TO_WE	CS_TO_OE	WAIT_T0_DATA	CS_WIDTH	ALE_TO_CS	ALE_WIDTH

Bits	Field Name	Field Description	R/W	Reset
31:28	CS_TO_CS	Idle timing cycles between back-to-back operations	R/W	0x4
27:24	WE_TO_CS	Write Enable un-assertion to Chip Select un-assertion	R/W	0xF
23:22	OE_TO_CS	Output Enable un-assertion to Chip Select un-assertion	R/W	b01
21:19	CS_TO_WE	Write Enable assertion after asserting Chip Select signal	R/W	b00 0

Bits	Field Name	Field Description	R/W	Reset
18:16	CS_TO_OE	Output Enable assertion after asserting Chip Select signal	R/W	b000
15:11	WAIT_TO_DATA	WAIT TO DATA: This parameter has different definitions depending on the mode: <u>Synchronous Burst Mode</u> -- In this mode WAIT_TO_DATA is the delay (in clock cycles) after assertion of IO_WE_L or IO_OE_L until data is written or sampled. <u>Single Asynchronous Wait Mode</u> -- In this mode WAIT_TO_DATA is the delay (in clock cycles) after the assertion of IO_WAIT until data is written or sampled. <u>Single Asynchronous Mode</u> -- WAIT_TO_DATA is not used in this mode. <u>Synchronous Burst Wait Mode</u> -- WAIT_TO_DATA is not used in this mode.	R/W	b0000 1
10:6	CS_WIDTH	Width of Chip Select pulse. Used only in device Wait mode. IO_WAIT_L will be checked after CS_WIDTH cycles from IO_CS_L assertion.	R/W	b110 00
5:3	ALE_TO_CS	Chip-Select assertion time after un-asserting IO_ALE Note: In nand flash this is used as WE_TO_CLE and WE_TO_ALE	R/W	b10 0
2:0	ALE_WIDTH	Width of the external IO_ALE signal	R/W	b010

22.7.1.5 FLASH_CSTIME_PARMB[n]

Register ID: **0x40 + n**

Address Offset: **0x100 + (n x 4)**

These Read/Write registers determine the Chip-Select B Timing Parameters for the ten Flash ROM areas in the Peripherals I/O Bus address space. The Area Number *n* varies from 0 to 9. All the timing values are in terms of Flash clock cycles (33.33/66.67 MHz). The register value after Reset is 0x0000 83CF.

31:27	26:12	11:6	5:0
Reserved	WAIT_TIMEOUT	WE_WIDTH	OE_WIDTH

Bits	Field Name	Field Description	R/W	Reset
31:27	Reserved	Reserved	RO	b0000 0
26:12	WAIT_TIMEOUT	Timeout value for IO_WAIT_L signal. Use only in device Wait mode.	R/W	b000 0000 0000 1000
11:6	WE_WIDTH	Width of the Write Enable pulse	R/W	b0011 11
5:0	OE_WIDTH	Width of the Output Enable pulse. Not used in burst mode or device Wait mode	R/W	b00 1111

22.7.2 Flash Memory Controller Registers

There are four additional registers for ROM controller functions other than the properties of the ten chip select memory areas. They involve configurations, interrupts and the related error conditions.

22.7.2.1 FLASH_INT_MASK

Register ID: 0x50

Address Offset: 0x140

This Read/Write register masks interrupts from the Flash/PCMCIA memory interface. The value after system reset is 0x0000 0004.

31:8	7	6	5	4	3	2	1	0
Reserved	RYBYMASK	RDYMASK	WPMAS	CDMASK	RSV	RESET	REG_ACCESS	PCMCIA_EN
Field Description								
Bits	Field Name	R/W	Reset					
31:8	Reserved	RO	0x0000 00					
7	RYBYMASK	R/W	b0					
6	RDYMASK	R/W	b0					
5	WPMAS	R/W	b0					
4	CDMASK	R/W	b0					
3	Reserved	RO	b0					
2	RESET	R/W	b1					
1	REG_ACCESS	R/W	b0					
0	PCMCIA_EN	R/W	b0					

22.7.2.2 FLASH_INT_STATUS

Register ID: 0x60

Address Offset: 0x180

This Read/Write register indicates Interrupt Status for Flash ROMs. Bits [9:11] are only valid when FLASH_INT_MASK.PCMCIA_EN is set. The value after reset is 0x0000 0000.

31:19												18		17		16	
Reserved												ILL_PC_INT		WERR_INT		RYBY_INT	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1:0			
RSV	WAIT_TO_INT	MUTL_CS_INT	ILL_ADDR_INT	RDY_INT	WP_INT	CD_INT	RYBY_STS	RDY_STS	WP_STS	CD2_STS	CD1_STS	BVD2_STS	BVD1_STS	RSV			

Bits	Field Name	Field Description	R/W	Reset
31:19	Reserved	Reserved. Reset value = b0000 0000 0000 0	RO	See left
18	ILL_PC_INT	Set and Interrupt when Illegal access to disabled PCMCIA address space. Cleared by writing a '1' to this bit field.	R/W	b0
17	WERR_INT	Set and Interrupt when Write access Error in the Flash controller. Cleared by writing a '1' to this bit field.	R/W	b0
16	RYBY_INT	Set and Interrupt when positive transition in RYBY_L signal. Implies programming/erasure of the Flash is completed. Cleared by writing a '1' to this bit field.	R/W	b0
15	Reserved	Reserved	RO	b0
14	WAIT_TO_INT	Set and Interrupt when there is a Wait signal Time-Out. Valid only in device acknowledge mode. Address accessed is logged in the Error Address Register. Cleared by writing a '1' to this bit field.	R/W	b0
13	MUTL_CS_INT	Set and Interrupt when multiple Chip Selects are asserted for a memory access. The address accessed is logged in the Error Address Register. Cleared by writing a '1' to this bit field.	R/W	b0
12	ILL_ADDR_INT	Set and Interrupt when there are no chip selects asserted for a memory access. The Illegal Address accessed is logged in the Error Address Register. Cleared by writing a '1' to this bit field.	R/W	b0
11	RDY_INT	Set and interrupt when there is a change in the READY signal from the enabled PCMCIA card. Cleared by writing a '1' to this bit field.	R/W	b0
10	WP_INT	Set and interrupt when there is a change in the WP signal from the enabled PCMCIA card. Cleared by writing a '1' to this bit field.	R/W	b0
9	CD_INT	Set and interrupt when there is a change in either CD1 or CD2 signals from the enabled PCMCIA card. Cleared by writing a '1' to this bit field.	R/W	b0
8	RYBY_STS	Status of the RYBY_L signal from the Flash Memory	RO	b0
7	RDY_STS	Status of the READY signal from the PCMCIA card	RO	b0
6	WP_STS	Status of the WP signal from the PCMCIA card	RO	b0
5	CD2_STS	Status of the CD2 signal from the PCMCIA card	RO	b0
4	CD1_STS	Status of the CD1 signal from the PCMCIA card	RO	b0
3	BVD2_STS	Status of the BVD2 signal from the PCMCIA card	RO	b0
2	BVD1_STS	Status of the BVD1 signal from the PCMCIA card	RO	b0
1:0	Reserved	Reserved	RO	b00

22.7.2.3 FLASH_ERROR_STATUS

Register ID: **0x70**
Address Offset: **0x1C0**

This Read Only register indicates interrupt Error Status for the Flash ROM controllers. The value after Reset is 0x0000 0000.

31:10	9:0
<i>Reserved</i>	CS_ERR_INT

Bits	Field Name	Field Description	R/W	Reset
31:10	<i>Reserved</i>	<i>Reserved</i>	RO	b0000 0000 0000 0000 0000 00
9:0	CS_ERR_INT	Indicates which of the ten Chip Select areas that have an Error condition causing an interrupt	RO	b00 0000 0000

22.7.2.4 FLASH_ERROR_ADDR

Register ID: **0x80**
Address Offset: **0x200**

This Read Only register indicates the interrupt Error Address for the Flash ROM controllers. The value after Reset is 0x0000 0000.

31:0	ERROR_ADDR

Bits	Field Name	Field Description	R/W	Reset
31:0	ERROR_ADDR	This contains the Error Address causing the controller interrupt	RO	0x0000 0000

22.7.3 NAND Flash Memory Control Registers

The following registers support NAND Flash operation.

22.7.3.1 CLE[n]

Register ID: 0x90 + n
Address Offset: 0x240 + ($n \times 4$)

7:0	COMMAND			
Bits	Field Name	Field Description	R/W	Reset
7:0	COMMAND	Content of this register is sent on IO_AD[31:24] during the command phase of the NAND transfer. See Figure 22-6 .	R/W	b0000 0000

22.7.3.2 ALE[n]

Register ID: 0xA0 + n
Address Offset: 0x280 + ($n \times 4$)

7:0	ADDR			
Bits	Field Name	Field Description	R/W	Reset
7:0	ADDR	Content of this register is sent on IO_AD[31:24] during the address phase of the NAND transfer.	R/W	b0000 0000

22.7.4 Notes

IO_AD[0] is used as CLE (command latch enable) during the command phase of NAND Flash transactions.

22.7.5 AC Timing for Flash Enhancements

See CLE register description and [Figure 22-6](#) and [Figure 22-7](#).

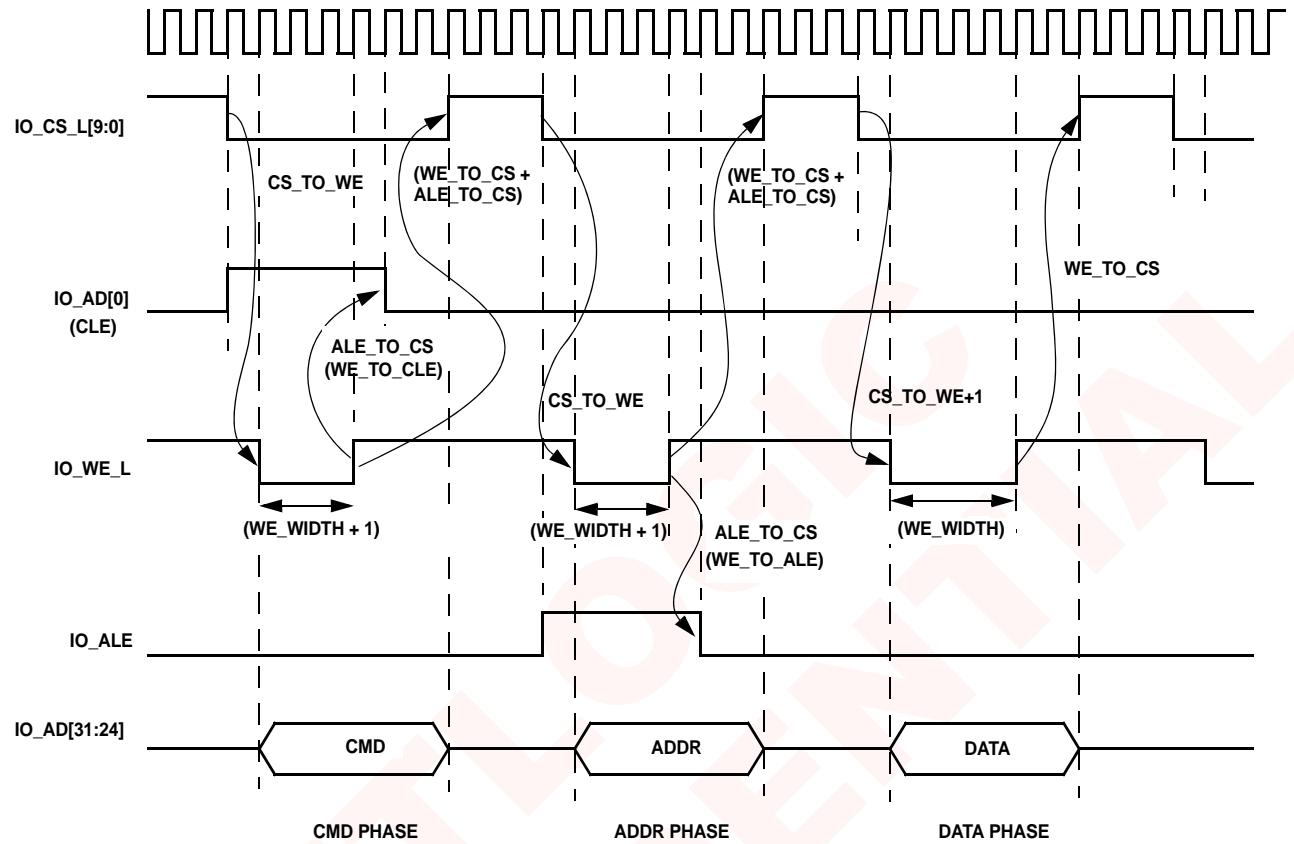
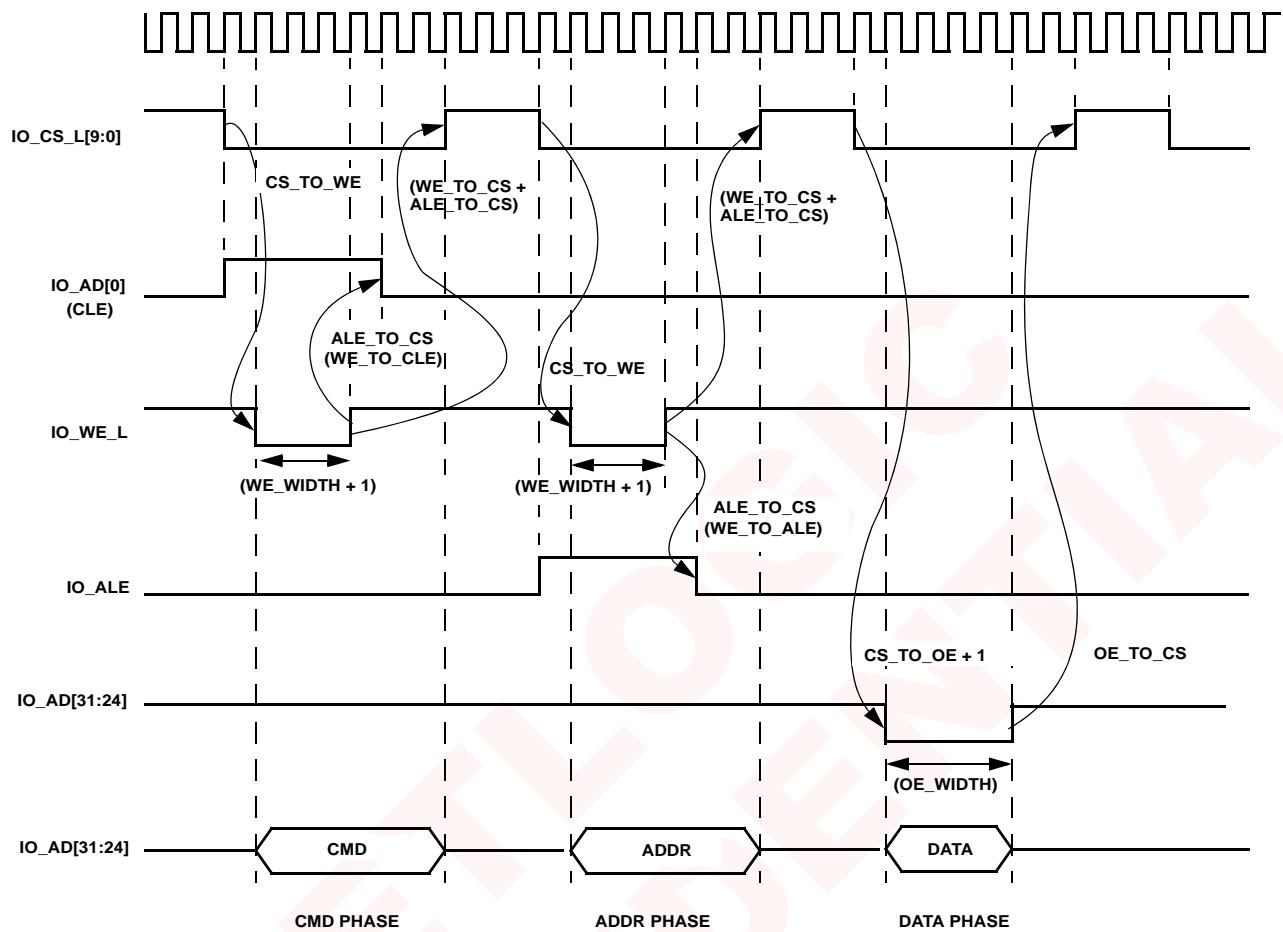
Figure 22-6. Write Timing for COMMAND and Address Transfer

Figure 22-7. Read Timing for COMMAND and Address Transfer



Chapter 23 GPIO and the Peripherals Interface

23.1 Introduction

This chapter covers the GPIO section of the Peripherals Interface. For a main diagram of the Peripherals Interface in the XLS architecture, see previous two chapters. The GPIO signal group contains the GPIO signal pins and some general system functions such as the various PLLs, clocks and testing controls.

23.1.1 GPIO Port Signals

The twenty-five signals of the General-Purpose Input/Output port are programmable and all but seven can serve multiple functions, including generating interrupts to the XLS cores. Fourteen of the signals are used by the internal PCMCIA controller when it is used with the Peripherals I/O Bus to support PCMCIA CompactFlash. Six of the signals are used to support Boot Flash Memories on the Peripherals I/O Bus.

Most GPIO signals can be configured as inputs, outputs or interrupt inputs (edge or level sensitive). Each of those GPIOs has its own direction control, interrupt enable and input and output data bit registers. All alternate signal names and their GPIO connections are given in [Table 23-1](#).

Table 23-1. GPIO Port and Flash Memory Signal Descriptions

Signal Name		Signals	GPIO / Alternate I/O	GPIO / Alternate Description
GPIO	Alternate			
GPIO0	CF_REG_L	1	I/O / O	GPIO (interrupt input) / CompactFlash Register selects attribute memory
GPIO1	CF_CE1_L	1	I/O / O	GPIO (interrupt input) / CompactFlash Card Enable 1
GPIO2	CF_CE2_L	1	I/O / O	GPIO (interrupt input) / CompactFlash Card Enable 2
GPIO3	CF_CD1_L	1	I/O / I	GPIO (interrupt input) / CompactFlash Card Detect 1
GPIO4	CF_CD2_L	1	I/O / I	GPIO (interrupt input) / CompactFlash Card Detect 2
GPIO5	CF_READY	1	I/O / I	GPIO (interrupt input) / CompactFlash Ready
GPIO6	CF_WP	1	I/O / I	GPIO (interrupt input) / CompactFlash Write Protect
GPIO7	CF_BVD1	1	I/O / I	GPIO (interrupt input) / CompactFlash Battery Voltage Detect 1
GPIO8	CF_BVD2	1	I/O / I	GPIO (interrupt input) / CompactFlash Battery Voltage Detect 2
GPIO9	CF_RESET	1	I/O / O	GPIO (interrupt input) / CompactFlash reset
GPIO[13:10]	IO_AD[3:0]	4	I/O / I/O	GPIO (interrupt input) / I/O bus Address and Data Parity (see also Section 22.4.2.1)
GPIO14	FLASH_CPUID0	1	O	Flash memory CPU ID 0 always. Never programmable.
GPIO15	FLASH_CPUID1	1	O	Flash memory CPU ID 1 always. Never programmable.
GPIO16	FLASH_CPUID2	1	O	Flash memory CPU ID 2 always. Never programmable.
GPIO17	FLASH_RDY	1	I	Flash memory Ready always. Never programmable.

Table 23-1. GPIO Port and Flash Memory Signal Descriptions (continued)

Signal Name		Signals	GPIO / Alternate I/O	GPIO / Alternate Description
GPIO	Alternate			
GPIO18	FLASH_ADV	1	O	Flash memory Address Valid always. Never programmable.
GPIO19	FLASH_RESET_N	1	O	Flash memory reset always. Never programmable.
GPIO20	-	1	I/O	General Purpose I/O (interrupt input) Note: if IO_AD[12] is set to 1 (Flash33MHz mode), then until POWER_GOOD is 0, the CORE_CLOCK (66.66 MHz) will be output on GPIO20. Once POWER_GOOD goes high, CORE_CLOCK/2 will be output on GPIO20. If IO_AD[12] is 0 while POWER_GOOD is low, this will not occur.
GPIO21	-	1	I/O	General Purpose I/O (interrupt input)
GPIO22	-	1	I/O	General Purpose I/O (interrupt input)
GPIO23	THERMAL_INTRPT	1	I	Thermal Interrupt always. Never programmable for other functions
GPIO24	-	1	I/O	General Purpose I/O (interrupt input)

23.1.2 General System Functions Supported by GPIO Pins

In addition to the twenty-five GPIO pins, the GPIO Peripherals Interface also contain registers for the following general system functions:

- DRAM Channel Pair AB Clock rate and PLL control
- DRAM Channel Pair CD Clock rate and PLL control
- System Control:
 - Reset configuration, thermal control and peripheral configuration management
 - Interrupts and clock and delay settings
 - Cache operation
- System Testing:
 - BIST
 - Random number generator

23.1.3 Obtaining the Processor ID Value using the GPIO Register

The ProcessorID value can be retrieved by reading the GPIO_FUSEBANK register located at offset address 0x1808C. An 8-bit field is used to store the ProcessorID. The location of this field within the 32-bit GPIO register differs depending on the product family and the silicon revision.

Note that the RevisionID column in the table below is shown only for completeness and cannot be read from this register. But it can be obtained via execution of the JTAG IDCODE instruction, or by reading the COP0 15 register.

The encoding of the ProcessorID values for each of the XLS products and their relative location within the GPIO_FUSEBANK register is shown in [Table 23-2](#).

Table 23-2. ProcessorID Value Locations in GPIO Register

Product	Silicon Revision ^a	Field Location (8-bits)	Encoding (8-bits)
XLS616	B0	23:16	0x40
XLS608			0x4A
XLS416			0x44
XLS408			0x4E
XLS404			0x4F
XLS408-Lite	A1	21:14	0x88
XLS404-Lite			0x8C
XLS208			0x8E
XLS204			0x8F
XLS108			0xCE
XLS104			0xCF

a. The silicon revision information is not available by reading the GPIO register.

The RevisionID can be obtained by reading bits 7:0 of COP0, register 15; or by executing the JTAG IDCODE instruction and reading bits 31:28.

In the above table, the encoding for the XLS208 and XLS204 parts are shown as 0x8E and 0x8F respectively. These numbers differ from what was initially documented in the COP0 ProcessorID register in Chapter 4 of the XLS Programmer's Manual. The manual contained values of 0x83 and 0x81 respectively.

The 0x8E and 0x8F values were originally assigned to the Condor versions of the XLS208 and XLS204. These products were never released. Instead, the released products were the Raven versions, originally called the XLS208R and the XLS204R. The Raven version had processor ID value of 0x83 and 0x81. But because the 0x8E and 0x8F numbering scheme was already in place, the Raven versions of the XLS208 and XLS204 were assigned these values, and the 'R' designation was dropped. So the Raven versions currently have the encodings originally assigned to the Condor parts.

Subsequent versions of the XLS Programmer's Manual contain the updated processor ID values for these parts of 0x8E and 0x8F.

23.2 Programming Model

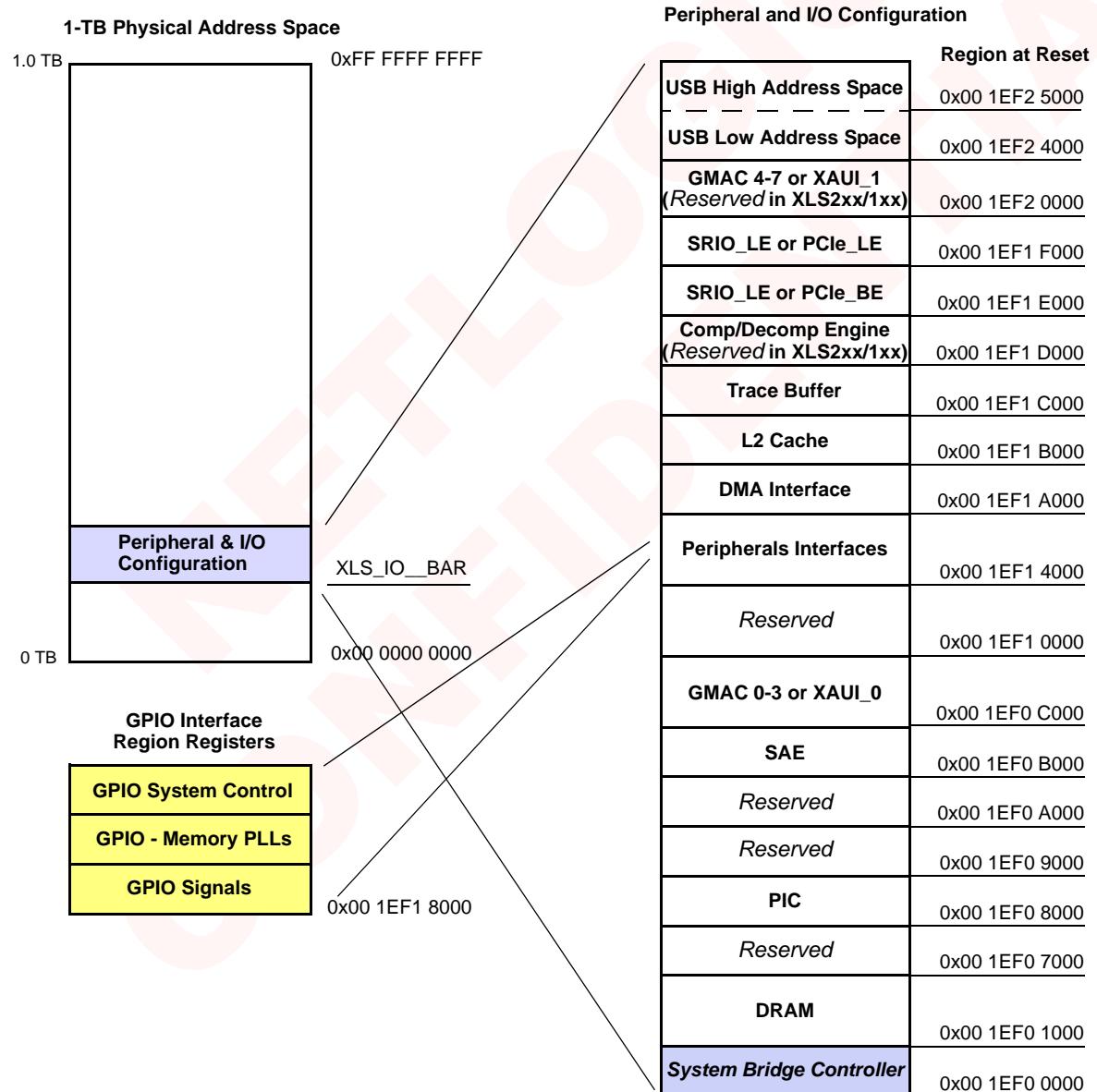
23.2.1 Register Addressing for GPIO Interface

Operation of the GPIO portion of the Peripherals Interface is governed by registers in the GPIO address space as shown in the system memory map in [Figure 23-2](#). The offset for the GPIO interface registers address space is given in [Table 23-1](#).

Figure 23-1. GPIO Sub-Region in the Peripherals & I/O Configuration Region

Peripheral	Peripheral Device Name	Region Offset	Region at Reset
General Purpose I/O (GPIO)	XLS_IO_DEV_GPIO	0x1 8000	0x00 1EF1 8000

Figure 23-2. Memory Map of the GPIO Interface



23.2.2 Setup for Operation

As is the case for the other interfaces in the XLS, the GPIO may need setup related to the Programmable Interrupt Controller such as entries in the Interrupt Redirection Table, or IRT. See [Chapter 10, “Programmable Interrupt Controller”](#) for details.

23.3 GPIO Registers

There are fifty-five (55) registers in the GPIO Interface sub-region as listed in [Table 23-3](#). They are divided by section for the GPIO signal pins, the memory PLLs and general system control registers.

Register IDs are given in hexadecimal. All register address offsets are on 4-byte boundaries and are computed as 4 times the Register ID value.

Table 23-3. Summary of GPIO Interface Registers

Register ID	Register Address Offset (hex)	Name and Description	RW	Reset Value
Section 23.3.1, “General Purpose Input/Output Signal Registers”				
0x00	0x00	GPIO Interrupt Enable Register	R/W	0x0
0x01	0x04	GPIO Input Inversion Register	R/W	0x0
0x02	0x08	GPIO I/O Direction Register	R/W	0x0
0x03	0x0C	GPIO I/O Data Write Register	R/W	0x0
0x04	0x10	GPIO I/O Read Register	RO	
0x05	0x14	GPIO Interrupt Clear Register	R/W	0x0
0x06	0x18	GPIO Interrupt Status Register	RO	0x0
0x07	0x1C	GPIO Interrupt Type Register	R/W	0x0
0x08	0x20	GPIO Software Reset	R/W	0x0
Section 23.3.2, “DRAM Clock Rate Registers”				
0x09	0x24	GPIO DRAM1 PLL Enable Register	R/W	0x1
0x0A	0x28	GPIO DRAM1 PLL Ratio Register	R/W	
0x0B	0x2C	GPIO DRAM1 PLL Reset Register	R/W	0x1
0x0C	0x30	GPIO DRAM1 PLL status Register	RO	
0x0D	0x34	GPIO DRAM2 PLL Enable Register	R/W	0x1
Section 23.3.3, “PLL Control and Status Registers”				
0x0E	0x38	GPIO Reserved Register		
0x0F	0x3C	GPIO PCIe PLL Control Register	R/W	
		XLS6xx, XLS4xx-Lite and XLS4xx		0x125
		XLS2xx and XLS1xx		0x04125
0x10	0x40	GPIO SGMII PLL Control Register	R/W	0x7141
Section 23.3.4, “Clock Divider Control Registers”				
0x11	0x44	GPIO Compression/Decompression Engine CLK Divider Register	R/W	0x2
0x12	0x48	GPIO Security Accelerator Engine CLK Divider Register	R/W	0x0
0x13	0x4C	GPIO Security Accelerator Engine ECC CLK Divider Register	R/W	0x2
0x14	0x50	GPIO DMA Engine CLK Divider Register	R/W	0x3
Section 23.3.5, “GPIO System Control Registers”				
0x15	0x54	GPIO Power On Reset Configuration Register	RO	
0x16	0x58	GPIO Thermal Control and Status Register	R/W	0x0
0x17	0x5C	GPIO Thermal Shift Pattern Register	R/W	0xaaaa

Table 23-3. Summary of GPIO Interface Registers

Register ID	Register Address Offset (hex)	Name and Description	RW	Reset Value
0x18	0x60	GPIO Bist ALL GO Status	RO	
0x19	0x64	GPIO Bist EACH GO Status	RO	
0x1A	0x68	GPIO Reserved Register	RO	0x0
0x1B	0x6C	GPIO Reserved Register		
0x1C	0x70	GPIO Reserved Register		
0x1D	0x74	GPIO Reserved Register		
0x1E	0x78	GPIO Reserved Register		
0x1F	0x7C	GPIO Reserved Register		
0x20	0x80	GPIO SGMII PHY Control Register	R/W	0x7e6801
0x21	0x84	GPIO Reserved Register	RO	0x0
0x22	0x88	GPIO Reserved Register	RO	0x0
0x23	0x8C	GPIO Fusebank register	RO	
0x24	0x90	GPIO SRIO Clock Control Register	R/W	0x0
0x25	0x94	GPIO Interrupt Mapper Register	R/W	0x0
0x26	0x98	GPIO External Interrupts register	R/W	0x0
0x27	0x9C	GPIO Reserved Register	RO	
0x28	0xA0	GPIO CPU Reset Register	R/W	0xfe
0x29	0xA4	Low Power Disable Register	RW	0x0
0x2A	0xA8	GPIO Reserved Register		
0x2B	0xAC	GPIO Random Number Generator Register	RO	0x0
0x2C	0xB0	GPIO Reserved Register	RO	0x0
0x2D	0xB4	CPU Clock Disable register	RW	0x0
0x2E	0xB8	GPIO Reserved Register	RO	0x0
0x2F	0xBC	GPIO Reserved Register	RO	0x0
0x30	0xC0	GPIO Reserved Register	RO	0x0
0x31	0xC4	GPIO Reserved Register	RO	0x0
0x32	0xC8	GPIO Reserved Register	RO	0x0
0x33	0xCC	GPIO Reserved Register	RO	0x0
0x34	0xD0	GPIO Reserved Register	RO	0x0
0x35	0xD4	GPIO Reserved Register	RO	0x0
0x36	0xD8	GPIO Reserved Register	RO	0x0

23.3.1 General Purpose Input/Output Signal Registers

These registers configure the twenty-five General Purpose Input/Output signals in this interface.

23.3.1.1 GPIO Interrupt Enable

GPIO Register ID: 0x00

GPIO Register Address Offset: 0x000

This Read/Write register enables interrupts on the GPIO signal pins. The value after reset is 0x0

31:25	24:20	19:14	13:0	
Reserved	EN[24:20]	Reserved	EN[13:0]	
Bits	Field Name	Field Description	R/W	Reset
31:25	Reserved	Reserved	RO	0
24:20	EN[14:20]	Enable GPIO interrupts on the specified pins (bit field number is the same as GPIO pin): 0 = Disable interrupt 1 = Enable interrupt	R/W	0
19:14	Reserved	Reserved	RO	0
13:0	EN[13:0]	Enable GPIO interrupts on the specified pins (bit field number is the same as GPIO pin): 0 = Disable interrupt 1 = Enable interrupt	R/W	0

23.3.1.2 GPIO Input Inversion Control

GPIO Register ID: 0x01
GPIO Register Address Offset: 0x004

This Read/Write register selects Inversion of Inputs on the GPIO signal pins. The value after reset is 0x0].

31:25		24	23	22:20	19:14	13:0
<i>Reserved</i>		INV[24]	<i>Reserved</i>	INV[22:20]	<i>Reserved</i>	INV[13:0]
Bits	Field Name	Field Description			R/W	Reset
31:25	<i>Reserved</i>	<i>Reserved</i>			RO	0
24	INV[24]	Selects signal Inversion on specified pins (bit field number is the same as GPIO pin): 0 = Input is not Inverted 1 = Input is Inverted			R/W	0
23	<i>Reserved</i>	<i>Reserved</i>			RO	0
22:20	INV[22:20]	Selects signal Inversion on specified pins (bit field number is the same as GPIO pin): 0 = Input is not Inverted 1 = Input is Inverted			R/W	0
19:14	<i>Reserved</i>	<i>Reserved</i>			RO	0
13:0	INV[13:0]	Selects signal Inversion on specified pins (bit field number is the same as GPIO pin): 0 = Input is not Inverted 1 = Input is Inverted			R/W	0

23.3.1.3 GPIO IO Direction Control

GPIO Register ID: 0x02
GPIO Register Address Offset: 0x008

This Read/Write register selects the Input/Output Direction of the GPIO signal pins. The value after reset is 0x0

31:25	24	23	22:20	19:14	13:0	
<i>Reserved</i>	DIR[24]	<i>Reserved</i>	DIR[22:20]	<i>Reserved</i>	DIR[13:0]	
Bits	Field Name	Field Description			R/W	Reset
31:25	<i>Reserved</i>	<i>Reserved</i>			RO	0
24	DIR[24:0]	Specify signal direction on specified pin (bit field number is the same as GPIO pin): 0 = Input 1 = Output			R/W	0
23	<i>Reserved</i>	<i>Reserved</i>			RO	0
22:20	DIR[22:20]	Specify signal direction on specified pin (bit field number is the same as GPIO pin): 0 = Input 1 = Output			R/W	0
19:14	<i>Reserved</i>	<i>Reserved</i>			RO	0
13:0	DIR[13:0]	Specify signal direction on specified pin (bit field number is the same as GPIO pin): 0 = Input 1 = Output			R/W	0

23.3.1.4 GPIO Output Write

GPIO Register ID: 0x03
GPIO Register Address Offset: 0x00C

This Read/Write register Writes to the Output GPIO signal pins. The value after reset is 0x0.

31:25	24	23	22:20	19:14	13:0
Reserved	WVAL[24]	Reserved	WVAL[22:20]	Reserved	WVAL[13:0]

Bits	Field Name	Field Description	R/W	Reset
31:25	Reserved	Reserved. Reset value = b0000 000	RO	See left
24	WVAL[24]	Specify the value output on the specified pin (bit field number is the same as GPIO pin): 0 = Write a logical zero to the GPIO pin 1 = Write a logical one to the GPIO pin	R/W	0
23	Reserved	Reserved	RO	0
22:20	WVAL[22:20]	Specify the value output on the specified pin (bit field number is the same as GPIO pin): 0 = Write a logical zero to the GPIO pin 1 = Write a logical one to the GPIO pin	R/W	0
19:14	Reserved	Reserved	RO	0
13:0	WVAL[13:0]	Specify the value output on the specified pin (bit field number is the same as GPIO pin): 0 = Write a logical zero to the GPIO pin 1 = Write a logical one to the GPIO pin	R/W	0

23.3.1.5 GPIO Input Read

GPIO Register ID: 0x04
GPIO Register Address Offset: 0x010

This Read Only register shows the Input GPIO signal pin values. The value after reset is 0x0.

31:25	24:20	19:14	13:0
Reserved	RVAL[24:20]	Reserved	RVAL[13:0]

Bits	Field Name	Field Description	R/W	Reset
31:25	Reserved	Reserved	RO	0
24:20	RVAL[24:20]	Value of the Read input GPIO pins [24:20] (bit field number is the same as GPIO pin): 0 = Logical zero on the GPIO pin 1 = Logical one on the GPIO pin	RO	0
19:14	Reserved	Reserved	RO	0
13:0	RVAL[13:0]	Value of the Read input GPIO pins [13:0] (bit field number is the same as GPIO pin): 0 = Logical zero on the GPIO pin 1 = Logical one on the GPIO pin	RO	0

23.3.1.6 GPIO Interrupt Clear

GPIO Register ID: **0x05**
GPIO Register Address Offset: **0x014**

This Read/Write register Clears individual Interrupt status bits for GPIO signal pins. The value after reset is 0x0.

31:25	24:20	19:14	13:0
<i>Reserved</i>	CLR[24:20]	<i>Reserved</i>	CLR[13:0]

Bits	Field Name	Field Description	R/W	Reset
31:25	<i>Reserved</i>	<i>Reserved</i>	RO	0
24:20	CLR[22:20]	Clear specified GPIO interrupt status register bits (bit field number is the same as GPIO pin): 0 = Do not clear the selected interrupt status bit 1 = Clear the selected interrupt status bit	R/W	0
19:14	<i>Reserved</i>	<i>Reserved</i>	RO	0
13:0	CLR[13:0]	Clear specified GPIO interrupt status register bits (bit field number is the same as GPIO pin): 0 = Do not clear the selected interrupt status bit 1 = Clear the selected interrupt status bit	R/W	0

23.3.1.7 GPIO Interrupt Status (GPIO_INT_STATUS)

GPIO Register ID: 0x06

GPIO Register Address Offset: 0x018

This Read Only register reads the Interrupt Status bits for GPIO signal pins. The value after reset is 0x0000 0000.

31:25	26	25	24	23	22:20	19:14	13:0
Reserved	INT_LO_L	INT_HI_L	STS[24]	Thermal Interrupt	STS[22:20]	Reserved	STS[13:0]

Bits	Field Name	Field Description	R/W	Reset
31:27	Reserved	Reserved	RO	0
26	INT_LO_L	INT_HI_L Interrupt requested 0 = No interrupt requested 1 = Interrupt requested	RO	0
25	INT_HI_L	INT_LO_L Interrupt requested 0 = No interrupt requested 1 = Interrupt requested	RO	0
24	STS[24]	Interrupt status of GPIO pin 24 0 = No interrupt requested 1 = Interrupt requested	RO	0
23	Thermal Interrupt	Thermal Interrupt Requested 0 = No interrupt requested 1 = Interrupt requested	RO	0
22:20	STS[22:20]	Interrupt status of GPIO pins [22:20] 0 = No interrupt requested 1 = Interrupt requested	RO	0
19:14	Reserved	Reserved	RO	0
13:0	STS[13:0]	Interrupt status of GPIO pins [13:0] 0 = No interrupt requested 1 = Interrupt requested	RO	0

23.3.1.8 GPIO Interrupt Type

GPIO Register ID: 0x07
GPIO Register Address Offset: 0x01C

This Read/Write register determines the Interrupt detection Type for GPIO signal pins. The value after reset is 0x0.

31:25		24:0		
Reserved		ITYP[24], ITYP[22:20], ITP[13:0]		
Bits	Field Name	Field Description	R/W	Reset
31:25	Reserved	Reserved	RO	0
24	ITYP[24]	Select edge or level Interrupt detection Type on GPIO pin 24: 0 = Edge sensitive 1 = Level sensitive	R/W	0
23	Reserved	Reserved	RO	0
22:20	ITYP[22:20]	Select edge or level Interrupt detection Type on GPIO pins [22:20]: 0 = Edge sensitive 1 = Level sensitive	R/W	0
19:14	Reserved	Reserved	RO	0
13:0	ITYP[13:0]	Select edge or level Interrupt detection Type on GPIO pins [13:0]: 0 = Edge sensitive 1 = Level sensitive	R/W	0

23.3.1.9 GPIO Reset

GPIO Register ID: 0x08
GPIO Register Address Offset: 0x020

XLS soft reset; XLS reset triggerable from software. The value after system reset is 0x0.

31:1			0	
Reserved			RESET	
Bits	Field Name	Field Description	R/W	Reset
31:1	Reserved	Reserved	RO	b0000 0000 0000 0000 0000 0000 0000 0000
0	RESET	Software initiated reset of XLS: 0 = No action 1 = Apply software-triggered reset to XLS	R/W	b0

23.3.2 DRAM Clock Rate Registers

The basic DRAM controller clock rate is controlled and determined by the following registers in the GPIO Peripherals Interface registers group.

23.3.2.1 GPIO DRAM AB PLL Enable

GPIO Register ID: 0x09
GPIO Register Address Offset: 0x024

This Read/Write register enables the clock to the DRAM AB controller. The value after reset is 0x1.

31:1		0
<i>Reserved</i>		EN
Bits	Field Name	Field Description
31:1	Reserved	<i>Reserved</i> Reset value = b0000 0000 0000 0000 0000 0000 0000 000
0	EN	DRAM AB controller clock Enable: 0 = Disable the clock to the DRAM AB controller 1 = Enable the clock to the DRAM AB controller

23.3.2.2 GPIO DRAM AB PLL Clock Ratio

GPIO Register ID: 0x0A
GPIO Register Address Offset: 0x028

This Read/Write register determines the DRAM AB PLL clock ratio for various DRAM device clock rates. The value after reset is 0x0000033F.

31:11		10:8	7:0
<i>Reserved</i>			DIVQ DIVF
Bits	Field Name	Field Description	R/W Reset
31:7	Reserved	<i>Reserved</i> b0000 0000 0000 0000	RO See left
10:8	DIVQ	Output divider for DRAM AB PLL See Table 23-4 for values	R/W b011
7:0	DIVF	Feedback divider for DRAM AB PLL See Table 23-4 for values	R/W b00111111

Typical ratio values for common DRAM clock frequencies are in [Table 23-4](#).

Table 23-4. Typical DRAM AB PLL Ratio Values

Feedback Divider DIVF Field Value	Output Divider DIVQ Field Value	PLL Output
0x2f	0x3	200 MHz
0x37	0x3	233 MHz
0x3f	0x3	266 MHz
0x47	0x3	300 MHz
0x4f	0x3	333 MHz
0x57	0x3	366 MHz
0x2f	0x2	400 MHz

23.3.2.3 GPIO DRAM AB PLL Reset

GPIO Register ID: 0x0B

GPIO Register Address Offset: 0x02C

This Read/Write register resets the DRAM AB PLL. The value after reset is 0x0000 0001.

31:1	0
Reserved	RST_AB_PLL
Field Description	
31:1	Reserved Reset value = b0000 0000 0000 0000 0000 0000 0000 000
0	RS_AB_PLL Reset the DRAM AB PLL: 0 = Reset applied to DRAM AB PLL 1 = No reset applied to DRAM AB PLL

23.3.2.4 GPIO DRAM AB PLL Status

GPIO Register ID: 0x0C
GPIO Register Address Offset: 0x030

This Read Only register indicates the DRAM AB PLL lock Status. The value after reset is 0x0000 000X.

31:2			1	0
Reserved			LCK_STS	LCK_CNT
Bits	Field Name	Field Description	R/W	Reset
31:2	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0000 0000 000	RO	See left
1	LCK_STS	DRAM AB PLL lock Status: 0 = Unlocked 1 = Locked	RO	bX
0	LCK_CNT	PLL is locked for 256 cycles.	RO	

23.3.2.5 GPIO DRAM CD PLL Enable

GPIO Register ID: 0x0D
GPIO Register Address Offset: 0x034

This Read/Write register enables the clock to the DRAM CD controller. The value after reset is 0x0000 0001.

31:1			0	
Reserved			EN	
Bits	Field Name	Field Description	R/W	Reset
31:1	Reserved	Reserved. Reset value = b0000 0000 0000 0000 0000 0000 0000 000	RO	See left
0	EN	DRAM CD controller clock Enable: 0 = Disable the clock to the DRAM CD controller 1 = Enable the clock to the DRAM CD controller	R/W	b1

23.3.3 PLL Control and Status Registers

23.3.3.1 GPIO PCIE PLL Control

The details of this register depend on the XLS family as described in the following two sections.

GPIO PCIE PLL Control (XLS6xx, XLS4xx-Lite and XLS4xx)

GPIO Register ID: 0x0F

GPIO Register Address Offset: 0x03C

This Read/Write register controls the PCIE PLL. The value after reset is 0x125.

31:10		9	8:7	6:5	4:0
Reserved		PLL_VP12	PLL_PRE	PLL_NCYC5	PLL_NCYC
Bits	Field Name	Field Description			R/W
31:10	Reserved	Reserved			RO 0
9	Reserved	Reserved			R/W b0
8:7	PLL_PRE	PLL Prescale			R/W b10
6:5	PLL_NCYC5	PLL Ncyc5			R/W b01
4:0	PLL_NCYC	PLL Ncyc			R/W b00101

GPIO PCIE PLL Control (XLS2xx and XLS1xx)

GPIO Register ID: 0x0F

GPIO Register Address Offset: 0x03C

This Read/Write register controls the PCIE PLL. The value after reset is 0x04125.

31:20		19:16	15:11	10	9	8:7	6:5	4:0
Reserved	PHY_TXC_ALN	PHY_ACJTG	PHY_RTUNE	PLL_VP12	PLL_PRE	PLL_NCYC5	PLL_NCYC	
Bits	Field Name	Field Description			R/W	Reset		
31:20	Reserved	Reserved			RO 0	0		
19:16	PHY_TXC_ALN	PHY TX Clk Align			R/W	b0000		
15:11	PHY_ACJTG	PHY Acjt LVL			R/W	b01000		
10	PHY_RTUNE	PHY RTune			R/W	b0		
9	PLL_VP12	PLL Is VP12			R/W	b0		
8:7	PLL_PRE	PLL Prescale			R/W	b10		
6:5	PLL_NCYC5	PLL Ncyc5			R/W	b01		
4:0	PLL_NCYC	PLL Ncyc			R/W	b00101		

23.3.3.2 GPIO SGMII[0-3]/XAUI[0] PLL Control

GPIO Register ID: 0x10
GPIO Register Address Offset: 0x040

This Read/Write register controls the PLL for SGMII ports 0 - 3 or XAUI port 0. The value after reset is 0x7104.

31:16	15	14:12	11:9	8:7	6:5	4:0
Reserved	PLL_VP12	PLL_PROP_CTL	PLL_INT_CTL	PLL_PRE	PLL_NCYC5	PLL_NCYC

Bits	Field Name	Field Description	R/W	Reset
31:16	Reserved	Reserved	RO	0
15	PLL_VP12	Reserved	R/W	0
14:12	PLL_PROP_CTL	Reserved	R/W	111
11:9	PLL_INT_CTL	Reserved	R/W	000
8:7	PLL_PRE	MPLL prescale value (see Table 23-5)	R/W	10
6:5	PLL_NCYC5	MPLL Ncyc5 (see Table 23-6)	R/W	00
4:0	PLL_NCYC	MPLL Ncyc (see Table 23-6)	R/W	100

Note: Bits 8:7 represent the two-bit MPLL prescale value, which defines what adjustment, if any, is done to the input Reference Clock (REFCLK) before the PLL stage. [Table 23-5](#) and the PLL Clocking section of the XLS Data Book define the options.

Note: Bits 6:5 and bits 4:0 represent the two programming fields that define how the PLL will divide the clock signal fed to it. [Table 23-6](#) defines the options.

Table 23-5. PLL Prescale [1:0] Decoding

MPLL_prescale[1:0] (PLL_PRE[8:7])	Description
'00'	Reference Clock is fed to the MPLL as-is.
'01'	Reference Clock frequency is doubled before feeding it to the MPLL.
'10'	Reference Clock frequency is divided by 2 before feeding it to the MPLL (use this for an input frequency of 125 MHz).
'11'	(Invalid)

Table 23-6. MPLL Divider Settings

MPLL Divider	MPLL_ncyc[4:0] (PLL_NCYC[4:0])	MPLL_Ncyc5[1:0] (PLL_NCYC5[6:5])
4	00000b	00b
5	00000b	01b
8	00001b	00b
9	00001b	01b
10	00001b	10b
12	00010b	00b
13	00010b	01b
14	00010b	10b
15	00010b	11b
16	00011b	00b
17	00011b	01b
18	00011b	10b
19	00011b	11b
20	00100b	00b
21	00100b	01b
22	00100b	10b
23	00100b	11b
24	00101b	00b
25	00101b	01b
26	00101b	10b
27	00101b	11b
28	00110b	00b
29	00110b	01b
30	00110b	10b
31	00110b	11b

Table 23-6. MPLL Divider Settings (continued)

MPLL Divider	MPLL_ncyc[4:0] (PLL_NCYC[4:0])	MPLL_Ncyc5[1:0] (PLL_NCYC5[6:5])
32	00111b	00b
33	00111b	01b
34	00111b	10b
35	00111b	11b
36	01000b	00b
37	01000b	01b
38	01000b	10b
39	01000b	11b
40	01001b	00b
41	01001b	01b
42	01001b	10b
43	01001b	11b
44	01010b	00b
45	01010b	01b
46	01010b	10b
47	01010b	11b
48	01011b	00b
49	01011b	01b
50	01011b	10b
51	01011b	11b
52	01100b	00b
53	01100b	01b
54	01100b	10b
55	01100b	11b
56	01101b	00b
57	01101b	01b
58	01101b	10b
59	01101b	11b
60	01110b	00b
61	01110b	01b
62	01110b	10b
63	01110b	11b

23.3.4 Clock Divider Control Registers

These registers in this group control clock dividers for the indicated modules.

23.3.4.1 GPIO CDE Clock Divider

GPIO Register ID: 0x11

GPIO Register Address Offset: 0x044

This Read/Write register controls the clock divider for the Compression/Decompression Engine. The value after reset is 0x2. (This register does not apply to XLS2xx and XLS1xx devices.)

		31:3	2:0	
		Reserved	CDE_CLK_DIV	
Bits	Field Name	Field Description	R/W	Reset
31:3	Reserved	Reserved	RO	0
2:0	CDE_CLK_DIV	Compression/Decompression Engine core clock divider ratio 000 = Reserved 001 = Reserved 010 = divide by 3 (default) 011 = divide by 4 100 = divide by 5 101 = divide by 6 110 = Reserved 111 = Reserved Note: These bits are Reserved in XLS2xx and XLS1xx devices.	R/W	b10

23.3.4.2 GPIO Security Accelerator Engine CLK Divider

GPIO Register ID: 0x12

GPIO Register Address Offset: 0x048

This Read/Write register controls the clock divider for the Security Acceleration Engine. The value after reset is 0x0.

		31:3	2:0	
		Reserved	SAE_CLK_DIV	
Bits	Field Name	Field Description	R/W	Reset
31:3	Reserved	Reserved	RO	0
2:0	SAE_CLK_DIV	Security Acceleration Engine core clock divider ratio 000 = divide by 1 (default) 001 = divide by 2 010 = divide by 3 011 = divide by 4 100 = divide by 5 101 = divide by 6 110 = Reserved 111 = Reserved	R/W	0

23.3.4.3 GPIO Security Accelerator Engine ECC CLK Divider

GPIO Register ID: 0x13
GPIO Register Address Offset: 0x04C

This Read/Write register controls the clock divider for the Security Acceleration Engine ECC. The value after reset is 0x2.

31:3			2:0	
Reserved			SAE_ECC_CLK_DIV	
Bits	Field Name	Field Description	R/W	Reset
31:3	Reserved	Reserved	RO	0
2:0	SAE_ECC_CLK_DIV	Security Acceleration Engine core clock divider ratio 000 = Reserved 001 = Reserved 010 = divide by 3 (default) 011 = divide by 4 100 = divide by 5 101 = divide by 6 110 = Reserved 111 = Reserved	R/W	0x2

23.3.4.4 GPIO DMA Clock Divider

GPIO Register ID: 0x14
GPIO Register Address Offset: 0x050

This Read/Write register sets the DMA Engine clock divider. The value after reset is 0x3.

31:3			2:0	
Reserved			DMA_CLK_DIV	
Bits	Field Name	Field Description	R/W	Reset
31:3	Reserved	Reserved	RO	0
2:0	DMA_CLK_DIV	DMA Engine clock divider ratio 000 = Reserved 001 = Reserved 010 = divide by 3 011 = divide by 4 (default) 100 = divide by 5 101 = divide by 6 110 = Reserved 111 = Reserved	R/W	0x3

23.3.5 GPIO System Control Registers

The following registers in the GPIO block govern or report status of various system functions in the XLS processors.

23.3.5.1 GPIO Reset Configuration

GPIO Register ID: 0x15

GPIO Register Address Offset: 0x054

This Read Only register shows the values latched from the IO_AD lines after a power-on reset. These values configure the system.

Note: Where values here differ from those in the XLS Data Book, refer to the Data Book for correct current usage.

31:28 27:26 23 22 21:20 19 18 17 16									
RSVD	PCEI_SRIO_SEL	RSVD	USB_DEV	PCIE_CFG SRIO_CFG	FLASH33	DIAG_EN	RUN_BIST	BOOT_NAND	
15 14 13 12 11 10:8 7:0									
BOOT_PCMCIA	FLASH_CFG	PCMCIA_EN	PARITY_EN	BIGEND	PLL1_OUT_DIV	PLL1_FB_DIV			

Table 23-7. GPIO Reset Configuration Register

Bits	Field Name	Field Description	R/W	Reset
31:28	Reserved	Reserved	RO	-
27:26	PCIE_SRIO_SEL	PCIe or SRIO Select: These bits reflect the status of pins IO-AD[5:4] at reset time. They Select either PCIe (default) or SRIO (for the XLS6xx and XLS4xx) 00 = PCIe selected, SRIO not available 01 = SRIO selected, frequency set at 1.25 Gbaud (1.0 Gbps) 10 = SRIO selected, frequency set at 2.25 Gbaud (2.0 Gbps) 11 = SRIO selected, frequency set at 3.125 Gbaud (2.5 Gbps)	RO	Note 1
25	XAUI_PORT1_SEL	XAUI Port 1 Select: This read-only status bit represents the XAUI Port 1 Enable strapping option (for the XLS6xx, XLS416 and XLS408). Its value is the level of the IO_AD6 pin, latched at reset time. The options are: 0 = Disabled - Port is configured as SGMII ports 4-7 1 = Enabled - Port is configured as a 4-lane XAUI Port 1	RO	Note 1
24	XAUI_PORT0_SEL	XAUI Port 0 Select: This read-only status bit represents the XAUI Port 0 Enable strapping option (for the XLS6xx and XLS416). Its value is the level of the IO_AD7 pin, latched at reset time. The options are: 0 = Disabled - Port is configured as SGMII ports 0-3 1 = Enabled - Port is configured as a 4-lane XAUI Port 0	RO	Note 1
23	Reserved	Reserved	RO	-
22	USB_DEV	USB Device: This read-only status bit represents the USB interface Host/Device Mode selection strapping option. Its value is the level of the IO_AD9 pin, latched at reset time. The options are: 0 = Run Device mode operation 1 = Run Host Mode operation	RO	Note 1

Table 23-7. GPIO Reset Configuration Register (*continued*)

Bits	Field Name	Field Description	R/W	Reset
21:20	PCIE_CFG or SRIO_CFG	<p>PCIe CFG (root/device cfg): These two read-only status bits represent the PCI Express Root/Device Configuration strapping option. Their values are the levels of the IO_AD[11:10] pins, latched at reset time. The options are:</p> <p><u>For XLS6xx and XLS4xx</u></p> <ul style="list-style-type: none"> 00 = End Point x4 01 = Root Complex x4 10 = End Point x1, Root Complex x3 11 = Root Complex x4 <p><u>For XLS4xx-Lite</u></p> <ul style="list-style-type: none"> 00 = End Point x4 01 = Root Complex x4 10 = End Point x1 11 = Root Complex x1 <p><u>For XLS2xx</u></p> <ul style="list-style-type: none"> 00 = End Point x1, Root Complex x3 01 = Root Complex x4 10 = End Point x1, Root Complex x3 (same as '00') 11 = Root Complex x4 (same as '01') <p><u>For XLS1xx</u></p> <ul style="list-style-type: none"> 00 = End Point x1, Root Complex x1 01 = Root Complex x2 10 = End Point x1, Root Complex x1 (same as '00') 11 = Root Complex x2 (same as '01') 	RO	Note 1
19	FLASH33_EN	Flash 33 MHZ Enable: This read-only status bit represents the FLASH interface speed selection strapping option. Its value is the level of the IO_AD12 pin, latched at reset time. The options are:	RO	Note 1
18	BIST_DIAG_ENABLE	BIST Diagnostics Enable: This read-only status bit represents the BIST Enable strapping option. Its value is the level of the IO_AD13 pin, latched at reset time. The options are:	RO	Note 1
17	BIST_RUN_ENABLE	BIST RUN Enable: This read-only status bit represents the BIST Run Enable strapping option. Its value is the level of the IO_AD14 pin, latched at reset time. The options are:	RO	Note 1
16	BOOT_NAND	Boot from NAND: This read-only status bit represents the BOOT FROM NAND FLASH strapping option. Its value is the level of the IO_AD15 pin, latched at reset time. The options are:	RO	Note 1

Table 23-7. GPIO Reset Configuration Register (*continued*)

Bits	Field Name	Field Description	R/W	Reset
15	BOOT_PCMCIA	Boot from PCMCIA: This read-only status bit represents the BOOT FROM PCMCIA strapping option. Its value is the level of the IO_AD16 pin, latched at reset time. The options are: 0 = disable boot from PCMCIA 1 = enable boot from PCMCIA NOTE: The IO_AD16 pin (and therefore this bit) must be '0' in order for the XLS to boot from NAND Flash by strapping IO_AD15 = 1 at reset time.	RO	Note 1
14	FLASH_CFG	Flash 32-bit Data Configuration: This read-only status bit represents the 32-bit Flash Data strapping option. Its value is the level of the IO_AD17 pin, latched at reset time. The options are: 0 = 32-bit address / 16-bit data 1 = 32-bit address / 32-bit data NOTE: The level of IO_AD17 does not alter the Flash interface Address bus width.	RO	Note 1
13	PCMCIA_EN	PCMCIA Enable Status: This read-only status bit represents the PCMCIA Enable strapping option. Its value is the level of the IO_AD18 pin, latched at reset time. The options are: 0 = PCMCIA Disabled 1 = PCMCIA Enabled	RO	Note 1
12	PARITY_EN	Parity Enable Status: This read-only status bit represents the Parity Check Enable strapping option. Its value is the level of the IO_AD19 pin, latched at reset time. The options are: 0 = Parity Disabled 1 = Parity Enabled Note that if this bit is set to '1' (either by software or by strapping IO_AD19 HIGH during reset), then GPIO[13:10] cannot be used for interrupts. See Section 22.4.2.1 for a discussion of this topic.	RO	Note 1
11	BIGEND	Big Endian Mode Enable Status: This read-only status bit represents the Big Endian Mode Enable strapping option. Its value is the level of the IO_AD20 pin, latched at reset time. The options are: 0 = Little Endian Mode 1 = Big endian Mode	RO	Note 1
10:8	PLL1_OUT_DIV	PLL1 (Core PLL) Output Divider: These read-only status bits represent the Core PLL Output Divider strapping option. Their values are the levels of the IO_AD[23:21] pins, latched at reset time. The collective value of these three bits defines the XLS Core PLL output divider. See the XLS Data Book for divider values.	RO	Note 1
7:0	PLL1_FB_DIV	PLL1 Feedback Divider: These read-only status bits represent the Core PLL Feedback Divider strapping option. Their values are the levels of the IO_AD[31:24] pins, latched at reset time. The collective value of these eight bits defines the XLS Core PLL feedback divider. See the XLS Data Book for divider values.	RO	Note 1

Note 1: Reset values depend upon strapping at reset time.

23.3.5.2 GPIO Thermal Control and Status

GPIO Register ID: 0x16
GPIO Register Address Offset: 0x058

This Read/Write register enables the Thermal Finite State Machine which controls the reduction in the system clock rate to reduce power consumption using the shift pattern in the companion GPIO Thermal Shift Pattern register. Also allows software to initiate thermal interrupt. The value after reset is 0x0000 0000.

31:2		1	0
Reserved		SW_INT	THERMAL_FSM_EN

Bits	Field Name	Field Description	R/W	Reset
31:2	Reserved	Reserved Reset value = b0000 0000 0000 0000 0000 0000 0000 00	RO	
1	SW_INT	Initiate thermal Interrupt (GPIO23) by internal Software setting of this bit.	R/W	0
0	THERMAL_FSM_EN	Thermal FSM: 0 = Disable Thermal FSM 1 = Enable Thermal FSM	R/W	0

23.3.5.3 GPIO Thermal Shift Pattern

GPIO Register ID: 0x17
GPIO Register Address Offset: 0x05C

This Read/Write register contains the Shift Pattern used to reduce the system clock rate by the thermal finite state machine. The value after reset is 0x0000 AAAA.

31:16		15:0
Reserved		SHIFTPAT

Bits	Field Name	Field Description	R/W	Reset
31:16	Reserved	Reserved	RO	0x0000
15:0	SHIFTPAT	Shift pattern mask for the system clock to provide a repeated sequence with a lower clock rate for reduced power dissipation. AAAA -> divide by 2 8888 -> divide by 4. To use this value, you must turn off the THERMAL_FSM_EN bit in the GPIO Thermal Control and Status register, then program this register and then turn on the THERMAL_FSM_EN bit.	R/W	0xAAAA

23.3.5.4 GPIO_BIST_ALL_STATUS_

This Read Only register indicates all BIST Status. The value after reset is 0x0000 0000.

GPIO Register ID: 0x18

GPIO Register Address Offset: 0x060

31:2	1	0
<i>Reserved</i>	DONE	PASS_FAIL

Bits	Field Name	Field Description	R/W	Reset
31:2	<i>Reserved</i>	<i>Reserved</i>	RO	b0000 0000 0000 0000 0000 0000 0000 00
1	DONE	Status of All BIST complete: 0 = All BIST not done 1 = All BIST done	RO	b0
0	PASS_FAIL	Result of BIST: 0 = All BIST failed 1 = All BIST passed	RO	b0

23.3.5.5 GPIO_BIST_EACH_STATUS_2

GPIO Register ID: 0x19
GPIO Register Address Offset: 0x064

This Read Only register indicates BIST Status of individual modules. The value after reset is 0x0000 0000.

31	30	29	28	27	26	25	24	23	22	21	20
RSVD	USB_PHY_BIST	GIO_BIST	DRC11_BIST	DRC10_BIST	DRC01_BIST	DRC00_BIST	CDE_BIU_BIST	CDE_CORE_RF_BIST	CDE_CORE_RAM_BIST	PCIE_BIU_BIST	PCIE_CORE_BIST
19	18	17	16	15	15	13	12	11:8	7:4	3:2	1:0
DMA_BIST	USB_BIST	SEC_ECC_BIST	SEC_BIST	GMAC1_BIST	GMAC0_BIST	NBRS_BIST	NBRB_BIST	L2_RAM_BIST	L2_RF_BIST	CPU_RAM_BIST	CPU_RF_BIST
Bits	Field Name		Field Description							R/W	Reset
31	<i>Reserved</i>		<i>Reserved</i>							RO	0
30	USB_PHY_BIST		USB PHY BIST passed Status 0 = BIST failed 1 = BIST passed								0
29	GIO_BIST		GIO BIST passed Status 0 = BIST failed 1 = BIST passed							RO	0
28	DRC11_BIST		DRC11 BIST passed Status 0 = BIST failed 1 = BIST passed							RO	0
27	DRC10_BIST		DRC10 BIST passed Status 0 = BIST failed 1 = BIST passed							RO	0
26	DRC01_BIST		DRC01 BIST passed status 0 = BIST failed 1 = BIST passed							RO	0
25	DRC00_BIST		DRC00 BIST passed status 0 = BIST failed 1 = BIST passed							RO	0
24	CDE_BIU_BIST		Compression/Decompression Engine BIU_BIST passed Status 0 = BIST failed 1 = BIST passed Note: This bit is <i>Reserved</i> in XLS2xx and XLS1xx devices.							RO	0
23	CDE_RF_BIST		Compression/Decompression Engine RF_BIST passed status 0 = BIST failed 1 = BIST passed Note: This bit is <i>Reserved</i> in XLS2xx and XLS1xx devices.							RO	0
22	CDE_CORE_RAM_BIST		Compression/Decompression Engine RAM_BIST passed status 0 = BIST failed 1 = BIST passed Note: This bit is <i>Reserved</i> in XLS2xx and XLS1xx devices.							RO	0

Bits	Field Name	Field Description	R/W	Reset
21	PCIE_BIU_BIST	PCIE_BIU_BIST passed status 0 = BIST failed 1 = BIST passed	RO	0
20	PCIE_CORE_BIST	PCIE_CORE_BIST passed status 0 = BIST failed 1 = BIST passed	RO	0
19	DMA_BIST	DMA BIST status 0 = BIST failed 1 = BIST passed	RO	0
18	USB_BIST	USB_BIST status 0 = BIST failed 1 = BIST passed	RO	0
17	SEC_ECC_BIST	SEC_ECC_BIST 0 = BIST failed 1 = BIST passed	RO	0
16	SEC_BIST	Security Acceleration Engine BIST 0 = BIST failed 1 = BIST passed	RO	0
15	GMAC1_BIST	GMAC1_BIST 0 = BIST failed 1 = BIST passed Note: This bit is Reserved in XLS2xx and XLS1xx devices.	RO	0
14	GMAC0_BIST	GMAC0_BIST 0 = BIST failed 1 = BIST passed	RO	0
13	NBRS_BIST	NBRS (Bridge Top) BIST passed Status 0 = BIST failed 1 = BIST passed	RO	0
12	NBRB_BIST	NBRB (Bridge Bottom) BIST passed status 0 = BIST failed 1 = BIST passed	RO	0
11:8	L2_RAM_BIST	L2_RAM_BIST 0 = BIST failed 1 = BIST passed	RO	0
7:4	L2_RF_BIST	L2_RF_BIST 0 = BIST failed 1 = BIST passed	RO	0
3:2	CPU_RAM_BIST	CPU_RAM_BIST 0 = BIST failed 1 = BIST passed	RO	0
1:0	CPU_RF_BIST	CPU_RF_BIST 0 = BIST failed 1 = BIST passed	RO	0

23.3.5.6 GPIO_SGMII[0-3]/XAUI[0]_PHY_CTL

GPIO Register ID: 0x20
GPIO Register Address Offset: 0x080

This Read/Write register sets the PHY parameters for SGMII ports 0 - 3 or XAUI port 0. The value after reset is 0x7E6801

31:23	22:18	17:13	12:8	7	6:3	2:0
RSVD	PHY Tx LVL	PHY Los LVL	PHY Acjt LVL	PHY RTune	PHY TX Clk Align	PHY CKO word con

Bits	Field Name	Field Description	R/W	Reset
31:23	Reserved			0
22:18	PHY Tx LVL	Transmit Signal Level. Fine resolution setting of transmit signal level, common to all lanes connected to one support core Pk-Pk output level (without attenuation) is $1180 * (48 + tx_lvl/2) / 63.5\text{mV}$		0x37
17:13	PHY Los LVL	LOS detection threshold. Recommended and reset value is decimal 19		0x13
12:8	PHY Acjt LVL	AC JTAG Level. This field is used only during wafer testing. Recommended value 8.		0x08
7	PHY RTune	Resistor Tuning. Rising edge causes cycling of the tuning SAR May be transitioned asynchronously. Must be held high for at least 2 cycles of SGMII or XAUI reference clock.		0
6:3	PHY TX Clk Align	Rising edge is a request for realignment of the internal Tx word clock applied to the TX_CK signal. This field is needed only if the TC_CK path is interrupted or changed after the automatic alignments. Automatic alignments are triggered by transitions of TX_EN[2:0], which enable the transmit clocks. This field may be transitioned asynchronously and should be kept high for at least two cycles of SGMII or XAUI reference clock.		0
2:0	PHY CKO word con	This field controls which clock is driven on the CKO_WORD output. Changing between selections will cause CKO_WORD to transition from one clock to the other glitchlessly. 0: Buffered version of prescaler output clock 1: PLL frequency/5 clock 2: PLL frequency/10 clock 3: Undefined >3: Disabled		1

23.3.5.7 GPIO SGMII[4-7]/XAUI[1] PLL Control**GPIO Register ID:** 0x21**GPIO Register Address Offset:** 0x084

This Read/Write register controls the PLL for SGMII ports 4 - 7 or XAUI port 1. The value after reset is 0x7104.

31:16	15	14:12	11:9	8:7	6:5	4:0
Reserved	PLL_VP12	PLL_PROP_CTL	PLL_INT_CTL	PLL_PRE	PLL_NCYC5	PLL_NCYC

Bits	Field Name	Field Description	R/W	Reset
31:16	Reserved	Reserved	RO	0
15	PLL_VP12	Reserved	R/W	0
14:12	PLL_PROP_CTL	Reserved	R/W	111
11:9	PLL_INT_CTL	Reserved	R/W	000
8:7	PLL_PRE	MPLL prescale value (see Table 23-5)	R/W	10
6:5	PLL_NCYC5	MPLL Ncyc5 (see Table 23-6)	R/W	00
4:0	PLL_NCYC	MPLL Ncyc (see Table 23-6)	R/W	100

Note: Bits 8:7 represent the two-bit MPLL prescale value, which defines what adjustment, if any, is done to the input Reference Clock (REFCLK) before the PLL stage. [Table 23-5](#) and the PLL Clocking section of the XLS Data Book define the options.

Note: Bits 6:5 and bits 4:0 represent the two programming fields that define how the PLL will divide the clock signal fed to it. [Table 23-6](#) defines the options.

23.3.5.8 GPIO_SGMII[4-7]/XAUI[1]_PHY_CTL

GPIO Register ID: 0x22
GPIO Register Address Offset: 0x088

This Read/Write register sets the PHY parameters for SGMII ports 4 - 7 or XAUI port 1. The value after reset is 0x7E6801

31:23	22:18	17:13	12:8	7	6:3	2:0
RSVD	PHY Tx LVL	PHY Los LVL	PHY Acjt LVL	PHY RTune	PHY TX Clk Align	PHY CKO word con

Bits	Field Name	Field Description	R/W	Reset
31:23	Reserved			0
22:18	PHY Tx LVL	Transmit Signal Level. Fine resolution setting of transmit signal level, common to all lanes connected to one support core Pk-Pk output level (without attenuation) is $1180 * (48 + tx_lvl/2) / 63.5\text{mV}$		0x37
17:13	PHY Los LVL	LOS detection threshold. Recommended and reset value is decimal 19		0x13
12:8	PHY Acjt LVL	AC JTAG Level. This field is used only during wafer testing. Recommended value 8.		0x08
7	PHY RTune	Resistor Tuning. Rising edge causes cycling of the tuning SAR May be transitioned asynchronously. Must be held high for at least 2 cycles of SGMII or XAUI reference clock.		0
6:3	PHY TX Clk Align	Rising edge is a request for realignment of the internal Tx word clock applied to the TX_CK signal. This field is needed only if the TC_CK path is interrupted or changed after the automatic alignments. Automatic alignments are triggered by transitions of TX_EN[2:0], which enable the transmit clocks. This field may be transitioned asynchronously and should be kept high for at least two cycles of SGMII or XAUI reference clock.		0
2:0	PHY CKO word con	This field controls which clock is driven on the CKO_WORD output. Changing between selections will cause CKO_WORD to transition from one clock to the other glitchlessly. 0: Buffered version of prescaler output clock 1: PLL frequency/5 clock 2: PLL frequency/10 clock 3: Undefined >3: Disabled		1

23.3.5.9 GPIO_FUSEBANK

GPIO Register ID: 0x23
GPIO Register Address Offset: 0x08C

This read-only register stores the contents of the on-chip fusebank and allows the user a way to read the processor ID value.

In the XLS processor family, there are two production silicon revisions, A1 and B0. The bit assignments for this register differ depending on the product type and silicon revision of the particular device.

The bit assignments shown below are for the GPIO_FUSEBANK register used in silicon revision A1 and applies to the following products:

XLS408-Lite, XLS404-Lite, XLS208, XLS204, XLS108, and XLS104.

31:22		21:14	13:0	
Reserved		ProID	Reserved	
Bits	Field Name	Field Description	R/W	Reset
31:22	Reserved	Reserved	RO	n/a
21:14	ProID	8-bit Processor ID value	RO	n/a
13:0	Reserved	Reserved	RO	n/a

The bit assignments shown below are for the GPIO_FUSEBANK register used in silicon revision B0 and applies to the following products:

XLS616, XLS608, XLS416, XLS408, and XLS404.

31:24		23:16	15:0	
Reserved		ProID	Reserved	
Bits	Field Name	Field Description	R/W	Reset
31:24	Reserved	Reserved	RO	n/a
23:16	ProID	8-bit Processor ID value	RO	n/a
15:0	Reserved	Reserved	RO	n/a

23.3.5.10 GPIO_SRIO_CLK_CTRL

GPIO Register ID: 0x24
GPIO Register Address Offset: 0x090

This register determines the SRIO clock frequency. The value after reset is 0x0000 0000.

31:3		2:0
Reserved		SRIO_CLK_DIV
Bits	Field Name	Field Description
31:3	<i>Reserved</i>	<i>Reserved</i>
2:0	SRIO_CLK_DIV	SRIO Clock Divider. The SRIO clock frequency is the core clock frequency divided by (SRIO_CLK_DIV + 1).

23.3.5.11 GPIO_INT_MAP

GPIO Register ID: 0x25
GPIO Register Address Offset: 0x094

This Read/Write register determines the Mapping of GPIO Interrupts to the PIC. The value after reset is 0x0000 0000.

31:25	24:0	
Reserved	INT[24:0]	
Bits	Field Name	Field Description
31:25	<i>Reserved</i>	<i>Reserved</i>
24:0	INT[24:0]	For each GPIO pin, determines mapping of interrupt to PIC: 0 = GPIO pin Interrupt mapped to INT14 (PIC0) Interrupt 1 = GPIO pin Interrupt mapped to INT30 (PIC1) Interrupt

23.3.5.12 GPIO_EXT_INT

GPIO Register ID: 0x26
GPIO Register Address Offset: 0x098

This Read/Write register controls input and output functionality for interrupts on two pins, INT_LO_L and INT_HI_L. It determines masking for mapping to the PIC of external Interrupts on the pins. For the case where the pins are used as outputs to external device interrupt inputs, it controls pin High/Low levels. The value after reset is 0x0000 0000.

31:4		3	2	1	0	
Reserved		HI_MASK	LO_MASK	HI_CTL	LO_CTL	
Bits	Field Name	Field Description			R/W	Reset
31:2	Reserved	Reserved		RO	0x0000 000	
3	HI_MASK	INT_HI_L interrupt Mask: 0 = Interrupt permitted 1 = Interrupt masked When INT_HI_L pin is used as input for interrupt to the PIC, if this bit is '1', the XLS won't generate interrupt when pin is '0'			R/W	b0
2	LO_MASK	INT_LO_L interrupt Mask: 0 = Interrupt permitted 1 = Interrupt masked When INT_LO_L pin is used as input for interrupt to the PIC, if this bit is '1', the XLS won't generate interrupt when pin is '0'			R/W	b0
1	HI_CTL	Control INT_HI_L as interrupt output. 0 = no action 1 = generate interrupt on INT_HI_L by pulling the pin low.			R/W	b0
0	LO_CTL	Interrupt using INT_LO_L 0 = no action 1 = generate interrupt on INT_LO_L by pulling the pin low.			R/W	b0

Discussion:

INT_HI_L input interrupt is connected to the Programmable Interrupt Controller PIC0 interrupt (IRT14) and INT_LO_L input interrupt is connected to the PIC1 Interrupt (IRT30). INT_HI_L and INT_LO_L are open drain output signals. Each pin is bi-directional and can be used by XLS to trigger interrupts in the device the pin is connected to, or the XLS can be interrupted by an external device driving the pin. An active-low on the pin generates an interrupt.

The following describes behavior:

XLS taking interrupt:

When the XLS's GPIO controller sees a level '0' on the pin, it sends an interrupt to the PIC. If INT_HI_L is '0', then PIC0 (entry 14 in the PIC IRT table) is used. If INT_LO_L is '0', then PIC1 (entry 30 in the PIC IRT table) is used to finally set the EIRR/EMRR.

XLS driving interrupt:

When software writes a 1 to either HI_CTL or LO_CTL then the GPIO controller will drive the corresponding pin to '0'. The pin will be held at 0 as long as the register field is at 1.

23.3.5.13 GPIO_CPU_RST

GPIO Register ID: 0x28
GPIO Register Address Offset: 0x0A0

This Read/Write register resets the individual XLS cores. The value after reset is 0x0000 00FE.

31:4	3	2	1	0
Reserved	CPU3_RST	CPU2_RST	CPU1_RST	CPU0_RST

Bits	Field Name	Field Description	R/W	Reset
31:4	Reserved	Reserved	R/W	0x0000 00
3	CPU3_RST	Reset for CPU3	R/W	b1
2	CPU2_RST	Reset for CPU2	R/W	b1
1	CPU1_RST	Reset for CPU1	R/W	b1
0	CPU0_RST	Reset for CPU0	R/W	b0

23.3.5.14 LOW_PWR_DIS

The details of this register depend on the XLS family as described in the following two sections.

LOW_PWR_DIS for XLS6xx, XLS4xx-Lite and XLS4xx Devices

GPIO Register ID: 0x29

GPIO Register Address Offset: 0x0A4

This Read/Write register can be used to disable XLS functional blocks in a design by clock gating to reduce power consumption. However, it should only be used on unimplemented interfaces. The value after reset is 0x0000 0000.

31:9	8:7	6	5	4	3	2	1	0
Reserved	SAE_DIS	DMA_DIS	CDE_DIS	PCIE_DIS	USB_DIS	GMAC_QD_1_DIS	GMAC_QD_0_DIS	LP_DIS
Field Description								
Bits	Field Name							R/W
31:9	Reserved	Reserved						RO
8:7	SAE_DIS	Disable Security Acceleration Engine 00 = enable 01 = Reserved 10 = Reserved 11 = disable SA Engine						R/W
6	DMA_DIS	Disable DMA Engine 0 = enable 1 = disable DMA Engine. Does not disable DMA in other modules.						R/W
5	CDE_DIS	Disable Compression/Decompression Engine 0 = enable 1 = disable CDE						R/W
4	PCIE_DIS	Disable PCI Express interface 0 = enable 1 = disable PCIE						R/W
3	USB_DIS	Disable USB interface 0 = enable 1 = disable USB						R/W
2	GMAC_QD_1_DIS	GMAC Quad 1 (GMAC_4 through GMAC_7) disable 0 = enable 1 = disable						R/W
1	GMAC_QD_0_DIS	GMAC Quad 0 (GMAC_0 through GMAC_3) disable 0 = enable 1 = disable						R/W
0	LP_DIS	Low Power Disable. Allows disabling 'power reduction by clock gating' functionality. 0 = enable low power mode (default) 1 = disable low power mode.						R/W

LOW_PWR_DIS for XLS2xx and XLS1xx Devices**GPIO Register ID:** 0x29**GPIO Register Address Offset:** 0x0A4

This Read/Write register can be used to disable XLS functional blocks in a design by clock gating to reduce power consumption. However, it should only be used on unimplemented interfaces. The value after reset is 0x0000 0000.

31:12	11:10	9	8	7:4	3	2	1	0
Reserved	SAE_DIS	DMA_DIS	CDE_DIS	PCIE_DIS	USB_DIS	GMAC_QD_1_DIS	GMAC_QD_0_DIS	LP_DIS

Bits	Field Name	Field Description	R/W	Reset
31:12	Reserved	Reserved	RO	b0
11:10	SAE_DIS	Disable Security Acceleration Engine 00 = enable 01 = Reserved 10 = Reserved 11 = disable SA Engine	R/W	0
9	DMA_DIS	Disable DMA Engine. 0 = enable 1 = disable DMA Engine. Does not disable DMA in other modules.	R/W	0
8	CDE_DIS	Disable Compression/Decompression Engine 0 = enable 1 = disable CDE Note: This bit is Reserved in XLS2xx and XLS1xx devices.	R/W	0
7:4	PCIE_DIS	Disable PCI Express interface 0 = enable 1 = disable PCIE	R/W	0
3	USB_DIS	Disable USB interface 0 = enable 1 = disable USB	R/W	0
2	GMAC_QD_1_DIS	GMAC Quad 1 (SGMII_4 through SGMII_7) disable 0 = enable 1 = disable Note: This bit is Reserved in XLS2xx and XLS1xx devices.	R/W	0
1	GMAC_QD_0_DIS	GMAC Quad 0 (SGMII_0 through SGMII_3) disable 0 = enable 1 = disable	R/W	0
0	LP_DIS	Low Power Disable. Allows disabling 'power reduction by clock gating' functionality. 0 = enable low-power mode (default) 1 = disable low-power mode.	R/W	0

23.3.5.15 GPIO_RANDOM

GPIO Register ID: 0x2B
GPIO Register Address Offset: 0x0AC

This Read Only register provides a 32-bit output of the Random Number Generator. The initial value after reset is undetermined.

31:0	RAN			
Bits	Field Name	Field Description	R/W	Reset
31:0	RAN	Random number output	RO	-

23.3.5.16 GPIO_CPU_CLK_DIS

GPIO Register ID: 0x2D
GPIO Register Address Offset: 0x0B4

This Read/Write register disables individual CPU Clocks. Depending on number of CPUs in the processor model, some bits will have no effect. The value after reset is 0x0000 0000.

31:2	3	2	1	0	
Reserved	DIS_CPU3	DIS_CPU2	DIS_CPU1	DIS_CPU0	
Bits	Field Name	Field Description		R/W	Reset
31:4	Reserved	Reserved		RO	0x0000 00
3	DIS_CPU1	Disable CPU3 by setting this bit to b1		R/W	b0
2	DIS_CPU1	Disable CPU2 by setting this bit to b1		R/W	b0
1	DIS_CPU1	Disable CPU1 by setting this bit to b1		R/W	b0
0	DIS_CPU0	Disable CPU0 by setting this bit to b1		R/W	b0

NETLOGIC
CONFIDENTIAL



Chapter 24 IEEE 1588 Precision Time Protocol

24.1 Introduction

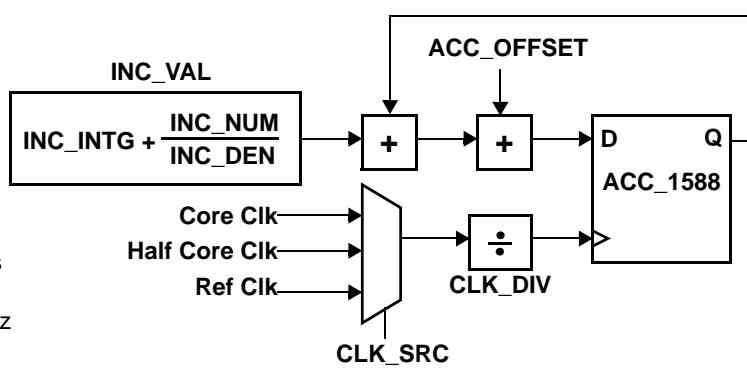
The XLS6xx and XLS4xx support the IEEE1588 standard, version 2, which is used to very-precisely synchronize the clocks of various devices within a network. The resolution is on the order of a few hundred nano seconds. The process follows these steps:

1. The Master sends a SYNC_MESSAGE to the slave device. The Master detects/samples the precise time that the SYNC_MESSAGE leaves the interface.
2. The Slave receives the SYNC_MESSAGE and records the precise time when it received the message.
3. The Master then sends a FOLLOW_UP message containing the precise time when the SYNC_MESSAGE (SOP) left the master interface.
4. The Slave sends a DELAY_REQ message to the Master. The Slave detects/samples the precise time that the DELAY_REQ message leaves the interface.
5. The Master samples the exact time it receives the DELAY_REQ message. It then sends back a DELAY_RESP message containing the exact time the DELAY_REQ message was received.
6. The Slave then uses the following information to synchronize its clock to the Master clock:
 - a. Exact time the SYNC_MESSAGE was received by the Slave
 - b. The Time field in the FOLLOW_UP message received by the Slave
 - c. Exact time when the DELAY_REQ left the interface.
 - d. The Time field in the DELAY_RESP message received by the Slave

The logic diagram of the on-chip PTP controller is shown in [Figure 24-1](#).

Figure 24-1. PTP Controller Logic Diagram

- INC_VAL is the increment value added to the ACC_1588 register every clock cycle. This is made up of the following registers:
 - INC_INTG the integer part
 - INC_NUM the fractional numerator
 - INC_DEN the fractional denominator where $\text{INC_NUM} < \text{INC_DEN}$
- ACC_OFFSET value is added to the ACC_1588 register each time the ACC_OFFSET is updated by software.
- ACC_1588 is a read-only register that returns the current value of the timer.
- CLK_SRC is used to select between the 1 GHz core clock, half core clock, and the 125 MHz reference clock.
- CLK_DIV is used to divide the clock source



24.1.1

Benefits of IEEE 1588 Version 2

Many telecommunications applications are waiting for IEEE 1588 V2 enhancements such as **unicast messaging** and **shorter frame sizes** to reduce the bandwidth consumed by the protocol messaging. Version 2 also introduces new PTP devices called **transparent clocks** that are used to prevent error accumulation in cascaded topologies, and Version 2 provides **extensions to enable redundant systems**.

The best PTP performance can be achieved by minimizing the number of nodes between the clock source and the slave devices. When devices are stacked in a hierarchy, error accumulates at each level and low-accuracy intermediate devices will dominate the error experienced by the end devices.

Since hierarchical architectures are common in public and private IP networks, IEEE1588 V2 is being developed with considerable input from experts in Telecomm, Networking, test/measurement, and Industrial Controls, who understand the special requirements of large distributed networks with many hops and inherent redundancy. This experienced team is now helping to specify support for **new features** such as:

- V2 Grand Master Clock with Telecom profiles
- Sub-nanosecond accuracy, which is especially useful for synchronizing test and measurement equipment.
- Faster synchronization (SYNC) message rates. Unlike V1, which specifies that SYNC messages occur no faster than once a second, V2 defines a much wider range of mean SYNC message rates, even allowing for rates that are much greater than 1000 messages per second. However, most systems are expected to use mean SYNC message rates much less than this to reduce network traffic and optimize between network loading and synchronization performance. Faster SYNC message rates generally require more hardware timestamp assistance, and the CPU needs more performance to process the increased number of messages. Faster SYNC message rates are critical for telecommunications, residential Ethernet, and many control applications.
- Shorter PTP messages, unicast messaging, new messages (path delay request/response/response follow-up) and message fields. These all reduce network bandwidth overhead and the resulting potential network queuing delays. These message optimizations are critical for telecommunications, residential Ethernet, and many control applications.
- Transparent Clocks (TC), which can be used to prevent error accumulation in cascaded topologies. Network-induced jitter can occur when each packet takes a slightly different time to pass through a network switch.

Resulting time sync degradation can be avoided if switches implement a transparent clock at each network hop. End-to-end & peer-to-peer Transparent Clock devices calculate the delay incurred by a packet passing through the device (called the residence time) and update a time correction field in the PTP payload to account for this delay. Thus, each switch appears to be a “wire” which does not skew the time calculation for packets passing through them.

- Fault Tolerance, which is needed to ensure no single network element failure can cause sub-tending clocks to fail. In a system with redundant grandmasters, the redundant grandmaster must be able to detect failure of the first grandmaster, and the sub-tending clocks must be able to switch from one grandmaster to another without incurring unacceptable frequency or phase jumps, or losing the clock for sub-tending nodes. Fault tolerance is critical in Telecom applications and many control applications, where safety and reliability are required.
- TLV extension to extend the protocol features/options. This allows the protocol to support extensions to meet the requirements of future applications.
- Profiles. These define what PTP features and settings are needed for different market applications.

24.1.2

Boundary and Transparent Clocks

Boundary Clocks were defined in IEEE 1588 V1 to support synchronization within networks containing several subnets. Boundary clocks typically have more than two ports, with one port serving as a PTP slave to an upstream clock master, and the other ports serving as PTP clock masters to downstream PTP slaves. Boundary Clocks employ a servo loop to recover the clock on the slave port, and then the recovered clock feeds Boundary Clock ports serving as PTP clock masters.

Due to the cumulative effect of multiple servo loops, cascading multiple Boundary Clocks within a network can significantly degrade clock accuracy of the system. Quality of the crystal oscillator used in a Boundary Clock can also impact the accuracy of the PTP clocks in the system. If a poor quality crystal is used in a Boundary Clock, PTP slaves attached to the Boundary Clock device will likely have inferior synchronization to the PTP clock master.

Transparent Clocks were added in IEEE 1588 V2 to update and append a timing correction field within certain message. The correction field is updated with a value representing the “residence time” required for that message to pass through the Transparent Clock.

Here is some additional information about Boundary and Transparent clocks:

- Boundary Clocks do not propagate Sync, Follow_Up, Delay_Req, or Delay_Resp messages.
- Boundary Clocks are suitable for topologies with a small number of switches.
- Cascading Boundary Clocks introduce the cascade effect. A Boundary Clock distributes timing based on its local clock, and each clock depends on the quality of all preceding clocks.
- Boundary Clocks can cause accuracy and stability problems in highly cascaded or daisy-chained network topologies because Boundary Clocks contain IEEE 1588 control loops. Cascading Boundary Clocks is just like cascading phase locked loops in the sense that the jitter starts to accumulate down the chain.
- Transparent Clocks do not generate timestamps. Instead they update correction fields in event messages according to the residence time of the event message within the Transparent Clock.
- The correction field will accumulate the value of the residence multiplied by the number of cascaded Transparent Clocks. Ordinary and boundary clocks account for the value of the correction field in their calculations. This results in reduced jitter and increased clock accuracy and stability.
- There are two different types of Transparent Clocks: End-to-End (E2E) and Peer-to-Peer (P2P). P2P Transparent Clocks are different than E2E Transparent Clocks in that each port on a P2P node computes the peer link path delay with its link partner on another P2P node.
- The link path delay calculation is accomplished through new V2 messages: PDELAY_REQ, PDELAY_RESP, and PDELAY_RESP_FOLLOWUP (optional). P2P Transparent Clocks add the ingress link path delay time to the value in the correction field (in addition to the residence time of the message).
- P2P Transparent Clocks only work with nodes supporting the peer delay function. This means that they work with ordinary and boundary clocks that support the peer delay function, as well as other P2P Transparent Clocks.

24.1.3 XLS6xx and XLS4xx PTP Programming Registers

The XLS IEEE1588 PTP function is accessible through the SGMII or the XAUI controllers and the Networking Accelerator. The registers summarized in [Table 24-1](#) are used to set up and monitor the frequency timing relationships between the XLS device and other devices connected through the SGMII or XAUI ports.

Table 24-1. IEEE1588 PTP Register Summary

Name	Register ID	Address Offset	Description	Notes
1588_PTP_INC_DEN	0x070	0x1C0	1588_PTP Clock Frequency set	
1588_PTP_INC_NUM	0x071	0x1C4		
1588_PTP_INC_INTG	0x072	0x1C8		
1588_PTP_OFFSET_LO	0x073	0x1CC	ACC_Offset value added to timer	Lower 32 bits of offset
1588_PTP_OFFSET_HI	0x074	0X1D0		Upper 32 bits of offset
1588_PTP_TX_LATCH_LSB	0x075	0x1D4	Lower 32-bits of Timer when last 1588_PTP Packet was sent out	
1588_PTP_TX_LATCH_MSB	0x076	0x1D8	Upper 32-bits of Timer when last 1588_PTP Packet was sent out	
1588_PTP_CONTROL	0x077	0x1DC	These bits activate timers and inserts values to reset timers	
1588_PTP_TMR1_LOW	0x078	0x1E0	Interrupt when Timer 1 times out	Lower 32 bits of timer
1588_PTP_TMR1_HI	0x079	0x1E4		Upper 32 bits of timer
1588_PTP_TMR2_LOW	0x07A	0x1E8	Interrupt when Timer 2 times out	Lower 32 bits of timer
1588_PTP_TMR2_HI	0x07B	0x1EC		Upper 32 bits of timer
1588_PTP_TMR3_LOW	0x07C	0x1F0	Interrupt when Timer 3 times out	Lower 32 bits of timer
1588_PTP_TMR3_HI	0x07D	0x1F4		Upper 32 bits of timer

24.1.4 IEEE-1588_PTP Registers

24.1.4.1 1588_PTP_INC_DEN

This is the Denominator of the fraction of the Incremental Value added to the ACC_1588 Timer Counter every clock cycle. (See [Figure 24-1, “PTP Controller Logic Diagram,” on page 1159.](#))

Register ID: 0x070

Address Offset: 0x1C0

Bits	Name	Description	R/W	Reset
31:0	INC_DEN	INC_INTG + (INC_NUM / INC_DEN)	R/W	0xFFFF_FFFF

24.1.4.2 1588_PTP_INC_NUM

This is the Numerator of the fraction of the Incremental Value added to the ACC_1588 Timer Counter every clock cycle. (See [Figure 24-1, “PTP Controller Logic Diagram,” on page 1159.](#))

Register ID: 0x071

Address Offset: 0x1C4

Bits	Name	Description	R/W	Reset
31:0	INC_NUM	INC_INTG + (INC_NUM / INC_DEN)	R/W	0xFFFF_FFFF

24.1.4.3 1588_PTP_INC_INTG

This is the Integer part of the Incremental Value added to the ACC_1588 Timer Counter every clock cycle. (See [Figure 24-1, “PTP Controller Logic Diagram,” on page 1159.](#))

Register ID: 0x072

Address Offset: 0x1C8

Bits	Name	Description	R/W	Reset
31:0	INC_INTG	INC_INTG + (INC_NUM / INC_DEN)	R/W	0x0

24.1.4.4 1588_PTP_OFFSET_LO

Register ID: 0x073

Address Offset: 0x1CC

Bits	Name	Description	R/W	Reset
31:0	TIMER_OFFSET_LO	This is the Lower 32 bits of the 64-bit offset (LSB) that added to the 1588 Timer Value	R/W	0x0

24.1.4.5 1588_PTP_OFFSET_HI

Register ID: 0x074

Address Offset: 0x1D0

Bits	Name	Description	R/W	Reset
31:0	TIMER_OFFSET_HI	This is the Upper 32 bits of the 64-bit offset (MSB) that added to the 1588 Timer Value	R/W	0x0

24.1.4.6 1588_PTP_TX_LATCH_LSB

Register ID: 0x075

Address Offset: 0x1D4

Bits	Name	Description	R/W	Reset
31:0	1588_PTP_TX_LATCH	Holds the lower 32 bits of the Timer, when the last 1588_PTP Packet was sent out.	R	0x0

24.1.4.7 1588_PTP_TX_LATCH_MSB

Register ID: 0x076

Address Offset: 0x1D8

Bits	Name	Description	R/W	Reset
31:0	1588_PTP_TX_LATCH	Holds the upper 32 bits of the Timer, when the last 1588_PTP Packet was sent out.	R	0x0

24.1.4.8 1588_PTP_CONTROL

Register ID: 0x077

Address Offset: 0x1DC

Bits	Name	Description	R/W	Reset
31:13	Reserved	Reserved	R	0x0
12	RTC_LATCH	For latching the RTC Timer value into PTP1588_TIMER_LATCH_VAL register	R/W	0
11	MDINT_LVM	Set this for low-voltage mode MDINT	R/W	0
10	MDC_OPN_DRN	Set external MDIO Clock pin status 1: external MDIO Clock pin is open drain 0: external MDIO Clock pin is standard clock output	R/W	'1' in XAUI Mode '0' in SGMII Mode
9	MDIO_LV_MODE	1: MDIO and MDINT are set to 1.2V for XAUI mode 0: MDIO and MDINT are set to 2.5V for SGMII mode	R/W	'1' in XAUI Mode '0' in SGMII Mode
8	MDIO_INT_ONLY	When set, external MDIO Clock is off	R/W	0
7	MDIO_EXT_ONLY	When set, internal MDIO is bypassed	R/W	0
6	TIMER_1_START	Set to '1' to Start Timer 1	R/W	0
5	TIMER_2_START	Set to '1' to Start Timer 2	R/W	0
4	TIMER_3_START	Set to '1' to Start Timer 3	R/W	0
3	SET_FREQ_MUL	Inserts the Frequency Multiplier Value to the Timer, this bit should be toggled from '0' to '1' for offset insertion. FREQ_MUL = [INC_INTG + (INC_NUM / INC_DEN)] Until this is set, the old values are used.	R/W	0

Bits	Name	Description	R/W	Reset
2	INSERT_OFFSET	Inserts the Offset Value to the Timer; this bit should be toggled from '0' to '1' for offset insertion.	R/W	0
1	USE_ALT_QUAD_TIMER	Use the Timer Value of other GMAC Quad	R/W	0
0	TIMER_RESET	Resets the Timer – The SET_FREQ_MUL must be toggled after the Reset to start incrementing	R/W	0

24.1.4.9 1588_PTP_TMR1_LOW

Register ID: 0x078

Address Offset: 0x1E0

Bits	Name	Description	R/W	Reset
31:0	TIMER_1[31:0]	Interrupt TIMER_1_1588 is generated when this Timer expires. The interrupt is a periodic Interrupt.	R	0x0

24.1.4.10 1588_PTP_TMR1_HI

Register ID: 0x079

Address Offset: 0x1E4

Bits	Name	Description	R/W	Reset
31:0	TIMER_1[63:32]	Interrupt TIMER_1_1588 is generated when this Timer expires. The interrupt is a periodic Interrupt.	R	0x0

24.1.4.11 1588_PTP_TMR2_LOW

Register ID: 0x07A

Address Offset: 0x1E8

Bits	Name	Description	R/W	Reset
31:0	TIMER_2[31:0]	Interrupt TIMER_2_1588 is generated when this Timer expires. The interrupt is a periodic Interrupt.	R	0x0

24.1.4.12 1588_PTP_TMR2_HI

Register ID: 0x07B

Address Offset: 0x1EC

Bits	Name	Description	R/W	Reset
31:0	TIMER_2[63:32]	Interrupt TIMER_2_1588 is generated when this Timer expires. The interrupt is a periodic Interrupt.	R	0x0

24.1.4.13 1588_PTP_TMR3_LOW

Register ID: 0x07C

Address Offset: 0x1F0

Bits	Name	Description	R/W	Reset
31:0	TIMER_3[31:0]	Interrupt TIMER_3_1588 is generated when this Timer expires. The interrupt is a periodic Interrupt.	R	0x0

24.1.4.14 1588_PTP_TMR3_HI

Register ID: 0x07D

Address Offset: 0x1F4

Bits	Name	Description	R/W	Reset
31:0	TIMER_3[63:32]	Interrupt TIMER_3_1588 is generated when this Timer expires. The interrupt is a periodic Interrupt.	R	0x0

24.1.5 1588_PTP Control in the Network Accelerator

Beside the 1588_PTP registers listed here, there are a number of 1588_PTP control functions described in the Network Accelerator chapter. Check on the following functions when setting up the 1588_PTP timer.

1. [Section 13.10.2.2, “RxControl”](#) at address 0xA1 Bits[12:11] - for the 1588 Time-Stamp format in the Prepads
2. Different Prepad Formats
 - a. [Table 13-17: Prepad Format ‘01’ \[27:192\]](#)
 - b. [Table 13-18: Prepad Format ‘10’ \[255:192\]](#)
 - c. [Table 13-19: Prepad Format ‘11’ \[255:192\]](#)
3. Interrupt [Section 13.10.2.7, “IntReg”](#) at address 0xA6 bits[31:29]
4. Interrupt Mask [Section 13.10.2.6, “IntMask”](#) at address 0xA5 bits[31:29] – these bits are set whenever the 1588_PTP Timers expire
5. Interrupt Mask [Section 13.10.2.6, “IntMask”](#) at address 0xA5 bit[28] TxTS1588 interrupt bit is set whenever the 1588_PTP Frame is transmitted out
6. [Section 13.10.9.1, “1588_PTP_SOURCE_DIV”](#) – Source Clock is divided by (this value +1)
7. [Section 13.10.9.2, “1588_PTP_SOURCE”](#) – Source Clock (core clock, half core clock, or reference clock)
8. In the Description of the Transmit Descriptor format, FBID 126 must be added as a special case to be used for Tx_1588_PTP packets.

When a packet with FBID = 126 is transmitted out of an interface, the time stamp is stored in the registers [1588_PTP_TX_LATCH_LSB](#) and [1588_PTP_TX_LATCH_MSB](#), and an interrupt is generated with TxTS1588 bit in the Interrupt vector ([IntMask\[28\]](#)) set to ‘1’. After the packet is transmitted, these descriptors are treated by the Network Accelerator Tx side like the FBID = 127 type, where the descriptor is consumed after the packet is sent out (i.e. not freed back).



Chapter 25 XLS Debugging Features

25.1 Introduction

The XLS Processor Family Control and Maintainability Resources include:

- EJTAG and Debug Testing
- System Diagnostics
- Performance Monitoring
- Instruction Statistical Sampling
- Trace Buffer
- Network Accelerator Debug Registers

This chapter describes these capabilities.

See also:

[Section 4.2.4.28 Debug register discussion.](#)

[Section 4.3.2.6, “MsgConfig1”.](#)

[Section 7.4.4, “L2 Cache Access Through JTAG”.](#)

[Section 8.10.6, “MISC FUNCTION CTL1”, CACHEABLE DISABLE \(accessing L2 configuration registers through JTAG\)](#)

25.2 EJTAG and Debug Testing

The XLS processor family supports an enhanced industry-standard IEEE-1149.1 (JTAG) boundary-scan test access port (TAP) to enable a full-featured development and debug environment for the XLS Processor Family. This interface supports a variety of JTAG (EJTAG) in-circuit emulators. The key EJTAG capabilities are listed as follows:

- Supports download / upload to EJTAG memory, Single step execution and breakpoints
- Supports NetLogic master and slave modes
- JTAG probe can initiate transactions as a master
- It responds to CPU accesses to the EJTAG memory as a slave
- Probe and CPU access are controlled using Address, Data and Control Registers
- Debug Interrupts supported
- Supports the following registers through the JTAG probe

```

IMPCODE ALL
ADDRESS EJTAGBOOT
DATA NORMALBOOT
CONTROL FASTDATA
BYPASS
  
```

- Maximum test clock frequency JTAG_TCK = 30 MHz

25.3 Obtaining Processor ID and Revision ID via the IDCODE Instruction

The processor ID and silicon ID information can be obtained by executing the IDCODE instruction. When this instruction is executed, a 32-bit value is returned that has the following format.

Table 25-1. IDCODE Data Format

Bits			
31:28	27:20	19:12	11:0
RevisionID	0x00	ProcessorID	0x449

When the IDCODE instruction is executed, the hardware retrieves the ProcessorID and RevisionID values from the fuses and places them into the associated fields of the IDCODE instruction as shown in [Table 25-1](#).

In the XLS processor, two RevisionID values are used. If the value read on bits 31:28 is 0x3, the silicon revision is B1. If the value read is 0x1, the silicon revision is A1.

[Table 25-2](#) shows the encoding of the RevisionID and ProcessorID fields for the XLS processor family.

Table 25-2. XLS ProcessorID and RevisionID Values per Product

Product	Silicon Revision	RevisionID (bits 31:28)	ProcessorID (bits 19:12)
XLS616	B1	0x3	0x40
XLS608			0x4A
XLS416			0x44
XLS408			0x4E
XLS404			0x4F
XLS408-Lite	A1	0x1	0x88
XLS404-Lite			0x8C
XLS208			0x8E
XLS204			0x8F
XLS108			0xCE
XLS104			0xCF

25.4 EJTAG Register Interface

The following section defines the EJTAG register functions.

25.4.1 TAP_INST_REG, TAP Instruction Register

The Instruction register controls selection of accessed data register(s), and controls the setting and clearing of the EJTAGBOOT indication.

Bits	Name	Description	R/W	Reset
49:39	Reserved	Reserved. (Not Used.)	R/W	0
BIST Functions through TAP Controller			R/W	0
38:35		~Sleep Bits for PLL	R/W	0
34		~Scan ATSpeed Mode Enable ^a	R/W	0
33		~Functional Test Mode ¹	R/W	0
32		~Fuse Scan Enable ¹	R/W	0
31		~Chip Reset through Tap Controller ¹	R/W	0
30		~Run Bist Through Tap Controller ¹	R/W	0
29		~Bist Reset Memory ¹	R/W	0
28		~Bist Diag Enable ¹	R/W	0
27:26		~Bist Algo Mode ¹	R/W	0
25:22		EJTAG Address Bits. 1: IMPCODE 2: Address 3: Data 4: Control 5: EJTAGALL 6: EJTAGBOOT 7: NORMALBOOT 8: FASTDATA 9: Reserved A: Address (Read Only) B: Data Read only) C: Control (Read Only) D: EJTAGALL (Read Only) All other opcodes are Reserved and should not be used.	R/W	0
21:16		Address Bits [8:3] for BIST, BSCAN and EJTAG	R/W	0
15		~clkRatio[2] ¹	R/W	0
14		~clkRatio[1] ¹	R/W	0
13		~clkRatio[0] ¹	R/W	0
12		~freezeMode ¹	R/W	0
11		~setupMode[1] ¹	R/W	0
10		~setupMode[0] ¹	R/W	0
9:7		Address Bits [2:0] for BIST, BSCAN and EJTAG	R/W	0
6		~testMode ¹	R/W	0
5		~forceDis ¹	R/W	0
4		~SelectJtagOut ¹	R/W	0

Bits	Name	Description	R/W	Reset
3		~SelectJtagIn ¹	R/W	0
2:0		Opcode. Indicates type of EJTAG operation. 0x6 (IDCODE access) 0x2 (any other EJTAG access)	R/W	0

a. The ~ symbol indicates the values shifted into these bit fields are the inverted version of the values observed at the TAP ports.

Bit Fields Relevant for EJTAG Controller Access:

- Address Bits [8:0] = 0xFB (For any EJTAG access)

25.4.2 IMPCODE

The Implementation register is a 32-bit read-only register that identifies features implemented in this EJTAG compliant processor, mainly those accessible from the TAP.

31:29	28	27:25	24	23	22:21	20:17	16	15	14	13:1	0
EJTAGver	R4k	0	DINT sup	0	ASID Size	0	MIPS 16	0	No DMA	0	MIPS 32/64

Fields		Description	R/W	Reset
Bits	Name			
31:29	EJTAGver	Indicates the EJTAG version: 0: Version 1 and 2.0 1: Version 2.5 2: Version 2.6 3-7: Reserved	RO	2
28	R4k	Indicates R4k privileged environment: 0: R4k privileged environment 1: R3k privileged environment	RO	0
27:25	0	Ignored on writes; return zeros on reads.	RO	0
24	DINTsup	Indicates support for DINT signal from probe: 0: DINT signal from the probe is not supported by this processor 1: Probe can use DINT signal to make debug interrupt on this processor	RO	1
23	0	Ignored on writes; return zeros on reads.	RO	0
22:21	ASIDsize	Indicates size of the ASID field: 0: No ASID in implementation 1: 6-bit ASID 2: 8-bit ASID 3: Reserved	RO	0
20:17	0	Ignored on writes; return zeros on reads.	RO	0
16	MIPS16	Indicates MIPS16™ ASE support in the processor: 0: No MIPS16 support 1: MIPS16 is supported	RO	0
15	0	Ignored on writes; return zeros on reads.	RO	0
14	NoDMA	Indicates no EJTAG DMA support: 0: Reserved 1: No EJTAG DMA support	RO	0
13:1	0	Ignored on writes; return zeros on reads.	RO	0
0	MIPS32/64	Indicates 32-bit or 64-bit processor: 0: 32-bit processor 1: 64-bit processor See the R4k bit above for indication of privileged environment.	RO	1

25.4.3 Address Register

The read-only Address register provides the address for a processor access. Functionality varies depending on processor rev number, as defined below.

Functionality in Address Register for Slave Mode

77:76		75:73		72:70		69:67		66:35		34:0								
CPU ThreadID		CPU ID		Address Space		Command		Byte Enables		Address								
Fields		Description									R/W	Reset						
77:76	CPU ThreadID	CPU ThreadID. This bit should be set to 0.									R/W	0						
75:73	CPU ID	CPU ID. This bit should be set to 0.									R/W	0						
72:70	Address Space	This field is initially set to 0 by software. The internal Bridge logic then modifies this value based on which PCI BAR the address maps to: 000: Register space 001: Memory space 010: I/O space 011: Configuration space 100: PCIe configuration space 101 - 111: Reserved									R/W	0						
69:67	Command	This field indicates the type of command being executed: 000: Idle 001: Write 010: Read 011: Read exclusive 100: Upgrade 101: Invalidate 110: Reserved 111: Write error									R/W	0						
66:35	Byte Enables	Cache line byte enables. This 32-bit field represents each byte of the 32-byte cache line. Bit 35 of this field is the byte enable for byte 0, while bit 66 is the byte enable for byte 31.									R/W	0						
34:0	Address	This field stores the 35-bit cache-line aligned physical address.									R/W	0						

Functionality in Address Register for Master Mode

77		76		75:73	72:70	69:67	66:35	34:0		
IodML2Alloc		IodMCoherent		CPU ID	Address Space	Command	Byte Enables	Address		
Fields		Description							R/W	Reset
77	IodML2Alloc	This bit determines whether or not the cache-line address and corresponding data will be cached in the L2 cache. 1: cache data 0: do not cache data							R/W	0
76	IodMCoherent	This bit determines whether the address is to be cache-coherent. 1: Address is cache-coherent 0: Address is not cache-coherent							R/W	0
75:73	CPU ID	CPU ID							R/W	0
72:70	Address Space	This field is initially set to 0 by software. The internal Bridge logic then modifies this value based on which PCI BAR the address maps to: 000: Register space 001: Memory space 010: I/O space 011: Configuration space 100: PCIe configuration space 101 - 111: Reserved							R/W	0
69:67	Command	This field indicates the type of command being executed: 000: Idle 001: Write 010: Read 011: Read exclusive 100: Upgrade 101: Invalidate 110: Reserved 111: Write error							R/W	0
66:35	Byte Enables	Cache line byte enables. This 32-bit field represents each byte of the 32-byte cache line. Bit 35 of this field is the byte enable for byte 0, while bit 66 is the byte enable for byte 31.							R/W	0
34:0	Address	This field stores the 35-bit cache-line aligned physical address.							R/W	0

25.4.4 Data Register

The read/write Data register is used for opcode and data transfers during processor accesses.

256		255:0				
Response		Data				
Fields		Description			R/W	Reset
256	Response	Response. 0: Normal 1: Error The Response bit is valid only for Read transactions.			RO	0
255:0	Data	Data			R/W	0

25.4.5 TAP Control Register

The 32-bit EJTAG TAP Control Register (ECR) handles processor reset and soft reset indication, Debug Mode indication, access start, finish, and size and read/write indication. The ECR also:

- controls debug vector location and indication of serviced processor accesses,
- allows a debug interrupt request,
- indicates processor low-power mode, and
- allows implementation-dependent processor and peripheral resets.

135	134	133	132:131	130	129	128	127:96	95:64	63:32	31:0	
Fields		Description								R/W	Reset
135	OCP Master/ Slave Mode	OCP Master/Slave Mode 0: Probe cannot initiate a transaction. Only CPU transactions are permitted. 1: Probe can initiate a transaction. (CPU transactions are not permitted)								R/W	0
134	EJTAG Probe Access	EJTAG Probe Access 0: No pending probe access 1: Pending probe access The EJTAG controller must clear this bit after completing the probe access on the IOBIU.								R/W	0
133	Processor Reset Occurred	Processor Reset Occurred 0 - No Reset Occurred 1 - Reset Occurred This bit must be cleared to acknowledge the reset has occurred. The EJTAG Control Register is not updated if this bit is 1.								R/W	0
132:131	Reserved	Reserved								-	0

Fields		Description	R/W	Reset
Bits	Name			
130	Processor Read/Write Status	Processor Read/Write Status 0: Processor Read 1: Processor Write	RO	0
129	Processor Access Pending	Processor Access Pending 0: No Pending processor access 1: Pending processor access The probe control software must clear this bit after completing the processor access. Write of 1 is ignored.	R/W	0
128	Reserved	Reserved	-	0
127:96	EJTAG Probe Enable	EJTAG Probe Enable 127:124: Reserved 123:120: Reserved 121:116: Reserved 115:112: Reserved 111:108: CPU3 (Threads 0-3) 107:103: CPU2 (Threads 0-3) 103:100: CPU1 (Threads 0-3) 99:96: CPU0 (Threads 0-3)	R/W	0
95:64	EJTAG Debug Boot	EJTAG Debug Boot 95:92: Reserved 91:88: Reserved 87:84: Reserved 83:80: Reserved 79:76: CPU3 (Threads 0-3) 75:72: CPU2 (Threads 0-3) 71:68: CPU1 (Threads 0-3) 67:64: CPU0 (Threads 0-3)	R/W	0
63:32	EJTAG Debug Interrupt	EJTAG Debug Interrupt 63:60: Reserved 59:56: Reserved 55:52: Reserved 51:48: Reserved 47:44: CPU3 (Threads 0-3) 43:40: CPU2 (Threads 0-3) 39:36 : CPU1 (Threads 0-3) 35:32: CPU0 (Threads 0-3)	R/W	0
31:0	CPU Debug Mode Status	CPU Debug Mode Status 31:38: Reserved 27:24: Reserved 23:20: Reserved 19:16: Reserved 15:12: CPU3 (Threads 0-3) 11:8: CPU2 (Threads 0-3) 7 :4 : CPU1 (Threads 0-3) 3:0: CPU0 (Threads 0-3)	RO	0

25.4.6 Debug Control Register (DCR)

The Debug Control Register (DCR) controls and provides information about debug issues.

38:32		31:0		
Rsvd		EJTAG Probe Enable		
Bits	Name	Description	R/W	Reset
38:32	Reserved	Reserved		0
31:0	EJTAG Probe Enable	EJTAG Probe Enable		0

Notes

1. EJTAG Memory/Reg Access

Standard EJTAG memory range

- Bridge maintains EJTAG BAR AT (0xFFFF_FFFF_FF20_0000 - 0xFFFF_FFFF_FF3F_FFFF)
- CPU will place PROBE_MODE and CPU Thread ID bits in the "way" field of the system ring when initiating a EJTAG request.

way[2] = Probe mode access
way[1:0] = CPU Thread ID

- Bridge will route request to JTAG space when:

EJTAG_MEM_ACCESS => (ADDRESS == EJTAG_BAR) & uncacheable & way[2]

Bridge will send address to JTAG controller in following form:

EJTAG_ADDRESS[21:0] = ADDRESS[21:0]
EJTAG_ADDRESS[23:22] = way[1:0] (thread id)

- JTAG Controller

```
If (EJTAG_ADDRESS[21:20] == 2)
    ==> Initiate JTAG request
else
    ==> Return DCR register (DCR register defined in EJTAG Spec)
```

2. EJTAG Boot

- EJTAG controller will assert debug_interrupt to all requested CPUs during reset
- CPU will immediately take a debug interrupt right after reset.

25.5 System Diagnostics

The 32-bit address error registers summarized in [Table 25-3](#) control and monitor addressing errors and the available devices throughout the XLS Processor that can cause them. They are in the System Bridge Controller region of the MDI address space as shown in Chapter 8. See [Table 8-2: System Bridge Controller Register Summary](#) for context.

There are six detected address errors:

- Reserved Space - There is no matching BAR space for the address
- Multiple Hits - Address matches more than one BAR space
- Illegal Cached Access - Cached access to an uncacheable space. PCI-X is an example.
- Disabled Device Access - Attempted access to a disabled device
- L1 Tag Error - Uncorrectable bit error in tag address in L1 cache
- L2 Tag Error - Uncorrectable bit error in tag address in L2 cache

The errors are reported in the AERR0_LOGn or AERR1_LOGn logging registers for the devices selected in their AERR_DEVID fields. The first four error types are detected by the bridge when it is selected in the AERR_STAT field. The last two error types are logged when the corresponding cache is selected in the AERR_STAT field.

Register IDs are in decimal; register offset addresses are on four-byte boundaries.

Table 25-3. Address Error Registers in System Bridge Controller

Name	Reg ID	Description
DEVICE_MASK	0x25	This 32-bit Read/Write register is a mask representing in real-time the active devices in an XLS Processor Family member. Some bits are determined by hard-wired connections while others are maintained by software. Reset values depends upon the specific XLS Family Member.
AERR0_LOG1	0x26	Address Error Zero interrupt log register number one.
AERR0_LOG2	0x27	Address Error Zero interrupt log register number two. Lower-order offending address bits.
AERR0_LOG3	0x28	Address Error Zero interrupt log register number three. Higher-order offending address bits.
AERR0_DEVSTAT	0x29	Address Error Zero interrupt Device Status register. This Read/Write register identifies the source device which caused the AERR0 interrupt. Devices include DMA, CDE, PCIe, SAE, GMAC, USB, L2, CPUn.
AERR1_LOG1	0x2A	Address Error One interrupt log register number one.
AERR1_LOG2	0x2B	Address Error One interrupt log register number two. Lower-order offending address bits.
AERR1_LOG3	0x2C	Address Error One interrupt log register number three. Higher-order offending address bits.
AERR1_DEVSTAT	0x2D	Address Error One interrupt Device Status register. This Read/Write register identifies the source device which caused the AERR1 interrupt.
AERR0_EN	0x2E	Enables the System Bridge Controller to detect and report AERR0 interrupts.
AERR0_UPG	0x2F	This Read/Write register can be configured to allow certain AERR0 interrupts to be Upgraded to AERR1. The corresponding AERR0_EN bits must also be enabled. Correspondingly, the PIC IRT must also be programmed to trigger interrupts to a CPU.
AERR0_CLEAR	0x30	This Read/Write register is used to Clear AERR0 interrupts. Write any value to bit 0 to clear.
AERR1_CLEAR	0x31	This Read/Write register is used to Clear AERR1 interrupts. Write any value to bit 0 to clear.
SBE_COUNTS	0x32	This Read Only register Counts Single-Bit Errors that occur in the system. If any of these values become non-zero, and the corresponding BITERR_INT_EN bit is set, then an interrupt will be generated.
DBE_COUNTS	0x33	This Read Only register Counts Double-Bit Errors that occur in the system. If any of these values become non-zero, and the corresponding BITERR_INT_EN bit is set, then an interrupt will be generated.
BITERR_INT_EN	0x34	This Read/Write register Enables Bit-Error Interrupts.

25.6 Performance Monitoring

25.6.1 Performance Monitoring of Processor Events

Two pairs of performance counter registers provide a means for software to count processor events globally or per Thread. Register pair PerfCntrCtl0 and PerfCntr0 and register pair PerfCntrCtl1 and PerfCntr1 provide this facility. See [Section 4.2.4.30, “PerfCntrCtl0”](#), [Section 4.2.4.31, “PerfCntr0”](#), [Section 4.2.4.32, “PerfCntrCtl1”](#) and [Section 4.2.4.33, “PerfCntr1”](#).

The 64 types of events that can be counted are shown in [Table 4-1](#). Event categories include:

- Instructions fetched and retired, branch instructions
- Instruction and Data Cache Unit statistics
- Instruction and Data TLB statistics
- Instruction Fetch Unit statistics
- Instruction Execution Unit statistics
- Load/store Unit statistics
- Cycle Count

The E bit (bit 0) of the PerfCntrCtl[0,1] registers enable event counting when the EXL bit in the Status register is 1 and the ERL bit in the Status register is 0. Counting is never enabled when the ERL bit in the Status register or the DM bit in the Debug register is 1. The K, S, and U bits (bits 1, 2 and 3) enable counting in kernel, supervisor and user modes, respectively. The K bit enables event counting only when the EXL and ERL bits in the Status register are 0.

The IE bit (bit 4) enables an interrupt request when the corresponding counter overflows (when counter bit 31 goes to 1). Note that IE simply enables the interrupt request. The actual interrupt is still gated by the normal interrupt masks and enable in the Status register.

The EVENT field selects the event to be counted by the corresponding Counter Register is shown in [Table 4-1](#). If the G bit (bit 13) is set, events from all Threads are counted, otherwise only events from the Thread indicated by the TID field are counted.

25.6.2 Performance Monitoring of Memory Transactions

The System Bridge Controller provides two register pairs to facilitate counting memory transactions through the System Bridge Controller.

The [EVENT_CNT_CNTRL1](#) register is used to select types of events that will cause its companion register [EVENT_CNTR1](#) to increment. Any number of event types may be enabled.

The [EVENT_CNTR1](#) register will increment every time an enabled event is triggered. Bit-fields in [EVENT_CNT_CNTRL1](#) labeled “HIT” enable accepted memory requests to be counted; bit-fields labeled “RETRY” enable the counting of events where no agent was able to service the request, therefore causing a retry to be sent out.

The [EVENT_CNT_CNTRL2](#) register and its companion register [EVENT_CNTR2](#) operate similarly.

Sources of HIT and RETRY events include:

- DRAM Channel A, B, C, D
- Bridge Register
- PCIe
- Peripherals Interface
- Serial I/O
- Other I/O

25.7 Instruction Statistical Sampling

In addition to the performance counters, each processor core contains an instruction statistical sampling feature, providing additional performance monitoring. This feature can be used to profile the behavior of a section of code over a period of time.

When activated, the sampling hardware “tags” a randomly-selected executing instruction, and records some of the events that occur to that instruction. When the instruction retires, an interrupt is taken and the driver software records the events in a table indexed by the address of the instruction. The collected data can be used to analyze the behavior of a given instruction over time (e.g. the instruction at address 0x100 takes instruction-cache misses 5% of the time), or to compare the behavior of different sections of code (e.g. the branch at address 0x200 is mispredicted 2% of the time, but the branch at address 0x300 is mispredicted 5% of the time).

The feature is activated by setting the ENABLE bit (bit 0) of the [ICU_SAMPLING_SETUP](#) register. This register allows the user to identify the Thread from which an instruction should be picked, whether instructions should be picked from all threads, how frequently instructions should be picked, and whether an interrupt should be taken when the instruction retires. If the PC_MODE bit (bit 13) of this register is set, then the user must first specify the address of the instruction to be picked by programming the [ICU_SAMPLING_PC](#) and [ICU_SAMPLING_PC_UPPER](#) registers. Otherwise, these two registers may be read once the sampling is complete to identify the address of the instruction that was picked by the hardware.

Once armed, the hardware waits for a random period of time determined by the [ICU_SAMPLING_LFSR](#) register. When that period of time has expired, the [ICU_SAMPLING_TIMER](#) register is triggered and counts for 1024 clock cycles. During these cycles, the first instruction in the pipeline that meets the user-programmed criteria is selected for sampling.

As a time-out feature, if no suitable instruction is found during these 1024 cycles, or if the chosen instruction does not retire or take an exception during these 1024 cycles, then the relevant state is reset and the instruction selection and sampling process is attempted again.

When the selected instruction retires, the event history of that instruction is recorded in the [PRF_SMP_EVENT](#) register. If enabled, an interrupt will be taken to notify the software that a sample has completed. Otherwise, the software may periodically poll ENABLE bit (bit 0) of the [ICU_SAMPLING_SETUP](#) register. When it is cleared, the sample is done.

Note the “one-shot” nature of the sampling mechanism; once a sample has completed, the feature automatically disarms itself to prevent the captured data from being overwritten. It is the responsibility of software to read the data from the [PRF_SMP_EVENT](#) register, clear all the bits, and re-arm the sampling mechanism to capture the next sample.

25.8 Watchpoint Debugging

The WatchLo and WatchHi registers together provide the interface to a watchpoint debug facility which initiates a watch exception if an instruction or data access matches the address specified in the registers. See [Section 4.2.4.24, “WatchLo”](#) and [Section 4.2.4.25, “WatchHi”](#). As such, they duplicate some functions of the EJTAG debug solution (see [Section 25.2, “EJTAG and Debug Testing”](#)). Watch exceptions are only taken if the EXL and ERL bits are zero in the Status register. If either bit is set, the WP bit is set in the Cause register, and the watch exception is deferred until both the EXL and ERL bits are zero.

The XLS Processor provides one pair of WatchLo and WatchHi registers. Software may determine that WatchLo and WatchHi registers are implemented via the WR bit of the Config1 register. See the discussion of the M bit in the [WatchHi](#) register description.

The [WatchLo](#) register specifies the base virtual address and the type of reference (instruction fetch, load, store) to match. All three reference types are supported.¹

Data watch exceptions will be taken by the XLS Processor for PREFETCH instructions, as long as PREFETCH instructions are not disabled in the LSU configuration register. Data watch exceptions will be taken for the fetch and lock CACHE instructions, but not for other types.

The [WatchHi](#) register contains information that qualifies the virtual address specified in the [WatchLo](#) register: an ASID, a G(lobal) bit, and an optional address mask. If the G bit is one, any virtual address reference that matches the specified address will cause a watch exception. If the G bit is zero, only those virtual address references for which the ASID value in the [WatchHi](#) register matches the ASID value in the [EntryHi](#) register cause a watch exception. The optional mask field provides address masking to qualify the address specified in [WatchLo](#).

25.9 Trace Buffer

The XLS Processor Family includes a Trace Buffer (TB) to store address traces from the system Distributed Interconnect. The TB can store up to 256 address traces. The XLS processor provides a programmable trigger mechanism to start and stop trace collection based on a filter. The filter is a set of AND'ed match criteria including the address, the transaction type, snoop result, interrupt raised, etc. The trace buffer can operate in two modes:

1. Start trace on filter match and continue collecting all subsequent transactions until a specified number of transactions is reached, or
2. Collect only transactions that match the filter until a specified number of transactions is reached.

Both the CPU and JTAG can configure the Trace Buffer and read the captured information. The trace buffer can be configured to generate an interrupt when the trace is complete. The Trace Buffer can also be used as a scratch buffer in a FIFO manner.

The Trace Buffer register space resides in the XLS IO base: XLS_IO_DEV_TB, see [Table 25-4](#). See Chapter 8 for a complete listing of Peripheral and I/O Devices in the Configuration Region.

Table 25-4. Trace Buffer Register Space

Peripheral	Device Name	Sub-Region Offset	Sub-Region Offset Address at Reset
Trace Buffer	XLS_IO_DEV_TB	0x1 C000	0x00 1EF1 C000

The Trace Buffer has 256 entries. The trace size (number of requests to log before stopping) cannot be larger than 255.

The Trace Buffer logs only request information, which is further specified below in the register definitions.

After reset, the Trace Buffer is initialized for trace collection. It continuously writes each new system ring request into the Trace Buffer until it detects a match.

A match condition is specified in the Match registers (defined below), and the match values are enabled in the Control (CTL) register (see register definition below).

Once a match is made, it counts each new request pushed into the Trace Buffer, until the trace-size specified in the Control register is reached. At that point, it issues an interrupt. It will no longer collect request information until the user writes the INIT register.

The Trace Buffer has two modes of request collection as specified in the Mode field in the Control Register. If set, the Trace Buffer will only collect requests that match that match criteria. It will continue to collect these matching requests until the trace size is reached. If cleared, the Trace Buffer will collect all requests it sees after the match, until it reaches the trace size.

-
1. Software may determine which enables are supported by a particular Watch register pair by setting all three enable bits and reading them back to see which ones were actually set.

The user reads the trace information by writing the read command into the ACCESS register, then reading the information popped from the Trace Buffer from the Read Data Registers (RDRs).

The user can also push data into Trace Buffer by writing data into the Write Data Registers (WDRs) and writing the write command into the ACCESS register.

25.9.1

Trace Buffer Operation

The Trace Buffer records transactions on the system bus. Each transaction entry is 105 bits long and hence occupies four 32-bit registers. The fields of the entry are given in the [Section 25.10, “Trace Buffer Registers”](#).

The number of transactions n must be specified to a maximum of 255. Upon recording n entries, the TB raises an interrupt. The interrupt handler is then responsible for reading the recorded entries from the TB and writing them to a buffer that is accessible to the user through the file system interface. After raising the interrupt, the TB stops recording and must be re-initialized to resume collecting transactions.

The TB interface includes these registers:

- Two request-match registers

These registers are used to specify the values of the transaction parameters we are interested in.

- Two request address registers

These are used to specify the address values of the transaction one is interested in. Only the 35-bit address of the cache-line must be specified. While the first 32 bits goes in the first request address register, the remaining 3 bits are specified in the lower bits of the second register

- Control register

Besides specifying the values in the above two registers one also has to indicate the fields (of transactions) we are interested in by setting the corresponding flags. The number of transactions to collect is also specified in the control register.

- Init register

As indicated above, the TB stops collecting traces once it is done collecting the specified number of transactions. The TB can be triggered again by setting this register.

- Status register

This is seldom used by users and is typically used by those who implement the low-level interfaces.

25.9.2

Programming the Trace Buffer

1. The Trace Buffer is enabled at reset and begins collecting every transaction immediately. The user may want to disable it so as to start from scratch. To accomplish this, disable the Trace Buffer by setting the DISABLE_TRACE_BUFFER bit in the Control Register ([Section 25.10.5, “Control Register”](#)).
2. Once the Trace Buffer is disabled, the user may want to enable the Trace Buffer interrupt in the PIC and identify a CPU thread to service it.
3. With the Trace Buffer disabled, the user can configure the match criteria in Request Match #1, Request Match #2, Request Address Low Match, and Request Address High Match and the Control Register. The last act should be to clear the DISABLE_TRACE_BUFFER bit.
4. Once the DISABLE_TRACE_BUFFER bit is cleared, the user must initialize the trace buffer by setting the Init bit in the Initialization Register ([Section 25.10.6, “Initialization](#)

- [Register](#)). Please note that to initialize the Trace Buffer, it must see the INIT bit go from low to high, so the user must write a zero to it, then a one to ensure that this happens.
- Once the match criteria and REQCNT parameters are met, then the Trace Buffer will disable itself. If enabled, an interrupt will then be generated. At this point, the user may inspect the Trace Buffer contents by using the read command.

25.9.3

Extracting Trace Buffer Entries

This section describes how to calculate the number of records to pop from the Trace Buffer to get to a desired entry. The Status Register ([Section 25.10.10, “Status Register”](#)) has four important fields for this calculation, as follows:

- MPTR - the Trace Buffer entry with the first match
- RDPTR - the Trace Buffer entry which will be read on a read command
- WRPTR - the Trace Buffer entry which will be written into on a write command
- FULL - if set, indicates that the Trace Buffer is full of valid transactions

In addition, the programmer will need to know the REQCNT from the Control Register. ([Section 25.10.5, “Control Register”](#)).

If the Trace Buffer FULL field is clear (i.e. not full), then the MPTR field value matches the number of entries to read before getting to the first matching entry.

If the Trace Buffer FULL field is set (i.e. full), then the programmer needs to calculate the number of reads required to get to the first matching entry according to the following pseudocode:

```
if (WRPTR >= RDPTR)
    number_of_valid_entries = WRPTR - RDPTR;
else
    number_of_valid_entries = 256 - (RDPTR - WRPTR);
number_of_reads = number_of_valid_entries - REQCNT;
```

25.10

Trace Buffer Registers

Note: Unless otherwise stated, reset value is 0.

Note: Undefined fields of configuration registers are *Reserved*. Their reset value is 0 and they are read-only.

25.10.1

Request Match Register 1

Register Address Offset: 0x00

Bits	Name	Description	R/W	Reset
31:25	Reserved	Reserved		0
24:24	RCACHE	Cacheable The transaction is cacheable, so all caches will snoop the request.		0
23:20	SSTAT	Snoop error status An error was detected on the transaction. The bit position identifies where the error was found: [20]: Bridge address error, [21]: L1 tag error: [22]: L2 tag correctable error: [23]: L2 tag uncorrectable error:		0
19	Reserved	Reserved		0

Bits	Name	Description	R/W	Reset
18:12	TRHREADID	Transaction ID Every transaction is given a transaction ID by the agent (CPU, L2, etc.) that issued the transaction. The transaction ID is unique to that node ID, but not across all transactions for all nodes.		0
11:9	Reserved	Reserved		0
8:4	RCONNID (NODE_ID)	Node ID A number identifying the agent that issued the request 0 = CPU 0 1 = CPU 1 2 = CPU 2 3 = CPU 3 8 = L2 9 = L2 19 = GMAC Quad 0 20 = SEC 21 = Serial I/Os and peripherals 24 = DMA 25 = CDE (XLS6xx, XLS4xx-Lite and XLS4xx devices only) 26 = PCIe 27 = USB 28 = GMAC Quad 1 (XLS6xx, XLS4xx-Lite and XLS4xx devices only)		0
3	Reserved	Reserved		0
2:0	RCMD	Command. The type of transaction 1 = write command 2 = read command 3 = Read-exclusive command. Any copies of the cache line will be invalidated, and the issue of the read will obtain exclusive ownership of the cache line 4 = Upgrade. The issuer is upgrading the cache line state from Shared to Modified. All other copies are invalidated and the rest are undefined		0

25.10.2 Request Match Register #2

Register Address Offset: 0x01

Bits	Name	Description	R/W	Reset
31:12	Reserved	Reserved		0
11:8	RSLT	The state of the transaction during the snoop: [8]: CPUSHR: Shared state in the CPU [9]: CPUMOD: Modified state in the CPU [10]: L2SHR: Shared state in L2 [11]: L2MOD: Modified state in L2		0
7:0		Reserved		0

25.10.3 Request Address Low Match Register

Register Address Offset: 0x02

Bits	Name	Description	R/W	Reset
31:0	ADDRESS_LOW	Physical address [36:5]		0

25.10.4 Request Address High Match Register

Register Address Offset: 0x03

Bits	Name	Description	R/W	Reset
31:3	Reserved	Reserved		
2:0	ADDRESS_HIGH	Physical address [39:37]		0

25.10.5 Control Register

Register Address Offset: 0x04

Bits	Name	Description	R/W	Reset
31:25	Reserved	Reserved		0
24	DISABLE_TRACE_BUFFER	Stops collection		0
23:16	REQCNT	Number of request cycles to capture after match.		0
15:9	Reserved	Reserved		0
8	COLLECTION_MODE	Collection Mode: 0: All 1: Matching		0
7	MADDRF	Match ADDR field		0
6	MRSLT	Match RSLT field		0
5	RESERVED	Reserved		0
4	MRCACHE	Match RCACHE field		0
3	MSTAT	Match STAT field		0
2	MRTHREADED	Match RTHREADED field		0
1	MRCCONNID	Match RCONNID field		0
0	MRCMD	Match RCMD field		0

25.10.6 Initialization Register

Register Address Offset: 0x05

Bits	Name	Description	R/W	Reset
31:1	Reserved	Reserved		0
0	INIT	Re-initialize trace buffer after trace collection		0

25.10.7 Access Register

Register Address Offset: 0x06

Bits	Name	Description	R/W	Reset
31:1	Reserved	Reserved		0
0	COMMAND	If 0, pop Trace Buffer entry and capture it in the Read Data Registers. If 1, push contents of Write Data Registers into the Trace Buffer.		0

25.10.8 Read Data Registers (RDR) 0 – 3

Register Address Offset: 0x07, 0x08, 0x09, 0x0A

These four RDR registers are combined to access read data as follows:

Read Data[105:0] == {RDR3, RDR2, RDR1, RDR0}

Bits	Name	Description	R/W	Reset
105:103	RCMD	Command read, write, etc. (3 bits)		0
102:100	RWAY	(3 bits) The L1 way allocated for the transaction		0
99	RCACHE	Cacheable bit Used to tell caches to snoop on transaction. (1 bit)		0
98	RL2ALLOC	L2 allocate Indicates whether to allow cache-line to be allocated into L2. (1 bit)		0
97	RSIZE	If 1; full 32-byte transaction. If 0, then RBYTEEN identifies bytes being accessed. (1 bit)		0
96:65	RBYTEEN	Byte mask of bytes being accessed. (32 bits)		0
64	RTAGPAR	(1 bit) Tag parity of the L1 snoop tag		0
63:29	RADDR	35-bit cache-line physical address. (35 bits)		0
28:21	RTHREADID	Transaction ID of the node initiating the transaction. (8 bits)		0
20:16	RCONNID	Node ID of initiator (i.e. CPU0 or CPU7, or L2-bank0, or security block). (5 bits)		0
15:12	SRSLT	Snoop result status - [12]: CPUSHR [13]: CPUMOD [14]: L2SHR [15]: L2MODE		0
11:9	SRETRY	Who retried the transaction - [9]: CPU [10]: L2 [11]: bridge		0
8:5	SSTAT	Snoop error status [5]: bridge [6]: L1tag [7]: L2ctag [8]: I2uctag		0
4:0	RSPHITID	This field is not operational -- Use the initiator ID (RTHREADID or RCONNID) and address to provide useful information about the transaction.		0

25.10.9 Write Data Register (WDR) 0 – 3

Register Address Offset: 0x0B, 0x0C, 0x0D, 0x0E

Write Data[104:0] == {WDR3, WDR2, WDR1, WDR0}. You can write data into the trace buffer to use it as a temporary FIFO.

Bits	Name	Description	R/W	Reset
104:0	WDATA	Write data		0

25.10.10 Status Register

Register Address Offset: 0x0F

Bits	Name	Description		Reset
31:24	MPTR	Trace Buffer pointer of first match entry in Trace Buffer		0
23:16	RDPTR	Current Trace Buffer read pointer		0
15:8	WRPTR	Current Trace Buffer write pointer		0
7:4	STATE	Current Trace Buffer collection state One-shot vector: 4: COLLECT — collecting trace information before a match 5: MATCHED — collecting trace information after a match 6: DONE — done collecting trace 7: DISABLED — disabled from collecting trace		0
3:2	Reserved	Reserved		0
1	FULL	Trace Buffer is full		0
0	EMPTY	Trace Buffer is empty		0

25.10.11 Trace Buffer Driver Example

In this example, based on a Linux or UNIX device driver environment, the user interface to the TB is provided through the character device “/dev/tracebuffer”. The standard Linux/UNIX ioctl() interface allows the user to program the device, and the read() system call is used to read the contents of the TB.

Hence, the programmer must create the character device “tracebuffer” through the command

```
mknod /dev/tracebuffer c 252 0
```

The magic number 252 corresponds to the major device number.

Below is pseudocode for a program to control and read the TB.

```
int main() {
    // open the tracebuffer device
    fd = open("/dev/tracebuffer", "r");

    for (number of snapshots) {
        trace();

        // read the contents
        while (read(fd, ...) > 0) {
            write the contents to output or a file;
        }
    }

    trace();
    // the ioctl interface is then used to program the tracebuffer.
}
```

```

    request_match_register_one = (val1 | val2 | val3);
    write to request match register one;

    request_match_register_two = (val1 | val2 | val3);
    write to request match register two;

    set the fields of the control register;
    write to control register;

    initialize the trace buffer by writing to the INIT register;
}

```

25.11 Network Accelerator Debug Registers

The Network Accelerators contain a number of debugging registers, such as performance counters and word count registers for the various FIFOs, as described below.

Operation of the Networking Accelerators is governed by registers in the sub-region address space as shown in the system memory map in [Figure 13-4](#). The offsets for the eight sub-region address spaces are given in [Table 13-1](#). All individual register offsets apply to the corresponding sub-region for the interface controller which shares the register space.

25.11.1 DebugCounter0_A

Address Offset 0x226 in GMAC memory subregion

Debug counter for Debug Event 0A.

Bits	Name	Description	R/W	Reset
31:0	DebugCounter	The debug counter is similar to performance counters. This counts the number of times the DebugEvent0_A is detected.	R/W	0

25.11.2 DebugCounter0_B

Address Offset 0x227 in GMAC memory subregion

Debug counter for Debug Event 0B.

#	Name	Description	R/W	Reset
31:0	DebugCounter	Debug counter, similar to performance counters. This counts the number of times DebugEvent0_B is detected.	R/W	0

25.11.3 DebugEvent0_A

Address Offset 0x224 in GMAC memory subregion

Debug Event Indicator A

Bits	Name	Description	R/W	Reset
31:6	Reserved	Reserved	R	0
5:0	DebugEvent0_A	Debug Event is one of the following: GMACS: 31: AsyncFIFO_OverFlow 0 30: AsyncFIFO_OverFlow 1 29: AsyncFIFO_OverFlow 2 28: AsyncFIFO_OverFlow 3 27: Discard 0 26: Discard 1 25: Discard 2 24: Discard 3 23: IngressSOP 22: IngressEOP 21: SOP out of Async FIFO 20: EOP out of Async FIFO 19: Async FIFO Getting Full 18: Async FIFO Getting Empty 17: RxFifo > WM_High 16: RxFifo < WM_Low 15: IoDwRegFrInPush, 14: IoDwP2PFrInPush, 13: IoDwRegFrInPush1, 12: Reserved, 11: FrOutFIFOWE, 10: Class0DwFIFOWE, 9: Class1DwFIFOWE, 8: Class2DwFIFOWE, 7: Class3DwFIFOWE, 6: IoDwTx0Push, 5: IoDwTx1Push, 4: IoDwTx2Push, 3: IoDwTx3Push, 2: RegFrInDmaChannelDV, 1: DmaRegFrInChannelDV, 0: RegFrInDmaChannelDV1	R/W	0

Bits	Name	Description	R/W	Reset
		XGMII debug events: 31:AsyncFIFO_OverFlow 30:DiscardPkt 29:IG_ABORT 28:EOP_ERR 27:DIP4_ERR 26:SOP_ERR 25:SOP 24:EOP 23:IngressSOP 22:IngressEOP 21: AsyncFifoGettingFull 20: AsyncFifoGettingEmpty 19: Rx0EarlyFull 18: Rx1EarlyFull 17: Rx2EarlyFull 16: Rx3EarlyFull 15: IoDwRegFrInPush, 14: IoDwP2PFrInPush, 13: 0, 12: 0, 11: FrOutFIFOWE, 10: Class0DwFIFOWE, 9: Class1DwFIFOWE, 8: Class2DwFIFOWE, 7: Class3DwFIFOWE, 6: IoDwTx0Push, 5: IoDwTx1Push, 4: IoDwTx2Push, 3: IoDwTx3Push, 2: RegFrInDmaChannelDV, 1: DmaRegFrInChannelDV,	R/W	0

25.11.4 DebugEvent0_B

Address Offset: 0x225 for GMAC

Debug Event Indicator 0B

Bits	Name	Description	R/W	Reset
31:0	DebugEvent0_B	Debug Events are same as DebugEvent0_A	R/W	0

25.11.5 DebugCounter1_A

Address Offset: 0x22E for GMAC

Debug Counter 1A

Bits	Name	Description	R/W	Reset
31:0	DebugCounter	Debug counter, similar to performance counters. This counts the number of times the DebugEvent1_A is detected.	R/W	0

25.11.6 DebugCounter1_B

Address Offset: 0x22F for GMAC

Debug Counter 1B

Bits	Name	Description	R/W	Reset
31:0	DebugCounter	Debug counter, similar to performance counters. This counts the number of times the DebugEvent1_B is detected.	R/W	0

25.11.7 DebugEvent1_A

Address Offset: 0x22C for GMAC

Debug Event Indicator 1A.

Bits	Name	Description	R/W	Reset
31:0	DebugEvent1_A	Debug Event is one of the following: GMACS: Debug Event is one of the following: 31: p2pDescIn 30: p2pDescFree 29: p2pAddressSend 28: p2pDataReceive 27: EOP3 26: EOP2 25: EOP1 24: EOP0 23: TxAbort_NoRetry_3 22: TxAbort_NoRetry_2 21: TxAbort_NoRetry_1 20: TxAbort_NoRetry_0 19: Retry_3 18: Retry_2 17: Retry_1 16: Retry_0 15: DmaRegFrInChannelDV1 14 P2PFRInDmaChannelDV, 13: DmaP2PFRInChannelDV, 12: Reserved 11: Reserved 10: FrOutDmaChannelDV, 9: DmaFrOutChannelDV, 8: SpClass0DmaChannelDV 7: DmaSpClass0ChannelDV 6: SpClass1DmaChannelDV 5: DmaSpClass1ChannelDV 4: SpClass2DmaChannelDV 3: DmaSpClass2ChannelDV 2: SpClass3DmaChannelDV 1: DmaSpClass3ChannelDV 0: 0	R/W	0

25.11.8 DebugEvent1_B

Address Offset: 0x22D for GMAC

Bits	Name	Description	R/W	Reset
31:5	Reserved	Reserved	R	0
4:0	DebugEvent1_B	Debug Events are same as DebugEvent1_A.	R/W	0

25.11.9 DebugCounter2_A

Address Offset: 0x24E for GMAC

Bits	Name	Description	R/W	Reset
31:0	DebugCounter	Debug counter, similar to performance counters. This counts the number of times that DebugEvent1_A is detected.	R/W	0

25.11.10 DebugCounter2_B

Address Offset: 0x24F for GMAC

Bits	Name	Description	R/W	Reset
31:0	DebugCounter	Debug counter, similar to performance counters. This counts the number of times that DebugEvent1_B is detected.	R/W	0

25.11.11 DebugEvent2_A

Address Offset: 0x24C for GMAC

Bits	Name	Description	R/W	Reset
31:5	Reserved	Reserved	R	0
4:0	DebugEvent2_A	GMAC Debug Event is one of the following: 31:14: Reserved 13: DestIdValid on MessageOutInterface 12: SendMessageFirstSet 11: SendMsgSingle 10: SendMsgFirstEntry 9: SendMsgLastEntry 8: RxDescriptorValid 7: FBID127_getting_eaten_away 6: FBID 127 on non PType 5: length 0 non PType ^a 4: length 0 PType ^a 3: FBID 127 on PType 2: Unaligned PType 1: Bad length PType 0: FBID 127 and len 0 on PType	R/W	0

a. Be aware that these two counters may count spurious events.

25.11.12 DebugEvent2_B

Address Offset: 0x24D for GMAC

Bits	Name	Description	R/W	Reset
31:5	Reserved	Reserved	R	0
4:0	DebugEvent2_B	Debug Events are same as DebugEvent2_A	R/W	0

25.11.13 DebugDescWordCounts

Address Offset: 0x23C

Word count of the FIFO described by DEBUG_DESC_WORD_COUNTS_SEL.

Bits	Name	Description	R/W	Reset
31:0	DescWordCount	Returns the word count of the FIFO described by DebugDescWordCountsSel (0x23D)	R	0

25.11.14 DebugDescWordCountsSel

Address Offset: 0x23D

Select Debug Description Word Counts

Bits	Name	Description	R/W	Reset
31:5	Reserved	Reserved	R	0
4:0	SEL	Description See Table 25-5 (All Counts are in units of Descriptors)	R/W	0

Table 25-5. DebugDescWordCountsSel Register SEL Field

#	Description
0	Reserved
1	Reserved
2	Reserved
3	RegFrInSpillMemBytes1
4	RegFrInSpillTmpCounts1 11:8 SpillOutBuf 7:4 SpillInBufA 3:0 SpillInBufB
5	RegFrInFIFOWordCount1
6	Class0SpillMemBytes
7	Class0SpillTmpCounts
8	Class0WordCount
9	AddressFIFOWord0Count
10	Class1SpillMemBytes
11	Class1SpillTmpCounts
12	Class1WordCount
13	AddressFIFOWord1Count
14	Class2SpillMemBytes
15	Class2SpillTmpCounts
16	Class2WordCount
17	AddressFIFOWord2Count
18	Class3SpillMemBytes
19	Class3SpillTmpCounts
20	Class3WordCount
21	AddressFIFOWord3Count
22	FrOutSpillMemBytes
23	FrOutSpillTmpCounts
24	31:24 Tx3DescWordCount 23:16 Tx2DescWordCount 15:8 Tx1DescWordCount 7:0 Tx0DescWordCount

Table 25-5. DebugDescWordCountsSel Register SEL Field (continued)

#	Description
25	31:24 Tmp3WordCount 23:16 Tmp2WordCount 15: 8 Tmp1WordCount 7:0 Tmp0WordCount
26	31:24 Tx_A_P_Word3Count 23:16 Tx_A_P_Word2Count 15:8 Tx_A_P_Word1Count 7:0 Tx_A_P_Word0Count
27	13:7 FreeOutFIFO1WordCount 6:0 FreeOutFIFO0WordCount
28	Reserved
29	Reserved
30	Reserved
31	Reserved

25.11.15 DebugRxDescInMAC

Register Address: 0x23E

Word count of total number of Descriptors in the ingress part of the MAC.

Bits	Name	Description	R/W	Reset
31:0	DescWordCount	Returns the word count of total number of Descriptors in the ingress part of the MAC.	R	0

25.11.16 DebugTxDescInMAC

Register Address: 0x23F

Word count of total number of Descriptors in the egress part of the MAC.

Bits	Name	Description	R/W	Reset
31:0	DescWordCount	Returns the word count of total number of Descriptors in the egress part of the MAC	R	0

25.12 Debug Interrupt Overview

The XLS Processor has 25 general purpose pins which can be configured as input, output or interrupt pins. When configured as interrupt pins, they can be used by external devices to generate an interrupt to an nCPU™ NetLogic virtual CPU or set of nCPUs.

The maskable/non-maskable attribute is controllable from the XLS PIC interrupt-redirection-table (IRT) entry with which the GPIO interrupt is associated.

By making use of the nCPU/thread mask in the IRT table, the masked or non-masked (NMI) interrupt can be broadcast or multicast to a set of nCPUs. Thus, the capability exists to generate a global broadcast NMI from an external device.

The following interrupt characteristics can be defined for debug interrupts:

- Edge- vs. Level triggering
- Rising vs. Falling edge
- High vs. Low level

A debug interrupt can also be configured to use the local/global interrupt round-robin scheduling algorithm from the PIC IRT entry.

NETLOGIC
CONFIDENTIAL



Appendix A Multi-Threading Advantages

A.1

Multi-Threading Improves Processor Performance

To meet the needs of next-generation systems and applications while continuing to provide improved performance, new processor designs must incorporate innovative architectural concepts. Multi-threading is one of the most promising techniques for enabling this new class of processors. This discussion describes the advantage of multi-threading, the unique approach of the XLR/XLS Processor, and the flexibility of its implementation for software developers. The easiest application programming approach uses the parallel method with independent packet processing occurring simultaneously on different threads. The parallel approach minimizes software impacts to existing single image code and the threading hides memory latencies due to cache misses during table searches.

The XLR/XLS Processor Families are designed to enable 4-way multi-threading within each of its processor cores, thus supporting from 4-to-32 threads in from one-to-eight cores. This large number of hardware contexts enables software to more effectively leverage the inherent parallelism exhibited by packet processing applications. In order to better understand the benefits of the XLR/XLS Processor Family architecture, it is instructive to first review the limits of today's more conventional processor architectures.

A.1.1

Traditional Processor Design

Nearly every commercially available integrated general-purpose processor shipping in volume today is designed using a single-threaded architecture, which is performance and application limited by today's standards. As applications are becoming more and more network-centric, this legacy processor design approach fails to address the throughput requirements of today's converging compute and networking paradigm. This evolving packet-oriented environment is characterized by high memory access latencies, which are not effectively managed by conventional processor architectures. This weakness can severely impact processor performance and workload efficiency. When a memory access cannot be serviced immediately and no additional instructions are ready to be executed, conventional processors stall and waste valuable processing cycles. In Section A.1.2 a simple graphical simulation of architectural design is used to illustrate the problems with current designs and the effectiveness of an intelligent multi-threaded approach.

A.1.2

Single-Thread, Single-Issue Processors

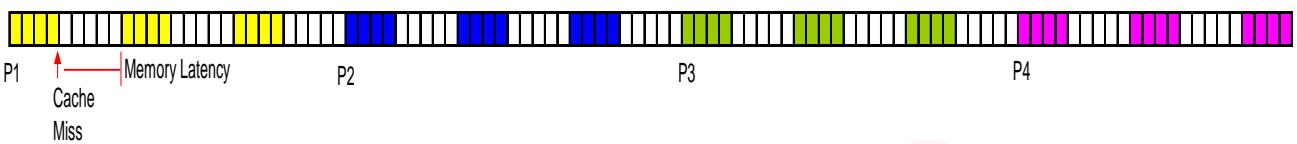
[Figure A-1](#) illustrates typical cycle use and inefficiencies in a single-threaded, single-issue processor. In this example, four packets, each represented by a unique color and indicated by "Px", are processed in order. As cache misses naturally occur, the resulting memory access latencies leave wasted processing cycles. The total amount of time to process the workload in this example is 103 cycles.

Figure A-1. Single-Thread, Single-Issue Throughput

Single Thread / Single Issue

Cycles

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 00 01 02 03



Pre-fetching of data may help reduce wasted cycles. However, useful pre-fetching assumes that there are other independent instructions that can be issued and processed while memory accesses are completed. This is not often the case, and some code is simply not amenable to pre-fetching.

A.1.3 Multiple-Issue Processor Design

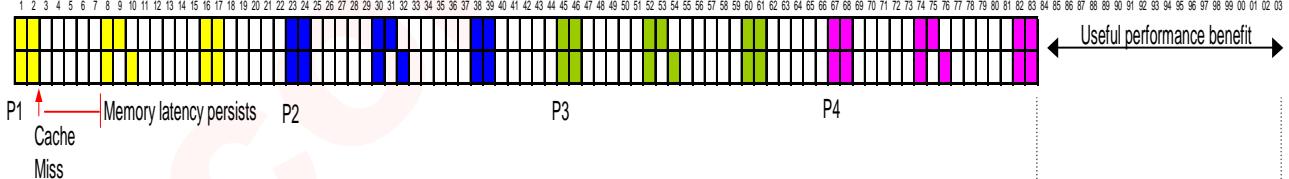
For some applications, instruction-level parallelism can improve performance, executing more instructions per cycle. This is commonly known as a multi-issue (superscalar) design. The ability to execute more than one instruction at a time is accomplished by adding additional functional units (width) to the design. This single-threaded, multiple-issue processor design has been a popular approach to extending processor performance in recent years. [Figure A-2](#) compares the potential processor utilization improvements between multi-issue and single-issue processor designs. The same number of instructions is processed in a shorter amount of time due to the increased number of parallel functional units. The result is a noticeable improvement in performance. However, memory access latencies are still not reduced.

Figure A-2. Single-Thread, Dual-Issue Throughput

Single Thread / Dual Issue

Cycles

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 00 01 02 03

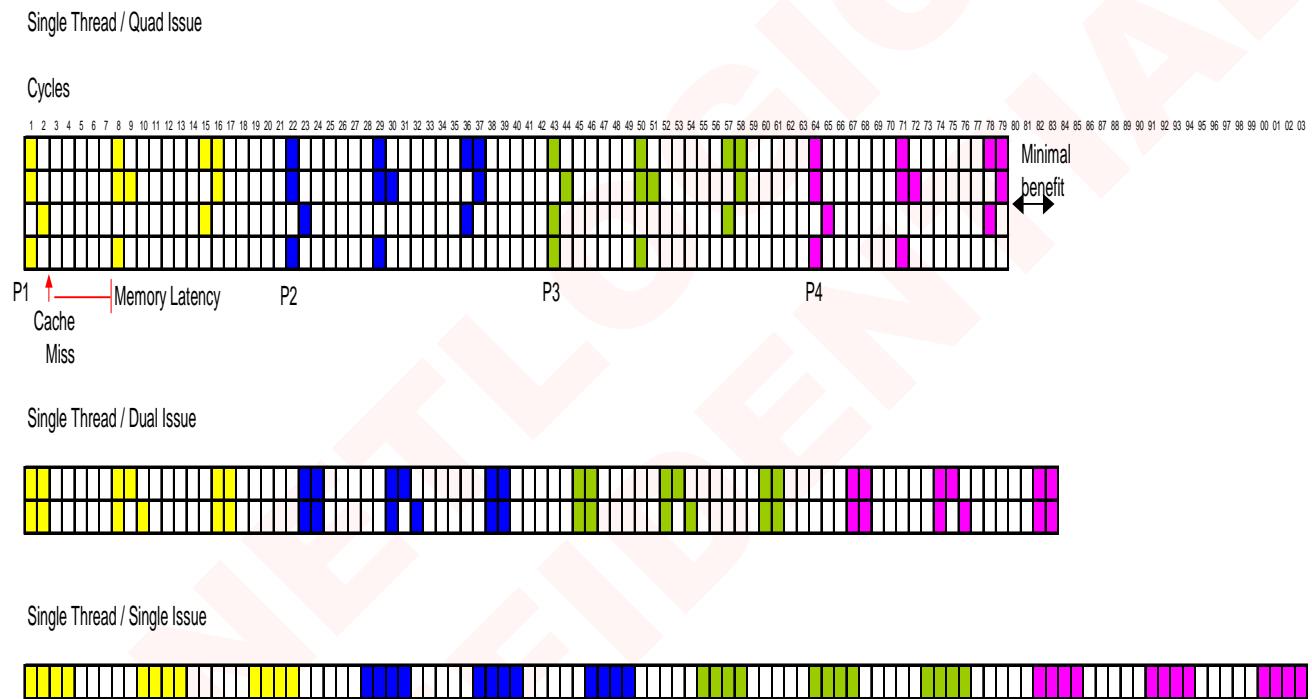


Single Thread / Single Issue

A common design approach has been to continue extending superscalar architectures in an effort to extract even greater parallelism out of the processor. While performance may continue to be improved, there's a point of diminishing return. The limits presented by instruction dependencies (i.e., the available parallelism) and long-latency operations within the single executing thread are simply too great to overcome.

The reduced effectiveness of adding functional units is illustrated in [Figure A-3](#). Moving from a dual-issue to a quad-issue design may improve throughput somewhat, but the payback is not high. Memory latencies become even more of an impact on processor utilization. In addition, many of the added available cycles are wasted due to problems that simply cannot be fully addressed with an increased number of functional units. These include such problems as branch misprediction, load delays, control hazards and TLB misses.

Figure A-3. Single-Thread, Quad-Issue Throughput



While a multi-issue (superscalar) processor has greater potential than single-issue designs, the high memory-latency nature of packet processing applications will eliminate any or all performance advantages. Optimized compiler support becomes more and more crucial to achieving performance gains. And while this yields higher performance than a single-issue single-thread processor, this architecture exhibits the same inefficiency in handling frequent long-latency memory references.

A.1.4

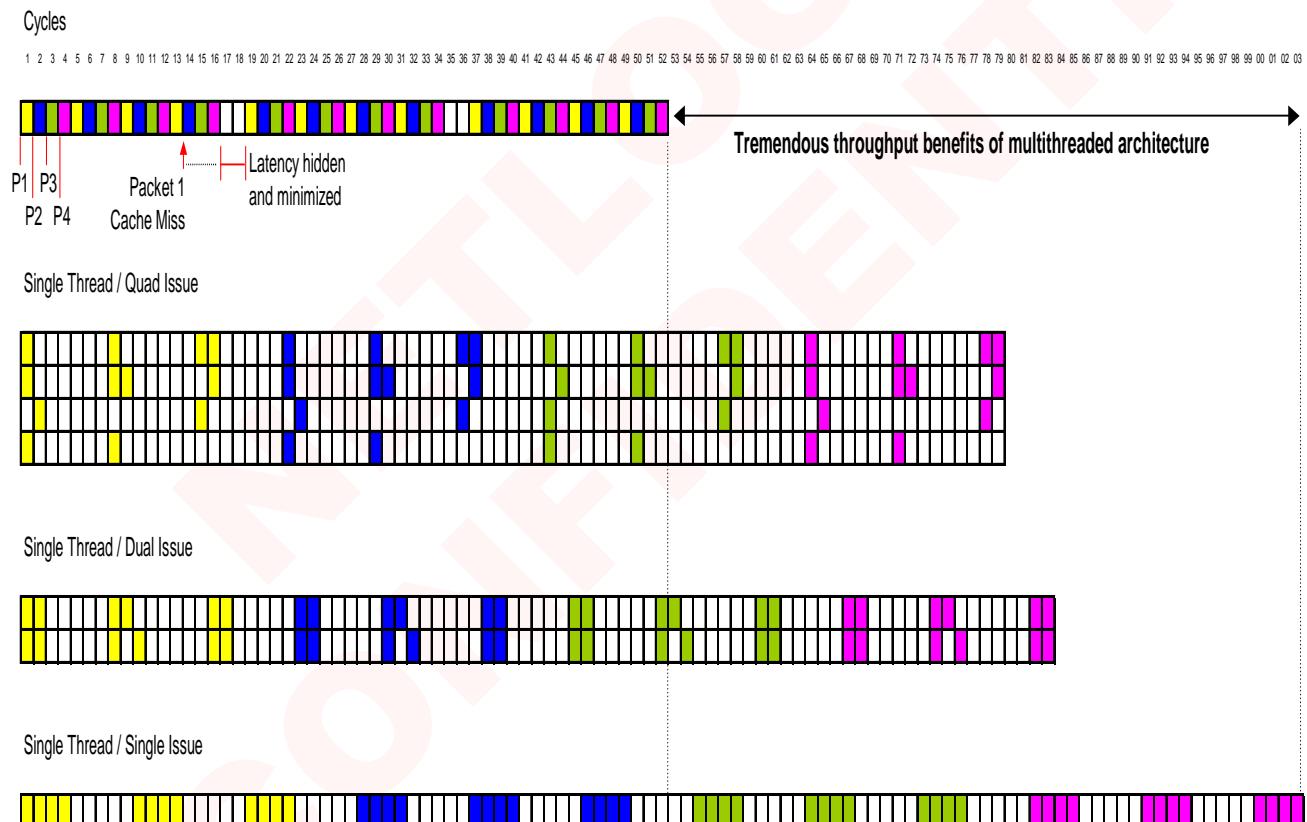
The Multi-Threaded Single-Issue Approach

The approach of adding more CPU cores and / or increasing a processor's superscalar width (greater rates of multi-issue) has plateaued in performance. The inefficiencies illustrated in the previous examples are much more effectively addressed by a multi-threaded architecture. This

approach takes advantage of, and more fully exploits, the packet level parallelism by increasing the total workload in a given amount of time. Memory latencies can be effectively hidden by a well-designed multi-threaded processor, and dramatically improving overall throughput. With this approach, while one thread becomes stalled waiting for requested memory data to return, other threads can continue to process instructions. This maximizes processor resources by minimizing or even completely eliminating the wasted cycles. This workload efficiency improvement is illustrated in [Figure A-4](#). Note in this 4-way threaded example, a packet is processed every fourth cycle. In the clock cycle immediately following packet 1 (yellow), thread 2 can begin operating on packet 2 (blue). Likewise, thread 3 (green) can immediately follow thread 2 and thread 4 (red) can immediately follow thread 3. This process will continue on a round robin basis among the 4 separate threads and corresponding packets. The benefit is most apparent in the event of a cache miss, during which time useful work can be applied to other independent packets while stalled threads wait for memory. Here, the memory latencies are effectively hidden and workload efficiency is highly optimized.

Figure A-4. Multi-threaded Throughput

4-Way Threaded / Single Issue



The example in [Figure A-4](#) demonstrates the advantage of a multi-threaded architecture. NetLogic's XLR/XLS processor not only implements this highly efficient threaded design, but it does so with from 4-to-32 threads implemented in from one-to-eight separate XLR/XLS cores.

The threads of the XLR/XLS processor are capable of achieving outstanding performance without a software paradigm change. If desired, software need not have any knowledge of threads and can simply treat the system as 4-to-32 individual processors connected coherently

in a shared memory environment. For example, an operating system such as Linux requires no software changes to allow it to run as a multi-way SMP operating system on the XLR/XLS processor. This allows software engineers to develop code while leveraging the same standard tool chains and operating systems that have been used in the past development efforts.

**NETLOGIC
CONFIDENTIAL**



Appendix B Revision History

This appendix lists areas where substantive changes were made since the previous release. Informational changes in text marked with change bars throughout.

Table B-1. Listing of Major Revision Areas and Items

Revision	Chap/Section	Area and Change Descriptions
Rev. 1.0		Initial release of this document for customers
Rev 1.1		<p>Significant changes since version 1.0 are listed below. Look for the change bars to see all the changes in this manual since Rev. 1.0.</p> <p>All registers — Register IDs & Indices (Indexes) all now represented as hexadecimal Register IDs. Register addresses all now represented as hexadecimal Address Offsets, write the relevant base address.</p>
	Section 1.2	Updated the list of XLS Processor Family Key Features and added more figures showing complete XLS systems for the various XLS devices.
	Section 4.3	Expanded and updated the section about Coprocessor 2 (COP2) Registers
	Section 6.4.2.1	Updated L1I_WAY_ENABLE register description.
	Section 6.4.2.14	Updated L1 D_WAY_DISABLE register description.
	Chapter 13	Numerous edits. Change bars implemented.
	Section 13.7.1	Updated CAM4_Table.
	Section 18.7	Updated USB register descriptions for USB_COHERENT_MEM_BASE, USB_COHERENT_MEM_LIMIT, USB_L2ALLOC_MEM_BASE, USB_L2ALLOC_MEM_LIMIT, USB_READEX_MEM_BASE, USB_READEX_MEM_LIMIT.
	Section 22.7.1.3	Updated register description for FLASH_CSDEV_PARM[n].
	Section 23.3.1	Updated register descriptions for GPIO Interrupt Enable, GPIO Input Inversion Control, GPIO IO Direction Control, GPIO Output Write, GPIO Input Read, GPIO Interrupt Clear, and GPIO Interrupt Status.
	Section 23.3.2.4	Updated register description for GPIO DRAM AB PLL Status.
	Section 25.4	Updated register descriptions for TAP_INST_REG, TAP Instruction Register, and Address Register.
	Section 25.10.1	Updated register description for Request Match Register 1.

Table B-1. Listing of Major Revision Areas and Items (*continued*)

Revision	Chap/Section	Area and Change Descriptions
Rev 1.3	General updating and refinement of details throughout and Updates relating to XLS Product Alerts	
	Table 1-1	XLS4xx devices re-documented with support for all 8 GMACs and all 8 SGMII ports
	Section 1.5	The Multi-Threading description was moved to Appendix B
	Section 2.1.1	Added MIPS32 & MIPS64 Conformance document references
	Section 2.5	Added Instruction Timing relationships
	Section 2.6	Added Code Compatibility section with reference to XLR code compatibility
	Chapter 8	General re-write to add more descriptive material Added some missing bit descriptions to register fields
	Chapter 9	General re-write to add more descriptive material Added some missing bit descriptions to register fields
	Section 11.3.4	Revised DRAM configurations supported
	Chapter 12	General re-write to add more descriptive material Added some missing bit descriptions to register fields
	Table 12-1	FMN Station IDs 105, 106, and 107 are valid for all XLS devices; none of these station IDs are reserved. All XLS devices have four-channel DMA.
	Chapter 13	Changed "CRC32" to "CRC7", except in 13.8 Prepad which is CRC32. References to internal PCIe module "data bus interface" (DBI) removed.
	Chapter 14	Addition of information on Elliptical Curve Cryptography (ECC)
	16.5.2.16	Added bit functions to I/O_CTRL - Interface Control Register
Rev. 1.3 (Cont.)	Chapter 18	Detailed PCIe discussion added.
	Section 19.10.3	PCIe Configuration Register access mechanism updated
	Section 19.4.3.4	Added IIC Interrupts details
	Chapter 20	Additional information on NAND Flash technology and XLS NAND Flash support.
	Chapter 20 & 21	Clarification of limitations on use of GPIO[13:10] as interrupts.
Rev. 2.00	Chapter 1	New XLS616 and XLS416 devices added and XLS608 upgraded. Optional SRIO and XAUI Interfaces, and IEEE-1588 PTP function added just for XLS616/608 and XLS416.
	Chapter 8	Added memory control functions for XLS616 and XLS416 devices and SRIO and XAUI Interfaces
	Chapter 12	Added Fast-Messaging functions for XLS616 and XLS416 devices (two more cores, etc.) and SRIO and XAUI Interfaces
	Chapter 13	Added Network Accelerator functions for XLS616 and XLS416 devices and SRIO and XAUI Interfaces, Added PREPAD data for formats 01, 10, 11
	Chapter 15	Added data to support XLS616 and XLS416 devices
	Chapter 16	Modified SGMII/GMAC Interface to show effect of optional XAUI ports
	Chapter 17	Added this new chapter for XAUI Interface
	Chapter 19	New XLS616, XLS608 and XLS416 device data added to PCIe interface. The XLS408/404 devices now support one PCIe 1x4 channel and two 1x1 channels.
	Chapter 20	Added this new chapter for SRIO Interface
	Chapter 23	Revised Pin-Strapping options added to "GPIO Reset Configuration Reg. (0x15)" for new XAUI and SRIO interface selection and configuration
	Chapter 24	Added this new chapter for IEEE 1588 Precision Time Protocol
	Chapter 25	Added or modified parameters to support new XLS616 and XLS416 devices

Table B-1. Listing of Major Revision Areas and Items (*continued*)

Revision	Chap/Section	Area and Change Descriptions
Rev 3.00	General	Change Document Number from "2041PM" to "10799V300PM" Add XLS108 and XLS104 devices (similar to the XLS208 & 204) to the XLS Family with all applicable architectural features. The XLS616, XLS608, XLS416, XLS408 and XLS404 devices have been upgraded to Rev "B" with increased capabilities as shown in Block Diagrams, family listings, and applicable Interface chapters.
	Chapter 1 Introduction	Add XLS108 and XLS104 devices to Block Diagrams and listing summaries. Change XLS408 and XLS404 interfaces in the Block Diagrams.
	Chapter 4 uP Cores	4.2.3.17 – Add new Processor ID numbers for upgraded Rev "B" devices 4.4.3.7 -- Add restriction Note for "SpecCacheMissEn" bit
	Chapter 5 CPU Instr.	5.3.9 -- Add MsgWait instruction sequence to "Exceptions."
	Chapter 6 L1 Cache	6.4.2.13 -- Add note about poisoning interrupt logging.
	Chapter 7 L2 Cache	7.4.2 -- Add Caution Note
	Chapter 8 Memory & I/O	8.5.1 -- Add "System Bridge Controller Read-Access Requirements" 8.9.16 BIT_ERR_EN register -- Enable/disable bit levels reversed (reset = disabled) 8.10.6 Add MISC FUNCTION CTL1[5] "SRIO_ORDERING_MODE"
	Chapter 9 DMA	9.2.1.4 -- Added Caution Note about sending "zero-sized checksum/CRC requests" 9.3.1.1 Add restriction to "Simple Transfer messages"
	Chapter 10 PIC	Table 10-3, For XLS6xx/4xx Bits[28:29] Add PCIE_LINK[2 & 3] - "FATAL ERROR" is merged with "PCIE_LINK Interrupts". In SRIO mode, Bits[26:29] are assigned to SRIO Controllers 0-3 respectively.
	Chapter 11 DRAM	A single bank of 32/36-bit DRAMs is supported by the XLS1xx series. 11.3.4 Updated supported DRAM configurations data

Table B-1. Listing of Major Revision Areas and Items (*continued*)

Revision	Chap/Section	Area and Change Descriptions
Rev 3.00 (Cont)	Chapter 12 Fast Msg Net	Table 12-1, PCIe Bucket allocation for XLS408 & XLS404 has been changed. They now use the same bucket allocation as the XLS616. The separate XLS408 & XLS404 buckets are no longer used.
	Chapter 13 Net Accel.	The XLS1xx series supports GMAC Ports 0&1 only. 13.4.2.1 and 13.7.2.3 -- Add Read and Write restrictions during GMAC traffic flow. Tables 13-5 & 13-6 -- Tx P2D & P2P Packet Descriptor[61] RdEx Add "Usage" restriction Table 13-16, bits[127:0] -- Add Note about the "raw Parser extracted key" 13.10.2.2-- RxControl[13:12] moved to [12:11], RGMII Mode[11] moved back to [10], bit[9] Rsvd 13.10.2.3 -- Add requirement for DescPackCtrl.RegularSize to be greater than a quarter of MTU size or Maximum Frame Length of the interface. 13.10.6 Transmit Data FIFO Configuration -- Add details and figure.
	Chapter 14 Security	Tables 14-1 thru 14-3 -- Add performance numbers for 750 and 600 MHz operation 14.3.2, bottom of page, Initialization Vector/Cipher Offset changed final value from "1" to "8/16"
	Chapter 15	15.3.3.5, Deflate Save -- add restriction and Note for "greater than 32 Byte first block" 15.3.3.5, Inflate Restore, -- restriction added to "warm-up data requirements" 15.4.1.3 -- Add "Reported Block Length Setup" requirements
	Chapter 16 SGMII/ RGMII	16.3.2 -- Note added restricting minimum transmit packet size to at least nine bytes 16.5.2.1 -- Note added about reading "Reserved" registers 16.5.2.5 -- Note added to "RETRANS_MAX" 16.5.2.6 -- Note added to "MAX_FM_LEN" set up 16.5.2.10 -- MGMT_CLK_SEL '111' value was changed from "28" to "54". I/O_CTRL[24] was "PHY_MODE" is now "Reserved" Section 16.5.2.18 – 29, MAC_ADDR_MASK0 was shown ANDed with MAC_ADDR0, and MAC_ADDR_MASK1 was shown as ANDed with MAC_ADDR1; they should be ANDed with MAC_ADDR2 and MAC_ADDR3, respectively. MAC_ADDR0_LO/HI and MAC_ADDR1_LO/HI are used for full or exact match. Section 16.5.2.30 – CAUTION added for possible interface-hang, plus Workaround
	Chapter 17 XAUI	Add XAUI Interface for the XLS408 device.
	Chapter 18 USB	Add one USB Port 0 Interface for the XLS108 and XLS104 devices.
	Chapter 19 PCIe	Add XLS1xx series PCIe configurations. Revise XLS6xx/4xx/2xx Config diagrams Define XLS1xx Registers and Device Numbers. 19.6.2 and 19.7.4 -- Change XLS408 & XLS404 Register allocation. Removed whole Section 19.8.4 "XLS408 and XLS404 Global PCI Express Interface Registers" -- The XLS4xx devices now use the same register set (Section 19.8.3) as the other XLS device Families. 19.7.3 -- Add new section to "Ensure Writes Complete Before Reads" 19.10.3 -- Revise RC-Mode device numbers for XLS6xx, XLS4xx and XLS2xx, and add device numbers for XLS1xx series.

Table B-1. Listing of Major Revision Areas and Items (*continued*)

Revision	Chap/Section	Area and Change Descriptions
Rev 3.00 (Cont)	Chapter 20 SRIO	<p>20.2 Add optional SRIO Interface for the XLS408 and XLS404 devices</p> <p>Figure 20-9, Status Queue Entry changed from “4-words” to “8-words, but bytes 16-31 are Reserved”.</p> <p>20.4.5.2 Following Figure 20-12, Each entry on a Mailbox Queue is changed from “4-words” to “8-words, but bytes 16-31 are Reserved”.</p> <p>20.5.3 SRIO_CTRL[30:24] -- Add four new bit fields [27:24], [28], [29] & [30]</p> <p>20.7.7 Change the Status-Queue pointers to 32-byte alignment and add programming procedure</p> <p>20.7.11 Change mailbox-Queue pointers to 32-byte alignment and add programming procedure.</p>
	Chapter 21 UART, I2C	21.4.3.1 -- Change requirements for “Sequential Write Transfers”
	Chapter 23 GPIO & Peripherals I/F	<p>23.3.3 & 23.3.3.1 Add PLL_CTRL Register Default and data for XLS1xx</p> <p>23.3.5.1 GPIO Reset Configuration[21:20] PCIE_CFG changed XLS408 & XLS404, added XLS1xx -- for optional SRIO_CFG added XLS404 configuration value.</p>
	Chapter 25 Debugging	<p>Read Data Register[4:0] RSHITID is not functional (see bit description for work-around)</p> <p>25.10.11 -- Note added to “length 0 non PType” and “length 0 PType” counters</p>

Table B-1. Listing of Major Revision Areas and Items (*continued*)

Revision	Chap/Section	Area and Change Descriptions
Rev. 3.10	General	Add back the details for the XLS408-Lite and XLS404-Lite devices. These devices are still being supported for designs that are already committed. The XLS408 and XLS404 are recommended for new designs, since they have more features and capabilities.
	Chapter 4 Core Registers	4.2.3.17 -- Bits[15:8] are the Processor ID values for all of the XLS devices.
	Chapter 8 Memory & I/O	8.10.6 -- Change Register Address from "0x0E6" to "0x0EC"
	Chapter 10 PIC	Table 10-3 "PIC Interrupt Redirection Table Map" -- Add the PIC-IRT details for the XLS4xx-Lite devices.
	Chapter 12 Fast Msg Net	Table 12-1 -- Add PCIe Bucket ID and allocation for XLS408-Lite & XLS404-Lite buckets back into the table.
	Chapter 13 Net. Accelerator	13.10.2.2 -- Revise Reset value for bits [8:6]
	Chapter 14 Security	14.3.2 Alignment Concept "Initialization Vector - Ciper Offset" --change (n + 14 + 20 + 8 + 1) to (n + 14 + 20 + 8 + 8/16) 14.7.7 Description of Config3 Register added.
	Chapter 15 Compression	Section 15.4.5.1, Section 15.4.5.3 -- Change bit 53 of message descriptor to reserved and modifiy range of Length field to bits [52:40]. Section 15.4.5.4 -- The maximum length specified in a data descriptor (bits [55:40]) is limited to 0xFFE0.
	Chapter 17 XAUI Interface	Section 17.5.25 – 28, MAC_ADDR_MASK0 was shown ANDed with MAC_ADDR0, and MAC_ADDR_MASK1 was shown as ANDed with MAC_ADDR1; they should be ANDed with MAC_ADDR2 and MAC_ADDR3, respectively. MAC_ADDR0_LO/HI and MAC_ADDR1_LO/HI are used for full or exact match.
	Chapter 19 PCIe	19.2.2 -- Add XLS408-Lite and XLS404-Lite configuration options Fig. 19-2 -- Add Internal XLS4xx-Lite PCIe Topology 19.3.1.1 -- Add XLS4xx-Lite configuration end-point options 19.6.2 -- Add XLS4xx-Lite Interrupt setup 19.8.4 -- Add XLS4xx-Lite Global PCI Express Interface Registers 19.10.3 -- Add XLS4xx-Lite Device Number of the address to access Configuration Regs.
	Chapter 23 GPIO	23.3.5.1 -- Update GPIO Configuration Register with XLS4xx-Lite configuration details [25:24] Change XLS408 configuration options. 23.3.5.6 -- Added field descriptions to GPIO_SGMII_PHY_CTL register. Corrected reset value for PHY Acjt LVL field.
	Chapter 24 1588_PTP	24.1.4.8 -- Add control bits [12:7] and change R/W status of all bits from "R" to "R/W" 24.1.5 -- Add reference section "1588_PTP Control in Network Accelerator"
Rev.3.11 Draft	Chapter 5 Instructions	5.3.3 - 5.3.6 Clarify COP2 Move instructions 5.3.10 & 5.3.12, LDADDW and LADDD, remove "integer overflow" from exceptions
	Chapter 10 PIC	10.3.1, Table 10-4, IPIBase, redefine bits[31:16] to show unused (reserved) bits
	Chapter 14 SEC	Tables 14-7 & 14-20 "Data Dsecriptor Vector[53]" change "WRB_COH" to "Reserved"
	Chapter 16	16.5.3.1 "Use of Carry and Carry Mask Registers" subsection, second paragraph corrected to refer to the OverFlowEn field in the StatCtrl register instead of AUTOZ.
	Chapter 17	17.6.4 "Carry and Carry Mask Registers", second paragraph corrected to refer to the OverFlowEn field in the StatCtrl register instead of AUTOZ.

Table B-1. Listing of Major Revision Areas and Items (*continued*)

Revision	Chap/Section	Area and Change Descriptions
Rev.3.20	All chapters	Removed B designation from the 4xx and 6xx series parts. Changed all 404A and 408A designations to 404-Lite and 408-Lite respectively.
	Chapter 4 MIPS	Section 4.2.3.17. Added stepping information to bits 7:0 in Revision ID field.
	Chapter 9 DMA	Updated Table 9-1, Message formats.
	Chapter 12 FMN	Updated Table 12-2 (SGMII MsgBucketSize) and associated text.
	Chapter 13 NetworkAccel	Added Section 13.3.6.3, Read Exclusive bit and Cache Line Invalidation.
	Chapter 14 SecurityAccel	Updated descriptions in Table 14-14, Checksum Destination Address.
	Chapter 19 PCIe	Added Section 19.4.4, PCIe Configuration Requests. Added registers in Sections 19.8.3.66 and 19.8.3.67.
	Chapter 23 GPIO	Added Section 23.3.5.7, SGMII[4-7] PLL control register. Added Section 23.3.5.8, SGMII[4-7] PHY control register.
Rev.3.21	Chapter 15 Compression Decompression Engine	Made minor corrections throughout the chapter.
	Chapter 20 SRIO	Corrected description of bit 27 in 20.6.18 P0_EAS_CSR register. Also corrected typos for bits 22 and 23.
	Chapter 23 GPIO	Corrected description of GPIO20 in Table 23-1, "GPIO Port and Flash Memory Signal Descriptions". Added description for 23.3.5.9 GPIO_SRIO_CLK_CTRL register at 0x24.
Rev.3.30	Chapter 16 SGMII-RGMII Chapter 12 Fast Message Network	Removed caution and workaround from Section 16.5.2.30. The caution only applies to the XLS4xxLite and is documented in the Revision Guide. Added Section 12.2.5, PCIE_MSG_TX_THRESHOLD register. This register also resides in the PCIe chapter and was copied here to indicate that it is duplicated in multiple blocks. Converted entire document to the NetLogic template. Updated Table 12-2, msgBucketSize Registers.
3.40	Chapters 1-4, 6-8, 12, and 15	Changed vCPU to nCPU™ NetLogic virtual CPU.
	Chapter 11 DRAM	Added description of DDR_DYN_DATA_PAD_CTRL register, a register that is only present in the XLS 100 and 200 series of processors.

NOTES:

NETLOGIC
CONFIDENTIAL

**NETLOGIC
CONFIDENTIAL**

NetLogic Microsystems
18920 Forge Drive
Cupertino, CA 95014
+1 (408) 434-5700

<http://www.netlogicmicro.com>