

2013.12.30

cv\_54006



Subscribe



Send Feedback

The hard processor system (HPS) in the Altera® SoC FPGA device includes a stand-alone, full-featured ARM® Cortex™-A9 MPCore™, single- or dual-core 32-bit application processor. The Cortex-A9 microprocessor unit (MPU) subsystem is composed of a Cortex-A9 MPCore, a level 2 (L2) cache, an Accelerator Coherency Port (ACP) ID mapper, and debugging modules.

## Features of the Cortex-A9 MPU Subsystem

The Altera Cortex-A9 MPU subsystem provides the following features:

- One or two Cortex-A9 processors
- Interrupt controller
- Private interval and watchdog timer for each processor
- Global timer
- TrustZone® system security extensions
- Symmetric multiprocessing (SMP) and asymmetric multiprocessing (AMP) modes
- Debugging modules

© 2013 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, HARDCOPY, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at [www.altera.com/common/legal.html](http://www.altera.com/common/legal.html). Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

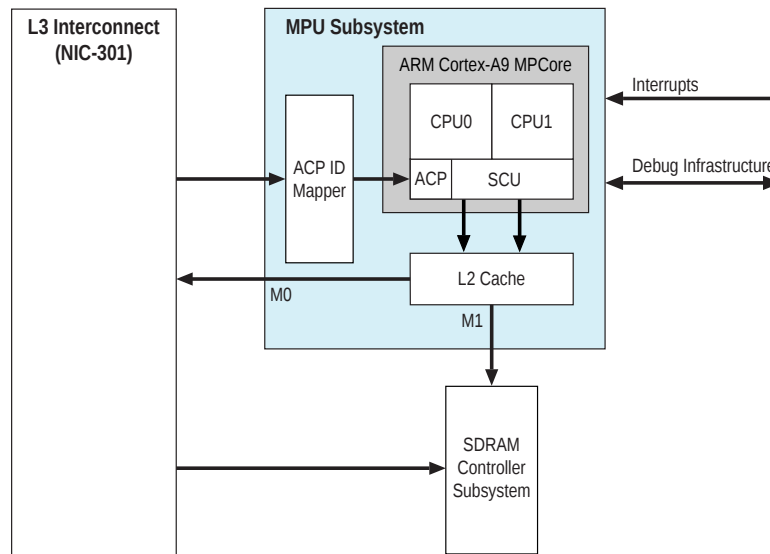
ISO  
9001:2008  
Registered



## Cortex-A9 MPU Subsystem Block Diagram and System Integration

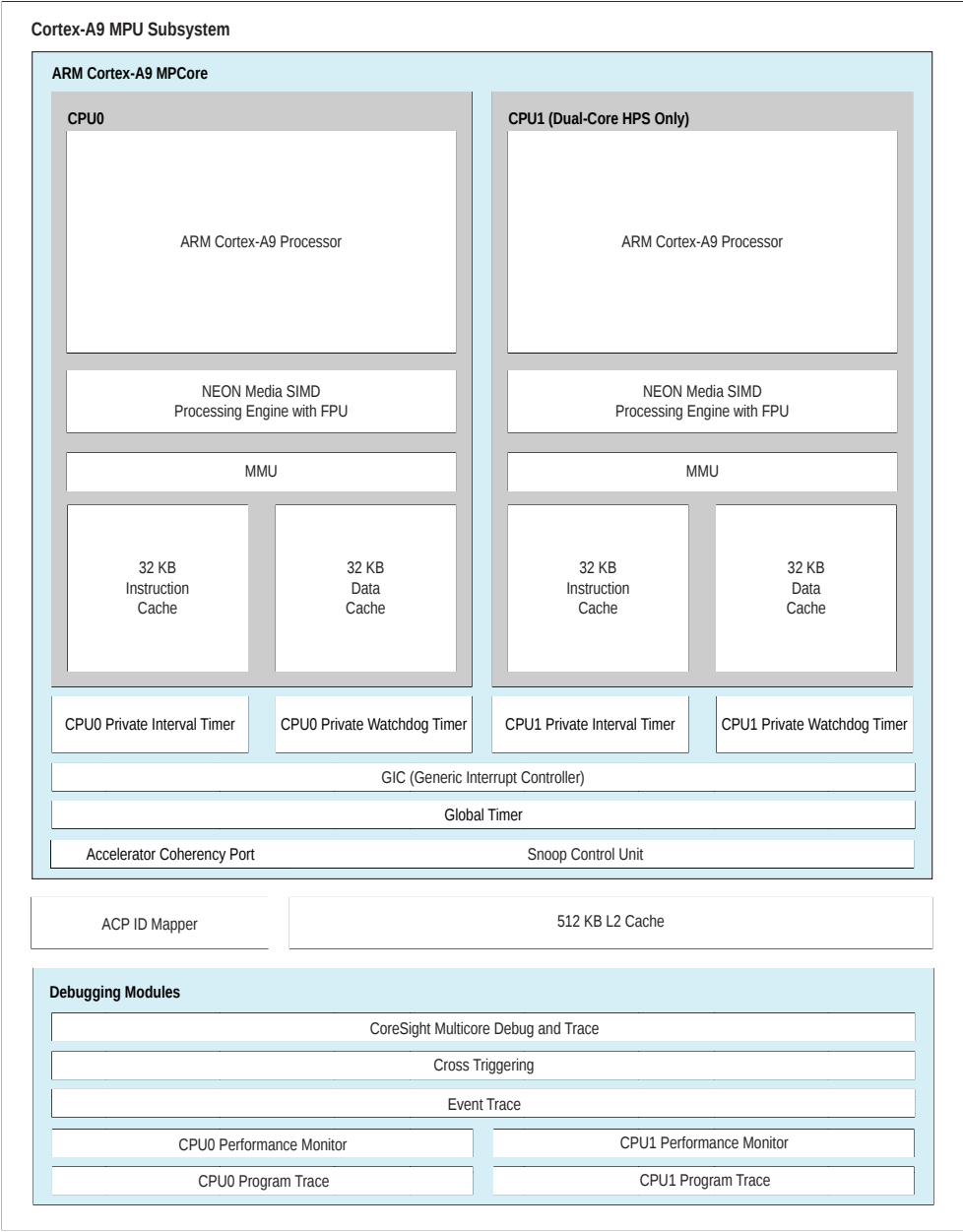
### Cortex-A9 MPU Subsystem with L3 Interconnect

This figure shows a dual-core MPU subsystem in the context of the HPS, with the L2 cache. The L2 cache can access either the level 3 (L3) interconnect fabric or the SDRAM.



# Cortex-A9 MPU Subsystem Internals

This figure shows a block diagram of the Altera Cortex-A9 MPU subsystem.



## Cortex-A9 MPU Subsystem Components

The Altera Cortex-A9 MPU subsystem consists of the following hardware blocks:

- ARM Cortex-A9 MPCore
- ARM L2C-310 L2 cache controller
- ACP ID mapper
- Debugging and trace features

### Cortex-A9 MPCore

The MPU subsystem includes a stand-alone, full-featured ARM Cortex-A9 MPCore single- or dual-core 32-bit application processor. The processor, like other HPS masters, can access IP in the FPGA fabric through the HPS-to-FPGA bridges.

### Functional Description

The ARM Cortex-A9 MPCore contains the following blocks:

- One or two Cortex-A9 Revision r3p0 processors operating in SMP or AMP mode
- Snoop control unit (SCU)
- Private interval timer for each processor core
- Private watchdog timer for each processor core
- Global timer
- Interrupt controller

Each transaction originating from the Altera Cortex-A9 MPU subsystem can be flagged as secure or nonsecure.

### Implementation Details

**Table 6-1: Cortex-A9 MPCore Processor Configuration**

This table shows the parameter settings for the Altera Cortex-A9 MPCore.

Feature	Options
Cortex-A9 processors	1 or 2
Instruction cache size per Cortex-A9 processor	32 KB
Data cache size per Cortex-A9 processor	32 KB
TLB size per Cortex-A9 processor	128 entries
Media Processing Engine with NEON™ technology per Cortex-A9 processor <sup>(1)</sup>	Included
Preload Engine per Cortex-A9 processor	Included
Number of entries in the Preload Engine FIFO per Cortex-A9 processor	16

<sup>(1)</sup> Includes support for floating-point operations.

Feature	Options
Jazelle DBX extension per Cortex-A9 processor	Full
PTM interface per Cortex-A9 processor	Included
Support for parity error detection <sup>(2)</sup>	Included
ARM_BIST	Included
Master ports	Two
Accelerator Coherency Port	Included

#### Related Information

[ARM Infocenter \(www.infocenter.arm.com\)](http://www.infocenter.arm.com)

## Cortex-A9 Processor

Each Cortex-A9 processor includes the following hardware blocks:

- ARM NEON™ single instruction, multiple data (SIMD) coprocessor with vector floating-point (VFP) v3 double-precision floating point unit for media and signal processing acceleration
  - Single- and double-precision IEEE-754 floating point math support
  - Integer and polynomial math support
- Level 1 (L1) cache with parity checking
  - 32 KB four-way set-associative instruction cache
  - 32 KB four-way set-associative data cache
- CoreSight™ Program Trace Macrocell (PTM) supporting instruction trace

Each Cortex-A9 processor supports the following features:

- Dual-issue superscalar pipeline with advanced branch prediction
- Out-of-order (OoO) dispatch and speculative instruction execution
- 2.5 million instructions per second (MIPS) per MHz, based on the Dhrystone 2.1 benchmark
- 128-entry translation lookaside buffer (TLB)
- TrustZone security extensions
- Configurable data endianness

<sup>(2)</sup> For a description of the parity error scheme and parity error signals, refer to the Cortex-A9 Technical Reference Manual, available on the ARM website (infocenter.arm.com).

- Jazelle® DBX Extensions for byte-code dynamic compiler support
- The Cortex-A9 processor architecture supports the following instruction sets:
  - The ARMv7-A performance-optimized instruction set
  - The memory-optimized Thumb®-2 mixed instruction set
    - Improves energy efficiency
    - 31% smaller memory footprint
    - 38% faster than the original Thumb instruction set
  - The Thumb instruction set—supported for legacy applications
- Each processor core in the Altera HPS includes a memory management unit (MMU) to support the memory management requirements of common modern operating systems.

The Cortex-A9 processors are designated CPU0 and CPU1.

Detailed documentation of ARM Cortex-A9 series processors is available on the ARM website ([infocenter.arm.com](http://infocenter.arm.com)).

#### Related Information

[ARM Infocenter \(www.infocenter.arm.com\)](http://www.infocenter.arm.com)

## Interactive Debugging Features

Each Cortex-A9 processor has built-in debugging capabilities, including the following features:

- Six hardware breakpoints, including two with Context ID comparison capability
- Four watchpoints

The interactive debugging features can be controlled by external JTAG tools or by processor-based monitor code.

For more information about the interactive debugging system, refer to the *Debug* chapter of the Cortex-A9 Technical Reference Manual, available on the ARM website ([infocenter.arm.com](http://infocenter.arm.com)).

#### Related Information

[ARM Infocenter \(www.infocenter.arm.com\)](http://www.infocenter.arm.com)

## L1 Caches

Cache memory that is closely coupled with an associated processor is called level 1, or L1 cache. Each Cortex-A9 processor has two independent 32 KB L1 caches—one for instructions and one for data—allowing simultaneous instruction fetches and data access.

Each L1 cache is four-way set associative, with 32 bytes per line, and supports parity checking.

## Preload Engine

The preload engine (PLE) is a hardware block that enables the L2 cache to preload selected regions of memory. The PLE signals the L2 cache when a cache line will be needed in the L2 cache, by making the processor data master port start fetching the data. The processor data master does not complete the fetch or return the data to the processor. However, the L2 cache can then proceed to load the cache line. The data is only loaded to the L2 cache, not to the L1 cache or processor registers.

The preload functionality is under software control. The following PLE control parameters must be programmed:

- Programmed parameters, including the following:
  - Base address
  - Length of stride
  - Number of blocks
- A valid bit
- TrustZone memory protection for the cache memory, with an NS (non-secure) state bit
- A translation table base (TTB) address
- An Address Space Identifier (ASID) value

For more information about the PLE, refer to the *Preload Engine* chapter of the *Cortex-A9 Technical Reference Manual*, available on the ARM website ([infocenter.arm.com](http://infocenter.arm.com)).

#### Related Information

[ARM Infocenter \(www.infocenter.arm.com\)](http://www.infocenter.arm.com)

## Floating Point Unit

Each ARM Cortex-A9 processor includes full support for IEEE-754 floating point operations. The floating-point unit (FPU) fully supports half-, single-, and double-precision variants of the following operations:

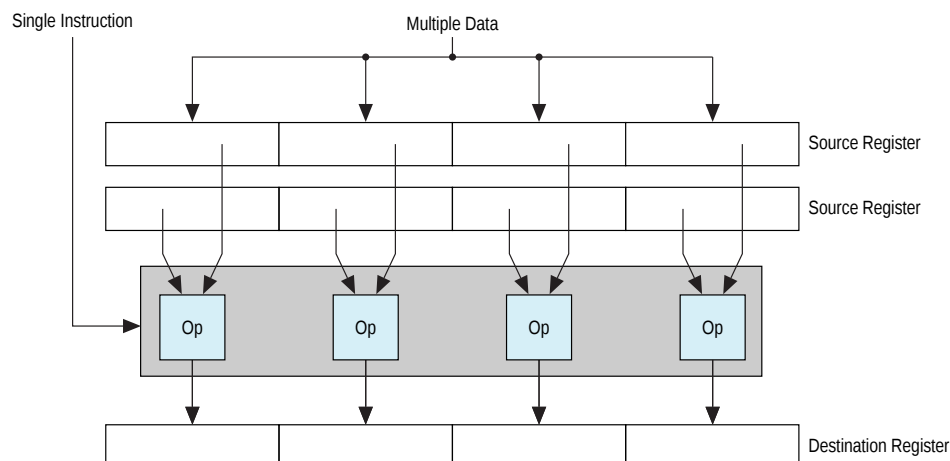
- Add
- Subtract
- Multiply
- Divide
- Multiply and accumulate (MAC)
- Square root

The FPU also converts between floating-point data formats and integers, including special operations to round towards zero required by high-level languages.

## NEON Multimedia Processing Engine

The NEON multimedia processing engine (MPE) provides hardware acceleration for media and signal processing applications. Each ARM Cortex-A9 processor includes an ARM NEON MPE that supports SIMD processing.

## Single Instruction, Multiple Data (SIMD) Processing



### Features of the NEON MPE

The NEON processing engine accelerates multimedia and signal processing algorithms such as video encoding and decoding, 2-D and 3-D graphics, audio and speech processing, image processing, telephony, and sound synthesis.

The Cortex-A9 NEON MPE performs the following types of operations:

- SIMD and scalar single-precision floating-point computations
- Scalar double-precision floating-point computation
- SIMD and scalar half-precision floating-point conversion
- 8-bit, 16-bit, 32-bit, and 64-bit signed and unsigned integer SIMD computation
- 8-bit or 16-bit polynomial computation for single-bit coefficients

The following operations are available:

- Addition and subtraction
- Multiplication with optional accumulation (MAC)
- Maximum or minimum value driven lane selection operations
- Inverse square root approximation
- Comprehensive data-structure load instructions, including register-bank-resident table lookup

For more information about the Cortex-A9 NEON MPE, refer to the Cortex-A9 NEON™ Media Processing Engine Technical Reference Manual, Revision r3p0, which you can download from the ARM website ([infocenter.arm.com](http://infocenter.arm.com)).

#### Related Information

[ARM Infocenter \(www.infocenter.arm.com\)](http://www.infocenter.arm.com)

### Memory Management Unit

The MMU is used in conjunction with the L1 and L2 caches to translate virtual addresses used by software to physical addresses used by hardware. Each processor has a private MMU.



## TLBs Supported By the MMU

TLB Type	Memory Type	Number of Entries	Associativity
Micro TLB	Instruction	32	Fully associative
Micro TLB	Data	32	Fully associative
Main TLB	Instruction and Data	128	Two-way associative

## TLB Features

The main TLB has the following features:

- Lockable entries using the lock-by-entry model
- Supports hardware page table walks to perform look-ups in the L1 data cache

For more information about the MMU, refer to the *Memory Management Unit* chapter of the *Cortex-A9 Technical Reference Manual*, available on the ARM website (infocenter.arm.com).

The MPU address map is divided into the following regions:

- The boot region
- The SDRAM region
- The FPGA slaves region
- The HPS peripherals region

### Related Information

[ARM Infocenter \(www.infocenter.arm.com\)](http://www.infocenter.arm.com)

## The Boot Region

The boot region is 1 MB in size, based at address 0. After power-on, or after reset of the L3 interconnect, the boot region is occupied by the boot ROM, allowing the Cortex-A9 MPCore to boot. Although the boot region size is 1 MB, accesses beyond 64 KB are illegal because the boot ROM is only 64 KB.

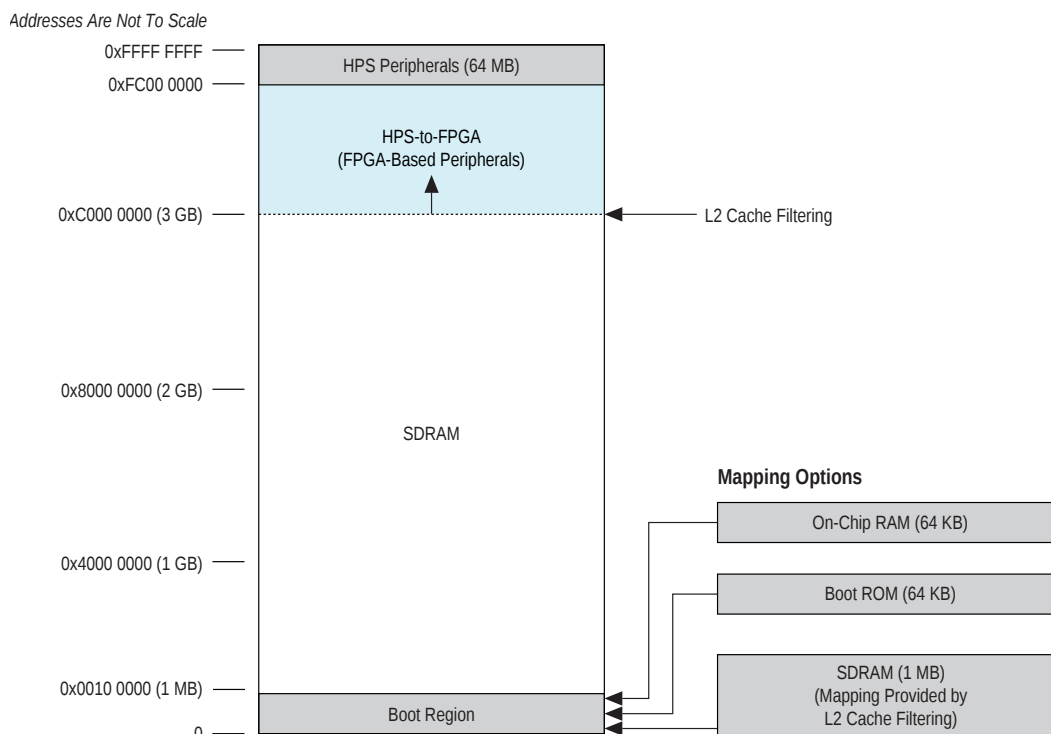
The 1 MB boot region can be subsequently remapped to the bottom 1 MB of SDRAM region.

**Note:** Alternatively, the boot region can be mapped to the 64 KB on-chip RAM. For more information, refer to the *Interconnect* chapter in the *Cyclone V Device Handbook, Volume 3*.

### Related Information

[The SDRAM Region](#) on page 6-10

## The MPCore Address Map



### The SDRAM Region

The SDRAM region starts at address 0x100000 (1 MB). The top of the region is determined by the L2 cache filter.

The L2 cache contains a filtering mechanism that routes accesses to the SDRAM and L3 interconnect. The filter defines a filter range with start and end addresses. Any access within this filter range is routed to the SDRAM subsystem. Accesses outside of this filter range are routed to the L3 interconnect.

The start and end addresses are specified in the following register fields:

- `reg12_addr_filtering_start.address_filtering_start`
- `reg12_address_filtering_end.address_filtering_end`

To remap the lower 1MB of SDRAM into the boot region, set the filter start address to 0x0 to ensure accesses between 0x0 and 0xFFFFF are routed to the SDRAM. Independently, you can set the filter end address in 1 MB increments above 0xC0000000 to extend the upper bounds of the SDRAM region. However, you achieve this extended range at the expense of the FPGA peripheral address span. Depending on the address filter settings in the L2 cache, the top of the SDRAM region can range from 0xBF000000 to 0xFB000000.

#### Related Information

[L2 Cache](#) on page 6-28

### The FPGA Slaves Region

The Cortex-A9 MPU subsystem supports the variable-sized FPGA slaves region to communicate with FPGA-based peripherals. This region can start as low as 0xC0000000, depending on the L2 cache filter settings. The top of the FPGA slaves region is located at 0xFB000000. As a result, the size of the FPGA slaves region can range from 0 to 0x3F000000 bytes.

## The HPS Peripherals Region

The HPS peripherals region is the top 64 MB in the address space, starting at 0xFC000000 and extending to 0xFFFFFFFF. The HPS peripherals region is always allocated to the HPS dedicated peripherals for the Altera Cortex-A9 MPU subsystem.

## Performance Monitoring Unit

Each Cortex-A9 processor has a Performance Monitoring Unit (PMU). The PMU supports 58 events to gather statistics on the operation of the processor and memory system. Six counters in the PMU accumulate the events in real time. The PMU counters are accessible either from the processor itself, using the Coprocessor 14 (CP14) interface, or from an external debugger. The events are also supplied to the PTM and can be used for trigger or trace.

For more information about the PMU, refer to the *Performance Monitoring Unit* chapter of the *Cortex-A9 Technical Reference Manual*, available on the ARM website ([infocenter.arm.com](http://infocenter.arm.com)).

### Related Information

[ARM Infocenter \(www.infocenter.arm.com\)](http://www.infocenter.arm.com)

## MPCore Timers

There is one interval timer and one watchdog timer for each processor.

### Functional Description

Each timer is private, meaning that only its associated processor can access it. If the watchdog timer is not needed, it can be configured as a second interval timer.

Each private interval and watchdog timer has the following features:

- A 32-bit counter that optionally generates an interrupt when it reaches zero
- Configurable starting values for the counter
- An eight-bit prescaler value to qualify the clock period

### Implementation Details

The timers are configurable to either single-shot or auto-reload mode. The timer blocks are clocked by mpu\_periph\_clk, running at ¼ the rate of mpu\_clk.

For more information about private timers, refer to “About the private timer and watchdog blocks” in the *Global timer, Private timers, and Watchdog registers* chapter of the *Cortex-A9 MPCore Technical Reference Manual*, available on the ARM website ([infocenter.arm.com](http://infocenter.arm.com)).

### Related Information

[ARM Infocenter \(www.infocenter.arm.com\)](http://www.infocenter.arm.com)

## Generic Interrupt Controller

### Functional Description

The Generic Interrupt Controller (GIC) supports up to 180 interrupt sources, including dedicated peripherals and IP implemented in the FPGA fabric. In a dual-core system, the GIC is shared by both Cortex-A9 processors. Each processor also has 16 banked software-generated interrupts and 16 banked private peripheral interrupts.

## Implementation Details

The configuration and control for the GIC is memory-mapped and accessed through the SCU. The GIC are clocked by mpu\_periph\_clk, running at  $\frac{1}{4}$  the rate of mpu\_clk.

For more information about the GIC, refer to the *Interrupt Controller* chapter of the *Cortex-A9 MPCore Technical Reference Manual*, available on the ARM website ([infocenter.arm.com](http://infocenter.arm.com)).

### Related Information

- [GIC Interrupt Map for the Cyclone V SoC HPS](#) on page 6-12

The following table shows the interrupt map.

- [ARM Infocenter \(www.infocenter.arm.com\)](http://www.infocenter.arm.com)

### *GIC Interrupt Map for the Cyclone V SoC HPS*

The following table shows the interrupt map.

**Table 6-2: GIC Interrupt Map**

GIC Interrupt Number <sup>(3)</sup>	Source Block	Interrupt Name	Combined Interrupts	Triggering
32	CortexA9_0	cpu0_parityfail	<sup>(4)</sup>	Edge
33	CortexA9_0	cpu0_parityfail_BTAC	—	Edge
34	CortexA9_0	cpu0_parityfail_GHB	—	Edge
35	CortexA9_0	cpu0_parityfail_I_Tag	—	Edge
36	CortexA9_0	cpu0_parityfail_I_Data	—	Edge
37	CortexA9_0	cpu0_parityfail_TLB	—	Edge
38	CortexA9_0	cpu0_parityfail_D_Outer	—	Edge
39	CortexA9_0	cpu0_parityfail_D_Tag	—	Edge
40	CortexA9_0	cpu0_parityfail_D_Data	—	Edge
41	CortexA9_0	cpu0_deflags0	—	Level
42	CortexA9_0	cpu0_deflags1	—	Level
43	CortexA9_0	cpu0_deflags2	—	Level
44	CortexA9_0	cpu0_deflags3	—	Level

<sup>(3)</sup> To ensure that you are using the correct GIC interrupt number, your code should refer to the symbolic interrupt name, as shown in the **Interrupt Name** column. Symbolic interrupt names are defined in a header file distributed with the source installation for your operating system.

<sup>(4)</sup> This interrupt combines the interrupts named cpu0\_parityfail\_\*.

GIC Interrupt Number <sup>(3)</sup>	Source Block	Interrupt Name	Combined Interrupts	Triggering
45	CortexA9_0	cpu0_deflags4	—	Level
46	CortexA9_0	cpu0_deflags5	—	Level
47	CortexA9_0	cpu0_deflags6	—	Level
48	CortexA9_1	cpu1_parityfail	<sup>(5)</sup>	Edge
49	CortexA9_1	cpu1_parityfail_BTAC	—	Edge
50	CortexA9_1	cpu1_parityfail_GHB	—	Edge
51	CortexA9_1	cpu1_parityfail_I_Tag	—	Edge
52	CortexA9_1	cpu1_parityfail_I_Data	—	Edge
53	CortexA9_1	cpu1_parityfail_TLB	—	Edge
54	CortexA9_1	cpu1_parityfail_D_Outer	—	Edge
55	CortexA9_1	cpu1_parityfail_D_Tag	—	Edge
56	CortexA9_1	cpu1_parityfail_D_Data	—	Edge
57	CortexA9_1	cpu1_deflags0	—	Level
58	CortexA9_1	cpu1_deflags1	—	Level
59	CortexA9_1	cpu1_deflags2	—	Level
60	CortexA9_1	cpu1_deflags3	—	Level
61	CortexA9_1	cpu1_deflags4	—	Level
62	CortexA9_1	cpu1_deflags5	—	Level
63	CortexA9_1	cpu1_deflags6	—	Level
64	SCU	scu_parityfail0	—	Edge
65	SCU	scu_parityfail1	—	Edge
66	SCU	scu_ev_abort	—	Edge

<sup>(3)</sup> To ensure that you are using the correct GIC interrupt number, your code should refer to the symbolic interrupt name, as shown in the **Interrupt Name** column. Symbolic interrupt names are defined in a header file distributed with the source installation for your operating system.

<sup>(5)</sup> This interrupt combines the interrupts named cpu1\_parityfail\_\*.

GIC Interrupt Number <sup>(3)</sup>	Source Block	Interrupt Name	Combined Interrupts	Triggering
67	L2-Cache	l2_ecc_byte_wr_IRQ	—	Edge
68	L2-Cache	l2_ecc_corrected_IRQ	—	Edge
69	L2-Cache	l2_ecc_uncorrected_IRQ	—	Edge
70	L2-Cache	l2_combined_IRQ	<sup>(6)</sup>	Level
71	DDR	ddr_ecc_error_IRQ	—	Level
72	FPGA	FPGA_IRQ0	—	Level or Edge
73	FPGA	FPGA_IRQ1	—	Level or Edge
74	FPGA	FPGA_IRQ2	—	Level or Edge
75	FPGA	FPGA_IRQ3	—	Level or Edge
76	FPGA	FPGA_IRQ4	—	Level or Edge
77	FPGA	FPGA_IRQ5	—	Level or Edge
78	FPGA	FPGA_IRQ6	—	Level or Edge
79	FPGA	FPGA_IRQ7	—	Level or Edge
80	FPGA	FPGA_IRQ8	—	Level or Edge
81	FPGA	FPGA_IRQ9	—	Level or Edge
82	FPGA	FPGA_IRQ10	—	Level or Edge
83	FPGA	FPGA_IRQ11	—	Level or Edge
84	FPGA	FPGA_IRQ12	—	Level or Edge
85	FPGA	FPGA_IRQ13	—	Level or Edge
86	FPGA	FPGA_IRQ14	—	Level or Edge
87	FPGA	FPGA_IRQ15	—	Level or Edge
88	FPGA	FPGA_IRQ16	—	Level or Edge

<sup>(3)</sup> To ensure that you are using the correct GIC interrupt number, your code should refer to the symbolic interrupt name, as shown in the **Interrupt Name** column. Symbolic interrupt names are defined in a header file distributed with the source installation for your operating system.

<sup>(6)</sup> This interrupt combines the following interrupts: DECERRINTR, ECNTRINTR, ERRRDINTR, ERRRTINTR, ERRWDINTR, ERRWTINTR, PARRDINTR, PARRTINTR, and SLVERRINTR

GIC Interrupt Number <sup>(3)</sup>	Source Block	Interrupt Name	Combined Interrupts	Triggering
89	FPGA	FPGA_IRQ17	—	Level or Edge
90	FPGA	FPGA_IRQ18	—	Level or Edge
91	FPGA	FPGA_IRQ19	—	Level or Edge
92	FPGA	FPGA_IRQ20	—	Level or Edge
93	FPGA	FPGA_IRQ21	—	Level or Edge
94	FPGA	FPGA_IRQ22	—	Level or Edge
95	FPGA	FPGA_IRQ23	—	Level or Edge
96	FPGA	FPGA_IRQ24	—	Level or Edge
97	FPGA	FPGA_IRQ25	—	Level or Edge
98	FPGA	FPGA_IRQ26	—	Level or Edge
99	FPGA	FPGA_IRQ27	—	Level or Edge
100	FPGA	FPGA_IRQ28	—	Level or Edge
101	FPGA	FPGA_IRQ29	—	Level or Edge
102	FPGA	FPGA_IRQ30	—	Level or Edge
103	FPGA	FPGA_IRQ31	—	Level or Edge
104	FPGA	FPGA_IRQ32	—	Level or Edge
105	FPGA	FPGA_IRQ33	—	Level or Edge
106	FPGA	FPGA_IRQ34	—	Level or Edge
107	FPGA	FPGA_IRQ35	—	Level or Edge
108	FPGA	FPGA_IRQ36	—	Level or Edge
109	FPGA	FPGA_IRQ37	—	Level or Edge
110	FPGA	FPGA_IRQ38	—	Level or Edge
111	FPGA	FPGA_IRQ39	—	Level or Edge

<sup>(3)</sup> To ensure that you are using the correct GIC interrupt number, your code should refer to the symbolic interrupt name, as shown in the **Interrupt Name** column. Symbolic interrupt names are defined in a header file distributed with the source installation for your operating system.

GIC Interrupt Number <sup>(3)</sup>	Source Block	Interrupt Name	Combined Interrupts	Triggering
112	FPGA	FPGA_IRQ40	—	Level or Edge
113	FPGA	FPGA_IRQ41	—	Level or Edge
114	FPGA	FPGA_IRQ42	—	Level or Edge
115	FPGA	FPGA_IRQ43	—	Level or Edge
116	FPGA	FPGA_IRQ44	—	Level or Edge
117	FPGA	FPGA_IRQ45	—	Level or Edge
118	FPGA	FPGA_IRQ46	—	Level or Edge
119	FPGA	FPGA_IRQ47	—	Level or Edge
120	FPGA	FPGA_IRQ48	—	Level or Edge
121	FPGA	FPGA_IRQ49	—	Level or Edge
122	FPGA	FPGA_IRQ50	—	Level or Edge
123	FPGA	FPGA_IRQ51	—	Level or Edge
124	FPGA	FPGA_IRQ52	—	Level or Edge
125	FPGA	FPGA_IRQ53	—	Level or Edge
126	FPGA	FPGA_IRQ54	—	Level or Edge
127	FPGA	FPGA_IRQ55	—	Level or Edge
128	FPGA	FPGA_IRQ56	—	Level or Edge
129	FPGA	FPGA_IRQ57	—	Level or Edge
130	FPGA	FPGA_IRQ58	—	Level or Edge
131	FPGA	FPGA_IRQ59	—	Level or Edge
132	FPGA	FPGA_IRQ60	—	Level or Edge
133	FPGA	FPGA_IRQ61	—	Level or Edge
134	FPGA	FPGA_IRQ62	—	Level or Edge

<sup>(3)</sup> To ensure that you are using the correct GIC interrupt number, your code should refer to the symbolic interrupt name, as shown in the **Interrupt Name** column. Symbolic interrupt names are defined in a header file distributed with the source installation for your operating system.



GIC Interrupt Number <sup>(3)</sup>	Source Block	Interrupt Name	Combined Interrupts	Triggering
135	FPGA	FPGA_IRQ63	—	Level or Edge
136	DMA	dma_IRQ0	—	Level
137	DMA	dma_IRQ1	—	Level
138	DMA	dma_IRQ2	—	Level
139	DMA	dma_IRQ3	—	Level
140	DMA	dma_IRQ4	—	Level
141	DMA	dma_IRQ5	—	Level
142	DMA	dma_IRQ6	—	Level
143	DMA	dma_IRQ7	—	Level
144	DMA	dma_irq_abort	—	Level
145	DMA	dma_ecc_corrected_IRQ	—	Level
146	DMA	dma_ecc_uncorrected_IRQ	—	Level
147	EMAC0	emac0_IRQ	<sup>(7)</sup>	Level
148	EMAC0	emac0_tx_ecc_corrected_IRQ	—	Level
149	EMAC0	emac0_tx_ecc_uncorrected_IRQ	—	Level
150	EMAC0	emac0_rx_ecc_corrected_IRQ	—	Level
151	EMAC0	emac0_rx_ecc_uncorrected_IRQ	—	Level
152	EMAC1	emac1_IRQ	<sup>(7)</sup>	Level
153	EMAC1	emac1_tx_ecc_corrected_IRQ	—	Level
154	EMAC1	emac1_tx_ecc_uncorrected_IRQ	—	Level
155	EMAC1	emac1_rx_ecc_corrected_IRQ	—	Level
156	EMAC1	emac1_rx_ecc_uncorrected_IRQ	—	Level

<sup>(3)</sup> To ensure that you are using the correct GIC interrupt number, your code should refer to the symbolic interrupt name, as shown in the **Interrupt Name** column. Symbolic interrupt names are defined in a header file distributed with the source installation for your operating system.

<sup>(7)</sup> This interrupt combines sbd\_intr\_o, lpi\_intr\_o, and pmt\_intr\_o.

GIC Interrupt Number <sup>(3)</sup>	Source Block	Interrupt Name	Combined Interrupts	Triggering
157	USB0	usb0_IRQ	—	Level
158	USB0	usb0_ecc_corrected_IRQ	—	Level
159	USB0	usb0_ecc_uncorrected_IRQ	—	Level
160	USB1	usb1_IRQ	—	Level
161	USB1	usb1_ecc_corrected_IRQ	—	Level
162	USB1	usb1_ecc_uncorrected_IRQ	—	Level
163	CAN0	can0_sts_IRQ	—	Level
164	CAN0	can0_mo_IRQ	—	Level
165	CAN0	can0_ecc_corrected_IRQ	—	Level
166	CAN0	can0_ecc_uncorrected_IRQ	—	Level
167	CAN1	can1_sts_IRQ	—	Level
168	CAN1	can1_mo_IRQ	—	Level
169	CAN1	can1_ecc_corrected_IRQ	—	Level
170	CAN1	can1_ecc_uncorrected_IRQ	—	Level
171	SDMMC	sdmmc_IRQ	—	Level
172	SDMMC	sdmmc_porta_ecc_corrected_IRQ	—	Level
173	SDMMC	sdmmc_porta_ecc_uncorrected_IRQ	—	Level
174	SDMMC	sdmmc_portb_ecc_corrected_IRQ	—	Level
175	SDMMC	sdmmc_portb_ecc_uncorrected_IRQ	—	Level
176	NAND	nand_IRQ	—	Level
177	NAND	nandr_ecc_corrected_IRQ	—	Level
178	NAND	nandr_ecc_uncorrected_IRQ	—	Level
179	NAND	nandw_ecc_corrected_IRQ	—	Level

<sup>(3)</sup> To ensure that you are using the correct GIC interrupt number, your code should refer to the symbolic interrupt name, as shown in the **Interrupt Name** column. Symbolic interrupt names are defined in a header file distributed with the source installation for your operating system.

GIC Interrupt Number <sup>(3)</sup>	Source Block	Interrupt Name	Combined Interrupts	Triggering
180	NAND	nandw_ecc_uncorrected_IRQ	—	Level
181	NAND	nande_ecc_corrected_IRQ	—	Level
182	NAND	nande_ecc_uncorrected_IRQ	—	Level
183	QSPI	qspi_IRQ	—	Level
184	QSPI	qspi_ecc_corrected_IRQ	—	Level
185	QSPI	qspi_ecc_uncorrected_IRQ	—	Level
186	SPI0	spi0_IRQ	<sup>(8)</sup>	Level
187	SPI1	spi1_IRQ	<sup>(8)</sup>	Level
188	SPI2	spi2_IRQ	<sup>(8)</sup>	Level
189	SPI3	spi3_IRQ	<sup>(8)</sup>	Level
190	I2C0	i2c0_IRQ	<sup>(9)</sup>	Level
191	I2C1	i2c1_IRQ	<sup>(9)</sup>	Level
192	I2C2	i2c2_IRQ	<sup>(9)</sup>	Level
193	I2C3	i2c3_IRQ	<sup>(9)</sup>	Level
194	UART0	uart0_IRQ	—	Level
195	UART1	uart1_IRQ	—	Level
196	GPIO0	gpio0_IRQ	—	Level
197	GPIO1	gpio1_IRQ	—	Level
198	GPIO2	gpio2_IRQ	—	Level

<sup>(3)</sup> To ensure that you are using the correct GIC interrupt number, your code should refer to the symbolic interrupt name, as shown in the **Interrupt Name** column. Symbolic interrupt names are defined in a header file distributed with the source installation for your operating system.

<sup>(8)</sup> This interrupt combines the following interrupts: ssi\_txe\_intr, ssi\_txo\_intr, ssi\_rxf\_intr, ssi\_rxo\_intr, ssi\_rxu\_intr, and ssi\_mst\_intr.

<sup>(9)</sup> This interrupt combines the following interrupts: ic\_rx\_under\_intr, ic\_rx\_full\_intr, ic\_tx\_over\_intr, ic\_tx\_empty\_intr, ic\_rd\_req\_intr, ic\_tx\_abrt\_intr, ic\_rx\_done\_intr, ic\_activity\_intr, ic\_stop\_det\_intr, ic\_start\_det\_intr, and ic\_gen\_call\_intr.

GIC Interrupt Number <sup>(3)</sup>	Source Block	Interrupt Name	Combined Interrupts	Triggering
199	Timer0	timer_l4sp_0_IRQ	<sup>(10)</sup>	Level
200	Timer1	timer_l4sp_1_IRQ	<sup>(10)</sup>	Level
201	Timer2	timer_oscl_0_IRQ	<sup>(10)</sup>	Level
202	Timer3	timer_oscl_1_IRQ	<sup>(10)</sup>	Level
203	Watchdog0	wdog0_IRQ	—	Level
204	Watchdog1	wdog1_IRQ	—	Level
205	Clock manager	clkmgr_IRQ	—	Level
206	Clock manager	mpuwakeup_IRQ	—	Level
207	FPGA manager	fpga_man_IRQ	<sup>(11)</sup>	Level
208	CoreSight	nCTIIRQ[0]	—	Level
209	CoreSight	nCTIIRQ[1]	—	Level
210	On-chip RAM	ram_ecc_corrected_IRQ	—	Level
211	On-chip RAM	ram_ecc_uncorrected_IRQ	—	Level

**Related Information**

[Implementation Details](#) on page 6-12

**Global Timer**

The MPU features a global 64-bit, auto-incrementing timer, which is primarily used by the operating system.

<sup>(3)</sup> To ensure that you are using the correct GIC interrupt number, your code should refer to the symbolic interrupt name, as shown in the **Interrupt Name** column. Symbolic interrupt names are defined in a header file distributed with the source installation for your operating system.

<sup>(10)</sup> This interrupt combines TIMINT1 and TIMINT2.

<sup>(11)</sup> This interrupt combines the following interrupts: fpga\_man\_irq[7..0].

## Functional Description

The global timer is accessible by the processors using memory-mapped access through the SCU. The global timer has the following features:

- 64-bit incrementing counter with an auto-incrementing feature. It continues incrementing after sending interrupts.
- Memory-mapped in the private memory region.
- Accessed at reset in Secure State only. It can only be set once, but secure code can read it at any time.
- Accessible to both Cortex-A9 processors in the MPCore.

## Implementation Details

Each Cortex-A9 processor has a private 64-bit comparator that generates a private interrupt when the counter reaches the specified value. Each Cortex-A9 processor uses the banked ID, ID27, for this interrupt. ID27 is sent to the GIC as a Private Peripheral Interrupt (PPI).

The global timer are clocked by mpu\_periph\_clk, running at  $\frac{1}{4}$  the rate of mpu\_clk.

For more information about the global timer, refer to “About the Global Timer” in the *Global timer, Private timers, and Watchdog registers* chapter of the Cortex-A9 MPCore Technical Reference Manual, available on the ARM website (infocenter.arm.com).

### Related Information

[ARM Infocenter \(www.infocenter.arm.com\)](http://www.infocenter.arm.com)

## Snoop Control Unit

The SCU manages data traffic for the Cortex-A9 processors and the memory system, including the L2 cache. In a multi-master system, the processors and other masters can operate on shared data. The SCU ensures that each processor operates on the most up-to-date copy of data, maintaining cache coherency.

## Functional Description

The SCU is used to connect the Cortex-A9 processors and the ACP to the L2 cache controller. The SCU performs the following functions:

- When the processors are set to SMP mode, the SCU maintains data cache coherency between the processors.

**Note:** The SCU does not maintain coherency of the instruction caches.

- Initiates L2 cache memory accesses
- Arbitrates between processors requesting L2 access
- Manages ACP access with cache coherency capabilities.

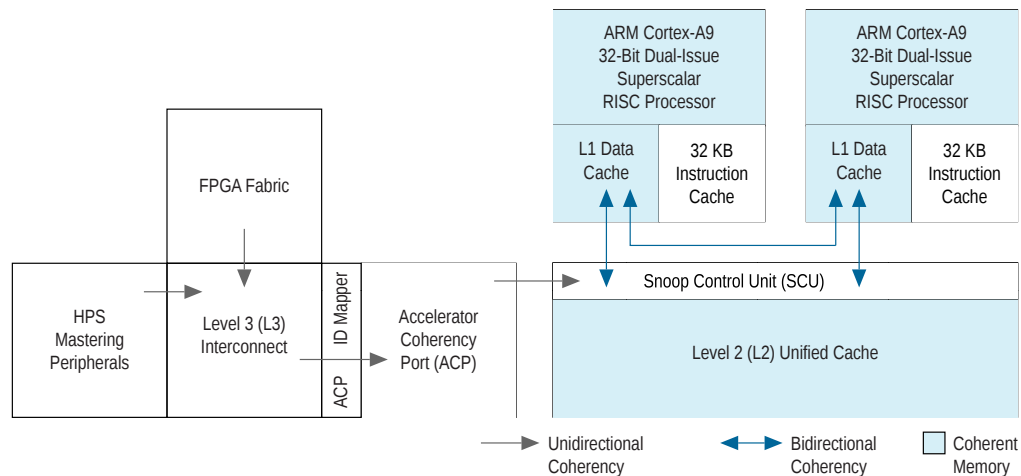
For more information about the SCU, refer to the *Snoop Control Unit* chapter of the *Cortex-A9 MPCore Technical Reference Manual*, available on the ARM website (infocenter.arm.com).

### Related Information

[ARM Infocenter \(www.infocenter.arm.com\)](http://www.infocenter.arm.com)

**Coherent Memory, Snoop Control Unit, and Accelerator Coherency Port**

The SCU in a dual-processor system, illustrating the flow of data among the L1 data caches and the SCU.

**Related Information**

[Accelerator Coherency Port](#) on page 6-22

**Implementation Details**

When the processor writes to any coherent memory location, the SCU ensures that the relevant data is coherent (updated, tagged, or invalidated). Similarly, the SCU monitors read operations from a coherent memory location. If the required data is already stored within the other processor's L1 cache, the data is returned directly to the requesting processor. If the data is not in L1 cache, the SCU issues a read to the L2 cache. If the data is not in the L2 cache memory, the read is finally forwarded to main memory. The primary goal is to minimize power consumption and maximize overall memory performance.

The SCU maintains bidirectional coherency between the L1 data caches belonging to the processors. When one processor writes to a location in its L1 cache, if the same location is cached in the other L1 cache, the SCU updates it.

Non-coherent data passes through as a standard read or write operation.

The SCU also arbitrates between the Cortex-A9 processors if both attempt simultaneous access to the L2 cache, and manages accesses from the ACP.

**Accelerator Coherency Port**

The ACP allows peripherals—including FPGA-based peripherals—to maintain data coherency with the Cortex-A9 MPCore processors and the SCU. Dedicated peripherals in the HPS, and those built in FPGA logic, access the coherent memory through the ACP ID mapper and the ACP.

The high-bandwidth peripherals, including the FPGA data ports, connect to the L3 interconnect.

**Related Information**

- [ACP ID Mapper](#) on page 6-24
- [Coherent Memory, Snoop Control Unit, and Accelerator Coherency Port](#) on page 6-22

## Burst Sizes and Byte Strokes

The ACP improves system performance for hardware accelerators in the FPGA fabric. However, in order to achieve high levels of performance, you must use the one of the recommended burst types. The other burst types have significantly lower performance.

### Related Information

[Recommended Burst Types](#) on page 6-23

### *Recommended Burst Types*

Table 6-3: Recommended Burst Types

Burst Type	Beats	Width (Bits)	Address Type	Byte Strokes
Wrapping	4	64	64-bit aligned	Asserted
Incrementing	4	64	32-bit aligned	Asserted

### Related Information

[Burst Sizes and Byte Strokes](#) on page 6-23

### *Burst Guidelines*

**Note:** If the slave port of the FPGA-to-HPS bridge is not 64 bits wide, you must supply bursts to the FPGA-to-HPS bridge that are upsized or downsized to the burst types above. For example, if the slave data width of the FPGA-to-HPS bridge is 32 bits, then bursts of eight beats by 32 bits are required to access the ACP efficiently.

**Caution:** If the address and burst size of the transaction to the ACP matches either of the conditions above, the logic in the MPU assumes the transaction has all its byte strokes set. If the byte strokes are not all set, then the write does not actually overwrite all the bytes in the word. Instead, the cache assumes the whole cache line is valid. If this line is dirty (and therefore gets written out to SDRAM), data corruption might occur.

## Exclusive and Locked Accesses

The ACP does not support exclusive accesses to coherent memory. The ACP supports exclusive accesses to non-coherent memory; however, it is important that the exclusive access transaction is not affected by the upsizing and downsizing logic of the FPGA-to-HPS bridge or the L3 interconnect. If the exclusive access is broken into multiple transactions due to the sizing logic, the exclusive access bit is cleared by the bridge or interconnect and the exclusive access fails.

**Note:** Altera recommends that exclusive accesses bypass the ACP altogether, either through the 32-bit slave port of the SDRAM controller connected directly to the L3 interconnect or through the FPGA-to-SDRAM interface.

For more information about the exclusive access support of the SDRAM controller subsystem, refer to the *SDRAM Controller Subsystem* chapter in the *Cyclone V Device Handbook, Volume 3*.

The ACP ID mapper does not support locked accesses. To ensure mutually exclusive access to shared data, use the exclusive access support built into the SDRAM controller.

**Related Information****SDRAM Controller Subsystem**

For more information, refer to the *SDRAM Controller Subsystem* chapter in the *Cyclone V Device Handbook, Volume 3*.

## ACP ID Mapper

The ACP ID mapper is situated between the level 3 (L3) interconnect and the MPU subsystem ACP slave. It is responsible for mapping 12-bit Advanced Microcontroller Bus Architecture (AMBA®) Advanced eXtensible Interface (AXI™) IDs (input IDs) from the L3 interconnect to 3-bit AXI IDs (output IDs) supported by the ACP slave port.

The ACP ID mapper also implements a 1 GB coherent window into 4 GB address space.

### Functional Description

The ACP slave supports up to six masters. However, custom peripherals implemented in the FPGA fabric can have a larger number of masters that need to access the ACP slave. The ACP ID mapper allows these masters to access the ACP.

The ACP ID mapper resides between the interconnect and the ACP slave of the MPU subsystem. It has the following characteristics:

- Support for up to six concurrent ID mappings
- 1 GB coherent window into 4 GB MPCore address space
- Remaps the 5-bit user sideband signals used by the Snoop Control Unit (SCU) and L2 cache.

For more information about AXI user sideband signals, refer to the CoreLink Level 2 Cache Controller L2C-310 Technical Reference Manual, which you can download from the ARM website ([infocenter.arm.com](http://infocenter.arm.com)).

**Related Information**

**ARM Infocenter ([www.infocenter.arm.com](http://www.infocenter.arm.com))**

### Implementation Details

The ACP is accessed by masters that require access to coherent memory. The ACP slave port can be accessed by the master peripherals of the L3 interconnect, as well as by masters implemented in the FPGA fabric (via the FPGA-to-HPS bridge).

The ACP ID mapper supports the following ID mapping modes:

- Dynamic mapping
- Fixed mapping

Software can select the ID mapping on a per-ID basis. For input IDs that are configured for fixed mapping, there is a one-to-one mapping from input IDs to output IDs. When an input ID is configured for dynamic mapping, it is automatically mapped to an available output ID. The dynamic mode is more flexible because the hardware handles the mapping. The hardware mapping allows you to use one output ID for more than one input ID. Output IDs are assigned to input IDs on a first-come, first-served basis.

Out of the total of eight output IDs, only six are available to masters of the L3 interconnect. The first two output IDs (0 and 1) are dedicated to the Cortex-A9 processor cores in the MPU subsystem, leaving the last six output IDs (2-7) available to the ACP ID mapper. Output IDs 2-6 support fixed and dynamic modes of operation while output ID 7 supports dynamic only.



The operating modes are programmable through accesses to the control and status registers in the ACP ID mapper, available via the level 4 peripheral bus connection. At reset time, the ACP ID mapper defaults to dynamic ID mapping for all output IDs except ID 2, which resets to a fixed mapping for the Debug Access Port (DAP) input ID.

#### Related Information

[Cortex-A9 MPU Subsystem with L3 Interconnect](#) on page 6-2

### ID Intended Usage

**Table 6-4** summarizes the expected usage of the 3-bit output IDs, and their settings at reset.

**Table 6-4: ID Intended Usage**

Output ID	Reset State	Intended Use
7	Dynamic	Dynamic mapping only
6	Dynamic	Fixed or dynamic, programmed by software.
5		
4		
3		
2	Fixed at 0x001 (DAP)	Assigned to the input ID of the DAP at reset. After reset, can be either fixed or dynamic, programmed by software.
1	—	Not used by the ACP ID Mapper
0		

### AXI User Sideband Override

For masters that cannot drive the AXI user sideband signal of incoming transactions, the ACP ID mapper can control overriding this signal. The ACP ID mapper can also control which 1 GB coherent window into memory is accessed by masters of the L3 interconnect. Each fixed mapping can be assigned a different user sideband signal and memory window to allow specific settings for different masters. All dynamic mappings share a common user sideband signal and memory window setting.

### Transaction Capabilities

At any one time, the ACP ID mapper can accept and issue up to 15 transactions per ID mapping. Read and write ID mappings are managed in separate lists, allowing more unique input IDs to be remapped at any given time. If a master issues a series of reads and writes with the same input ID, there are no ordering restrictions.

Because there are only six output IDs available, there can be no more than six read and six write transactions with unique IDs in progress at any one time. The write acceptance of the ACP slave is five transactions, and the read acceptance is 13 transactions. Only four coherent read transactions per ID mapping can be outstanding at one time.

## Dynamic Mapping Mode

In dynamic mode, every unique input ID that is received from the L3 master port is assigned to an unused output ID. The new output ID is applied to the transaction as it is issued to the ACP slave of the SCU. Any transaction that arrives to the ACP ID mapper with an input ID that matches an already-in-progress transaction is mapped to the same output ID. Once all transactions on an ID mapping have completed, that output ID is released and can be used again for other input IDs.

## Fixed Mapping Mode

In fixed mode, output IDs 2 through 6 can be assigned by software to a specific 12-bit input ID. This ability makes it possible to use the lock-by-master feature of the L2 cache controller, because the input transaction ID from the master is always assigned to a specific output ID. Unlike dynamic mode, ID 7 is not available for fixed mapping because it is reserved for dynamic mode only to avoid system deadlocks.

The ACP ID mapper has two banks of registers to control the behavior of the mappings, namely, a request bank and a read-only status bank. Both banks contain the same number of registers. To change the settings for a particular mapping (either a specific fixed ID, or all dynamic mappings), software should write to the appropriate register in the request bank. The hardware examines the request, and only applies the change when safe to do so, which is when there are no outstanding transactions with the output ID. When the change is applied, the status register is updated. Software should check that the change has actually taken place by polling the corresponding status register.

## HPS Peripheral Master Input IDs

**Table 6-5: HPS Peripheral Master Input IDs**

The input IDs issued from the interconnect for each HPS peripheral master that can access the ACP ID mapper

Interconnect Master	ID <sup>(12)</sup>
DMA	00000xxxx011
EMAC0	10000xxxx001
EMAC1	10000xxxx010
USB0	100000000011
USB1	100000000110
NAND	1xxxxxxxx100
ETR	100000000000
DAP	000000000001
SD/MMC	100000000101
FPGA-to-HPS bridge	0xxxxxxxx100

### Control of the AXI User Sideband Signals

The ACP ID mapper module allows control of the AXI user sideband signal values. Not all masters drive these signals, so the ACP ID mapper makes it possible to drive the 5-bit user sideband signal with either a default value (in dynamic mode) or specific values (in fixed mode).

There are registers available to configure the default values of the user sideband signals for all transactions, and fixed values of these signals for particular transactions in fixed mapping mode. In dynamic mode, the user sideband signals of incoming transactions are mapped with the default values stored in the register. In fixed mapping mode, the input ID of the transaction is mapped to the 3-bit output ID and the user sideband signals of the transaction are mapped with the values stored in the register that corresponds to the output ID. One important exception, however, is that the ACP ID mapper always allows user sideband signals from the FPGA-to-HPS bridge to pass through to the ACP regardless of the user sideband value associated with the ID.

<sup>(12)</sup> Values are in binary. The letter *x* denotes variable ID bits each master passes with each transaction.

## Memory Region Remap

The ACP ID mapper has 1 GB of address space, which is by default a view into the bottom 1 GB of SDRAM. The mapper also allows transactions to be routed to different 1 GB-sized memory regions, called pages, in both dynamic and fixed modes. The two most significant bits of incoming 32-bit AXI address signals are replaced with the 2-bit user-configured address page decode information. The page decoder uses the values shown in [Table 6-6](#).

**Table 6-6: Page Decoder Values**

Page	Address Range
0	0x00000000–0x3FFFFFFF
1	0x40000000–0x7FFFFFFF
2	0x80000000–0xBFFFFFFF
3	0xC0000000–0xFFFFFFFF

With this page decode information, a master can read or write to any 1 GB region of the 4 GB memory space while maintaining cache coherency with the MPU subsystem.

Using this feature, a debugger can have a coherent view into main memory, without having to stop the processor. For example, at reset the DAP input ID (0x001) is mapped to output ID 2, so the debugger can vary the 1 GB window that the DAP accesses without affecting any other traffic flow to the ACP.

## L2 Cache

The MPU subsystem includes a secondary 512 KB L2 shared, unified cache memory.

### Functional Description

The L2 cache is much larger than the L1 cache. The L2 cache has significantly lower latency than external memory. The L2 cache is up to eight-way associative, configurable down to one-way (direct mapped). Like the L1 cache, the L2 cache can be locked by cache line, locked by way, or locked by bus master.

The L2 cache implements error correction codes (ECCs) and ECC error reporting. The cache can report a number of events to the processor and operating system.

### Cache Controller Configuration

The L2 cache consists of the ARM L2C-310 L2 cache controller configured as follows:

- 512 KB total memory
- Eight-way associativity
- Physically addressed, physically tagged
- Line length of 32 bytes
- Critical first word linefills
- Support for all AXI cache modes, as shown in [Table 6-7](#).

- Single event upset (SEU) protection
  - Parity on Tag RAM
  - ECC on L2 Data RAM

For more information about SEU errors, refer to the *System Manager* chapter in the *Cyclone V Device Handbook, Volume 3*.

- Two slave ports mastered by the SCU
- Two master ports connected to the following slave ports:
  - SDRAM controller, 64 bit slave port width
  - L3 interconnect, 64 bit slave port width
- Cache lockdown capabilities as follows:
  - Line lockdown
  - Lockdown by way
  - Lockdown by master (both processors and ACP masters)
- TrustZone support
- Cache event monitoring.

**Table 6-7: AXI Cache Mode Support**

Cache Mode
Write-through <sup>(13)</sup>
Write-back <sup>(13)</sup>
Read allocate
Write allocate
Read and write allocate

#### Related Information

- [L2 Cache Event Monitoring](#) on page 6-31
- [System Manager](#)  
For more information, refer to the *System Manager* chapter in the *Cyclone V Device Handbook, Volume 3*.

## L2 Cache Address Filtering

The L2 cache can access either the L3 interconnect fabric or the SDRAM. The L2 cache address filtering determines how much address space is allocated to the HPS-to-FPGA bridge and how much is allocated to SDRAM, depending on the configuration of the memory management unit.

*Memory Management Unit* describes how the address space is set based on L2 cache address filtering.

<sup>(13)</sup> Restrictions exist when using ECCs. For more information about SEU protection, refer to the *System Manager* chapter in the *Cyclone V Device Handbook, Volume 3*.

**Related Information**

- [Cortex-A9 MPU Subsystem with L3 Interconnect](#) on page 6-2
- [Memory Management Unit](#) on page 6-8

**ECC Support**

The L2 cache has the option of using ECCs to protect against SEU errors in the cache RAM.

Enabling ECCs does not affect the performance of the L2 cache. The ECC bits are calculated only for writes to the data RAM that are 64 bits wide (8 bytes, or one-quarter of the cache line length). The ECC logic does not perform a read-modify-write when calculating the ECC bits. The ECC protection bits are not valid in the following cases:

- Data is written that is not 64-bit aligned in memory
- Data is written that is less than 64 bits in width

In these cases the Byte Write Error interrupt is asserted. Cache data is still written when such an error occurs. However, the ECC error detection and correction continues to function. Therefore, the cache data is likely to be incorrect on subsequent reads.

To use ECCs, the software and system must meet the following requirements:

- L1 and L2 cache must be configured as write-back allocate for any cacheable memory region
- FPGA soft IP using the ACP must only perform the following types of data writes:
  - 64-bit aligned in memory
  - 64 bit wide accesses

For more information about SEU errors, refer to the *System Manager* chapter in the *Cyclone V Device Handbook, Volume 3*.

**Related Information**[System Manager](#)

For more information, refer to the *System Manager* chapter in the *Cyclone V Device Handbook, Volume 3*.

**Implementation Details**

[Table 6-8](#) shows the parameter settings for the cache controller.

**Table 6-8: Cache Controller Configuration**

Feature	Meaning
Cache way size	64 KB
Number of cache ways	8 ways
Tag RAM write latency	1
Tag RAM read latency	1
Tag RAM setup latency	1

Feature	Meaning
Data RAM write latency	1
Data RAM read latency	2
Data RAM setup latency	1
Parity logic	Parity logic enabled
Lockdown by master	Lockdown by master enabled
Lockdown by line	Lockdown by line enabled
AXI ID width on slave ports	6 AXI ID bits on slave ports
Address filtering	Address filtering logic enabled
Speculative read	Logic for supporting speculative read enabled
Presence of ARUSERMx and AWUSERMx sideband signals	Sideband signals enabled

For further information about cache controller configurable options, refer to the CoreLink Level 2 Cache Controller L2C-310 Technical Reference Manual, available on the ARM website (infocenter.arm.com).

#### Related Information

[ARM Infocenter \(www.infocenter.arm.com\)](http://www.infocenter.arm.com)

## L2 Cache Lockdown Capabilities

The L2 cache has three methods to lock data in the cache RAMs:

- Lockdown by line—Used to lock lines in the cache. This is commonly used for loading critical sections of software into the cache temporarily.
- Lockdown by way—Allows any or all of the eight cache ways to be locked. This is commonly used for loading critical data or code into the cache.
- Lockdown by master—Allows cache ways to be dedicated to a single master port. This allows a large cache to look like smaller caches to multiple master ports. The L2 cache can be mastered by CPU0, CPU1, or the six ACP masters, for a total of eight possible master ports.

For more information about L2 cache lockdown capabilities, refer to “Cache operation” in the *Functional Overview* chapter of the *CoreLink Level 2 Cache Controller L2C-310 Technical Reference Manual*, available on the ARM website (infocenter.arm.com).

#### Related Information

[ARM Infocenter \(www.infocenter.arm.com\)](http://www.infocenter.arm.com)

## L2 Cache Event Monitoring

The L2 cache supports the built-in cache event monitoring signals shown in [Table 6-9](#). The L2 cache can count two of the events at any one time.

Table 6-9: L2 Cache Events

Event	Description
CO	Eviction (cast out) of a line from the L2 cache.
DRHIT	Data read hit in the L2 cache.
DRREQ	Data read lookup to the L2 cache. Subsequently results in a hit or miss.
DWHIT	Data write hit in the L2 cache.
DWREQ	Data write lookup to the L2 cache. Subsequently results in a hit or miss.
DWTREQ	Data write lookup to the L2 cache with write-through attribute. Subsequently results in a hit or miss.
EPFALLOC	Prefetch hint allocated into the L2 cache.
EPFHIT	Prefetch hint hits in the L2 cache.
EPFRCVDS0	Prefetch hint received by slave port S0.
EPFRCVDS1	Prefetch hint received by slave port S1.
IPFALLOC	Allocation of a prefetch generated by L2 cache controller into the L2 cache.
IRHIT	Instruction read hit in the L2 cache.
IRREQ	Instruction read lookup to the L2 cache. Subsequently results in a hit or miss.
SPNIDEN	Secure privileged non-invasive debug enable.
SRCONFS0	Speculative read confirmed in slave port S0.
SRCONFS1	Speculative read confirmed in slave port S1.
SRRCVDS0	Speculative read received by slave port S0.
SRRCVDS1	Speculative read received by slave port S1.
WA	Allocation into the L2 cache caused by a write, with write-allocate attribute, miss.

For more information about the built-in L2 event monitoring capability, refer to “Implementation details” in the *Functional Overview* chapter of the *CoreLink Level 2 Cache Controller L2C-310 Technical Reference Manual*, available on the ARM website ([infocenter.arm.com](http://infocenter.arm.com)).

In addition, the L2 cache events can be captured and timestamped using dedicated debugging circuitry.



For more information about L2 event capture, refer to the *Debug* chapter of the Cortex-A9 MPCore Technical Reference Manual, available on the ARM website ([infocenter.arm.com](http://infocenter.arm.com)).

#### Related Information

[ARM Infocenter \(www.infocenter.arm.com\)](http://www.infocenter.arm.com)

## Debugging Modules

The MPU subsystem includes debugging resources through ARM CoreSight on-chip debugging and trace. The following functionality is included:

- Individual program trace for each processor
- Event trace for the Cortex-A9 MPCore
- Cross triggering between processors and other HPS debugging features

### Program Trace

Each processor has an independent PTM that provides real-time instruction flow trace. The PTM is compatible with a number of third-party debugging tools.

The PTM provides trace data in a highly compressed format. The trace data includes tags for specific points in the program execution flow, called waypoints. Waypoints are specific events or changes in the program flow.

The PTM recognizes and tags the waypoints listed in [Table 6-10](#).

**Table 6-10: Waypoints Supported by the PTM**

Type	Additional Waypoint Information
Indirect branches	Target address and condition code
Direct branches	Condition code
Instruction barrier instructions	—
Exceptions	Location where the exception occurred
Changes in processor instruction set state	—
Changes in processor security state	—
Context ID changes	—
Entry to and return from debug state when Halting debug mode is enabled	—

The PTM optionally provides additional information for waypoints, including the following.

- Processor cycle count between waypoints
- Global timestamp values

For information about global timestamps, refer to the *CoreSight Debug and Trace* chapter in the *Cyclone V Device Handbook, Volume 3*.

- Target addresses for direct branches

For more information about the PTM, refer to the CoreSight PTM-A9 Technical Reference Manual, available on the ARM website ([infocenter.arm.com](http://infocenter.arm.com)).

#### Related Information

- **CoreSight Debug and Trace**  
For more information, refer to the *CoreSight Debug and Trace* chapter in the *Cyclone V Device Handbook, Volume 3*.
- **ARM Infocenter ([www.infocenter.arm.com](http://www.infocenter.arm.com))**

## Event Trace

Events from each processor can be used as inputs to the PTM. The PTM can use these events as trace and trigger conditions.

For more information about the trigger and trace capabilities, refer to the CoreSight PTM-A9 Technical Reference Manual, Revision r1p0, available on the ARM website ([infocenter.arm.com](http://infocenter.arm.com)).

#### Related Information

- **Performance Monitoring Unit** on page 6-11
- **ARM Infocenter ([www.infocenter.arm.com](http://www.infocenter.arm.com))**

## Cross-Triggering

The PTM can export trigger events and perform actions on trigger inputs. The cross-trigger signals interface with other HPS debugging components including the FPGA fabric. Also, a breakpoint in one processor can trigger a break in the other.

For detailed information about cross-triggering and about debugging hardware in the MPU, refer to the *CoreSight Debug and Trace* chapter in the *Cyclone V Device Handbook, Volume 3*.

#### Related Information

##### **CoreSight Debug and Trace**

For more information, refer to the *CoreSight Debug and Trace* chapter in the *Cyclone V Device Handbook, Volume 3*.

## Cortex-A9 MPU Subsystem Register Implementation

The following configurations are available through registers in the Cortex-A9 subsystem:

- All processor-related controls, including the MMU and L1 caches, are controlled using the Coprocessor 15 (CP15) registers of each individual processor.
- All SCU registers, including control for the timers and GIC, are memory map accessible
- All L2 cache registers are memory-mapped.

For an address map of peripheral slave ports, including the SCU and L2 cache, refer to the *Introduction to the Hard Processor System* chapter in the *Cyclone V Device Handbook, Volume 3*. For detailed definitions of the registers for the Altera Cortex-A9 MPU subsystem, refer to the Cortex-A9 MPCore Technical Reference Manual, and the CoreLink Level 2 Cache Controller L2C-310 Technical Reference Manual, Revision r3p2, available on the ARM website (infocenter.arm.com).

### Related Information

- [Introduction to the Hard Processor System](#)  
For more information, refer to the *Introduction to the Hard Processor System* chapter in the *Cyclone V Device Handbook, Volume 3*.
- [ARM Infocenter \(www.infocenter.arm.com\)](http://www.infocenter.arm.com)

## Document Revision History

Date	Version	Changes
December 2013	2013.12.30	Correct SDRAM region address in MPCore Address Map
November 2012	1.2	Minor updates.
May 2012	1.1	<ul style="list-style-type: none"><li>• Add description of the ACP ID mapper</li><li>• Consolidate redundant information</li></ul>
January 2012	1.0	Initial release.