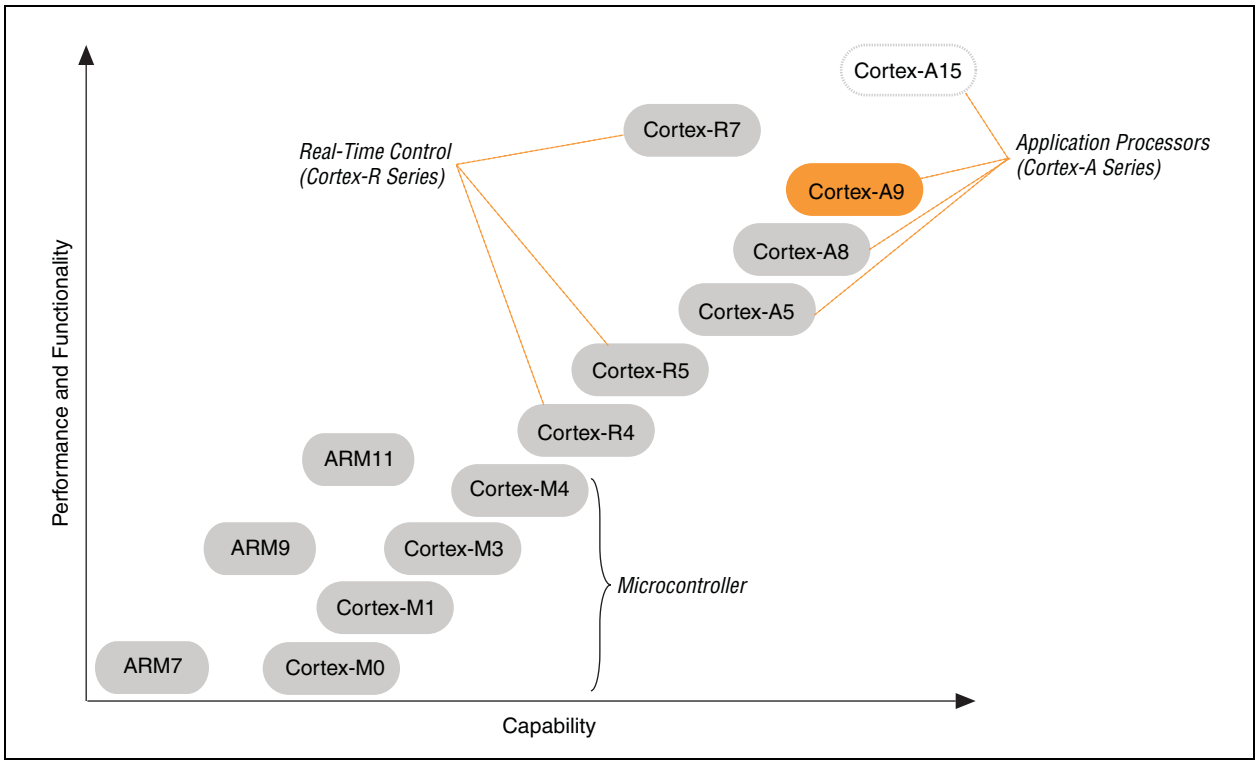
SoC FPGA ARM Cortex-A9 MPCore Processor Advance Information Brief

This document describes the dual-core ARM® Cortex™-A9 MPCore™ processor integrated in the hard processor system (HPS) of the Altera Cyclone® V and Arria® V system on a chip (SoC) FPGAs. This innovative HPS contains a microprocessor unit (MPU) with a dual-core ARM Cortex-A9 MPCore 32-bit application-class processor, memory controllers, and a rich set of system peripherals, hardened in Altera's most advanced 28-nm FPGA fabric. These SoC FPGAs provide the performance, power, and cost savings of hard logic, with the flexibility and time-to-market benefits of programmable logic.

ARM Cortex-A9 MPCore Processor Architecture

ARM processors are the standard in embedded systems. Altera SoC FPGAs leverage one of ARM's latest, high-performance Cortex-A9 processor architectures. The Cortex-A9 architecture provides industry-leading performance, the latest ARM features and capabilities, and is widely deployed in products ranging from wireless handsets to tablet computers. Figure 1 shows the progression of ARM processor performance and capability.

Figure 1. ARM Processor Family Lineup



101 Innovation Drive
San Jose, CA 95134
www.altera.com

Subscribe

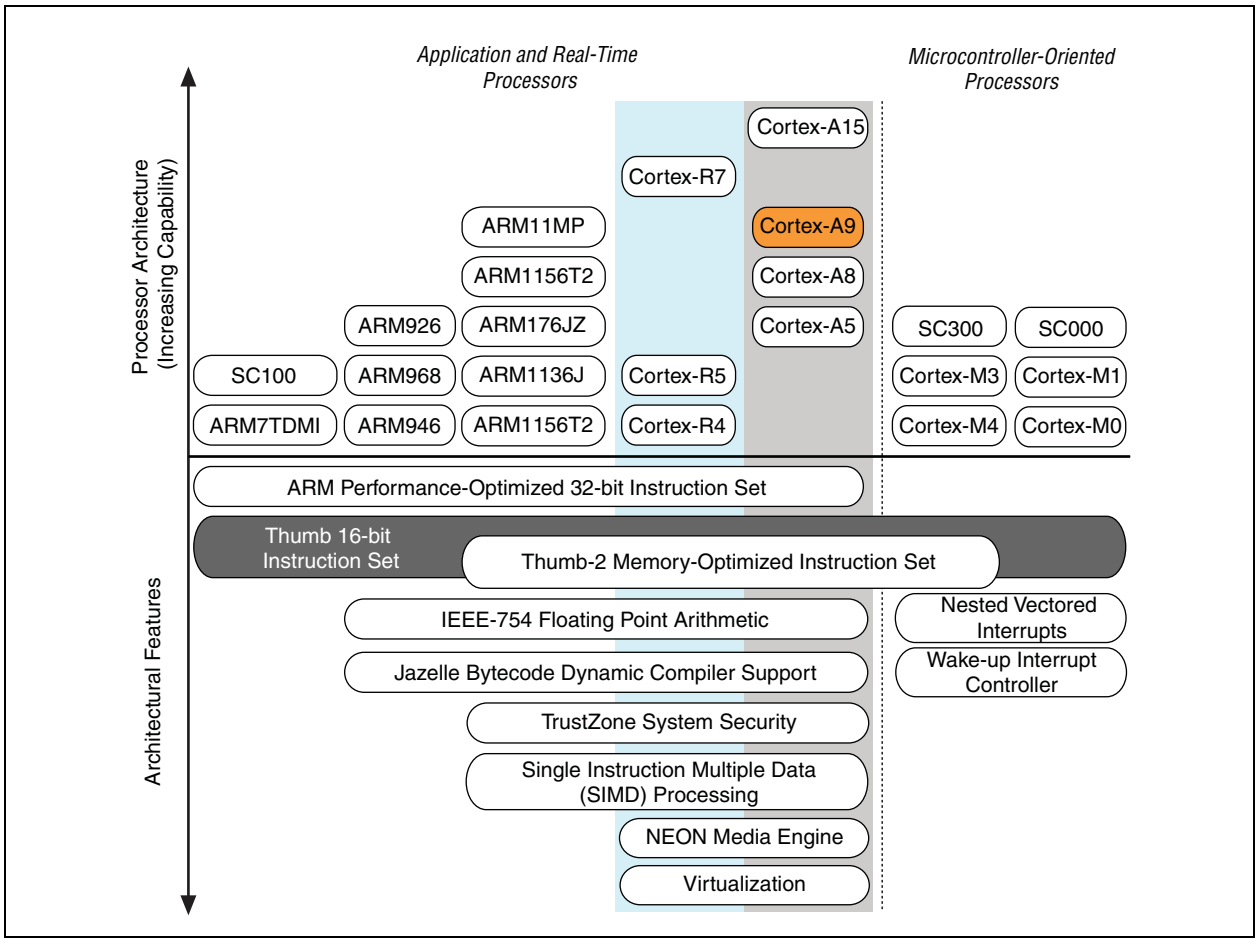
The dual-core ARM Cortex-A9 MPCore processor in Altera SoC FPGAs is designed for maximum performance and power efficiency, implementing the widely-supported ARMv7 instruction set architecture to address a broad range of industrial, automotive, and wireless applications. The Cortex-A9 MPCore processor architecture includes the following features:

- Dual-core multiprocessing, supporting both symmetric multiprocessing (SMP) and asymmetric multiprocessing (AMP)
- Multi-issue superscalar, out-of-order, speculative execution 8-stage pipeline delivering 2.5 DMIPS/MHz per CPU
- Advanced branch prediction
- Single- and double-precision IEEE standard 754-1985 floating-point mathematical operations
- ARM NEON™ 128-bit single instruction multiple data (SIMD) media processing engine
- Jazelle® byte-code dynamic compiler support
- TrustZone® architecture for enhanced system security

For more details about ARM Cortex-A9 processors, refer to the *ARM Cortex-A9 Processors* white paper.

As shown in Figure 2, the Cortex-A9 processor architecture supports the ARM performance-optimized instruction set and the latest memory-optimized Thumb®-2 mixed instruction set.

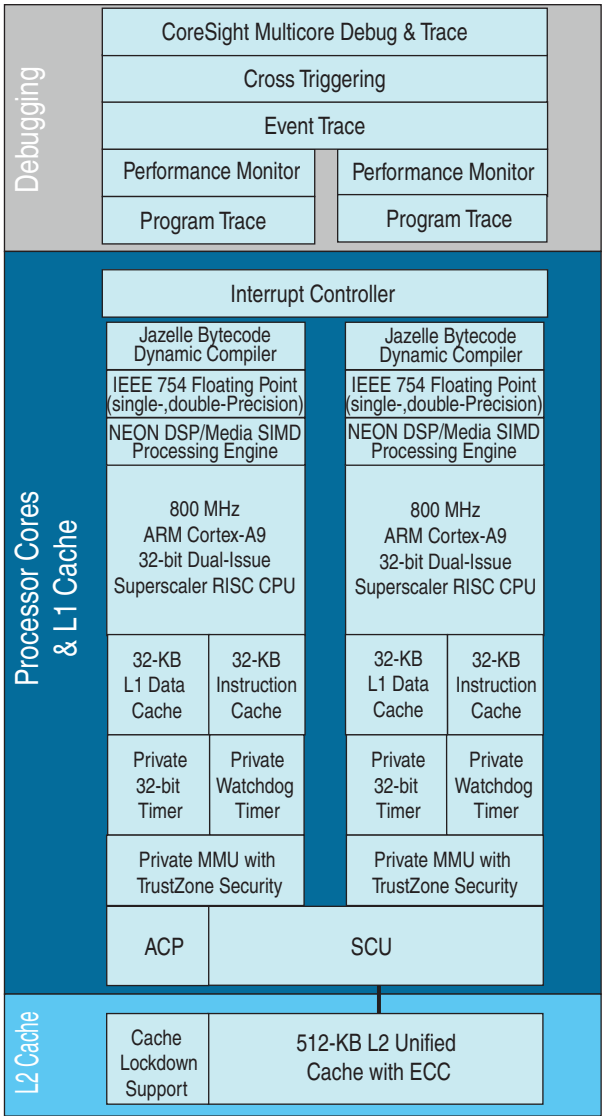
Figure 2. ARM Processor Family Architectural Features



The Thumb-2 instruction set optimizes processing performance in systems with narrow memory data paths and improves energy efficiency. On average, Thumb-2 code has a 31% smaller memory footprint and runs 38% faster than the original Thumb instruction set.

Figure 3 provides a detailed block diagram of the MPU subsystem. The MPU subsystem includes two Cortex-A9 processor cores, the level 2 (L2) cache and memory subsystem, Snoop Control Unit (SCU), Accelerator Coherency Port (ACP), and debug functions.

Figure 3. MPU Subsystem



Cache Memory

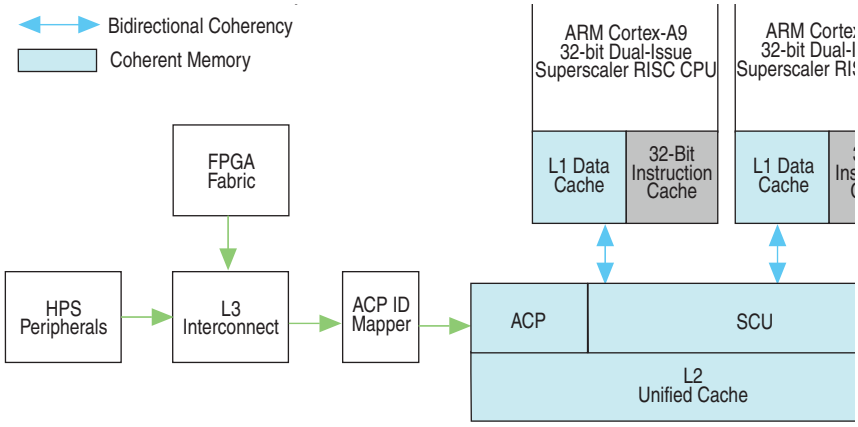
Cache memory improves the performance of a processor-based system and helps reduce power consumption. Cache memory that is tightly integrated with an associated processor core is called level 1 (L1) cache. Each Cortex-A9 CPU has two independent 32-kilobyte (KB) L1 caches—one for instructions and one for data—allowing simultaneous instruction fetches and data access. Each L1 cache is 4-way set associative and has an eight-word line length.

The HPS also includes a 512-KB L2 shared, unified cache memory (instruction and data for both Cortex-A9 cores). The L2 cache is 8-way set-associative with programmable locking by line, way, and master. The L2 cache includes error correction code (ECC) reporting.

Snoop Control Unit (SCU)

The snoop control unit (SCU) is an integral part of the cache memory systems and manages data traffic for the two Cortex-A9 CPUs and the memory system, including the L2 cache. In a multiprocessor system, each CPU may operate on shared data. The SCU ensures that each processor core operates on the most up-to-date copy of data, maintaining cache coherency. Figure 4 shows the coherent memory, SCU, and ACP.

Figure 4. Coherent Memory, SCU, and ACP



The SCU maintains bidirectional coherency among the L1 data caches ensuring both CPUs access to the most recent data. When a CPU writes to any coherent memory location, the SCU ensures that the relevant data is coherent (updated/tagged/invalidated). Similarly, the SCU monitors read operations from a coherent memory location. If the required data is already stored in the L1 caches, the data is returned directly to the requesting CPU. If the data is not in the L1 cache, the L2 cache checks its contents before the data is finally retrieved from the main memory.

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

Accelerator Coherency Port

The ACP allows level 3 (L3) interconnect masters—such as Ethernet media access controller (EMAC), direct memory access (DMA), and FPGA-to-HPS bridge—to share data coherently with the MPU subsystem. With the ACP, read accesses to coherent memory regions always return the most current data, whether in L1 cache, L2 cache, or main memory. Similarly, write operations to coherent memory regions cause the SCU to force coherence before the data is forwarded to the memory system.

The ACP ID mapper is located between the L3 interconnect and the ACP. The ARM ACP port is designed to support up to eight unique transactions concurrently (eight unique transaction IDs are supported). However, the FPGA fabric can have any number of masters requesting coherent transactions. The ACP mapper dynamically allocates the available eight transaction IDs to the requesting masters to ensure that all masters have access to coherent memory regions.

IEEE standard 754-1985 Floating Point Unit

Both ARM Cortex-A9 processor cores include full support for IEEE standard 754-1985 floating-point operations—important for imaging, signal processing, scientific computing, and graphics.

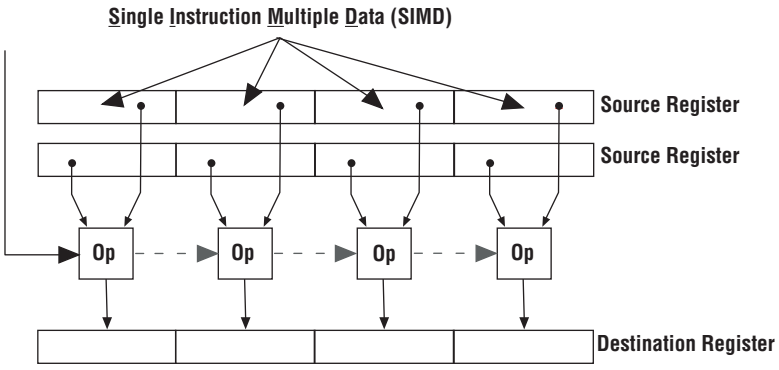
The floating-point unit (FPU) fully supports single- and double-precision add, subtract, multiply, divide, multiply/accumulate, and square-root operations. The FPU also converts between floating-point data formats and integers, including special operations to round-towards-zero required by high-level languages.

The FPU greatly increases performance for applications that heavily rely on floating-point arithmetic operations such as advanced control algorithms, imaging (scaling, 3D transforms), fast Fourier transforms (FFT), and digital filtering in graphics.

NEON Media Processing Engine

Both of the ARM Cortex-A9 processor cores include an ARM NEON media processing engine (MPE) that supports simultaneous operations on multiple data, also called SIMD processing, shown in Figure 5. The NEON processing engine accelerates multimedia and signal processing algorithms, such as video encoding/decoding, 2D/3D graphics, audio and speech processing, image processing, telephony, and sound synthesis. The SIMD architecture completes some signal processing algorithms up to eight times faster than a scaler processor.

Figure 5. SIMD Processing



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

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

The available operations include the following functionality:

- Addition and subtraction
- Multiplication with optional accumulation
- 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 details on the ARM NEON processing engine, including application benchmarks, refer to the *NEON Technology Introduction* presentation.

Jazelle Dynamic Byte-Code Compiler Support

Each Cortex-A9 CPU includes support for ARM Jazelle technology, a combined hardware and software solution. ARM Jazelle software is a full-featured, multi-tasking Java Virtual Machine (JVM), highly optimized to exploit Jazelle architecture extensions.

The ARM Jazelle Direct Bytecode eXecution (DBX) technology supports direct execution of Java bytecodes. This flexibility allows a Cortex-A9 processor to efficiently run an established operating system, middleware, and Java applications.

The ARM Jazelle Runtime Compilation Target (RCT) technology supports efficient ahead-of-time (AOT) and just-in-time (JIT) compilation with Java and other execution environments.

TrustZone System-Wide Security

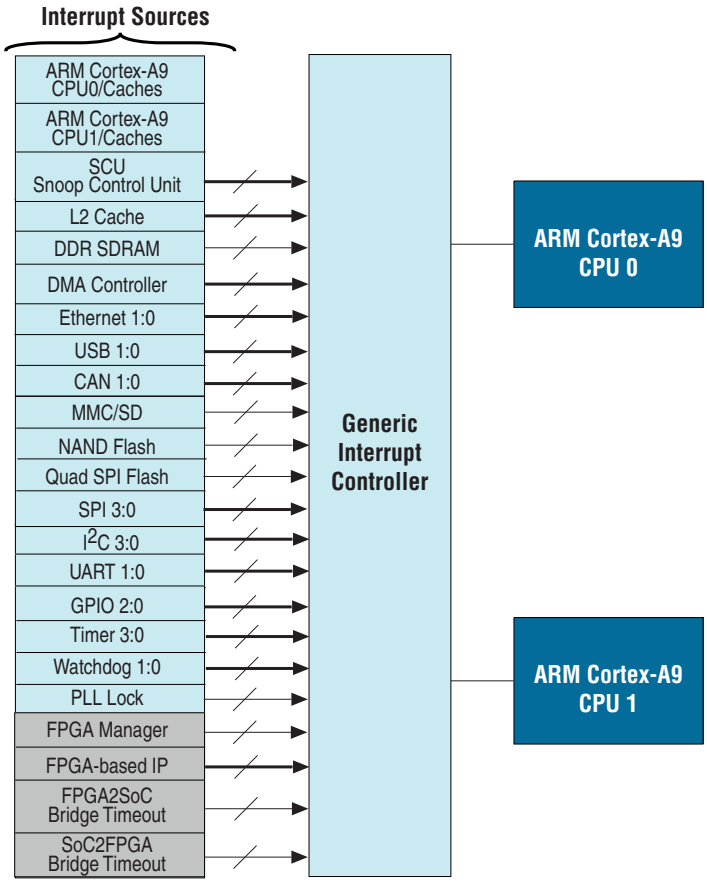
ARM's TrustZone technology lets you create secure subsystem extensions throughout the HPS and FPGA fabric. This system-wide approach enables secure transactions between the processors, peripherals, and memory, ensuring that errant or malicious software cannot interact with or record data traffic between devices in the secure domain.

The processor, DMA controller, and FPGA-to-HPS bridge support security on a per-transaction basis. The SDRAM controller subsystem supports secure and non-secure regions for each port of the SDRAM multiport front end. At boot time the processors and slave ports in the system are set to a secure state. The security settings for most of the slave ports can be modified under software control.

Generic Interrupt Controller

The Generic Interrupt Controller (GIC) is shared by both Cortex-A9 CPUs, as shown in Figure 6. There are over 130 unique interrupt sources, including the dedicated HPS peripherals and functions implemented in the FPGA fabric. There are up to 64 unique interrupts originating from IP in the FPGA fabric. The FPGA configuration manager and the timeout signals from the FPGA-to-HPS and HPS-to-FPGA bus bridges are also potential interrupt sources.

Figure 6. Interrupts Sources to Interrupt Controller



For some peripherals, such as the USB controllers, multiple interrupt sources are combined to a single interrupt to the processors. Each USB controller supports up to 32 interrupts from individual USB endpoints, all of which are combined into a single interrupt to the processor.

Private CPU Timers

As shown in Figure 3, each Cortex-A9 processor contains both private timers and shares a global timer. Private timers are only accessible from the associated processor core. Each CPU has a private 32-bit interval timer and a private 32-bit watchdog timer. The 32-bit interval timer has a programmable 8-bit prescaler, supports single-shot or auto-reload modes, and has an optional processor interrupt.

The 32-bit watchdog timer is primarily designed to react to errant programs by resetting the CPU. When the watchdog timer is enabled, application software must periodically reset the counter. If the counter reaches zero, it implies that the application software is stuck in an infinite loop or is otherwise locked. When the watchdog timer reaches zero, the system resets the CPU. If not used as a watchdog timer, it becomes an optional second interval timer.

Shared Global Timer

Both CPUs share a global 64-bit auto-incrementing timer, which is primarily used by the operating system. Each CPU has a private 64-bit compare value that generates private interrupts when the counter reaches the specified value.

HPS Boot Options

The HPS of the Altera SoC FPGA can operate independently from the FPGA fabric. The processor booting does not depend upon the FPGA fabric, unless desired.

After a power-on reset or processor reset, one processor begins executing from an on-chip RAM containing the primary bootloader program while the second processor is held in reset. The boot processor reads the state of three boot select pins that specify where the initial software image is stored, typically in flash memory. In the case of a boot from flash memory, the ROM code initializes the boot source interface, copies the initial software image to the internal SRAM, and then executes the program. The initial software image includes the entry point to the user's application software image. In the case of a boot from the FPGA fabric, the CPU waits until the FPGA fabric reports that it is configured and has entered user mode before the initial software image is copied from the FPGA fabric to internal SRAM and program execution begins.

The processor can boot from any of the following sources:

- External Quad Shared Peripheral Interrupt (SPI) flash memory (NOR)
- External NAND flash memory (Open NAND Flash Interface (ONFI) 1.0-compliant)
- External MultiMediaCard (MMC)/Secure Digital (SD) flash memory
- Via the FPGA fabric—the CPU waits until the FPGA fabric reports that it is configured and has entered user mode

The initial software image also contains various elements to protect the image against accidental or intentional modification. If the processor cannot successfully boot from the specified location, the HPS automatically reverts to the location of the last image loaded. If there are no previous images loaded, then the system attempts to boot from the FPGA fabric.

System Interconnect

The remainder of the HPS is located outside of the MPU subsystem, as shown in Figure 7. The processor accesses the rest of the HPS through a pair of 64-bit Advanced Microcontroller Bus Architecture (AMBA®) Advanced eXtensible Interface (AXI™) masters. The high-bandwidth peripherals, including the FPGA data ports, connect to the L3 interconnect structure. The L3 interconnect is further partitioned into three major sub-switches. The L3 interconnect uses a multilayer, non-blocking architecture that supports multiple, simultaneous transactions between peripherals, sub-switches, SDRAM, and the MPU subsystem. Each L3 bus master has programmable priority.

SoC FPGAs use the 32-bit AMBA High-performance Bus (AHB™) as a low-power bus to low-power peripherals, and high-performance 64-bit AXI to high-performance peripherals. The lower-bandwidth peripherals reside on the level 4 (L4) bus, implemented as a 32-bit ARM Advanced Peripheral Bus (APB™). Some peripherals, like the DMA controller, have low-bandwidth control connections on the L4 interconnect and high-bandwidth data transfer connections on the L3 interconnect.

Figure 7. SoC FPGA Hard Processor System Connections

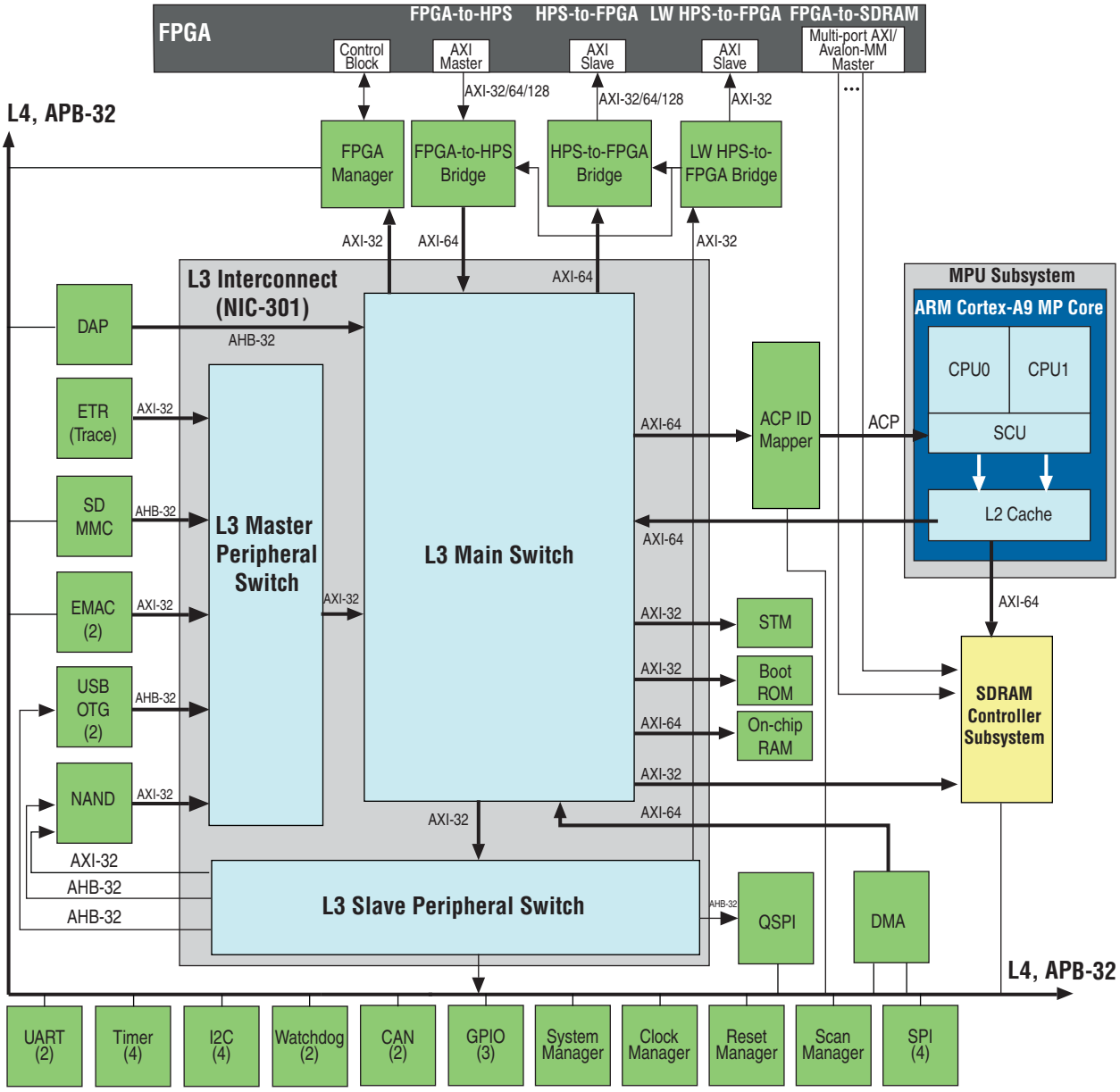


Table 1 lists the connections between the various system bus masters and slaves. Not every bus master communicates with every bus slave, although the MPU subsystem, the FPGA-to-HPS bridge, lightweight HPS-to-FPGA bridge, and the Debug Access Port (DAP) have universal access.

Table 1. Master/Slave Connection Matrix

Masters \ Slaves	L4	FPGA2HPSREGS	FPGA2HPSREGS	USB0	USB1	NAND REGS	NAND DATA	QSPIDATA	FPGAMGRDATA	HPS2FPGAREGS	ACPIDMAP	ROM	OCRAM	SDRAM Subsystem	LWHPS2FPGAREGS
Masters															
MPU subsystem	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	—	✓	✓	✓	✓
FPGA-to-HPS	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
DMA	✓	—	—	—	—	—	✓	✓	✓	✓	✓	—	✓	✓	✓
EMAC0	—	—	—	—	—	—	—	—	—	✓	✓	—	✓	✓	✓
EMAC1	—	—	—	—	—	—	—	—	—	✓	✓	—	✓	✓	✓
USB0	—	—	—	—	—	—	—	—	—	✓	✓	—	✓	✓	✓
USB1	—	—	—	—	—	—	—	—	—	✓	✓	—	✓	✓	✓
NAND	—	—	—	—	—	—	—	—	—	✓	✓	—	✓	✓	✓
SD/MMC	—	—	—	—	—	—	—	—	—	✓	✓	—	✓	✓	✓
Embedded Trace Router (ETR)	—	—	—	—	—	—	—	—	—	✓	—	—	✓	✓	✓
Debug Access Port	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	—	✓	✓	✓

SoC FPGA System Address Map

The HPS address map specifies the addresses of slaves such as memory and peripherals as viewed by the MPU and other masters.

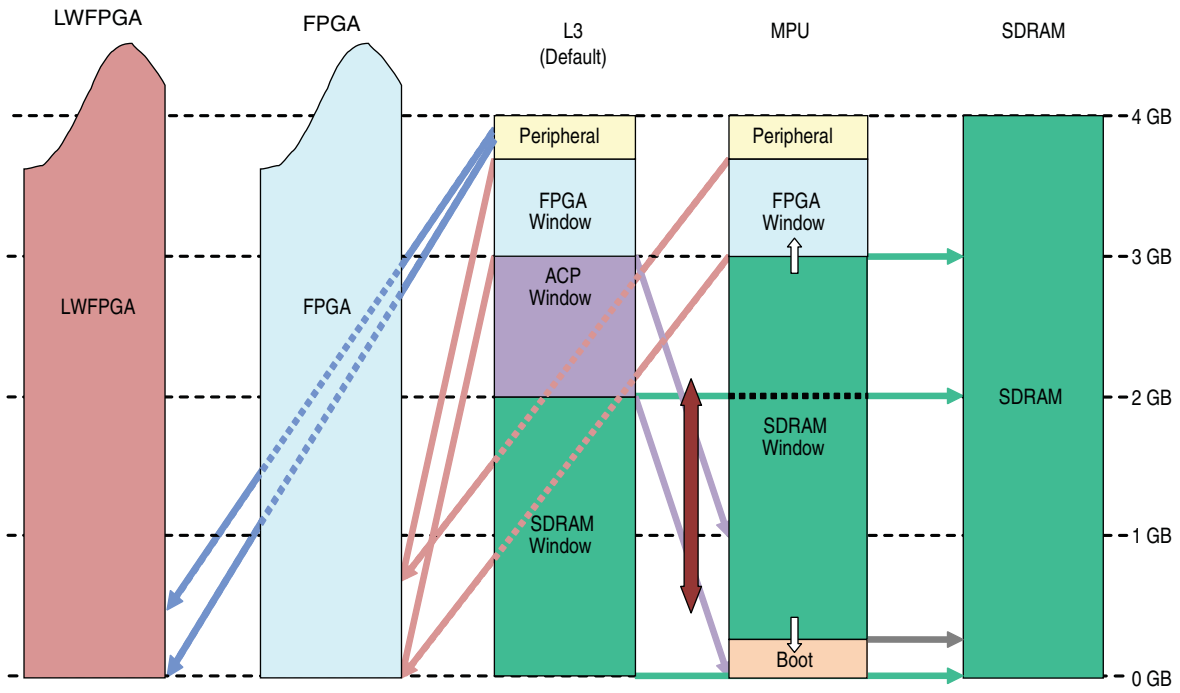
The HPS has multiple address spaces that are either contained inside the HPS or are at the boundary between the FPGA fabric and HPS. Each address space in the HPS system address map is 4 GB in total, as shown in Table 2 and Figure 8.

Table 2. HPS Address Spaces

Name	Size	Masters
MPU	4 GB	MPU
L3	4 GB	All L3 masters except MPU
FPGASLAVES	Unlimited	All L3 masters accessing HPS-to-FPGA AXI bridge
LWFPGASLAVES	Unlimited	All L3 masters accessing lightweight HPS-to-FPGA AXI bridge

Figure 8 shows the relationships between the HPS address spaces. Some address spaces have windows into other address spaces. The thin colored arrows indicate windows mapped to other address ranges (the arrows point to mapped address spaces). For example, the ACP window in the L3 address space provides access into the bottom 1 GB of the CPU address space. The vertical arrows in the SDRAM window of the CPU address space shows that the boundaries of the SDRAM window can be moved in the direction of the arrows.

Figure 8. HPS Address Spaces



The top 64 megabytes (MB) in the CPU and L3 address spaces is always allocated to the HPS dedicated peripherals. The CPU and L3 peripherals support up to 1 GB of address space to communicate with slave peripherals in FPGA fabric, minus the 64 MB at the top allocated to the HPS dedicated peripherals.

You can address architecture on the FPGA fabric without limits. A 1 MB window in the address map allows sharing between the FPGA fabric and the HPS. (soft logic in the FPGA fabric performs address decoding). The L3 and CPU address spaces provide a window of nearly 1 GB (1 GB minus 64 MB for peripherals) into the FPGA address space.

After power-on or a cold or warm reset, the bottom 1 MB is allocated to the boot ROM for the MPU subsystem and L3 masters. Under software control, the lower 1 MB can be remapped to the 64 KB on-chip RAM, or the bottom 1 MB of external SDRAM.

The MPU subsystem directly addresses a minimum of nearly 3 GB. The L3 masters support up to 2 GB of SDRAM. Access to coherent memory from the L3 masters is performed via the 1 GB of address space allocated to the ACP, as described in "Accelerator Coherency Port" on page 5.

Besides transactions over the system bus, IP implemented in the FPGA fabric have a separate private data path to the full 4 GB of SDRAM.

Communication Between the HPS and FPGA Fabric

Systems implemented with separate CPU and FPGA devices usually suffer from communication bottlenecks between the two devices. Although the HPS and the FPGA fabric can operate entirely independently in the SoC FPGA, they share two high-bandwidth, 128-bit AMBA AXI bus bridges called FPGA-to-HPS and HPS-to-FPGA bridges. IP built in the FPGA fabric have access to HPS L3 and L4 slaves via the FPGA-to-HPS bridge. Similarly, HPS L3 masters have access to slaves in the FPGA fabric via the HPS-to-FPGA bridge. Both bridges have configurable data path widths of 32, 64, or 128 bits. The variable data width allows the data rate to be tuned when crossing between the FPGA fabric and HPS L3 buses to maximize system performance. For example, 64-bit wide transactions from a 400-MHz L3 master switch can be rate-matched to 128-bit wide transactions at 200MHz into the FPGA fabric.

An additional lightweight HPS-to-FPGA bridge provides accesses to the control and status register (CSR) slave ports in the FPGA fabric. Use of this bridge frees up the remaining HPS-to-FPGA bridge to perform high speed bursting transactions. The lightweight HPS-to-FPGA bridge is 32-bit because CSR slaves are typically 32 bits wide.

All communication from the FPGA fabric side is synchronized by a locally-provided clock source. Similarly, the L3 interconnect has its own clock source. The SoC FPGA provides the necessary asynchronous clock-crossing logic between the two domains.

The FPGA-to-HPS and HPS-to-FPGA bus bridges are AMBA AXI-3 compatible. In addition, the FPGA fabric supports other bus interface standards. Altera's Qsys system integration tool allows you to connect soft logic in the FPGA based on AMBA AXI-3, Avalon Memory-Mapped (Avalon-MM), and Avalon Streaming (Avalon-ST) interface standards. Qsys creates a high-performance network-on-a-chip (NoC) interconnect in the FPGA fabric that allows components, based on different bus standards, to communicate seamlessly.

You can flag transactions over the FPGA-to-HPS bridge as TrustZone secure or non-secure on a per-transaction basis. Transactions over the HPS-to-FPGA bridges are secure during the boot process, but they can be secured or unsecured through software control.

System-Level Management

The HPS and the FPGA fabric are both stand-alone entities that can operate independently from one another. However, the HPS can directly configure and monitor the FPGA fabric via the FPGA configuration manager.

Beyond the SoC FPGA's on-chip FPGA fabric, the processor can also configure additional external FPGAs from the same configuration memory source.

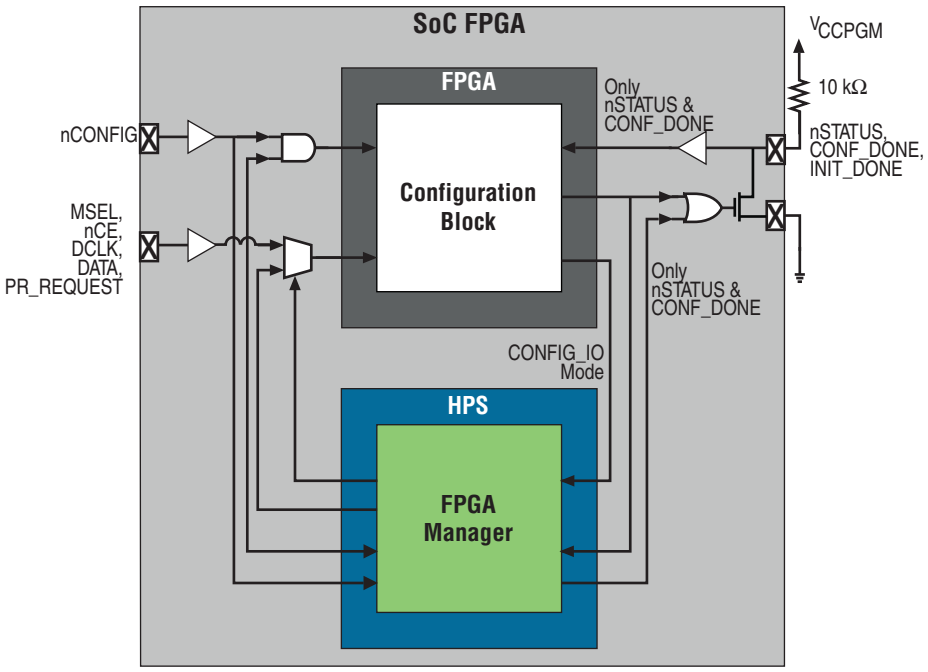
FPGA Manager

The FPGA manager is a 32-bit peripheral, as shown in Figure 9. The configuration interface is familiar to existing Altera FPGA designers and is similar to the passive-parallel configuration mode used to configure other Altera FPGA devices from an external processor. In addition to the ability to program the FPGA fabric from the HPS, the FPGA fabric also supports configuration from external, active, or passive sources.

The FPGA manager provides various configuration options:

- Configure the full FPGA
- Configure just the I/O and have the remainder of the FPGA configured over the PCI Express® (PCIe®) port
- Partially reconfigure portions of the FPGA fabric
- Provide decompression on a compressed bitstream image
- Decrypt an FPGA bitstream previously encrypted using Advanced Encryption Standards AES, thereby providing an additional layer of system security
- In high-reliability applications, perform bitstream scrubbing to further protect against single event upsets (SEUs)

Figure 9. FPGA Manager



Besides configuring the internal FPGA fabric, the HPS can optionally program other FPGAs in the system. The CPU controls the CONF_DONE pin and can delay completion of multi-FPGA configurations.

Clock Manager

The clock manager generates and manages all the system clocks in the HPS.

The clock manager controls the three primary phase-locked loops (PLLs), including the output frequency generated, the phase, and the delay from the selected clock input. Similarly, the CPU can monitor the lock status for each PLL.

Table 3 shows the clock sources for each of the PLLs.

Table 3. Clock Sources for Clock Manager PLLs

PLL	Primary Oscillator Input	Secondary Oscillator Input	From FPGA Fabric Reference Clock
Main PLL	✓	—	—
SDRAM PLL	✓	✓	SDRAM reference clock
Peripheral PLL	✓	✓	Peripheral reference clock

The HPS architecture automatically handles signals that cross between clock domains. Reducing the clock frequency helps reduce overall system power consumption. Under processor control, the clock manager can change the main system clock without affecting the clocks controlling the peripherals or SDRAM interface. Directly changing the PLL's operating frequency using the multiply and divide values provides fine-grain frequency control and long settling times between changes. Changing the post-scale counters associated with the PLL results in quick changes, but with reduced frequency resolution.

Reset Manager

The reset manager determines the source and type of reset to the system and performs any necessary prequalification of a reset signal before forwarding it to the HPS or FPGA fabric. Based on the type of reset, a reset signal might reset only a portion of the entire device.

In the HPS, there are three reset request types:

- Cold reset—forces the entire HPS, including debugging functions, into a known default state to begin/restart the boot process.
- Warm reset—recovers a non-responsive system. This reset only affects a portion of the system and not the debugging infrastructure described in "Debugging Infrastructure" on page 17. A warm reset does not clear the system configuration options set during a cold reset or power-on condition.
- Debug reset—recovers only debug and trace functions should they become nonresponsive.

Global cold reset requests reset the entire device. The following sources can initiate a cold reset request:

- The power-on reset (POR) monitor
- The nPOR input pin
- Cold reset request from FPGA fabric
- Software-based cold reset request

The following sources can initiate a warm reset request:

- Watchdog reset from an expired watchdog timer associated with either of the two CPU-private watchdog timers.
- The nRST bidirectional, open-drain pin. Any associated cold or warm reset causes the HPS to actively drive this pin to a logic low to reset external components.
- Warm reset request from FPGA fabric.
- Software-based warm reset request.

Reset requests might also originate from the following sources:

- DAP debug controller
- FPGA fabric debug reset request

Debugging Infrastructure

Developing and debugging a highly-integrated SoC device can be challenging, especially because so much of the application is deeply embedded in the device. To speed up development and verification, the Altera SoC FPGA includes extensive debugging resources that provide system visibility via ARM's CoreSight™ on-chip debug and trace IP that include the following features:

- Individual interactive debug for each CPU
- Individual program trace for each CPU
- Cross triggering between CPUs
- Cross triggering between CPU and debugging functions

Interactive Debug

Each Cortex-A9 processor has built-in debug capabilities, including the following items:

- Six hardware breakpoints, including two with context ID comparison capability to differentiate between different code streams
- Four watchpoints
- Three control sources, including the external JTAG or Serial Wire Debug (SWD) tools, by FPGA soft IP, or by CPU-based monitor code

Program Trace

Each processor has an independent Program Trace Macrocell (PTM) that provides real-time instruction flow trace capability. The PTM is compatible with a number of third-party debugging tools. The PTM trace information is highly compressed to maximize throughput by tagging specific points in the program execution flow, called *waypoints*. Waypoints are changes to the program flow or specific events, such as an exception, branches, changes in context, and changes in processor instruction or security state.

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

- The number of CPU cycles between waypoints
- The global time stamp value
- Target addresses for direct branches

External debugger software decodes the trace information into a form that more closely resembles the original program flow.

Event Trace and Cross Trigger

Specific events from each CPU can be traced or used as trigger conditions. The PTM can export trigger events and perform actions on trigger inputs. For example, a breakpoint in CPU0 can trigger a break in CPU1. The cross-trigger signals also interface with other HPS debug components including the FPGA fabric. The FPGA fabric can send and receive triggers to and from the HPS, allowing custom-built debugging hardware including communication over high-speed serial interfaces.

Performance Monitoring

Each Cortex-A9 processor has a Performance Monitoring Unit (PMU). The PMU provides 58 events that 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 from either the processor itself or the external debugger. The events are also supplied to the PTM and can be used as trigger or trace conditions.

SignalTap II Embedded Logic Analyzer

The FPGA fabric of the SoC FPGA is fully supported by Altera's SignalTap™ II Embedded Logic Analyzer.

Further Information

- SoC FPGA Overview:
  www.altera.com/devices/processor/soc-fpga/proc-soc-fpga.html
- *SoC FPGA Product Overview Advance Information Brief* (AIB):
  www.altera.com/literature/hb/soc-fpga/aib-01017-soc-fpga-overview.pdf
- Arria V FPGAs: Balance of Cost, Performance, and Power:
  www.altera.com/devices/fpga/arria-fpgas/arria-v/arrv-index.jsp
- Cyclone V FPGAs: Lowest System Cost and Power:
  www.altera.com/devices/fpga/cyclone-v-fpgas/cyv-index.jsp
- Dual-Core ARM Cortex-A9 MPCore Processor
  www.altera.com/devices/processor/arm/cortex-a9/m-arm-cortex-a9.html
- Using the Virtual Target with the ARM Cortex-A9 MPCore Processor:
  www.altera.com/devices/processor/arm/cortex-a9/virtual-target/proc-a9-virtual-target.html

- Qsys—Altera's System Integration Tool: www.altera.com/products/software/quartus-ii/subscription-edition/qsys/qtsqsys.html
- Processors from Altera and Embedded Alliance Partners: www.altera.com/devices/processor/emb-index.html
- Presentation: "ARM NEON Technology Introduction": http://www.arm.com/files/pdf/AT_-_NEON_for_Multimedia_Applications.pdf
- Webcast: "Getting to Know the ARM-Based SoC FPGA": www.altera.com/education/webcasts/all/wc-2011-arm-based-soc-fpga.html
- White Paper: *Strategic Considerations for Emerging SoC FPGAs* http://www.altera.com/literature/wp/wp-01157-embedded-soc.pdf
- White Paper: *ARM Cortex-A9 Processors* http://www.arm.com/files/pdf/ARMCortexA-9Processors.pdf

Document Revision History

Table 4 shows the revision history for this document.

Table 4. Document Revision History

Date	Version	Changes
October 2011	1.0	Initial release.
February 2012	1.1	Edited technical figures.