Architecture Matters: Choosing the Right SoC FPGA for Your Application

White Paper

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

WP-01202-1.0

Document publication date: November 2013

Subscribe

Contents

Introduction 1–1
- SoC FPGAs Available Today 1–1
- Benefits Over Two-chip Processor and FPGA Applications 1–3
- Benefits Over ASIC-Based Processors 1–3
- Benefits Over Other Processors or Microcontrollers 1–4
- How to Choose the Right SoC FPGA for a Specific Application 1–4

System Performance 1–4
- Importance of the L3 Interconnect: Central Switch for Maximum Performance 1–5
- Processor-to-FPGA Interconnect: Ensuring the Benefit of an Integrated Device 1–6
- DDR Memory Controller Performance 1–8
- FPGA Connection to Processor's DDR Memory Controller 1–11
- Hardware Acceleration and Cache Coherency 1–13
- Additional Memory Controllers Improve Maximum System Performance 1–13
- System Reliability and Flexibility 1–14
- Protecting Memory Contents with ECC 1–14
- Memory Protection for Shared Memory 1–16
- Watchdog Reset and Its Effect on FPGA Logic 1–17
- Fail-Safe Booting and Configuration 1–18

Flexibility 1–18
- Multiple Options for Processor Boot and FPGA Configuration 1–19
- Multiple Boot Images 1–20
- On-Chip FPGA Interfaces 1–21
- Common Footprint Density/Transceiver/Feature Migration 1–23

System Cost 1–24
- Integrated Functionality 1–24
- High-Speed Transceivers 1–24
- Power Supply Cost 1–25

Power 1–25
- Power Savings via Integration 1–25
- Power-Saving Modes 1–26
- Power-On/Off Sequencing Requirements 1–27

Future Roadmap 1–27
- Altera's Three-Generation Processor Roadmap 1–28
- The Foundation: Silicon Process Technology 1–28
- FinFET Technology 1–29
- Tools Roadmap 1–30

Development Tools 1–31
- Development Tool Challenges 1–33
- ARM Compatibility a Given; FPGA Implementation a Difference 1–33
- Whole-Chip Debugging 1–33
- FPGA-Adaptive Debugging 1–34
- Single Debug Cable for ARM and FPGA Development 1–34
- Cross-Triggering Between Hardware and Software 1–35
- Tracking and Monitoring Hardware and Software Events 1–35
- Profiling CPUs and FPGA 1–36
- Multicore Debugging 1–36
- Standard Tools and Flows 1–37
- ARM DS-5 Altera Edition Toolkit 1–37
- Development Tools Summary 1–38

Conclusion 1–38
Further Information 1–39
Acknowledgements 1–39
Document Revision History 1–39

Architecture Matters: Choosing the Right SoC FPGA for Your Application

WP-01202-1.0 White Paper

This white paper presents and analyzes various metrics to help system architects, engineers and managers decide if SoC FPGAs are a fit for their application and, if so, which vendor's devices may be best suited. Key aspects of this paper are also highlighted in an online video series, A Look Inside: SoC FPGAs.

Introduction

SoC FPGA devices integrate both processor and FPGA architectures into a single device. Melding the two technologies provides a variety of benefits including higher integration, lower power, smaller board size, and higher bandwidth communication between the processor and FPGA. Best-in-class devices exploit the unique advantages of a merged processor and FPGA system while retaining the benefits of a stand-alone processor and FPGA approach.

SoC FPGAs Available Today

At present, there are primarily three SoC FPGAs on the market, as shown in Table 1. The processors in these devices are fully dedicated, "hardened" processor subsystems, (not a soft intellectual property (IP) core implemented in the FPGA fabric). All three of these device families employ a full-featured ARM® processor with a complete memory hierarchy and dedicated peripherals that largely boot, run, and act like any "normal" ARM processor.

The Microsemi SmartFusion2 SoC FPGAs are based around the ARM Cortex™-M3 embedded processor, primarily targeting microcontroller applications. The Altera SoC FPGA and Xilinx Zynq-7000 devices use a dual-core ARM Cortex-A9 application processor. In addition to the processor, an SoC FPGA includes a rich set of peripherals, on-chip memory, an FPGA-style logic array, and plentiful I/O. Refer to Table 1.

Table 1. Commercially-Available SoC FPGAs (Part 1 of 3)

	Altera SoC FPGAs	Xilinx Zynq-7000 EPP	Microsemi SmartFusion2
Processor	ARM Cortex-A9	ARM Cortex-A9	ARM Cortex-M3
Processor Class	Application processor	Application processor	Microcontroller
Single or Dual Core	Single or Dual	Dual	Single
Processor Max. Frequency	1.05 GHz	1.0 GHz	166 MHz
L1 Cache	Data: 32 KB Instruction: 32 KB	Data: 32 KB Instruction: 32 KB	No data cache Instruction: 8 KB
L2 Cache	Unified: 512 KB, with error correction code (ECC)	Unified: 512 KB	Not available

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

Table 1. Commercially-Available SoC FPGAs (Part 2 of 3)

	Altera SoC FPGAs	Xilinx Zynq-7000 EPP	Microsemi SmartFusion2
Memory Management Unit (MMU)	Yes	Yes	Yes
Floating-Point Unit/NEON™ Multimedia Engine	Yes	Yes	Not available
Acceleration Coherency Port (ACP)	Yes	Yes	Not available
Interrupt Controller	Generic (GIC)	Generic (GIC)	Nested, vectored (NVIC)
On-Chip Processor RAM	64 KB, with ECC	256 KB, no ECC	64 KB, no ECC
Direct Memory Access Controller	8-channel ARM DMA330 32 peripheral requests (FPGA + hard processor system	8-channel ARM DMA330 4 peripheral requests (FPGA only)	1-channel HPDMA 4 requests
External Memory Controller	Yes	Yes	Yes
Memory Types Supported	LPDDR2, DDR2, DDR3L, DDR3	LPDDR2, DDR2, DDR3L, DDR3	LPDDR, DDR2, DDR3
External Memory ECC	16 bit, 32 bit	16 bit	8 bit, 16 bit, 32 bit
External Memory Bus Max. Frequency	400 MHz (Cyclone® V SoC), 533 MHz (Arria® V SoC)	533 MHz	333 MHz
Processor Peripherals	1x quad SPI controller with 4 chip selects 1x NAND controller (single- and multilevel cell - MLC or SLC) 2x 10/100/1G Ethernet controller 2x USB 2.0 On-the-Go (OTG) controller 1x SD/MMC/SDIO controller 2x UART 4x I^{2}C controller 2x CAN controller 2x SPI master, 2x SPI slave controller 4x 32 bit general-purpose timers 2x 32 bit watchdog timers	1x quad SPI or dual quad SPI controller with 2 chip selects 1x static memory controller (NAND-SLC, NOR, or SSRAM) 2x 10/100/1G Ethernet controller 2x USB 2.0 OTG controller 2x SD/SDIO controller 2x UART 2x I^{2}C controller 2x CAN controller 2x SPI controllers (master or slave) 2x 16 bit triple-mode timer/counters 1x 24 bit watchdog timer	1x 10/100/1G Ethernet controller 2x USB 2.0 OTG controller 2x UART 2x I^{2}C controller 1x CAN controller 2x SPI 2x general-purpose timers 1x watchdog timer 1x real-time clock (RTC)
FPGA Fabric	Cyclone V, Arria V	Artix-7, Kintex-7	Fusion2
FPGA Logic Density Range	25 K to 462 K LE	28 K to 444 K LC	6 K to 146 K LE
Hardened Memory Controllers in FPGA	Up to 3, with ECC	Not available	Not available
High-speed Transceivers	Available at all densities	Higher-density devices only	Higher-density devices only

Table 1. Commercially-Available SoC FPGAs (Part 3 of 3)

	Altera SoC FPGAs	Xilinx Zynq-7000 EPP	Microsemi SmartFusion2
Analog Mixed Signal (AMS)	Not available	2 x 12-bit, 1 MSPS analog-to-digital converters (ADCs)	Not available
Boot Sequence	Processor first, FPGA first, or both simultaneous	Processor first	Processor boot, FPGA non-volatile

Does an SoC FPGA make sense for a next-generation system design? The following three questions may help a system designer, architect, or hardware manager make that determination:

- Does the existing design use an FPGA and a separate microprocessor?
- Does the current generation use a proprietary ASIC that includes a microprocessor?
- Is a microprocessor being used today, but would benefit from a peripheral set more tailored to the application?

Benefits Over Two-chip Processor and FPGA Applications

For designs that already use an FPGA and a separate microprocessor, these devices should definitely be considered.

The SoC FPGA likely provides at least comparable, and likely superior, functionality and performance, but at a lower board space, lower power, and lower system cost—by as much as 50% less. Integrating these technologies on the same piece of silicon eliminates the cost of one of the plastic packages, and one device saves a lot of board space compared with two. If both the CPU and FPGA in the design use separate external memories, it may also be possible to consolidate both into one memory device, saving even more system cost, board space, and power. Because the signals between the processor and the FPGA now reside on the same silicon, communication between the two consumes substantially less power compared to using separate chips. Plus, thanks to thousands of internal connections between the processor and the FPGA, an integrated solution has substantially higher bandwidth and lower latency compared to a two-chip solution.

Benefits Over ASIC-Based Processors

What if the current design uses a proprietary ASIC that includes a microprocessor? Most design teams that currently use ASIC technology have probably already investigated FPGAs at some point and likely use them during the prototyping or emulation phase. For many ASIC designers, the previous lack of an ARM processor has been a barrier to using FPGA technology for full production. This new breed of SoC FPGAs delivers a fully-functional, fully-compatible, high-performance dual-core ARM Cortex-A9 processor running up to 1 GHz with today's 28 nm process technology, eliminating that barrier.

Because SoC FPGAs leverage programmable logic technology, programmable designs benefit from all the traditional FPGA advantages over standard ASIC technology, such as:

- *No expensive mask charges or minimum purchase requirements*—Build and ship a single, cost-effective SoC FPGA solution or millions

- *Faster time to market*—No manufacturing lead times; devices available off-the-shelf at major electronics distributors
- *Lower risk*—Reprogram the SoC FPGA at any time, even after the product has shipped. Supports in-field updates and upgrades
- Adapt to changing markets requirements and emerging standards
- No additional licensing or royalty payments for the embedded processor, high-speed transceivers, or other advanced system technology

Benefits Over Other Processors or Microcontrollers

In the final scenario, systems that typically use stand-alone microprocessors or high-end microcontrollers but not FPGAs may still benefit from these new SoC FPGAs. Why? Many designers research the available off-the-shelf processors and often settle for a device that only approximately fits the application—the selected processor is missing an Ethernet port, USB channels, interrupt lines, and so on. The power of these SoC FPGAs is that a custom ARM microprocessor derivative can be created—instantly, right on the desktop. System designs that have been forced to accept compromises due to the lack of off-the-shelf processor derivatives can now be customized to fit the application. Hence, the design can be differentiated both in hardware and software, making it more difficult for competitors to copy or emulate.

How to Choose the Right SoC FPGA for a Specific Application

At first glance, the programmable SoC offerings in Table 1 from various vendors might appear similar. They all integrate an ARM processor, various peripherals, and an FPGA into a single device. In practice, however, it is critical to closely evaluate these offerings, and look deeper than the data sheet. The underlying architecture and its implications must be evaluated relative to a specific application. The SoC FPGA architecture matters. Closer examination and consideration reveals many significant differences at an architectural level.

So how does a designer choose? This white paper presents design considerations and engineering trade-off decisions for choosing the best programmable SoC for an application. The selection criteria centers on these six areas:

- System performance
- System reliability and flexibility
- System cost
- Power consumption
- Future roadmap
- Development tools

System Performance

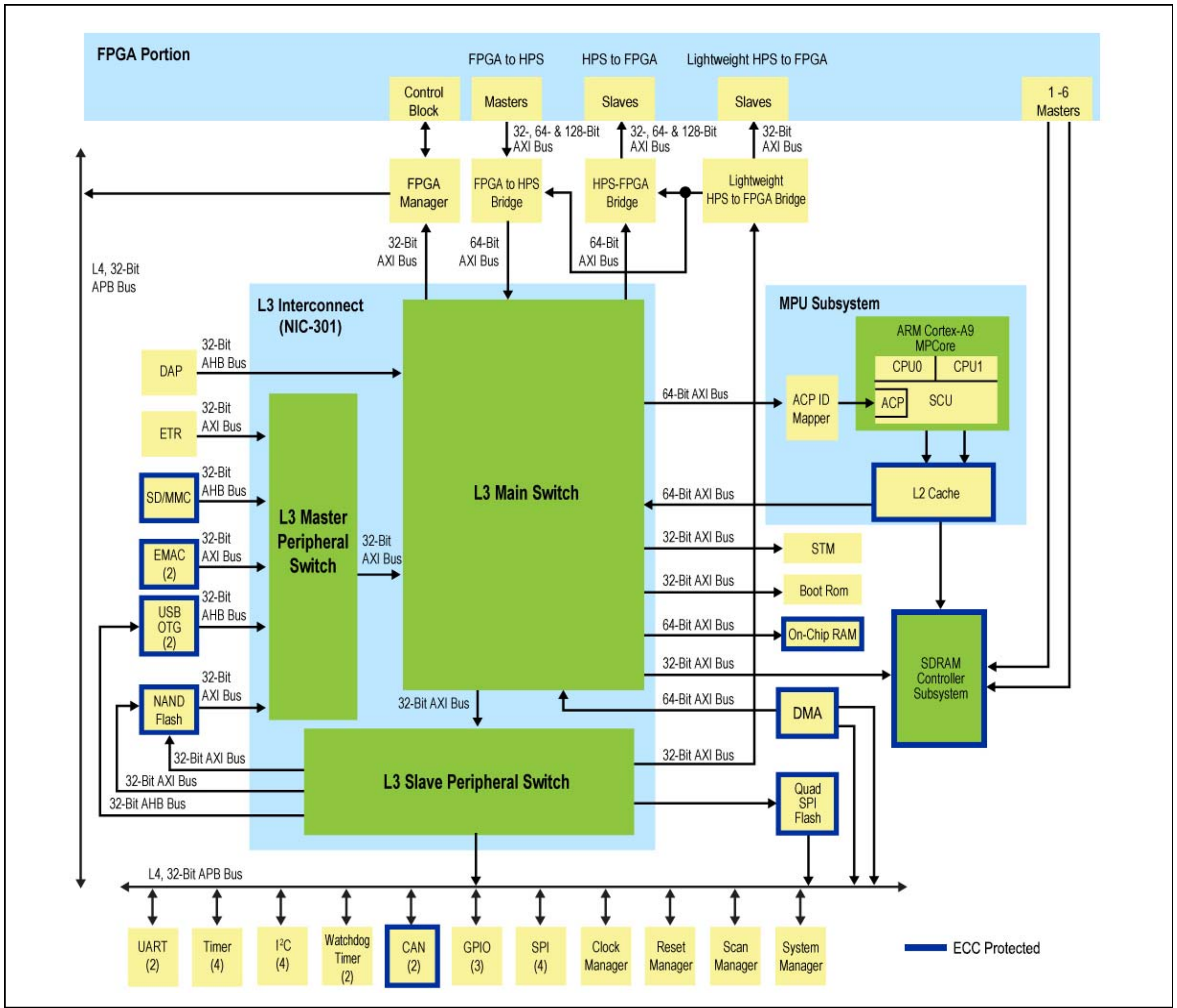
Ultimately, two areas of the SoC FPGA architecture dominate the efficient movement of data between the different elements:

- The interconnect
- Memory bandwidth, both on-chip and off

Importance of the L3 Interconnect: Central Switch for Maximum Performance

The first item to consider in an SoC architecture is the Level-3 (L3) interconnect. The L3 interconnect, named for being the next level beyond the L1 and L2 caches for data transfers, routes data between the memory, FPGA fabric, processor, and peripherals. Table 2 shows the SoC FPGA vendors feature comparison for the L3 system interconnect.

Figure 1. Altera SoC FPGA Interconnect Architecture



Altera SoC FPGAs provide an L3 system interconnect comprised of three switches—L3 main switch, L3 master peripheral switch, L3 slave peripheral switch—implemented using ARM's AMBA® NIC-301 Network Interconnect infrastructure as shown in Figure 1.

Altera SoCs use a reduced-hierarchy bus to minimize latency with a non-blocking switch architecture. The interconnect was designed to support multiple, simultaneous transactions from multiple masters with sufficient bandwidth such that each master can run without stalling ("non-blocking"). For arbitration, each master can be assigned its own priority level to guide bus arbitration. Masters with equal priority are arbitrated using a least recently used (LRU) algorithm.

Alternative SoC FPGA architectures may use a multi-level hierarchy that may introduce latency. Distributed arbitration is analogous to having multiple traffic cops. This approach necessitates the use of a central quality of service (QoS) module to ensure that no master gets stuck. Distributed arbitration also presents tuning challenges and may conflict with DDR memory controller port arbitration.

Table 2. L3 System Interconnect Feature Comparison in SoC FPGA Devices

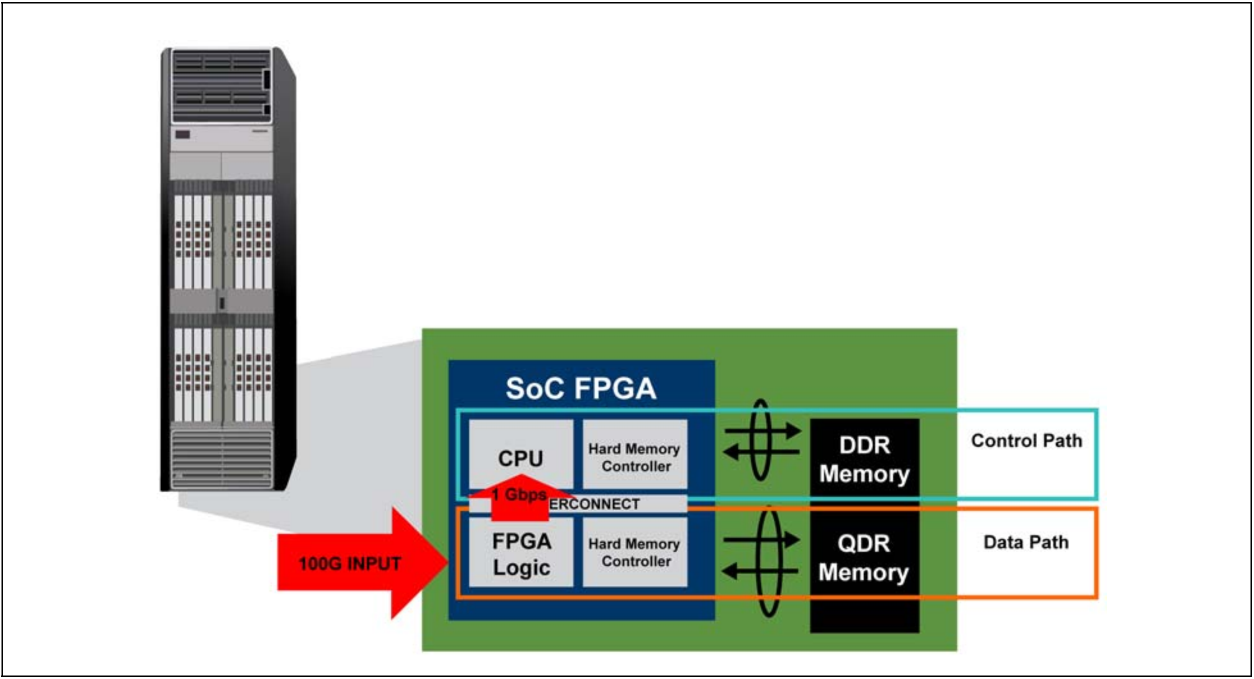
Function/Feature	Altera SoC FPGA	Vendor B
Infrastructure	ARM AMBA NIC-301	ARM AMBA NIC-301
Bus Hierarchy	Reduced hierarchy	Multi-level
Arbitration	Programmable priority for each master. LRU for requests of equal priority.	Distributed, regulated by QoS block.

Processor-to-FPGA Interconnect: Ensuring the Benefit of an Integrated Device

One of the significant promises of the SoC FPGA architecture is the tight, on-chip coupling of the processor and FPGA. To realize this performance promise, it is critical that the processor-to-FPGA interconnect be constructed with sufficient bandwidth (width and speed) and of the right types so as not to become the bottleneck of system data transfers.

To illustrate this point, imagine that a communication line card application, shown in Figure 2, needs to process 100 gigabits per second of network data. The FPGA can nimbly handle the incoming data. However, even if the processor only touches 1% of the traffic, a substantial 1 Gbps of data must flow across the interconnect between the FPGA logic and the processor. Fortunately, SoC FPGAs available today support up to 125 Gbps or more of throughput between the FPGA logic and the processor, substantially more than enough for this type of application.

Figure 2. Communication Line Card Requires Over 100 Gbps Interconnect Bandwidth Between FPGA and Processor



In terms of structure, in some SoC FPGA devices the datapath and control path compete for bandwidth. The processor may need to set up and configure accesses to hardware accelerators in the FPGA logic. If these control transactions compete with data traffic, they may block the high-throughput data traffic, halting the continuous processing of incoming data. Similarly, control signals may be delayed by high-bandwidth data traffic, adding to control latency.

To prevent this, the Altera SoC FPGAs feature a second, low-latency, non-blocking,”lightweight” interconnect bridge. The processor accesses control registers in the FPGA via this simple 32 bit ARM Advanced eXtensible Interface (AXI™) interface, without blocking or affecting the high-throughput data flow, shown in blue in Figure 3. Meanwhile, high-bandwidth data connections between the processor and FPGA support 32, 64, or 128 bit wide transactions, shown in red in Figure 3. Table 3 shows the configuration of both sets of datapaths.

Figure 3. Altera SoC FPGAs Feature High-Throughput Datapath and Non-Blocking, Low-Latency Control Path

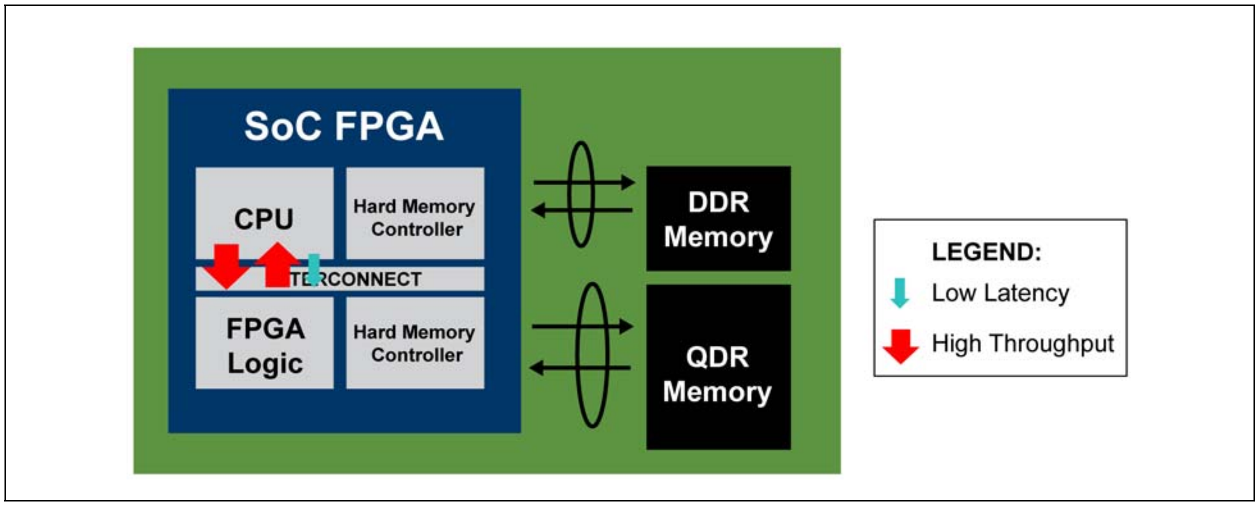


Table 3. Processor-to-FPGA System Interconnect Features in SoC FPGAs

Function/Feature	Altera SoC FPGA	Vendor B
High-Bandwidth Processor/FPGA Interconnect	1x 32/64/128 bit AXI (CPU→FPGA) 1x 32/64/128 bit AXI (FPGA→CPU)	2x 32 bit AXI (CPU→FPGA) 2x 32 bit AXI (FPGA→CPU)
Low-Latency Processor/FPGA Interconnect	1x 32 bit AXI (CPU→FPGA)	Must utilize one of the high-bandwidth busses
Total Processor/FPGA Interconnect Maximum Theoretical Bandwidth (1)	10.8 GB/s	4.8 GB/s
Processor/FPGA Interconnect Data Width	x32, x64, or x128	Fixed x32
Processor/FPGA Transaction Buffers	16 writes + ECC 16 reads + ECC	8 writes 8 reads

Note:

(1) Assumes 150 MHz interconnect bus speed. Theoretical maximum is bus speed times data width. Actual realized bandwidth will be lower due to protocol and buffering overhead.

DDR Memory Controller Performance

When selecting DDR DRAM for a design, one would typically assume that the memory speed would dominate the realized performance (see Table 4). However, other factors in terms of how intelligent the memory data transfers are prioritized, scheduled, and processed can have significant impact on overall memory performance.

To illustrate this effect, consider two SoC FPGA devices with different memory bus speeds as shown in Figure 4. Both have a dual-core ARM Cortex-A9 processor running at the same frequency of 667 MHz. However, one device has an external memory operating at 400 MHz, while the other uses an external memory running at 533 MHz. At first glance, one would expect the system with 533 MHz memory to exhibit 33% higher performance due to the higher memory performance. However, advances in the memory controller architecture produce some noticeably different results.

Figure 4. SoC FPGA Memory Performance Comparison

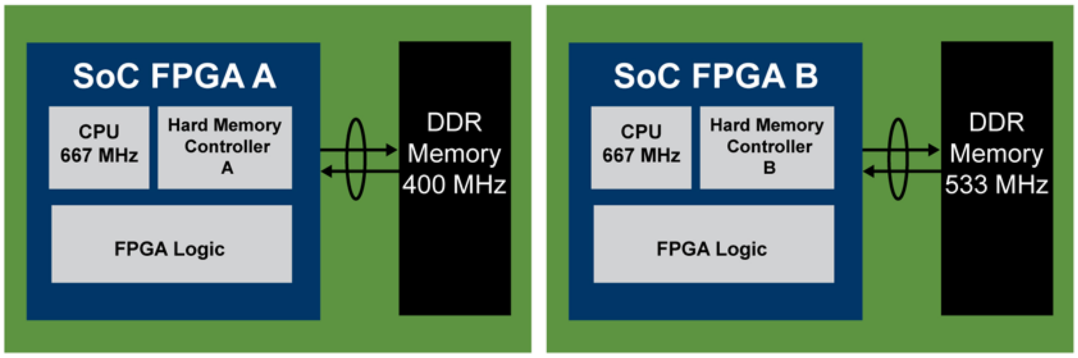
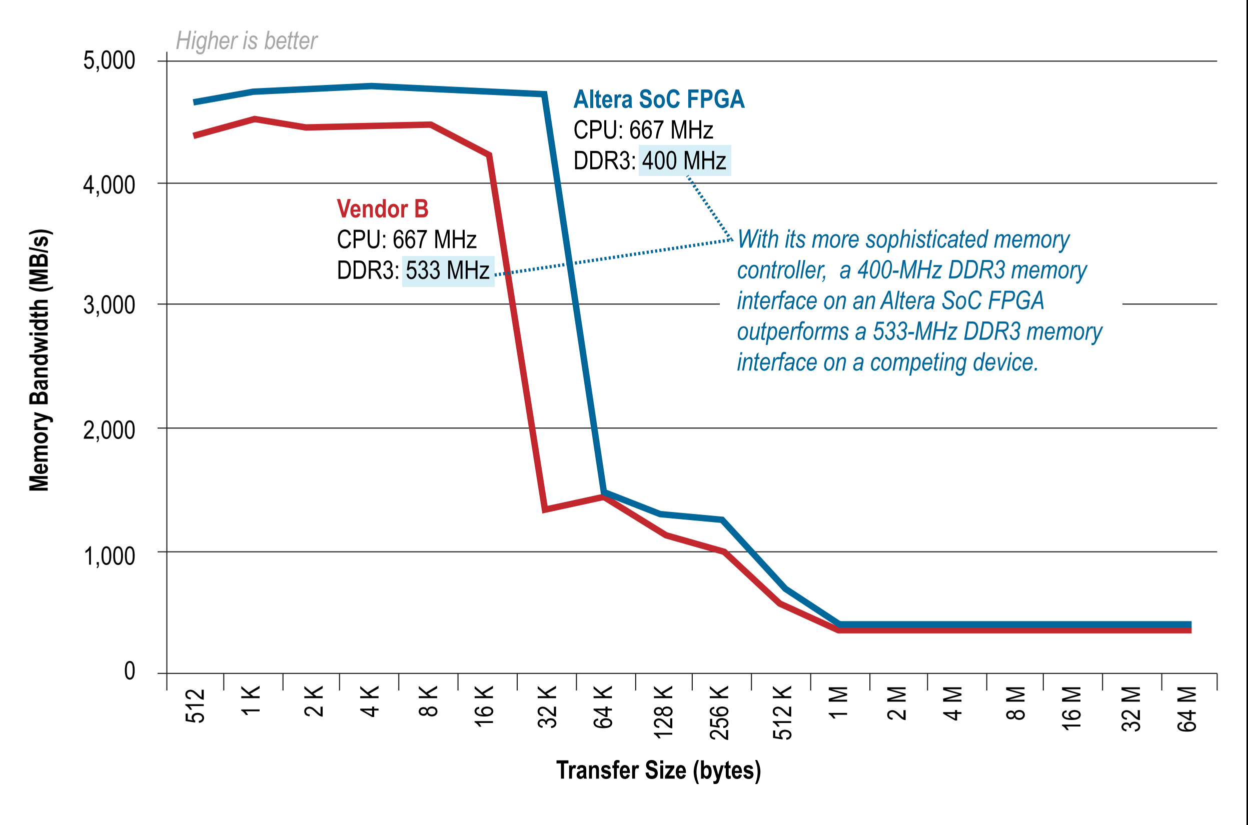


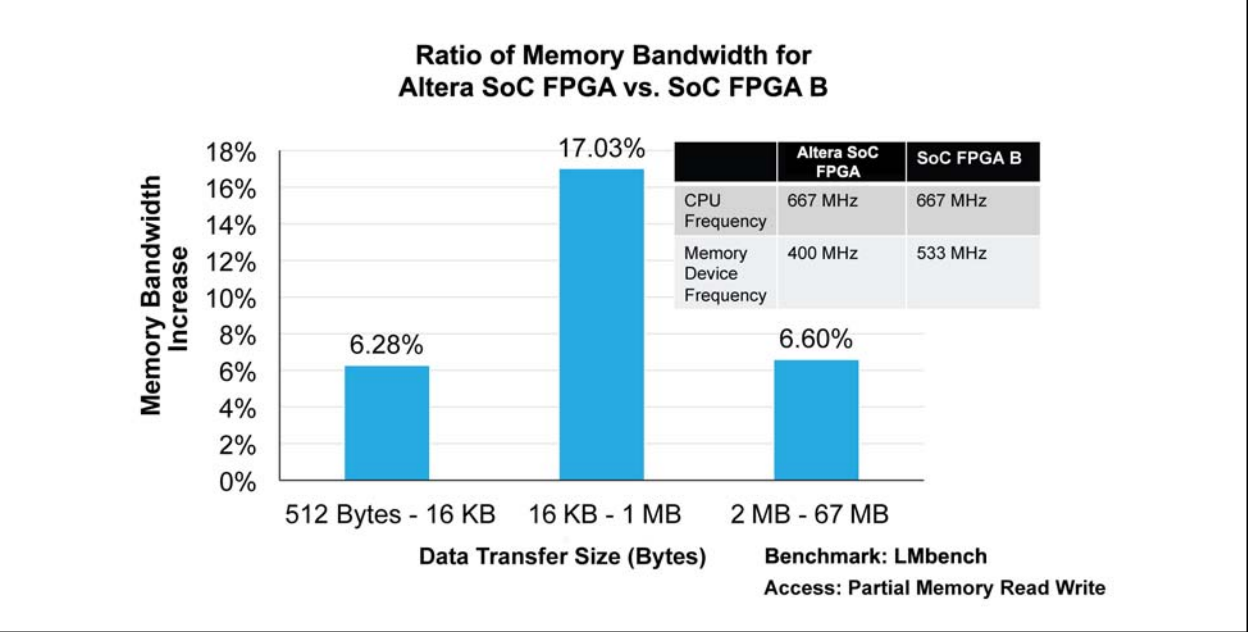
Figure 5 provides the results of a system performance benchmark called LMbench. Altera selected LMbench as it is an industry-standard benchmark (www.bitmover.com/lmbench) known for exercising the memory system performance. The partial read/write case illustrates transfers of a typical embedded application using LMbench version 3.

Figure 5. LMbench Partial Read/Write Memory Bandwidth Test Demonstrates Benefits of Advanced Controller



The bandwidth observed decreases in stages as the data size moves from the L1 cache to the L2 cache to external memory.

Figure 6. LMbench Memory Bandwidth Difference Grouped by Data Transfer Size



	Altera SoC FPGA	SoC FPGA B
CPU Frequency	667 MHz	667 MHz
Memory Device Frequency	400 MHz	533 MHz

Across the full range of small, medium, and large memory accesses, shown in Figure 6, the SoC FPGA with the more advanced memory controller, the Altera SoC FPGA, extracts up to 17% more memory bandwidth despite a slower memory operating frequency.

These results demonstrate that when comparing SoC FPGAs, it is important to check the measured memory system performance, not just the memory bus specifications. This is another example of where architecture matters. Modern memory controllers employ sophisticated algorithms to maximize efficiency from system memory. These algorithms extract maximum bandwidth by managing transaction priority, reordering command and data, and scheduling pending transactions using sophisticated algorithms like deficit weight round robin. Additional performance comes by customizing the memory controller via software to best fit the system's custom data profile, set priorities, assign ports or transaction channels, and even tweak the share of bandwidth between them. A better memory controller not only extracts more bandwidth from system memory, but also may enable the memory to run at a lower frequency and thereby saving power.

Table 4. External Memory Controller Support Comparison

Function/Feature	Altera SoC FPGA	Vendor B	Vendor C
Hardened External Memory Controller for Processor System	Yes	Yes	Yes
Maximum Supported Address Space	4G	1G	4G
Memory Types Supported	LPDDR2, DDR2, DDR3L, DDR3	LPDDR2, DDR2, DDR3L, DDR3	LPDDR, DDR2, DDR3
Data Width Configuration Modes	x8 x16 x16+ECC x32 x32+ECC	x16 x16+ECC x32	x8 x8+ECC x16 x16+ECC x32 x32+ECC
Integrated ECC Support	16 bit, 32 bit	16 bit	8 bit, 16 bit, 32 bit
External Memory Bus Maximum Frequency	400 MHz (Cyclone V SoC), 533 MHz (Arria V SoC)	533MHz	333 MHz

FPGA Connection to Processor's DDR Memory Controller

To save cost in SoC FPGA applications, functions built in the FPGA section can optionally access system main memory through the processor's DDR memory controller. However, sharing the processor's memory controller could potentially limit the performance of either the processor or the FPGA. Consequently, the connection from the FPGA to the processor's memory controller must be optimized for bandwidth.

As shown in Table 5, both the Altera SoC FPGAs and those from Vendor B both have 256 total bits out of the FPGA headed toward the processor's memory controller. In Vendor B's device, two of the four 64 bit ports are switched down to the memory controller while two of the four 64 bit ports are switched to the on-chip memory (OCM). In Altera SoC FPGAs, all lines of the 256 bit port are connected directly to the processor memory controller and can be configured with to up to six independent command/response ports, four read ports, or four write ports. Each port sharing the 256 bit interface can potentially support different bus protocols, different data widths, and different configuration. For example on the Altera SoC FPGA, the FPGA interface to the processor's DDR memory controller simultaneously supports a 128 bit Avalon® Memory-Mapped interface and two 64 bit AXI ports. At its maximum interface clock rate, the Altera FPGA-to-DDR memory interface supports up to 9,600 MB/sec peak bandwidth.

This extra bandwidth is a result of the direct connection between the FPGA interface and the processor's memory controller, unencumbered by intervening switches or interconnect layers. The direct connection supplies every memory port with the maximum possible bandwidth and maximum flexibility for prioritizing those transactions. In contrast, the four ports on Vendor B's interface are multiplexed down to two ports on the processor's memory controller, which reduces maximum bandwidth.

These FPGA interfaces to the processor's memory controller also support relative prioritization of traffic. The Altera SoC FPGAs offer eight absolute prioritization levels to custom-tailor communication within the device. Transactions from the FPGA can be dynamically and individually prioritized. Vendor B's SoC FPGAs offer two absolute prioritization levels.

Table 5. FPGA Connections to Processor's DDR Memory Controller

Function/Feature	Altera SoC FPGA	Vendor B
FPGA-to-DDR Memory Interconnect Path	256 bit, AXI/Avalon-MM interface (FPGA→DRAM)	4x 64 bit AXI (FPGA→DRAM and on-chip RAM)
Individual Port Size Options	8/16/32/64/256 bit	32/64 bit
Maximum FPGA to Interconnect Ports	6 command/response ports 4 read ports 4 write ports	4 x64 read ports 4 x64 write ports
Maximum Interconnect to Processor DDR Hard Memory Controller Ports	6 command/response ports 4 read ports 4 write ports	2 x64 read port 2 x64 write port (multiplexed down from four ports)
Connection	Direct	Switched (four FPGA ports multiplexed down to two DDR memory ports in memory interconnect)
FIFO Size	16x256 = 512 B + ECC	128x64 = 1 KB
Relative Traffic Prioritization	Yes	Yes
Absolute Prioritization Levels	8	2
Maximum Ports for AXI Exclusive Memory Sharing	Across all ports, all IDs	1 port, 2 IDs

The SoC FPGAs with an ARM Cortex-A9 processor also support ARM's AXI Exclusive feature—essentially, a special, hardware-based semaphore operation for a transaction, but without dedicating the bus to a particular master for the duration of the operation. AXI Exclusive semaphore-type operations do not impact bus access latency or the maximum achievable bandwidth. On Altera SoC FPGA devices, the AXI Exclusive feature supports transactions across all DDR memory ports. In other devices, the feature is only available on a port-by-port basis.

Hardware Acceleration and Cache Coherency

One of the additional potential benefits of the integrated processor and FPGA system is the ability to boost system performance by accelerating compute-intensive functions in FPGA logic. The processor can be offloaded by accelerating practically anything in FPGA logic—from calculating a cyclic-redundancy check (CRC) to offloading the entire TCP/IP stack. When the FPGA-based accelerator produces a new result, the data needs to be passed back to the processor as quickly as possible, so that the processor can update its view of the data.

ARM Cortex-A9-based SoC FPGAs include a feature called an Accelerator Coherency Port (ACP). Through the ACP, new data produced by an FPGA-based hardware accelerator is transferred directly to the processor's L2 cache via a low-latency direct connection—not just quickly but coherently.

Because the ACP logic automatically maintains coherency, a coherent data transfer requires approximately 30 cycles. The alternative method to ensure data coherency is to flush the L2 cache, which requires hundreds of cycles to complete. As shown in Table 6, Altera SoC FPGAs support coherent transactions for both FPGA-based functions and for processor peripherals. Other SoC FPGAs only support FPGA functions via a single dedicated port and do not support transactions from processor peripherals.

ARM originally designed the ACP interfacefor full-custom system-on-chip devices, which generally have only a few dedicated accelerators or a few peripherals that require ACP support. Consequently, the ARM ACP interface only supports eight total transactions in flight or pending. However, because of the SoC FPGA's flexible and programmable architecture, there may be many more hardware accelerators that require coherent support. To support more than eight such functions, Altera SoC FPGAs incorporate an ACP ID mapper that supports an unlimited number of pending transactions with any eight transactions currently in flight.

Table 6. Accelerator Coherency Port Differences in SoC FPGAs

	Altera SoC	Vendor B
FPGA-Based Masters Supported by ACP	Yes	Yes
Processor Peripheral Masters Supported by ACP	Yes	No
ACP ID Mapper	Yes	No
ACP In-Flight Transactions Supported	8	8 total in flight or pending
ACP Pending Transactions Supported	Unlimited	8 total in flight or pending
ACP Port Configuration	x64 AXI	x64 AXI
ACP Port Clock Source	½ CPU Clock (400 MHz)	FPGA (150 MHz)

Additional Memory Controllers Improve Maximum System Performance

SoC FPGAs all include a dedicated DDR hard memory controller as part of the processor subsystem, primarily to store and retrieve code and data for the processor. For cost-saving purposes, the processor's memory controller also can be shared with logic functions in the FPGA.

For maximum performance in some applications, however, it may be best to keep the processor's and FPGA's memory controllers separate, as outlined in Table 7. If the application software is particularly demanding, then the processor benefits from having its own exclusive memory array. Likewise, high-bandwidth FPGA applications also benefit from having their own exclusive memory arrays.

Table 7. Using Processor and FPGA Memory Interfaces for Various Application Types

Application Type	Processor Memory Controller	FPGA Memory Controller(s)
Lowest Cost	Processor and FPGA functions share a common DDR memory subsystem using processor's memory controller	Unused
Processor and FPGA Share Large Common Memory Area	Processor and FPGA functions share a common DDR memory subsystem using processor's memory controller	Available for other FPGA functions
Demanding Computational System	Processor's memory controller dedicated to servicing the processor	Any FPGA functions use FPGA memory controller to offload HPS memory controller
High-Bandwidth FPGA Function	Processor uses processor's memory controller, possibly shared with other lower-bandwidth FPGA functions	FPGA exclusively uses FPGA memory controller(s)

As shown in Table 8, Altera's 28 nm SoC FPGAs also include one or three independent hard DDR memory controllers dedicated to FPGA logic functions, with the same sophisticated features and capabilities of the processor's memory controller. All the commercially-available SoC FPGAs support adding dedicated memory controllers in the FPGA fabric, built from programmable logic. The disadvantage is that these soft controllers compete for FPGA resources with other application logic. Constructing and closing timing on soft memory controllers also takes away valuable design time which could be spent developing more valuable proprietary IP.

Table 8. Dedicated Hard Memory Controllers and Soft Memory Controllers Exclusively for FPGA Applications

Function/Feature	Altera SoC FPGA	Vendor B	Vendor C
Hard Memory Controller(s) in FPGA Fabric	1 to 3, depending on device	Not available	Not available
Soft Memory Controller(s) in FPGA Fabric	Yes, uses FPGA logic	Yes, uses FPGA logic	Yes, uses FPGA logic

System Reliability and Flexibility

Highly integrated SoC FPGAs also help create more reliable systems. Two important aspects help differentiate between the available SoC FPGA devices.

- How much memory protection is available in the system?
- How does the SoC FPGA respond to software bugs?

Protecting Memory Contents with ECC

The need for detecting, correcting, and monitoring errors is a growing trend in designs today. As memory sizes and densities continue to increase, so does the need and importance for error checking and correction. Most modern systems include dedicated hardware to help ensure data integrity.

For more information, refer to the *Error Correction Code in SoC FPGA-Based Memory Systems* white paper.

From an SoC FPGA perspective, this includes error correction code or ECC protection—not only as part of the memory controller, but also integrated within the processor's on-chip memories, caches, and peripheral buffers. ECC circuitry makes a system more robust and resilient against unexpected data errors or corrupted data.

While an immediate and essential reaction is to add ECC to the system main memory for applications where data integrity is critical, it is important that everything that can be protected is protected. In addition to main memory, it is also important to ensure that the L2 cache and on-chip RAM are also ECC protected. This is another area where architecture matters. A well-thought-out architecture considers every step in the data transfer path and includes appropriate protection at each step. Unless it is built into the device, ECC protection is incredibly difficult and expensive to add.

Table 9 summarizes the ECC circuitry implemented throughout the system.

Table 9. ECC in SoC FPGAs

	Altera SoC FPGA	**Vendor B**
L1 Cache	Parity (part of Cortex-A9 implementation)	Parity (part of Cortex-A9 implementation)
L2 Cache	Yes	No ECC
External DDR Memory Controller (Single Error Correction and Double Error Detection**)**	x16 x32	x16 only
On-Chip RAM	Yes	Parity, No ECC
Quad SPI Controller	Yes	No ECC
NAND Controller	512 byte ECC sector size (4, 8, or 16 bit correction) 1,024 byte ECC sector size (24 bit correction)	1 bit hardware support with software assist
SD/MMC/SDIO Controller	Yes	No ECC
DMA Controller	Yes	No ECC
10/100/1G Ethernet Controller	Yes	No ECC
USB 2.0 OTG Controller	Yes	No ECC

The L1 caches are integral parts of the ARM Cortex-A9 processor. For performance reasons and due to their smaller size, the L1 caches as implemented by ARM use parity to detect single-bit errors, but do not include full ECC.

Another area worth highlighting is the NAND flash. NAND flash is useful for file system storage, but somewhat less reliable than NOR flash. Therefore, ECC on the NAND flash is useful to eliminate errors. Earlier versions of NAND flash memory controllers include single-bit ECC protection. While this is of some benefit, the CPU must get involved for anything beyond that, which can result in a significant performance impact since the CPU must manage ECC in software. More modern NAND memory controllers include multi-bit ECC protection. Specifically, the NAND flash controller implemented in the Altera SoC FPGA includes 4, 8, and 16 bit correction for 512 byte sector sizes and 24 bit correction for 1,024 byte sector sizes. This provides greater, needed protection without diminishing performance.

Memory Protection for Shared Memory

Memory protection is a feature often associated with more advanced processors. Whether it is called a memory management unit (MMU) or memory protection unit (MPU), the processor's memory protection unit prevents errant or illegal processor transactions from reading or corrupting other memory regions. In the Cortex-A9 processor, ARM extends this protection concept with TrustZone® technology, which provides a system-wide approach for security-sensitive systems.

Some SoC FPGAs extend memory protection to the FPGA. Why is memory protection from or for the FPGA needed? The processor and FPGA can share a single external DDR memory interface in order to save cost, reduce board space, or save power. What if the custom FPGA logic accidentally overwrites a section of memory belonging to the processor's data, application code, or operating system (OS) kernel? This may cause a system fault or vector the processor off in the wrong direction.

To prevent this from happening, specific memory regions may be dedicated to the operating system and embedded software applications while other memory regions may be dedicated to FPGA-based functions, as shown in Figure 7. Via memory protection, the FPGA-based functions are prevented from corrupting the operating system or embedded software regions.

Figure 7. DDR Memory Protection Where Processors and FPGA Share a Common Memory

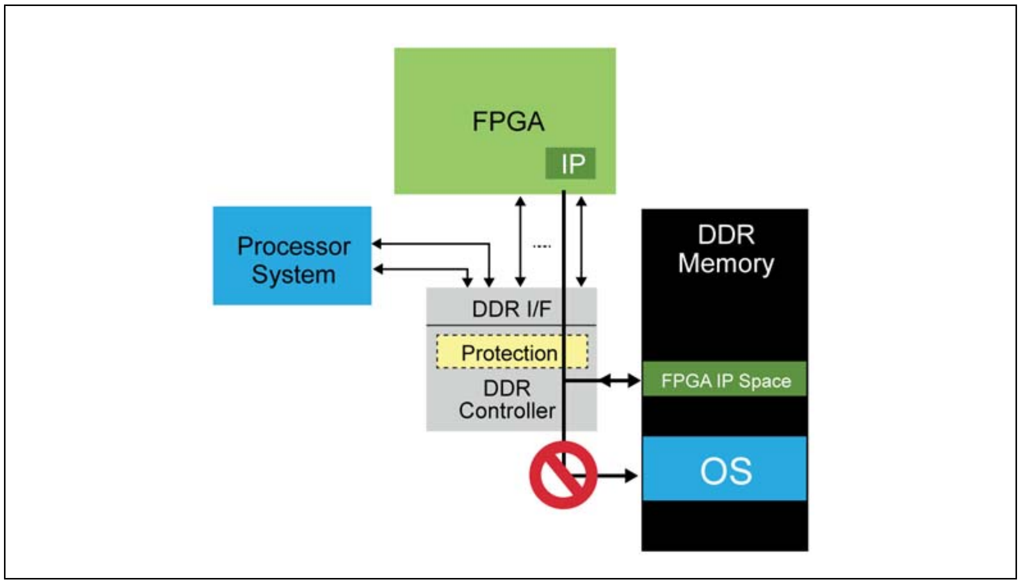


Table 10 summarizes the memory protection for FPGA accesses to external memory.

Table 10. Memory Protection for FPGA Accesses to External Memory

Function/Feature	Altera SoC FPGA	Vendor B
TrustZone Security	Yes	Yes
TrustZone Region Size Granularity	1 MB boundary	64 MB boundary
Memory Protection	20 user-definable protection rules. Each rule defines TrustZone Address range Master ID range Port range (mask) Inclusive/exclusive	TrustZone

Both SoC FPGAs support ARM's TrustZone security features; however, the Altera SoC FPGA protects regions with finer granularity, down to 1 MB. Furthermore, the Altera SoC FPGA supports 20 user-definable protection rules for a specific region. This allows finer tuning and more precise control, making it possible to prevent FPGA masters from accessing undesired regions.

Watchdog Reset and Its Effect on FPGA Logic

Watchdog timers prevent errant software from disabling a system. If the processor crashes in previous-generation two-chip processor plus FPGA solutions, the FPGA continues to operate while the processor's watchdog timer resets the processor and the system recovers as gracefully as possible. A properly architected SoC FPGA supports the same "independent" behavior plus provides the option to reconfigure the FPGA, if desired. It should not, however, mandate the FPGA to reconfigure in all cases unless that is the desired behavior, as specified by the system designer. In many cases, it may be critical for the FPGA logic to continue to monitor and respond to external stimuli while the processor resets itself. Therefore, it is important to inspect how the FPGA reconfiguration is handled in this circumstance.

As shown in Table 11, the reset circuitry in Altera SoC FPGAs matches historical usage. The reset circuitry for the processor and FPGA operate independently, although both optionally communicate reset events to the other. The developer decides how the FPGA portion should respond to a CPU reset, either by simply resetting portions of the configured FPGA logic, by completely reconfiguring the FPGA, or by completely ignoring it. In the case of SoC FPGA Vendor B, the FPGA logic is always reconfigured when a CPU reset occurs.

Table 11. CPU Reset in SoC FPGAs

Function/Feature	Altera SoC FPGA	Vendor B
FPGA Response to CPU Reset	User defined: Reset flip-flops in FPGA logic as specific in user design, or Reconfigure the FPGA logic, or No response	FPGA ALWAYS reconfigured

Fail-Safe Booting and Configuration

Being fully-programmable, single-chip systems, SoC FPGAs must successfully boot the processor and configure the FPGA before becoming fully functional. SoC FPGAs provide a fail-safe recovery method in case of a boot or configuration failure—a crucial feature for systems that support remote, in-field system updates. As summarized in Table 12, SoC FPGAs provide "fail-safe" recovery should there be a physical defect during configuration. The SoC FPGA device automatically loads an alternative configuration image if a CRC error occurs either in the configuration header or in the configuration image itself.

Altera SoC FPGAs provide additional fail-safe recovery for other logical defects. After the Altera SoC FPGA successfully boots, the boot loader software sets a bit indicating a successful configuration. However, if the boot loader fails to set the bit, then a watchdog timer triggers a warm reset to restart the boot process. When the Altera SoC FPGAs restarts the boot process, the processor sees that the previous boot attempt failed and chooses the last known good image.

Table 12. Fail-Safe Processor Boot/FPGA Configuration in SoC FPGA Devices

Function/Feature	Altera SoC FPGA	Vendor B
Fail-Safe Reboot on Physical Boot Defect	Yes	Yes
Fail-Safe Reboot on Logical Boot Defect	Yes	No

Flexibility

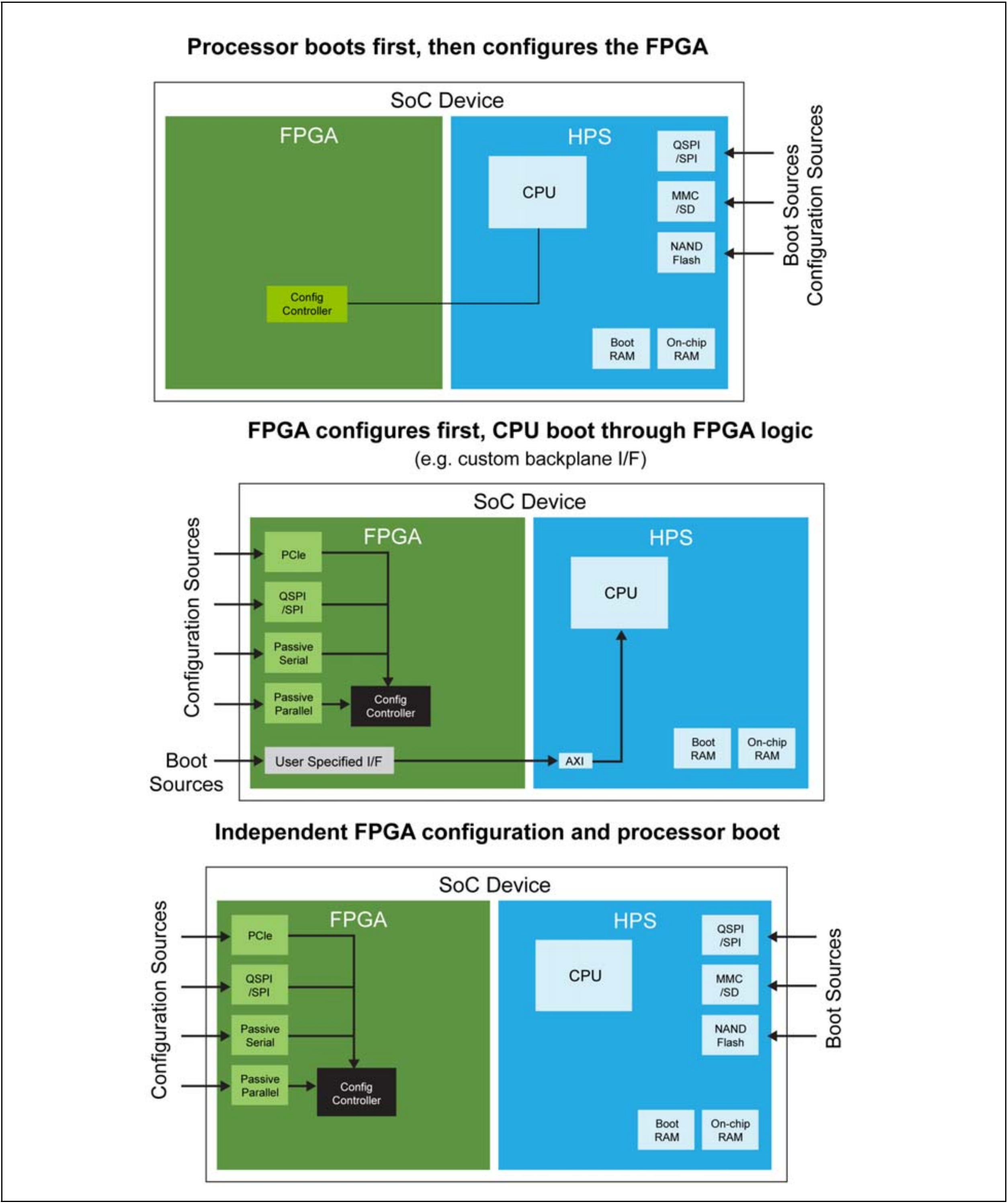
Flexibility is a common reason that many designers use FPGAs in the first place. Fully-programmable SoCs simply extend design flexibility to the system level. This section highlights three architectural details to consider when choosing an SoC FPGA:

- Processor boot and FPGA configuration options
- On-chip FPGA interfaces
- Common package footprints

Multiple Options for Processor Boot and FPGA Configuration

The need for flexibility begins at boot. There are three options in SoC FPGAs for booting the processor and configuring the FPGA as illustrated in Figure 8.

Figure 8. SoC FPGA Processor Boot and FPGA Configuration Options



All SoC FPGAs support the processor-like “CPU first” method (Figure 8, top), where the processor boots first and then configures the FPGA under software control. This mode functions like a normal processor boot, except that the processor configures the FPGA as a big “peripheral” device. The advantage of this mode is that it adheres to the traditional approach of bringing up the processor first, and existing boot code may translate easily; possible disadvantages of this approach would be if the system has configuration time constraints that won’t tolerate a delay while the processor boots or there are advantageous functions the FPGA can perform while the processor is still booting.

The second option (Figure 8, middle) has the FPGA configure first and then boots the CPU through FPGA logic. One use of this method could be to have the FPGA examine and secure the system before allowing the processor to boot, or various other secure boot modes. Another case would be to use the FPGA to bring up a custom backplane which could then be used to boot the processor.

The third option (Figure 8, bottom) is completely independent processor boot and FPGA configuration mechanisms. In this example, the processor boots from one of its flash memory sources. Independently, the FPGA configures from one of its data sources. Consequently, the FPGA subsystem can configure fast enough—in as little as 13 ms—to allow a PCI Express® (PCIe®) interface to configure the remainder of the FPGA.

Table 13. Processor Boot and FPGA Configuration Options in SoC FPGAs

Function/Feature	Altera SoC FPGA	Vendor B
CPU Boots First, CPU Configures FPGA	Yes	Yes
FPGA Configures First, CPU Boots through FPGA Fabric or Backplane	Yes	No
CPU Boots Independently and FPGA Configures Independently	Yes	No

Table 13 shows the different boot modes supported by two of the SoC FPGAs. Currently, the Altera SoC FPGA is the only ARM Cortex-A9 processor-based SoC FPGA that was designed to support all three of these options.

Multiple Boot Images

Many SoC developers prefer to store their boot images in quad SPI flash due to its inherent reliability (NOR technology), relatively low cost, and minimal I/O requirements. For systems in which the processor is responsible for configuring the FPGA, the flash boot image will contain both hardware and software content including:

- CPU boot code
- Operating system (OS)/real-time operating system (RTOS)
- Application code and data
- FPGA configuration

Frequently multiple “boot images” are desired: One to hold the factory default image and at least one to hold system updates. The factory default image is always stored in case the update fails to load properly. The system then automatically reverts back to a known good image and the update can be retried.

Estimates for full boot images based on “minimal” and “substantial” software requirements, and hardware images for small, medium, and large FPGA densities are provided in Table 14.

For quad SPI devices, the amount of storage needed may be an issue depending on the SoC vendor selected. Altera provides a quad SPI interface that support up to a 4 GB address range, and up to four chip selects. Vendor B’s quad SPI supports a 16 MB address range with up to two chips selects, limiting total boot image size to 32 MB.

Table 14. Boot Image Size Requirements and Mapping to Quad SPI Devices

Software Requirements	**Minimal**			**Substantial**		
User Space Code (MB)	5	5	5	50	50	50
Linux Kernel (MB)	3	3	3	5	5	5
Boot Code (MB)	0.5	0.5	0.5	0.5	0.5	0.5
FPGA Density	**Small**	**Medium**	**Large**	**Small**	**Medium**	**Large**
FPGA HardwareImage (MB)	2.4	6.1	14.4	2.4	6.1	14.4
Total Storage Required						
Single Image (MB)	11	15	23	58	62	70
Dual Image (MB)	22	29	46	116	123	140
Altera SoC FPGA						
Single Image (# Quad SPI Devices)	1	1	1	1	1	1
Dual Image (# Quad SPI Devices)	1	1	1	1	1	2
Vendor B						
Single Image (# Quad SPI Devices)	1	1	2	N/A	N/A	N/A
Dual Image (# Quad SPI Devices)	2	2	N/A	N/A	N/A	N/A

As shown by the table, Altera SoC FPGAs can support multiple large boot images. Vendor B’s SoC FPGAs are limited in terms of the size and number of boot images they can handle. Estimates based on maximum quad SPI device size of 1 Gb (128 MB).

On-Chip FPGA Interfaces

Flexibility also extends to on-chip FPGA interfaces. There are times when an application demands a feature-rich, standards-based interface; and times when something simple or customized is all that is needed.

For those applications that demand advanced features, SoC FPGAs use ARM’s AXI to connect the processor, hard peripherals, and FPGA logic. The AXI standard provides a fast and wide interface using a proven industry standard. But what about the IP cores that don’t need all the feature richness of AXI? What if scalability is more important? At one end of the spectrum a custom, 1000-wire interface may be what is needed; at the other end of the spectrum, a single wire may be needed to blink an LED or read a switch.

To address this need for variation, in addition to the AXI interface, the Altera SoC FPGAs support the Avalon® Memory-Mapped (Avalon-MM) interface and Avalon Streaming (Avalon-ST) interface. These scalable Altera interface standards offer the right fit for less demanding or other specific functions. This enables the IP designer to choose the optimum interface for each function. This also allows existing Altera FPGA customers to continue to use IP built on these interfaces without a forced migration to AXI for IP that would not benefit.

For more information about Avalon interfaces, refer to the *Avalon Interface Specification*.

Table 15 summarizes the FPGA IP interfaces.

Table 15. FPGA IP Interfaces Supported in SoC Devices

Function/Feature	Altera SoC FPGA	Vendor B
FPGA IP Interfaces	AXI Avalon-MM Avalon-ST	AXI

Common Footprint Density/Transceiver/Feature Migration

Common package footprints provide additional flexibility during design, development, and deployment. Altera designed its SoC FPGAs so that developers can easily migrate a design between devices with different gate densities, but available in the same package footprint as highlighted in Figure 9. Additionally, developers can migrate between devices with and without transceivers within a common package footprint. For additional cost reduction, the non-transceiver versions are available with either a dual-core or single-core processor. These options allow a single printed circuit board platform to meet different cost and feature objectives.

Figure 9. Density/Package Migration within Altera Cyclone V SoCs

Altera: Devices without Transceivers

Device Family	KLE	484	672	896	Pins
		19 x 19	23 x 23	31 x 31	Package Size
		0.8 mm	0.8 mm	1.0 mm	Ball Pitch
Cyclone V SoC FPGA	25	66	138		I/O Count
	40	66	138		
	85	66	138	288	
	110	66	138	288	
HPS I/O[1]		161	188	188	

Vendor B: Devices without Transceivers

Device Family	KLC	225	400	484	Pins
		13 x 13	17 x 17	19 x 19	Package Size
		0.8 mm	0.8 mm	0.8 mm	Ball Pitch
A	28	54	100		I/O Count
	85		125	200	
PS I/O[1]		86	130	130	

Altera: Devices with Transceivers

Device Family	KLE	672	896	896	1,152	1,517	Pins
		23 x 23	31 x 31	31 x 31	35 x 35	40 x 40	Package Size
		1.0 mm	1.0 mm	1.0 mm	1.0 mm	1.0 mm	Ball Pitch
Cyclone V SoC FPGA	25	145, 6					I/O Count (I/O, Transceiver)
	40	145, 6					
	85	145, 6	288, 9				
	110	145, 6	288, 9				
Arria V SoC FPGA	350			178, 12	350, 18	528, 30	
	460			178, 12	350, 18	528, 30	
HPS I/O[1]		188	188	216	216	216	

Vendor B: Devices with Transceivers

Device Family	KLC	485	484	676	900	1,156	Pins
		19 x 19	23 x 23	27 x 27	31 x 31	35 x 35	Package Size
		0.8 mm	1.0 mm	1.0 mm	1.0 mm	1.0 mm	Ball Pitch
A	74	150, 4					I/O Count (I/O, Transceiver)
B	125	150, 4	163, 4	250, 8			
	350			250, 8	362, 16	362, 16	
	444				362, 16	362, 16	
PS I/O[1]		157	130	130	130	130	

Note:

1. Includes DRAM dedicated I/O.

System Cost

Almost every system shipping today is under increasing cost pressure. While SoC FPGAs are an innovative new product with advanced features, Altera designed its SoC FPGAs with both component and system costs in mind. A single SoC FPGA may cut costs by up to 50% below the individual components it replaces, and it will likely also reduce system costs. Table 16 compares system cost factors.

When considering cost for SoC FPGAs, it is important to look at three key areas:

- How many equivalent functions are already integrated in the SoC?
- Does the application require high-speed transceivers? If so, how many?
- What is the associated power supply cost?

Integrated Functionality

Just how integrated is the SoC FPGA solution? Depending on the application, a single SoC FPGA might contain the system equivalent of the processor, all of its peripherals, multiple DSPs, plentiful on-chip memory, high-speed transceivers, clock management, and copious custom logic. Regardless, there are plenty of questions to ask.

- Does it offer both a single-core and a dual-core processor version?
- In addition to the ARM processor cores, what peripherals are integrated?
- How many hard memory controllers does it have?
- Does it have integrated phased-lock loops (PLLs)?
- Are there ways you can save cost with configuration options?
- Does the SoC FPGA include hardened memory controllers for FPGA applications or do you need to allocate additional FPGA logic for the controllers?
- Are there common package footprints to allow for platform cost optimization?

High-Speed Transceivers

High-speed transceivers are another critical feature that can significantly impact the cost of a design. Altera SoC FPGAs include high-speed transceiver options across the full range of the product line. Specifically, high-speed transceivers are available as an option in the low-end, entry-level devices as well as the largest full-featured devices. High-speed transceivers are critical for applications like PCIe. Otherwise, an external interface component is needed which adds to the system bill of materials (BOM). On the other hand, some embedded designs may not require high-speed transceivers, and Altera offers SoC FPGA variants that do not include high-speed transceivers to reduce the SoC FPGA component cost.

Power Supply Cost

The number and capacity of required voltage rails significantly impact the cost and complexity of your design. All SoC FPGAs require multiple voltage rails, but some require fewer rails than others. Also, some SoC FPGAs require stringent power-on and power-off sequencing controls that mandate more sophisticated—and thereby expensive—power supplies. In particular, power-off sequencing becomes difficult due to all of the potential power-loss conditions that might occur. Ideally, it is best to avoid power-on or power-off conditions, especially if those requirements affect the long-term reliability of the device. Altera SoC FPGAs do not have any power-on or -off sequencing requirements. Refer to Table 16.

Table 16. SoC FPGA System Cost Factor Comparison

Function/Feature	Altera SoC FPGA	Vendor B
Single- and Dual-Core Processor Option	Yes	No (Dual-core only)
Hardened Memory Controller in Both Processor System and FPGA Fabric	Yes (1 in processor system, up to 3 in FPGA)	No (1 in processor system, none in FPGA)
All Devices with High-Speed Transceivers (necessary for integrated PCIe)	Yes	No (2 of 6 without high speed transceivers)
Integrated Analog Mixed Signal	No	Yes (2 x 12-bit, 1 MSPS ADCs)
Relative Traffic Prioritization	Yes	Yes
Spectrum of Logic Densities	25, 40, 85, 110, 350, 460 KLE	28, 74, 85, 125, 350, 444 KLC
Package Migration	Yes	Limited
Power-Off Sequencing Requirement	No	Yes (additional external circuitry required)

Power

Power savings has become an increasing factor in many designs, if not the driving factor.

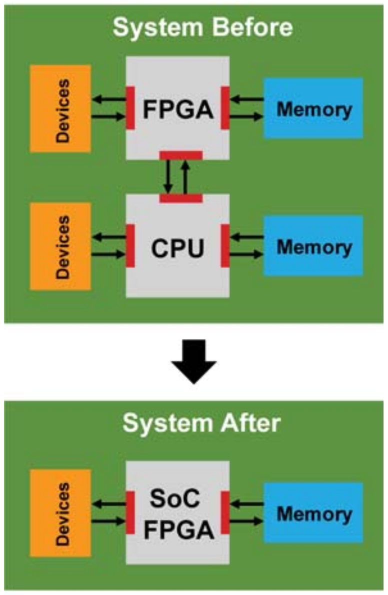
When selecting between SoC FPGA devices, there are three important areas to examine relative to power:

- Integration
- Power-saving modes
- Power-on/off sequencing requirements

Power Savings via Integration

As illustrated in Figure 10, simply integrating the processor and FPGA components into a single SoC FPGA potentially reduces system power by 10% to 30%. I/Os carrying signals between devices, often at higher voltages, are one of the most power-hungry functions in an application.

Figure 10. Integrating Processor and FPGA into a Single SoC FPGA Reduces Power-Hungry, Inter-Chip I/O Connections



As mentioned in the "DDR Memory Controller Performance" on page 8, a smarter memory controller also saves power. Because it more efficiently transfers data, a smart memory controller operates at a lower clock frequency without sacrificing memory bandwidth. For example, as shown in Figure 5, it is possible to achieve comparable or better performance with 400 MHz DDR3 using a smart memory controller vs. 533 MHz DDR3 with a traditional memory controller. This extra efficiency and lower clock rate save crucial milliwatts from the system power budget.

Power-Saving Modes

SoC FPGAs employ a variety of power- and cost-saving power features. Because a majority of the power is consumed in the FPGA portion of the device, it is important that the processor system and FPGA have separate, independent power planes. To save power, the processor can then place the FPGA in a low-power mode via software control.

Additionally, the processor can control other power-saving features including:

- Turn off the clocks to currently unused functions (clock domain gating)
- Set PLL and clock divider controls to scale clock frequencies according to current processing needs
- Place the processor into one of the available sleep modes, later waking the processor using an interrupt
- Place the DDR memory controller into one of its low-power modes

For additional information, see the *Achieving Lowest System Power with Low-Power 28 nm FPGAs* white paper.

Power-On/Off Sequencing Requirements

In order to preserve device reliability or guarantee certain power-up states, silicon vendors may provide specific power-on and power-off sequencing requirements as outlined in Table 17. While power-on sequencing requirements are fairly common, power-off specifications are rare as a means of protecting the device. The implication is that additional circuitry must be added to the power supply, or the system manufacturer could face long-term reliability concerns.

For devices with power-off sequencing requirements, precautions must be taken to avoid the failure of an individual power rail, thus causing a violation of the specifications. This requires comparative analog circuitry to monitor the rails, and appropriate protective circuitry must be added. To ensure the proper power-off sequence, sufficient energy storage must also be supplied.

Altera SoC FPGAs are built with internal device protection such that any order of power-on or power-off sequencing is acceptable. Altera does provide a recommended power-on sequence, but as a guide to system power supply designers to help minimize cost, it does not have any reliability implications. Other SoC FPGA vendors do have power-on and power-off sequencing, which if violated repeatedly, can lead to long-term reliability concerns for the device.

Altera SoC FPGAs are guaranteed to bring up I/Os in a tri-state, avoiding any board-level driver contention. Other SoC FPGA vendor devices cannot guarantee this if power-on sequencing requirements are violated.

Furthermore, Altera SoC FPGAs support "hot socketing," where the device can be inserted into a board that is already under power. This functionality is not specified from other SoC FPGA vendors.

Table 17. SoC FPGA Power-On and Power-Off Sequencing Requirements

Function/Feature	Altera SoC FPGA	Vendor B
Power-On Sequence	None	Sequencing is required when I/O banks drive 3.3 V peripherals, in order to maintain device reliability
Power-on Sequence for I/O Bring-Up in Tri-State	None	Power-on sequence is required to assure I/Os come up in tri-state
Power-Off Sequence	None	Sequencing is required when I/O banks drive 3.3 V peripherals, in order to maintain device reliability
Hot Socketing Capability	Yes	Not specified

Future Roadmap

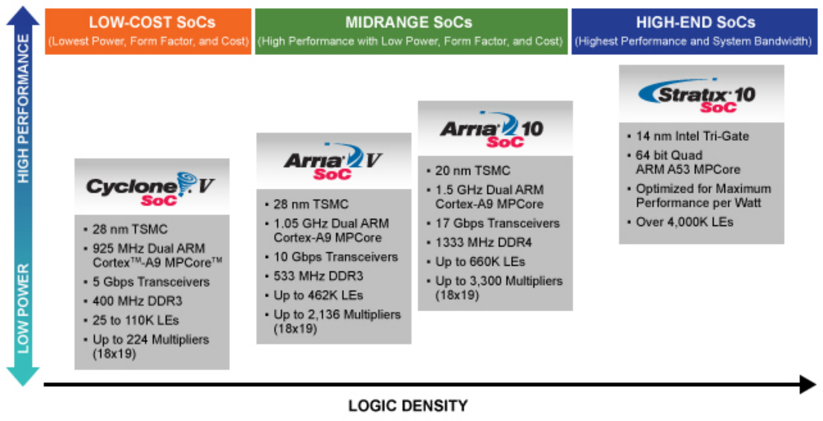
Selecting a new processor architecture is a major decision. It is important to evaluate whether the vendor's product roadmap will meet future application requirements, allow system differentiation, and offer system competitive advantages for the long run. Given the large software investment involved, it is important that the software base migrates easily to future generations. Thus, it is critical to find out not just what the SoC vendor promises for the next-generation product, but to ask questions like:

- What level of investment are you making in this product line?
- How will you give my system design a competitive edge in the future?
- What is your tools roadmap?

Altera's Three-Generation Processor Roadmap

In order to meet the processing requirement needs for the applications SoC FPGAs are targeting (communications infrastructure, industrial, automotive, high-performance computing, military, aerospace, medical, multifunction printers and others), Altera has developed a three-generation processor roadmap as shown in Figure 11.

Figure 11. Altera SoC Product Portfolio Roadmap



LOW-COST SoCs
(Lowest Power, Form Factor, and Cost)
MIDRANGE SoCs
(High Performance with Low Power, Form Factor, and Cost)
HIGH-END SoCs
(Highest Performance and System Bandwidth)
HIGH PERFORMANCE
LOW POWER
Cyclone V SoC
• 28 nm TSMC
• 925 MHz Dual ARM Cortex™-A9 MPCore™
• 5 Gbps Transceivers
• 400 MHz DDR3
• 25 to 110K LEs
• Up to 224 Multipliers (18x19)
Arria V SoC
• 28 nm TSMC
• 1.05 GHz Dual ARM Cortex-A9 MPCore
• 10 Gbps Transceivers
• 533 MHz DDR3
• Up to 462K LEs
• Up to 2,136 Multipliers (18x19)
Arria 10 SoC
• 20 nm TSMC
• 1.5 GHz Dual ARM Cortex-A9 MPCore
• 17 Gbps Transceivers
• 1333 MHz DDR4
• Up to 660K LEs
• Up to 3,300 Multipliers (18x19)
Stratix 10 SoC
• 14 nm Intel Tri-Gate
• 64 bit Quad ARM A53 MPCore
• Optimized for Maximum Performance per Watt
• Over 4,000K LEs
LOGIC DENSITY

The roadmap begins with the 28 nm Cyclone V and Arria V SoC FPGAs which are the main focus of this paper. In the second generation at 20 nm, the Arria 10 SoC FPGA processor subsystem remains the same, consisting of a dual-core ARM Cortex-A9 MPCore processor. The dual-core ARM A9 maintains software compatibility for ease of software migration, while adding an 87% processor performance boost above the first generation due the benefit of 20 nm process technology. Enhancements in the areas of security and memory support also will be added to the second generation. The third-generation SoC FPGA processor subsystem pushes the bounds at the high end even further with the integration of a quad-core ARM Cortex-A53 processor in the Stratix 10 SoC FPGA. The 64 bit A53 adds a substantial performance boost while still being conscientious of power. If desired, two of the four cores can be run in a 32 bit mode to maintain compability with second-generation software, while the other two cores can be run at 64 bits for new applications.

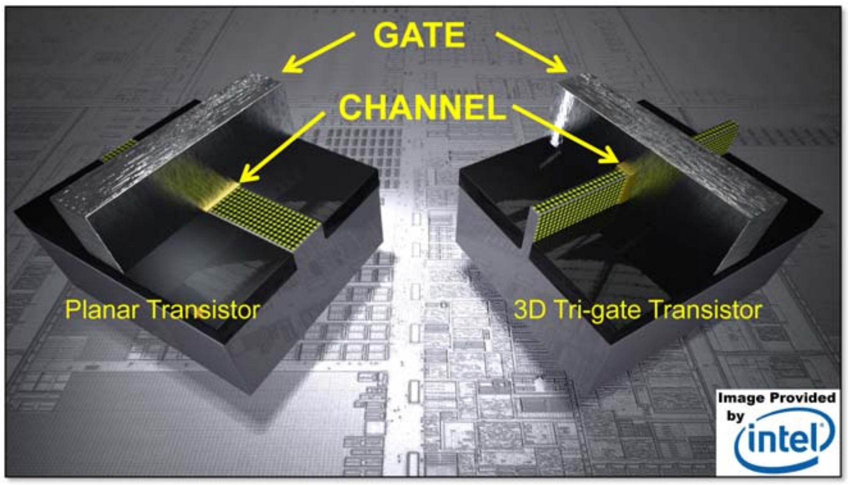
The Foundation: Silicon Process Technology

The underlying foundation of all silicon component roadmaps is the silicon process technology. Today, most SoC FPGAs are built on 28 nm silicon processes. The next major advancement in process technology is FinFET technology.

FinFET Technology

FinFET transistors are poised to revolutionize the semiconductor industry by moving from two-dimensional design to three-dimensional design by flipping the channel on its side, as shown in Figure 12.

Figure 12. FinFET's Three-Dimensional Structure Reduces Power, Leakage, and Area

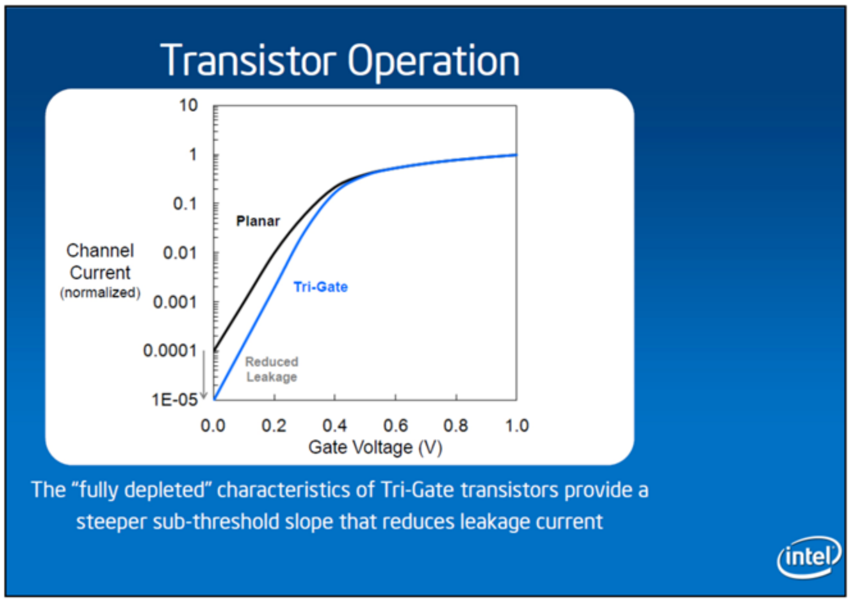


Tri-Gate Surrounds Channel on Three Sides To Increase Channel Performance and Reduce Power

(Graphic courtesy of Intel Corp.)

The benefit of this new structure is higher density, lower leakage, and reduced active power. As shown in Figure 13, Intel's three-dimensional FinFET design techniques, called "Tri-Gate," provide an order of magnitude lower leakage than traditional two-dimensional planar technology.

Figure 13. FinFET Design Reduces Leakage Current



(Graphic courtesy of Intel Corp.)

Intel has led the way in FinFET technology. Intel's first generation was on 22 nm, and they are now on to their second generation of "Tri-Gate" technology at 14 nm. Altera SoC FPGAs will intercept Tri-Gate technology at the 14 nm process node.

For more information on Altera's plans for implementing FinFET technology, refer to *The Breakthrough Advantage for FPGAs with Tri-Gate Technology* white paper.

Tools Roadmap

For debug and development tools, Altera has formed a long-term, strategic relationship with ARM. In December 2012, the two companies announced a unique agreement, whereby the companies jointly developed an ARM DS-5™ embedded software development toolkit with FPGA-adaptive debug capabilities for Altera SoC FPGAs. The ARM Development Studio 5 (DS-5) Altera Edition Toolkit removed the debugging barrier between the integrated dual-core CPU subsystem and FPGA fabric in Altera SoC FPGAs. By combining the most advanced multi-core debugger for the ARM architecture with the ability to adapt to the logic contained in the FPGA, the new toolkit provides embedded software developers an unprecedented level of full-chip visibility and control through the standard DS-5 user interface. This cooperation will continue moving forward by providing feature and performance enhancements to extend FPGA-adaptive debugging to the Altera future silicon roadmap, including the Stratix 10 SoC FPGA.

In parallel with these efforts, Altera has adopted the OpenCL™ standard on an FPGA to offer significantly higher performance at much lower power than is available today from other hardware architectures (CPU, GPUs, etc). Since OpenCL leverages the ANSI C language with additional extentions, development time for a heterogeneous FPGA-based system (CPU + FPGA) using the OpenCL standard has a significant time-to-market advantage compared to traditional FPGA development using lower-level hardware description languages (HDLs) such as Verilog or VHDL. Altera joined The Khronos Group in 2010 and is an active contributor to the upcoming OpenCL 2.0 specification. Altera has developed an SDK for OpenCL that provides a compiler to compile OpenCL code to HDL. The compiler takes the kernel code and generates a programming file. This programming file is then downloaded into the FPGA to run hardware acceleration or other functions.

In October 2013, Altera announced that its SDK for OpenCL is conformant to the OpenCL 1.0 standard and it is now included on the Khronos Group list of OpenCL conformant products. Altera is the only company to offer an FPGA-optimized OpenCL solution at this time, allowing software developers to harness the massively parallel architecture of an FPGA for system acceleration. Altera advancements around OpenCL and multicore, heterogenous, parallel processing will continue into the future, enhancing performance and increasing designer productivity with SoC FPGAs.

For more information about Altera's OpenCL offering for SoC FPGAs, refer to *Implementing FPGA Design with the OpenCL Standard* white paper.

Development Tools

SoC FPGAs open a wealth of possibilities for faster, cheaper, and more energy-efficient electronic products. However, the innovation in hardware must be matched by similar innovation in development and debug tools. Software ultimately determines how successful a designer will be using these devices. For broader use, software developers must find SoC FPGAs and their features to be as easy and efficient to utilize as stand-alone processors. Table 18 summarizes many of the differences between the Altera SoC Embedded Design Suite (EDS) development environment, which leverages the ARM DS-5 Altera Edition tools, and the debugging tools provided by Vendor B.

Table 18. In-System Debugging and Development Tool Features for SoC FPGA Devices (Part 1 of 2)

Function/Feature	Altera SoC EDS (with ARM DS-9 Altera Edition)	Vendor B's Debug Tools
Versions Compared	13.1	2013.3
FPGA-Adapative Debugging	Yes	No
All ARM Processor and FPGA Tools Operate Over Single USB Cable	Yes	No
Auto Display of Peripheral Registers	Yes	No
Display of VFP and Neon Registers	Yes	No
Debug: Single-Step, Watchpoints, etc.	Yes	Yes
CPU↔FPGA CoreSight Compliant Cross-Triggering	Yes	No Vendor proprietary

Table 18. In-System Debugging and Development Tool Features for SoC FPGA Devices (Part 2 of 2)

Function/Feature	Altera SoC EDS (with ARM DS-9 Altera Edition)	Vendor B's Debug Tools
CPU↔FPGA Cross-Triggering with Timestamps and Trace Data Stream	Yes ARM CoreSight™ compliant using System Trace Macrocell (STM)	No Available with purchase of additional third-party hardware and software
Processor Trace Support	Yes	No Requires additional third-party hardware and software
Trace Buffer	32 KB	4 KB
Route Trace Packets to Alternative Destinations (e.g. DRAM or high-speed transceiver)	Yes Coresight embedded trace router	No
Route Trace Packets to External Trace Probe	Yes	Yes
FPGA Information Included in ARM Trace Stream	Yes Uses ARM CoreSight STM	Yes Vendor proprietary solution
Native Linux Support for Hardware-Assisted Trace	Yes Kernel and application	No
Concurrent Multicore Debugger	Yes ARM DS-5 specifically designed for multicore systems	No
Multicore Debugging in Asymmetric Multiprocessing (AMP) Applications	Yes	Yes
Multicore Debugging with Symmetric Multiprocessing (SMP) Operating Systems	Yes	No
Linux Kernel Awareness	Yes	No
Non-Intrusive Code Profiling	Yes ARM Streamline including processor, FPGA, and power profiling; refer to ds.arm.com/ds-5/optimize/ (see Figure 16)	No
Semi-Hosting Support (communication between host and ARM processors over JTAG) Refer to: infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0471c/Bgbjjgij.html	Yes	No
FPGA Logic Analyzer	SignalTap™ II Logic Analyzer	Yes
Bare-Metal Application Development	Modifiable hardware libraries with friendly, open BSD licensing	Vendor proprietary BSP project build
Hardware VFP and NEON Compiler Support	Yes (Linux) Support for Bare-Metal compiler planned for 14.0	Yes (Linux/Bare Metal)

Software development has long dominated the project schedule. The hybrid nature of a processor plus FPGA on the same device adds a new dimension to development. Careful consideration must be given to how this dimension will affect the project schedule, the engineering team's learning curve, and past investment in software tools.

Development Tool Challenges

The "FP" in "FPGA" means "field programmable," which means that the hardware engineering team will program the hardware during the course of the development project, and this hardware can even be reconfigured during run time. This field programmability leads to two important software implications compared to traditional SoC devices:

- The CPU software and the FPGA programs will be developed and debugged alongside each other. This is a big departure. Previously, the embedded software was developed on top of fixed hardware with a traditional SoC.
- Since the FPGA hardware definition is user-defined, the software development tools and board support packages (BSPs) that ship with the SoC FPGAs will support all the standard peripherals for the SoC FPGA. They are not pre-loaded with any memory map information or debugging hooks for the FPGA-based peripherals the hardware team may create.

These are very important implications that which demonstrate the idea that architecture matters applies as much to the software as to the hardware.

ARM Compatibility a Given; FPGA Implementation a Difference

First and foremost, it is critical that the tools for these new devices be ARM-compatible and leverage the ARM ecosystem. All the SoC FPGAs currently on the market leverage ARM processor IP, which generally includes support from the vast ecosystem for ARM processor software development tools. However, each vendor deals differently with the added dimension of the FPGA portion of the device. These differences particularly impact the following areas:

- Whole-chip debug
- Profiling CPUs and FPGA
- Multicore debug
- Standard tools and flows

Whole-Chip Debugging

Debugging applications on a stand-alone processor is a well understood problem with a mature software ecosystem that supplies proven solutions. With SoC FPGAs, the SoC is no longer pre-defined, and consequently the debugging tools must support a number of new constructs:

- Additional user-defined peripherals implemented in the FPGA
- Software functions that include hardware acceleration blocks implemented in the FPGA
- Custom logic blocks in the FPGA that implement proprietary algorithms

Traditional software debugging tools were never designed to touch custom-built functions in the FPGA, and the traditional FPGA tools have no hooks back to the software tools. Until recently, there was a virtual wall between the CPU and FPGA subsystems. To break through this debugging barrier, a toolset must provide:

- Whole-chip visibility to the processor and the FPGA subsystems
- Cross-triggering and in-system trace between CPU and FPGA subsystem
- System-wide monitoring for software, CPU hardware, and FPGA hardware events
- Performance profiling

The capabilities listed above represent a new era in the debugging world, where a debugging tool can adapt itself to the debugging target. Ideally, the debugging tool can be almost as flexible as the FPGA, giving developers the best of both worlds—proven and adaptive.

FPGA-Adaptive Debugging

"FPGA-adaptive" means that the software debug tools automatically adapt to changes in the hardware due to changes in the FPGA logic. In other words, as the hardware engineers make iterations to the FPGA, the software debug view should just update automatically—with any FPGA-based peripheral automatically appearing in the register view.

What functions are required in an FPGA-adaptive debugger? The FPGA-resident hardware design must be visible in the programmer's model, viewable by its register set in the register window of the debugger. When the hardware team changes or upgrades the FPGA logic, the software debugger should be able to pick up the difference and make the new hardware visible in the debugger. With this capability, the user can now see and control the FPGA subsystem as easily as the CPU subsystem.

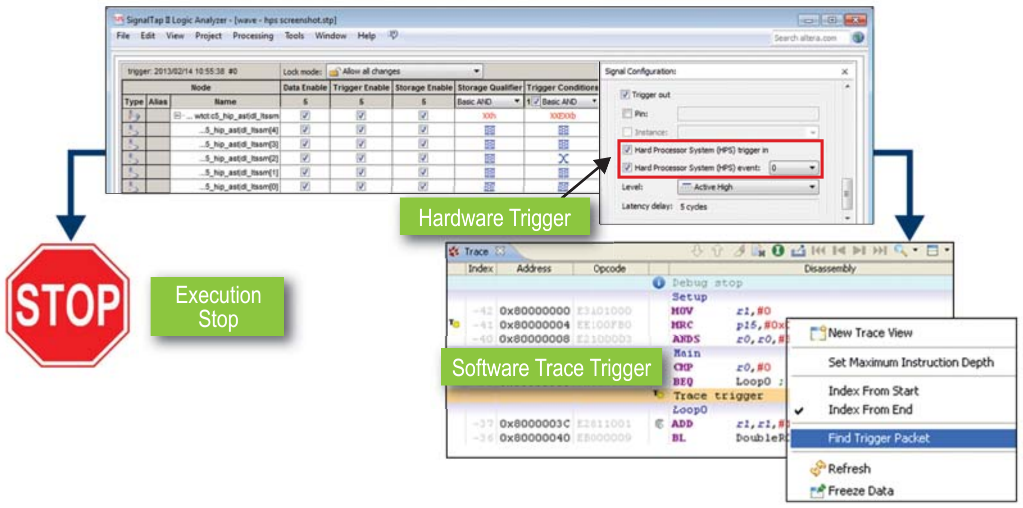
Single Debug Cable for ARM and FPGA Development

Though debug tools and associated cables are available on the software side for the ARM processor and on the hardware side using the FPGA vendor's tools and cable, using both at the same time would mean using two cables and two sets of independent tools. On a practical level, most developers want a single, low-cost JTAG cable that supports both the hardware and software tools. This way, both the hardware and software teams can work together, with all of their tools, all at the same time – paving the way for true collaboration.

Cross-Triggering Between Hardware and Software

When a bug makes an unwelcome appearance, the development team invariably wants to determine whether it is a hardware or software issue. Finding the cause is much easier if the processor subsystem and FPGA subsystem can cross-trigger from code to waveform or from waveform to code, enabling the development team to find and track how and why a particular condition occurred in the system. Figure 14 shows a cross-triggering example from the ARM DS-5 Altera Edition Toolkit software. Cross-triggering, trace, and global time-stamping are valuable features for IP verification, custom driver development, and the system integration portion of your project.

Figure 14. Cross-Triggering from the Hardware World to the Software World



```
Index  Address     Opcode     Disassembly
                              Debug stop
                              Setup
-42    0x80000000  E3101000   MOV    r1,#0
-41    0x80000004  EE100F80   MRC    p15,#0x
-40    0x80000008  E2100003   ANDS   r0,r0,#
                              Main
                              CMP    r0,#0
                              BEQ    Loop0 ;
                              Trace trigger
                              Loop0
-37    0x8000003C  E2811001   ADD    r1,r1,#
-36    0x80000040  EB000009   BL     DoubleR
```

In contrast, Vendor B's debugging tools appear to support cross-triggering but using a proprietary, non-CoreSight mechanism. While the proprietary scheme might allow cross-trigger between both the processor and FPGA hardware, it lacks some of the built-in additional visibility provided by ARM's DS-5 environment, such as global timestamping. Furthermore, Vendor B has a much smaller trace buffer (4 KB vs. 32 KB in Altera SoC FPGAs). Vendor B does optionally support some extended trace capabilities, but this requires additional third party debugging hardware and software, available at an extra charge.

Tracking and Monitoring Hardware and Software Events

Besides finding the location of a fault, it is also valuable to find out exactly how and why the system entered the faulty state. The ARM System Trace Module (STM) enables tracking of CPU-based software events. The application software can issue hardware and software event “bread crumbs” as the system executes over time to monitor system behavior and to gain deep insights into its operation. In an “FPGA adaptive” debugging environment, STM enables event monitoring of both the CPU and FPGA domains without having to stop the system.

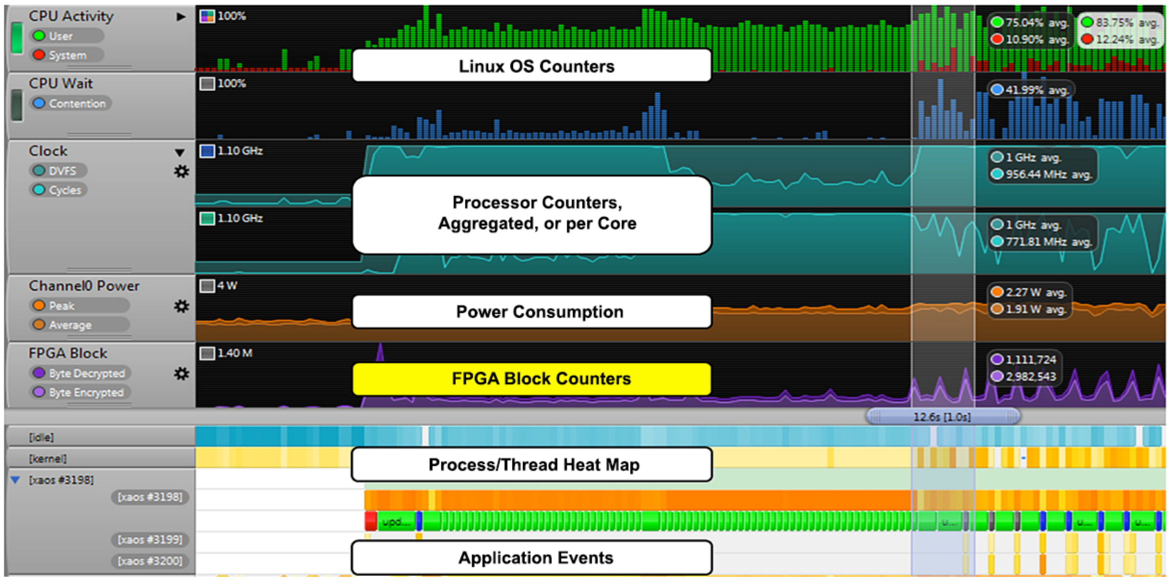
Profiling CPUs and FPGA

System profiling, part of any good debugger, helps developers find answers to common questions:

- Where are the hot spots in the system?
- Where do the CPU cores spend their time?
- How efficiently is cache being used?

To make profiling truly helpful in a processor-based system with an integrated FPGA, FPGA events must be part of the profile, too. Otherwise, without FPGA-adaptive debugging, the developer only sees and controls part of the chip! Figure 15 shows how the ARM DS-5 Streamline tool contained in the ARM DS-5 Altera Edition Toolkit enables non-intrusive profiling of both the processor and FPGA elements of the SoC FPGA.

Figure 15. ARM DS-5 Streamline Tool Enables Visibility Between SoC FPGA's Processor and FPGA



Multicore Debugging

Just as PCs have moved to multicore processors, multicore is also becoming the norm in embedded systems. As the embedded world moves to multicore, it is important that the development tools move as well. Developing software for multicore platforms is much more complicated than single core. On which core should the breakpoint be set? On which core is the software running at any particular time? These questions become critical for multicore debug.

In multicore debugging, the ability to simultaneously and independently control and monitor the processor cores is essential. In some cases, stopping both cores on a single breakpoint is desired. However, in other cases, it may be preferable to stop only one processor on a breakpoint while the other core continues to execute code. It is also valuable to have visibility to the software running on each of the cores. The debugger and analysis tools should be designed to specifically support multicore applications. In contrast, the GNU GDB-based debugger tools were originally designed in the single-core era. A GDB-based debugger works great, but only on a single core at a time. When using a GDB-based debugger on a multicore system, breakpoints can be set up across multiple cores. However, when the application software ultimately encounters one of the breakpoints, only the core that triggered the breakpoint occurred is observable. Essentially, only one core can be debugged at a time. Visibility of the other cores is lost during the debug session, which is extremely limiting for multicore debugging.

Fortunately, ARM and its ecosystem partners have responded to this multicore challenge, and have developed high-quality, powerful multicore debugging tools. When choosing an SoC FPGA, it is important to choose an SoC FPGA family that provides easy access to a true multicore debugger.

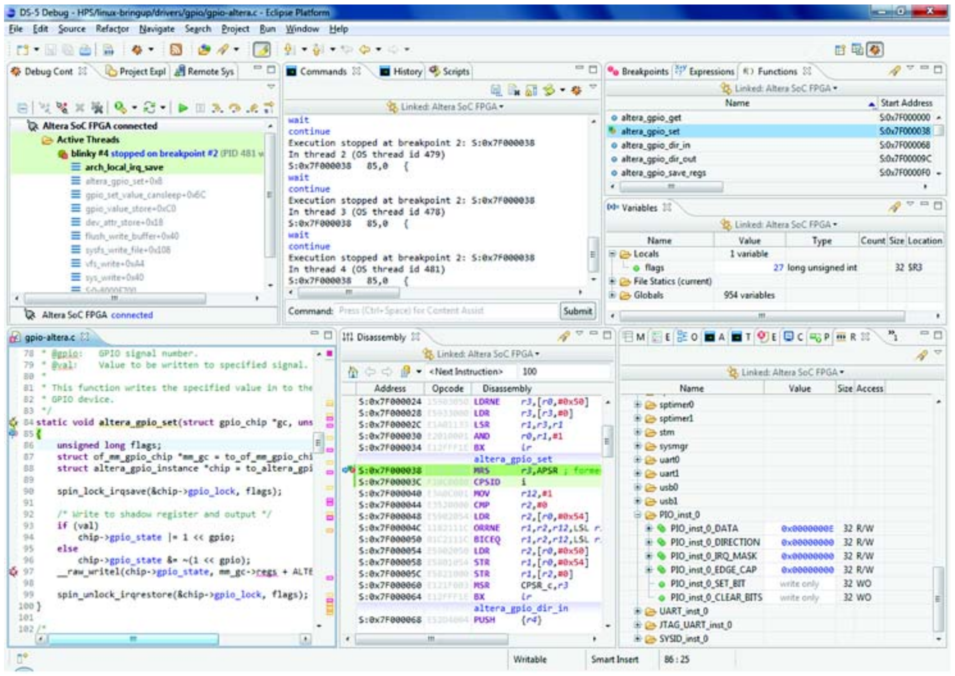
Standard Tools and Flows

For software engineers, productivity is not delivered with a new “flow” or a new “end-to-end” tool foreign to everybody. On the contrary, software engineers are generally far more productive with familiar, proven tools put into the hands of people who are already know how to use them. New hardware features are accessible from within the familiar tool. Most engineers want to edit, compile, download, and debug their application using widely-supported “standard” tools without new, proprietary flows.

ARM DS-5 Altera Edition Toolkit

To meet the needs for software development and debug of SoC FPGAs described in this paper, Altera decided to team with industry leader ARM to develop a special edition of the industry-standard ARM DS-5 Toolkit to support the unique advantages and features of Altera SoC FPGAs. This industry-leading arrangement, the ARM DS-5 Altera Edition Toolkit, offers FPGA-adaptive debug and other key multicore features using a familiar, industry-standard interface as shown in Figure 16. The package also enables the use of a single Altera USB-Blaster™ II cable for both hardware and software debug.

Figure 16. The ARM DS-5 Altera Edition Toolkit Interface is Already Familiar to Many ARM Developers



```
Altera SoC FPGA connected
  Active Threads
    blinky #4 stopped on breakpoint #2 (PID 481 w
      arch_local_irq_save
      altera_gpio_set+0x8
      gpio_set_value_cansleep+0x6C
      gpio_value_store+0xC0
      dev_attr_store+0x18
      flush_write_buffer+0x40
      sysfs_write_file+0x108
      vfs_write+0xA4
      sys_write+0x40
Altera SoC FPGA connected
```

```
wait
continue
Execution stopped at breakpoint 2: S:0x7F000038
In thread 2 (OS thread id 479)
S:0x7F000038   85,0   {
wait
continue
Execution stopped at breakpoint 2: S:0x7F000038
In thread 3 (OS thread id 478)
S:0x7F000038   85,0   {
wait
continue
Execution stopped at breakpoint 2: S:0x7F000038
In thread 4 (OS thread id 481)
S:0x7F000038   85,0   {
```

Name	Start Address
altera_gpio_get	S:0x7F000000
altera_gpio_set	S:0x7F000038
altera_gpio_dir_in	S:0x7F000068
altera_gpio_dir_out	S:0x7F00009C
altera_gpio_save_regs	S:0x7F0000F0

Name	Value	Type	Count	Size	Location
Locals	1 variable				
flags	27	long unsigned int		32	$R3
File Statics (current)					
Globals	954 variables				

```
 * @gpio:   GPIO signal number.
 * @val:    Value to be written to specified signal.
 *
 * This function writes the specified value in to the
 * GPIO device.
 */
static void altera_gpio_set(struct gpio_chip *gc, uns
{
    unsigned long flags;
    struct of_mm_gpio_chip *mm_gc = to_of_mm_gpio_chi
    struct altera_gpio_instance *chip = to_altera_gpi

    spin_lock_irqsave(&chip->gpio_lock, flags);

    /* Write to shadow register and output */
    if (val)
        chip->gpio_state |= 1 << gpio;
    else
        chip->gpio_state &= ~(1 << gpio);
    __raw_writel(chip->gpio_state, mm_gc->regs + ALTE

    spin_unlock_irqrestore(&chip->gpio_lock, flags);
}

/*
```

```
Address          Opcode   Disassembly
S:0x7F000024              LDRNE    r3,[r0,#0x50]
S:0x7F000028              LDR      r3,[r3,#0]
S:0x7F00002C              LSR      r1,r3,r1
S:0x7F000030              AND      r0,r1,#1
S:0x7F000034              BX       lr
                          altera_gpio_set
S:0x7F000038              MRS      r3,APSR ; forme
S:0x7F00003C              CPSID    i
S:0x7F000040              MOV      r12,#1
S:0x7F000044              CMP      r2,#0
S:0x7F000048              LDR      r2,[r0,#0x54]
S:0x7F00004C              ORRNE    r1,r2,r12,LSL r
S:0x7F000050              BICEQ    r1,r2,r12,LSL r
S:0x7F000054              LDR      r2,[r0,#0x50]
S:0x7F000058              STR      r1,[r0,#0x54]
S:0x7F00005C              STR      r1,[r2,#0]
S:0x7F000060              MSR      CPSR_c,r3
S:0x7F000064              BX       lr
                          altera_gpio_dir_in
S:0x7F000068              PUSH     {r4}
```

Name	Value	Size	Access
sptimer0			
sptimer1			
stm			
sysmgr			
uart0			
uart1			
usb0			
usb1			
PIO_inst_0			
PIO_inst_0_DATA	0x0000000E	32	R/W
PIO_inst_0_DIRECTION	0x00000000	32	R/W
PIO_inst_0_IRQ_MASK	0x00000000	32	R/W
PIO_inst_0_EDGE_CAP	0x00000000	32	R/W
PIO_inst_0_SET_BIT	write only	32	WO
PIO_inst_0_CLEAR_BITS	write only	32	WO
UART_inst_0			
JTAG_UART_inst_0			
SYSID_inst_0			

For more details on the ARM DS-5 Altera Edition Toolkit, refer to www.altera.com/devices/processor/arm/cortex-a9/software/proc-arm-development-suite-5.html

Development Tools Summary

An excellent in-system debugging tool that offers simultaneous insight and control into both the ARM processor and FPGA logic portions of the SoC FPGAis critical for implementing the advanced features in these new devices while keeping project schedules on track. A side-by-side comparison of the Altera EDS and another SoC FPGA vendor’s tools is provided in Table 18 on page 31.

Conclusion

SoC FPGAs are a powerful new class of programmable devices that are applicable to a wide range of electronic designs. The commercially-available devices integrate a standard ARM processor—either a Cortex-M3 or a more powerful, dual-core Cortex-A9—with a rich set of peripherals, on-chip memory, a high-speed internal interconnect architecture, a hierarchy of on-chip memory, and a leading-edge FPGA fabric. While the available devices seem similar at first glance, the underlying architecture matters.

This white paper discussed a number of criteria to select the best SoC FPGA for your particular application, including system performance, design reliability and flexibility, system cost, power consumption, future product roadmaps, and the important role that development tools will play into the success of these SoC FPGAs.

For further assistance in selecting an SoC FPGA for a specific design, refer to the design consideration checklist for SoC FPGAs for a summary of information contained in this white paper.

Further Information

- Video series: A Look Inside: SoC FPGAs:
  www.altera.com/socarchitecture
- SoC Overview:
  www.altera.com/socfpga
- White paper: *Real-Time Challenges and Opportunities in SoCs*
  www.altera.com/literature/wp/wp-01190-real-time-socs.pdf
- White paper: *Error Correction Code in SoC FPGA-Based Memory Systems*
  www.altera.com/literature/wp/wp-01179-ecc-embedded.pdf
- White paper: *Achieving Lowest System Power with Low-Power 28-nm FPGAs*
  www.altera.com/literature/wp/wp-01181-lowest-system-power.pdf
- White paper: *FPGA-Adaptive Software Debug and Performance Analysis*
  www.altera.com/literature/wp/wp-01198-fpga-software-debug-soc.pdf
- ARM White Paper: *Better Trace for Better Software*
  www.arm.com/files/pdf/Better_Trace_for_Better_Software_-_CoreSight_STM_with_LTTng_-_19th_October_2010.pdf
- ARM Development Studio 5 (DS-5) Altera Edition Toolkit:
  ds.arm.com/altera

Acknowledgements

- Todd Koelling, Sr. Manager, SoC Products, Altera Corporation

Document Revision History

Table 19 shows the revision history for this document.

Table 19. Document Revision History

Date	Version	Changes
November 2013	1.0	Initial release.