

The Advantages of Hard Subsystems in Embedded Processor PLDs

Introduction

Programmable logic has achieved a level of integration sufficient to incorporate entire systems into a single device. This trend towards system-on-a-programmable chip (SOPC) design has been accelerated by the advent of products from programmable logic device (PLD) vendors that offer processor functionality. One class of these products provides this capability by including the processor embedded into the silicon die of the device. This approach offers several benefits in comparison to other approaches, foremost of which are higher levels of performance and lower device cost that results mostly from die size reduction.

In processor-enabled digital systems, the processor is always accompanied by a host of other functions, including memory, interfaces, and other peripherals. As a result, PLD vendors seeking to embed processors into their devices must also consider the advantages and disadvantages of embedding these other functions into the die as a “hard” subsystem. In addition to device cost and performance benefits, this approach results in an easier development process and shorter design cycles, thereby improving time-to-market. Using the Altera ARM®-based Excalibur™ embedded processor PLD as an example, this paper describes in detail the cost, performance, and functional advantages provided by including a complete processor subsystem in the die of products combining processors and programmable logic.

Elements of a Processor Subsystem

To explore the benefits of a hard IP processor subsystem in a PLD, one should first determine the elements of that subsystem. Digital systems are as varied as the products for which they are designed, but the majority that rely on a processor also include a common set of functions that complement or otherwise support the processor. A short list of these functions includes memory (e.g., RAM and ROM), memory controllers (i.e., SDRAM and flash), timers, and a standard communications interface like a universal asynchronous receiver/transmitter (UART). Dual-port SRAMs would also be a good addition because embedded memory blocks configured for this operation are already a valued feature in high-density PLDs. To manage multiple peripherals in the subsystem, an interrupt controller for isolation and/or prioritization should be included. Additionally, because the processor must run embedded software application code, robust hardware debugging capabilities and an embedded trace module are valuable additions for ease of system debugging and validation.

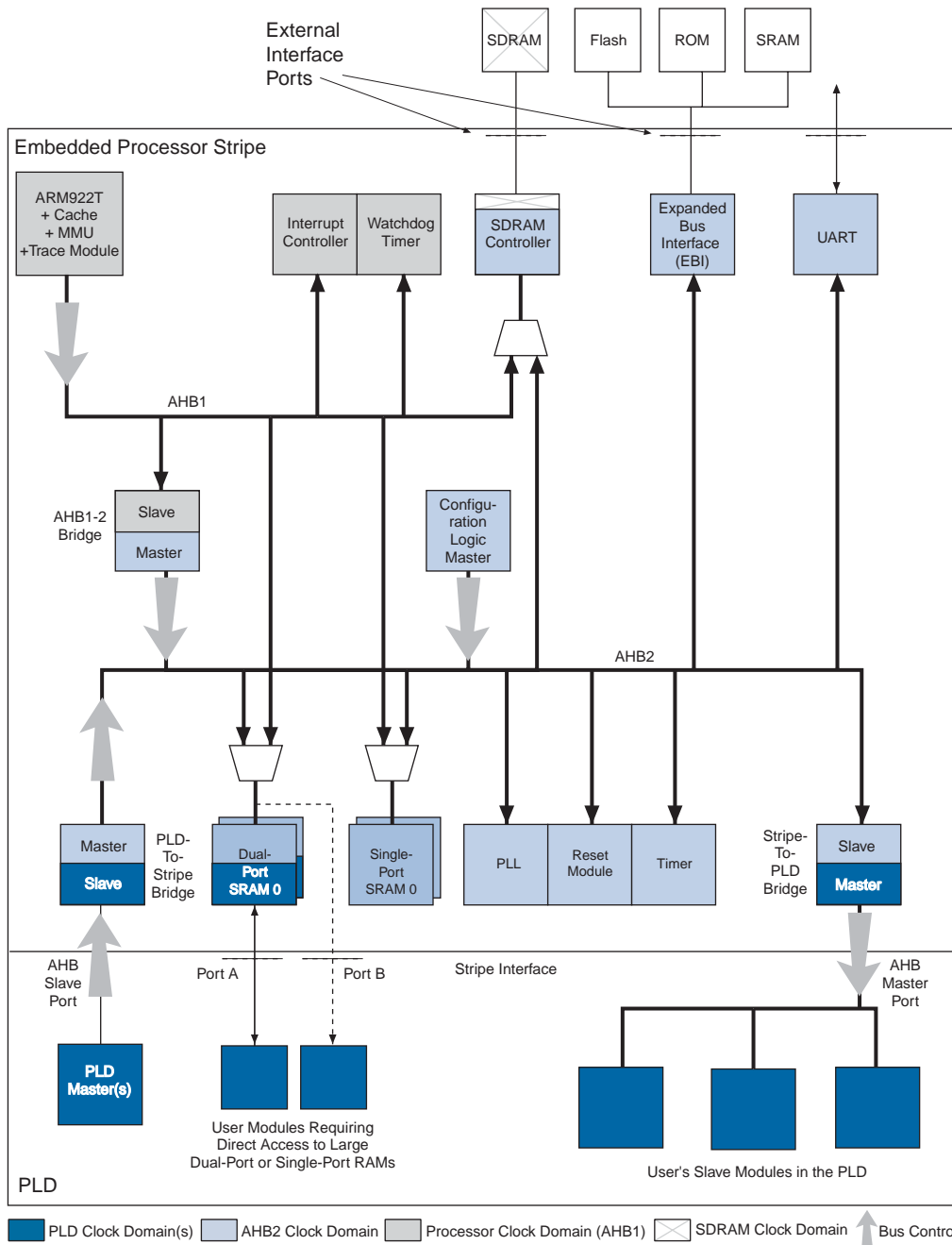
To provide efficient communication between these peripherals and the processor, the subsystem should include a bus structure of dedicated routing resources and data management protocols. Such a bus structure will also have to accommodate connections to functions inside the programmable logic portion of the device, as well as interfaces to external peripherals. Because this processor subsystem is part of a larger PLD, configuration circuitry should also be added to allow the processor to manage and control the configuration of the programmable logic portion of the device, greatly expanding the application opportunities for the device. For example, this capability allows the device to respond dynamically to changing system needs by reconfiguring itself.

Finally, because this product is directed towards the embedded systems developer, it should include the best possible real-time operating system (RTOS) support. In addition to a memory management unit (MMU), which is generally part of the processor core, most of the elements required to support generally available, high-reliability RTOSs have already been listed, including a dedicated means for providing input to and getting status from the processor (the UART), timers, and an interrupt controller. The inclusion of these resources allows the highest degree of flexibility for RTOS users, including the ability to run an RTOS without having to configure the programmable logic portion of the device.

Figure 1 shows a block diagram of the processor subsystem of an ARM-based Excalibur device (called the “stripe”), which includes many of the elements previously discussed. The ARM processor core in Excalibur devices operates at

speeds of up to 200 MHz, equivalent to 210 Dhrystone MIPS. It includes a single-port SRAM, which is useful for storing critical code and/or data tables, as well as a dual-port SRAM, which is accessible to the programmable logic portion of the device, making it a valuable resource for sharing data between the processor and other functions implemented in programmable logic. An interrupt controller, UART, and two timers are present, including a general-purpose timer and a watchdog timer to provide system checks. The integrated SDRAM controller supports single data rate (SDR) or double data rate (DDR) SDRAM at up to 133 MHz or 266 MHz, respectively, and the device supports up to 512 megabytes (Mbytes) of SDRAM. An embedded trace module in the processor core monitors the address and data and control signals, and reports compressed information off-chip to a trace port interface, allowing stripe operations to be observed in real-time and at full operational bus speeds.

Figure 1. ARM-Based System Architecture



For clock generation, two phase-locked loops (PLLs) are utilized: one for the processor and peripherals, and one for the SDRAM controller. Both PLLs are frequency-programmable via device configuration and the processor, allowing the clocks to be dynamically altered while maintaining device operation. In addition, the clock multiplication provided by the PLLs allows the processor to be clocked at the full 200 MHz using a lower-frequency reference clock signal. Finally, the PLLs can be disabled for low-power operation.

One AMBA™ high-performance bus (AHB) operating at full system speed (up to 200 MHz) connects the processor to the most bandwidth-hungry peripherals. Another AHB, operating at up to 100 MHz, provides communication between the main AHB, the other peripherals, and the PLD interfaces. In addition, the subsystem offers an expansion bus interface (compatible with industry-standard flash memory, SRAM, ROM, and other memory-mapped peripherals) that can support up to four external devices, each up to 32 Mbytes.

A PLD-to-stripe bridge gives masters in the programmable logic portion of the device access to slave elements in the processor subsystem within the stripe. This capability allows external bus interfaces to access shared data in an attached SDRAM device. Also, a stripe-to-PLD bridge gives bus masters in the stripe access to slave elements in the programmable logic portion of the device. Both bridges include synchronization logic, allowing the master and slave interfaces to reside in different clock domains from the AHB bus to which they are connected. Furthermore, both bridges support write-posting and read pre-fetching to increase throughput.

Cost Benefits of Hard-Processor Subsystems

One of the primary advantages of using a hard-processor subsystem is lower device and development cost. The device cost benefit derives mainly from the reduced die area that is required to implement the functions in the subsystem. In a PLD that does not incorporate a complete processor subsystem, the user must consume logic resources to build the peripherals and other processor-related functions required by the design. A simple analysis of the logic and memory resources required to replace the functionality of the subsystem indicates the cost impact of using “soft” intellectual property (IP) instead of a hard subsystem.

The combined logic element (LE) usage (a basic building block of PLDs) of the peripherals in the processor subsystem (based on their soft IP equivalents) is an estimated 6,000 LEs, not including the memory, the trace module, the configuration circuitry, the reset/mode control logic, or the clock-generation PLLs. Of these items, the memory consumes the most die area. To implement the 16-Kbyte dual-port SRAM and the 32-Kbyte SRAM offered in the smallest ARM-based Excalibur device, the EPXA1 device, would require over 190 additional embedded system blocks (ESBs). So if the EPXA1 device does not include a hard-processor subsystem, it would require nearly two and a half times more LEs and over eight times the number of ESBs to offer the same functionality in programmable logic.

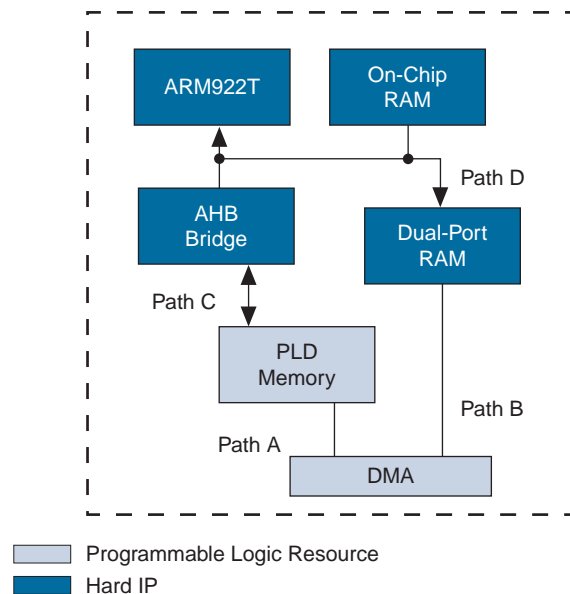
The same analysis of the largest ARM-based Excalibur device, the EPXA10 device, reveals a slightly different situation, though with similar results. The EPXA10 device would require only about a 15% increase in LE count to implement the subsystem functionality, but the 128-Kbyte dual-port SRAM and 256-Kbyte SRAM offered in the EPXA10 device would require over 1,500 ESBs in addition to the 160 already present in the device. In both cases, replacing the hard subsystem with the required programmable logic resources would more than double the size of the die, substantially increasing the cost of the device.

Besides the associated cost benefits of using a smaller die, hard-processor subsystems also reduce engineering design resource costs. For example, all elements of a hard subsystem can be rigorously tested and verified by the vendor, ensuring correct operation and timing. A vendor may also offer a larger package of development support in the form of hardware development boards and software routines that are tailored specifically to the hard subsystem. Finally, the framework of an existing hard subsystem, including robust support for RTOSs with a known set of peripherals, allows software development for the overall design to proceed concurrently with the development of the PLD design, reducing the overall design development time. All of these factors contribute to reducing the engineering burden on the design team as well as reducing time-to-market.

Performance Benefits of Hard-Processor Subsystems

One of the primary advantages of using dedicated silicon to implement any digital function is performance. PLDs that include embedded processor cores obtain the best performance by also embedding the processor subsystem. Evaluating the performance of the most common operations performed by such a device, which in most cases are memory accesses, demonstrates this proposition. Figure 2 shows a diagram of a test design used to compare the speed of the two memory access paths within the device. The first path, Path A, represents a memory access from a direct memory access (DMA) controller within the PLD to an ESB also within the PLD. The second path, Path B, represents a memory access from the same DMA to the dual-port SRAM in the processor subsystem.

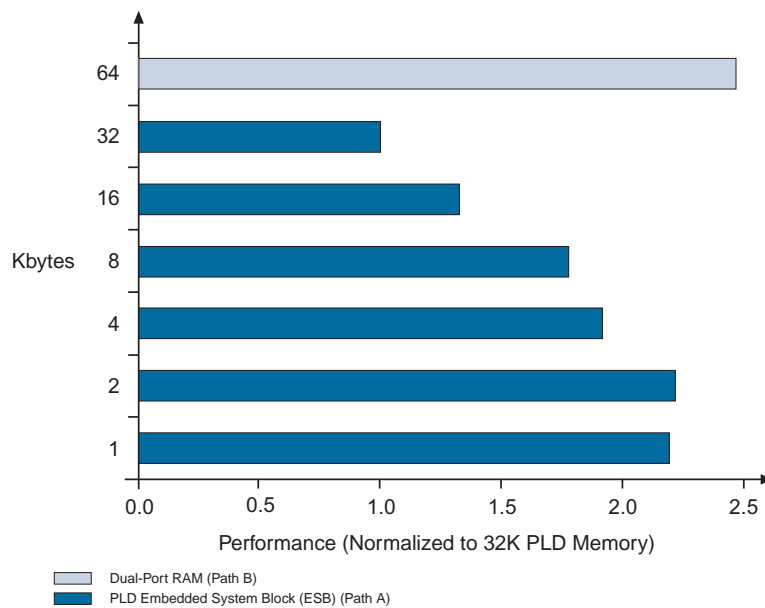
Figure 2. Test Design Used Compare Performance of Memory Accesses



The test design was implemented in an actual Excalibur device, and speed measurements were taken across a range of memory block sizes where the different memory blocks were composed of cascaded ESBs. Figure 3 shows a comparison of the measurement data, normalized to the speed of a DMA access to a 32 kilobyte (Kbyte) memory block. As expected, the speed of memory accesses to an ESB-based memory decreases as the size of the memory block increases. This increase applies to any PLD that creates larger memory blocks by combining smaller ones because of the greater levels of logic that must be combined to address larger memories.

The data shows that the speed of the memory access to the dual-port SRAM in the stripe is two and a half times faster than the speed of a memory access that takes place solely within the programmable logic portion of the device, even when the memory block size is half that of the dual-port RAM in the stripe. If memory modules such as the dual-port SRAM were not embedded into the die as part of the processor subsystem, then the only memory available for use by a designer would be the memory modules that reside in the programmable logic portion of the device. When cascaded together to create blocks approaching the size of the dual-port SRAM in the stripe, these memory modules cannot achieve a similar speed, indicating a significant performance benefit from including memory in the hard-processor subsystem embedded in the PLD die.

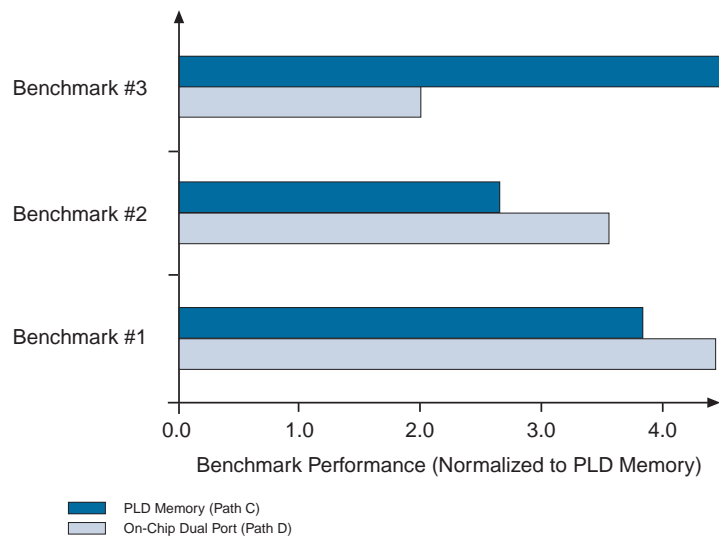
Figure 3. Memory Access Performance Comparison (DMA to Subsystem Dual-Port RAM vs. DMA to Programmable Logic ESB)



The block diagram in Figure 2 also shows the memory access paths from the processor to two different types of memory. Path C is from the processor to the embedded memory in the programmable logic portion of the device (composed of ESBs), and Path D is from the processor to memory that is part of the processor subsystem (the dual-port SRAM). As in the previous example, performance data using three different software benchmarks was extracted from this test design implemented in an ARM-based Excalibur device. The algorithms consisted of commonly used functions such as descriptor and data handling, and table lookups with poor locality of reference.

The data is shown in the graph in Figure 4. The result is that the software benchmarks ran two to four times faster using the dual-port SRAM in the processor subsystem, compared to using memory in the programmable logic portion of the device. The larger potential difference in performance in this case, as compared to the previous, is to be expected because Path B must use routing resources in the programmable logic portion to get to its destination. By comparison, Path D is wholly within the processor subsystem and thereby benefits from dedicated silicon resources.

Figure 4. Memory Access Performance Comparison (Processor to Subsystem Dual-Port RAM vs. Processor to Programmable Logic ESB)



The measurements in Figure 4 illustrate the case of a specific programmable device, but the same general principle can be applied across all such devices with equal validity. With respect to advances in PLD architecture, delays such as the memory accesses described here will certainly be dramatically decreased in future device families. However, with all other factors, such as semiconductor process, being equal, a hard subsystem of dedicated silicon will always enjoy a performance advantage over the exact same set of functions implemented in programmable logic.

Functional Benefits of Hard-Processor Subsystems

By including select elements in a hard-processor subsystem, PLD vendors can offer greater functionality in their embedded processor PLDs. For example, the inclusion of a basic set of peripherals (i.e., memory, controllers, and external interfaces) allows the processor to boot from external memory, execute code, and communicate with the external world without interaction with or intervention from the programmable logic portion of the device. As a result, the processor can operate and perform useful functions regardless of if and how the programmable logic portion of the device is configured. This capability ensures that an intelligent host (the processor) can be functional at all times during system operation, even when the programmable logic portion of the device is being configured.

As previously stated, the inclusion of configuration management circuitry allows the processor to manage the configuration of the programmable logic. This inclusion eliminates the need for a separate dedicated mechanism or device for configuration of the programmable logic portion of the device because the processor's non-volatile memory (usually flash memory) can be used for this purpose as well. When combined with RTOS support, this capability allows designers to easily develop a system that utilizes multiple configuration files for the programmable logic portion of the device. These configuration files can be stored and accessed locally in external memory, or the processor might retrieve them from an attached network. The ability to run an RTOS at all times allows the processor to easily support standard communications protocols for this and other purposes, such as running a TCP/IP stack to download configuration files and serving web pages that indicate the programmable logic's configured state.

Conclusion

By integrating a complete hard-processor subsystem into the die of an embedded processor PLD, PLD vendors can offer their users significant benefits in the areas of better performance, lower cost, and enhanced functionality. The elements of this subsystem should include basic peripherals such as timers and external and communications interfaces, as well as support functions like interrupt controllers, buses, and clock generators. High-demand items for system-level design, such as large memories, are also a wise addition. With the addition of a few more elements such as

configuration circuitry and an MMU, extra configuration options are made available that broaden the device's application.

The cost in silicon for a complete hard subsystem is low compared to the high degree of functionality it brings to the device, especially when contrasted with the cost of implementing the entire subsystem in programmable logic. In addition, the engineering costs of developing, simulating, and verifying a subsystem are eliminated, shortening time-to-market. All of these factors present a compelling rationale for PLD vendors to offer complete hard-processor subsystems in their embedded processor PLDs in order to meet the rigorous demands of today's systems designers.



101 Innovation Drive
San Jose, CA 95134
(408) 544-7000
<http://www.altera.com>

Copyright © 2002 Altera Corporation. All rights reserved. Altera, The Programmable Solutions Company, the stylized Altera logo, specific device designations, and all other words and logos that are identified as trademarks and/or service marks are, unless noted otherwise, the trademarks and service marks of Altera Corporation in the U.S. and other countries.* All other product or service names are the property of their respective holders, including the following: ARM and the ARM-Powered logo are registered trademarks of ARM Limited. Altera products are protected under numerous U.S. and foreign patents and pending applications, maskwork rights, and copyrights. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera Corporation. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

