WP123 (v1.1) August 18, 2000

Using FPGAs with ARM Processors

Author: Brant Soudan

Summary

This white paper discusses interfacing Xilinx FPGAs with off-the-shelf ARM processors. It covers some of the available ARM Application Specific Standard Products (ASSPs) and describes some of the Xilinx plus ARM development systems currently available for engineers to evaluate. Techniques and features that improve design performance are also included to help achieve maximum throughput.

Background

FPGAs are known for providing designers with several benefits in system design. One of the most important has been lessening the time to market. The quicker a company gets its products to market, the more market share it can capture from its competitors. This could mean millions of dollars in income to an established company and make or break a young company.

Another major benefit that FPGAs provide is flexibility. Designers can modify their design up to the day that the product is released to customers. And now with the concept of Internet Reconfigurable Logic (IRL), designs can be modified even after they are shipped to customers.

More recently, FPGAs have become attractive for other reasons. Because the FPGA cost per gate has come down significantly, there is no longer a 'price penalty' associated with the benefits of programmable logic. FPGAs have become an attractive option in many high volume applications. FPGA density having dramatically increased, Xilinx is now shipping multi-million gate devices. Certainly, this number will continue to climb in the future. This trend not only allows the design engineer to consider programmable logic for larger designs, but also allows them to absorb the functionality of other on-board chips. Fewer chips mean lower cost and easier board layout.

And finally, there is the benefit of performance. Application Specific Integrated Circuits (ASICs) built with cutting edge technology will always be faster, but FPGAs are close behind. Along with better on-chip performance, the I/O performance in Xilinx FPGAs has improved. One of the ways used to improve I/O performance and provide more flexibility was the introduction of Select I/O, which gives the designer the ability to choose an I/O standard suited to his system design. With the Xilinx FPGA's flexible Select I/O feature, programmable logic chips can now interface with almost any other component. Xilinx has already documented how to interface with an array of memories and peripheral busses. Xilinx has created the Memory Corner as a one-stop memory shop, providing solutions for leading edge memory technology. More information on the Memory Corner can be found at:
http://www.xilinx.com/products/xaw/memory/index.htm

Another component that is found in almost every system is the microprocessor, which comes in many flavors to suit system needs. The ARM microprocessor has gained popularity because of its features, peripherals, low power, and flexibility. Interfacing an ARM microprocessor to a Xilinx FPGA is described in the paragraphs following.

Designing a System

When designing a system, an engineer who chooses to use an ARM microprocessor, has to make one decision early on, i.e., whether the ARM microprocessor can be designed into an ASIC or placed on a board using an ASSP. If the designer chooses an ASIC, the microprocessor core will be designed into a custom chip along with the other necessary logic. In this case, there may or may not be an FPGA on the board. The components that interface with the microprocessor will be designed into the ASIC. However, ARM has developed many

variants of their microprocessor core for use across a wide range of applications. The basic ARM families are as follows: ARM 7, ARM 9, ARM 10, and StrongARM. Each family consists of several members with slightly different features. For example, the ARM 7 has five family members as follows: ARM7TDMI, ARM7TDMI-S, ARM710T, ARM720T, and ARM740T.

If the designer chooses to use an ASSP, the microprocessor is delivered in an off-the-shelf part. Most ASSPs are built around the ARM 7 core. Along with the microprocessor, several peripherals are generally included. The peripherals include memory (RAM, ROM, FLASH), LCD controllers, UARTs, DMA controllers, serial ports, counter/timers, and Phase Lock Loops (PLLs). Table 1 presents a list of ASSPs containing the ARM microprocessor. For more details, see the ARM website at:

http://www.arm.com/Pro+Peripherals/ASP/assp.htmlAlcatel

Table 1: **ASSPs Containing ARM Microprocessors**

Vendor	Part No.	Application	ARM CPU
ADI	AD20MSP430	SoftFone™ GSM Solution	7TDMI
AKM	AK2471	PDC Base Band Chipset	7TDMI
Alcatel	MTC-20276/7	ADSL Controller	7TDMI
Alcatel	MTK-20280	ISDN Controller	7TDMI
Alcatel	MTC-30585	SFSK PLC MODEM	7TDMI
Alcatel	MTK-20141	ADSL Modem	7TDMI
Atmel	AT76C503	Wireless LAN MAC	7TDMI
Atmel	AT76C510	Wireless LAN MAC	7TDMI
Atmel	AT76C551	Bluetooth Controller	7TDMI
Atmel	AT76C501	Wireless LAN MAC	7TDMI
Atmel	AT76C502	Wireless LAN MAC	7TDMI
Basis	BC6911	DSL Network Processor	7TDMI
Cirrus	SH8668	3Ci™ ATA Drive	7TDMI
Cirrus	SH8665	3Ci™ ATA Drive	7TDMI
Conexant	CN9414	Cable Modem	940T
Conexant	Modem	Modem Chipsets	7TDMI
Intel	D5313	PDC Processor	7TDMI
Intel	D5205	TDMA Baseband	7TDMI
Intel	Level One IXE100	Policy Accelerator	StrongARM
Lucent	T8302	Phone Chipset	940T
LSI Logic	L64324	Ethernet Switch	7TDMI
Mitel	MT92101	VoIP Chipset	7TDMI
MykoTronx	MKY-85	Encrypt/Decrypt Engine	7TDMI
MykoTronx	MKY-82	Encryption Engine	7TDMI
NetSilicon	Net+ARM	Network Processors	7TDMI
PrairieComm	PCI3610	TDMA Baseband	7TDMI

Table 1: **ASSPs Containing ARM Microprocessors**

Vendor	Part No.	Application	ARM CPU
Pijnenburg	PCC-ISES	Cryptographic Accelerator	7TDMI
Philips	VWS26001	Bluetooth Controller	7TDMI
Philips	VWS23112	DECT Processor	7TDMI
Philips	VWS22100	GSM Processor	7TDMI
Philips	VMS747	Security Processor	7TDMI
Philips	VCS94250	Serial Storage SSA	7TDMI
Qualcomm	MSM3000	CDMA Baseband	7TDMI
Qualcomm	MSM3100	CDMA Baseband	7TDMI
Samsung	KS17F80013	CATV	7TDMI
Samsung	KS32C65100	Ink Jet/Laser Fax	7TDMI
Samsung	KS32C6200	Ink Jet Printer	7TDMI
Samsung	KS32C6400	Ink Jet Printer	7TDMI
Samsung	KS32C6500	Ink Jet Printer	7TDMI
Samsung	KS32C6100	Laser Printer	7TDMI
Samsung	KS32P6632	PCMCIA/ATA Card	7TDMI
SiRF	SiRFstar II GSP2e	GPS Chipset	7TDMI
Sirius	DIRAC	Digital Spread Spectrum	7TDMI
Sound Vision	Clarity 2	Digital Camera	720
Virata	Atom Family	Network Processor	7TDMI

ARM Development Systems Available Today

Several products that use Xilinx FPGAs and ARM Cores have been developed by ARM. A complete list of products can be found at ARM's website:
http://www.arm.com/DevSupp/ordering.html

ARM's website contains both Integrator Platform Boards and Integrator Modules. Two Integrator Platform Boards available are the following:

- Standard Development Platform (Integrator/SP): ARM KPI-0100A
- ASIC Development Platform (Integrator/AP): ARM KPI-0101A

Figure 1 is a high-level block diagram of the Integrator/SP. For more information from ARM on Integrator/SP, visit ARM's website at: http://www.arm.com/products/Integrator/sp.html

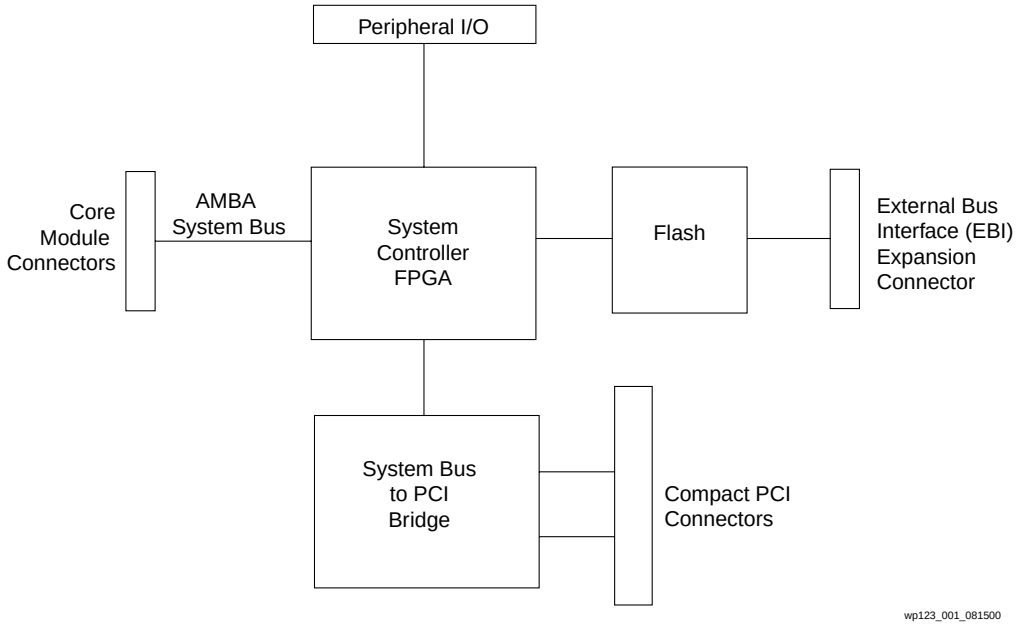

Figure 1: **Standard Development Platform (Integrator/SP) Block Diagram**

Figure 2 is a high-level block diagram of the Integrator/AP. For more information from ARM on Integrator/AP, visit ARM's website at: http://www.arm.com/products/Integrator/ap.html

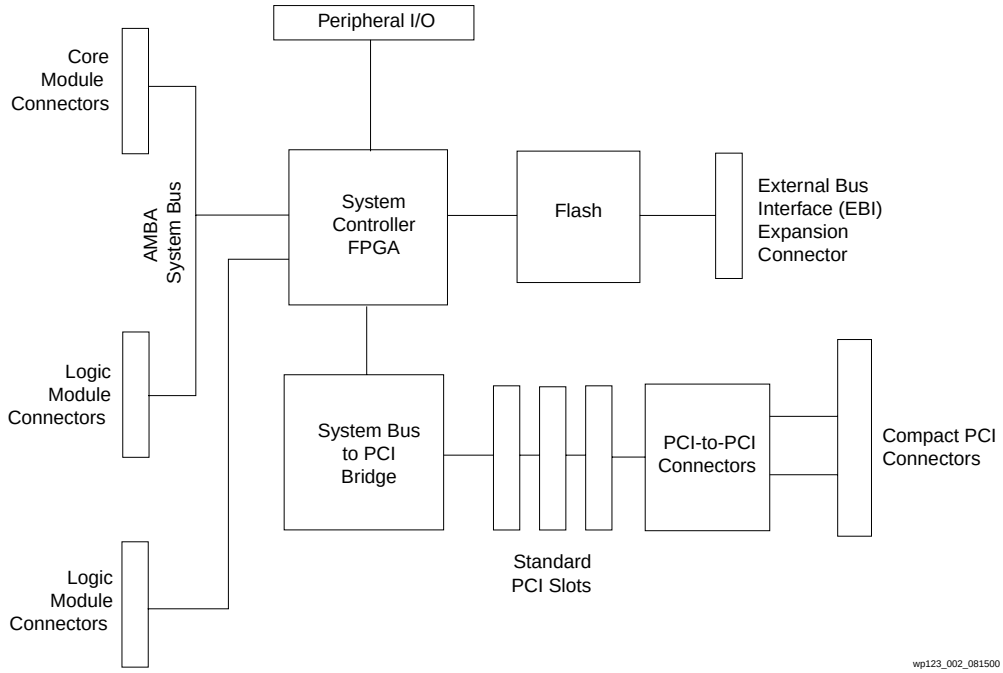

Figure 2: **ASIC Development Platform (Integrator/AP) Block Diagram**

The Integrator Modules developed by ARM are the following:

- Integrator/CM 7TDMI (ARM7TDMI Core Module)
- Integrator/CM720T (ARM720T Core Module)
- Integrator/CM 940T (ARM940T Core Module)

All modules contain the same architecture. The only difference is the ARM Core that is dropped in. Figure 3 is a block diagram of the Integrator Modules.

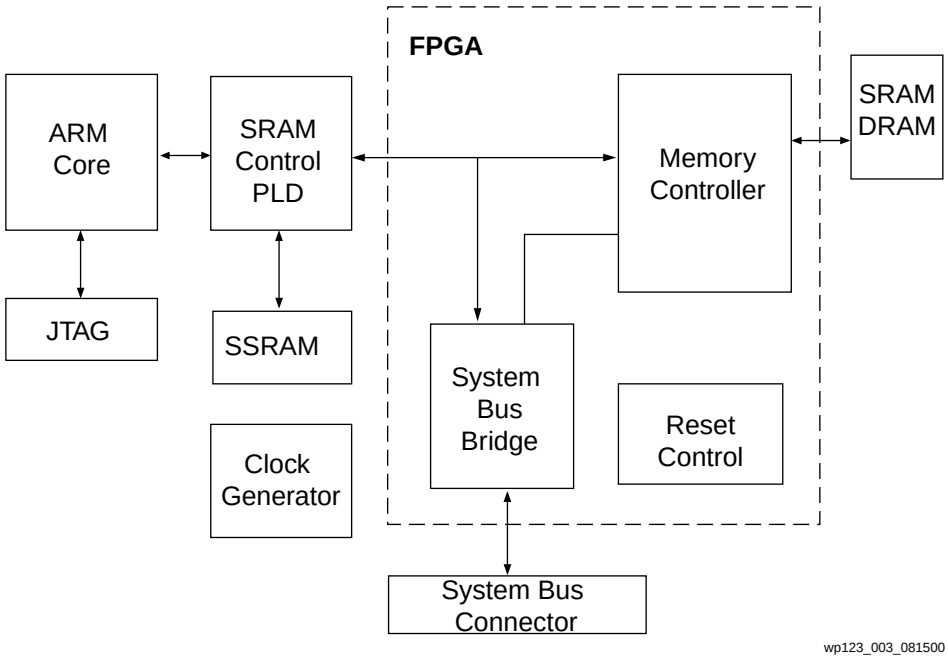

Figure 3: **Integrator Modules Block Diagram**

For more information from ARM on the above Core Modules, visit ARM's website at:

http://www.arm.com/products/Integrator/cm.html

StrongARM® PCI Development Kit from Avnet

The Intel® StrongARM™ PCI Development Kit gives embedded system designers a time-to-market boost by integrating all the key product technologies in a single environment. An Intel StrongARM SA1110 processor provides the processing power, while a Xilinx® Spartan II XC2S100 implements a glueless interface between the SA1110 processor bus and the industry-standard PCI bus.

For more information on the StrongARM PCI Development Kit, visit Avnet's website at:

http://www.ads.avnet.com/solutions/strongarm/

Enhancing System Performance

When developing a system with an FPGA, there are some design aspects to be considered, such as achieving maximum throughput and enhancing general performance. Given the same microprocessor architecture, there are two methods of increasing throughput, i.e., increasing the clock frequency and executing instructions in parallel.

To increase the clock frequency, the designer needs to consider several things. First, the microprocessor must be able to run at target frequency. This should not be a problem given that some ARM ASSPs can be clocked at over 200 MHz. Next, looking inside the FPGA, there are two different aspects of the design to be considered: the internal timing between synchronous elements and the I/O timing (i.e., getting data on and off chip).

Internal Timing

The internal timing is affected by both logic delay and routing delay.

Reducing Logic Delay

The logic delay is represented primarily by delay through Look Up Tables (LUTs). Logic delay may also come from using special resources in the Xilinx architecture, such as carry-logic, F5 and F6 muxes, and Tbufs. The software places these LUTs in the chip and connects them using the available routing. Designs that exceed the applied timing constraint tend to have a problem with either large routing delay and/or too many levels of logic.

Each LUT represents a level of logic. There are several approaches to reducing the number of logic levels. The design can be altered by moving logic around registers. Or, the designer can use different synthesis options and even another synthesis tool. And lastly, the design can be pipelined. The designer may wish to enter registers into the design. This approach significantly reduces the delay, but adds additional latency to the overall system.

Reducing Routing Delay

Routing delay can also be reduced using a number of techniques. Some of the easier, non-intrusive methods are as follows: increasing the placer effort in Place And Route (PAR), increasing the router effort in PAR, increasing the router iterations in PAR, and increasing the number of cost tables that PAR attempts. These techniques use the power of Xilinx software to increase performance without going back and modifying the design.

Another software technique available for designs that have few large routing delays is applying the prioritize constraint to the offending net(s) in the physical constraints file (.pcf). This constraint will tell the software to route the offending nets first, when all routing resources are available. Another method to aid the software in reducing routing delays is to help the placer place the components closer together. This can be done in several ways. One way is by using LOC and RLOC constraints. Another way is by using the floorplanner to graphically drag and drop parts of the design next to each other. These last few methods require intimate knowledge of the design and the Xilinx software.

I/O Timing

The other half of the timing battle is ensuring that data flows from one chip to another in the required time. The chip-to-chip delays happen in three main areas, i.e., the output delay from the chip sending the data, board delay, and input delay of the chip receiving the data. For the purposes of this white paper, only the FPGA's role as both the sender and receiver of data will be discussed here. Output timing refers to the amount of time needed to get signals from the last synchronous element in the FPGA to the pin of the FPGA. Likewise, input timing is the time needed from the pin of the FPGA to the first synchronous element of the FPGA. Board delay will not be discussed, other than ways of reducing the clock delay on the board using the Delay-Locked Loop (DLL), which will be discussed below.

Reducing I/O Timing

One way to reduce I/O timing is by registering both inputs and outputs and by placing these registers in the I/O blocks (IOBs) next to the FPGA's pins. Forcing the registers into the IOBs can be done in one of two ways. First, an 'IOB = TRUE' attribute can be given to all registers that need to be placed in the IOBs. Second, a command line option in MAP can be used to have the mapper pack them into the IOBs. This command line option is `-pr i`, `-pr o`, or `-pr b`. The first example will only pack input registers into the IOBs. The second will only pack output registers into the IOBs. And the last will pack both inputs and outputs into the IOBs.

Another way to speed the I/O timing is to adjust some of the IOB parameters. These parameters include the I/O standard, the slew rate, and the drive strength. For more information on these parameters, see the current Xilinx data sheets.

Another way of improving output timing is by using the DLL. The DLL reduces the delay on the clock tree, which results in extremely fast clock-to-out times. But also note that by reducing the delay of the clock signal, the setup time increases for inputs. For more detailed information, refer to *XAPP 132 Using the Virtex Delay-Locked Loop*:
http://support.xilinx.com/xapp/xapp132.pdf

Finally, to reduce board clock skew, use a second DLL in the FPGA to de-skew the board clock. When trying to de-skew a board clock, keep all the I/O and feedback trace lines of equivalent length to ensure proper clock de-skewing. An example is shown in Figure 4.

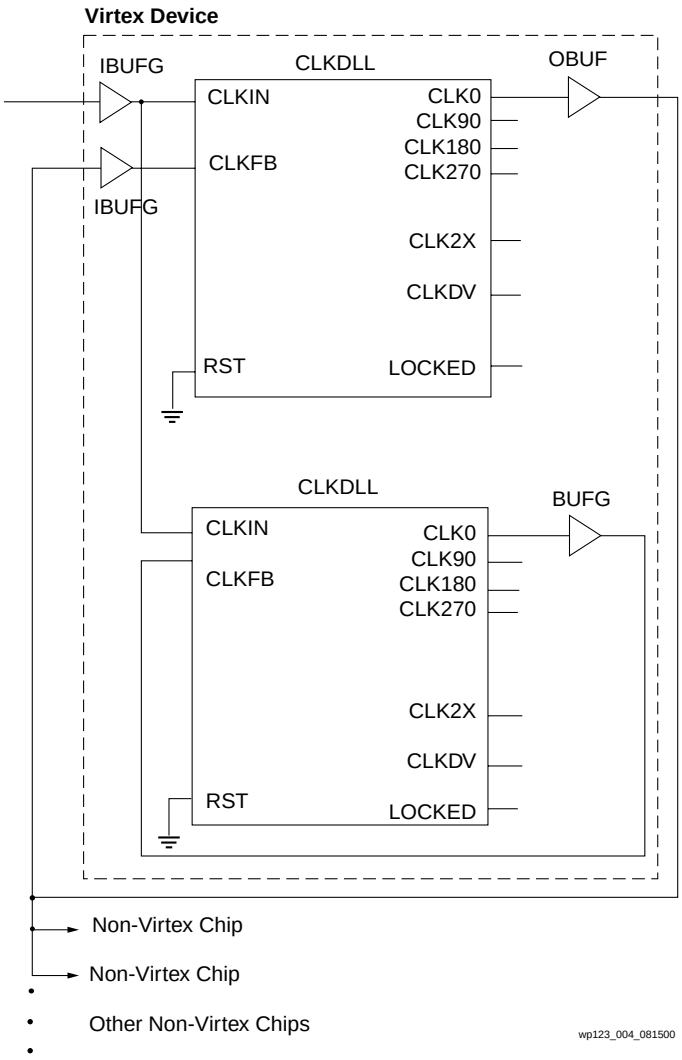

Figure 4: **Using a Second DLL to Reduce Board Clock Skew**

For more detailed information and example HDL code, refer to *XAPP 132 Using the Virtex Delay-Locked Loop*, http://support.xilinx.com/xapp/xapp132.pdf

References

TBD

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
?	1.0	Initial Xilinx release.