UNITECH '07 GABROVO

INTERNATIONAL SCIENTIFIC CONFERENCE
23 – 24 November 2007, GABROVO

IMPLEMENTATION OF SOFT-CORE PROCESSORS IN FPGAs

Petar Borisov Minev
Technical University of Gabrovo

Valentina Stoianova Kukenska
Technical University of Gabrovo

Abstract

Field programmable gate arrays (FPGAs) provide designers with the ability to quickly create hardware circuits. Increases in FPGA configurable logic capacity and decreasing FPGA costs have enabled designers to more readily incorporate FPGAs in their designs. FPGA vendors have begun providing configurable soft processor cores that can be synthesized onto their FPGA products. While FPGAs with soft processor cores provide designers with increased flexibility, such processors typically have degraded performance and energy consumption compared to hard-core processors. In this paper, we study the implementation of soft-core processors in FPGAs, and some of the decisions and design tradeoffs which must be made during the design process.

Keywords: soft-cores, soft-processors, FPGA, embedded systems.

1. INTRODUCTION

Microprocessors on field-programmable gate array (FPGA) chips are becoming an increasingly popular software implementation platform, due to their coexistence on-chip with custom logic. Such coexistence can reduce parts costs and board sizes, and can improve system performance due to reduced communication times between processor and FPGA. A hard-core processor is laid out on the chip next to the FPGA's configurable logic fabric. In contrast, a soft-core processor is synthesized onto the FPGA's fabric, just like any other circuit. Compared to hard-core microprocessors on some FPGA devices, soft-core processors have the advantages of utilizing standard mass-produced and hence lower-cost FPGA parts and of enabling a custom number of microprocessors per FPGA (subject to size constraints) – over 100 soft-core processors can fit on modern high-end FPGAs. However, soft-core processors have the disadvantages of reduced processor performance, higher power consumption, and larger size [1].

While any microprocessor soft-core could conceivably be mapped to an FPGA, FPGA vendors have in the past years introduced soft-core processors specifically targeted for FPGA implementation. Such FPGA soft-cores have instruction sets, arithmetic-logic units, register files, and other features specifically tailored to efficiently use FPGA resources, or perhaps more accurately, to avoid inefficient use of FPGA resources that may occur when synthesizing a general soft-core processor to an FPGA. The performance overhead of such soft-core processors on FPGAs compared to general soft-core processors on ASICs (application-specific integrated circuits) can thus be significantly less than the overheads when comparing FPGA versus ASIC implementations of general circuits [1].

A feature of FPGA soft-core processors is that of core configuration by the user (the application developer) through the setting of parameters. Configurable parameters may include instantiating a cache (and specifying its size), or instantiating a predefined datapath unit (like a multiplier or floating-point unit) and an accompanying instruction that uses the instantiated unit. Parameterized soft cores represent a different problem from that of developing custom datapath units and accompanying custom instructions, as done in application-specific instruction-set processors (ASIPs) like the ASIC-oriented ASIPs or FPGA-oriented ASIPs, due to the "on/off" (or limited number of) values of the parameters [1].

2. EXAMPLES OF SOFT PROCESSOR CORES

Today, we are witnesses of the emerging of many commercially soft-core processors, as well as supporting and development tools; some of principal products available are: Altera Nios/NiosII, LatticeMico32, and Xilinx MicroBlaze. They offer memory and logic elements with several Intellectual Property (IP) peripherals for the rapid development of System-on-Programmable-Chip (SoPC) [2].

2.1 MicroBlaze Soft Processor Core

A popular soft processor core example is Xilinx's MicroBlaze that can be customized with different peripheral and memory configurations. This soft processor core is a 32-bit Reduced Instruction Set Computer (RISC). This processor has a three-stage pipeline with variable length instruction latencies, typically ranging from one to three cycles. The tool used to accomplish the design is denominated Xilinx Platform Studio and with this friendly environment we are able to create a MicroBlaze based system instantiating and configuring cores from the provided libraries.

MicroBlaze was constructed around Harvard memory architecture. The 2 Local Memory Busses (LMB) are used to connect the instruction and data memories. The sizes of this memory as well as the number of peripheral used in a particular design are defined by the user. Additionally the On-Chip-Peripheral Bus is used to alleviate systems performance bottlenecks and is designed to support low-performance/speed peripherals such as UART, GPIO, USB, external bus controllers. A MicroBlaze system is presented in Fig. 6 as a good example of this technology.

The MicroBlaze can operate at up to 200 MHz within a Virtex-4 (4VLX40-12) component. The range of resources required to implement a MicroBlaze soft processor is between 900 and 2,600 Xilinx Look-Up Tables (LUTs), depending on how the processor is configured [2].

2.2 NIOS II Soft Processor Core

An another popular soft core processor example is Altera's NIOS II that has a load-store RISC architecture, in which many architectural parameters can be customized at design time. The user can decide between 16 or 32 bits of width in datapath, register file sizes; as well as cache size and custom instructions for the performing of user-defined operation in the speeding-up customized hardware. Those functionalities are supported by the builder development tools, and using the Nios II Integrated Development Environment (IDE) is possible to build, run, and debug software of several platforms. Altera also introduces a SOPC builder [38], for the rapidly creation and easily evaluation of embedded systems. The integration off-the-shelf intellectual property (IP) as well as reusable custom components is realized in a friendly way, diminishing the required time to set up a SoC and enabling to construct and designs in hours instead of weeks [2].

2.3 Mico32 Soft Processor Core

Both Xilinx and Altera created their own proprietary soft core processors, making the decision to accept a tougher adoption curve in exchange for saddling customers with an IP block that tended to lock their design into that particular FPGA vendor's devices.

Like Xilinx's MicroBlaze and Altera's Nios, Lattice's Mico32 is a soft-core RISC processor that can be easily dropped into an FPGA. Unlike the others, however, Mico32 is completely open [3]. Rather than take the lock-'em-in approach of their competitors, Lattice has gone the open source route, cleverly betting that enabling processor-based designs on their devices was much more important than locking customers into their architecture with an IP core.

LatticeMico32 uses fewer than 2,000 look-up tables (LUTs) on an FPGA, which makes it a very inexpensive engine for your embedded design. Because the processor is soft, you can configure it with just the options you want for your application. Optional features include things like data and instruction caches, user-defined instructions, and multipliers. This kind of application-specific customizability as well as the flexibility to add any number of processors to your design with only a small area penalty is the kind of flexibility that has made FPGA-based soft cores so popular among designers. Mico32 weighs in with 32 general-purpose registers, up to 32 external interrupts, and a dual Wishbone memory interface. Lattice estimates that the processors can run at over 100MHz on their low-cost 90nm ECP2 FPGAs.

In keeping with the open-source approach, Lattice chose the public domain Wishbone bus interface for Mico32 and has already announced a variety of available peripherals, including memory controllers, asynchronous SRAM, on-chip block memory, I/O ports, a 32-bit timer, a DMA controller, general-purpose I/O (GPIO), an I2C master controller, a serial peripheral interface (SPI), and a UART. These plug-on peripherals dramatically speed up system design, eliminating the need to custom-code many of the common hardware functions if you're building a Mico32-based embedded system [3].

3. DESIGN CONSIDERATIONS

3.1 Performance and Power

Two potentially critical system factors include the desired functionality and operational performance as well as the power required to implement the desired system functionality. There will typically be a delta between the power consumption and level of performance for fixed function processor implementations and potentially more flexible FPGA-based soft processor cores.

In order to compare the relative performance of soft processor cores a common processor benchmark approach must be used. Currently the most common benchmark is the DMIPS (Dhrystone Million Instructions Per Second) benchmark. The DMIPS benchmark is based on running an algorithm on a targeted processor core to measure its integer processing capabilities within a defined time period.

Additional performance considerations include the architecture of the soft processor core and its suitability for the targeted application. Factors to evaluate include: type and size of the memory and peripheral bus; size and model of address space; type and size of cache (instruction/data); type of controllers like DMA and interrupt structure; hardware accelerator capability (co-processor functionality); functional units such as the register file and execution units; type of pipeline and strategies to prevent stalls such as branch prediction.

Several factors influence power consumption including speed of operation, the number and type of resources required to implement the soft processor core and the characteristics of the FPGA component including static and dynamic power consumption vs. operational speed and temperature. One of the challenges associated with FPGA design is the difficulty of estimating power consumption. In an ideal development flow, schedule and resources will be allocated to design evaluation on a targeted development platform with an identical target FPGA component and soft processor implementation [4].

3.2 Design and Development Tools

The features and ease of use of the tool suite should be considered along with the tool design flow. Effective tool evaluation and analysis is important. The following factors can have a significant effect on design cycle efficiency: ease of use and feature set; design tool flow; development environment tool maturity; compatibility between major software releases; available training and quality of tool tutorials; debug and verification capabilities [4].

The tool suite (Figure 1) includes a collection of traditional software and FPGA design and development tools. The interaction between these two tool groups is commonly referred to as co-design or platform development tool. The software and soft processor core development tools are responsible for the parameterization of the soft core and associated peripherals and the implementation of processor buses, memory maps, interrupt structures and required processor peripherals. The software tools also include traditional compilation, linking, debug and download to the target processor.

FPGA design tools include the traditional development environments for capturing and synthesizing HDL code, simulation, place and route, debug and download of the design to the target FPGA platform.

3.3 Operating System Considerations

Another important design factor is the ability to utilize popular operating systems (OSs). Most embedded designs on 32-bit processors include an OS to reduce the design time of the software by providing an abstraction interface level to the software. Most operating systems include the OS and any lower-level software required to connect the OS to the hardware. This collection of software elements is commonly referred to as a board support package (BSP) (Figure 1). The BSP can include items such as the processor boot code and interrupt service routines for peripherals.

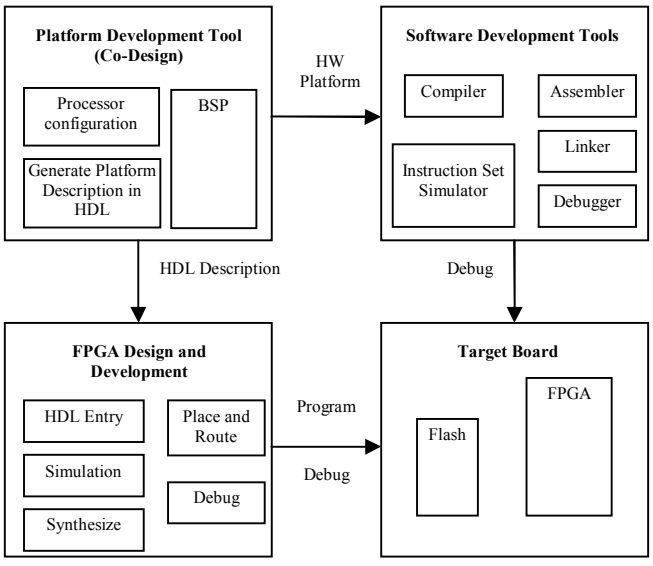

Fig. 1

Some important OS considerations include interrupt latency, kernel size, implementation of a robust set of services, and a collection of full-featured middleware. Some example middleware components include: USB stack; TCPI/IP stack; embedded web server; encryption algorithms; wireless Ethernet connection.

Other items which should be considered when selecting an OS include the API set, level of IDE integration, tasking models, kernel robustness, pre-emption, resource allocation, protection schemes and OS footprint.

Processor cores typically have a list of certified operating systems that have been pre-verified. If the design team does not have experience with the selected OS, it is advantageous for the team to be trained on the specifics of the OS to reduce development time and eliminate issues that could be encountered during development. Typical OS components include: task services; priority levels; timer management; memory management; application programmers interface (API); inter-task communication and synchronization [4].

3.4 Debug Options and Capability

The debug phase of a design will be iterative by nature and can consume a significant percentage of a design schedule without the correct tools and design access. The ability to efficiently debug a design can save weeks design effort and schedule. Robust debug features and capability are very important design efficiency factors. Some of the most effective tools for debugging a soft processor core design include: simulation (behavioral and timing); timing analysis; embedded logic analyzers and embedded bus analyzers; software simulators; non-intrusive real-time software debugger; trace capability; Hardware/Software logic analyzer triggers; board-level visual indicators, signal access ports and input control signals; standardized debug interface via JTAG bus [4].

3.5 Common Design Oversights

The following design factors are often overlooked by design teams new to implementing embedded FPGA soft processors. These factors should be given special consideration during each design cycle. Making a mistake in any of these areas may result in a significant impact to a project's cost or schedule.

Power consumption: Verify consumption on an evaluation board before final target board design and layout; Underestimating the FPGA resources required to implement the complete soft processor solution including all peripherals and bus structures: Implement design and analyze utilization report; Incomplete understanding of the impact of a soft processor's bus structure, memory interface overhead and peripheral interface speed on overall performance: Verify performance and functionality by implementing a design on a target evaluation board early in the design cycle; Not implementing or maintaining sufficient design margin for design migration and expansion: Select target FPGA components with room for growth with a common package to support potential future design enhancements [4].

4. CONCLUSION

This paper presents the implementation of soft-core processors in FPGAs, and some of the decisions and design tradeoffs which must be made during the design process. Making informed decisions during the design process reduces the time required to design, implement, debug and test an FPGA soft processor-based project. Important design factors are reviewed, common design oversights are discussed and soft-cores examples is presented.

Soft processor design teams will benefit from a system-oriented design approach, which considers the long-term effects of design decisions at each design phase. With a solid understanding of the overall design cycle, development tools options, and benefits of key design trade-studies, the design team can avoid many common design mistakes and oversights resulting in a more efficient and flexible design cycle.

5. REFERENCES

[1] Sheldon, D., R. Kumar, F. Vahid, R. Lysecky, D. Tullsen, Application-Specific Customization of Parameterized FPGA Soft-Core Processors, International Conference on Computer-Aided Design, ICCAD, San Jose, November 2006.

[2] Calderón, H., C. Elena, S. Vassiliadis, Soft Core Processors and Embedded Processing: a survey and analysis, Proceedings of -ProRISC, pp. 483-488, Veldhoven, The Netherlands, November 2005.

[3] Morris K., Soft Core War LatticeMico32 Opens the Field, FPGA and Structured ASIC, September 26, 2006.

[4] Cofer, R.C., B. Harding, FPGA Soft Processor Design Considerations, Programmable Logic DesignLine, October 12, 2005.