2009

An FPGA Multiprocessor System for Undergraduate Study

Christopher Korpela
United States Military Academy, christopher.korpela@westpoint.edu

Robert McTasney
United States Military Academy

AC 2009-2481: AN FPGA MULTIPROCESSOR SYSTEM FOR UNDERGRADUATE STUDY

Christopher Korpela, Department of Electrical Engineering and Computer Science

CHRISTOPHER M. KORPELA is an Assistant Professor in the Department of Electrical Engineering and Computer Science at the United States Military Academy at West Point. He received an M.S. in Electrical Engineering from the University of Colorado in 2006 and is a Senior Member of the Institute of Electrical and Electronic Engineers.

Robert McTasney, Department of Electrical Engineering and Computer Science

ROBERT J. MCTASNEY is an Assistant Professor in the Department of Electrical Engineering and Computer Science at the US Military Academy at West Point. He received the Ph.D. in Electrical Engineering from the University of Colorado in 2008 and is a member of the Institute of Electrical and Electronic Engineers.

An FPGA Multiprocessor System for Undergraduate Study

Abstract

We present our experiences using multiple soft processor cores on an FPGA to study advanced computer architecture at the undergraduate level. Our system instantiates multiple processor cores on a single FPGA device using the Altera Nios® II soft processor and associated CAD tools. With an easy to use development environment and powerful tools to quickly generate designs, an FPGA platform provides the necessary flexibility to quickly produce a working system. Students are able to easily modify and adapt their designs for a specific application. We demonstrate that multiprocessor systems can be developed, implemented and studied by undergraduate students due to the availability and accessibility of design tools and FPGA development boards. Further, these systems enhance the learning of multiprocessors and aptly compliment advanced computer architecture courses covering topics to include shared memory, synchronization, sequential consistency, and memory coherency.

1. Introduction

The last few years have seen a dramatic increase in the capabilities and performance of soft processor cores in Field Programmable Gate Arrays (FPGA). Once limited by relatively low clock speeds, a large number of gates, and cumbersome Computer Aided Design (CAD) tools, these soft processor cores are quickly becoming increasingly powerful for prototyping and high-speed designs. The industry shift toward multiple processors on a single die has brought the need to prototype these designs on reconfigurable systems. At the Intel Developer's Forum in 2005, the Intel President Paul Otellini stated: "We are dedicating all of our future product development to multicore designs. We believe this is a key inflection point for the industry." With this paradigm shift in processor technology, it becomes more critical that students are introduced to these types of systems at the undergraduate level.

FPGAs have become commonly available to undergraduate students in digital logic and computer architecture courses. Students are able to code in hardware description languages (Verilog and VHDL) to implement a simple AND gate or utilize industrial quality RISC-based processor cores for high-end designs. The design tools and hardware development boards accessible by undergraduate students have enabled them to produce designs that are comparable to those formerly restricted to the graduate level or in industry.

At the United States Military Academy at West Point, we study and utilize FPGAs in many of our electrical engineering courses. In the Department of Electrical Engineering and Computer Science, EE majors take a number of core courses to include EE360 (Digital Logic) and EE375 (Computer Architecture using VHDL). The program also offers a number of depth threads where the student can focus on a particular area of electrical engineering (i.e. computer architecture). Students learn digital logic in EE360 through the use of MSI (medium-scale integration) logic devices, CPLDs (complex programmable logic devices) and FPGAs. In EE375, students continue to learn computer architecture and VHDL by studying an in-house RISC-based processor core. The final course in the digital thread is EE484 (Advanced Computer

Architecture using VHDL). This senior-level elective covers topics to include branch prediction, static and dynamic scheduling, multiprocessors, memory and cache coherency, interconnects, and various I/O interfaces. Students also have the option to team up with a faculty member to focus on a particular area of interest in our EE489 course (Advanced Independent Study). In this particular case study, we wanted to investigate the feasibility of implementing a multiprocessor design at the undergraduate level. This paper presents our findings on building a multiprocessor system by a student with a faculty advisor. We evaluated two aspects of this multiprocessor design; first, the ease or difficulty of implementing this system on an FPGA by an undergraduate electrical engineering student and second, the correlation between this design and the topics covered in an undergraduate advanced computer architecture course. Due to our success, we have integrated this multiprocessor implementation into our senior-level advanced computer architecture class to give students the ability to study these systems in actual hardware. Further, this type of laboratory exercise can easily be ported to other schools with similar electrical engineering programs.

2. Related Work

Many universities have programmable logic devices and FPGAs integrated into their curriculum. We have also leveraged the educational resources of Altera's University Program and received engineering support from the Toronto Technology Center. Our primary reference is the Altera tutorial titled *Creating Multiprocessor Nios II Systems*.[1] We have also looked at FPGA work at other universities. At the Cornell University School of Electrical and Computer Engineering, students learn embedded design by using FPGAs to develop SOC (system-on-chip) devices. Their program has done extensive work with using FPGAs for embedded control and in electronic design.[2] At Georgia Tech, FPGAs are also integrated into the classroom where they are used to develop the skills that are necessary of an electrical engineer. FPGA-based SoPC development boards have been used over the past few years in their undergraduate classes. They have been successfully used in undergraduate student projects that allow for a wider variety of student projects as an alternative to more traditional off-the-shelf microcontrollers or other basic FPGA boards. For senior design projects, students have developed custom Nios soft processor cores with customized uClinux OS support to interface with various I/O peripherals. In the experience at Georgia Tech, coursework in digital logic design, computer architecture, and C programming is needed with VHDL or Verilog exposure being useful as well.[3] [4] Many of these schools do not introduce multicore designs until the graduate level. At West Point, we are integrating these systems in senior-level undergraduate courses.

3. Methodology

We defined success by the achievement of two objectives; the ability of the student to actually synthesize and implement the design in hardware and how this system reinforces concepts covered in our advanced computer architecture class. We piloted the hardware system with 3 students; one student conducted an independent study and a group of two students used the system for a final project. To successfully implement our multiprocessor system, we met a number of requirements, completed various steps in the design process, and overcame some difficult hurdles. First, the student must posses the requisite knowledge in digital logic, architecture, Assembly and C programming, and VHDL which is the standard hardware

description language in our department. This knowledge base is formed through our core curriculum. We were initially new to the hardware so we went through some tutorials and laboratory exercises using a simple and enhanced processor implemented in VHDL. Next, we transitioned to using the Nios II soft processor and SOPC (system on a programmable chip) Builder to generate relative simple designs. We utilized the DE2 Basic Computer reference design and built upon it to generate three soft processors. As noted earlier, the Altera University Program team and the Altera multiprocessor tutorial were particularly helpful in realizing this goal. After the design was successfully implemented, we focused on mapping topics covered in our advanced computer architecture course to the key design features of the FPGA design. We discovered that almost all of the key areas of multiprocessor design were reinforced by our hardware system. Finally, we then integrated this multiprocessor system as a laboratory exercise in our advanced computer architecture course.

4. Multiprocessor Hardware, Tools, and Implementation

Our design uses multiple soft processors instantiated on a single FPGA device. We break our implementation in to the various functional blocks by starting with the development board and its components. The Altera DE2 board (Figure 1), with its large number of I/O devices and peripherals, was specifically designed as an educational tool. The board includes switches, buttons, LEDs, 7 segment displays, and an LCD display to aid students with visualizing the functionality of the system programmed on to the FPGA. This board is equipped with a Cyclone II FPGA and comes with devices such as a built-in programmer, Ethernet, RS232, video, and USB among others. There are four types of memory: SDRAM, SRAM, Flash and a SD memory card slot.[5]

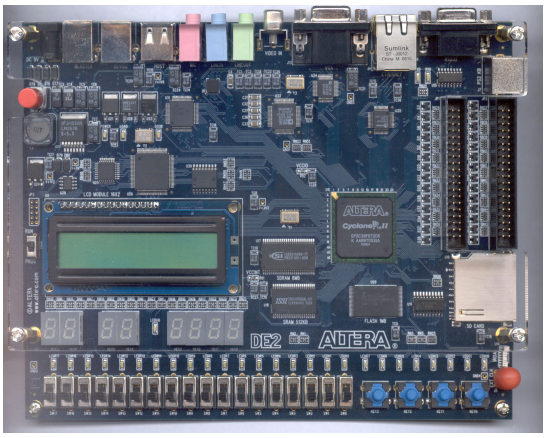

Figure 1: DE2 Development Board

4.1. Nios II Soft Processor

The Altera Nios II processor is a soft-core processor that provides a full 32-bit instruction set, data path, address space, 32 general-purpose registers and 32 interrupt sources. The soft processor has the ability to be configured by adding or removing features to meet the goals needed by a project and can be put on any board in the Altera FPGA Family. The Nios II processor has the following characteristics: access to a variety of on-chip peripherals and

interfaces to off-chip memories and peripherals; Hardware-assisted debug module enabling processor start, stop, step and trace under integrated development environment (IDE) control; and a software development environment based on the GNU C/C++ tool chain and Eclipse IDE.[6]

The advantage to using a soft processor is that multiple instantiations of the same CPU can be placed on to the FPGA. There are three variations of the Nios II: economy, standard and full with full being the largest in size and having the most functionality. The standard version is used in our configuration. It includes an instruction cache, branch prediction, hardware multiply and hardware divide. Depending on the type of application, the CPU can be upgraded or downgraded to meet performance and size specifications. Further, more CPUs can be added with their associated clocks and peripherals.

For the Nios II processor, there is a hardware mutex core available to protect shared resources. The mutex stands for mutual exclusion and is a synchronization mechanism that allows the processors to agree as to which of them should use the hardware resource exclusively. The mutex serves its function through a test-and-set operation. However, this mutex core does not physically protect resources in the system from being accessed at the same time by multiple processors. The software that is used must be designed to acquire the mutex before attempting to access the shared resource.[7]

4.2. System Implementation

In designing this system, the first step is to open up the Quartus II software and proceed to the SOPC Builder which will allow you to configure the processor that you want to use. You can choose to add multiple processors, as is done in this project, or you can simply add one. You also have the option and the ability to add other pieces of hardware that you need such as memory, clocks, and mutexes. The hardware mutex is one of the critical components in this multiprocessor system. Once the components are added to the design in SOPC Builder, the various components must be configured and linked together in order to operate and communicate in the correct manner. [8] After the system is generated, it must be compiled in Quartus so that it can be downloaded onto the FPGA.

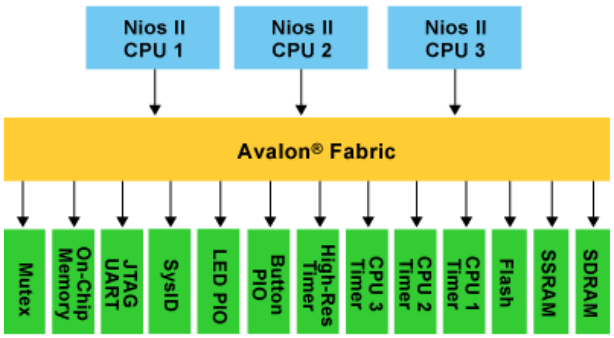

Figure 2: Multiprocessor System

Now that the FPGA is programmed with the processor, switch fabric and peripheral configuration that are required, it is necessary to develop the C code that will utilize the system resources and demonstrate the use of the hardware mutex. The C code in this case is a simple

program that checks to see which processor holds the mutex and responds according to whether the processor that is being examined has control of the mutex. In the Nios IDE, the three different processors must be made into their own projects in the workspace. In this setup, the shared memory resource is SRAM and only CPU1 is connected to the non-memory peripherals.

When the projects are built and compiled, the Nios IDE is then used to run the debugging session to experiment with the processors.[9] Running the session shows that the processors are indeed working as planned and the mutex is ensuring that only one of the processors is able to access the memory at a single time. The source code allows the processor to output a message when that processor has control of the mutex and is able to update its count. In Figure 3, it can be seen in the Nios IDE that the mutex was indeed stopping the other processors from being able to write to memory according to the code. This result is what we expected to happen with the implementation of the three processors and the mutex. The three processors attempt to write to memory, but only one of them is able to at a time. Even with more complex C code, the soft processors still function as expected without having the problem of attempting to access the same memory location at the same time.

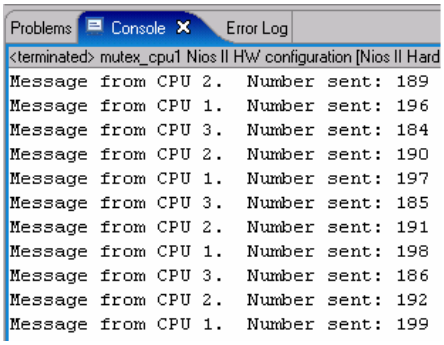

Figure 3: Mutex Working in Nios IDE

5. Mapping Multiprocessor Theory to Hardware Design

The second objective of this case study was to evaluate the correlation between the topics covered in our advanced computer architecture course and the multiprocessor system implemented on the FPGA. The Advanced Computer Architecture using VHDL course (EE484) builds upon the basic computer architecture course (some schools call it Computer Organization). As noted in the title, the course utilizes VHDL throughout the semester implementing designs on the DE2 FGPA development board. Topics such as branch prediction, superscalar machines, multiprocessors, and cache memory are some of the areas that are covered. The course textbook is the classic *Computer Architecture, A Quantitative Approach*, 4th Edition by Hennessy and Patterson. As the computer industry has shifted its focus on multiprocessors, so has the latest edition of this textbook. We took five main topics central to multiprocessor systems and compared them to the key aspects of the DE2 implementation. We have found that all five of these key concepts are part of our FPGA design. The five multiprocessor design features from the Hennessy and Patterson textbook we investigated were:

- *Shared Memory* – for multiprocessors with small processor counts, it is possible to share a single sized memory.
- *Synchronization* – the key ability to implement synchronization in a multiprocessor system is a set of hardware primitives with the ability to atomically read and modify a memory location.
- *Sequential Consistency* – requires that the result of any execution be the same as if the memory accesses executed by each processor were kept in order and the accesses among different processors were arbitrarily interleaved.
- *Mutual Exclusion* – locks and unlocks can be used straightforwardly to create mutual exclusion, as well implement more complex synchronization mechanisms.
- *Cache Coherence* – ensures that multiple processors see a consistent view of memory.

The correlation was assessed using student course feedback (3 students) and the course director / instructor assessment (2 instructors) of the five specific concepts in multiprocessor design. We also assessed the effectiveness of the new laboratory exercise to the course material. The combination of course feedback data and instructor input provided us with data to show the degree of correlation between theory and a practical exercise. The modified Likert scale found in Table 1 was used by both the students and instructors.

Correlation	1 Weak	2	3	4	5 Strong

Table 1: Correlation Scale

In most cases, the students agreed that the course was a positive experience for them and that the multiprocessor laboratory exercise complimented the course material. Instructor assessments were very similar, so the overall course results are presented. Particular topics that are addressed in the discussion are indicated in Figure 4. Our average of 4.6 across all topics indicates very strongly that the exercise reinforces concepts taught in the classroom.

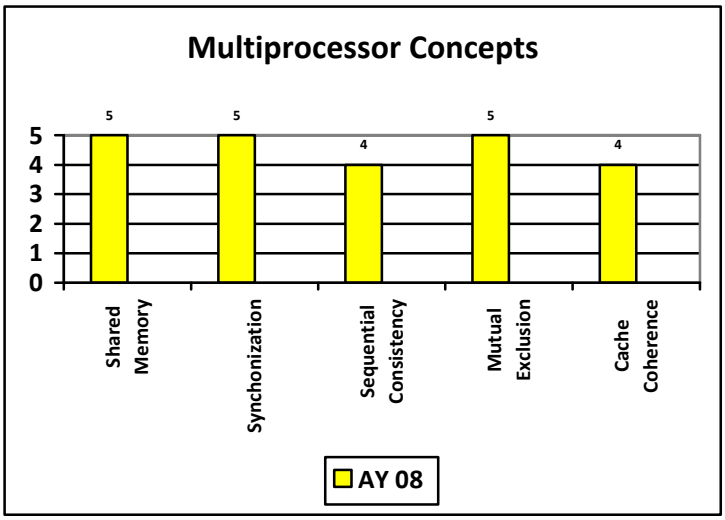

Figure 4: Multiprocessor Design Concepts

6. Conclusions

We demonstrate that an undergraduate electrical engineering student can implement multiple soft processor cores on an FPGA. Based on this success, we have integrated this design into a laboratory exercise in our Advanced Computer Architecture course. We further demonstrate that the practical implementation of this exercise fully compliments and supports the theoretical concepts covered in our course. As the computer industry has shifted towards multicore designs, it becomes imperative to introduce these systems early in an electrical engineering education. While the CAD tools have become easier to use, these types of projects require a significant amount of knowledge in VHDL design, C programming, FPGAs, and Integrated Development Environments. We recommend that only senior-level undergraduate students and above engage in multiprocessor design.

The views expressed are those of the authors and do not reflect the official policy or position of the U.S. Military Academy, the U.S. Department of the Army, the U.S. Department of Defense or the United States Government.

Bibliography

1. Creating Multiprocessor Nios II Systems, PDF File, Altera Corporation, http://www.altera.com/literature/tt/tt_nios2_multiprocessor_tutorial.pdf
2. ECE 5760 Advanced Microcontroller Design and System-on-chip, Professor Bruce Land, http://instruct1.cit.cornell.edu/courses/ece576
3. J.O. Hamblen, T.S. Hall, Using an FPGA Processor Core and Embedded Linux for Senior Design Projects, IEEE International Conference on Microelectronic Systems Education; pp33-34.
4. T. S. Hall and J. O. Hamblen, "System-on-a-Programmable-Chip Development Platforms in the Classroom," IEEE Transactions on Education, vol. 47, no. 4, pp. 502-507, Nov. 2004.
5. Altera's DE2 Development and Education Board, Altera Corporation http://university.altera.com/materials/boards/unv-de2-board.html
6. Nios II Processor Reference Handbook, PDF File, Altera Corporation, http://www.altera.com/literature/hb/nios2/n2cpu_nii5v1.pdf
7. D. Patterson, and J. Hennessy, *Computer Architecture A Quantitative Approach*, 4th Edition, San Mateo, CA: Morgan Kaufmann, 2007.
8. SOPC Builder Handbook, PDF File, Altera Corporation, http://www.altera.com/literature/hb/qts/qts_qii5v4.pdf
9. Nios II IDE documentation, website, Altera Corporation, http://www.altera.com/products/software/products/nios2/emb-nios2_ide.html