AC 2009-1542: COURSE DEVELOPMENT IN DIGITAL SYSTEMS TARGETING RECONFIGURABLE HARDWARE

Muhammad Hasan, Texas A&M University

Muhammad Zafrul Hasan received the B.Sc. in Electrical and Electronic Engineering from Bangladesh University of Engineering and Technology in 1988. He received the Master of Electronic Engineering from Eindhoven University of Technology (The Netherlands) in 1991 under a Philips postgraduate scholarship program. He subsequently held several faculty positions in an engineering college and in a university in Malaysia. He obtained the Ph.D. in Computer Engineering from New Jersey Institute of Technology. He was awarded the NJIT Hashimoto Fellowship in the academic year 2005-06. He is currently an Assistant Professor of Engineering Technology and Industrial Distribution at TAMU. His research interests include the design and implementation of dynamically reconfigurable computing systems, computer architecture and behavioral synthesis of digital systems.

Course Development in Digital Systems Targeting Reconfigurable Hardware

Abstract

It is important for engineering students to keep up-to-date with the changing technologies in order to fully exploit technology capabilities for implementing engineering designs. In doing so, students are well informed about the choices they have for developing a working prototype for their capstone design project. Field Programmable Gate Arrays (FPGAs) provide a flexible hardware platform to accommodate digital systems. FPGAs provide further opportunities for runtime reconfiguration that may be quite useful in applications requiring frequent changes in system behavior. In addition to having the necessary background in digital systems design, students need a tool that allows them to easily model their design such that the design could be implemented smoothly on FPGAs. Very High Speed Integrated Circuit (VHSIC) Hardware Description Language (VHDL) appropriately meets that need. VHDL even enables Intellectual Property (IP) cores to be incorporated seamlessly into a design that can be implemented on FPGAs. As such, a course needs to be developed encompassing microcontroller architecture, digital system design and implementation on FPGAs using VHDL. This paper describes the development of such a course with a case study of microprocessor design and implementation. The discussions also extend to incorporating an IP core processor in a system design that targets FPGAs.

Introduction

One of the major objectives of curriculum development is to enhance students' learning[1]. The factors that support this objective has been identified in[1] and in several other studies as: a) allowing students to be empowered, informed, and responsive learners who can assume a meaningful role in the society; b) providing engaging laboratory and continuous assessment of learning outcomes; c) utilizing students' pre-existing knowledge, providing examples and teaching subject matter in depth; and d) sharing ideas and materials so that projects can be built and connected to enhance the work of each other within a group. As such any course development activity needs to take these factors into account in order to ensure its success.

Students in Electronics and Telecommunications Engineering are introduced to Digital Logic as the first course that lays the foundation for many other courses. Such courses include Microprocessors / Microcontrollers, Assembly Language Programming, and Computer Architecture to name a few. With the knowledge built through this chain of courses, students are equipped with the necessary skills to embark on projects that involve Hardware / Software co-design. It is important to ensure a seamless transition from one course to the other in this chain. This enables the target students to integrate the body of knowledge gained.

Specialized hardware components could compensate for the speed of execution and power requirements for an application[2]. Moreover, reconfigurable hardware can ensure reusability of a given chip-area by many embedded applications, thus saving space. There is a clear demand for improved hardware platforms to enhance the performance of applications under cost-constraints. Current high-density FPGAs have the potential to satisfy this demand[3]. For a given number of I/O connections, FPGAs provide a low cost implementation platform for less than 1000 units.

Moreover, their reprogrammable features make it easy to test and debug the design. Partial reconfiguration of FPGAs provides further potential for execution-time savings by customizing and adapting the hardware with minimum changes[4]. As such, exploiting the capabilities of current FPGAs in order to implement the hardware for various types of students' projects becomes imperative.

However, traditional hardware design and implementation based on schematic capture is cumbersome. It needs a lot of effort geared towards building the circuit that may distract the designer from its intended purpose. Alternatively, the designer could express the functional requirements of the hardware in an intuitive way using higher level of abstractions. This ensures implementation details are taken care of by the automated tools rather than by the designer. This method of design specification and implementation enhances productivity. VHDL provides exactly this capability for a hardware designer. Moreover, it provides flexibility of integrating schematic-based older designs with designs of higher abstractions. The designs could be accurately simulated for functional and timing verifications. Subsequently, the designs could be targeted for implementation on various platforms such as Application Specific Integrated Circuits (ASICs) or FPGAs. As the latter is more prevalent in educational and research laboratories, we focus on such implementations. As a result, the VHDL has almost become a de-facto tool for realizing a hardware design into a working platform.

It is clear from the above discussion that the target engineering students need an exposure to hardware design and implementation process in order to enhance their learning experience. This exposure encompasses Digital Logic, Microprocessor, and Computer Architecture areas in a macro scale. FPGAs provide an excellent platform to realize their design with provision for future upgrade and tuning based on perceived requirements. VHDL is a design tool that can be efficiently utilized to map a hardware design into a realizable platform. In view of this, an objective is set forth to develop a Digital Systems course that targets reconfigurable hardware, such as FPGA, as a vehicle for design implementation utilizing the capabilities of high level VHDL abstraction tool. This paper discusses the development of such a course and the relevant experience gained.

Course Development

The set of learning outcomes for the course under discussion are as follows:

1) To learn VHDL to carry out logic design,
2) To understand basic processor components and their design ,
3) To use VHDL to implement a simple processor, and
4) To implement an IP processor with an external system in FPGA

Based on the above goals, the lecture contents are developed as described in the following paragraphs.

Register Transfer Language (RTL) and digital system design are introduced first. Micro-operations are the basis for sequential digital systems. Micro-operations specify much simpler actions than assembly language instructions. System requirements are specified by conditions and transfers of micro-operations. Pseudo-language nature of RTL is exposed to the students to

show that it can specify the logic of a system but does not translate directly into hardware without ambiguities[5]. It is expected that the students would be able to develop RTL statements corresponding to a given problem and implement it in hardware.

VHDL is introduced next. This has more formal structure than RTL and is widely accepted as a standard. The following topics are covered: modules, sequential statements and processes, compilation, simulation and synthesis, modeling registers and counters, behavioral and structural approaches. Students are expected to pick up the required skills to encode simple sequential and combinatorial designs in VHDL[6].

At this point, students are ready to make a transition to a systems perspective from component levels. So, stored program concept and microprocessor programming model are introduced. Among others, the following topics are discussed: instruction mnemonic, opcode and operand, instruction cycle.

Then a basic computer organization is discussed from a system designer point of view. System buses, processor organization, memory organization (internal and external), I/O organization, and interfacing concepts are introduced. They learn how to expand the memory capacity both horizontally (bit-width) and vertically (memory locations). Also, the techniques of memory mapping to a desired address are introduced.

After discussing all the above basics, the students are ready to be exposed to the activities of a system architect. In this context, processor design is embarked on. Students learn processor specification using Instruction Set Architecture (ISA) and state diagram, implementation of registers and data-paths, ALU, and control unit. Both hardwired and micro-sequencer based control units are covered. As a first introduction, a simple 8-bit processor with four instructions is designed. Its ISA is shown in Table 1 and the state diagram is shown in Figure 1[5]. Once they master the basics, a more complex sixteen instruction processor with conditional-jump capability is designed. A simulator is also introduced to aid in understanding how the designed processor works internally.

Table 1: Intended Instruction Set Architecture (ISA) and Representation

Instructions	Coded Representation
ADD	00AAAAAA
AND	01AAAAAA
INC	11XXXXXX
JMP	10AAAAAA

As the processor needs a memory to work with, memory design is the next step. Implementation of a processor-memory system using VHDL is discussed[7]. At this point, the full system is ready for synthesis, target FPGA downloading and testing.

It is important to be able to use and integrate existing designs into a new design to ensure reusability. It also ensures reliability of the whole design by incorporating previous designs that were proven to work. Picoblaze is a soft core processor available from Xilinx[8]. It is available as a

VHDL file that can be readily connected to a memory forming a fully functional system. It aids in rapid prototyping at an affordable cost. Moreover, tools provided by the vendor could be used to generate the required memory VHDL code with specified contents. To potentially benefit from this approach in future projects, students learn to port a Picoblaze IP core processor into an FPGA. This may also enable them to rapidly integrate systems with external peripheral devices.

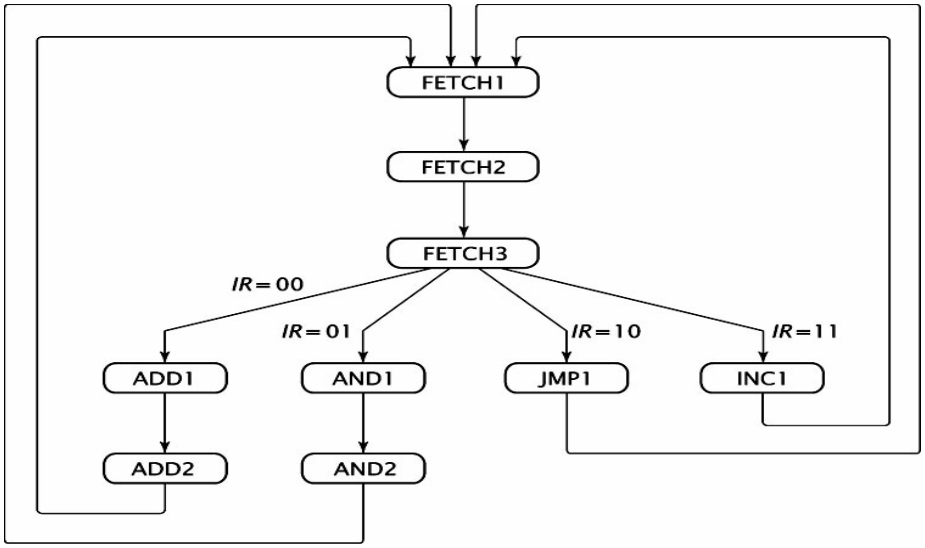

Figure 1: State Diagram of the Target Processor

Laboratory Development

The following experiments have been chosen to support the above mentioned objectives and the lecture contents.

Design and implementation of Modulo-6 counter as a state machine.

In this exercise the students basically learn how to translate a design specification into a realizable hardware. They follow a paper and pencil method to derive the expression of the system outputs. They also learn several alternatives to implement the design such as: register-decoder based, binary counter based, and FF-MUX based approaches.

Coding the design in VHDL – behavioral and structural approach, verification.
Here the students code their Modulo-6 counter design into VHDL. They are introduced to the process of synthesis, placing and routing, downloading the design bit-stream into an FPGA. They also learn the techniques of controllability and observability by tying the system inputs and outputs to switches and LEDs, respectively. This facilitates functional verification of the implemented design. In addition to the behavioral description of the design in VHDL, the students also learn to code the structural design involving the system equations.

Toll booth controller design, VHDL coding, verification.
Next, a more complex system design is embarked on. An automated toll booth controller is specified that accepts a defined amount (and a defined set of) coins for it to allow a car to pass through. Its state table and state diagram are developed. Subsequently, the expressions for the state variables and the outputs are derived. Then, the design is encoded in behavioral VHDL. The design is ported into the FPGA and verified following a similar approach as described for the above experiment.

8-bit processor design – specification using ISA and state diagram, implementation of registers and data-path, ALU, control unit, coding the design in VHDL, verification.
At this point, the students are prepared for a complete processor design that includes a few simple instructions. The starting point is the state diagram that is used to derive all the micro-operations of the processor. From this list, the involved data transfers are derived to design the data path for the processor involving all the registers and the bus interconnecting these registers. The ALU is also designed based on an adder, logical AND circuitry and a MUX to select the results. Each register and the ALU are coded in VHDL as a separate module. The control unit is designed using a counter, a decoder and a set of combinatorial circuit. Additional control signals necessary for the register section are also derived from another set of combinatorial circuit. Then all these modules are structurally interconnected using VHDL to form the complete 8-bit processor.
Implementation of a processor-memory system using VHDL – synthesis, target FPGA download and testing.
For the designed processor to function properly, a code memory needs to be designed. As such,
A 64 * 8 bit memory was designed in VHDL that can be accesses by the processor using its 6-bit address and 8-bit data buses. A binary code that involves repeated increment of the contents of the accumulator was embedded into the memory. Then this memory module was tied to the above processor to form the final system. This system was synthesized and implemented into the target FPGA board. Its functionality was tested by sampling the accumulator contents and observing them at the board output LEDs.

Porting Picoblaze IP core processor in FPGA, integration with external peripheral system.

The last experiment was to expose the students to the use of an IP core in designing systems. Xilinx provides one such core called Picoblaze[8]. It is an 8-bit soft microcontroller available in VHDL or as a gate netlist. The required assembly program (to be executed by this core) can be written in a text file. This text file acts as the input for a Xilinx tool (KCPSM3 assembler) that generates the VHDL code for the memory with the appropriate code to be executed by the Picoblaze. This VHDL module is joined together with the Picoblaze VHDL module at a higher abstraction level to form the functional system. This system is tested by sending out a specified bit pattern (from the core) to an output port that connects to the board output-LEDs.

Assessment of Student Performance

Students were evaluated in this course through tests, homework (HW), laboratory reports, laboratory performance, and participation. There were two tests – midterm and final each carrying a weight of 26 points (out of a total 100 points). This weight was assigned to highlight the importance of solving problems independently. Three HWs were given to the students for

submissions, each containing several problems. HWs were distributed evenly after every 5-weeks or so. They carried 26 points altogether. The basis of this weight assignment was to signify the benefits of being at par with the lectures and to encourage solving problems in consultation with others. The HWs were designed to prepare the students well for the tests. Laboratory reports were due following the completion of every experiment. Such report mainly included the working VHDL code written for that experiment along with the answers to questions testing the understanding of the experiment in general. These reports carried a weight of 10 points. Full laboratory attendance credited the students with 5 points. Performance of each student in the laboratory was assigned 7 points. This was based on their level of effort and the outcome of the experiments.

At this point, we can reflect quantitatively on the students' performance. First, the attendance in the laboratory was 100% implying a genuine interest in the practical aspects of the course. Following table summarizes the average class performance in different components of the evaluation process.

Table 2: Average Class Performance in Different Assessment Categories

Items	Midterm	HW #1	HW #2	HW #3	Final	Laboratory	Aggregate
Averages (%)	64.33	66.11	59.61	70.45	59.96	87.20	72.70

As seen from the above table, average points obtained in the HWs (roughly between 60% to 70%) are higher than that of tests (roughly between 60% to 64%). This implies that the students are more capable to solve problems with plenty of resources (such as books, notes/slides, and more importantly - time). Another observation of the above table highlights the inferior performance in the end term test (about 60%) compared to the midterm test. This might also be the consequence of scarce time-resource for preparation during the final weeks. However, the average points for the accumulated laboratory activities are quite higher (about 87%) than that of the other course work components (about 66%). This is the result of assured points for attendance and group submission of laboratory reports. Group submission allowed the best individual performance to be labeled for all members. The overall average (combination of all the components including laboratory) of the course is about 73% and seems satisfactory considering the level of the course, its delivery and the students' previous knowledge.

The next figure shows the distribution of the grades. Vertical axis represents the number of students obtaining a particular grade. As seen, total number of A's and B's almost equals that of number of C's representing a skew towards the tail end. However, it is consistent with the trend of the program.

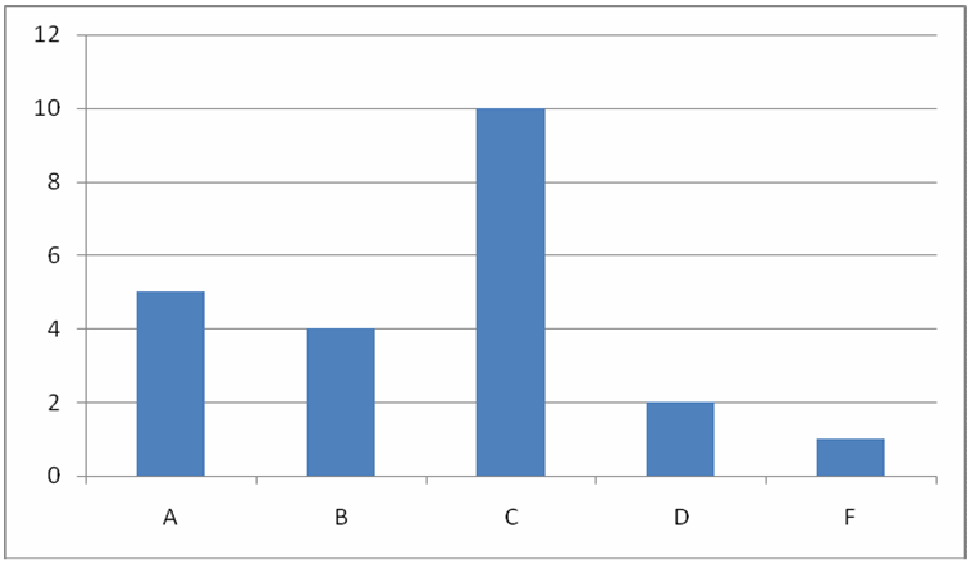

Figure 2: Distribution of Grades in 249 (The Course in Discussion)

In order to better appreciate the performance of these students in this course as compared to the prerequisite course of 219 (Digital Logic Design), the following figure has been included. It shows the grade distribution of the same students in the prerequisite course (One student transferred from another program and the record was not readily available). This indicates a peak around grade B. But, clearly the number of high achievers has increased from the 219 class (2 A's) to 249 class (5 A's).

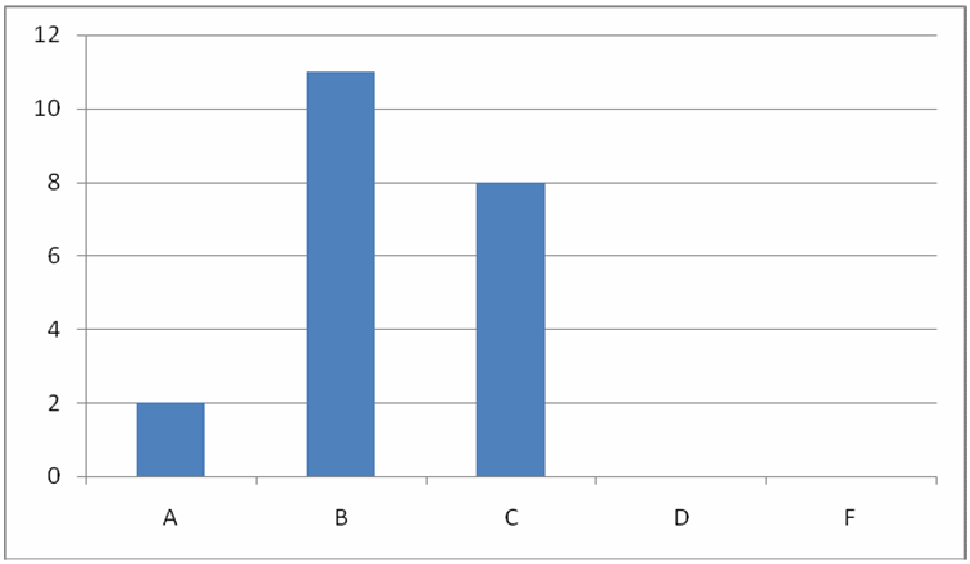

Figure 3: Distribution of Grades in 219 (The Prerequisite Course)

Conclusions and Further Considerations

The intent of developing such a course was to enhance student learning in the field of digital system design so that they are equipped with state-of-the art technology. With such capabilities, the students could readily implement prototype hardware system into reconfigurable logic devices. This would greatly facilitate realizing a concept of capstone projects into functional hardware. It is believed that the course has motivated and prepared the students in that direction. As such, it needs to be seen how this knowledge is transferred into and benefit the capstone design projects in successive years.

Although well accepted by the faculty and the students, the course needs further tuning and refinements. Techniques such as automatic documentation of the VHDL code could be introduced that facilitates readable code development and easy upgrades. In addition, Chipscope is a tool available within the Xilinx development environment that allows probing of signals that are not part of the inputs or the outputs[8]. This tool enables efficient debugging of the designed hardware. Also, the partial reconfiguration technique could be exploited in future laboratory experiments to dynamically adapt the FPGA hardware based on application requirements**.**

Bibliography

[1] Abul K. M. Azad, "Design and Development of an Introductory Digital Electronics Course within an Undergraduate Program", Journal of Engineering Technology, Spring 2008.

[2] M.Z. Hasan and S.G. Ziavras, "Runtime Partial Reconfiguration for Embedded Vector Processors," *Intern. Conf. Information Technology New Generations*, Las Vegas, Nevada, April 2-4, 2007.

[3] X. Wang and S. G. Ziavras, "A Framework for Dynamic Resource Management and Scheduling on Reconfigurable Mixed-Mode Multiprocessor", *IEEE International Conference on Field-Programmable Technology*, Singapore, Dec. 11-14, 2005.

[4] M.Z. Hasan and S.G. Ziavras, "Reconfiguration Framework for Multi-kernel Embedded Applications," 2nd Annual Reconfigurable and Adaptive Architecture Workshop (in conjunction with the 40th Annual *IEEE/ACM International Symposium on Microarchitecture*), Chicago, December 1, 2007.

[5] John D. Carpinelli, "Computer Systems Organization and Architecture", Addison Wesley Longman, Inc. 2001.

[6] Charles H. Roth, Jr. and Lizy Kurian John, "Digital Systems Design using VHDL", Thomson Learning, 2008.

[7] Douglas Perry, "VHDL: Programming by Example", McGraw Hill, 2002.

[8] www.xilinx.com, accessed in November, 2008.