The Future of Engineering Education
2024 Annual Conference & Exposition
Oregon Convention Center
Portland, OR . June 23 - 26, 2024
ASEE
Paper ID #43628

A Trilogy for Teaching and Learning Digital Electronics and Microprocessors

Prof. Wei-Jer (Peter) Han, Virginia Polytechnic Institute and State University

1. Background

According to the Moore's law, which is the observation that the number of transistors in an integrated circuit doubles about every two years. At present, one example of a GPU is the Nvidia H100, which has 80 billion transistors on a single chip. At the same time, on August 9, 2022, the President of the United States signed the CHIPS and Science Act. All of the above mean the education of digital and micro sequence should be modified to fit the trend because the growth of the numbers of digital engineers will not meet the growth of the Moore's law. In other words, as more semiconductor chips are produced, there should be more applications such as ASIC or custom-designed digital processors or controller utilizing these chips. It is possible to achieve this goal in undergraduate education. In the curriculum of a 4-year college, the courses in undergraduate digital or microprocessor sequence can be rearranged into the following format. First, introducing the combinational and sequential circuits. Second, introducing a specific microcontroller (μC) or microprocessor (μP). Third, which is where the innovative idea of this paper is, that is to model the specific μC or μP from the second phase with a HDL, then program it as it was in the second phase. At last, design a custom-made processor or any FSM with a HDL. The innovative idea of this paper is to develop students with skills that is to utilize a commercial μC or μP to complete application design in the second phase. Furthermore, utilizing a HDL to model the learned μC or μP so as to shorten the learning curve of designing a custom processor or digital peripheral circuits in the third phase. The benefit of including the use of a specific μC or μP in the second phase is to let the students immerse in the delicate design process such as programming in assembly language, so students can accomplish project development in the middle of the program. After the second phase, students can proceed with embedded system design or retreat with μC or μP design.

2. Introduction and prior development review

The goal of this paper is to provide a mean of educating undergraduate students to be able to design custom-made processors or peripherals whereas at the same time to possess the demanded skills of the embedded job market. When students learn how to use and design with a commercial μC or μP, it opens another door for students. Herein, using a commercial μC or μP is to depart from using an academic μC or μP model. Many textbooks and journal articles discuss how to improve teaching and learning effectiveness in this subject. In article [1], it describes the use of Microchip's PIC and TI's Cortex®-M4F based MCU as well as Digital Systems Design with FPGA using Verilog but it doesn't imply any method or idea to use a FPGA to develop a μC or μP. The curriculum in many colleges probably teaches with the same content and method but the result is not the purpose of this paper tries to present. On the other hand, some improvement has been made to enhance the learning of μC or μP such as what the article [2] shows with an experiment platform, but it again doesn't discuss any information to let students design their own processors or controllers. As described above, knowing how to design with a commercially available μC or μP opens a door to explore embedded system development which redirects the focus of the development of a digital system into software and hardware co-development. Ultimately, the time and energy spent on the embedded systems from faculty and students weighs more than designing a μC, μP, peripherals or a dedicated processor. An example of these can be found from articles [3] and [4]. Nevertheless, more and more improvement in μC or μP development have been published such as the personal learning environment (PLE) in microprocessors by DIY method [5] and a hardware and software framework for a simple academic processor (SAP) [6]. In article [6], it shows the use of Xilinx FPGA to develop SAP for education. Although the method is exceptional for educational

purpose, there is learning curve to overcome if students don't know how to *program* the specific SAP. The notion of *programming* or a programmer's model is actually an important determining-factor to a new hardware platform that can be adopted to learn the development of a new hardware such like a μC or μP in a relatively short time. In article [7], it discusses the use of soft-core-processors of the MicroBraze. Similar soft-core-processors are available to Altera (Intel) such as its Nios II processors, too. However, those soft-core-processors are limited to the vendor specific platforms. In the article [8], it illustrates the use of Altera's DE2 to design basic computers. In a more advanced setting, like article [9], it uses Xilinx's MicroBlaze again for biprocessor's SoC development. To make the material feasible to undergraduate students, these articles may not be easy to improve students' understanding, nor to use it for practical application in short time such as one semester. To improve this situation, the author reveals an idea in this paper to interconnect the use of a commercial μC or μP and develop the similar μC or μP model in a SoC platform. Since the development tool and the programmer's model to the hardware are all familiar to students, implementing a such μC or μP in FPGA can accelerate the learning process to the development of an application processor or controller. A Microchip PIC16 μC is used due to its simplicity to study its feasibility to achieve the goal of developing custom-made processor or controller in FPGA. The benefit of using a commercially available μC is also obvious in that not only students have learned it before but the development tool is also easy to acquire and lots of application notes are available to use after the soft processor or controller is implemented in FPGA.

3. Method of implementation

There will be three phases of learning with this proposal. The first part of the teaching and learning objective in this sequence is to cover the fundamentals of digital electronics, namely, the logic gates and flip flops as well as their applications. It's not necessary to introduce advanced HDL at this level because a HDL will not help students in understanding the application of digital systems, rather in the implementation of a digital design. At this level, students are still needed to wire protoboards, to know the traditional MSI circuits, to know arithmetic circuit, to trigger flip flops, to identify or design the difference between asynchronous and synchronous counters, to design state machine or to use algorithmic state machine design method, to program a CPLD, to know the technology used in A/D and D/A, and to know multivibrators and their applications. The above-mentioned topics will be sufficient to introduce the fundamentals of digital electronics and establish the reason behind the digital revolution but not to discuss computer engineering yet. There are many books addressing these topics, so these are not the subject of this paper.

After the first phase of learning, students can build digital systems with discrete logic ICs or PLD devices. The second phase of teaching or learning will be shifted into the application of a small microcontroller (8 bits) due to its simplicity. Students will learn the programmer's model of the CPU and program in assembly language. If the students have learned C/C++ before, correlating C/C++ and assembly language can envision students' understanding such as the use of pointers. The reason for using assembly language is to get students familiar with the use of status flags, and the use of opcode and operand in the instruction set, as well as to understand the addressing modes. Any useful algorithm in the assembly program should also be introduced. A simulator, emulator (ICE) or in-circuit programming or debugging tool to execute a written program and run in the target MCU is necessary. The use of stack, I/O programming, and the use of a debugger within an IDE (Integrated Development Environment) should

be demonstrated and used. If time permits, subroutine call and interrupt handling should be covered too, and a keypad for inputs and LED segment for outputs can be added as well. After this phase of learning, students should comprehend the use of assembly language to program a microcontroller. Using peripherals and interfacing I/O techniques can be included in the second phase. Compared to some traditional method of teaching in this sequence of subjects, we promote the use of a commercially available μC or μP at this stage instead of using an academic model of a μC or μP. That textbooks such as [14] adopt the MIPS system from Patterson and Hennessy's [15] model or using EC-1 and EC-2 models from [16] with simplified design for academic purpose are not our choice. We can however utilize a simplified model of μC or μP to get started. The simple computer model in [10] can serve this interim purpose because it has some similarity to the commercial μC or μP we will use.

Then, we enter the third phase. In this phase, a HDL (Verilog or VHDL) will be used to model logic elements, starting with logic gates, flip flops, register cells, a datapath, and an architecture for a stored-program computer. In this phase, we want to design a simple computer with basic architecture including an ALU, register files, instruction decoder and control unit. One way to do this is to model the microcontroller used in phase two and implement in FPGA since the students are familiar with its development from phase two. This is also the innovative idea which this paper presents. With the familiarity of the learned μC or μP from phase two, students can grasp the idea of implementation of the target μC or μP quicker. In addition, the software tool is available. Alternately, if implementing a commercially available μC or μP in FPGA is difficult for students in the first time, there can be another arrangement. That is to start with an architecture of a simpler processor with just a few registers in it and simpler ALU. The disadvantage of this method is that the instruction set will be limited, primitive and different from what was learned in phase two. Therefore, it can only be served as an interim purpose. The advantage however is that it possesses most functions of a processor so it can get students started much quicker. After students become acquainted with the idea of a processor design, we can advance the design into a commercially available computer that is learned from phase two and continued with the benefit of available tools and the familiarity to students.

4. Summary of Implementation

The first phase of the trilogy should be very similar to a traditional digital electronics course. Many textbooks are available to carry out the plan stated in the Background section. For combinational circuit, the traditional functions of four MSI circuits should be addressed, namely decoders, encoders, multiplexers and demultiplexers. In addition, the arithmetic circuit and the shifter such as the barrel shifter should be included. There are fewer issues with teaching and learning on the first phase, so students can proceed to the second phase after this stage which is to learn how to use a commercially available μC or μP.

There are two purposes to teach microcontroller in the second phase. One is that students can apply microcontroller-based systems to their workplace immediately after graduation. There are lots of application solvable by a simple μC or μP. Students may need to learn more interfacing techniques such as communication (UART, SPI, I2C, CAN) or A/D, D/A, but these are application specific and can be learned on-the-job. The benefit is that there are lots of employment opportunities requiring this skill set. Learning a commercially available μC or μP becomes an essential task. It is because it's easier to design a tool if the designer knows how to use the tool, so we want to design the same μC or μP to our FPGA in

the phase three. In our phase two, students will get ready for this purpose. Our goal is to develop the skills so students can design custom-made processors or controller in addition to just using the commercially available μC or μP. Having the μC or μP used in phase two to be designed through a FPGA is the best method to start μC or μP design. For example, we choose Microchip's PIC 8-bit microcontrollers as our beginning platform to learn its architecture, instruction set, assembly language programming and algorithms in phase two, and will implement it in phase three. Using PIC 8-bit μC is to balance the complexity and the completeness of a typical μC or μP system. Many books have been written to achieve this educational goal in phase two, so it shouldn't be a problem to find a good book as a text for students to learn how to design a μC or μP system after learning the fundamentals. Students should also focus on assembly language programming during our phase two process because the purpose of learning a commercial μC or μP is to understand how it operates so we can design it later, instead of using it with higher abstraction for embedded application.

The method to enter the third phase is the main theme of this paper and is also the innovative idea this paper presents. It's possible to be difficult for students to design the μC or μP used in phase two into a FPGA in phase three. That's why we see textbook such as [13] uses simplified models but those models still require assembler or simulator to verify its software operation. If difficulty in implementing a commercial μC or μP occurs, we can start with a simpler computer such as the example used in [10]. Nevertheless, the software tool such as assembler is still needed for software verification. In the example of a simple computer [10], the instruction set is limited but works well. The generation of the control word is intuitive with just a few logic gates. The datapath is simple with an ALU and there are only four register files in this design. The data can be obtained from an immediate addressing or direct addressing mode based on the opcode and operand of an instruction. The program counter is the only part that requires more attention. When designing the program counter for this simple computer [10], the best way is to realize it with logic gates and with MSI functions such as adder, decoder, encoder, multiplexer and demultiplexer. It's because designing with this process will help students to understand how to handle the program counter when dealing with branch or jump types of instructions. Besides, this also helps students design function or subroutine calling instructions when advancing to implement the learned μC or μP from the phase two. If this simple processor model as an interim is used in the phase three, students should also program it with code written in assembly language. Writing assembly code should not be difficult but may be slightly different because this interim processor is simpler in architecture smaller in complexity than the μC or μP students have used in the phase two. From reference [10], the online assembler can be found in this link [11]. Its instruction set is limited to the following; MOVA, MOVB, INC, ADD, SUB, DEC, AND, OR, XOR, NOT, SHL, SHR, LDI, ADI, LD, ST, BRZ, BRN and JMP. Although there is no stack to support subroutine call, this model can serve as an interim processor and can help students understand the basics of datapath, the control unit which generates control word, instruction decoder, and the program counter which generates the address to fetch the next instruction. Building this interim model can actually be arranged as a midterm project in the phase three. After the implementation of the interim processor is complete, a more complicated computer model, yet is familiar to students, can be introduced. That is the μC or μP which were used in phase two. In our example, we choose Microchip's PIC16 microcontroller to begin with due to its simplicity. In phase two, students have been taught with a commercially available μC or μP. We chose the Microchip's PIC microcontroller, specifically the PIC16F628A family to start. Students should have used the same microcontrollers in phase two so students should be very familiar with this processor. When writing the assembly code, students focus on the software which will be exerted on the hardware. When designing

the processor, students focus on how to design the hardware which will be cooperated with software. For example, an instruction to store the result of ALU into either the W (working) register or the F (file register) will require addition control lines to deal with this feature. The design of the processor will require two control lines to enable their corresponding selection. The thinking process is to design a tool (the computer) such that the tool can fulfill the user's (the programmer) need. The programmer's need has been familiarized because students have taken the course in phase two which is from the user's (the programmer) perspective. The design of the tool is to fulfill the programmer's need and so is the purpose of the third phase. In phase three, the same PIC16F architecture will be revisited. PIC16F has been explored before in phase two and now it will be designed by using a HDL. The example [12] will demonstrate the use of Verilog to implement a PIC16F628 as our example.

There are four clock cycles in one PIC16 instruction cycle. This requirement is to implement the subroutine type of program execution as well as interrupt. The first and most important building module in a computer is the register file. For a register file, it can be created uniformly so it has a write enable, a clear and a synchronization clock. The W register file is the special type of accumulator commonly used in CISC computers. Many registers including the W can be modeled with this generic register module. The PIC microcontroller also provides a powerful type of instruction which can store the result of ALU to the W register or a file register by specifying its destination. Refer to the PICF628A datasheet as in this link [13]. The ALU of PIC16F has two operands. One is always the W and the other can be either a literal data or from a file register. This option lets us design the ALU with an additional control signal to choose its destination (W of F). The instruction decoder is another important circuit for students to implement. Based on the instruction, the destination of the datapath can be either W or F, so there will be two control lines. The source of data can be literal or from file registers and so can the destination. The ALU will have its function code to select a function of the ALU, and the instruction decoder will generate corresponding control signals. For interrupt handling, the subsequently read and decoded instruction will have to be voided and should use the same method as processing subroutine to update the PC (Program Counter). In summary, there will be four clock cycles required to handle one instruction because of the requirement to handle interrupt and subroutine call. The implementation in FPGA in phase three is used with this example [12].

5. Conclusion

This project is still a work-in-progress. The teaching sequence can be implemented with an example a follows.

First, teach basic digital electronics and systems.

Second, teach PIC16F microcontroller or similar small µC or µP in assembly language.

Third, use HDL to model an interim and simpler processor first, then model the PIC16F which is taught in the phase two. Program the HDL-modeled processor to preform operation and compare what was learned in the phase two.

The proposed method engages the use-phase before the design-phase. The µC or µP to be designed was used before so the hurdle in the use phase is completely removed. With the designer's familiarity and availability of the software assembler and simulator, this proposed method should promote a better learning effectiveness compared with academic processors or models exemplified in some textbooks.

Reference

[1]. Continuous Improvement in Teaching Microprocessor Systems Design A Review of Efforts in Using different Tools, Techniques and Methods to Satisfy Students' Needs, by Jie Sheng, published on 2020, ASEE.

[2]. The Construction of a New MCU Experiment Platform, by Yulan Qi, published on 2010 International Conference on E-Health Networking, Digital Ecosystems and Technologies.

[3]. Teaching and Curriculum Development of Microprocessor Classes, by Roman Stemprok, published on 2000, ASEE.

[4]. Three-dimensional auxiliary teaching course study about MCU, by SuZhihua, LiMin, LiuYing, published on 2013 Fourth International Conference on Digital Manufacturing & Automation.

[5]. Towards the implementation of a personal learning environment in microprocessors by DIY method, by P. J. Sotorrío Ruiz, F. D. Trujillo Aguilera, et al, published on 2014 XI Tecnologias Aplicadas a la Ensenanza de la Electronica (Technologies Applied to Electronics Teaching) (TAEE)

[6]. Implementation of a hardware and software framework for a simple academic processor, by J. Ruiz, D. Guerrero, I. Gomez and J. Viejo, published on 2012 Technologies Applied to Electronics Teaching (TAEE)

[7]. Transforming the Microprocessor Class: Expanding Learning Objectives with Soft Core Processors, by Lynne Slivovsky and Albert Liddicoat, published on 2007 ASEE

[8]. Design based tutorials for System-on-Chip teachings, by Mouna Nakkar on 2009 International Conference on Engineering Education (ICEED)

[9]. Biprocessor SoC in an FPGA for Teaching Purposes, by Joaquín Olivares, Juan Gómez, José M. Palomares, and Miguel A. Montijano on Eighth IEEE International Conference on Advanced Learning Technologies

[10]. M Morris. Mano, et al., Logic and Computer Design Fundamentals, 5th ed. Hoboken, NY: Pearson Higher Education, 2016.

[11]. https://rawgit.com/vtsman/ECE-2504-assembler/noLabels/index.html

[12]. https://github.com/kiwih/pic16f-antastic/

[13]. https://ww1.microchip.com/downloads/en/DeviceDoc/40044G.pdf

[14]. David Money Harris and Sarah L. Harris, Digital Design and Computer Architecture, 2nd ed. Elsevier, MA: Elsevier, 2013.

[15]. David Patterson and John L. Hennessy, Computer Organization and Design, The Hardware/Software Interface, 5th ed. Elsevier, MA: Elsevier, 2014.

[16]. Enoch O. Hwang, Digital Logic & Microprocessor Design with Interfacing, 2nd ed. Cengage Learning, MA: 2018.