Spring 4-23-2019

MODERNIZATION OF LABORATORY CURRICULUM FOR THE UNDERGRADUATE DIGITAL SYSTEMS COURSE

Brolyne H. Onyango
University of Texas at Tyler

Recommended Citation

Onyango, Brolyne H., "MODERNIZATION OF LABORATORY CURRICULUM FOR THE UNDERGRADUATE DIGITAL SYSTEMS COURSE" (2019). *Electrical Engineering Theses.* Paper 42.
http://hdl.handle.net/10950/1310

MODERNIZATION OF LABORATORY CURRICULUM FOR THE UNDERGRADUATE DIGITAL SYSTEMS COURSE

by

BROLYNE HAWKAN ONYANGO

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Electrical Engineering
Department of Electrical Engineering

Mukul Shirvaikar, Ph.D., Committee Chair

College of Engineering and Computer Science

The University of Texas at Tyler
May 2019

The University of Texas at Tyler
Tyler, Texas

This is to certify that the Master's thesis of

BROLYNE HAWKAN ONYANGO

has been approved for the thesis requirements on
April 3rd, 2019
for the Master of Science in Electrical Engineering

Approvals:

[illegible]

Thesis Chair: Dr. Mukul V. Shirvaikar, Ph.D.

[illegible]

Member: Dr. Prabha Sundaravadivel,Ph.D.

Ron J. Pieper

Member: Dr. Ron J. Pieper, Ph.D.

[illegible]

Chair: Dr. Hassan El-Kishky, Ph.D.

[illegible] FOR JK

Dr. Javier Kypuros, Ph.D.
Dean, College of Engineering.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my advisor Professor Dr. Mukul Shirvaikar, for being my mentor. I am grateful for the guidance, support, advice, encouragement and patience he has continuously offered me throughout my course and also to help me complete my thesis successfully. I would also like to thank Dr. Hector Ochoa for his guidance and support during my thesis. Finally I would like to thank my committee members, Professor Dr. Ron J. Pieper, Dr. Prabha Sundaravadivel, for serving as my committee members even at hardship. I also want to thank them for letting my defense be an enjoyable moment, and for your brilliant comments and suggestions.

A special thanks to my uncle Elisha and his family for their constant encouragement and financial support while pursuing my Master of Science in Electrical Engineering and for all the sacrifices that you've made on my behalf. I would also like to thank all my friends who supported me in writing and encouraged me to strive towards my goal. I would like to thank each and every person for their encouragement and support to complete my thesis work successfully.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

MODERNIZATION OF LABORATORY CURRICULUM FOR THE UNDERGRADUATE DIGITAL SYSTEMS COURSE

Brolyne Hawkan Onyango

Thesis Chair: Mukul Shirvaikar, Ph. D.

The University of Texas at Tyler

May 2019

The job market is increasingly in need of Electrical Engineers with knowledge of Hardware Description Languages, yet the school curriculum on digital systems design, largely remains behind using the traditional gates to implement logic design. While this may have been favorable in the past, technological advances in the industry and availability of inexpensive hardware and software, where by students can begin appreciating the importance of Hardware Description Language (HDL) is on the rise. It is for this reason that this thesis aims to improve the quality of the digital Systems designs course by modernizing it and introducing the new concepts of HDL while at the same time making the transition easy for the student to grasp.

A novel hybrid approach is introduced to modernize the curriculum for the Digital Systems course, using traditional circuit construction, simulation software and implementation of circuits using reconfigurable logic. The NI Multisim circuit simulation software and a Digilent Basys-3 board are utilized. Students will use a breadboard and chips to construct basic combinational circuits using logic gates. Next, they will use the NI Multisim to build and simulate circuits that are difficult to build physically. Finally, a Field Programable Gate Array (FPGA) board will be utilized to implement the most complex circuits. Typically, the use of FPGA technology requires knowledge of HDL, which is considered too advanced for most sophomores. This problem is addressed by designing the complex circuits using the NI Multisim software graphical design

addressed by designing the complex circuits using the NI Multisim software graphical design suite. The students will only build the schematic circuit in the Multisim software with graphical pick-and-place components. The software suite is already equipped with a special plug-in application with translation capabilities that allows the students to download the circuit to the FPGA board. The board is equipped with enough peripherals to implement the circuits and provide an exciting experience for the students. Several labs will be designed using this approach and there will be a larger project at the end of the course. Our hybrid approach will familiarize the students with modern tools and design paradigms. Moreover, the observation of snippets of basic HDL code will lay the foundation for the study of this topic, which the students may learn later in advanced digital systems courses.

CHAPTER ONE

Introduction

All equipment used in military, medicine, manufacturing, entertainment and telecommunications is either partially or completely digital in nature. Therefore, Digital System Design is a fundamental course in Electrical and Electronics Engineering. It is in Digital Systems that students learn the basics of computer engineering. The students are introduced to fundamental concepts such as logic gates, Boolean algebra, design of combinational and sequential circuits that are the building blocks for computer circuits [1]. The course is traditionally taught using TTL gates where the students wire IC chips on a breadboard. However, when building more complex circuits it becomes increasingly difficult to build and debug the circuits due to the number of chips involved and the connections required. Circuit simulations are then performed for such circuits.

The job market today needs engineers proficient in Hardware Description Languages (HDL) since it is the market standard for Digital Systems Design. A survey meeting by EET responsible for industrial training of electrical engineers indicates that there is an increasing need for engineers with knowledge on HDL and reconfigurable logic [2]. The availability of complex programmable logic devices (CPLD) and field programmable gate arrays (FPGA) has today changed the world of digital systems design. It is now possible to design a digital system and employ it on a reconfigurable logic device. Further technological advances have led to production of inexpensive configurable devices that can be used for instruction in school.

1.1 Introduction to FPGA

A **field-programmable gate array (FPGA)** is an integrated circuit designed to be configured by a customer or a designer after manufacturing – hence the term "field-programmable". The FPGA configuration is generally specified using a hardware description language (HDL), similar to that used for an application-specific integrated circuit (ASIC). Circuit diagrams were previously used to specify the configuration, but this is increasingly rare due to the advent of electronic design automation tools. FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together", like many logic gates that can be inter-wired in different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory. Many FPGAs can be reprogrammed to implement different logic functions, allowing flexible reconfigurable computing as performed in computer software.

1.2 Objective of the thesis

This thesis aims to create a smooth transition from the traditional circuit building on breadboard to programming using HDL and FPGA boards to achieve logic designs. Many schools have done it by introducing HDL courses in advanced level of studies. Unfortunately, there are problems with increasing the number of credit hours for courses as government regulations are limiting the number of credit hours assigned to courses. There are also challenges on the timing with which to introduce the concepts of HDL programming. Introducing it in the sophomore year poses the challenge of course complexity since the students barely have any programming skills. Introducing HDL in junior year has the problem of burdening the students with a lot of information. Our approach for modernizing the curriculum solves these problems by giving the students a taste of the HDL and FPGA boards without having the necessary programming skills. It acts as a primer to advanced digital systems concepts that the students will study later in their career thus making the transition from basic logic to complex logic via programming smooth.

1.3 Organization of the thesis

This thesis is organized into 5 chapters. Chapter 1 introduces the topic of FPGA programming and the importance of having it in the curriculum. Chapter 2 gives a literature survey on past work on introducing the FPGA and HDL in the school curriculum for engineers. Chapter 3 gives a technical background on how the FPGA board should be set up, the required software and setup procedure for programming. Chapter 4 gives the laboratory experiments that have been modified to include the use of FPGA. Chapter 5 gives the conclusion arrived at during the implementation of this curriculum.

CHAPTER 2

Literature Survey

The idea of using FPGA's to teach digital logic design is not a new one, and many schools have tried to introduce the concept using various methods. A survey into past work shows many universities have taken different approaches toward introducing the HDL, FPGA concepts into their curriculum.

Mayer [2] uses an Altium Designer program together with Nano 3000 FPGA board to introduce a digital designs course with VHDL to undergraduate studies. Altium developed an electronic design automation EDA software called Altium designer. Altium designer unifies the design of printed circuits, the design of FPGA's and embedded system programming. The Nano 3000 board is based on the Xilinx's SPARTAN-3AN FPGA device. The project is sponsored by EET (Electronics Engineering Technology) advisory committee of Pittsburg State University, Kansas and Altium Limited in a bid to supply the job market with HDL skilled labor. The students used VHDL for two purposes to simulate a digital circuit they had designed and to synthesize the circuit by configuring an FPGA device. The pilot program was first tested as a graduate course in the Fall of 2011 at the Pittsburg State University. The curriculum also contained a project that the students had to complete by the end of the semester. The project chosen was a real-world problem and in this case was a bat counter which was developed by University of Pittsburg's Biology department. The project involved using sensors well placed so that whenever a bat flies between the beams produced by the sensors the beam is cut off and a count is initiated by the FPGA. The following tools were made available for the students to use during their project; Altium Designer suite (EDA), Nano Board 3000, "VHDL: A Starters Guide by Sudhakar Yalamanchil."

At the end of the course, a survey was done to establish how well the students grasped the concepts; configure FPGA's using Altium Designer EDA, configure FPGA's using VHDL, configure and program soft processors in an FPGA, use VHDL testbenches and virtual instruments to test and verify FPGA. The results show that 90% of the students used VHDL to implement their project. However, none of them used the soft cores and open bus to create their project. Using the soft microprocessors and the open bus would have allowed their projects to be more functional than simply using VHDL and FPGA only.

Carrol *et al*. of University of Texas at Arlington introduced a hierarchical project-based introduction to digital logic design course for their computer engineering and computer science course in the fall of 2103 [3]. In their approach the basic concepts of digital logic design were introduced as a building block towards creating a 4-bit Tiny Reduced Instruction Set Computer (TRISC). The course *CSE 2441 Introduction to Digital Logic* is a sophomore level course with 4 semester credit hour (SCH) taught in a 3-hour lecture and 3-hour laboratory per week format. In this approach the authors required the students to have taken a computer programming class as a prerequisite. The course used the Altera's Quartus 2 design software to capture and simulate all module designs. Less complex designs were implemented on a solderless board while the more complex designs were implemented using an Altera Cyclone 2 FPGA on the DE1 development board. Completed modules were subsequently integrated hierarchically to realize the CPU. This approach however required that the students use HDL (Verilog) in the implementation of modules on the FPGA board. In their findings after a survey of the course they realized most students had difficulty understanding the lab implemented on the FPGA due to their limited programming experience.

Rogriguez-Ponce et al. of the Universidad Automona de Queretaro, Mexico modernized their digital design course DDE 128 to include concepts of FPGA and VHDL. The 21-week semester course was offered during the fifth semester of the Automation and Control department of Universidad Automona de Queretaro with six hours of class lecture and two hours of laboratory work per week [4]. The course integrated the teaching of VHDL to go hand-in-hand with teaching of digital design concepts. An introduction to programming class is a prerequisite for the class. The students choose to use the DE2 Cyclone 2 FPGA board from Altera because it had more peripherals that were important for the complex mathematical algorithms that they were implementing. Some of the topics in digital design were like minimization using K-maps were scrapped because they no longer had any significance while using the FPGA. The problem with this approach as with the previous is that the students do not get enough exposure to the power of the FPGA board and VHDL because they are learning two concepts simultaneously. The course covers VHDL extensively and at the end of it they are expected to implement a project using the knowledge gained.

At Boise State University, Loo *et al* introduced FPGA and VHDL in their undergraduate digital systems design course covered in the sophomore year of the electrical engineering course [5]. Having realized that the industry requires more engineers with knowledge of FPGA designs, Loo et al modified the curriculum so that it incorporated both use of TTL gates to build digital circuits and the FPGA board. In the first 4 weeks of the course, students build digital designs using the 74xx series TTL chips by wiring them on a solderless breadboard. This reinforces the hands-on wiring and the modular design where students build complex circuits by wiring together simple components. After 4 weeks the students switch to the FPGA for design of circuits. The Xilinx ISE is chosen as the software for implementation and the Spartan-3 FPGA board. They ran into a couple of problems with the computers they used. They realized that to ran successful CAD for digital systems they had to upgrade their computers from the Pentium 4 they were using at the time to i5 to be able to run the labs smoothly. Since they were introducing VHDL programming to the course an introductory programming course was a prerequisite to take the course.

Inductive instruction using FPGA and VHDL is another approach chosen by Zhao and Huang of South Dakota school of Mines and Technology to enable students have a better understanding [6]. Previously, they realized that the students did not have any interest in Digital System Design course because they would not relate the concepts taught in class to any real-world scenarios. The traditional way of teaching, deductive teaching, which involves introducing theoretical concepts followed by mathematical concepts and finally solutions to math problems was not working for them. Thus, they introduced active and inductive techniques of teaching to encourage student interest in the course. In their Digital Design course CENG342 they started by introducing a project that will be implemented throughout the course. A seven-segment time-multiplexing circuit is developed on the Spartan 3 FPGA board using the Xilinx ISE software. Intensive VHDL is learnt in the course and at the end of it the students are required to complete a project that involves building MIPS digital processor to implement a small set of its instruction set. The slide switches on the FPGA board are used to load the 32-bit instructions 8 bits at a time. The instructions are executed, and the results and the operands displayed on the LEDs and the 7-segment display of the S3 board. The problem with this approach is that it requires extensive programming skills since the student will be fully engaged in the VHDL programming. At sophomore level this will be a challenge thus in many universities the introduction of HDL is introduced at later stages of the degree plan.

The research-based teaching was adopted by Texas A&M University Corpus Cristi for digital systems laboratories in a wholistic approach that include many components such as; microprocessors and microcontrollers, FPGA boards, CAD software and many more [7]. The aim of including all this is to create a laboratory curriculum that ensures the students have a vast knowledge that they will develop later when they take the microprocessors course and other advanced digital systems related courses. The FPGA is only used in a single lab where they program the Atlmel FPGA using VHDL.

Donzelini *et al*. of the University of Genoa, in Genova, Italy introduced learning Digital systems with DEEDS (Digital Electronics Education Design Suite) [8]. The Deeds has capability of converting a schematic circuit wired from the graphical interface to one that can be run on an FPGA. The students design the circuit on the graphical suite, test it using the virtual instruments available on the Deeds software, simulate the circuit in deeds then they can export the circuit onto the FPGA board. Deeds supports Quartus 2; DE1, DE2, DE2-115 boards by Terasic. In their course they eliminate the traditional circuit building on a breadboard and focus entirely of using Deeds to enhance student understanding of the course. In addition to the normal combinational and sequential circuits they include microprocessor interfacing and programming in assembly language in their course. Deeds has the capability of generating VHDL files for the circuits created by the students. Thus, they can examine the code see how the different circuit components interact. Additional features of Deeds are the slow clock mode which allows student to slow the internal clock of the FPGA and be able to see the working of sequential logic. This approach was introduced in the university in 2010 and the authors report an increased average evaluation from the students and the increased number of students that are interested in the course.

Shayesteh *et al.* of the Indiana University Purdue University – Indianapolis, improved their Introduction to Digital System Design course offered at sophomore level to include the use of VHDL and FPGA boards [9]. The 4-Semester credit hour course is a required course for the computer engineering and electrical engineering courses offered in the university. In the course they use the Vivado software from Xilinx together with Nexys 3 FPGA board from Digilent. The course focuses on introducing the students to VHDL as early as possible and make them get used to hardware description languages. Topics like number systems and logic minimization are all done in VHDL. Circuit logic using CMOS technology is also introduced in this course and implemented using HDL. The difference between this approach and the approach we are putting forward is that this approach does not include the use of the traditional bread board LED, TTL circuit construction. It jumps straight away to implementation in VHDL and FPGA. The first lab is a tutorial on Vivado and Nexys3 FPGA board. With little knowledge on programming, the students were unable to grasp some of the complex aspects of VHDL and thus in the later chapters the students did not do well.

In general, there is a push toward changing the digital systems curriculum towards the use of hardware descriptive languages. The biggest challenge is where in the course to introduce them and how deep should they be covered [10]. Most universities offer the introductory digital systems course in the sophomore year where the students barely have any programming skills hence it becomes a challenge to learn HDLs effectively. Moreover, the traditional TTL chip building on a bread board is still important so that students can appreciate how simple logic gates can be assembled to create more complex and meaningful circuits. It is therefore important to create a course that merges these concepts smoothly in a gradual and sensible manner to build from the gate level circuit building to circuit description using HDL as circuit complexity increases. It also very important for students to realize the circuits they build and for this the FPGA board is important. Our novel approach aims to achieve these.

CHAPTER 3

Technical Background

3.1 FPGA Background

FPGA technology came from Programable read-only memory (PROM) and programmable logic devices (PLDs) which could be programmed from the factory on in the field. Programable logic was hard-wired between logic gates. The FPGA is therefore designed to be configured by the customer or designer after it has been manufactured. Hardware Description Languages are used to configure or design circuits in the FPGA board. FPGAs contain an array of programmable logic blocks, and a hierarchy of reconfigurable interconnects that allow the blocks to be "wired together", like many logic gates that can be inter-wired in different configurations. Logic blocks can be configured to perform complex combinational functions, or merely simple logic gates like AND and XOR. In most FPGAs, logic blocks also include memory elements, which may be simple flip-flops or more complete blocks of memory. Many FPGAs can be reprogrammed to implement different logic functions, allowing flexible reconfigurable computing as performed in computer software.

3.2 Basys 3 FPGA Board

The Basys 3 is an entry-level FPGA development board designed exclusively for Vivado Design Suite, featuring Xilinx Artix-7 FPGA architecture. Basys 3 is the newest addition to the popular Basys line of FPGA development boards and is perfectly suited for students or beginners just getting started with FPGA technology. The Basys 3 includes the standard features found on all Basys boards: complete ready-to-use hardware, a large collection of on-board I/O devices, all required FPGA support circuits, a free version of development tools, and at a student-level price point.

Below are the features of the Basys 3 board:

- Features the Xilinx Artix-7 FPGA: XC7A35T-1CPG236C
- 33,280 logic cells in 5200 slices (each slice contains four 6-input LUTs and 8 flip-flops)

- 1,800 Kbits of fast block RAM
- Five clock management tiles, each with a phase-locked loop (PLL)
- 90 DSP slices
- Internal clock speeds exceeding 450 MHz
- On-chip analog-to-digital converter (XADC)
- Digilent USB-JTAG port for FPGA programming and communication
- Designed Exclusively for Vivado Design Suite (Vivado Design Suite WebPACK edition can be downloaded for free from Xilinx). Expanded features are available through purchase of the Design Edition.
- Free WebPACK™ download for standard use.
- Micro-B USB cable not included.
- Serial Flash
- USB-UART Bridge
- 12-bit VGA output
- USB HID Host for mice, keyboards and memory sticks
- 16 user switches
- 16 user LEDs
- 5 user pushbuttons
- 4-digit 7-segment display
- 4 Pmod ports: 3 Standard 12-pin Pmod ports, 1 dual purpose XADC signal / standard Pmod port

The Basys 3 board was chosen for this course because it is a relatively cheap board and has all the necessary features for an entry level learner of the FPGA. The ease of integration with the Multisim further supports the use of the board. A picture if the board is shown in figure 1 below.

3.3 Software requirements

In order to run the laboratories in this course we need a few software packages:

3.3.1 NI Multisim Design Suite

Multisim Design Suite Student Edition is required for the successful implementation of the proposed curriculum. The success of this approach is hinged on the fact that Multisim can convert a graphical design made through component pick and place through a graphical interface into HDL code that can be run on an FPGA board. Multisim 14.1 and newer versions have the "create PLD" option that is required to create a circuit to be implemented on an FPGA board.

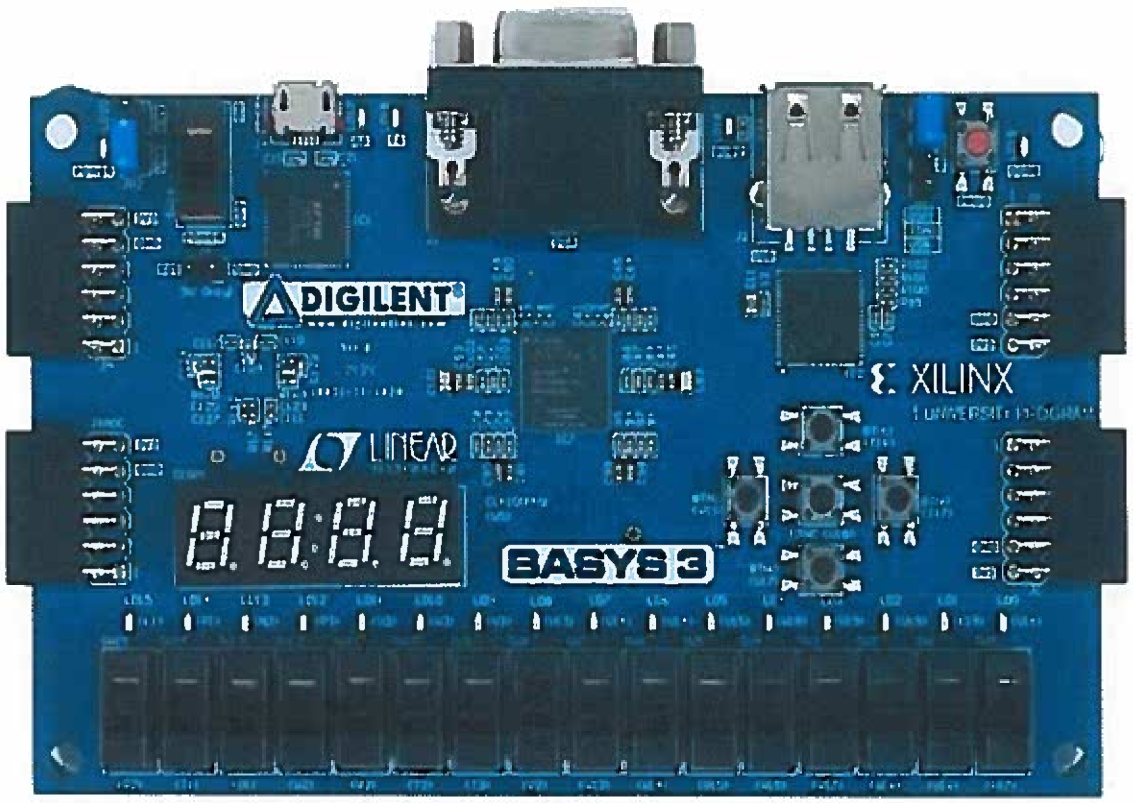


Figure 1: Basys 3 FPGA Board

Multisim 14.0.1 Education and later comes with built-in PLD support for the NI Digital System Development Board (DSDB) and many other Digilent Boards. This support includes a PLD configuration file that defines the names and properties of port connectors that a Multisim PLD design will use.

3.3.2 NI FPGA Plug in

The translation capability of Multisim is provided by a special plug-in from NI Multisim that can be downloaded for free from the NI website [11]. The plug-in is chosen for the particular operating system properties either 32-bit or 64-bit.

3.3.3 Adept Drivers

In addition to Multisim the board drivers are necessary of operation of the board. This driver can be downloaded for free at the Digilent Adept website for free to configure the Basys 3 ready for use [12].

3.4 Setting up the PLD design

Once the required software and driver is installed, you are ready to program the FPGA board. The steps below in figures 2 to 12 describe the process for creating a PLD design for the Basys 3 FPGA board, and the same steps can be used for the other boards.

1. In Multisim, select **File»New**.

2. Click the **PLD Design…** the click the **Create** button.

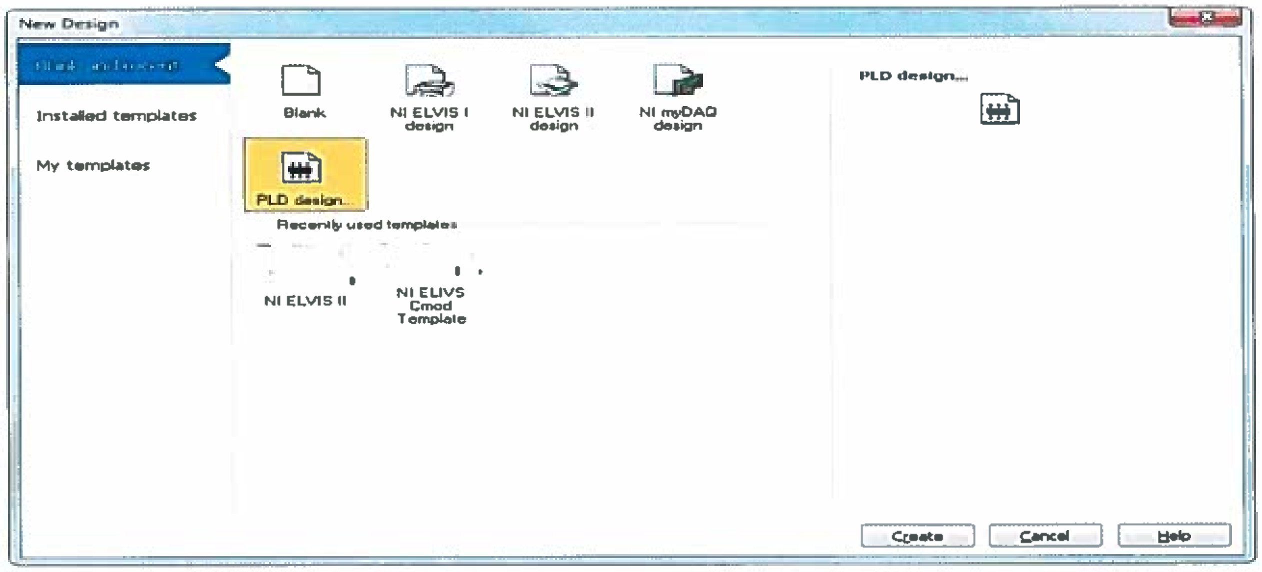
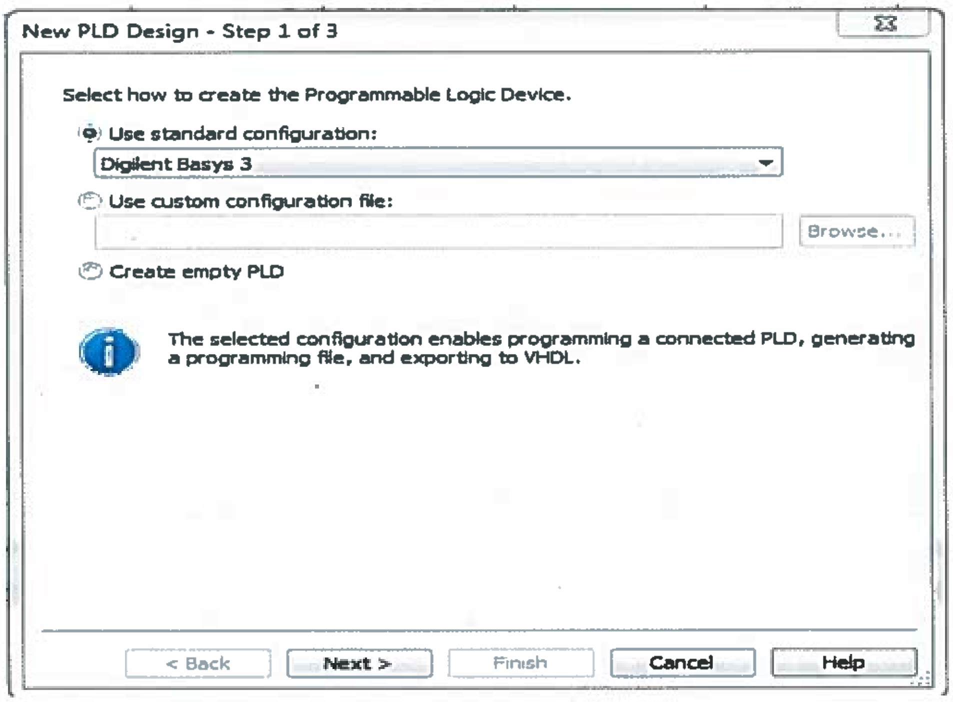


Figure 2: Choosing the PLD design on Multisim

3. Click the **Digilent Basys 3** down arrow to select your board. Click **Next**.


New PLD Design - Step 1 of 3

Select how to create the Programmable Logic Device.

(●) Use standard configuration: Digilent Basys 3

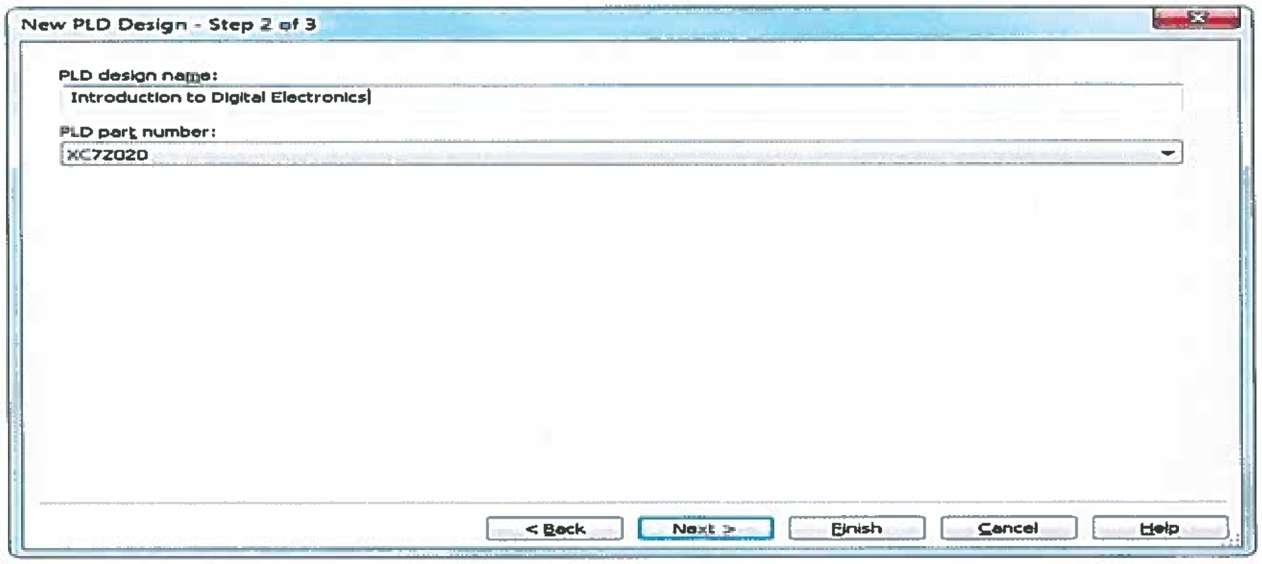
( ) Use custom configuration file:

( ) Create empty PLD

The selected configuration enables programming a connected PLD, generating a programming file, and exporting to VHDL.

Figure 3: Selecting the Board on Multisim

4.Enter *Introduction to Digital Electronics* in the PLD design name field and click **Next**.

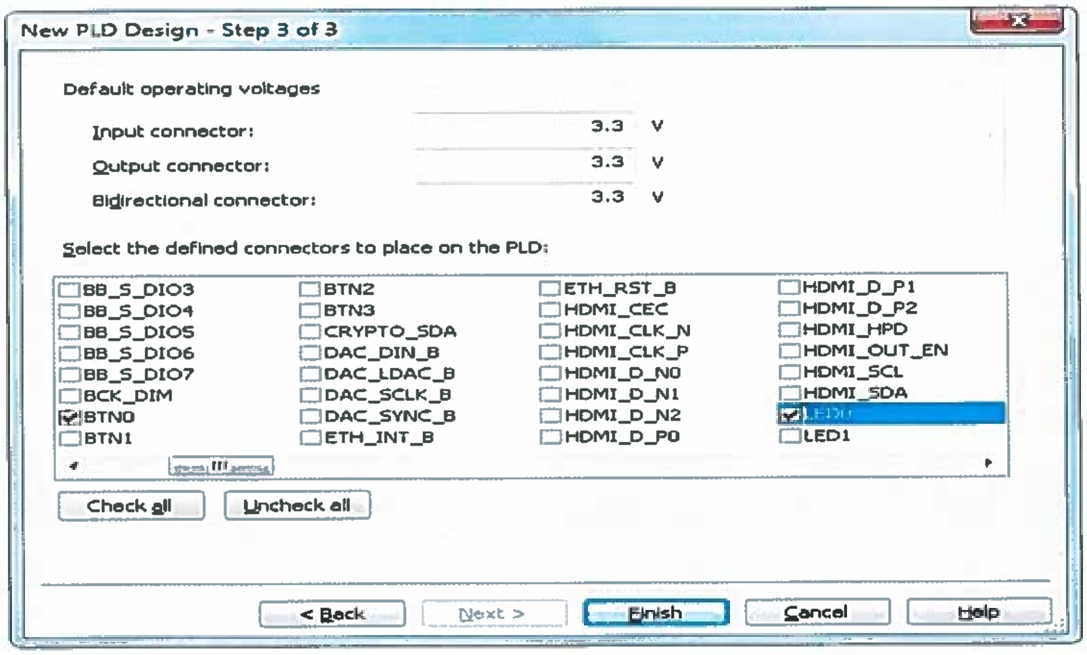

New PLD Design - Step 2 of 3

PLD design name: Introduction to Digital Electronics

PLD part number: XC7Z020

Figure 4: Name the design

5. The **New PLD Design** dialog allows you to select which peripherals you will use in your design. In this tutorial the LED LED0 and the push button BTN0 are selected. Click **Finish**.



New PLD Design - Step 3 of 3

Default operating voltages

Input connector:	3.3	V
Output connector:	3.3	V
Bidirectional connector:	3.3	V

Select the defined connectors to place on the PLD:

☐ BB_S_DIO3	☐ BTN2	☐ ETH_RST_B	☐ HDMI_D_P1
☐ BB_S_DIO4	☐ BTN3	☐ HDMI_CEC	☐ HDMI_D_P2
☐ BB_S_DIO5	☐ CRYPTO_SDA	☐ HDMI_CLK_N	☐ HDMI_HPD
☐ BB_S_DIO6	☐ DAC_DIN_B	☐ HDMI_CLK_P	☐ HDMI_OUT_EN
☐ BB_S_DIO7	☐ DAC_LDAC_B	☐ HDMI_D_N0	☐ HDMI_SCL
☐ BCK_DIM	☐ DAC_SCLK_B	☐ HDMI_D_N1	☐ HDMI_SDA
✓ BTN0	☐ DAC_SYNC_B	☐ HDMI_D_N2	✓ LED0
☐ BTN1	☐ ETH_INT_B	☐ HDMI_D_P0	☐ LED1

Figure 5: Selecting the peripherals on the board

6. The selected connectors are placed on the work space.

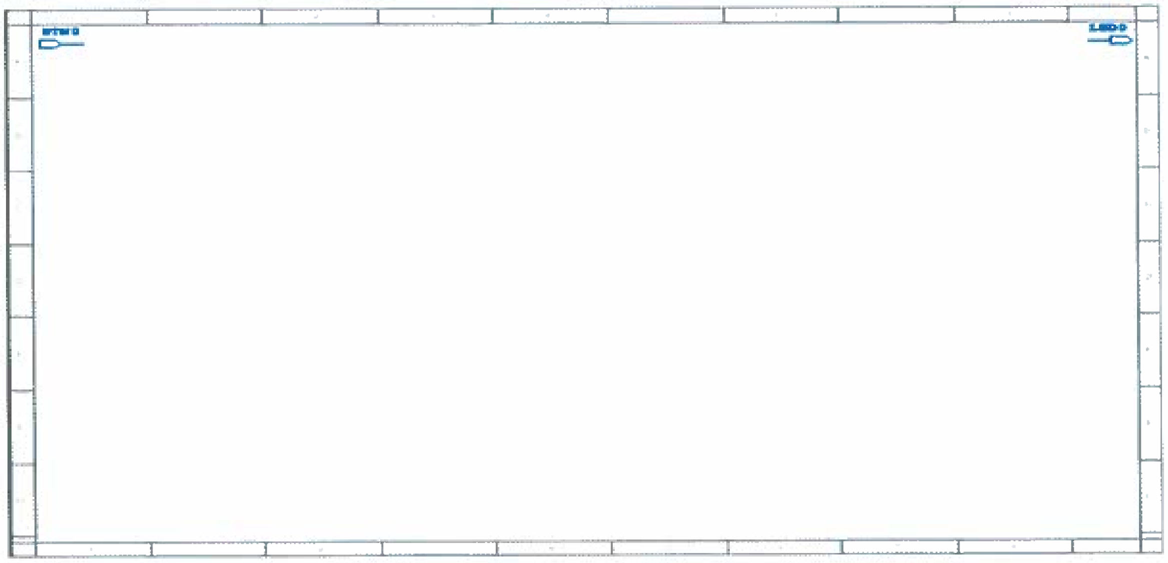
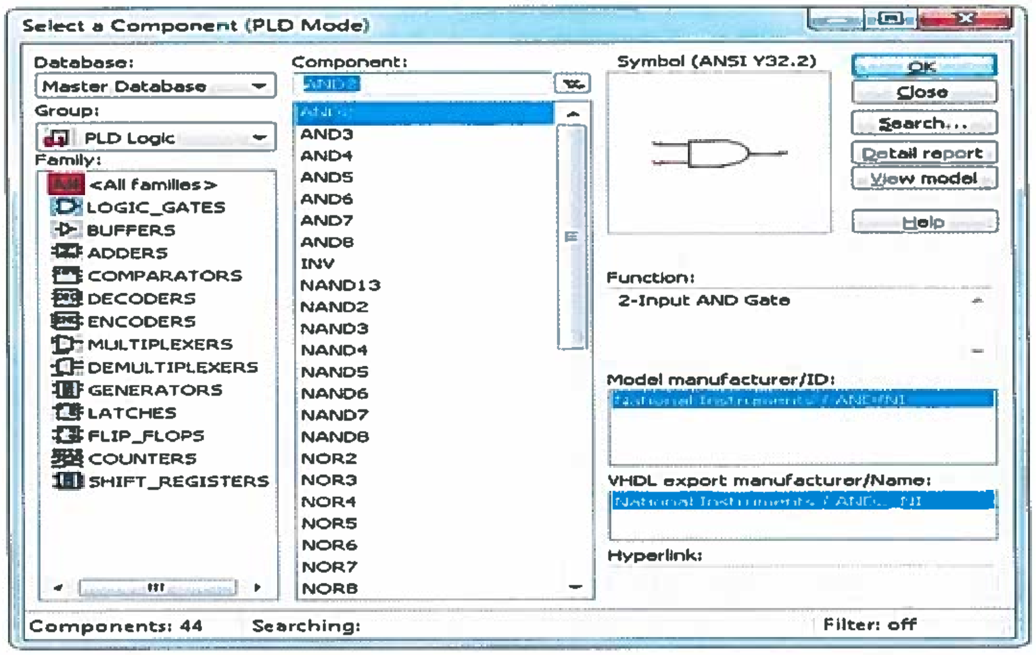


Figure 6: Selected Peripherals as they appear on the empty PLD design

3.4.2 Create a PLD Schematic in Multisim

1. Select Place»**Component.**

2. Select an **AND2** gate located in the **PLD Logic** group, **Logic_gates** family and click the **OK** button.


Select a Component (PLD Mode)

Database: Master Database

Group: PLD Logic

Family:
<All families>
LOGIC_GATES
BUFFERS
ADDERS
COMPARATORS
DECODERS
ENCODERS
MULTIPLEXERS
DEMULTIPLEXERS
GENERATORS
LATCHES
FLIP_FLOPS
COUNTERS
SHIFT_REGISTERS

Component: AND2
AND2
AND3
AND4
AND5
AND6
AND7
AND8
INV
NAND13
NAND2
NAND3
NAND4
NAND5
NAND6
NAND7
NAND8
NOR2
NOR3
NOR4
NOR5
NOR6
NOR7
NOR8

Symbol (ANSI Y32.2)

Function: 2-Input AND Gate

Model manufacturer/ID: National Instruments / AND2NI

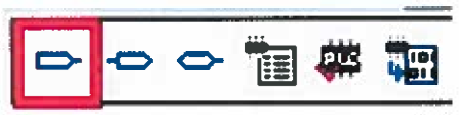
VHDL export manufacturer/Name: National Instruments / AND2_NI

Hyperlink:

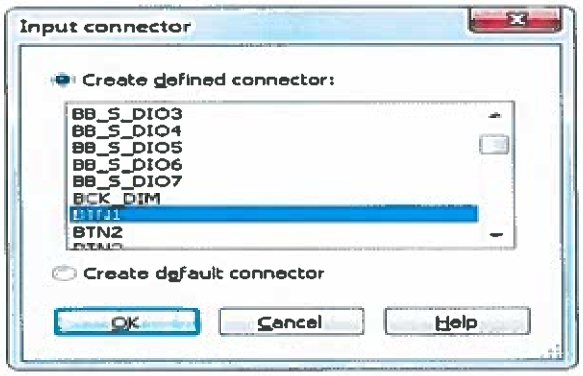
Components: 44 Searching: Filter: off

Figure 7: Selecting circuit components pick and place

3. Place another connector for the AND gate input by click the **Input connector** icon on the toolbar.



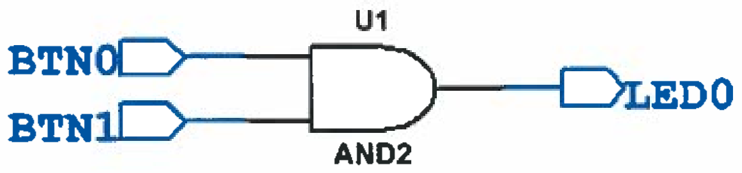
4. Select the push button **BTN1** and click **OK**.


Input connector

[✓ Create defined connector:]
BB_S_DIO3
BB_S_DIO4
BB_S_DIO5
BB_S_DIO6
BB_S_DIO7
BCK_DIM
BTN1
BTN2

Create default connector

5. Wire the AND gate to the connectors.



NOTE: The second button BTN1 could have been chosen while creating the schematic by choosing two buttons instead of one but here, it is created to show that it is possible to create your own button if you did not choose one while choosing the peripherals

3.4.3. Export the PLD Design to the FPGA

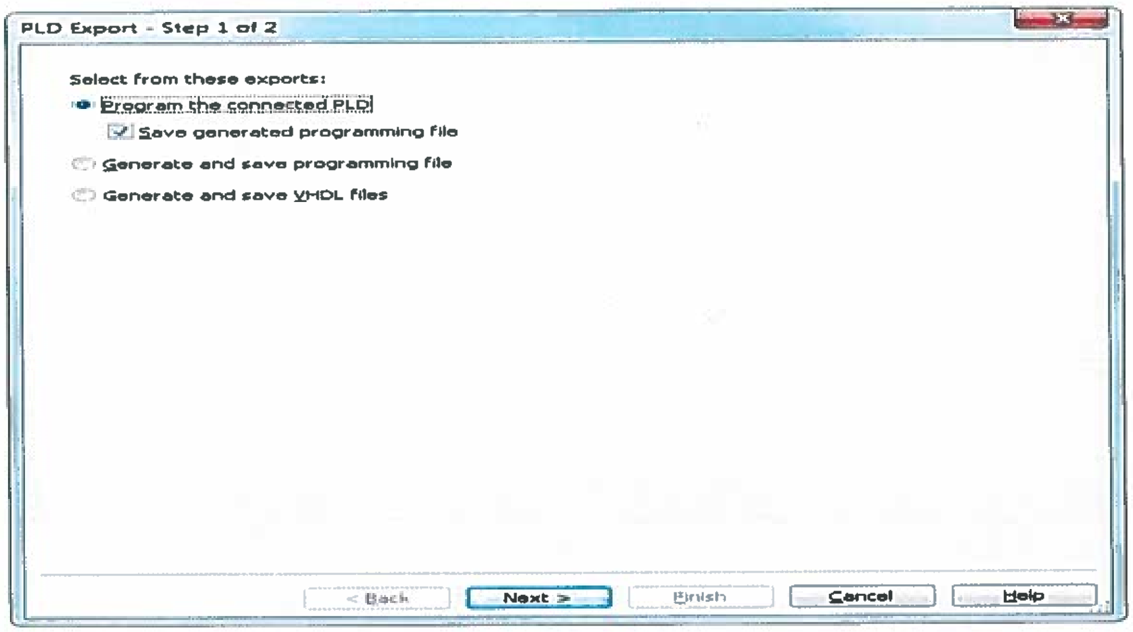
There are three options for exporting the digital logic from the PLD schematic:

- Programming the connected PLD – Allows students to deploy the design directly to the FPGA.
- Generate and save a programming file- Students can generate a bit file that can be used to program hardware later.
- Generate and save the VHDL- This option exports the VHDL netlist allowing students to view the VHDL code. You can import the VHDL code in the Xilinx environment and program the FPGA

In this tutorial you will program the FPGA board directly from the Multisim environment.

1. Select **Transfer»Export to PLD.**

2.Click the **Program the connected PLD** radio button and click **Next.**

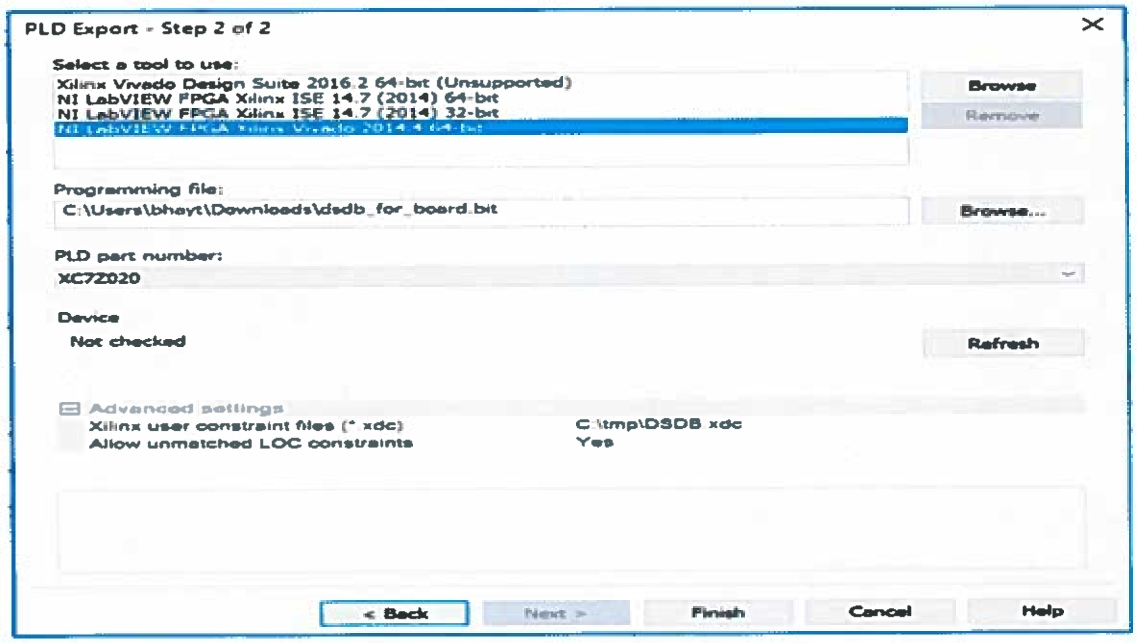

PLD Export - Step 1 of 2

Select from these exports:

- (●) Program the connected PLD
  - [✓] Save generated programming file
- ( ) Generate and save programming file
- ( ) Generate and save VHDL files

Figure 8: Generating the VHDL file

3. In the **Select a tool to use area**, select the Xilinx tool for you board.


PLD Export - Step 2 of 2

Select a tool to use:

Xilinx Vivado Design Suite 2016.2 64-bit (Unsupported)
NI LabVIEW FPGA Xilinx ISE 14.7 (2014) 64-bit
NI LabVIEW FPGA Xilinx ISE 14.7 (2014) 32-bit
NI LabVIEW FPGA Xilinx Vivado 2014.4 64-bit

Programming file:
C:\Users\bhayt\Downloads\dsdb_for_board.bit

PLD part number:
XC7Z020

Device
Not checked

Advanced settings

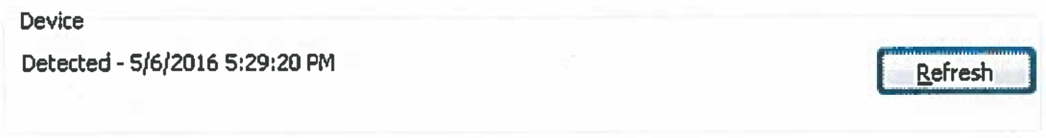
Xilinx user constraint files (*.xdc)	C:\tmp\DSDB.xdc
Allow unmatched LOC constraints	Yes

Figure 9: Exporting the PLD design

4. Connect the hardware to your computer and wait for Windows to detect the connection.

5. Make sure power is applied to the board and the power switch is set to the on position.

6. Click the **Refresh** button. The **Detected** message will appear if the board is detected by your computer.


Device

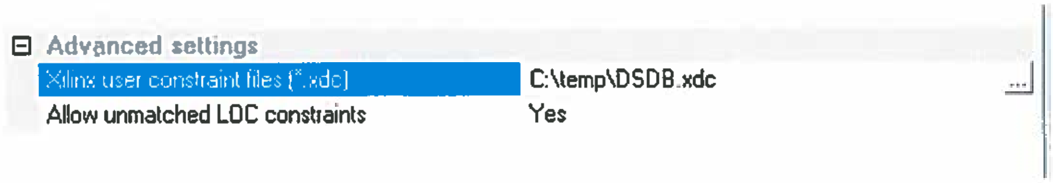
Detected - 5/6/2016 5:29:20 PM

Figure 10: Establishing connection between computer and the board

7. Click the **Finish** button to begin programming the board.

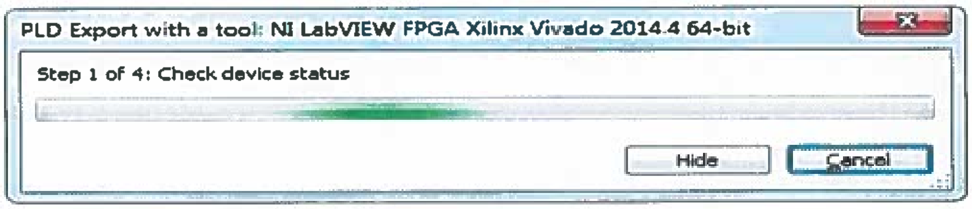
Note: Some Vivado versions do not support file path containing spaces for the XDC file. In these cases, you will get an "illegal file or directory name" error when trying to export the design. If this happens, copy the DSDB.xdc file stored in the installation folder:

`<Program Files>\National Instruments\Circuit Design Suite 14.0\pldconfig` to a local path such as `C:\temp`. Next, change the Xilinx user constraint file (*.xdc) in the Multisim PLD Export step 2 of 2 dialog to where you saved the file before exporting the design.


Advanced settings

Xilinx user constraint files (*.xdc)	C:\temp\DSDB.xdc
Allow unmatched LOC constraints	Yes

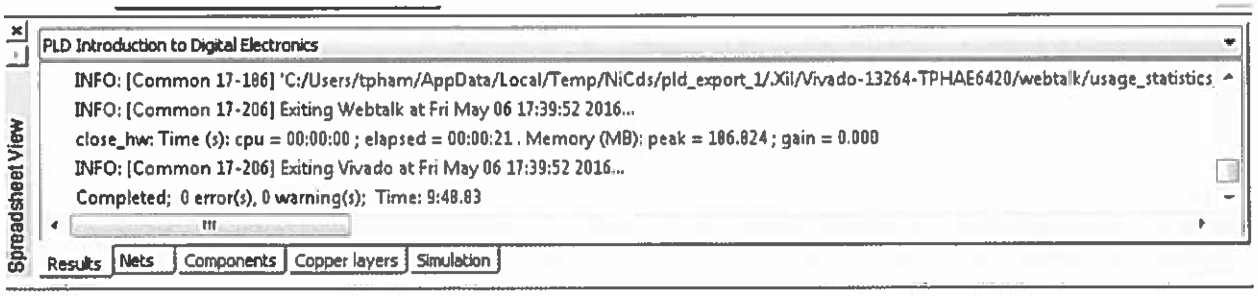
8. Multisim will automatically open the Xilinx tool in the background and perform all the requires steps to program the FPGA, no user interacting is required.


PLD Export with a tool: NI LabVIEW FPGA Xilinx Vivado 2014.4 64-bit

Step 1 of 4: Check device status

Figure 11: Programming the FPGA board

9. Once the FPGA is programmed, Multisim will display a message on the Spreadsheet View.



```
INFO: [Common 17-186] 'C:/Users/tpham/AppData/Local/Temp/NiCds/pld_export_1/.Xil/Vivado-13264-TPHAE6420/webtalk/usage_statistics
INFO: [Common 17-206] Exiting Webtalk at Fri May 06 17:39:52 2016...
close_hw: Time (s): cpu = 00:00:00 ; elapsed = 00:00:21 . Memory (MB): peak = 186.824 ; gain = 0.000
INFO: [Common 17-206] Exiting Vivado at Fri May 06 17:39:52 2016...
Completed; 0 error(s), 0 warning(s); Time: 9:48.83
```

Figure 12: Programming complete

10. You can now test the design built in Multisim on the real-hardware.

CHAPTER 4

The Curriculum

4.1 Current Curriculum

EENG 3302 Digital Systems offered at University of Texas in Tyler is a 3-credit hour course with 2-hours lecture and 3-hour laboratory per week. The main objectives of the course are:

- To design digital systems using simple logic elements.
- To demonstrate knowledge of sequential logic circuit elements like flip-flops, and latches and their applications.
- To demonstrate knowledge of advanced circuits like counters and registers
- To write laboratory reports with experimental results demonstrating visual and written communication skills.

Topics covered in the course are:

1. Introductory digital concepts
2. Number systems, operations and codes
3. Logic gates
4. Boolean Algebra and logic simplification
5. Karnaugh Maps
6. Combinational Logic
7. Flip-Flops and related devices
8. Counters, Shift registers, and sequential logic
9. Memory and storage
10. Introduction to microprocessors
11. Integrated Circuit technologies.

The laboratory exercises covered in this course are summarized in the Table 1 below.

Table 1: Summary of laboratory exercises currently offered in the EENG 3302 course

No	Lab Title	Learning Objective	Implementation
0	Instruments	To familiarize the students to the lab instruments.	
1	Logic gates and Boolean Algebra.	Test and observe the working of basic gates AND, OR, NAND, XOR, NOR, NOT.	On the breadboard using TTL chips and LED's.
2	De Morgan's Theorems	To verify De Morgan's Theorems of Boolean Algebra.	On the breadboard using TTL chips and LED's.
3	Combination Circuits	To simplify Boolean expressions to the and construct the simplest form and compare with the original form.	Simulate on Multisim using Word generator and Logic analyzer
4	Universal property of NAND & NOR	Verify the use of the universal gates as AND, NOT and OR gates.	On the breadboard using TTL chips and LED's.
5	Adders and Multiplexers	To implement an ADDER and a Data Selector and analyze its output.	Simulate on Multisim Software using Word generator and Logical analyzer
6	Encoders and Decoders	To implement an Encoder and a Decoder circuit.	Simulate on Multisim using Word generator and Logic analyzer
7	Seven Segment Display	To implement a BCD decoder with the seven-segment display.	Simulate on Multisim using Word generator and Logic analyzer
8	Comparators	Implement a comparator circuit	Simulate on Multisim using Word generator and Logic analyzer
9	Look Ahead Carry Adders	Implement the Look ahead carry adder.	Simulate on Multisim using the Word generator and Logica analyzer
10	Arithmetic and Logic Unit	Implement a simple ALU.	Simulate on Multisim using Word generator and Logical analyzer
11	Latches and Flipflops	Implement the flipflops	Simulate on Multisim using Word generator and Logic analyzer
12	Counters	Implement a BCD decade counter	Simulate on Multisim Software using Word generator and Logic analyzer

In the current curriculum, laboratory implementation is majorly two-fold, either through wiring of TTL basic gates on a solderless bread board or by simulation on NI Multisim and analyzed using a word generator and logic analyzer. The simple circuits are implemented by wiring the circuit components in the form of TTL chips on a solderless bread board and the circuit functionality is tested by connecting LEDs and switch buttons. As the circuits get more complex it becomes increasingly difficult to wire them on a bread board and some of the sequential circuits will not work properly if a lot of wiring is involved. In such instances the circuits are simulated on Multisim Design Suite from National Instruments (NI). The suite is equipped with virtual instrumentation to allow the students to test and simulate their circuits in the program.

4.2 Proposed Changes

In the new curriculum changes are made so that the laboratory exercises can feature use of FPGA boards to implement some of the complex circuits. These changes only affect the mode of implementation of some laboratory exercises that are tough to implement on a bread board and easy to implement on the Basys 3 board. The peripherals on the Basys 3 board are more than enough for the entry level FPGA programming. Table 2 shows the new summary of laboratory exercises after modernization of the course. Two combinational circuit lab exercises and one sequential lab have been modified so that they can be implemented on the Basys 3 FPGA board.

4.2.1 Combinational labs

Labs 7 and 8 have been modified so that they can be implemented on the FPGA board. Lab 7 involves building a BCD decoder that takes input in the form of a BCD number through four of the switches on the FPGA and outputs the actual number on the seven-segment display of the FPGA board. Below is a picture of such an implementation as it would appear on Multisim. The blocks in the circuit represent the functional circuit for each LED of the seven-segment display. A detailed figure for each circuit is found on figure 14 and 15.

Table 2: Laboratory Curriculum for the modernized Digital Systems course

No	Lab Title	Learning Objective	Implementation
0	Instruments	To familiarize the students to the lab instruments and how to use them	
1	Logic gates and Boolean Algebra.	Test and observe the working of basic gates AND, OR, NAND, XOR, NOR, NOT.	On the breadboard using TTL chips and LED's.
2	De Morgan's Theorems	To verify De Morgan's Theorems of Boolean Algebra.	On the breadboard using TTL chips and LED's.
3	Combination Circuits	To simplify Boolean expressions to the simplest form and construct the simplest form and compare with the original form.	Simulate on Multisim using the Word generator and Logic analyzer
4	Universal property of NAND and NOR.	Verify the use of the universal gates as AND, NOT and OR gates.	On the breadboard using TTL chips and LED's.
5	Adders and Multiplexers	To implement an ADDER and a Data Selector and analyze its output.	Simulate on Multisim using the Word generator and Logic analyzer
6	Encoders and Decoders	To implement an Encoder and a Decoder circuit.	Simulate on Multisim using the Word generator and Logic analyzer
7	**Seven Segment Display**	**To implement a BCD decoder with the seven-segment display.**	**On the Multisim software and Basys 3 FPGA board.**
8	**Comparators**	**Implement a comparator circuit**	**On the Multisim software and Basys 3 FPGA board**
9	Look Ahead Carry Adders	Implement the Look ahead carry adder.	Simulate on Multisim using the Word generator and Logic analyzer
10	Arithmetic and Logic Unit	Implement a simple ALU.	Simulate on Multisim using the Word generator and Logic analyzer
11	Latches and Flipflops	Implement the flipflops	Simulate on Multisim using the Word generator and Logic analyzer
12	**Counters**	**Implement a BCD decade counter**	**On the Multisim software and Basys 3 FPGA board.**
13	**Project**	**To design an elevator motor**	**On the Multisim**

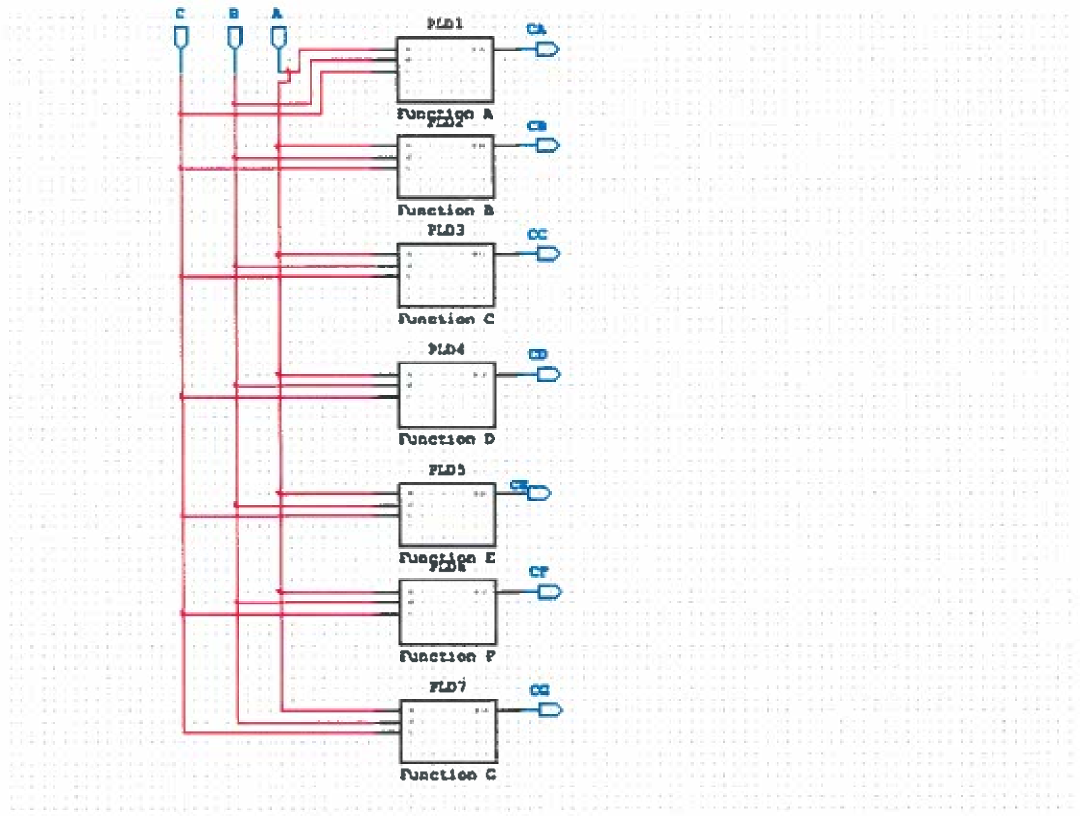


Figure 10: BCD Decoder with seven-segment display

Lab 8 involves implementing a comparator circuit, which takes in two binary numbers through its switches as inputs and lights up an appropriate LED to indicate either greater than, equal to or less than. An implementation of the circuit in Multisim is show in figure 15.

Although the students are still not familiar with Hardware Descriptive Languages (HDL), they get a chance to view how a typical HDL code would look like for the circuit they designed. Multisim produces a VHDL file that implements the circuit on the FPGA board. Figure 16 shows a VHDL snippet for the comparator circuit in figure 15.

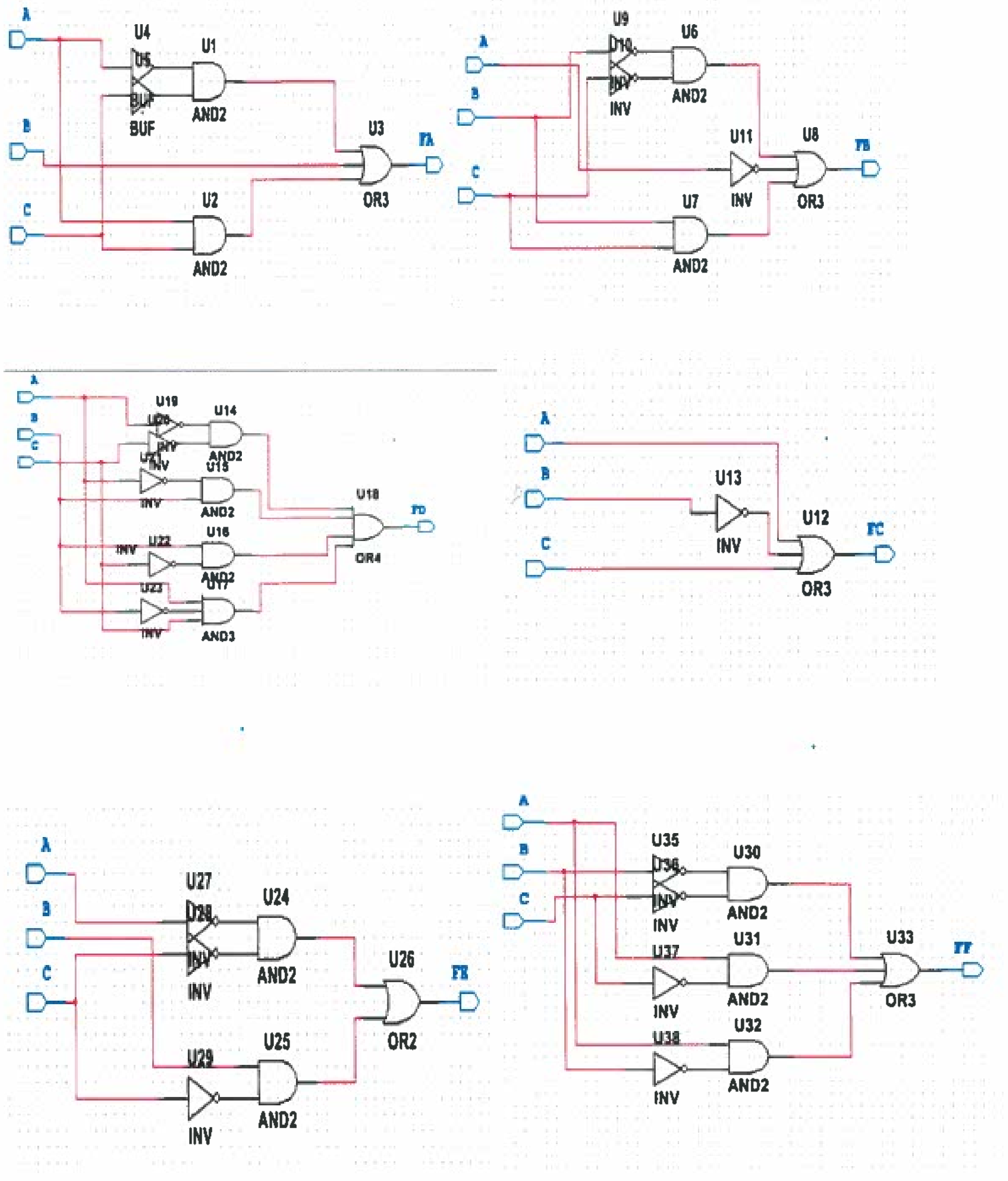


Figure 13: The figure represents six functional sub-circuits of the 7-segment display (a, b, c, d, e, f)

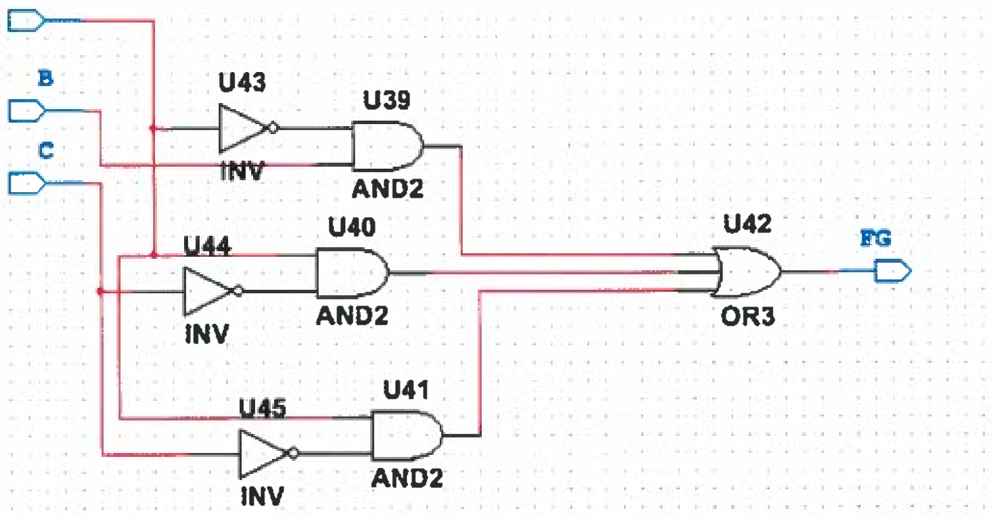


Figure 14: Function G of the seven-segment display

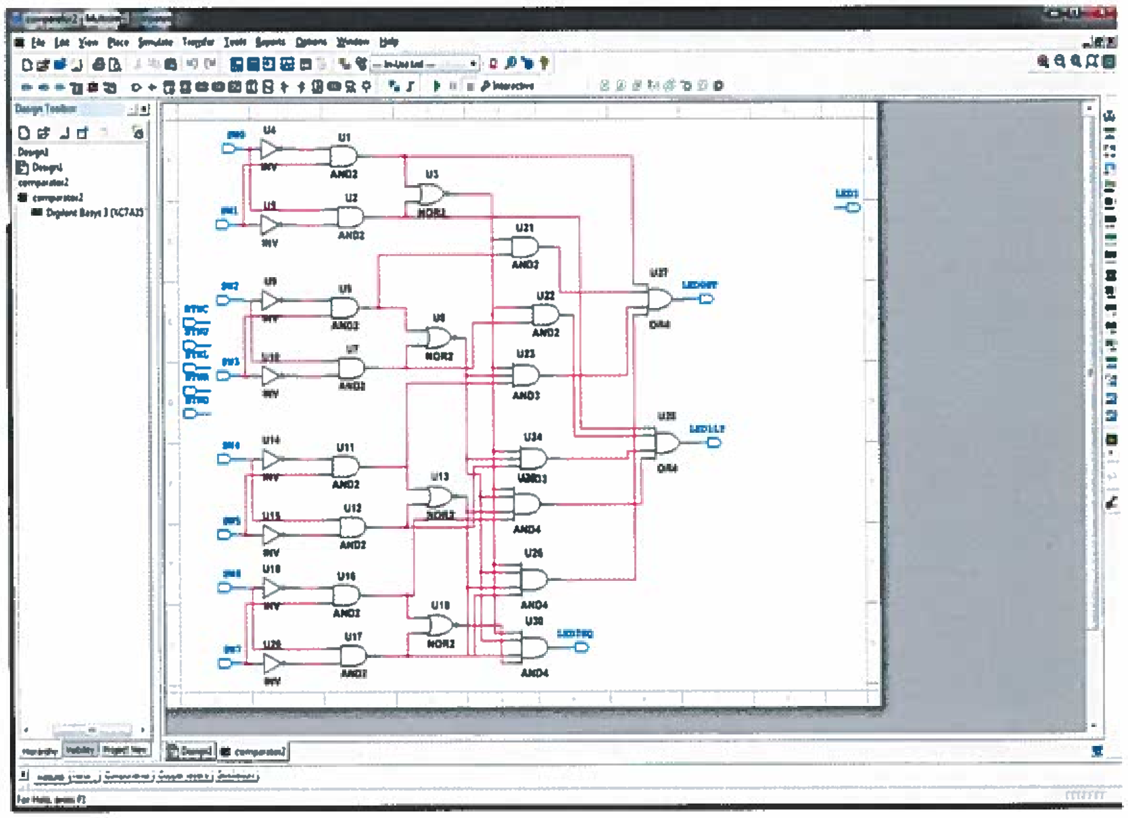
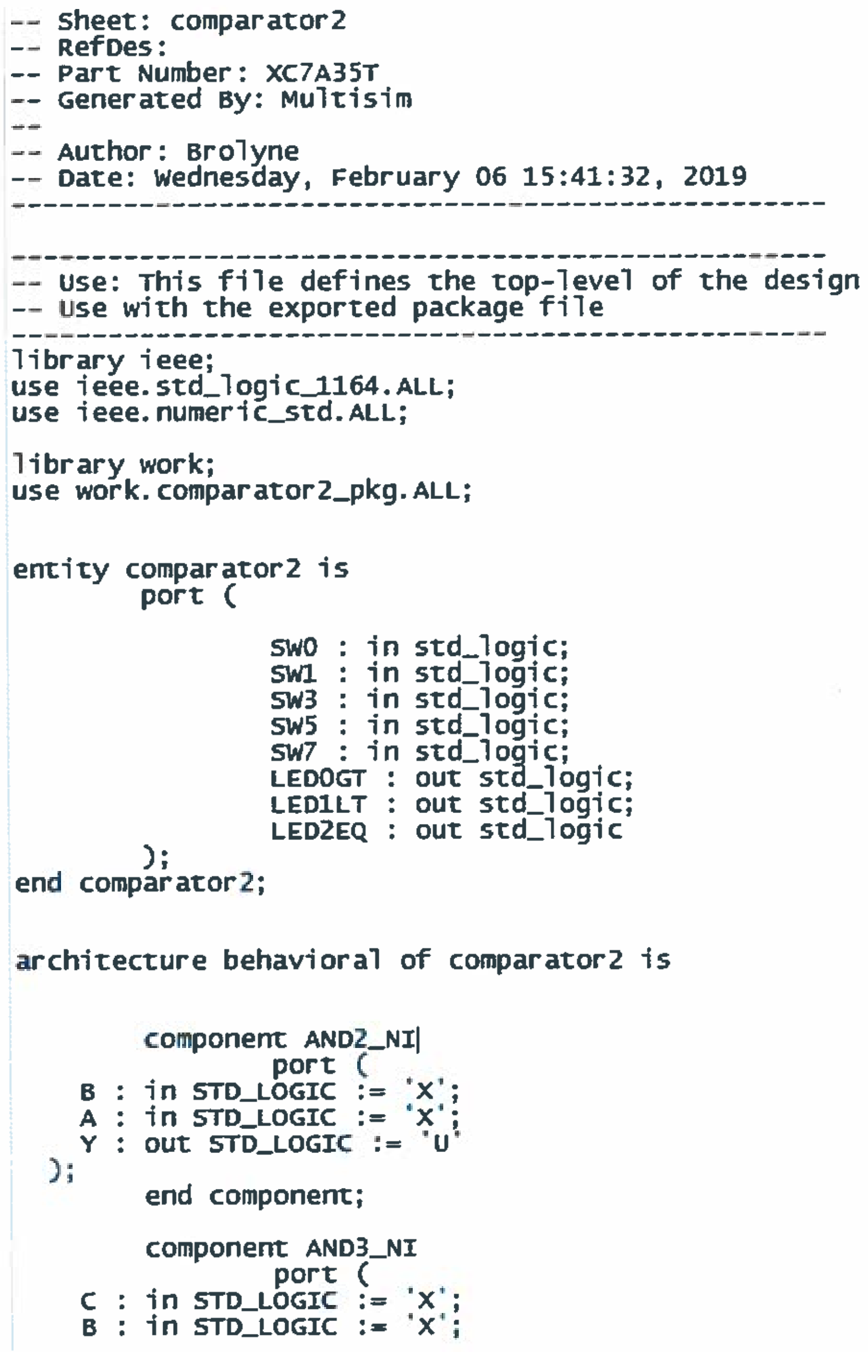


Figure 15: Comparator circuit as implemented on Multisim



```vhdl
-- Sheet: comparator2
-- RefDes:
-- Part Number: XC7A35T
-- Generated By: Multisim
--
-- Author: Brolyne
-- Date: Wednesday, February 06 15:41:32, 2019
----------------------------------------------------

----------------------------------------------------
-- Use: This file defines the top-level of the design
-- Use with the exported package file
----------------------------------------------------
library ieee;
use ieee.std_logic_1164.ALL;
use ieee.numeric_std.ALL;

library work;
use work.comparator2_pkg.ALL;


entity comparator2 is
        port (

                SW0 : in std_logic;
                SW1 : in std_logic;
                SW3 : in std_logic;
                SW5 : in std_logic;
                SW7 : in std_logic;
                LED0GT : out std_logic;
                LED1LT : out std_logic;
                LED2EQ : out std_logic
        );
end comparator2;


architecture behavioral of comparator2 is


        component AND2_NI
                port (
    B : in STD_LOGIC := 'X';
    A : in STD_LOGIC := 'X';
    Y : out STD_LOGIC := 'U'
  );
        end component;

        component AND3_NI
                port (
    C : in STD_LOGIC := 'X';
    B : in STD_LOGIC := 'X';
```

Figure 16: VHDL code for Comparator circuit

4.2.2 Sequential Circuit

The sequential circuit implemented on the FPGA Basys 3 board is the decade counter of lab 12. The counter uses the internal clock of the FPGA board to synchronize the flip-flops of the counter circuit. The seven-segment display of the FPGA board is used to display the count from the counter. An implementation of the counter on the Multisim is shown on figure 15.

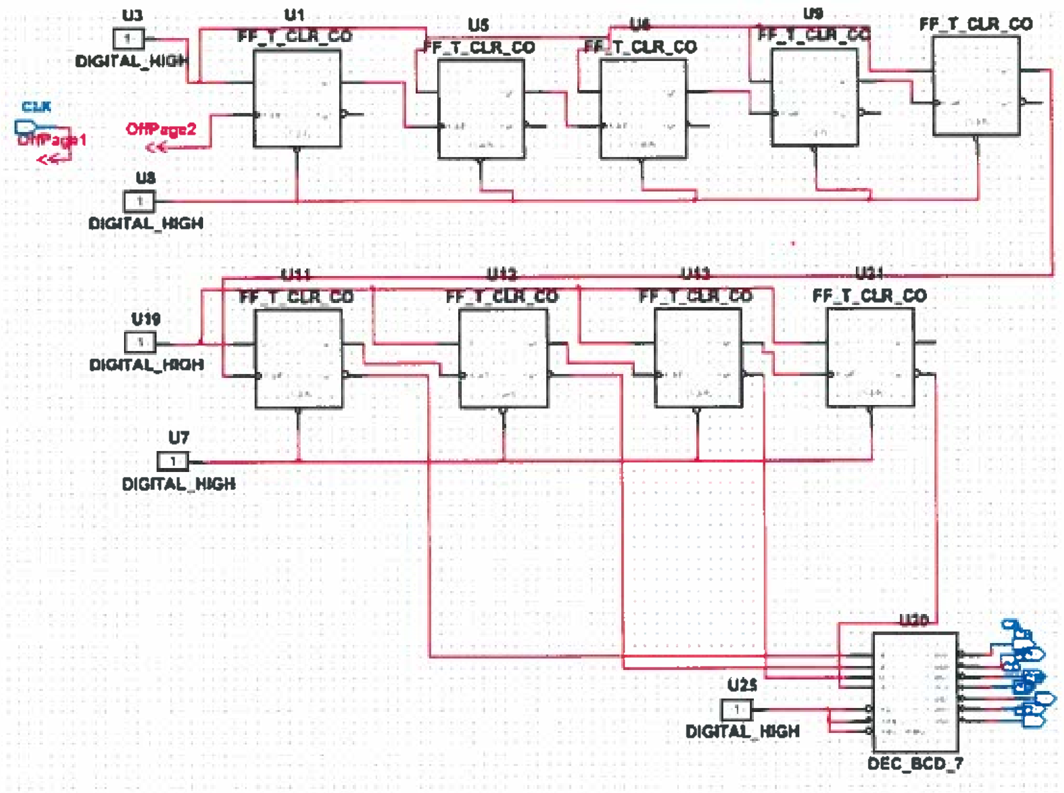


Figure 17: Counter implementation on Multisim

A major challenge in implementing a counter on the FPGA board and using the internal clock of the board to synchronize the flip flops in your counter is that the internal clock of the Basys 3 FPGA is that the clock frequency is very high 500MHz making it impossible to see the count on the seven-segment display. To solve this one must divide the frequency of the clock as many times as possible to bring it down to about 1KHz. To achieve this 25 T flip flop were used to bring the frequency down. Since this number of flip flops will not fit on to your counter design a separate sub circuit is built in your design with the 25 flipflops scaling down the frequency of the clock.

4.3 Project

At the end there is a project to be implemented that goes over the concepts covered during the course. The project is the design of a finite state machine in the form of a motor control for the lift of a building. The lift operates on a 3 storey building where it can go up and down. The lift stops when the sensors of the lift perfectly align with those at the door of a floor. The user should be able to choose the floor he desires to go to. This circuit is to be implemented on the FPGA board through Multisim. A picture showing the implementation of the control circuit is shown in figure 16.

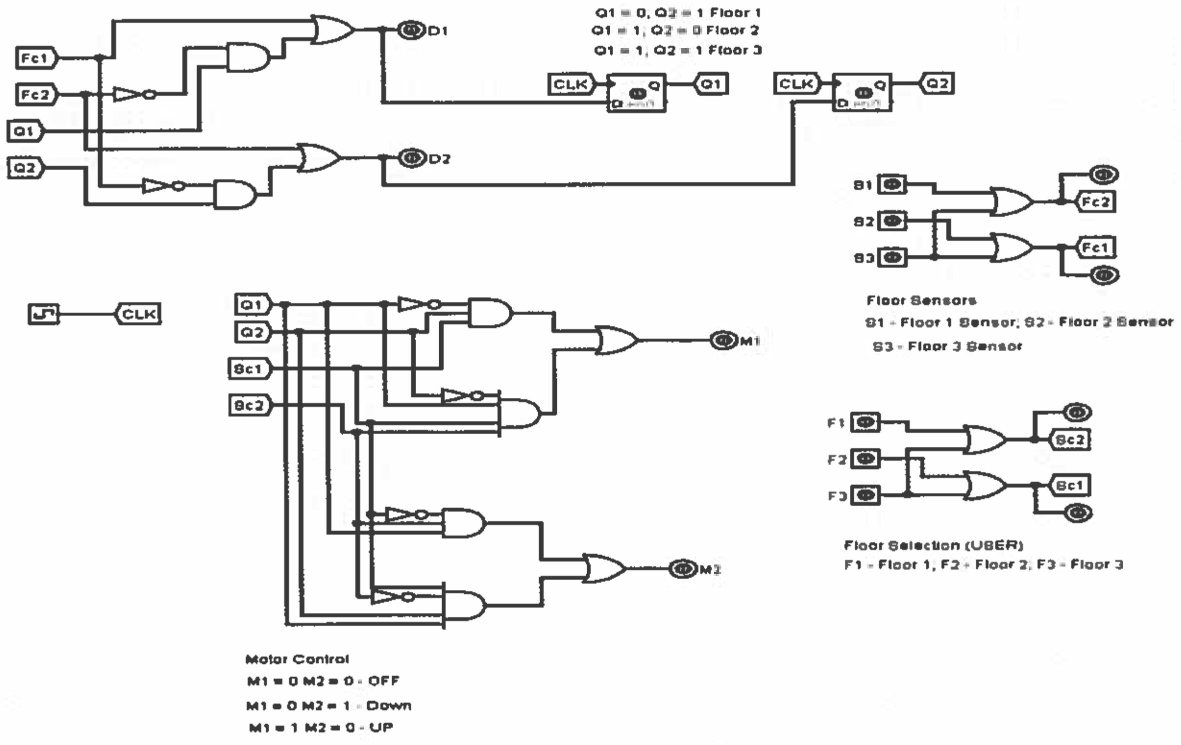


Figure 18: Motor control circuit for a Lift

CHAPTER 5

CONCLUSION

The intention is to make the Digital systems design course more exciting while at the same time introducing new concepts that the students will need in the future of their study in digital systems. Once the course is introduced, we intend to conduct a student survey and observe the level of satisfaction that the modernized course produces. A questionnaire will be filled by the students after taking the new labs and the data collected analyzed to gauge the students' reaction to the new approach.

The new curriculum will be introduced in the fall semester of 2019. Modified lab manuals will be available for the students use. A ten-minute tutorial guide is also available to guide the students how to install the required software and the setup for the use of the Basys-3 FPGA. The laboratory procedures have been developed and tested for; lab 7 binary to BCD converter, lab 8 comparator circuit and lab 12 a decade counter. Further work is underway to create a word generator that can be used to analyze digital circuits through the FPGA. This is meant to make analysis of truth tables for circuits easy.

References

1. H. A. Ochoa and M.V. Shirvaikar, "A Survey of Digital Systems Curriculum and Pedagogy in Electrical and Computer Engineering Programs." Proceedings of the 2018 of ASEE Gulf-Southwest Section Annual Conference, Austin, Texas, April 2018
2. E. A. Mayer, "Developing undergraduate FPGA curriculum using Altium Software and Hardware." American Society for Engineering Education", Pittsburg, Kansas 2012.
3. B. D. Carrol, S. N. Gieser, and D. Levine "A hierarchical project-based introduction to digital logic design course." 121st Annual Conference and Exposition. Indianapolis, Indiana. June 2014.
4. R. Rodriguez-Ponce, and J. Rodriguez-Resendiz, "Integrating VHDL into an undergraduate digital design course." IEEE International Conference on Teaching, Assessment and learning for Engineering. Kuta, Indonesia August 2013.
5. S. M. Loo, A. Planting and M. Murdock, "Introducing Field Programmable Gate Arrays into Sophomore Digital Circuits Course, "SEE Annual Conference & Exposition, Chicago, Illinois. June 2006.
6. Y. Zhao and S. Huang, "Improving Student Understanding of Digital Systems Design with VHDL via Inductive Instruction," ASEE Annual Conference & Exposition, Columbus, Ohio June 2017.
7. H. Shaalan, D. Kar and R. Bachnak, "Digital Systems Laboratory for Teaching and Research," ASEE Annual Conference, Salt Lake City, Utah April 2004.
8. G. Donzelini and D. Ponta," From Gates to FPGA: Learning Digital Design with Deeds." Interdisciplinary Engineering Design Education Conference, Genova, Italy June 2013.
9. S. Shayesteh, M. E. Rizkalla, L. Christopher, and Z. Ben Miled, "Attached Learning Model for First Digital System Design Course in ECE Program," ASEE Annual Conference & Exposition, New Orleans, Louisiana June 2016
10. M. Hassan, "Course Development in Digital Systems Targeting Reconfigurable Hardware," SEE Annual Conference & Exposition, Austin, Texas June 2009.
11. The NI FPGA from National Instruments Web-Site http://www.ni.com/fpga/, accessed January 30, 2019.
12. Digilent FPGA Web-Site https://store.digilentinc.com/digilent-adept-2-download-only/, accessed January 30, 2019.

Appendix

EENG 3302 Digital Systems

Lab 7 – Seven Segment Display

Objectives

Seven-segment displays are used with logic circuits that decode a binary coded decimal (BCD) number and activate the appropriate digits on the display.

After completion of this experiment, you will be able to use the ECG 3080 and the appropriate logic design for each segment to activate any digit.

Material Needed

7404 inverter

7408 AND gate

7432 OR gate

ECG 3080

Discussion

The segment decoding logic requires four binary coded decimal (BCD) inputs and seven outputs, one for each segment of the display as shown in the handout. The multiple-output truth table, shown below, is seven truth tables in one and could be separated into a separate table for each segment. A 1 in the segment output columns of the table indicates an activated segment.

Since the BCD code does not include the binary values 1010, 1011, 1100, 1101, 1110, 1111, these combinations will never appear on the inputs and can therefore be treated as don't care (X) conditions, as indicated on the truth table.

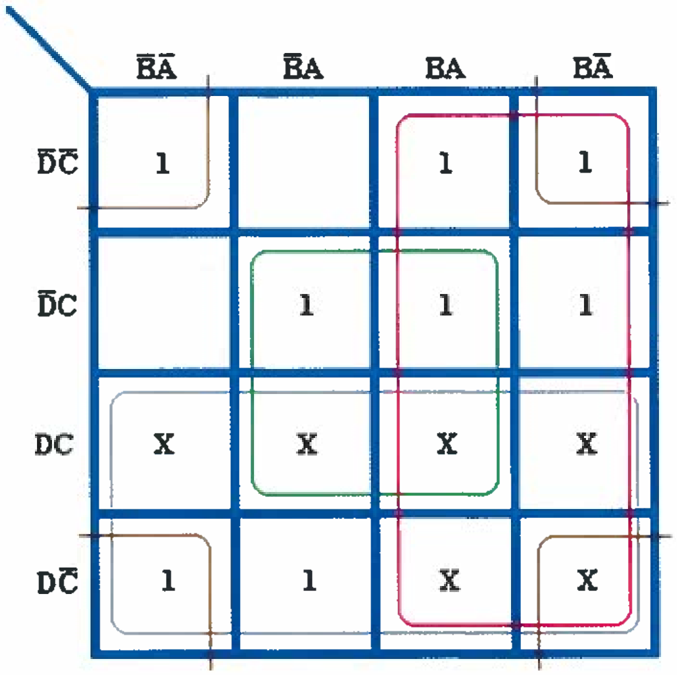
Decimal Digit	BCD Inputs				Segment Outputs						
	D	**C**	**B**	**A**	**a**	**b**	**c**	**d**	**e**	**f**	**g**
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	1	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	1	0	1	1
Decimal Digit	**BCD Inputs**				**Segment Outputs**						
	D	**C**	**B**	**A**	**a**	**b**	**c**	**d**	**e**	**f**	**g**
Invalid	1	0	1	0	1	1	1	1	1	1	1
	1	0	1	1	1	1	1	1	1	1	1
	1	1	0	0	1	1	1	1	1	1	1
	1	1	0	1	1	1	1	1	1	1	1
	1	1	1	0	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1

Boolean Expression for the Segment Logic:

The standard SOP expression for segment **a** from the truth table is:

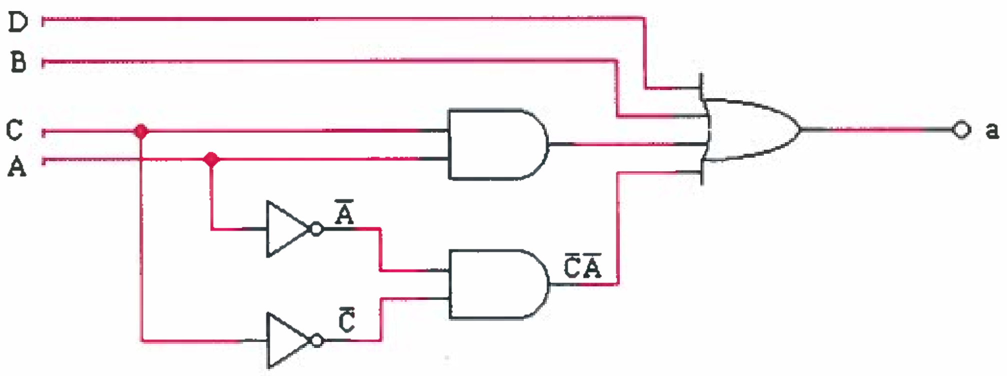
$$a = \bar{A}\bar{B}\bar{C}\bar{D} + AB\bar{C}\bar{D} + \bar{A}B\bar{C}\bar{D} + A\bar{B}C\bar{D} + ABC\bar{D} + \bar{A}BC\bar{D} + \bar{A}\bar{B}\bar{C}D + A\bar{B}\bar{C}D$$

The Karnaugh map for segment a is:



The minimum SOP expression is:

$$a = B + D + AC + \bar{A}\bar{C}$$



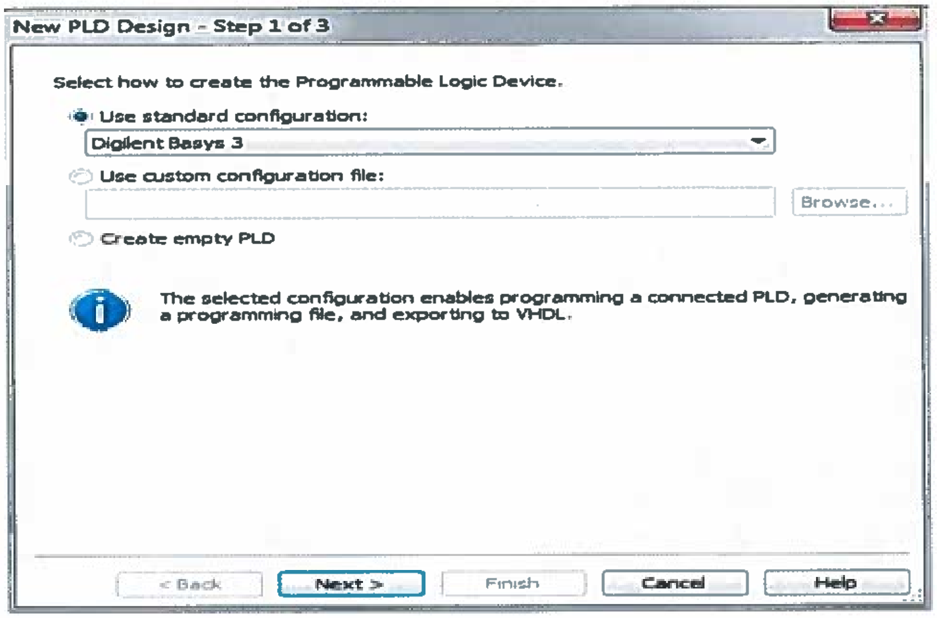
The minimum logic for each of the remaining six segments **b**, **c**, **d**, **e**, **f**, and **g**, can be obtained with a similar approach.

Procedure

Design the logic for segments a, b, c, d, e, f, and g. Include necessary Boolean algebra and expressions, truth tables, Karnaugh maps, and logic diagrams in your report.

Open Multisim Design Suite and under the files choose new then select new PLD Design and click the create radio button at the bottom of the pop-up window.

In the new window that pop-up under *use standard configuration* drop list choose **Digilent Basys 3** as shown below. Then choose next.



New PLD Design - Step 1 of 3

Select how to create the Programmable Logic Device.

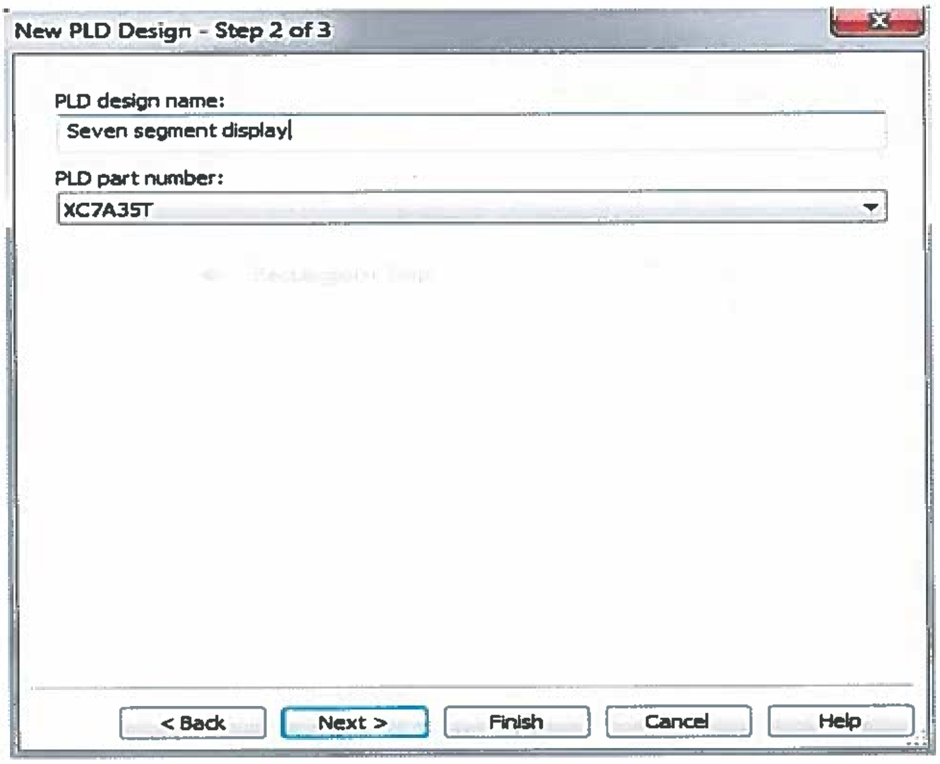
(●) Use standard configuration: Digilent Basys 3

( ) Use custom configuration file:

( ) Create empty PLD

The selected configuration enables programming a connected PLD, generating a programming file, and exporting to VHDL.

In the next window name your PLD design "Seven segment display" and click next as shown below.

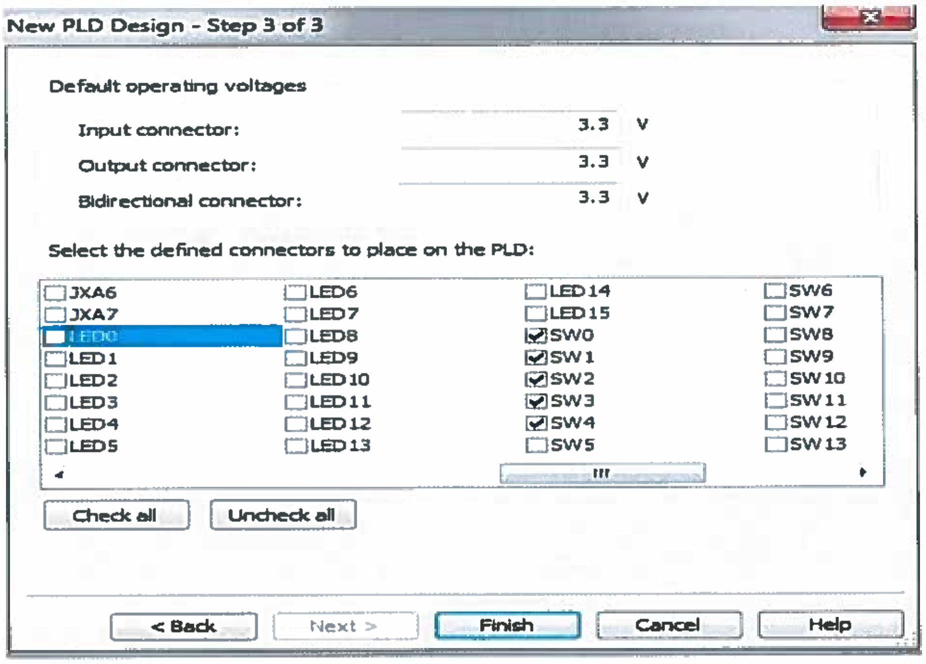


New PLD Design - Step 2 of 3

PLD design name: Seven segment display

PLD part number: XC7A35T

Choose the Basys 3 peripherals you need for the project in the next window. Make sure you tick CA, CB, CC, CD, CE, CF, CG which represent the seven segments of the display. Also choose three switches SW0, SW1, SW2 which represent switches on your board. Then choose finish. The figure below shows this window.



New PLD Design - Step 3 of 3

Default operating voltages

Input connector: 3.3 V

Output connector: 3.3 V

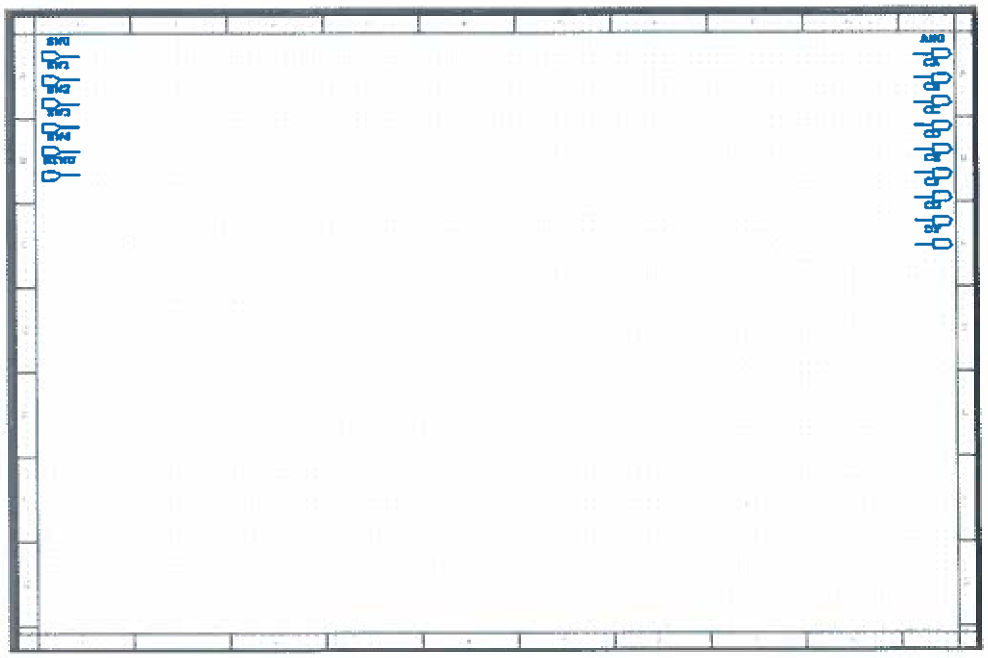
Bidirectional connector: 3.3 V

Select the defined connectors to place on the PLD:

☐ JXA6	☐ LED6	☐ LED14	☐ SW6
☐ JXA7	☐ LED7	☐ LED15	☐ SW7
☐ LED0	☐ LED8	✓ SW0	☐ SW8
☐ LED1	☐ LED9	✓ SW1	☐ SW9
☐ LED2	☐ LED10	✓ SW2	☐ SW10
☐ LED3	☐ LED11	✓ SW3	☐ SW11
☐ LED4	☐ LED12	✓ SW4	☐ SW12
☐ LED5	☐ LED13	☐ SW5	☐ SW13

A blank design page appears next with the peripherals you chose for your project. Now implement your design on this page attaching the appropriate peripherals and the logic gates necessary.

NOTE: Your design may not fit in the design space provided, you may explore the option of using sub-circuits to represent each of the segments then join then them to form your final circuit.



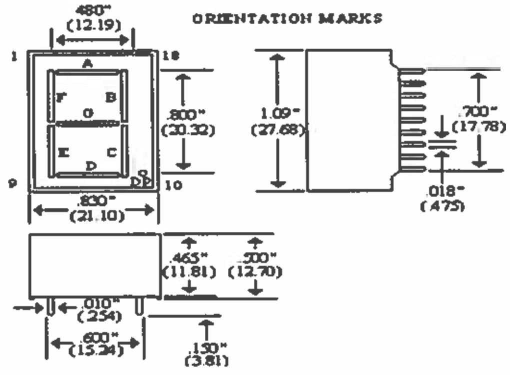
Once your design is complete attach your FPGA Basys 3 to your computer and program it by transferring your design to the FPGA board. Refer to the tutorial on how to program the Basys 3 board available in the modules.

Open the vhdl file generated for the circuit and observe how the different components interact with each other.

Exercises

Test your design on the FPGA board by entering binary digits through the switches and observe the value displayed on the 7-segment display.

1. No Pin	10. Decimal Point
2. Segment A	11. Segment D
3. Segment F	12. Common Anode
4. Common Anode	13. Segment C
5. Segment E	14. Segment G
6. Common Anode	15. Segment B
7. No Connection	16. No Pin
8. No Pin	17. Common Anode
9. No Pin	18. No Pin



EENG 3302 Digital Systems
Lab 8 – Comparators

Objectives
A comparator is a device that compares the magnitudes of two digital quantities and produces an output indicating the relationship of the quantities.

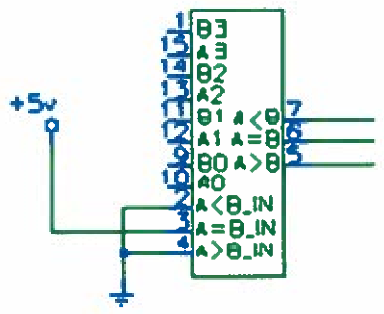
After completion of this experiment, you will be able to use the 7485 magnitude comparator.

Material Needed

- 7404 inverter
- 7408 AND gate
- 7432 OR gate
- 7485 magnitude comparator

Discussion

The purpose of a comparator is to compare two quantities and to indicate whether or not they are equal to each other or which quantity is larger. A block diagram for a magnitude comparator designed to compare two 4-bit binary numbers is given below.



4 bit magnitude comparator

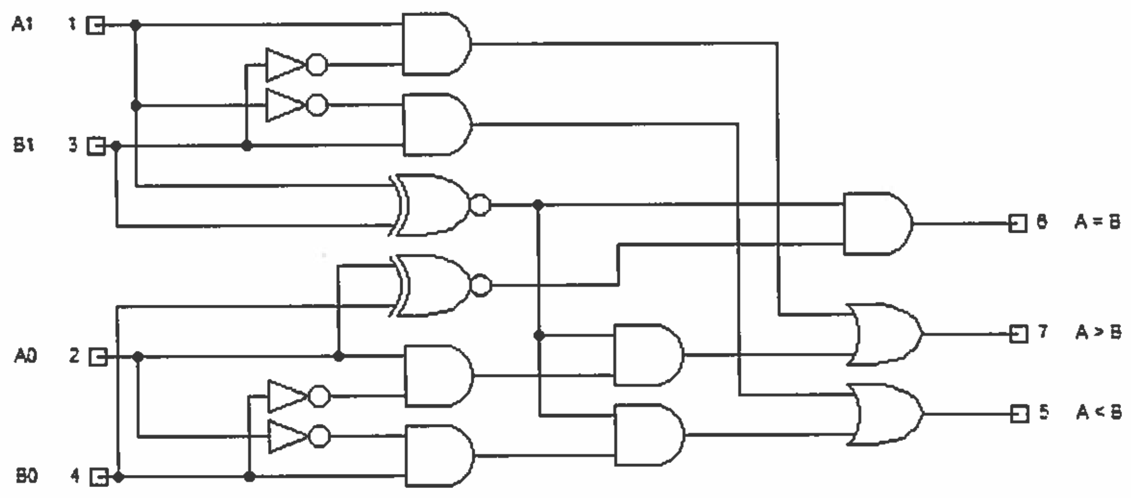
Three outputs are provided: A < B, A = B, A > B. It is important to notice that when two numbers are unequal, you determine the greater number by examining the most significant unequal bit; for example, the number 11010 is greater than 11001. The internal logic of the magnitude comparator uses this order to make the correct comparison.

Notice that the 7485-magnitude comparator has three cascading inputs that permit the comparators to be cascaded so that any number of bits can be compared. Cascading is done by connecting the outputs of a less significant comparator to the cascading inputs of a more significant comparator.

When there is no less significant comparator, the cascade inputs are connected as shown in the figure above. Notice that the active-HIGH S = B input is tied HIGH, and the activeHIGH A < B and A > B inputs are tied LOW. The reasons for these connections are as follows:

When only one 4-bit comparator is used, it is considered the least significant comparator for purposes of cascading. Logically, it should not be affected by the absence of a less significant comparator. It follows that the A = B cascade input should be active-HIGH, because the absent four bits of A could not be considered either more or less than the absent four bits of B. The cascaded inputs are not even examined unless the two 4-bit numbers being compared are equal. In this case, the only condition of the cascaded inputs that will not disturb this equality is that the cascaded input A = B is tied HIGH and the A < Band A > B inputs are tied LOW.

An example 2-bit comparator circuit is shown below.



Procedure

Design a four-bit magnitude comparator and implement it in Multisim following the design steps in the previous lab to achieve your PLD design.

Choose 8 switches SW0, SW1, SW2, SW3, SW4, SW5, SW6, SW7 where the first four switches represent one of the input number and the other four represent the second input.

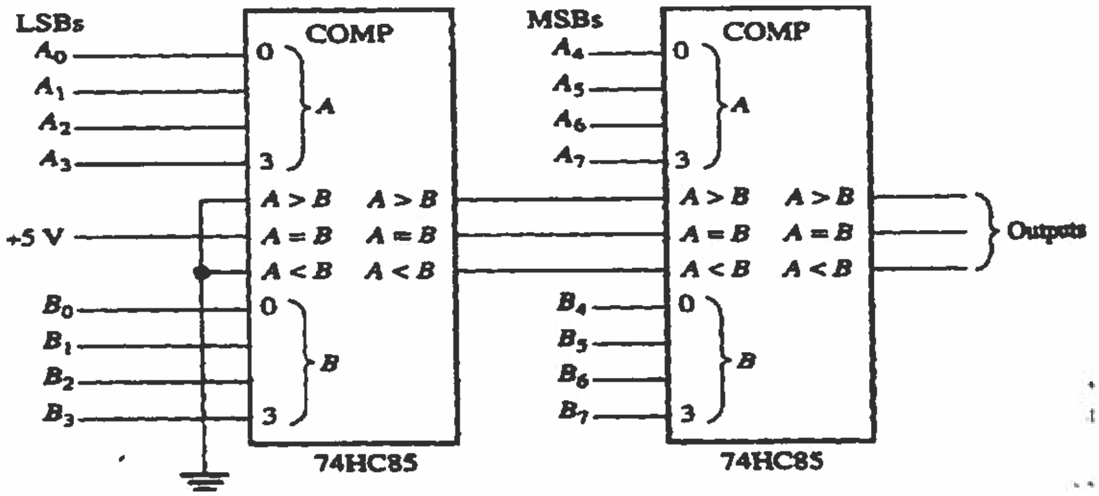
Let SW0 be the LSB for the first number and SW3 be the MSB for the same number also let SW4 be the LSB and SW7 be the MSB for the other number.

Also choose three LEDs LED0, LED1, LED2 where the LEDs represent the result of the comparison of the two numbers at the input.

Let LED0 represent the less than condition, LED1 represent the equal to condition and LED2 represent greater than condition. Program the Basys 3 board by transferring your design to the board as shown on the tutorial on how to program the Basys 3 FPGA board using Multisim.

Exercises

1. Using your design test the numbers below with each put on either inputs of your FPGA and record your results. Confirm if your design works perfectly
    a. A = 0001, B = 0000
    b. A = 0010, B = 0011
    c. A = 1000, B = 0111
    d. A = 0100, B = 0011
    e. A = 1001, B = 1001



2. Install two 7485 magnitude comparators as shown in the figure above to create a 8bit comparator and test with the numbers below:
    a. A = 00100100, B = 00101000
    b. A = 11001101, B = 10101100
    c. A = 00100110, B = 00100110
    d. A = 11110000, B = 11100001
    e. A = 00110100, B = 11001011

EENG 3302 Digital Systems

Lab 12 – Counters

Objectives

Counters, which consist of flip-flops and gates, are used to count pulses in digital systems. They can also be used as frequency dividers. The term asynchronous refers to events that do not have a fixed time relationship with each other and, generally, do not occur at the same time. An asynchronous counter is one in which the flip-flops within the counter do not change states at exactly the same time because they do not have a common clock pulse. All clock inputs of the flip-flops in a synchronous counter are connected to a common clock. As a result, the outputs of the flip-flops change only after the leading or trailing edge of the clock occurs.

After completion of this experiment, you will be able to build asynchronous and synchronous counters with flip-flops and gates.

Material Needed

- Your computer with Multisim installed
- Digilent Basys 3 FPGA Board

Discussion

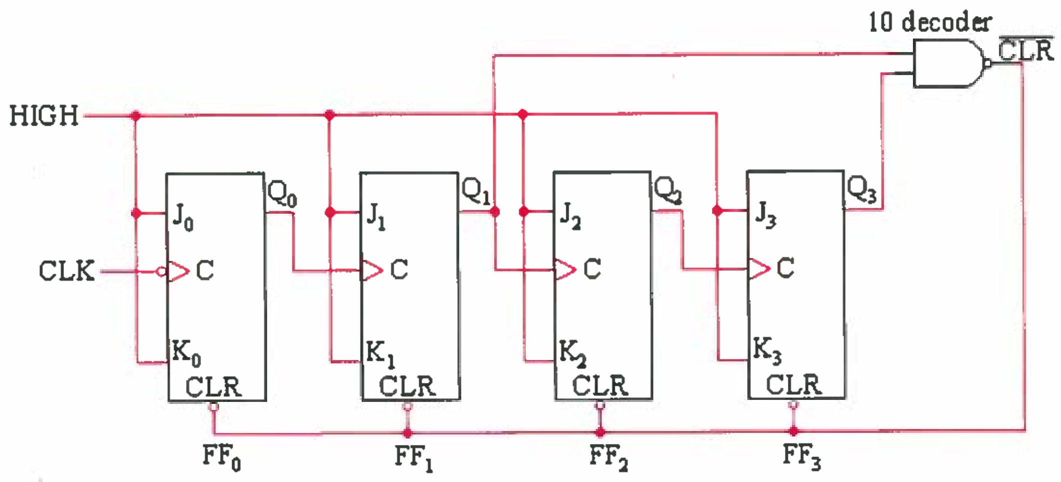
In a synchronous counter, the outputs of the flip-flops change simultaneously because they are triggered by a common clock. In the asynchronous or ripple counter, each flip-flop is triggered from the outputs of the preceding flip-flop, and the output of the last flip-flop cannot change until all of the preceding flip-flops have operated. Thus, propagation delays accumulate, thereby causing unreliable decoding in the ripple counter. These delays can cause spurious spikes, called glitches. In a synchronous counter, the total propagation time is limited to a single flip-flop, and therefore these problems are reduced.

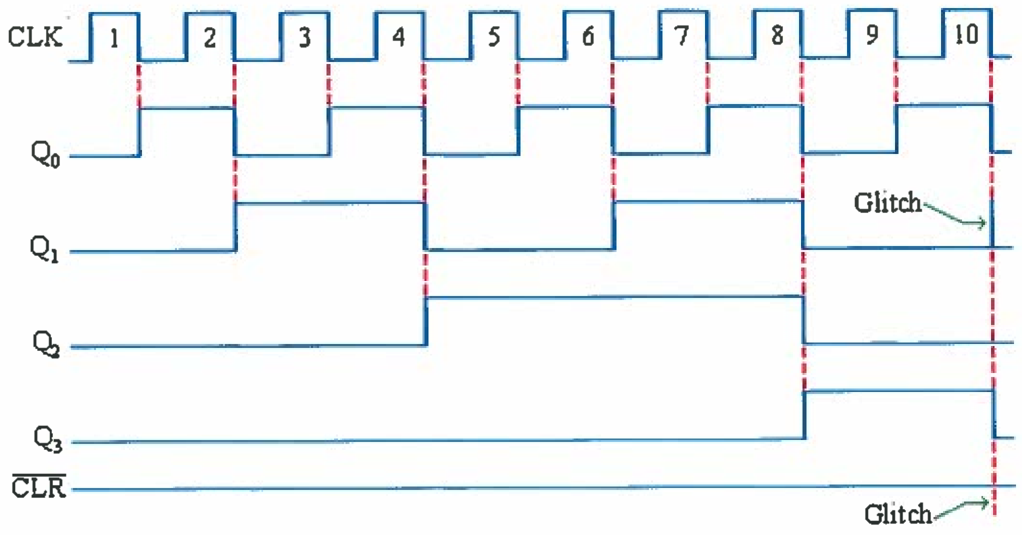
Counters with ten states in their sequence (modulus-10) are called decade counters. A decade counter with a count sequence of zero (0000) through nine (1001) is a BCD decade counter because its ten-state sequence is the BCD code.

To obtain a truncated sequence, it is necessary to force the counter to recycle before going through all of its normal states. The BCD decade counter must recycle back to the 0000 state after the 1001 state.

Notice in the figure below that only $Q_1$and $Q_3$are connected to the NAND gate inputs. This arrangement is an example of partial decoding, in which two unique states (Q_1= 1) and (Q_3 = 1) are sufficient to decode the count of ten, because none of the other states (0-

9) have both Q_1 and Q_3 HIGH at the same time. When the counter goes into count ten (1010), the decoding gate output goes LOW and asynchronous resets all the flip-flops.





An asynchronously clocked decade counter with asynchronous recycling

Refer to the figure below to follow the explanation of the synchronous counter. First, FF_0 (Q_0) toggles on each clock pulse, so the logic equation for its $J_0$and K_0 input is

$J_0=K_0=1$

This implemented by connecting J_0 and K_0 to a constant HIGH level. The FF_1 (Q_1) changes on the next clock pulse each time $Q_0=1$and $Q_3=0$, so the logic equation for the J_1 and K_1input is

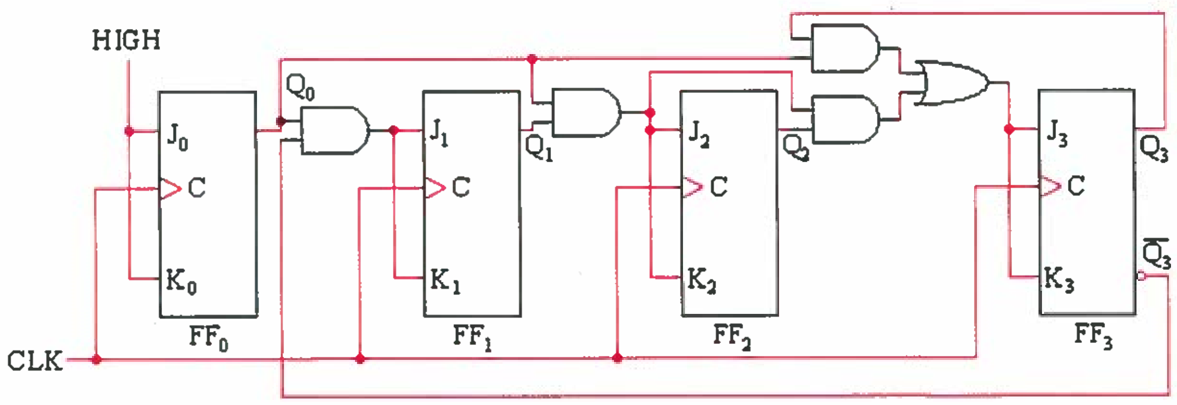
$$J_1 = K_1 = Q_0\,\bar{Q}_3$$

FF2 (Q_2) changes on the next clock pulse each time both $Q_0=1$ and $Q_1=1$.

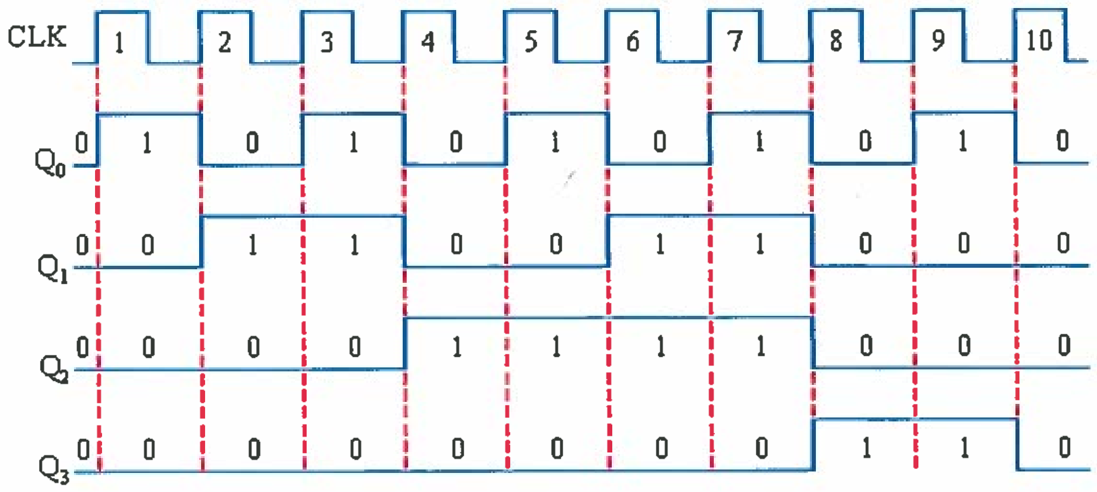
$$J_2 = K_2 = Q_0Q_1$$

FF3 (Q_3) changes to the opposite state on the next clock pulse each time $Q_0=1$, $Q_1=1$, and $Q_2=1$ or when $Q_0=1$ and $Q_3=1$.

$$J_3 = K_3 = Q_0Q_1Q_2 + Q_0Q_3$$



A synchronous BCD decade counter



Timing diagram for the BCD decade counter
(Q_0 is the LSB)

Procedure

Design the synchronous decade counter using a state transition diagram, state table, and Karnaugh maps.

Implement you design on Multisim PLD using the steps from the tutorial on how to program the Basys 3 FPGA board using Multisim.

When choosing your peripherals make sure you select the CLK which is the internal clock of the Basys 3 board and the seven segments CA, CB, CC, CD, CE, CF and CG to display your counter on the seven-segment display.

Use the in-built decoder circuit found under decoders in Multisim to decode your counter and display them on the seven-segment display. **(DEC_BCD_7)**

Note: The internal clock of the FPGA is about 500MHz and at such a frequency it is not possible to observe the count on the 7-segment display. To remedy this use frequency dividers to divide the clock frequency of the FPGA board down to about 1000Hz.

Tip 1: The frequency divider circuit may be too large thus not fitting on your single page design use multipage design so that you can have the frequency dividers on one page and your counter and decoder on another.

Tip 2: To use multipage design left click anywhere on your schematic and under the *place on schematic* menu find *Multipage.*

Exercise

Test your circuit on the FPGA board and observe if it counts from 0 to 9 and then resets to 0.

Modify your circuit to be an UP/DOWN counter between 0 and 9.