ECE 368 Digital Design - Spring 2016

Laboratory #2: (100pts)

Lab date: February 12th,2016

Lab Report Due: Friday February 19th,2016

Overview and Objectives:

In this laboratory assignment, you will be using the Xilinx ISE Design tools aimed at the Digilent Nexys 2 development board to become familiar with the VHDL design code entry, verification and simulation. You will be able to understand the principles of I/O systems and learn how to build your own interface with the various I/O's on the board. You will also examine the use of test bench's for design verification and develop and demonstrate a hardware text procedure.

Objectives for this lab:

- An introduction to understand VHDL code
- Learn the process of building and flashing on a Nexys 2 dev board
- Understand basic input/output (I/O)
- Understand debouncing concepts
- Understand VHDL operations that will help build a foundation
- Learn how to create a test bench to test a device
- Able to read and understand RTL diagrams
- Able to assemble all the concepts together

Lab sections:

1. Introduction with a counter
2. Applying button debouncing concepts
3. More in depth with a 7-Segment Display
4. Learning how to use a test bench with a ALU
5. Assembling all you learn together into a ALU with input

1. Introduction with a counter:

The counter described in this section of the lab can be used to create a clock divider in upcoming labs. The counter project contains six files, you don't have to worry about the contents of them for now. Right now you will be focusing on how to build this project first and then we will explain how it works.

Creating New Project in ISE Design Suite

You will need the counter project code which can be found here or in the class directory. Now lets start ISE Project Navigator. When you open up Xilinx ISE project navigator, you will be greeted by the welcome screen. You need to create a new project, you can do this either from the welcome screen or **File ->new project**.

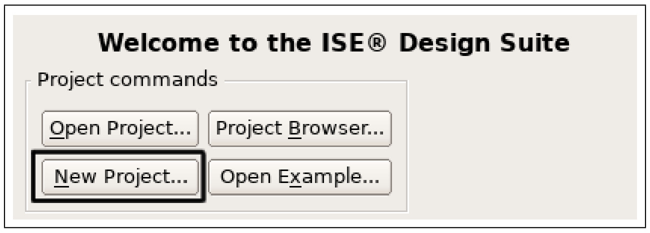
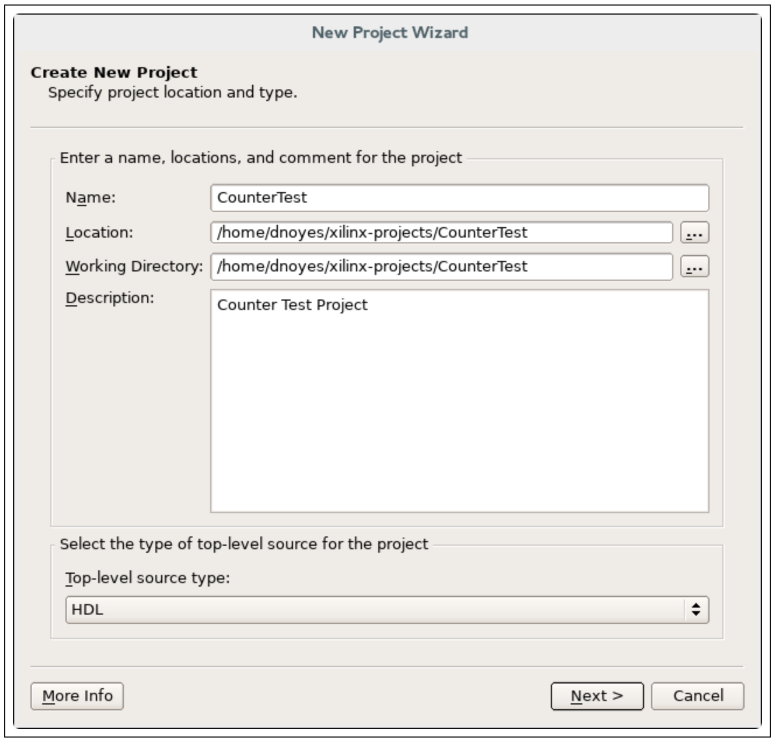


Figure 1: ISE® Design Suite welcome page

Steps for creating a new project in ISE:

1. Select **File ->new project** to open the New Project Wizard.
2. Enter a name for the project, for this project is will be **MealyTest**
3. Enter a location where the project will be located
4. Enter a description for the project
5. Select **HDL** in the top-level source type
6. Click next to advance to the project setting page
7. Enter the parameters for the project:
   Family : **Spartan3E**
   Device : **XC3S500E**
   Package : **FG320**
   Speed : **-4**
   Systhesis Tool : **XST (VHDL/Verilog)**
   Simulator : **ISim (VHDL/Verilog)**
   Preferred Language : **VHDL**
   Property Specification in Project File : **Store all values**
   Uncheck Manual Compile Order
   VHDL Source Analysis Standard : **VHDL-93**
   Uncheck Message Filtering
8. Double Check the Project Setting with the Setting Figure



Create New Project
Specify project location and type.

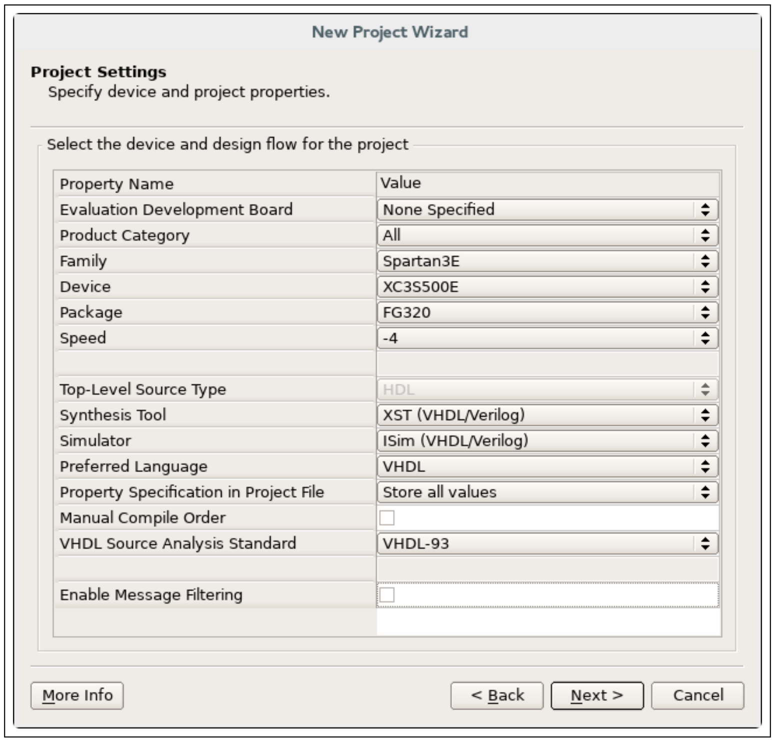
Enter a name, locations, and comment for the project

Name:	CounterTest
Location:	/home/dnoyes/xilinx-projects/CounterTest
Working Directory:	/home/dnoyes/xilinx-projects/CounterTest
Description:	Counter Test Project

Select the type of top-level source for the project

Top-level source type: HDL

Figure 2: New Project Wizard Create New Project page



Project Settings
Specify device and project properties.

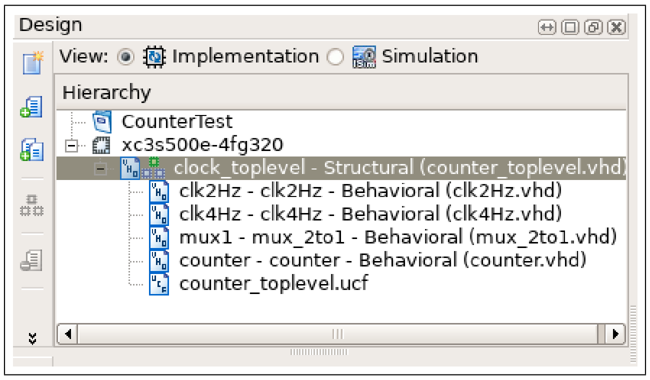
Select the device and design flow for the project

Property Name	Value
Evaluation Development Board	None Specified
Product Category	All
Family	Spartan3E
Device	XC3S500E
Package	FG320
Speed	-4
Top-Level Source Type	HDL
Synthesis Tool	XST (VHDL/Verilog)
Simulator	ISim (VHDL/Verilog)
Preferred Language	VHDL
Property Specification in Project File	Store all values
Manual Compile Order	☐
VHDL Source Analysis Standard	VHDL-93
Enable Message Filtering	☐

Figure 3: New Project Wizard Project Settings page

Adding VHDL Source to the Project

After you have created the project in ISE. You will now need to import the counter source files into the project. This can be done by going to **Project ->Add Source**. Select each of the six files in the Counter folder. After you imported all the files needed you will be presented with a clock_toplevel file as depicted in the next figure.

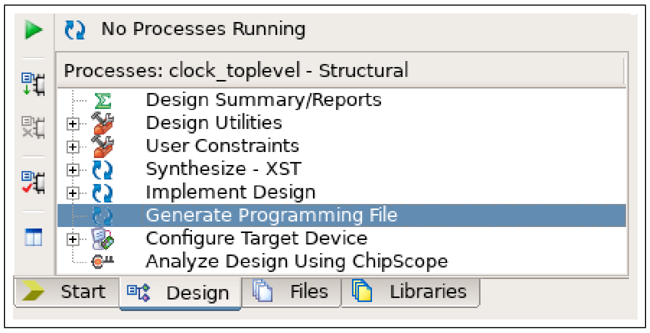


```
CounterTest
xc3s500e-4fg320
  clock_toplevel - Structural (counter_toplevel.vhd)
    clk2Hz - clk2Hz - Behavioral (clk2Hz.vhd)
    clk4Hz - clk4Hz - Behavioral (clk4Hz.vhd)
    mux1 - mux_2to1 - Behavioral (mux_2to1.vhd)
    counter - counter - Behavioral (counter.vhd)
    counter_toplevel.ucf
```

Figure 4: Clock_toplevel project design

Building the counter project

Now since you have a fully setup project. we are now going to build it. When you click on the "clock_toplevel" it will give you multiple options on the bottom on design tab. For building the project, you must run the "Generate Programming File". This can take a few minutes for ISE to process the VHDL files and create a programming file. During this process is where you will encounter various warnings and errors that may occur while building the project.



```
Design Summary/Reports
Design Utilities
User Constraints
Synthesize - XST
Implement Design
Generate Programming File
Configure Target Device
Analyze Design Using ChipScope
```

Figure 5: Clock_toplevel generate program file

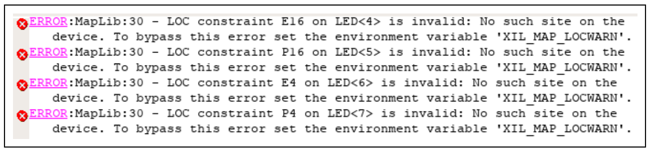
Building issue on a s3e500 board

While building the counter project, you may encounter an error while mapping the pins on a **s3e500** board. This can be resolved by changing the led pins in the user constraint file (UCF) file with the ones commented.

Flashing the counter project onto the board

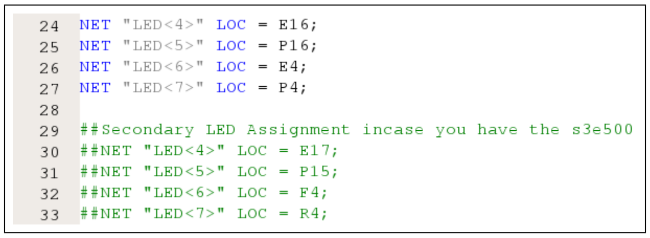
After build the project file you will see a green check mark appear on the "Generate Programming File". This means the project bit file was successfully created. Now it is the time to flash the Nexys 2 FPGA. In order to flash to the Nexys 2, you need to open up ISE iMPACT which can be found in **Tools ->iMPACT**.

Steps to Flash Nexyx 2 with ISE iMPACT



```
ERROR:MapLib:30 - LOC constraint E16 on LED<4> is invalid: No such site on the
   device. To bypass this error set the environment variable 'XIL_MAP_LOCWARN'.
ERROR:MapLib:30 - LOC constraint P16 on LED<5> is invalid: No such site on the
   device. To bypass this error set the environment variable 'XIL_MAP_LOCWARN'.
ERROR:MapLib:30 - LOC constraint E4 on LED<6> is invalid: No such site on the
   device. To bypass this error set the environment variable 'XIL_MAP_LOCWARN'.
ERROR:MapLib:30 - LOC constraint P4 on LED<7> is invalid: No such site on the
   device. To bypass this error set the environment variable 'XIL_MAP_LOCWARN'.
```

Figure 6: Counter pin mapping error



```
NET "LED<4>" LOC = E16;
NET "LED<5>" LOC = P16;
NET "LED<6>" LOC = E4;
NET "LED<7>" LOC = P4;

##Secondary LED Assignment incase you have the s3e500
##NET "LED<4>" LOC = E17;
##NET "LED<5>" LOC = P15;
##NET "LED<6>" LOC = F4;
##NET "LED<7>" LOC = R4;
```

Figure 7: Counter User Constraints File - Led

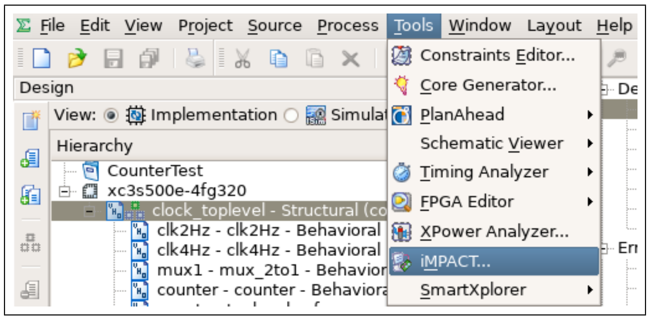

File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulat
Hierarchy
CounterTest
xc3s500e-4fg320
clock_toplevel - Structural (co
clk2Hz - clk2Hz - Behavioral
clk4Hz - clk4Hz - Behavioral
mux1 - mux_2to1 - Behavior
counter - counter - Behaviora
Constraints Editor...
Core Generator...
PlanAhead
Schematic Viewer
Timing Analyzer
FPGA Editor
XPower Analyzer...
iMPACT...
SmartXplorer

Figure 8: Xilinx iMPACT Location

1. Click on **Boundary Scan** under iMPACT Flows
2. Click on **Initialize Chain** on the tool bar or **File ->Initialize Chain** or Ctrl+i.
3. Wait until it ducceeded on the initialize chain
4. Click **Yes** on "Auto Assign Configuration Files".
5. Select the recently generated bit file in you projects folder
6. Click **No** on "Attach SPI or BPI PROM".
7. On the assign new configuration file select **Cancel All**.
8. Click **OK** on "Device Programming Properties".
9. Click the **xc3s500e** chip.
10. Double click **Program** under the iMPACT Processes
11. Wait a few seconds and it will say "Program Succeeded"
12. Check the Nexys 2 Dev board and if the led are incrementing you now have successfully flashed the Nexys 2.

Testing the Nexys 2 Counter

After you have successfully flashed the Nexys 2. Go play with the Counter and see what switch and button will affect the counter. There are three things you will need to look for on the Board. A reset, add/subtract mode, and change speed of the counter.

Counter Layout

As you might have played around with the device. You noticed that there were three inputs to this counter. The counter is comprised of a few entities, a clock, mux, counter, and a toplevel which binds them all together. You can see the register-transfer level(RTL) in the next figure which gives a block diagram of the project. You can generate you own RTL under **Design ->Synthesize - XST ->View RTL Schematic**

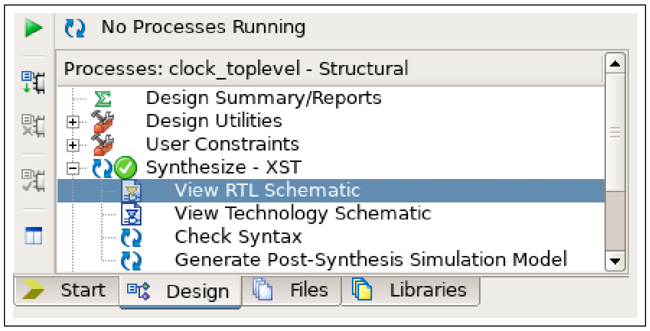


Figure 9: View RTL Schematic Location

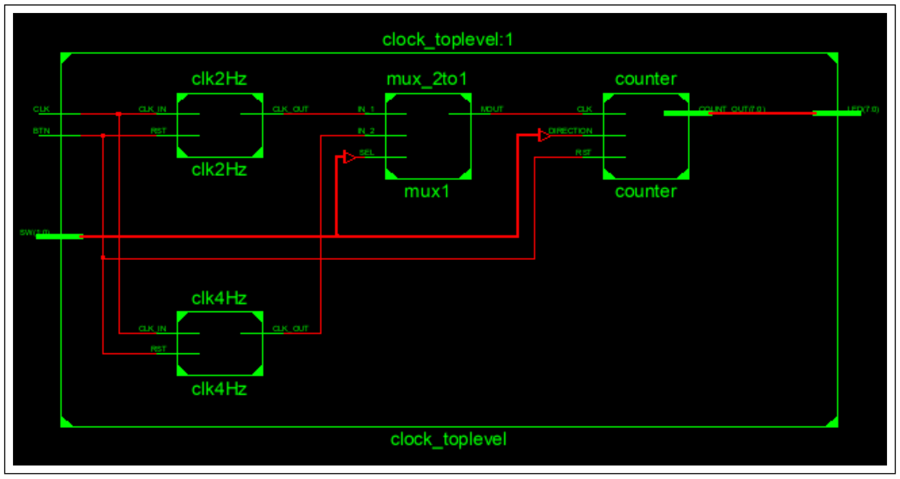


Figure 10: RTL Schematic for the counter

Breakdown of the Counter

While looking at the RTL schematic for the counter. You can see two clocks, each with a different name of ”clk2Hz” and ”clk4Hz”. Both are design to toggle after a certain number of cycles, for this one the input to each clock is 50 MHz. Both clock lines will then go through a mux 2 to 1 with a select switch to choose which clock to output. The output of the mux will then go into the clock input of the counter. The counter will add or subtract one from its current count signal. The counter toplevel connects all the components together and the **counter_topleve.ucf** will till the toplevel ports to the pins of the FPGA. Look at each file in the project and study how they work.

Counter Questions

While going through each segment you will be asked a few questions. These question will ask you about various parts of the lab besides these questions you will also need to demonstrate your work and knowledge to the TA or Professor. Don't forget to write down your answers and put them in your lab report.

1. Demonstrate a working counter.
   Reset the counter value.
   Change the counter speed.
   Change the counter to subtract instead of adding to the count.
2. Double the clock rate in the clk4Hz file. What does it do?
3. What action of the clock causes the counter to add or subtract(Rising Action? or Falling Action?).
4. What do you think could be different in this section of the lab?

2. Applying button debouncing concepts:

On the Nexys 2 board, there are four mechanical push buttons. These push buttons can be very noisy when you click them. With this in mine, you will need to debounce the push button signal. In this section of the lab you will be learning the concepts of debouncing a signal and how to implement it in your design.

Concepts of debouncing a signal

There are a variety of methods to debounce a signal. The method you will learn uses three flip flops, a xor gate, counter, and a bit compare. The input signal will go through one flip flop and then through another flip flop and will update it state based on the clock line. This will allow us to have two states of the signal. Each state will then go though an XOR gate to monitor the change in state. When there are two different states the xor will output a **1** to reset the counter. The counter will increment when no change occurs and the most significant bit will be used to update the last flip flop state. The last flip flop will output the debounced signal.

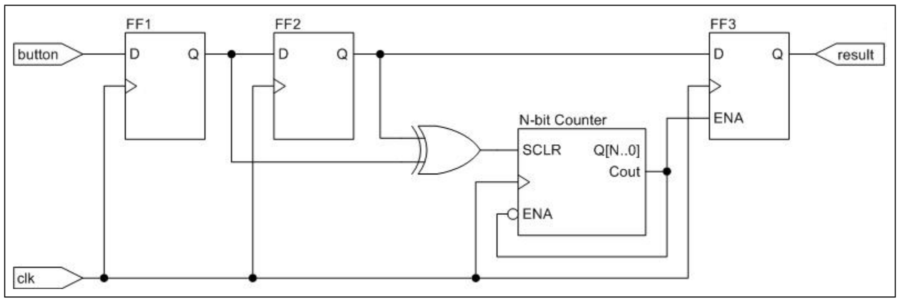


Figure 11: Debouncing circuit example(Source : eewiki)

Applying the concepts into VHDL

With learning the concepts of debouncing a signal. Next we need to apply the concepts into VHDL. First you will be needing signals to maintain the values of the first two flip flops, the XOR output and a location for the counter value. You will need to create a process routine to monitor the clock state. In the process, the flip flops will update their state when there is a **CLK'event** and with a value of **1**. This can be translated into a rising action on the clock line. Next you will need to check the flip flop states and reset the counter if both flip flops have different states. The output will happen if the most significant bit will turn **1**. It is recommend reviewing the debounce.vhd source and get a understanding of what is happening.

Building and Flashing the button debouncer project

Now is the time to build the button debouncer. First create a new project just like in the counter section. Add the source in the Button folder, there will be three files. Build, and flash it on the Nexys 2 dev board(refer to counter). Don't forget to check the VHDL code and see what is happening.

Button Debouncing Questions

1. Demonstrate to the TA or professor the button controller.
2. Why would you incorporate a enable in the button controller?

3. More in depth with a 7-Segment Display:

In this section, you will be learning how a 7-segment display can be incorporated in VHDL. While observing the Nexys 2 dev board, you will notice that there are four 7-segment displays. These displays can be very useful to use as a debug device since there can be 4 bytes of data displayed.

Reviewing 7-Segment Display

Just refresher each 7-segment display is tied to a common cathode while one anode per display. The cathodes are a ground sink while the anode supply the power. For the display you do not want to have the number flicker. So the display should oscillate around 60Hz. Since there are four displays you need to alternate between them. This will give us 240Hz for alternating between each number without flickering.

Applying the concepts into VHDL

In order to apply this in VHDL, you need to break down the 7-segment driver into reasonable chunks. First you will need a clock divider that will change a 50Mhz clock signal into 240Hz. Next you will need a 7-segment decoder, this can be done in a select statement in VHDL. The Last major part is the display driver to cycle between the 7-segment display and output the correct number. Look at the SevenSeg.vhd file to get a grasp of the concept. The top level connects the SevenSeg out to the board and helps provide values for the 7-segment display.

Building and Flashing the 7-segment Display project

Create a new project and add the source in the "SevenSegment" folder. Build the 7-segment display and flash it to the Nexys. Test the Seven Segment display by flipping the dip switches to control the first two displays.

7-Segment Display Questions

1. Adjust the VHDL code I/O to make the 7-seg display "FACE"
2. Demonstrate to the TA or professor the 7-seg display
3. What is the clock speed in the 7-segment display driver and why?

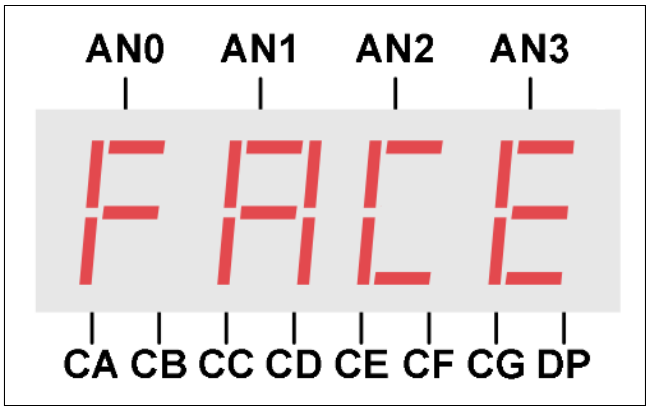


Figure 12: 7-segment Display output FACE

4. Learning how to use a test bench with a ALU:

In this section of the lab, you will be getting familiar with simulating VHDL in ISE. You are presented a complete source of an ALU. A ALU is a Arithmetic device which will output specific values based on operating codes(opcodes). For the ALU you will have two 8 bit input values(A,B) and a opcode. The output will have the result of the operation plus a condition code register.

Concepts of a ALU

In this Lab you will be using an 8-bit ALU. The ALU is a collection of operations. These operations can be add, subtract, AND, OR, Shift and much more. For the ALU you can see what operations it performs based on the opcodes in the table below.

OPCODE	operation	procedure	CCR
0000	ADDITION	RA <= RA + RB	NZVC
0001	SUBTRACTION	RA <= RA - RB	NZVC
0010	AND	RA <= RA & RB	NZVC
0011	OR	RA <= RA | RB	NZVC
0100	COMPARE	RA <= 0, NZ change	NZ
0101	ADD Immediate	RA <= RA + IMMED	NZVC
0110	AND Immediate	RA <= RA & IMMED	NZVC
0111	SHIFT LEFT	RA <= RA<<IMMED	VC
1000	SHIFT RIGHT	RA <= RA>>IMMED	
1001	LOAD WORD	RA <= ALU_MEM	
1010	STORE WORD	ALU_MEM <= RA	

Table 1: ALU OPCODE

Setting up a ALU in VHDL

Create a new project or use a existing and remove the files. Add all the source files in the ALU folder to the project. In this project you will be using the simulation utility in Xilinx. In order to run a simulation, you will need to create a Behavioral vhd file. This file will simulate the inputs to a VHDL design you are implementing. This will be able to aid in testing your code for and prone errors that may arise during your testing of the code.

Testing the ALU by simulation

In order to start simulation you must switch you view from Implementation to Simulation, this can be done by **Design ->View ->Simulation**. When you in simulation mode, select **ALU_tb_vhd**. Under the design tab you will see ISim Simulation and two operations. **Behavioral Check Syntax** and **Simulate Behavioral Model**. Run the Simulate Behavioral Model, there will be a graph popping up with the simulation results from the test bench file.

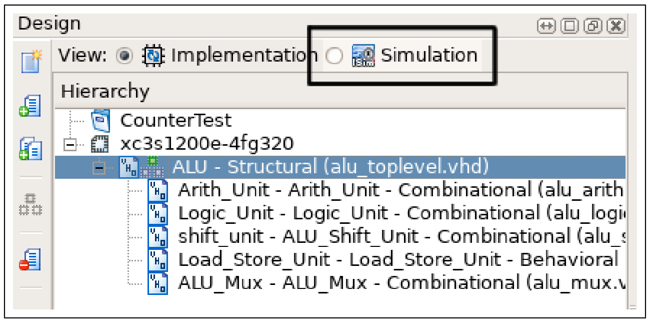

Design
View: [✓ Implementation] Simulation
Hierarchy
CounterTest
xc3s1200e-4fg320
ALU - Structural (alu_toplevel.vhd)
Arith_Unit - Arith_Unit - Combinational (alu_arith
Logic_Unit - Logic_Unit - Combinational (alu_logi
shift_unit - ALU_Shift_Unit - Combinational (alu_s
Load_Store_Unit - Load_Store_Unit - Behavioral
ALU_Mux - ALU_Mux - Combinational (alu_mux.v

Figure 13: Select Simulation Mode in ISE

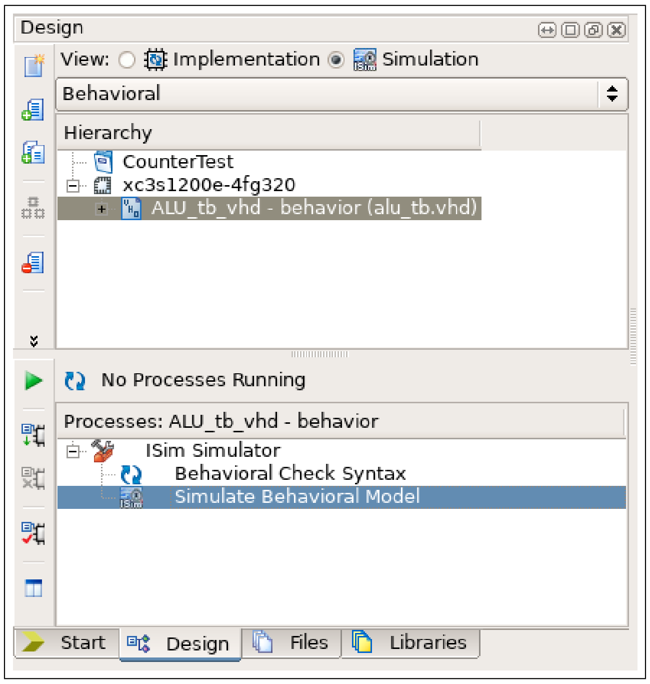


Figure 14: Start Simulation in ISE

ALU test bench Questions

1. Take a screen shot of the simulation from 50ns - 300ns
2. Change CCR value at line 110 from **1010** to **0000**. Rerun the simulation, what do you notice in the console and why does it happen?

5. Assembling all you learn together into a ALU with input:

With each section you learned various concepts and was given code to run. Now it is your turn to create some VHDL code. In the previous sections you learned how to interface with the Nexys board Dip switches, Led's, mechanical buttons, and the seven segment display. You also learned about a ALU. For this section you will be combining them together and creating a user input ALU. The method of input and output is all up to you.

Jumping into VHDL

Since this will be your first time coding VHDL for the project. You will have to devise a method for your ALU to interface with. First you need to create a block diagram of each device connected together. After that give more in depth design in a RTL. Build your design in VHDL with a reference of the top_level designs in the other sections of the lab.

Building, Flashing and testing a user input ALU project

This part will take you the most time to accomplish out of all the sections. You will have to debug you code and see how it runs on the Nexys 2 dev board. The building and flashing will always be the same methods in any lab. You will also be needing to create a Acceptance Test Plan(ATP) to validate that your project is working.

Final Questions

1. Demonstrate to the TA or professor the user input ALU.
2. What was the biggest challenge you overcame?
3. If you would rebuild this, how you change it?
4. What is your confidence level in VHDL and explain why.
5. Any ideas to improve the Lab?

Lab 2 Grade

Section	Description	Value	Score
Section 1		-10-	
1.1. Counter Demo	Demonstrate Counter to TA, Prof	5	
1.2. Counter Questions	Answer Questions about the Counter	5	
Section 2		-10-	
2.1. Button Demo	Demonstrate Button to TA, Prof	5	
2.2. Button Questions	Answer Questions about the Button Debouncer	5	
Section 3		-10-	
3.1. 7-Seg Demo	Demonstrate 7-Seg with display of **FACE**	5	
3.2. 7-Seg Questions	Answer Questions about the 7-Segment Display	5	
Section 4		-10-	
4.1. Screen shot of simulation	Screen shot of the simulation	5	
4.2. ALU Sim Questions	Answer Questions about the ALU Simulation	5	
Section 5		-10-	
5.1. ALU User Input Demo	Demonstrate ALU to TA, Prof	5	
5.2. ALU User Input Questions	Answer Questions about the ALU	5	
Lab Report		-50-	
Total		100	

Table 2: Lab Grade Breakdown Table