



ECE210-Intro. Digital Logic Design

LAB 0: Tutorial System Design Flow Using VHDL

Fall 2024

DATES

Date	Section	Time
24-Sep-2024	D21	08:00 - 10:50 AM
	D25	02:00 - 04:50 PM
25-Sep-2024	D31	08:00 - 10:50 AM
26-Sep-2024	D45	02:00 - 04:50 PM
01-Oct-2024	D22	08:00 - 10:50 AM
	D26	02:00 - 04:50 PM
02-Oct-2024	D32	08:00 - 10:50 AM
03-Oct-2024	D46	02:00 - 04:50 PM

INTRODUCTION

This lab manual is designed to familiarize students with the Xilinx Vivado tools that we will utilize throughout our lab exercises. The aim is to provide a solid foundation in understanding and using these tools effectively, enabling students to excel in their digital design projects and experiments.

Throughout the lab sessions, students will be introduced to various aspects of the Vivado Design Suite, including its features, functions, and capabilities. This lab manual will serve as a valuable resource, offering step-by-step instructions, illustrations, and examples to help students navigate the Vivado environment with ease.

It is important to emphasize that learning is an ongoing process, and we encourage students to actively engage with the material and push the boundaries of their understanding. By exploring the tools' capabilities beyond the given instructions, students can foster creativity, critical thinking, and problem-solving skills that will prove invaluable in their academic and professional pursuits.

LEARNING OBJECTIVES

- How to use the Xilinx Vivado tools for designing digital circuits and systems.
- Basics of VHDL [VHSIC (Very High Speed Integrated Circuit) Hardware Description Language].




ECE210-Intro. Digital Logic Design

LAB 0: Tutorial System Design Flow Using VHDL

Fall 2024


VHDL DESIGN FLOW

Design	Developing designs using VHDL code.
Simulation	Creating VHDL test benches for thorough design testing.
Synthesis	Transforming VHDL design into easily realizable components on FPGA fabric, such as standard digital logic, multiplexers, adders, and LUTs (Lookup Tables). This process generates a netlist.
Implementation	Executing all required steps to place and route the netlist on FPGA resources, adhering to the design's logical, physical, and timing constraints.
Programming and Debugging	Generating the FPGA bitstream file for programming and troubleshooting purposes.

	ECE210-Intro. Digital Logic Design
Fall 2024	LAB 0: Tutorial System Design Flow Using VHDL

LAB DEMONSTRATION REQUIREMENTS

Students must DEMO successful execution of instructions for parts 1 to 5 to a TA or LI.

	<h1>ECE210-Intro. Digital Logic Design</h1>
<p>Fall 2024</p>	<h2>LAB 0: Tutorial System Design Flow Using VHDL</h2>

INTRODUCTION TO VIVADO

This part of the tutorial is designed to familiarize students with the Xilinx Vivado tools that will be used to perform the lab exercises. Once the steps of the basic design flow are mastered, you are encouraged to explore Vivado's powerful capabilities.

Vivado: Xilinx Vivado is a software tool used for designing and implementing digital circuits on Xilinx FPGAs (Field-Programmable Gate Arrays). It provides a graphical interface to design and simulate circuits, synthesize them into a netlist, and map that netlist onto an FPGA device. It also allows for programming and debugging of the FPGA, and offers features such as IP (Intellectual Property) integration, timing analysis, and power optimization. Vivado is widely used in industries such as aerospace, telecommunications, and automotive, and is also popular in academic settings for teaching digital design and FPGA programming.



Figure 1: Vivado Icon

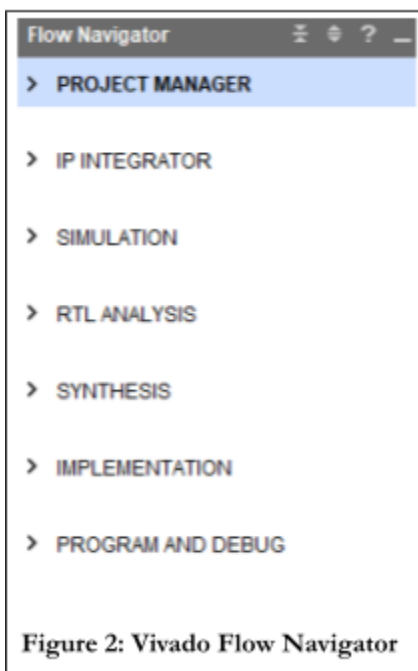


Figure 2: Vivado Flow Navigator

Vivado Flow Navigator: The Flow Navigator is a panel in the Vivado design environment that provides a visual representation of the various stages of the FPGA design process, from creating a new project to generating a bitstream for the target FPGA. It provides a step-by-step guide for the user to follow, and each step corresponds to a specific stage in the design process.

The Flow Navigator is divided into several sections, including:

Project Manager: this section allows you to create and manage your Vivado projects, and also provides access to source files, IP (Intellectual Property) cores, and other design elements. Here you create a high-level description of your design using a hardware description language (HDL) such as VHDL or Verilog. You can also create your design using a graphical interface, such as the Vivado IP Integrator.

Simulation: Testbenches are the standard way to test your designs, once you have your design you need to simulate it to ensure that it behaves as expected. Vivado includes a simulator that allows you to run different types of simulations.

Synthesis: After you've verified your design with simulation, the next step is synthesis. In this step, Vivado converts your high-level design into a gate-level netlist, which represents the design in terms of logic gates and other basic

elements like flip-flops and multiplexers.

Implementation: This is where your design is mapped onto the target device (in this course we will be using the Zybo z7 board), taking into account the physical constraints of the device, such as timing and power.

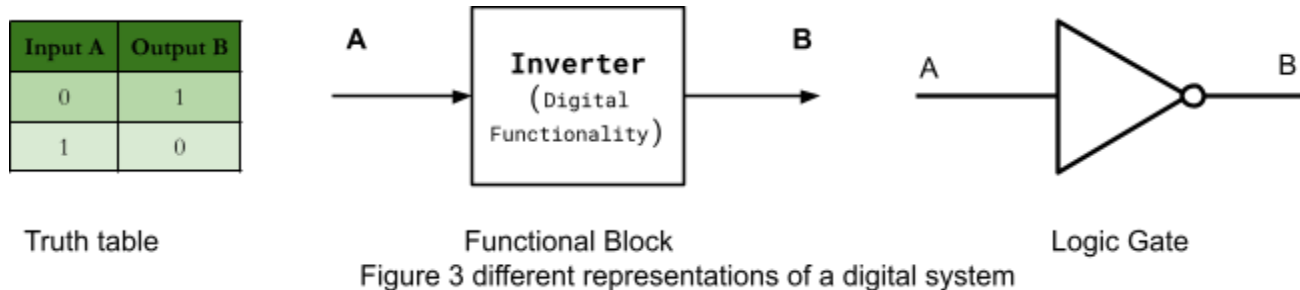
Bitstream generation: The final step is to generate a bitstream, which is the binary file that can be programmed onto the FPGA. Vivado generates the bitstream based on the design and implementation data and provides useful information about the number of components used and power consumption estimations.

BASIC DIGITAL SYSTEM DESIGN USING VHDL

VHDL, or VHSIC Hardware Description Language, is a widely-used language for describing and implementing digital circuits on integrated circuits (ICs), such as the Xilinx Zynq-7000 family part number **xc7z010 clg400-1L** on the Zybo Z7 board. Throughout each lab, you will be working with this board to bring your digital designs to life.

VHDL design files have a ".vhd" or ".vhdl" extension, while another popular hardware description language (HDL), Verilog, uses the ".v" extension for its files. Both languages enable designers to create and simulate digital systems on ICs.

A digital system can be conceptualized as a functional block with inputs and outputs. For instance, consider an inverter gate. When its input A is 0, the digital functionality transforms the output B to 1; conversely, when input A is 1, output B changes to 0. This relationship can also be represented using a truth table or simply by drawing a schematic diagram using logic gates:



Using VHDL, you can describe and implement various digital systems, ranging from simple gates like this inverter to complex designs that involve multiple components and logic blocks. As you progress through the labs, you'll gain hands-on experience with VHDL, developing a deeper understanding of digital design and their implementation on FPGA boards.

Now we will write a description for an inverter IO block using VHDL:

First we import any necessary library, most of the time we will use the IEEE standard library since it has all the main components we need.

```
LIBRARY IEEE; -- The IEEE VHDL library
USE IEEE.STD_LOGIC_1164.ALL; -- Using all in standard 1164
USE IEEE.NUMNERIC_STD.ALL; -- Using all in numeric standard
```



ECE210-Intro. Digital Logic Design

LAB 0: Tutorial System Design Flow Using VHDL

Fall 2024

Second, we need to declare the entity: The entity block is an essential part of every VHDL file. It serves as an interface to your digital circuit, defining its input and output ports. The entity block is crucial because it describes how your design will interact with the external environment, such as other components or systems. To declare an entity block, follow these steps:

1. Begin the declaration with the **ENTITY** keyword followed by the desired name for the entity.
2. Use the **IS** keyword to indicate the start of the entity definition.
3. Define the input and output ports of the entity within the **PORT** block, specifying their direction (**IN**, **OUT**, or **INOUT**) and data type (**STD_LOGIC** for single-bit or **STD_LOGIC_VECTOR** for multi-bit signals).
4. Close the entity declaration with the **END** keyword followed by the entity name.

following the steps above our inverter entity declaration looks like this:

```
ENTITY INVERTER IS
    PORT (
        A : IN STD_LOGIC; -- single-bit input.
        B : OUT STD_LOGIC -- single-bit output.
    );
END INVERTER;
```

Now that we have declared the entity block for our inverter, it's time to implement the design in our VHDL file. Below is the implementation of the architecture block of the INVERTER design. The architecture block is a vital part of a VHDL design file, where the actual implementation of the digital circuit is described. It defines the functionality and behavior of the entity using various design methodologies, such as dataflow, behavioral, or structural descriptions. The architecture block works in conjunction with the entity block, which provides the input and output interface for the digital circuit. In this implementation, we've chosen the **'BEHAVIORAL'** architecture, which describes the design's functionality in terms of its input-output relationships.

```
-- BEHAVIORAL is the architecture name
ARCHITECTURE BEHAVIORAL OF INVERTER IS
BEGIN
    B <= NOT A;
END BEHAVIORAL;
```

When using Vivado we use a constraint file to establish the association between our entity's IO declarations and the physical pins on the IC. This file links the entity declarations with their corresponding physical pins.



ECE210-Intro. Digital Logic Design

LAB 0: Tutorial System Design Flow Using VHDL

Fall 2024

Creating constraint files is beyond the scope of this lab, so you will be provided with ready-to-use constraint files for each exercise.

For the inverter, the constraint file content will look like this:

```
##Buttons
set_property -dict { PACKAGE_PIN K18 IOSTANDARD LVCMOS33 } [get_ports { A }]; #IO_L12N_T1_MRCC_35 Sch=btn[0]

##LEDs
set_property -dict { PACKAGE_PIN M14 IOSTANDARD LVCMOS33 } [get_ports { B }]; #IO_L23P_T3_35 Sch=led[0]
```

Notes:

- Any text prepended by a hash (#) character in a constraint file is commented out.
- As you can see the symbols A and B are resolved in this file and associated with physical pins.

The entire implementation of the inverter design file is shown below:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY INVERTER IS
    PORT (
        A : IN STD_LOGIC;
        B : OUT STD_LOGIC
    );
END INVERTER;

ARCHITECTURE BEHAVIORAL OF INVERTER IS
BEGIN
    B <= NOT A;
END BEHAVIORAL;
```



ECE210-Intro. Digital Logic Design

LAB 0: Tutorial System Design Flow Using VHDL

Fall 2024

PART 1

Marks-10%

1. Open Vivado by double-clicking on its icon on your desktop.
2. Begin a new project: click on **'File'**, then **'Project'**, and finally **'New'**. A wizard will pop up.
3. In the **'Project Name'** step, name your project **'tutorial'**. Choose where you want your project files to be stored, and then click **'NEXT'**.
4. In the **'Project Type'** step, keep the default selection, which is **'RTL Project'**, then click **'NEXT'**.
5. In the **'Add Sources'** step, choose **'VHDL'** as your target language at the bottom of the window. Then, create a new file named **'tutorial'**. Click **'OK'**, then **'NEXT'**.
6. In the **'Add Constraints'** step, download the **'Zybo-Z7-Master.xdc'** file from the eClass site and add it to your project directory. You can add it here using the **'Add Files'** button, then click **'NEXT'**.
7. In the **'Default Part'** step, choose **'Zynq-7000'** as the family. Then select **'xc7z010clg400-1'** as the part, and click **'NEXT'**.
8. Read New Project Summary and then click Finish. Since previously in step 5 we chose Create File, now a new window Module Definition will show up. Enter the following ports for our design, and click OK. Now we need to implement our design using VHDL, double click on tutorial.vhd file under Sources to edit. One open you can see that Vivado tools already made a default VHDL template. Above process can also be seen in a demo video at [this link](#).



I/O Port Definitions					
+ - ↑ ↓					
Port Name	Direction	Bus	MSB	LSB	
sw	in	<input checked="" type="checkbox"/>	3	0	
led	out	<input checked="" type="checkbox"/>	3	0	
btn0	in	<input type="checkbox"/>	0	0	

9. Complete design as shown below. The truth table of this design is given on the next page. The schematic of the implemented design is given in the appendix. Again iterating key points when you read and try to understand the given design,
 - a. The VHDL code within the process block runs sequentially.
 - b. The last four VHDL design lines updating LED status will run in parallel.



ECE210-Intro. Digital Logic Design

LAB 0: Tutorial System Design Flow Using VHDL

Fall 2024

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY tutorial IS
    PORT (sw      : IN STD_LOGIC_VECTOR(3 DOWNTO 0);
          led     : OUT STD_LOGIC_VECTOR(3 DOWNTO 0);
          btn0    : IN STD_LOGIC);
END tutorial;

ARCHITECTURE Behavioral OF tutorial IS
    SIGNAL y : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0000";
BEGIN
    encoder_and_decoder : PROCESS (btn0) IS
        BEGIN
            IF btn0 = '0' THEN
                y(0) <= sw(3) OR ((NOT sw(2)) AND sw(1));
                y(1) <= sw(3) OR sw(2);
                y(2) <= sw(0) OR sw(1) OR sw(2) OR sw(3);
                y(3) <= '0';
            ELSE
                y(0) <= (NOT sw(0)) AND (NOT sw(1));
                y(1) <= (NOT sw(1)) AND sw(0);
                y(2) <= sw(1) AND (NOT sw(0));
                y(3) <= sw(0) AND sw(1);
            END IF;
        END PROCESS;
        led(0) <= y(0);
        led(1) <= y(1);
        led(2) <= y(2);
        led(3) <= y(3);
    END Behavioral;
```

PART 2

Marks-10%

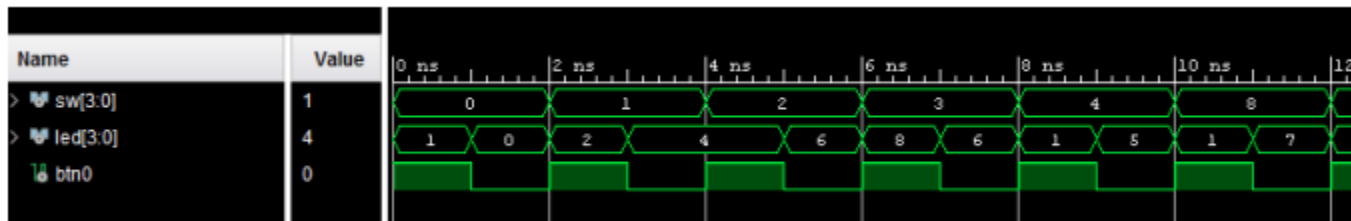
Now we are going to verify our design by writing a simulating VHDL testbench.

1. Click the “+” button in Sources window and select **Add or create simulation sources** in **Add Sources** window, click **Next**.
2. Click **Create File** and type file name **tutorial_tb** in the window shown and click **OK**.
3. On the next window click Finish and Click **OK** on **Define Module** window.
4. Now open **tutorial_tb.vhd** file under **Simulation Sources** in **Sources** window

Note: demo video at [this link](#).

5. View or download **tutorial_tb.vhd** from course web site and either type or copy file contents into your testbench file, **alternatively, you can simply remove the existing testbench file and include the add the testbench file in your project under simulation sources similar to step 1.**

6. In **Flow Navigator** under simulation, click **Run Simulation->Run Behavioural Simulation**, you should get displayed as shown below. Truth table for both circuits given as well for quick verification.



inputs		outputs			
sw0	sw1	led0	led1	led2	led3
0	0	1	0	0	0
1	0	0	1	0	0
0	1	0	0	1	0
1	1	0	0	0	1

2-4 bit decoder truth table




ECE210-Intro. Digital Logic Design

LAB 0: Tutorial System Design Flow Using VHDL

Fall 2024

inputs				outputs		
sw0	sw1	sw2	sw3	led0	led1	led2
0	0	0	0	x	x	0
1	0	0	0	0	0	1
x	1	0	0	1	0	1
x	x	1	0	0	1	1
x	x	x	1	1	1	1

4-2 bit priority-encoder truth table

 Fall 2024	<h1 style="text-align: center; color: blue;">ECE210-Intro. Digital Logic Design</h1>
	<h2 style="text-align: center; color: green;">LAB 0: Tutorial System Design Flow Using VHDL</h2>

PART 3: RTL ANALYSIS

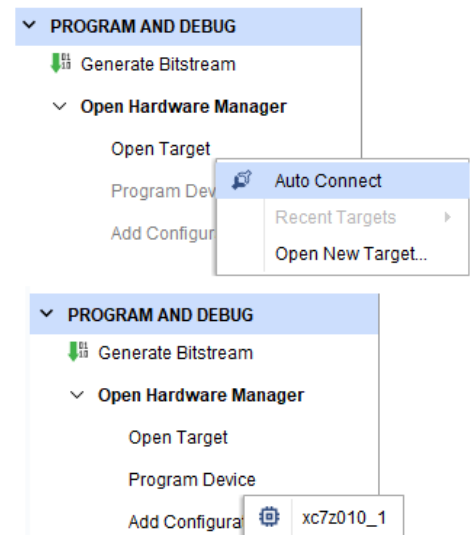
Marks-10%

1. This section demonstrates how to view the RTL schematic of design.
2. In Flow Navigator, expand RTL Analysis and Open Elaborated Design and then click Schematic

PART 4: SYNTHESIS, IMPLEMENTATION, BITSTREAM, PROGRAMMING FPGA

Marks-20%

1. In the **Flow Navigator**, expand **Synthesis** (a process of translating RTL specific design into gate level implementation.) and click **Run Synthesis** on next window click **OK**.
2. Once the synthesis process is complete, a window popup shows **Run Implementation** (a process to implement design on FPGA), select this option (default selection) and click **OK**, on next window click **OK** again. Zoom-In to see the implementation.
3. Once the implementation process is complete, a window popup, you can select **Open Implemented Design** to see how much FPGA fabric is occupied by design and to optimize. You can zoom-in to see details, colors specify occupied FPGA fabric.
4. Select **Generate Bitstream** option and click **OK**, on the next window click **OK**.
5. On successful completion, a new window popup, select **Open Hardware Manager**. Kindly make sure that the USB cable is properly connected with the Zybo board and its powered ON.
6. In the **Flow Navigator** window, now select **Open Target** then **Auto Connect** as shown below. If an error message window appears, under local host, right click and open following Xilinx part:
xilinx_tcf/Digilent/210351A77C0BA
7. In the **Flow Navigator** window, click **Program Device**, then select xc7z010_1, a popup **Program Device** window appears, click **Program**. Test your design on Zybo Z7 by toggling switches and pressing btn0 for monitoring decoder outputs.



After successful synthesis, students can also view schematic under,
SYNTHESIS->Open Synthesized Design

Note: The way that the constraint file was written for this project will display the binary value starting from the less significant bit to the most significant bit, this is because the boards are attached onto the table in an upside down fashion



ECE210-Intro. Digital Logic Design

LAB 0: Tutorial System Design Flow Using VHDL

Fall 2024

REFERENCES

- [VHDL essentials Youtube Playlist](#)
- [Zybo Z7 Documentation, Tutorials and Example Projects](#),
- [Zybo Z7 Reference Manual](#)
- “Digital System Design with FPGA: Implementation Using Verilog and VHDL” ; Cem Ünsalan, Ph.D., Bora Tar, Ph.D; Pub. Date: 2017 McGraw-Hill Education; ISBN: 9781259837906



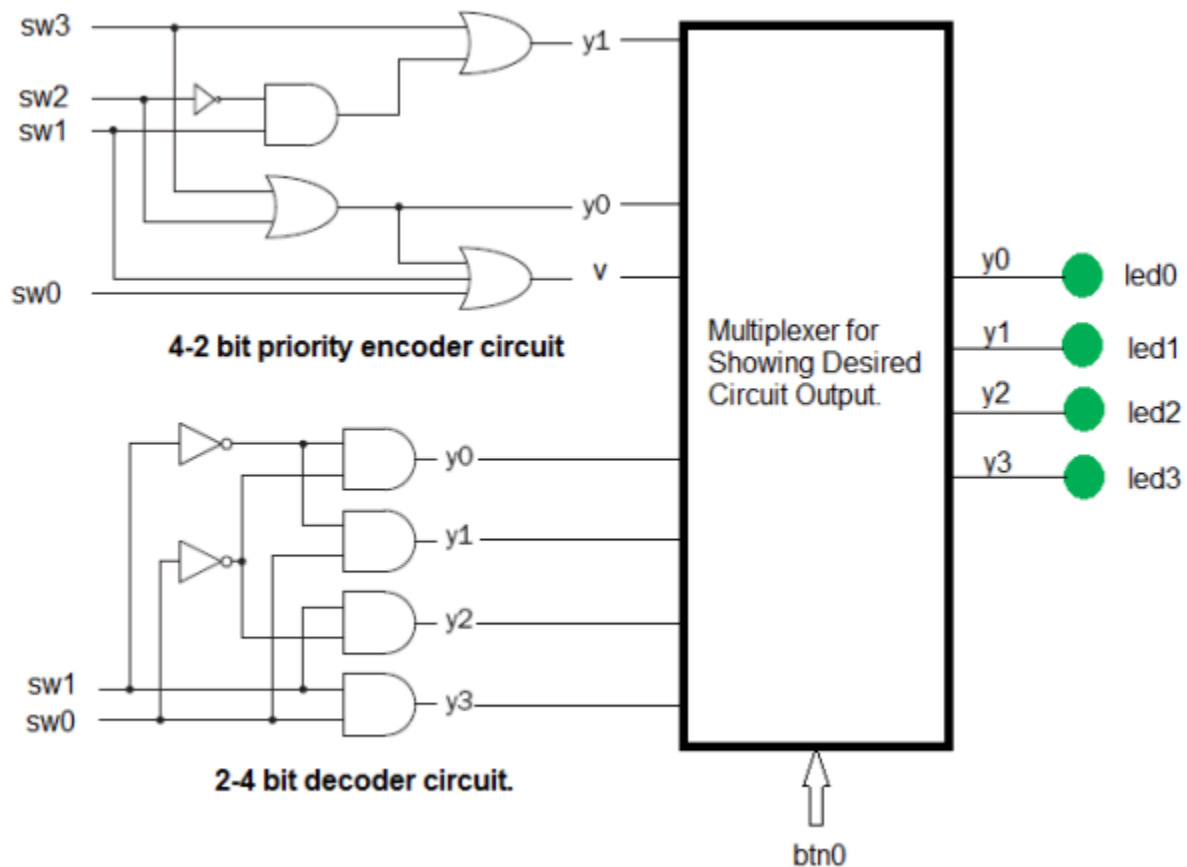
ECE210-Intro. Digital Logic Design


LAB 0: Tutorial System Design Flow Using VHDL

Fall 2024

APPENDIX

Schematic of design implemented in this tutorial lab0. Some of the technical terms mentioned will be covered later during lectures.



	ECE210-Intro. Digital Logic Design
Fall 2024	LAB 0: Tutorial System Design Flow Using VHDL

ECE 210 LAB 0 - MARKING GUIDELINES

Student Names:_____

Pre-Lab

(50) Successfully Attempted Lab Safety Video Quiz.

Sub-total_____/50

In-Lab

(50) Successfully Completed Lab0 Assignments.

TA Signatures:_____ Sub-total_____/50

Lab Report

No Lab Report.

Lab Report Total_____/100