The Hong Kong University of Science and Technology
Department of Electronic and Computer Engineering

EESM5060 **EMBEDDED SYSTEM** FALL 2023-24

LAB 2 – Simple Digital Logic

1. Lab Objective

The aims of this lab session are:

1.1 To familiarize with Xilinx Artix-7 Evaluation Platform with a series of simple exercises.

1.2 To describe a digital system using Verilog and implement it on a Xilinx FPGA, which is Artix-7 in the present case, using Xilinx's Vivado Design Suite - HLx Editions – 2020

2. Learning Outcomes

At the end of this lab session, you would have learned the following:

2.1 Describing a digital design in Verilog

2.2 Modeling, simulation, and implementation of digital circuits using Xilinx's Vivado Design Suite - HLx Editions

2.3 Understanding Xilinx Design Constraints (XDC) for pin assignment and timing constraints

2.3.1 Note that Vivado does not support the legacy UCF format (UCF2XDC migration).

2.4 Synthesizing and generating FPGA bit-stream for Xilinx FPGA

2.5 Overall idea of FPGA Electronic Design Automation Tool flow

Note:
You will be required to carry out the steps listed under "Learning Outcomes" in future lab sessions as well. The instructions provided in Lab 1 will not be repeated in the other labs with respect to these steps. So, you are expected to finish lab 1 diligently as it lays the foundation for other labs.

3. Lab Exercises

In the exercises in Lab 1, you will control the 8 LEDs on the board using the 8 slide switches.

Notice the location of the 8 LEDs and the 8 slide switches are on the right of the board as shown in Fig.1. The 8 slide switches are connected to 8 user pins on the FPGA through traces on the PCB, and the 8 LEDs are connected to another 8 user pins on the FPGA. It is these pins (each pin has a pin number) that are specified in your XDC file.

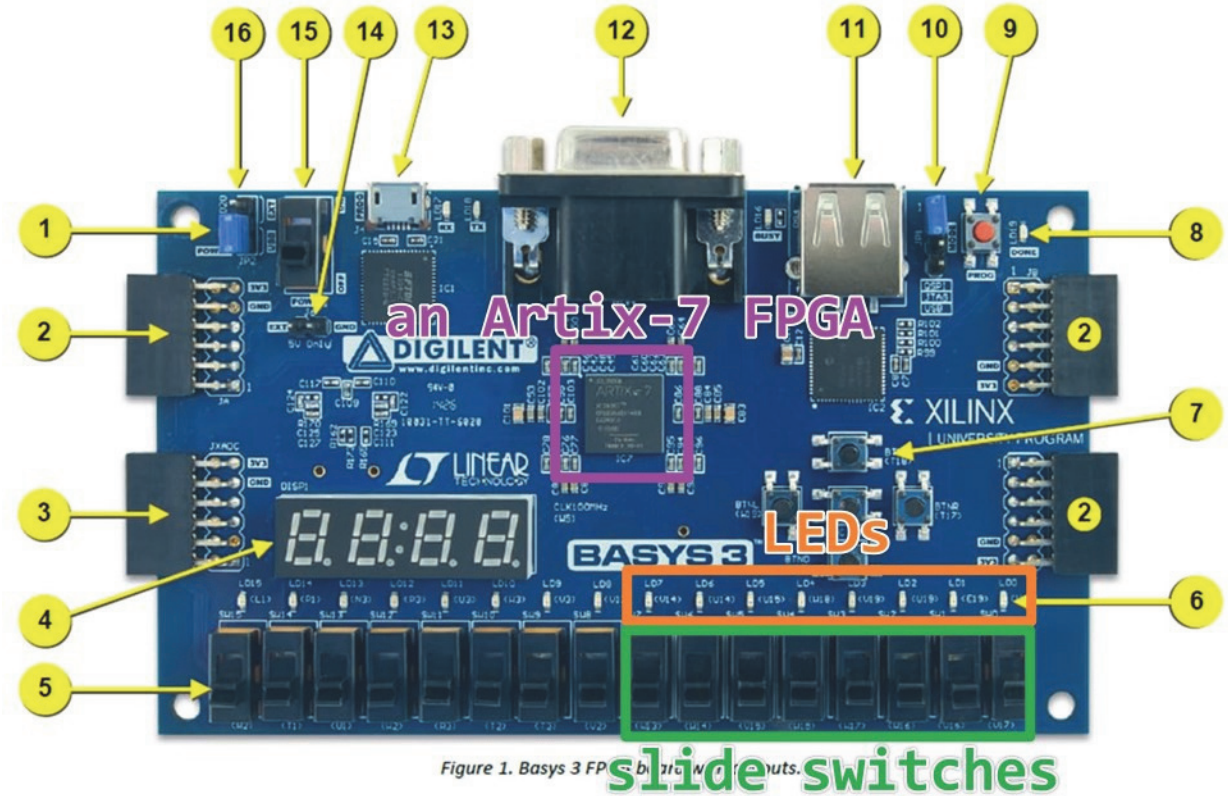


Figure 1. Basys 3 FP[illegible] board with callouts.

Callout	Component Description	Callout	Component Description
1	Power good LED	9	FPGA configuration reset button
2	Pmod port(s)	10	Programming mode jumper
3	Analog signal Pmod port (XADC)	11	USB host connector
4	Four digit 7-segment display	12	VGA connector
5	Slide switches (16)	13	Shared UART/ JTAG USB port
6	LEDs (16)	14	External power connector
7	Pushbuttons (5)	15	Power Switch
8	FPGA programming done LED	16	Power Select Jumper

Figure 1. A BASYS 3 Board, with An Artix-7 (XC7A35T) FPGA.

4. Exercise 1A

4.1 Controlling LEDs through slide switches

You are required to write a Verilog module that takes 8 bits from switches as the input and show their value as the output. Using the knowledge that an input bit value of 1 when connected to an LED will drive the LED high, making it glow, you can write the Verilog module.

4.2 Steps to be carried out

4.2.1 Double click on the Vivado icon to open up the welcome window of the development tool (as shown below). Now, click on "Create Project" to create a new project.

Figure 2. Open Your Vivado.

4.2.2 Name your project, e.g., 'Lab2-1A'.

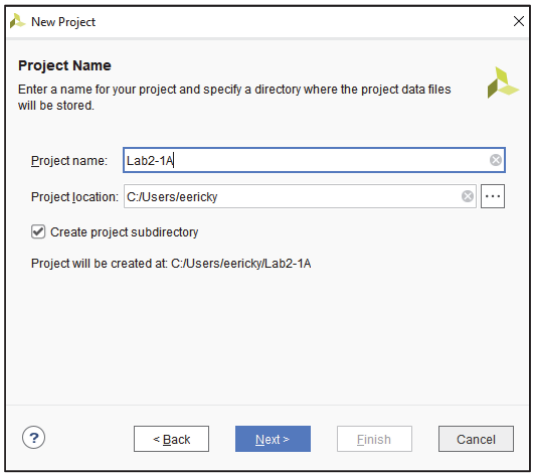


Figure 3. Name Your New Project.

4.2.3 In the next window, choose “RTL Project” as the project type. You can see the description of this type in the window.

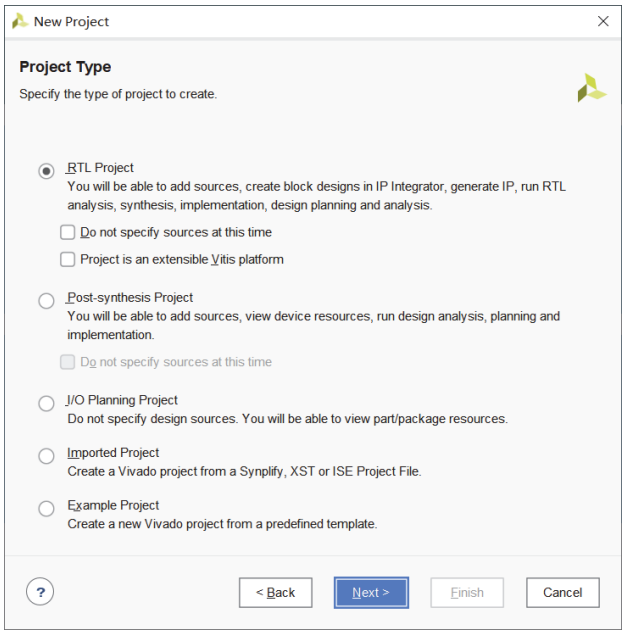


Figure 4. Choose “RTL Project” as the Project Type.

4.2.4 You can create a source file (Verilog/Verilog Header/SystemVerilog/VHDL/Memory File) for your new project or add sources from the existing projects. Click on “Create File”, and in the opened window choose “Verilog” for the “File type”, write a name for your file (“control_LED”), and click on “Ok”. Continue clicking on Next. Note that you can also choose “Add Files” to add existing files to the project.

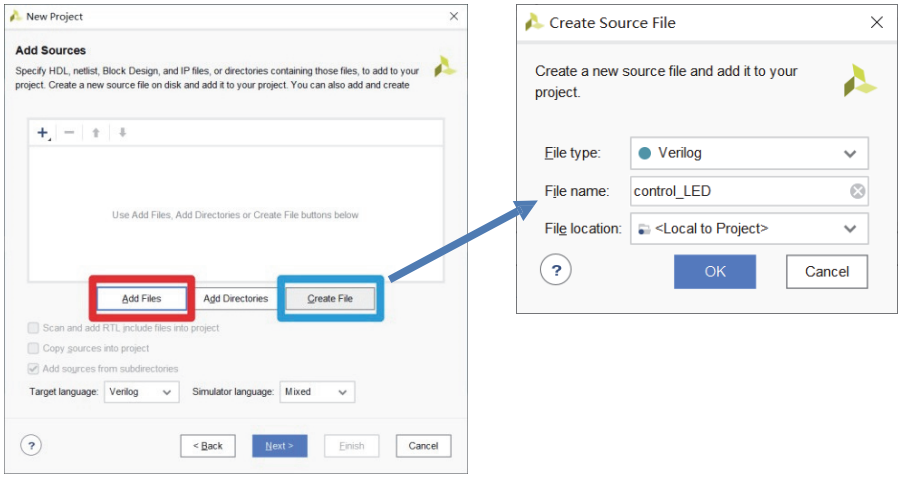


Figure 5. Create / Add Existing Verilog Source Files.

4.2.5 Similarly, you can create a constraint file or add an existing XDC file. Click on "Create File" and in the opened window write a name for your file("control_LED"), and click on "Ok". Continue clicking on Next.

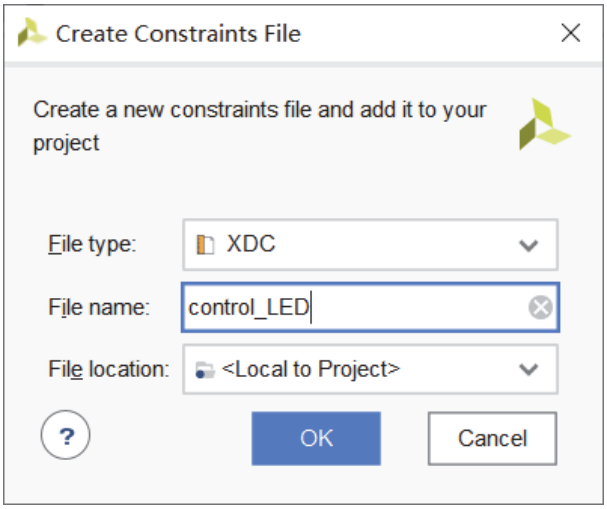


Figure 6. Create / Add An Existing Constraint (XDC) File.

4.2.6 In this window, choose "Artix-7" for the "Family", "-1" for "Speed grade", and "cpg236" for "Package". In the shown parts, select "xc7a35tcpg236-1". More information about the board can be found at https://reference.digilentinc.com/basys3/refmanual.

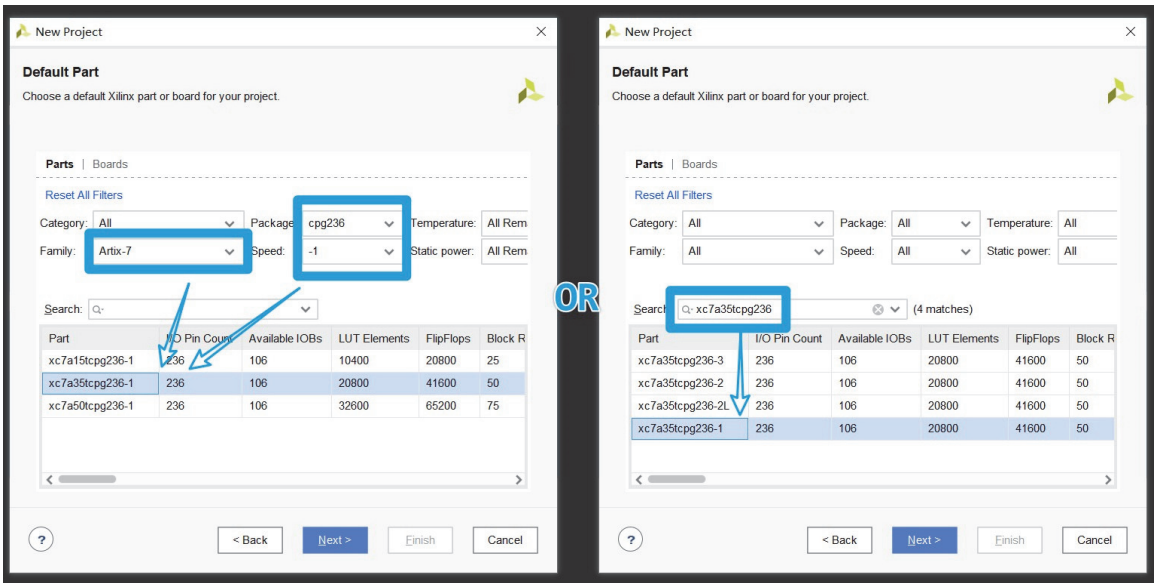


Figure 7. Choose the Xilinx Part/Board for Your Project.

4.2.7 Look at your new project summary, and click on the “Finish” button to continue.

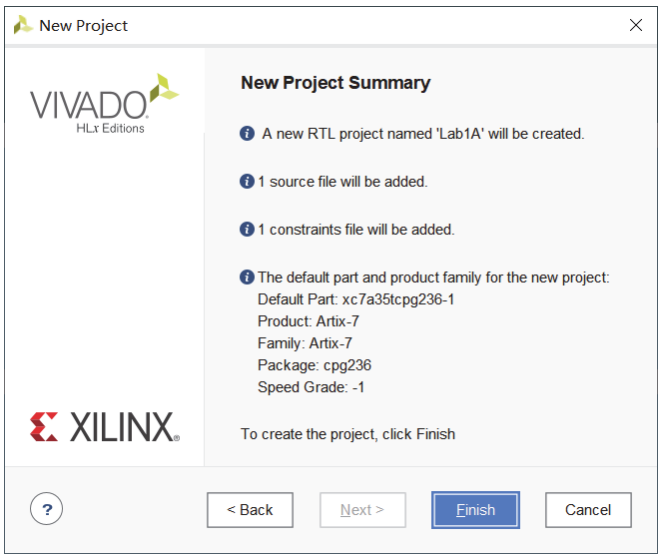


Figure 8. Double Check the Project Summary.

4.2.8 Define the input and the output ports of your module, and click on “OK” to continue.

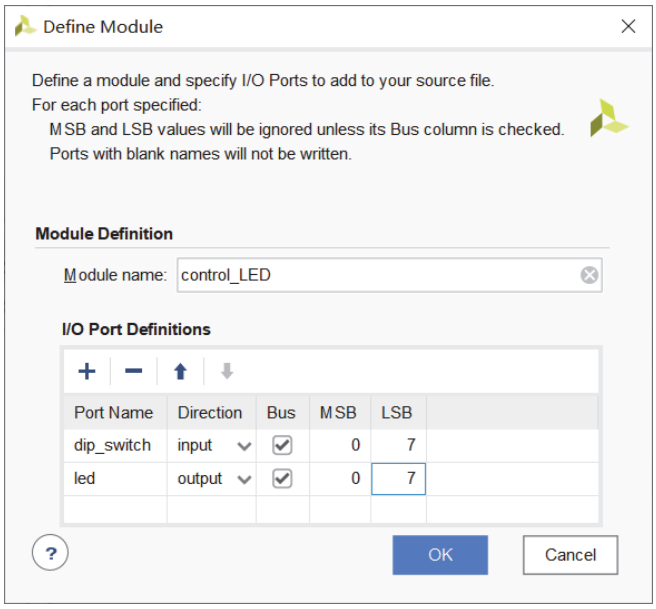


Figure 9. Define the I/O Ports of Your “control_LED” Module.

4.2.9 Now, the opened window is the main environment for your project, called the "Project Manager." You can explore it by seeing the options of each category in the toolbar on top of the window. In the left panel, the "Flow Navigator," you can see the "Settings," "Add Sources," "Language Template," "IP Catalog," "IP Integrator," "Simulation," "RTL Analysis," "Synthesis," "Implementation," and "Program and Debug." Each of these serves a part of the digital design flow. In the middle, you can see the windows for "Sources," "Properties," "Project Summary," and the reports and summaries for the execution of the project files.

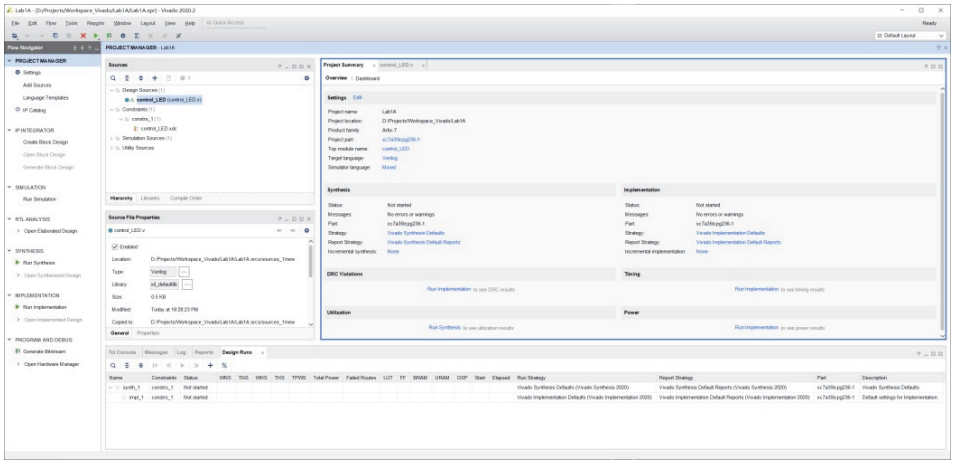


Figure 10. The Main Environment for Your Project. (Design Flow, Source, Summary, Log, etc.)

4.2.10 Double-click on the "control_LED.v" in the "sources" window. The VERILOG source file appears where the window is located on the right side. Note that the module shows the defined inputs and outputs that were selected previously. Now, you simply fill in the code to make the appropriate assignment to output.

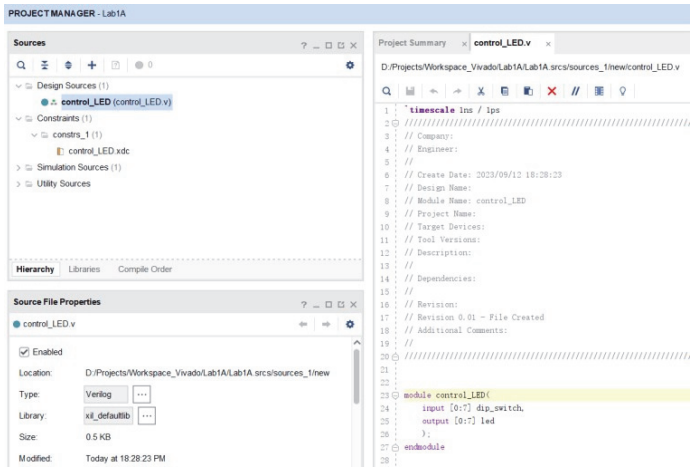


Figure 11. Open the Verilog Source and Fill in the Missing Code.

4.2.11 Change to "Simulation". Select Behavioral from the drop down menu.

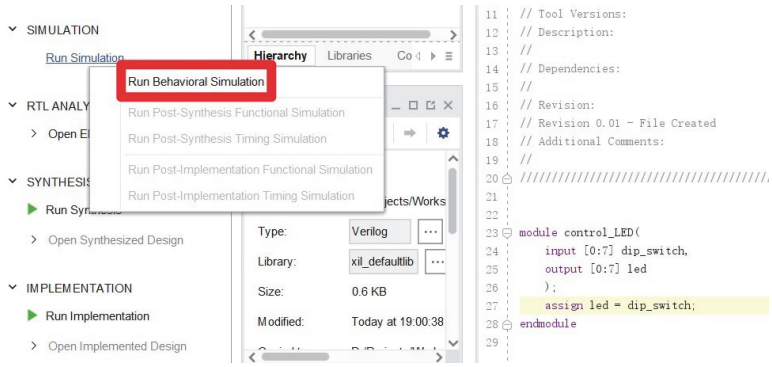


Figure 12. Run Behavioral Simulation.

4.2.12 A separate simulation window pops up. You can see input and output in the waveform window.

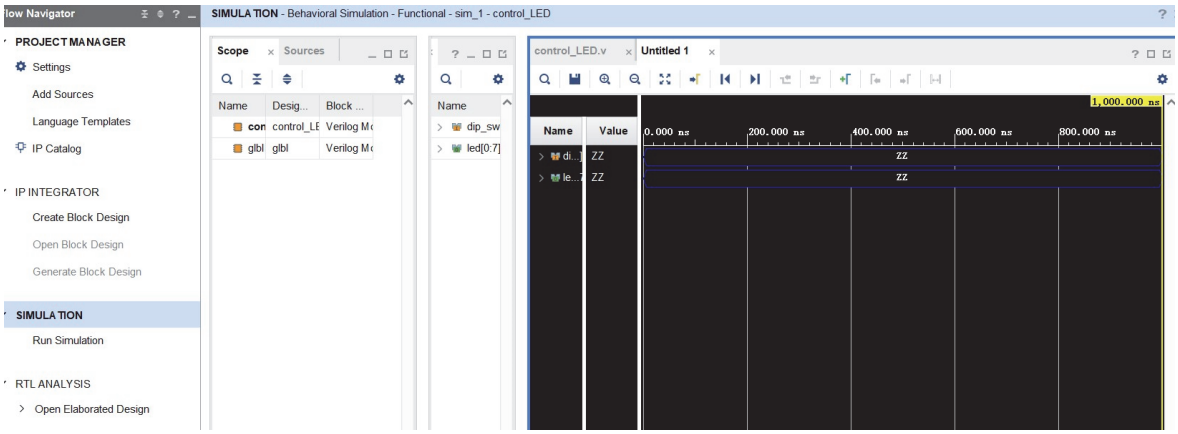


Figure 13. View the Default Waveform with No Input Value Assigned.

4.2.13 Right click on dip_switch and select Force Constant and type "10000001" in Force to Value; make sure Value radix in "Binary" click Apply and Ok.

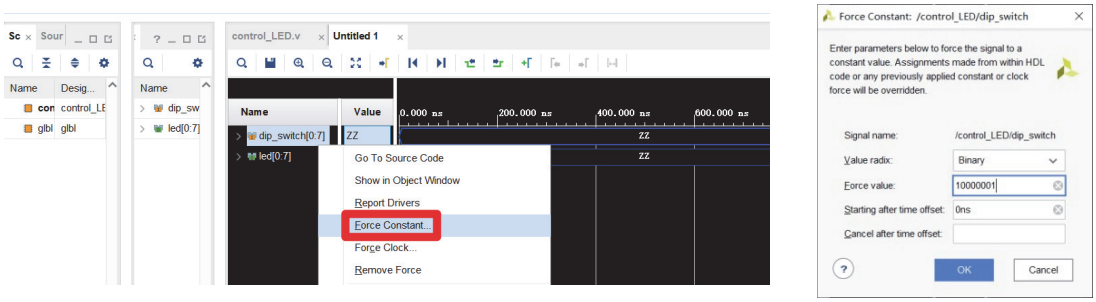


Figure 14. Force Constant Assignment.

(Note: previously named as "dip_switch", but on Basys3 we are using slide switches physically.)

4.2.14 Click on the run button in the top toolbar.

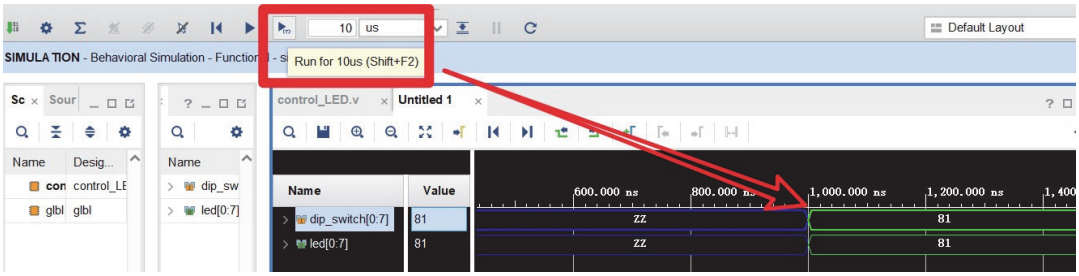


Figure 15. Run Simulation Again with Constant Input Assigned.

4.2.15 You can see that the inputs and the outputs match. Play with the zoom buttons to zoom in or out the waveform. If inputs and outputs are the same, your code is functionally correct. You just now performed "Functional Simulation" to test the functional correctness of your code. Try repeating steps 4.2.13 and 4.2.14 for other input values.

4.2.16 Close the simulation window.

4.2.17 The circuit has been implemented but the Xilinx tools still need to know which physical pins on the FPGA the input and output are mapped to. Double click on "control_LED.xdc". Now, write the tcl code to assign pins physically to FPGA. The pin assignment is provided in Table A and Table B in appendix. The tcl script for pin assignment is given below. More information about pin assignment can be found at https://reference.digilentinc.com/basys3/refmanual.

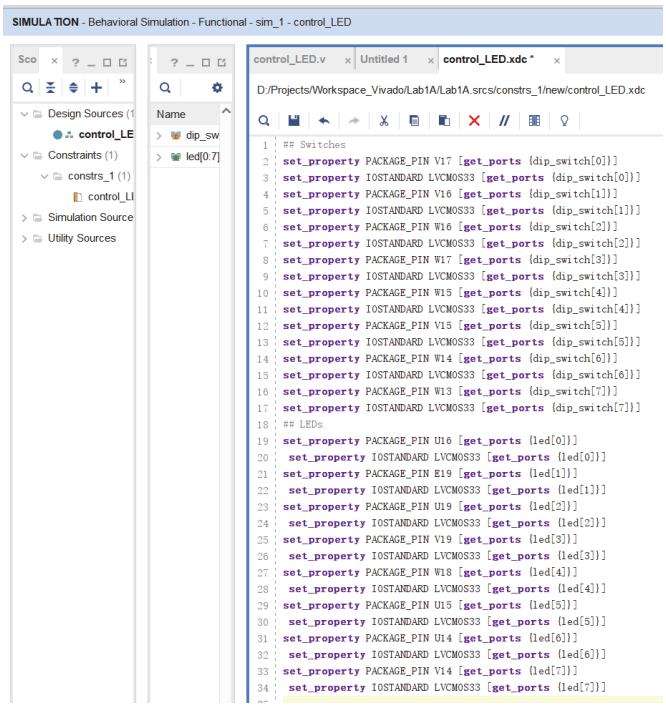
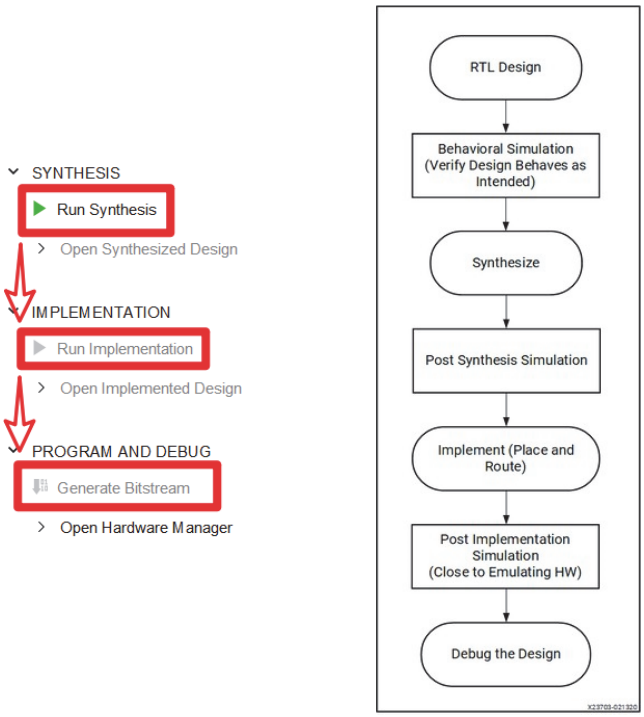


Figure 16. Pin Assignment in the XDC file.

4.2.18 Now that the design is finished, you must build the project. Click "**Run Synthesis**" on the left-hand menu towards the bottom. On successful completion, click on "**Run Implementation**". On successful completion of implementation, click on "**Open Implemented Design**" in the dialog box that appears. You can see information about resource usage and related data as shown below. Now click on "**Generate Bitstream**".



Note: At each step check warning or information at the very bottom of your screen.

Figure 17. Run Implementation Flow in Vivado.

4.2.19 On Successful "**Generate Bitstream**", you can program the bitstream on FPGA. Click on "**Open Hardware Manager**" → "**Open Target**" → "**Auto Connect**" → "**Program Device**". Then, you can play with the slide switches and see the corresponding LEDs on and off.

After you have done Exercise 1A above, proceed to do exercises 1B. Part of the previous steps (4.2.1 - 4.2.19) above will need to be repeated for 1B below.

5. Exercise 1B

5.1 You are required to implement a simple FIR filter and pass through behavioral simulation. The FIR filter is shown below. There is a line buffer with four elements. In every clock cycle, the design accepts one input data Data_in, *r1* will buffer its value, and the original data in the line buffer will shift right. For example, data in *r2* will be stored in *r3*. The data in the line buffer will multiply with a coefficient and at last you need to sum up the four multiplication results to get the final result. We provide you with an example code "fir_example.v" together with the testbench "fir_test.v" but you are required to fill in some verilog code in the missing part under the comments to complete the function described above. Finally you need to pass through the behavioral simulation to get the correct result.

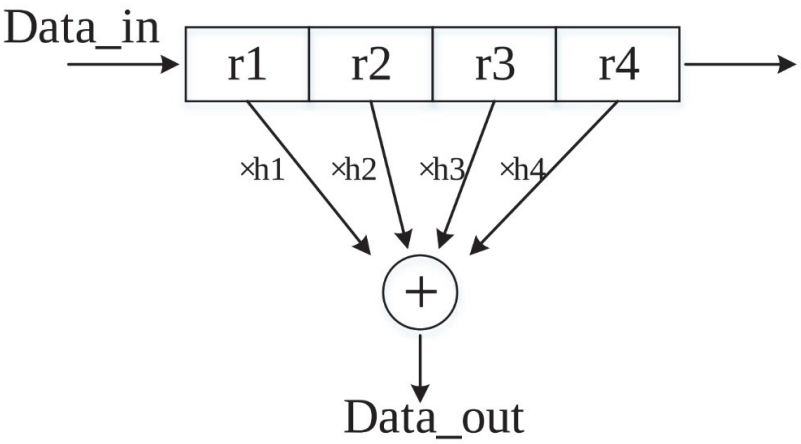


Figure 18. A Visualization of the Simple FIR Filter To Implement.

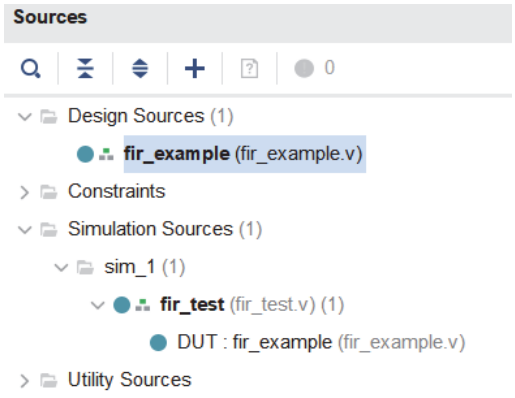


Figure 19. Import The Source Code and Testbench to Your Project, and "Set as Top".

5.2 You are required to implement a 3-to-8 decoder. We provide you with an example code, "decoder.v," but you are required to fill in some Verilog code in the missing part under the comments to complete the decoder. You are required to use the 3 slide switches for the 3 inputs and show the decoder output by driving the LEDs. Also, remember that you need to add the .xdc file for IO configuration. Please show the post-implementation timing simulation screenshot, and photos of your on-board test with some explanation of the relationship between the switches and LEDs.

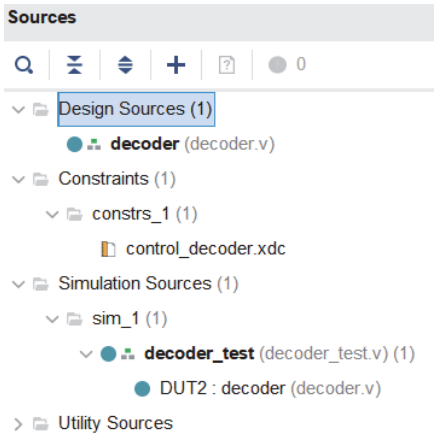


Figure 20. Project Hierarchy for the Lab1B-decoder Task.

3 to 8 Line Decoder

Inputs			Outputs							
x	y	z	D0	D1	D2	D3	D4	D5	D6	D7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

Figure 21. 3-to-8 Decoder Truth Table

APPENDIX I – Pins on Basys3

Table A: Slide Switch pins

Slide Switch	FPGA Pin
SW0	V17
SW1	V16
SW2	W16
SW3	W17
SW4	W15
SW5	V15
SW6	W14
SW7	W13

Table B: LED pins

LED	FPGA Pin
LD0	U16
LD1	E19
LD2	U19
LD3	V19
LD4	W18
LD5	U15
LD6	U14
LD7	V14