ECE 4730: Embedded Systems II

Department of Engineering Utah Valley University

Dr. Ehsan Rohani

This manual is in modified with permission from the original work by Ramu Endluri, He Zhou, Andrew Douglass and Sunil P Khatri by Ehsan Rohani and Brock Brown

Project 1: Using the Vivado for Hardware Implementation on FPGA

Objective

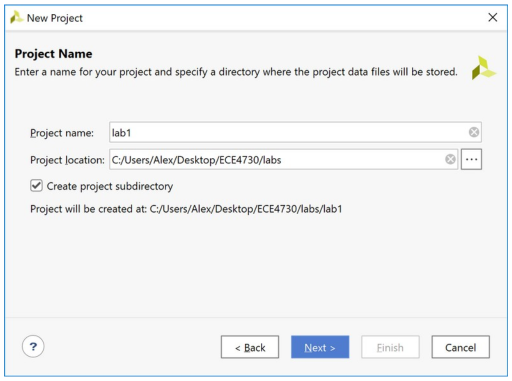
1. Familiarize you with Xilinx FPGA design flow in Vivado 2019.1
2. Refresh your Verilog writing skills.
3. Introduce the use and hardware inputs and outputs.

In This Lab

We will walk you through using Vivado to create a hardware, which lights up the LEDs on the ZYBO board depending on the status of the on-board DIP switches. Then we will guide you to make a Pulse-Width-Modulated (PWM) LED; and finally, you are required to implement a simple counter and jackpot game on your own with the knowledge gained from the first part of this lab

LED Light Up Walkthrough

1. Create a folder for your ECE4730 lab work . Spaces or special characters will cause problems (+,”,’,;,’,’,., etc.). You may use _ and -.
2. Launch the Vivado program and create a new design project.
   (a) Double click on the Vivado icon on the desktop.
   (b) Once in Vivado, select File → Create New Project
   (c) The New Project Wizard opens. Click Next. (Figure 1).


New Project
Project Name
Enter a name for your project and specify a directory where the project data files will be stored.
< Back
Next >
Finish
Cancel

Project name: lab1

Project location: C:/Users/Alex/Desktop/ECE4730/labs

☑ Create project subdirectory

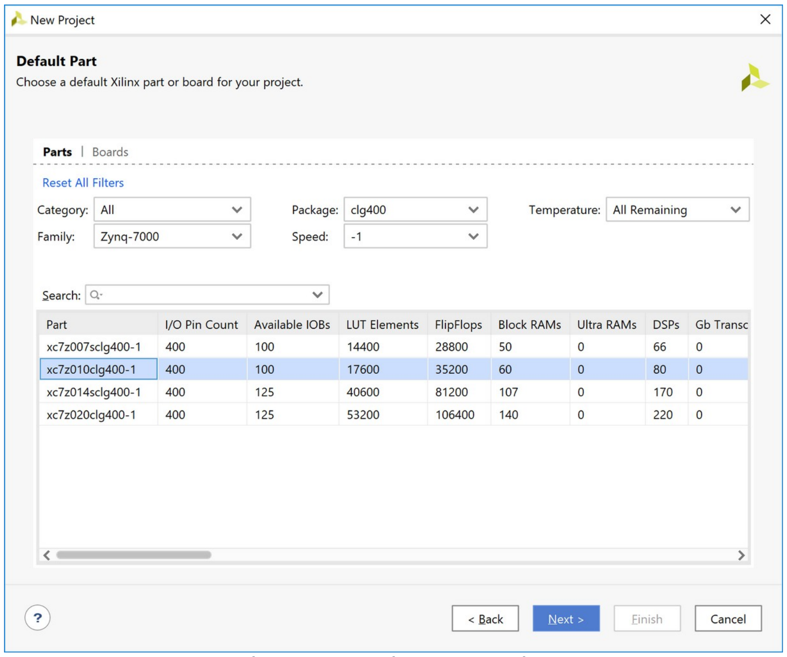
Project will be created at: C:/Users/Alex/Desktop/ECE4730/labs/lab1

Figure 1:Create New Project

(d) Select a Project Name (ex. lab1) and a Project Location (ex. /ECE4730/lab1 in your home directory). Then check the create project subdirectory and click Next.

(e) Select RTL project and leave 'Do not specify sources' unchecked at this time. Click Next. You will see 'Add Sources' window, select 'Target language' as Verilog and 'Simulator language' as Mixed. Click on the blue '+' button and select 'Create file', a window pop up will appear. Select 'File type' as Verilog, 'File name' as 'switch', and select 'File location' as '<Local to Project>', click 'OK' to create the Verilog source file. Click 'Next' and the 'Add Constraints(optional)' window will appear. We will add the Constraints File later in the lab. Click 'Next'.

(f) Next, the 'Default Part' window appears (Figure 2). We can select our hardware from the 'Parts' tab or from the 'Boards' tab. The 'Parts' tab lists Xilinx supported Parts(FPGAs) and the 'Boards' tab lists the supported boards. The ZYBO (Zynq Board) is an entry-level digital circuit development platform built around the Xilinx Zynq-7000 family, the Z-7010. The Z-7010 is based on the Xilinx All Programmable System-on-Chip (AP SoC) architecture, which integrates a dual-core ARM Cortex-A9 processor with a Xilinx 7-series Field Programmable Gate Array (FPGA) logic. In this lab and the next we will select the hardware from the 'Parts' tab and in the later labs we will select the hardware using the 'Boards' tab. Select the hardware using the following parameters.



New Project

Default Part

Choose a default Xilinx part or board for your project.

Parts | Boards

Reset All Filters

Category: All   Package: clg400   Temperature: All Remaining

Family: Zynq-7000   Speed: -1

Search:

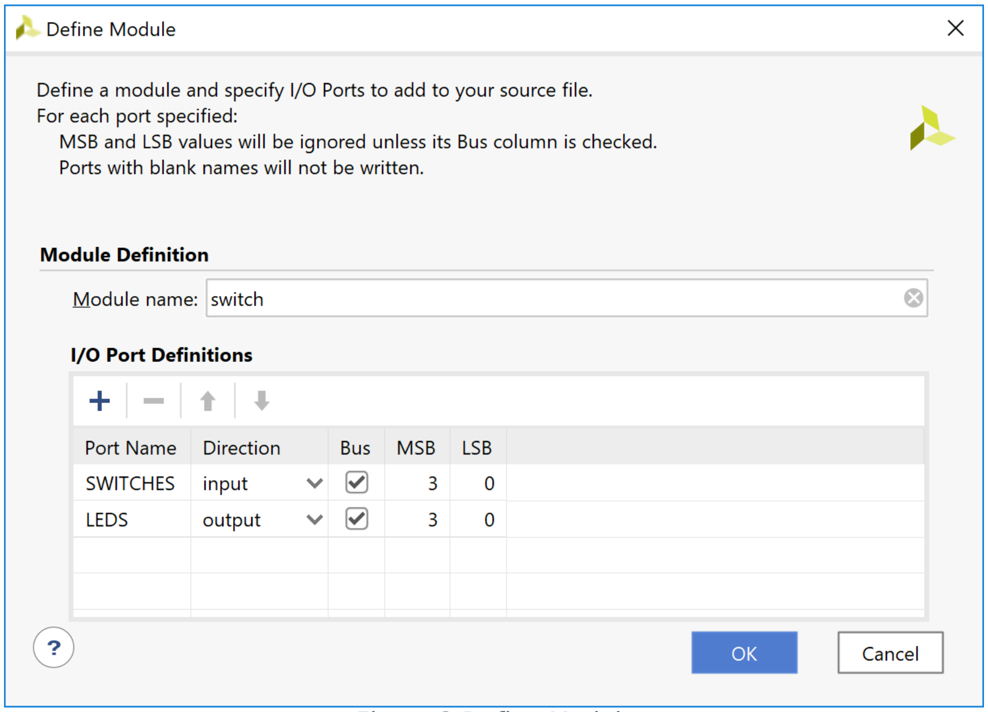
Part	I/O Pin Count	Available IOBs	LUT Elements	FlipFlops	Block RAMs	Ultra RAMs	DSPs	Gb Transc
xc7z007sclg400-1	400	100	14400	28800	50	0	66	0
xc7z010clg400-1	400	100	17600	35200	60	0	80	0
xc7z014sclg400-1	400	125	40600	81200	107	0	170	0
xc7z020clg400-1	400	125	53200	106400	140	0	220	0

Figure 2:Device Properties

Set the device properties to the following:
Device Family: Zynq-7000
Package: clg400
Speed Grade: -1

(g) You will see two devices. Select the device with part number ‘xc7z010clg400-1’ and click ‘Next’. Finally, review the information in the ‘New Project Summary’ window and hit ‘Finish’ to create project.

(h) Next, the ‘Define Module’ window will appears (Figure 3). This allows us to define the ports for our hardware module. Xilinx will auto generate part of our source file based on the information provided. Specify a port called ‘SWITCHES’. Set its direction to ‘input’, check Bus, set the Most Significant Bit (MSB) to 3, and set the Least Significant Bit (LSB) to 0. Specify another port called ‘LEDS’ and set the direction to ‘output’, check ‘Bus’, set MSB to 3, and set LSB to 0. This will create a 4-bit input port, which will connect to the on-board slide switches, and a 4-bit output port, which will connect to the on-board LEDs. Click ‘OK’.


Define Module

Define a module and specify I/O Ports to add to your source file.
For each port specified:
MSB and LSB values will be ignored unless its Bus column is checked.
Ports with blank names will not be written.

Module Definition

Module name: switch

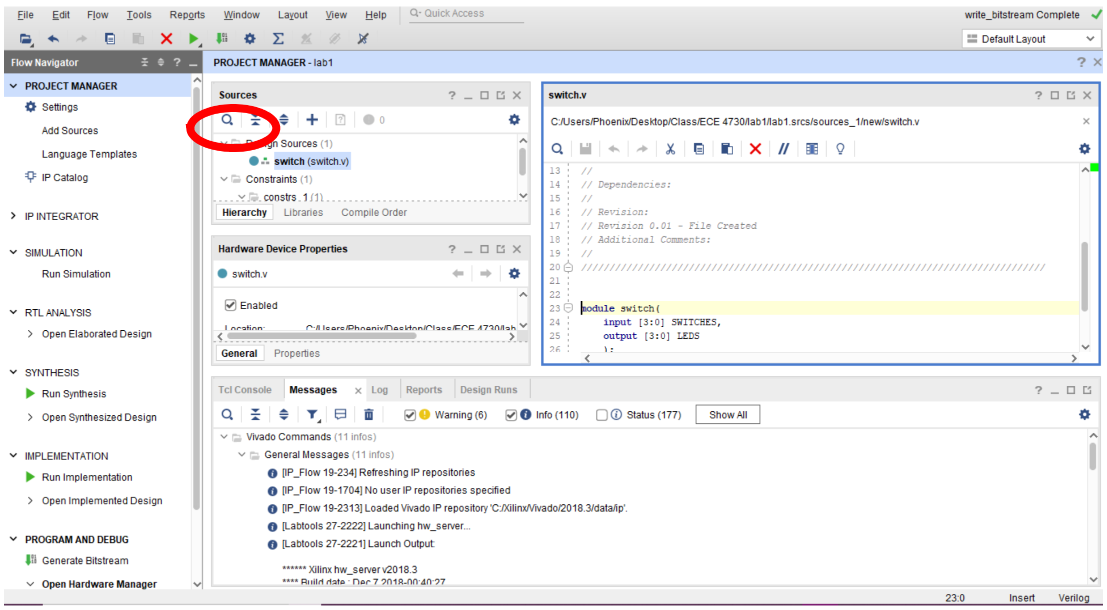
I/O Port Definitions

Port Name	Direction	Bus	MSB	LSB
SWITCHES	input	✓	3	0
LEDS	output	✓	3	0

Figure 3:Define Module

3. At this point, Vivado has created a new project and source file for us to modify. We will now create code to provide the desired functionality (i.e. to turn on LED[i] when Switch[i] is high).

   (a) From the 'Sources' window, open the 'switch.v' file. It will contain a Verilog module with the port declarations described in part 2(b).



C:/Users/Phoenix/Desktop/Class/ECE 4730/lab1/lab1.srcs/sources_1/new/switch.v

```
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module switch(
    input [3:0] SWITCHES,
    output [3:0] LEDS
    );
```

```
Vivado Commands (11 infos)
  General Messages (11 infos)
    [IP_Flow 19-234] Refreshing IP repositories
    [IP_Flow 19-1704] No user IP repositories specified
    [IP_Flow 19-2313] Loaded Vivado IP repository 'C:/Xilinx/Vivado/2018.3/data/ip'.
    [Labtools 27-2222] Launching hw_server...
    [Labtools 27-2221] Launch Output:

      ****** Xilinx hw_server v2018.3
      **** Build date : Dec 7 2018-00:40:27
```

Figure 4:Source Panel Location

   (b) Above 'endmodule', add the following line of code:

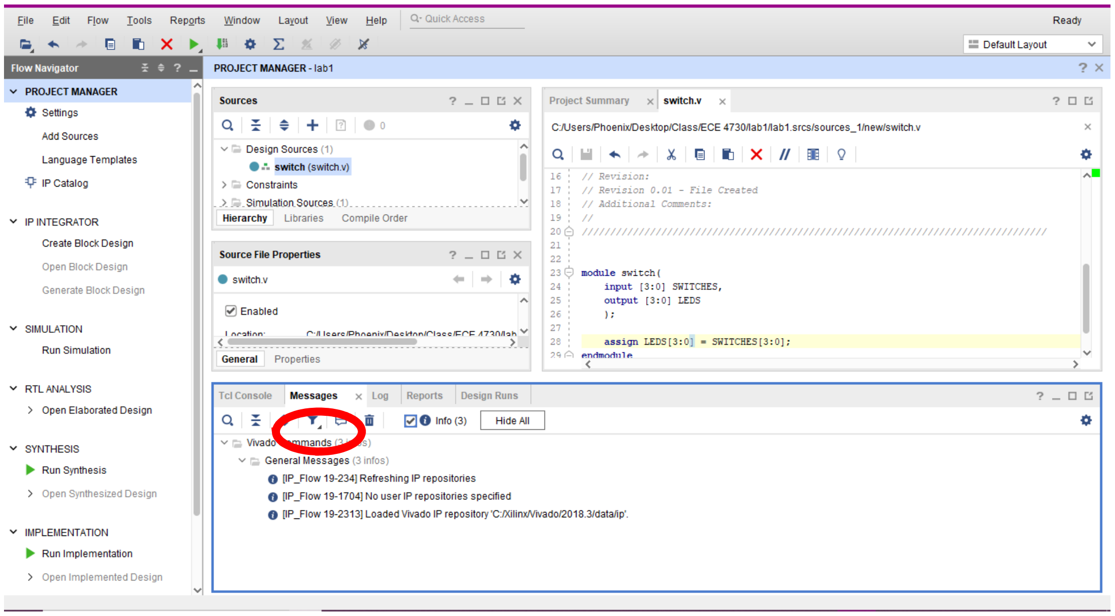
assign LEDS[3:0] = SWITCHES[3:0];
The resulting Verilog module should be as follows:

```
module switch(
   input [3:0] SWITCHES,
   output [3:0] LEDS
   );

   assign LEDS[3:0] = SWITCHES[3:0];

endmodule
```

(c) Click on ‘File’ → ‘Text Editor’ → ‘Save All Files’ to save your changes. Saving the source file will perform a ’Syntax Check’. When you save the file, if you have made any syntax errors in the source file, Vivado will show error messages corresponding to syntax errors in the messages panel. Please clear the errors and save your source file by pressing Ctrl+S or ‘File → Text Editor → Save All Files’.



C:/Users/Phoenix/Desktop/Class/ECE 4730/lab1/lab1.srcs/sources_1/new/switch.v

```
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
/////////////////////////////////////////////////////////////////////////////////


module switch(
    input [3:0] SWITCHES,
    output [3:0] LEDS
    );

    assign LEDS[3:0] = SWITCHES[3:0];
endmodule
```

```
Vivado Commands (3 infos)
  General Messages (3 infos)
    [IP_Flow 19-234] Refreshing IP repositories
    [IP_Flow 19-1704] No user IP repositories specified
    [IP_Flow 19-2313] Loaded Vivado IP repository 'C:/Xilinx/Vivado/2018.3/data/ip'.
```

Figure 5: Location of Message Tab

4. We now need to create the ‘Xilinx Design Constraints(XDC)’ file containing the location of the DIP switches and LEDs on the ZYBO Board. The .xdc file will be used to connect signals described in the Verilog file (LEDS[3:0] and SWITCHES[3:0] in our case) to the pins on the FPGA, which are hardwired to the LEDS and DIP switches on the ZYBO board.

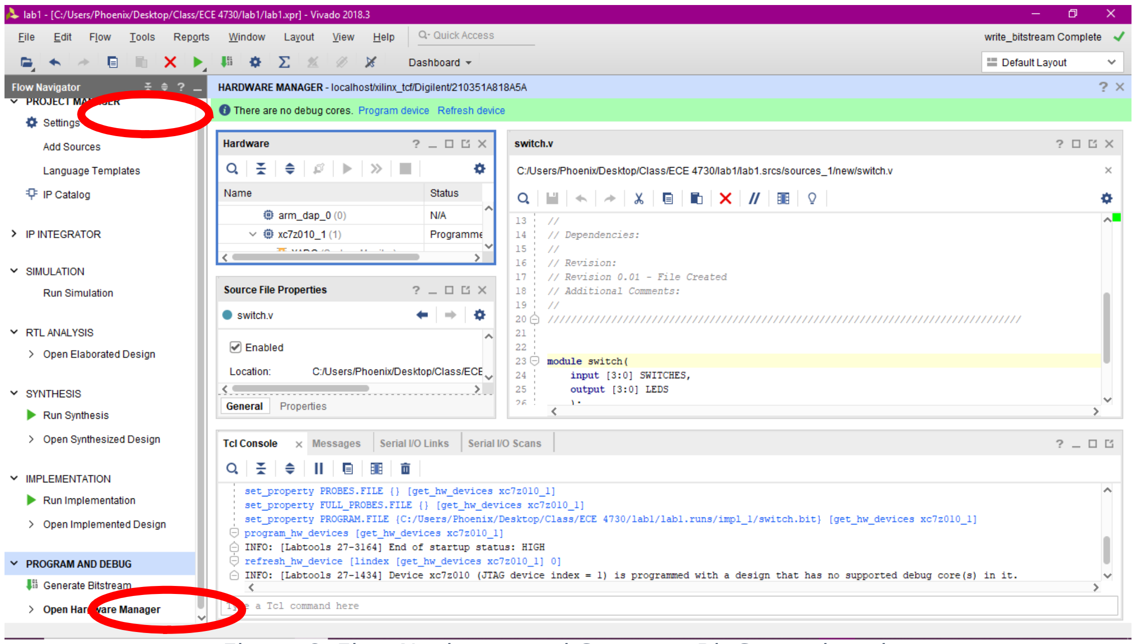
(a) Use your favorite text editor to create a new file called ‘switch.xdc’ in your lab1 project directory and copy the following text into the new file:

```
##Switches
set_property PACKAGE_PIN G15 [get_ports {SWITCHES[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[0]}]
set_property PACKAGE_PIN P15 [get_ports {SWITCHES[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[1]}]
set_property PACKAGE_PIN W13 [get_ports {SWITCHES[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[2]}]
set_property PACKAGE_PIN T16 [get_ports {SWITCHES[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SWITCHES[3]}]
##LEDs
set_property PACKAGE_PIN M14 [get_ports {LEDS[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LEDS[0]}]
set_property PACKAGE_PIN M15 [get_ports {LEDS[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LEDS[1]}]
set_property PACKAGE_PIN G14 [get_ports {LEDS[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LEDS[2]}]
set_property PACKAGE_PIN D18 [get_ports {LEDS[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LEDS[3]}]
```

(b) Note that adding or removing spaces from the above code may result in an 'Implementation Error'. After saving 'switch.xdc', return to the Sources panel in Vivado, right click on the constraints folder and select 'add sources'. Next, the 'Add Sources ' window will open. Select 'Add or create constraints' and click 'Next'. Click on the blue + button and select 'Add files' and navigate to the directory where you saved the constraint file. Select the constraint file and click 'OK'. Click 'Finish' to add the XDC file to your project.

5. At this point, both 'switch.v' and 'switch.xdc' should show up in the 'Sources' window. It is now time to create the hardware configuration for our specific FPGA and download the generated configuration to the ZYBO board.

   (a) Select 'switch.v' in the 'Sources' window. In the 'Flow Navigator' under 'Program and Debug' panel, click on 'Generate Bitstream'(See Figure 6). A warning will appear indicating 'No implementation results available'. Click 'Yes'. Press OK in the 'Launch Runs' window to launch 'Synthesis and Implementation'. This will run all the processes necessary to create a bitstream, which can be downloaded to the FPGA. Running these processes may take several minutes; progress is indicated by the spinning icon and output to the console. You can check the progress in the 'Design Runs' panel located at the bottom of the screen. When a process completes, a ✓ appears next to the appropriate process name. Vivado follows these steps before creating the Bitstream File : synthesize the Verilog, map the result to the FPGA hardware, place the mapped hardware, and route the placed hardware.



```
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
////////////////////////////////////////////////////////////////////////////////


module switch(
    input [3:0] SWITCHES,
    output [3:0] LEDS
```

```
set_property PROBES.FILE {} [get_hw_devices xc7z010_1]
set_property FULL_PROBES.FILE {} [get_hw_devices xc7z010_1]
set_property PROGRAM.FILE {C:/Users/Phoenix/Desktop/Class/ECE 4730/lab1/lab1.runs/impl_1/switch.bit} [get_hw_devices xc7z010_1]
program_hw_devices [get_hw_devices xc7z010_1]
INFO: [Labtools 27-3164] End of startup status: HIGH
refresh_hw_device [lindex [get_hw_devices xc7z010_1] 0]
INFO: [Labtools 27-1434] Device xc7z010 (JTAG device index = 1) is programmed with a design that has no supported debug core(s) in it.
```

Figure 6: Flow Navigator and Generate Bit Steam location

(b) Once the bitstream generation is completed, we need to download the bitstream to the FPGA on the ZYBO board. Turn on the power to the ZYBO board and make sure that the jumper JP5 is set in ‘JTAG’ mode. In the ‘Flow Navigator’ window, under the ‘Program and Debug’ panel click on ‘Open Hardware Manager’. Click on ‘Open Target’ and in the pop up select ‘Open New Target’ which will open the ‘Open New Hardware Target’ window. Click ‘Next’. select ‘Local Server’ in the ‘connect to’ field. Click ‘Next’. Select ‘xilinx tcf’ in Hardware Targets and ‘xc7z010 _1’ in Hardware Devices as in (Figure 7). The ‘arm dap 0’ device is ARM Cortex Processor which is not needed for this lab. Click ‘Next’ and go through the summary and Click ‘Finish’. Click on ‘Program Device’ in ‘Open Hardware Manager’, which is located in ‘Program and Debug’ in the ‘Flow navigator’, and select the FPGA ‘xc7z010 1’. Click ‘Program’ to program the FPGA on the ZYBO board.

(c) At this point, the FPGA should be programmed to function as described in ‘switch.v’. Verify this by toggling the DIP switches 0 through 3 and observe LEDs 0 through 3. Demonstrate your progress to the instructor.

Pulse-Width-Modulated (PWM) LED Guideline

1. Creating a new project

(a) Create a directory (folder) for your ECE 4730 lab1. Spaces or special characters will cause problems (+,”,’,;,’,’,., etc.). You may use _ and -.

(b) Launch Vivado and select “Create Project”.

(c) Select project name and location, select “Create project subdirectory,” then click “Next”.

(d) Select RTL project and leave "Do not specify sources" unchecked. Click "Next”.

(e) In the "Add Sources" window, set the following:

Target language: Verilog
Simulator language: Mixed.
Click on the blue "+" button and select "Create file"; a pop-up window will appear.

(f) In the popup window, select the following:

File type:	Verilog
File name:	switches
File Location:	'<Local to project>'

Then click on 'OK' to create the Verilog source file. Click 'Next'

(g) The 'Add Constraints (optional)' window appears. We'll do that later; click 'Next'.

(h) The 'Default Part' window appears. We can select the hardware from either the 'Parts' or 'Boards' tab. The 'Parts' tab allows you to design for only the FPGA portion, while the 'Boards' tab allows you to design for the FPGA and other built-in hardware portions (such as the Zynq processor). We'll be working with only the FPGA today, so in the parts tab, select:

Category:	All	Package:	clg400	Temperature:	All
Family:	Zynq-7000	Speed:	-1	Static Power:	All Remaining

Select the device with part ID 'xc7z010clg400-1' and click 'Next', then 'Finish'.

(i) A 'Define Module' window appears. This auto-generates the module header for us. Create the following two ports:

Port Name	Direction	Bus	MSB	LSB
SWITCHES	input	yes	3	0
LEDS	output	yes	3	0

2. Coding and Constraints

(a) From the 'Sources' window, open the created Verilog file ('.v' file extension). It will contain a Verilog module with the port declarations described above.

(b) Above 'endmodule', add the following line of code:

Your code to read the value of switch signal and change the LED's brightness.

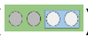
(c) Saving the file performs a 'Syntax Check' (errors will be seen as red text in the 'Messages' tab); save the file by pressing Ctrl + S or selecting 'File → Text Editor → Save All Files'.

3. Bitstream and Programming

(a) Download prj1.xdc from Canvas and place it in the directory created above (Part 1, step 1: ECE4730_Lab1).

(b) Click the ‘+’ button under ‘Sources’. Select ‘Add or create constraints’ and ‘Next’. Then click ‘Add Files’ and navigate to and select prj1.xdc, select ‘OK’ then ‘Finish’. See that prj1.xdc now shows up in the ‘Constraints’ directory in Vivado.

(c) In the left-most section (**Flow Navigator**), click on ‘Generate Bitstream’. Click ‘Yes’ to run synthesis and implementation, then OK in the following window to launch ‘Synthesis and Implementation’. (Continue with step 4. Then verify Synthesis completes in step 5).

(d) Verify the top right jumper (JP5) is set to JTAG mode (). Then plug the FPGA into the computer using the PROG and UART micro-USB port, flip the switch to ON.

(e) The ‘Bitstream Generation Completed’ window will pop up when completed. You may select any of the first three options and click ‘OK’, or simply click ‘Cancel’.

(f) Select ‘Open Hardware manager’ (in the ‘Flow navigator’ under ‘Program and Debug’). Select ‘Open Target’ and in the pop up select ‘Open New Target’. Click ‘Next’.

(g) Select ‘Local Server’, and ‘Next’. In the next window, select ‘xilinx_tcf’ in ‘Hardware Targets’ and the FPGA (xc7z010_1) in Hardware Devices instead of the arm processor. Click ‘Next’ and then ‘Finish’.

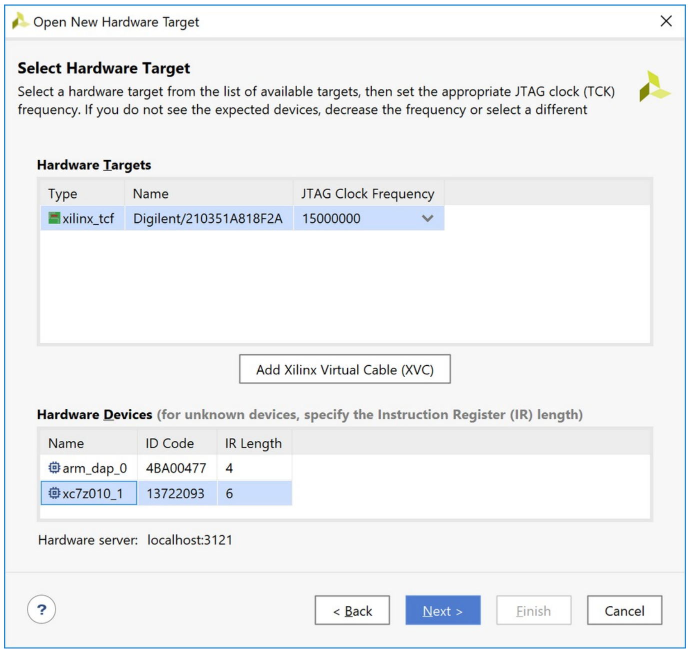
(h) Click ‘Program Device’ under ‘**Open Hardware Manager**’ and select the FPGA. Select ‘Program’.

(i) The FPGA should now be programmed according to the Verilog file. If the switches turn on the LEDS, then you did it!

(j) Good Job!

Implement the Jackpot Game on Your Own

1. Implement a 4-bit counter using the LEDs (You do not need the switches for this exercise). The count value should update approximately every 1 second. Use the BTN0 and BTN1 push buttons on the ZYBO board to control the direction of the count. For example, when the BTN0 button is pressed, the counter should count up. Likewise when the button BTN1 is pressed, the counter should count down. When neither button is pressed, the count should remain the same. Demonstrate this operation to the instructor upon completion.



Select Hardware Target

Select a hardware target from the list of available targets, then set the appropriate JTAG clock (TCK) frequency. If you do not see the expected devices, decrease the frequency or select a different

Hardware Targets

Type	Name	JTAG Clock Frequency
xilinx_tcf	Digilent/210351A818F2A	15000000

Hardware Devices (for unknown devices, specify the Instruction Register (IR) length)

Name	ID Code	IR Length
arm_dap_0	4BA00477	4
xc7z010_1	13722093	6

Hardware server: localhost:3121

Figure 7: Hardware Target

Hints:

- Do not forget to add clock and reset as input pins to your Verilog module.
- Skim through the user manual for the ZYBO board to determine the pin assignments for additional signals. The user manual may be found on the course website.
- After modifying the XDC to include the new ports (and removing unused ports) append the following text to the XDC:

```
#clk_100MHz
set_property PACKAGE_PIN K17 [get_ports clk_100MHz]
set_property IOSTANDARD LVCMOS33 [get_ports clk_100MHz]
create_clock -add -name sys_clk_pin -period 10.00 -waveform {0 5}
get_ports clk_100MHz]
```

Note: The above XDC lines provide Vivado with timing constraints necessary to ensure proper design operation and assume your signal for clock is labeled ‘CLOCK’.

- The L16 pin is the onboard clock with frequency 125MHz. Updates to the LEDs at this rate will not be visible, and thus, the incoming clock must be divided. Think back to your introductory digital logic class to determine how to divide a clock!

2. Design a ‘Jackpot game which works as follows: The LEDs glow in a one-hot fashion, which means that the LEDs are turned on one at a time in a sequential

manner. Get the transition to happen as fast as you can, while you can still make out which LED is on at a given of time. Assign a DIP switch to each of the LEDs. At any point in time, if you turn on the switch corresponding to the glowing LED, you win a Jackpot and all the LEDs start glowing!

Deliverables

1. [0.6 points] Demonstration of working portions of the lab (0.2 points for each part).
2. Submit a lab report with the following items:

   (a) Correct format including an Introduction, Procedure, Results[1], and Conclusion.

      a. [0.4 points] Verilog code for PWM
      b. [0.4 points] Verilog code for Jackpot Game
      c. [0.3 points] XDC file with clock
      d. [0.2 points] lab writeup

   (b) Answer the following questions:

      a. [0.1 points] What does the .xdc file do?
      b. [0.1 points] What does IOSTANDARD LVCMOS33 mean?
      c. [0.1 points] What is JTAG mode? Why are we using it?
      d. [0.1 points] How are the user push-buttons wired on the ZYBO board (i.e. what pins on the FPGA do each of them correspond to and are the signals pulled up or down)? You will have to consult the Master XDC file for this information.
      e. [0.2 points] What is the purpose of an edge detection circuit and how should it have been used in this lab?

[1] Results should include your commented codes.