EECE2323	Digital Logic Design Lab	Lab 0

Full Adder

1 Objective

The purpose of this lab experiment is to get you familiar with the tools and hardware development kit, PYNQ, that you will be using for all EECE2323 labs this semester. The first experiment is a tutorial. By stepping through all the instructions you will learn techniques and tools you will use throughout the course.

2 Conventions

The following conventions are used throughout all lab manuals.

Text Style	Use	Example
boldface	Actions you need to take	**Draw a wire from A to B.**
	Names of programs or tools	Open Vivado Simulator, **XSIM**, and enter...
italics	Menus and menu items	Select *File* → *Quit* to exit.
	Buttons	Click *Finish* to continue.
`monospace`	User interface objects	Enter a name into the `Net Name` field.
	Names of symbols	Insert two `nand2` gates.
	Text you enter into fields	Call your element `xorgate`.
	Items you pick from a list	Choose `Create Block Design` from the list.

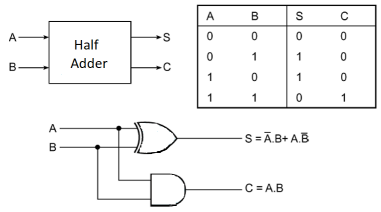
You should read through the entire lab manual so that you are familiar with the concepts and the procedure to be followed. Additional information that is important to understanding the experiment may not be printed in boldface, but you should read it anyway.

3 Overview

In this lab you will implement a FULL ADDER using the tools available in the **Xilinx Vivado** software package. A Full Adder's circuit can be used as a part of many other larger circuits like Ripple Carry Adder, which adds n-bits simultaneously. Full Adders are also used in ALU-Arithmetic Logic Unit.

A full adder has three one-bit numbers as inputs, often written as A, B, C_{in} where A and B are the operands and C_{in} is a carry bit from the previous less-significant stage. It produces two outputs, Sum and $Carry$.

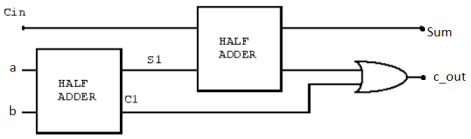
In this lab you will implement a Full Adder circuit using two Half Adders. The truth table and circuit diagram for the Half adder as well as the Full Adder are shown in *Figure 1.*



A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

(a) Half adder circuit and truth table

a	b	C_{in}	Sum	c_out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



(b) Full adder circuit and truth table

Figure 1: Digital Circuits and Truth Tables

This lab will teach you how to use the **Xilinx Vivado** design tools. The steps taken in this lab manual, as well as the processes and procedures that are described, will be repeated throughout the semester. This assignment will introduce you to these practices.

First, you will recreate the schematic shown in *Figure 1* with **Intellectual Property Integrator (IPI)**. Then, you will create a testbench and simulate your design with the Vivado Simulator, **XSIM**. Show the completed simulation to the TAs for the first part of your grade.

Once the simulation is complete, you will use the **Xilinx Vivado** synthesis and implementation tools to translate your design to a bitstream that can be used to program the Zynq's programmable logic (PL) section on the TUL PYNQ Z2. Finally you will use **Vivado Hardware Manager** to download the program to the Zynq's PL section (equivalent to an ARTIX-7 Field Programmable Gate Array (FPGA)) and test it in hardware. Then demonstrate the working hardware to the TAs for the second part of the lab grade.

For this tutorial lab, your grade will consist of the grade for the simulation and the grade for the hardware. There are no prelab assignments and no lab report for this lab. All other labs will have one or more prelab assignments, one or more simulation/hardware grades, and one lab report grade.

4 Full Adder

The first and only part of this lab assignment is to create the Full adder circuit using two half adders. Future labs will often have two or more sections of work to be done.

4.1 Prelab

There is no prelab assignment for this part of the lab.

4.2 Entering Your Design

The **Xilinx Vivado** toolkit is accessed through an interface called **Vivado IDE**. All of the design and implementation tools can be accessed, run, and controlled through **Vivado IDE**.

Launch Vivado IDE by selecting Vivado from the *Start* menu:

Start → All Programs → Xilinx Design Tools → Vivado 2022.1

4.2.1 Starting a New Project

After launching the Vivado tool, click on the Create Project to open the new project wizard. Then choose `lab0` as the project name. From the "Choose Project Location menu" browse to a file on either the Y: (network) or Z: drive.

==========

Note: Please do not save your projects in C: drive as the contents in the drive gets deleted as you log out. Create a folder in the Z: drive with your name, Create and save all your projects in this folder.

==========

Create a subdirectory for your project. *Select* `RTL Project` *but do not specify sources now. Press Next. In the* `Default Part` *tab in the* `Parts` *section, choose xc7z020clg400-1 under the part category. By pressing* `Next` *you will see the project summary. Press* `Finish` *to close the wizard.*

When you have created a new project your **Vivado IDE** window should look like *Figure 2*. Lab manual directions will often refer to the various "panes" of the **Vivado IDE** window, as labeled here.

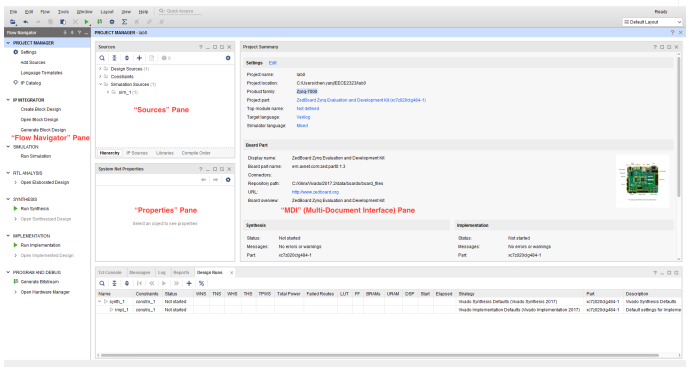


Figure 2: Vivado IDE Main Window

4.2.2 Creating the Block Design

The first step in this lab assignment is to recreate the schematic design for the FULL ADDER using **IPI**. First we must create a new Block Design to add to our project, then we can add the desired IPs in the canvas.

In the Flow pane select *Create Block Design* from the IP Integrator menu. In the `Design Name` field, enter `full_adder`. Do not change the location. Then click the *Ok* button. The "Diagram" canvas appears.

To see the project summary, click on the Window tab of the Vivado IDE, choose Project Summary. You can also open the Project Summary tab by clicking on the $\sum$ icon in the Vivado IDE toolbar.

Now there are two tabs, one for the block design you just created (`full_adder.bd`) and one for the "Project Summary". The Project Summary contains information about the project and allows access to the log files that are generated at different points in the process. Now click on the diagram tab to go back to the Block Design canvas.

You should see an empty canvas in the MDI pane, as shown in *Figure 3*. If not, double-click on the entry for your block design file (`full_adder.bd`) in the Sources pane.

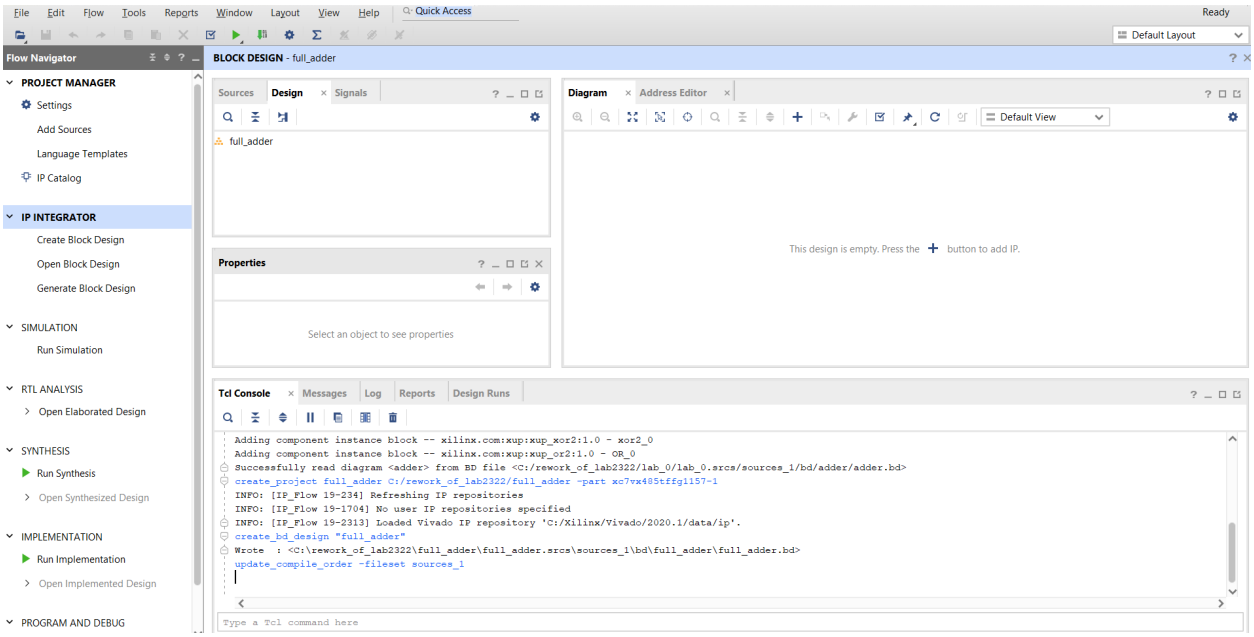


Figure 3: IPI empty canvas

Diagrams can be "floated" into their own window. You may find it easier to work with a Block Design window that takes up the entire screen. To float a document from the MDI pane, right-click on its tab at the top of the MDI pane and select *Float*. Once floated, you can re-dock the diagram in the MDI pane by selecting *Dock Window* from the top right of the window.

4.2.3 Entering a Design with IPI

You will now enter the schematic shown in *Figure 1* . Before that, you should add the Basic Elements repository to your project. **Select `Settings` under the Project Manager section in the Flow Navigator pane,*Figure4*. Then in the IP section, *Figure 5*, select add repository and point to the `basic_elements` folder (.zip version of this folder is available for download on the Canvas course webpage. You need to unzip it.) which includes the Basic Elements repository. To make the basic elements visible in the IP catalog, press `Refresh All`.**

We will start by first creating the Half Adder circuit. You will need one 2-input XOR gate and one 2-input AND gate to implement the Half Adder. For that click on the canvas, then press Ctrl+I to open IP catalog. In the IP catalog these gates are called `XUP 2-input XOR` and `XUP 2-input AND` respectively. Drag and drop them into your block diagram. Click on the canvas or press ESC to close the IP catalog.

Add wires by pointing to the gate ports and dragging the pencil toward the target port, connecting the gates together, as shown in *Figure 6*. Note that the possible connections are shown with a green check mark beside the interface.

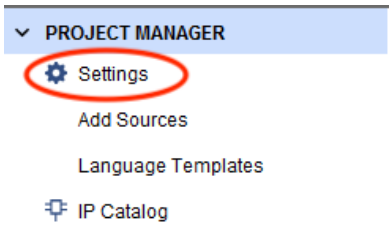


Figure 4: Project Settings (Circled)

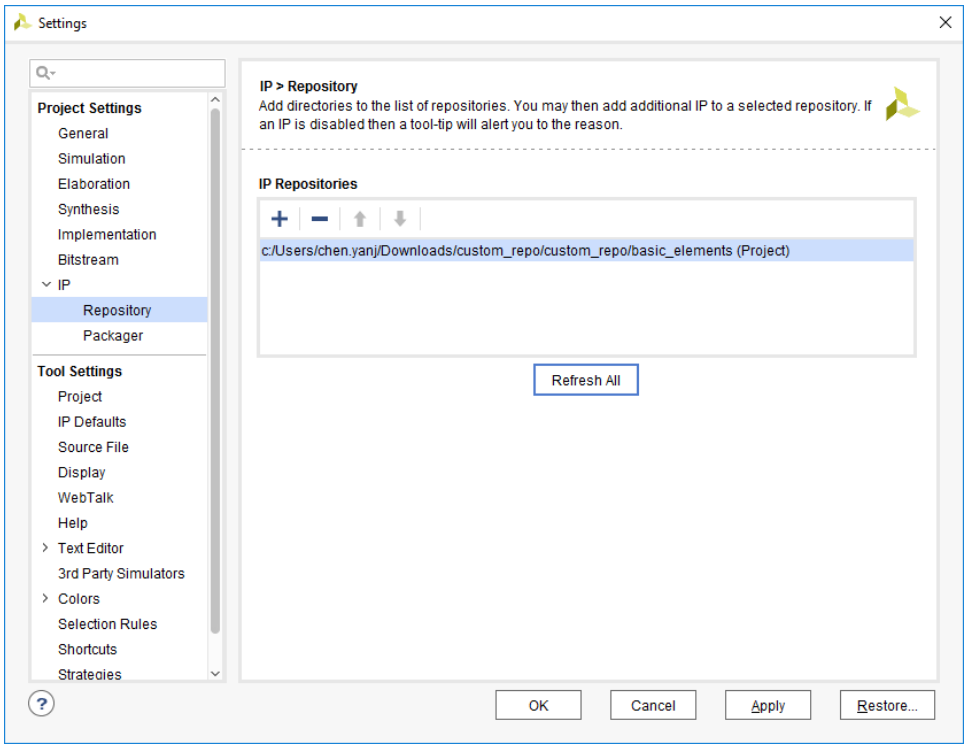


Figure 5: Project Settings, IP Section

Now you will indicate the top level ports (inputs and outputs) of your design, and name the signals. Right click on the ports that you want to make the design's I/Os, then select *Create Port* from the menu or press Ctrl+K after left clicking on the port. Then in the Properties pane name the AND gate as and2_0 and XOR gate as xor2_0 after selecting them. Similarly you can choose the names of the circuit's inputs a and b, and the output $carry$ and sum. You do not need to name the intermediate wires, but you can if you wish. When you are done your block design should look like Figure 6.

Usually a Block Design contains multiple IPs which could be packed together to create a higher level hierarchy. To implement our Full Adder IP we need to create two copies of the Half Adder IP. To do this, select all gates, then right click on of one them and choose `Create Hierarchy` from the menu. After that, change the name of the hierarchy to *HA* as shown in *Figure7*

We now have a *HA* block created. Select the *HA* block, right click and select *copy* and paste to create another *HA* block. Add an `XUP 2-input OR` from the IP catalog.

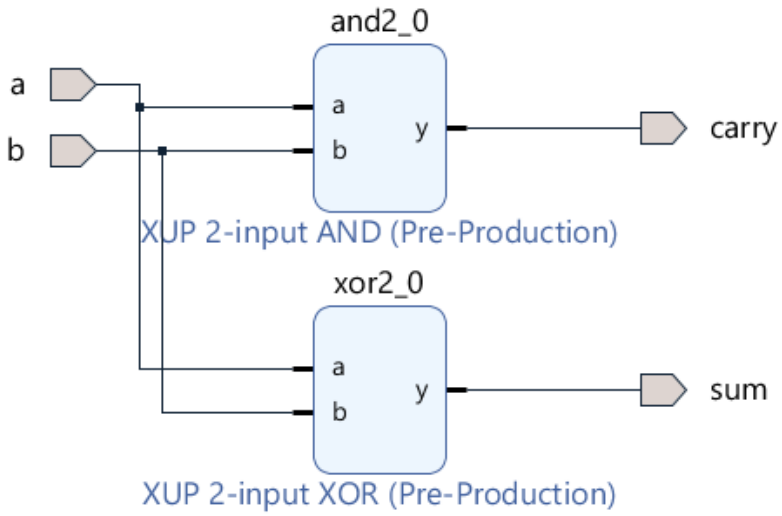


Figure 6: Half Adder Circuit

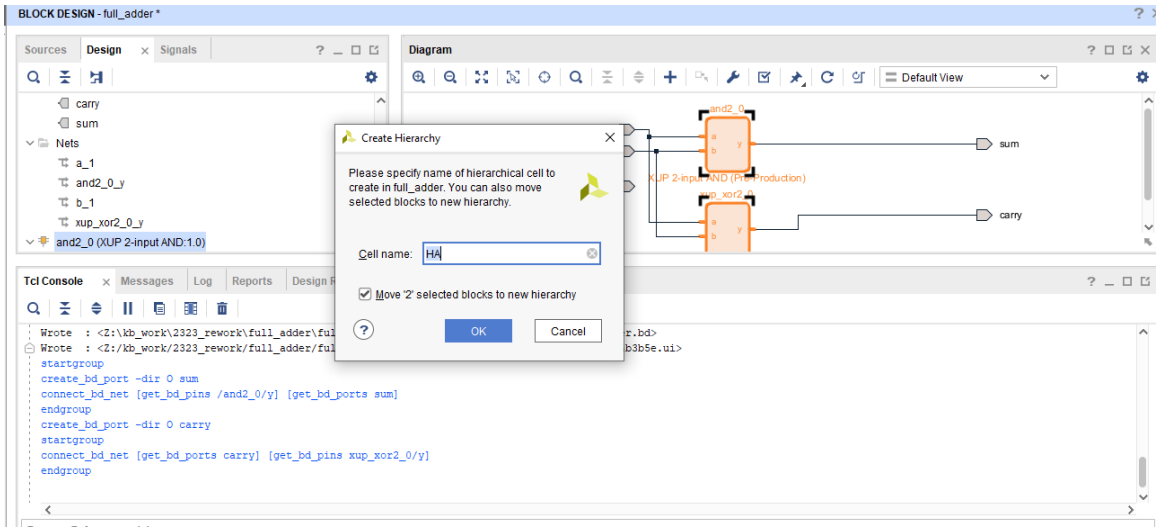


Figure 7: Creation of HA hierarchy

Create the Full adder circuit adding the necessary input and output ports as shown in *Figure8*

At this point you should run **DRC** (Design Rules Check) or **Validate Design** to ensure that your circuit is properly constructed. Click on the validate design button *Figure9* on the toolbar or press F6 after selecting the diagram tab. If any errors or warnings are displayed in the Transcript pane, ask the TA for assistance as the error messages are sometimes difficult to interpret.

We finally create the Full Adder Ip by selecting all blocks present in the design, then right clicking on of one them and choosing `Create Hierarchy` from the list. After that, change the name of the hierarchy to *FA*. The Full Adder IP should look like *Figure 10*. You should rename the carry output to *c_out*.

Once you are finished, save your Block Design by clicking on the Save button on the

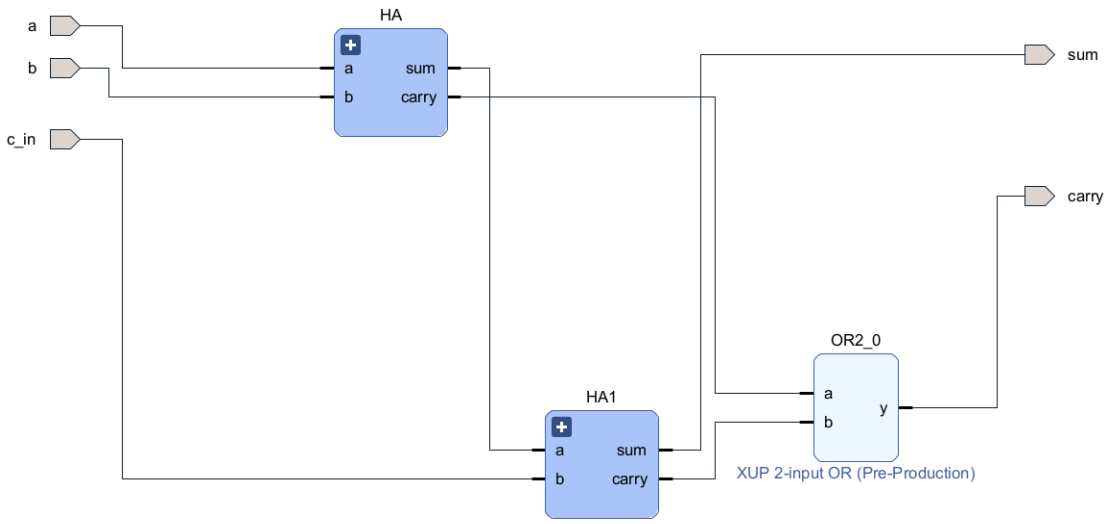


Figure 8: Completed Full Adder Circuit

Figure 9: Validate Design button (circled)

Figure 10: Full Adder completed block

toolbar, pressing Ctrl-S, or selecting *File* → *Save Block Design* from the menu.

We now create the top level HDL code from our full adder block design. To do this we right click on the full_adder.bd from the sources pane and choose `Create HDL Wrapper`. In the menu that shows-up choose `Let Vivado manage wrapper and auto-update`. Click ok. Then, you will see the verilog wrapper in the Sources pane. This is our Full adder top level module. The top level module is identified by the three blocks (adder_wrapper) that show up to the left of the name in sources pane

4.3 Simulating Your Design

Now that you have finished constructing the circuit you must test it. To test designs you use a software simulator which creates a computer simulation of the block design. To do this, you must

Signal	Time (ns) 0	100	200	300	400	500	600	700
a	0	1	0	0	1	1	0	1
b	0	0	1	0	1	0	1	1
c_in	0	0	0	1	0	1	1	1
sum	0	1	1	1	0	0	0	1
c_out	0	0	0	0	1	1	1	1

Table 1: Full Adder Test Vectors

specify the *test vectors*, or set of input values, that will be used to test your circuit. In this section you will use **XSIM** to simulate the circuit's response to those test vectors.

In this lab we will tell you what test vectors you should use. Since this is a very simple circuit with only three inputs, you will test all possible input combinations ($2^3 = 8$ possibilities). In some future labs it will be up to you to choose an appropriate set of vectors that demonstrate that your circuit works correctly.

4.3.1 Creating a Testbench with Vivado

The first step in a computer simulation is to create a testbench. A testbench may include three parts: Stimulus, Unit Under Test (UUT) and Monitor. For our case the testbench is a System Verilog module with a synthetic set of test vectors as the stimulus and the full adder design as the UUT. The Monitor section is integrated in the testbench module.

To create a testbench module click on the `Add Sources` under the Project Manager section in the Flow Navigator pane, and choose `Add or Create Simulation Sources`. In the next page click on `Create File` and create a testbench source file called `fulladder_tb.sv`. Make sure that the `Include all design sources for simulation` option is selected, then click Finish. Use *Figure 11* to write your testbench code in the source file you just created. Find the testbench file in the Sources pane. You will use the test vectors in Table 1 to test your `full_adder`. Please note that you are using the HDL wrapper the you created for the xorgate block design as the UUT for your testbench.

In the testbench code notice how the input values change after every 100ns. This is the way that you will create your own test vectors in future labs. Also, If you look at the testbench code you will see the `` `timescale 1ns \ 1ps `` statement at the top of the source file. It means that any delay (numbers preceding with #) in the code will have a unit of 1ns second and simulation will run with the precision of 1ps.

Once you are finished, save your testbench by clicking the Save button on the toolbar, pressing Ctrl-S, or selecting *File* → *Save* from the menu.

4.3.2 Running a Testbench with XSIM

To run the simulation press `Run Simulation` under the Simulation section in the Flow Navigator pane. Then pick `Behavioral Simulation` from the list. You will see the

```
`timescale 1ns / 1ps

module adder_tb();
logic a;
logic b;
logic c_in;
logic sum;
logic c_out;

full_adder_wrapper UUT(.a(a),.b(b),.c_in(c_in),.sum(sum),.c_out(c_out));

initial begin
     a =1'b0; b=1'b0; c_in=1'b0;
    #100 a=1'b1; b=1'b0; c_in= 1'b0;
    #100 a=1'b0; b=1'b1; c_in= 1'b0;
    #100 a=1'b0; b=1'b0; c_in= 1'b1;
    #100 a=1'b1; b=1'b1; c_in= 1'b0;
    #100 a=1'b1; b=1'b0; c_in= 1'b1;
    #100 a=1'b0; b=1'b1; c_in= 1'b1;
    #100 a=1'b1; b=1'b1; c_in= 1'b1;
   end

endmodule
```

Figure 11: Full Adder Testbench

Behavioral Simulation window after compilation of source files. The first step to run the simulation is to set the simulation time. We want to run the simulation for 800ns, so write 800 in the time box and make sure the unit is set to ns, see *Figure 12*. Then run the simulation until then by first clicking on `Restart` button and then on `Run for ...` button.

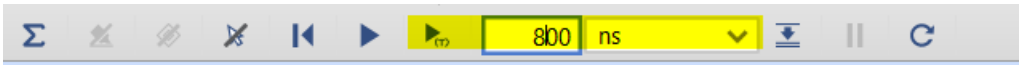


Figure 12: Simulation Toolbar: Restart button, Run all button, Run for button, time box, units list (Highlighted region) and Relaunch Simulation button

You have to compare the results that appear in your **XSIM** waveform with the results shown for the signal *c_out* and *sum* in Table 1. Your waveform should resemble *Figure 13*

If the results are correct, get the TA to check off your simulation. This is the first part of your lab grade.

4.4 Testing in Hardware

The last step is to test your design in hardware. To do this, you need to create a new source file that indicates how the design IOs connect to the TUL PYNQ Z2 hardware resources. In Vivado this

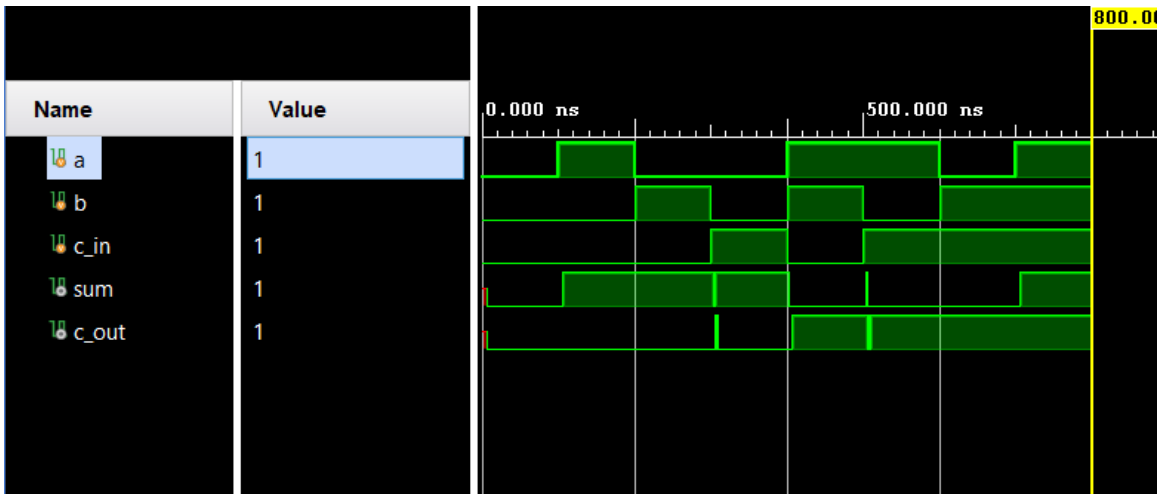


Figure 13: XSIM Waveform for full adder

file is called constraints file or XDC file. The *synthesis and implementation* tools will then create a *bitstream* that can be used to program the FPGA. Finally we will connect the FPGA board to the PC and download the program.

4.4.1 Creating a Constraints File

We will be using the ad-on board in order to observe the outputs and give inputs. The ad-on board contains 8 switches and 6 LED's out of which we will be assigning 3 switches to the inputs and 2 LED's to the outputs. To create a constraint or XDC file, click on `Add Sources` under the Project Manager section in the Flow Navigator pane, then choose `Add or Create Constraints` and by clicking on `Create File`, create a constraint file with name of ``fulladder'', then press `Finish`. The constraint file is in the design sources pane. You need to double click on the name for the file to come up.

```
1  ## Add-on board switches assigned for input
2  set_property -dict {PACKAGE_PIN v6 IOSTANDARD LVCMOS33} [get_ports a]
3  set_property -dict {PACKAGE_PIN y6 IOSTANDARD LVCMOS33} [get_ports b]
4  set_property -dict {PACKAGE_PIN B19 IOSTANDARD LVCMOS33} [get_ports c_in]
5
6  ## Add-on board LED's assigned for output
7  set_property -dict {PACKAGE_PIN B20 IOSTANDARD LVCMOS33} [get_ports sum];
8  set_property -dict {PACKAGE_PIN w8 IOSTANDARD LVCMOS33} [get_ports c_out];
9
10
```

Figure 14: Full Adder XDC file

Now we will connect the `Full Adders` inputs to switches on the ad-on board which we attach to the FPGA, and the outputs to LEDs on the ad-on board. Specifying the `LOC` attribute for the inputs attaches them to pins on the FPGA chip and the ad-on board;

Follow *Figure 14* to write your physical constraints in the XDC file you just created. You only need to copy the lines that are uncommented. The physical constraints are written based on Table 2. Remember to save the constraints file after you have finished.

Port	a	b	c_in	sum	carry
Device	SWA	SWB	SWC	LDA	LDB
LOC	v6	y6	B19	B20	w8

Table 2: LOC Attributes for Full Adder I/Os

Investigate the XDC file syntax and find out what each line means.

4.4.2 Implementing a Design with Vivado Synthesizer

The process of transforming your design specification (i.e. Block Design files) into a *bitstream* that can be used to program the FPGA is called *synthesis*. **Vivado IDE** includes the **Vivado Synthesis** toolkit to perform synthesis on your design. Since synthesis is specific to a particular FPGA, it is critical that your project be designed for the right part. In this case we are using the `xc7z020clg400-1` chip, which you should have set when you created a new project. If this part number does not appear in `Project Settings -> General -> Project Device` under the Project Manager in the Flow Navigator pane, ask a TA to help you fix it before going on.

Click on `Generate Bitstream` under the Program and Debug section in the Flow Navigator pane to run the synthesis process. Vivado IDE will run the Synthesis and Implementation processes automatically. At the top right of your screen there is a scroll bar that shows progress; you can see diagnostic output scroll past in the Transcript pane. If any errors occur they will be marked with a red circle with a cross in middle. Warnings are marked with a yellow triangle with an exclamation point. If you see errors or in your output, ask the TA for help. Some warnings can be ignored, but not all. Processes that complete successfully will be marked with a green circle with a check mark inside.

4.4.3 Programming the FPGA with Hardware Manager

To program the Zynq's PL section you will use the Vivado Hardware Manager. Make sure to plug the micro-USB into the board next to the power cable, as shown in *Figure 15*. After the bitstream is generated turn on the board and open the Hardware Manager either by clicking on the option that will pop-up at the end of bitstream generation or by clicking on `Open Hardware Manager` under the Program and Debug section in the Flow Navigator pane. In the Hardware Manager toolbar click on the `Open target` the choose `Auto Connect` from the list. In the same toolbar click on the `Program device`. Keep the settings unchanged in the window and press `Program`. Now you are ready to test your design in hardware.

4.4.4 Testing your Design

For this design, switches SWA,SWB and SWC which are located in the ad-on board are connected to the inputs a,b and c_in respectively and LED's LDA and LDAB which are also located in the

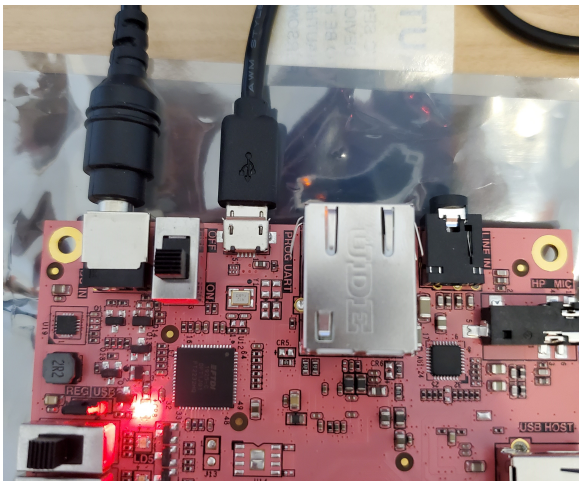


Figure 15: Micro-usb programming cable connected to the top-side of the board

ad-on board are connected to the outputs sum and carry respectively. Using the switches, run the same test vectors that you simulated (see *Table 1* and *Figure 16)*. Verify the working of your Full Adder by testing to see if the LED's turn on accordingly based on different inputs. Example of what you should observe in the board are shown in *Figure 17*

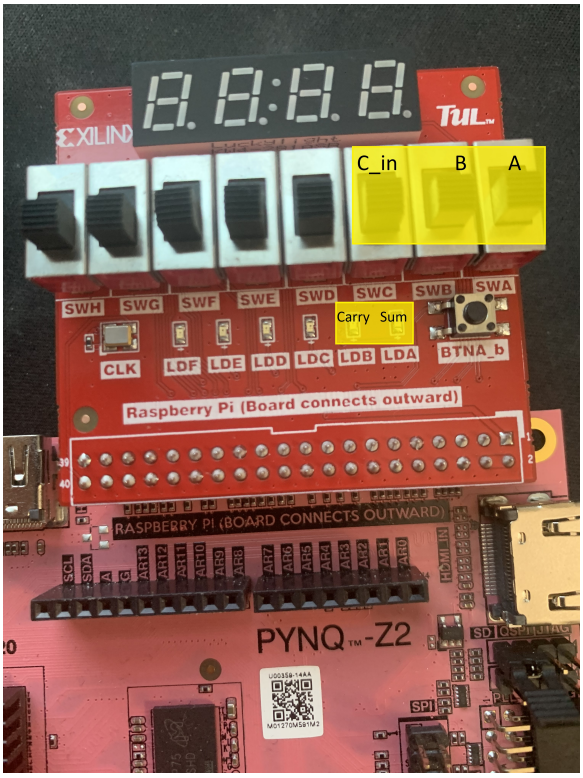
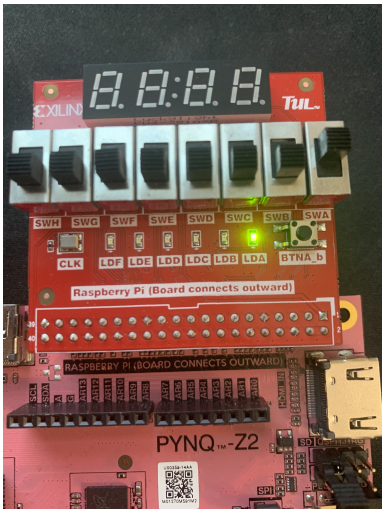
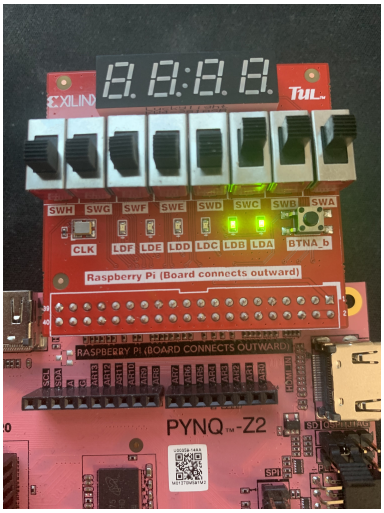


Figure 16: TUL PYNQ Z2 Switches

If your hardware works, demonstrate the tests for the TA. This is the other half of your lab grade.



(a) example 1: a=1,b=0,c_in=0 sum=1,carry=0



(b) example 2: a=1,b=1,c_in=1 sum=1,carry=1

Figure 17: Examples of synthesis outputs