CS223 Laboratory Assignment 2

Full adder, full subtractor and 2-bit adder on FPGA

Lab dates and times:

Section 1:	17.10.2020 Saturday	Even (08:30-10:00), Odd (10:00-11:30)
Section 2:	17.10.2020 Saturday	Even (15:30-17:00), Odd (17:00-18:30)
Section 3:	18.10.2020 Sunday	Odd (15:30-17:00), Even (17:00-18:30)
Section 4:	16.10.2020 Friday	Even (08:30-10:00), Odd (10:00-11:30)
Section 5:	12.10.2020 Monday	Odd (13:30-15:00), Even (15:00-16:30)
Section 6:	13.10.2020 Tuesday	Even (13:30-15:00), Odd (15:00-16:30)

Location: EA Z04 (in the EA building, straight ahead past the elevators)
Each student should attend the time block depending on their student id.

Preliminary Report (30 pts)

In the previous lab, you implemented half adder and subtractor using gates on the bread board. In this lab you will implement very similar circuits, but this time on **FPGA**. Today's lab needs considerable advance preparation. You need to learn how to work with Xilinx's design tool set before attending the lab. In addition, SystemVerilog models and testbenches should be prepared in advance, and assembled neatly into a Preliminary Report with a cover page and pages for the SystemVerilog codes. Each page should have a proper heading. The report should be uploaded on Moodle as a pdf file before the start of the lab. The content of the report will be as follows:

(a) A cover page including: course code, course name and section, the number of the lab, your name-surname, student ID, date.

(b) Circuit schematic for a 2-bit adder made from two full adders shown in Figure 1. (use full adders as black-boxes, you don't need to draw the logic diagram of full adder again).

(c) Behavioral SystemVerilog module for the full adder.

(d) Structural SystemVerilog module for the full adder and a testbench for it.

(e) Structural SystemVerilog module for the full subtractor and a testbench for it (refer to Figure 3).

(f) Structural SystemVerilog module for the 2-bit adder and a testbench for it. Use the full adder module you wrote in part (c).

Note that behavioral model describes the function of a module using Boolean equations and continuous assignment statements; whereas structural modeling refers to using and combining simpler pieces of modules (it is an application of hierarchy). You can refer to Chapter 4 of your textbook and slides of Chapter 4 on Moodle while preparing your modules and testbenches.

Additional pre-lab work:

You should read the following documents (available on Moodle) to be familiar with steps of design flow (Simulation, Synthesis, Implementation, Bitstream Generation, Downloading to FPGA board), using Xilinx **Vivado** tool. You can download, install and practice working with Xilinx Vivado on your own computer with free WebPACK license. Currently, Mac is not supported by Vivado, so if you are a Mac user, you would need to set up a virtual machine in order to use Vivado. Less ideal and a temporary solution would be to use the following site, but you will not be able to generate bitstream and program your FPGA: https://www.edaplayground.com/

- Suggestions for Lab Success.
- Basys 3 Vivado Decoder Tutorial.
- Vivado Tutorial.

- Basys 3 FPGA Board Reference Manual.

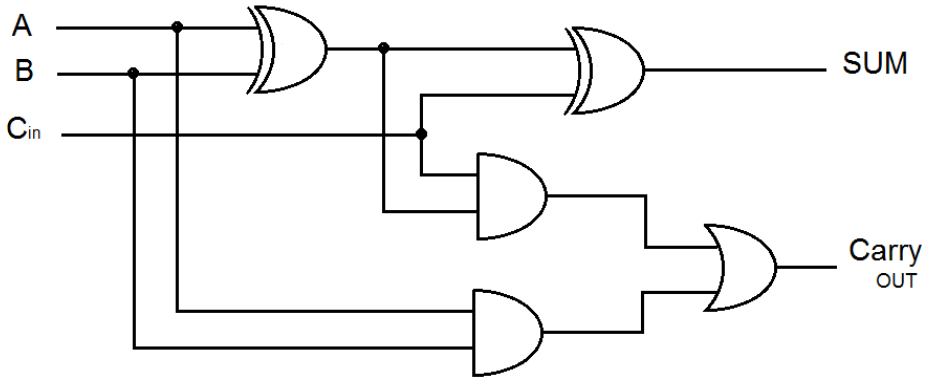

Figure 1: Full Adder

X	Y	Cin	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Figure 2: Truth table of the full adder

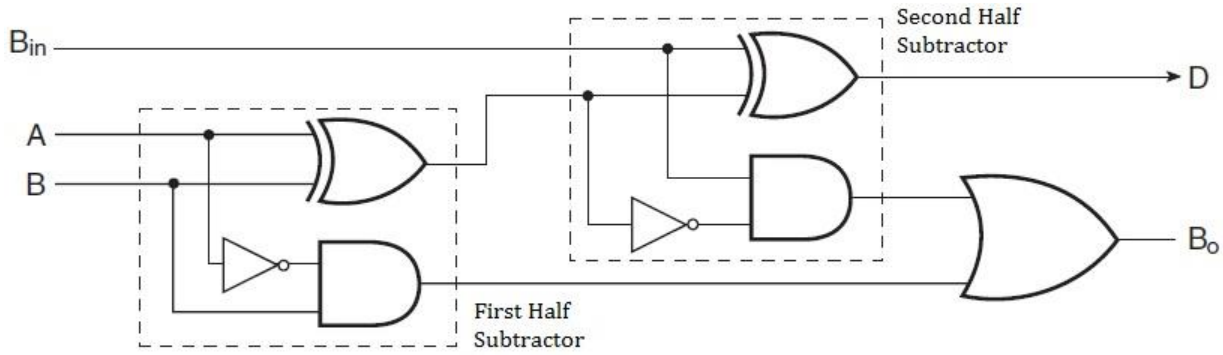

Figure 3: Full Subtractor

A	B	Bin	D	Bo
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Figure 4: Truth table of the full subtractor

Implementation on FPGA (70 pts)

In this step, you implement your modules on FPGA board. You don't need to connect your **Basys 3** board to the Beti board. Working with standalone Basys 3 and having it connected to your computer is enough for this lab. There are some switches and LEDs available on Basys 3 which you can use them.

- *Create a new Xilinx Vivado Project. Use appropriate names for files and folders, keeping the project in a directory where you can find it later and erase it (at the end of lab).*

(a) Simulation: Implement the full adder module in behavioral style (preliminary part-c). Then, using the SystemVerilog testbench code you wrote, verify in simulation that your circuit works correctly.

(b) Simulation: Implement the full adder module in structural style (preliminary part-d). Then, using the SystemVerilog testbench code you wrote, verify in simulation that your circuit works correctly.

(c) Simulation: Implement the full subtractor module in structural style (preliminary part-e). Then, using the SystemVerilog testbench code you wrote, verify in simulation that your circuit works correctly.

(d) Simulation: Implement the 2-bit adder module using two full adders you wrote (preliminary part-f). Then, using the SystemVerilog testbench code you wrote, verify in simulation that your circuit works correctly.

(e) When you are convinced that your codes work correctly, show the simulation results to your TA. Be prepared to answer questions that you may be asked.

(f) Program the FPGA: Now, follow the Xilinx Vivado design flow to synthesize, implement, generate bitstream file, and program all three modules to Basys 3 FPGA board.

(g) Test your design: Using the switches and LEDs (on Basys 3) that you have assigned in constraint file (.xdc), test your designs. When you are convinced that they work correctly, show the physical implementation results to the TA. Be prepared to answer questions that you may be asked.

Submit your code for MOSS similarity testing

Finally, when you are done and before leaving the lab, you need to copy all the SystemVerilog codes you wrote to a txt file named *StudentID_SectionNumber.txt* and upload it to Moodle. If you have multiple files, just copy and paste them in order, one after another inside text file. Even if you didn't finish or didn't get the SystemVerilog part working, you must submit your code to the Moodle for similarity checking. Your codes will be compared against all the other codes in all sections of the class, by the MOSS program, to determine how similar it is (as indication of plagiarism). So be sure that the code you submit is the code that you actually wrote yourself!

Clean Up

(1) Clean up your lab station, and return all the parts, wires, the Beti trainer board, etc. Leave your lab workstation for others the way you would like to find it.

(2) CONGRATULATIONS! You are finished with Lab#2 and are one step closer to becoming a computer engineer.

NOTES

--Advance work on this lab, and all labs, is strongly suggested.

LAB POLICIES

1. There are three computers in each row in the lab. Don't use middle computers, unless you are allowed by lab coordinator.
2. You borrow a lab-board containing the development board, connectors, etc. in the beginning. The lab coordinator takes your signature. When you are done, return it to his/her, otherwise you will be responsible and lose points.
3. Each lab-board has a number. You must always use the same board throughout the semester.
4. You must be in the lab, working on the lab, from the time lab starts until you finish and leave. (bathroom and snack breaks are the exception to this rule). Absence from the lab, at any time, is counted as absence from the whole lab that day.
5. No cell phone usage during lab. Tell friends not to call during the lab hours--you are busy learning how digital circuits work !
6. Internet usage is permitted only to lab-related technical sites. No Facebook, Twitter, email, news, video games, etc--you are busy learning how digital circuits work !
7. You need to be in lab on time and you should upload your preliminary report before the start of lab.