# CS 223 Digital Design Laboratory Assignment 2

# Arithmetic Circuits on FPGA

**Preliminary Report Due: March 11, 2024 09:00**

**Lab Dates and Times**

Section 1: March 14, 2024 Thur. 13:30-17:20 in EA-Z04
Section 2: March 11, 2024 Mon. 13:30-17:20 in EA-Z04

**Location:** EA-Z04 (in the EA building, straight ahead past the elevators)
**Groups:** Each student will do the lab individually. Group size = 1

## Preliminary Work [30]

**Note:** This part must be completed before coming to the lab. Prepare a neatly organized report of your work and submit on Moodle before the start of the lab. Include your code as **plain text, not image!**

In the previous lab, you implemented combinational circuits using gates on the breadboard. In this lab you will implement similar circuits, but this time on the FPGA. Today's lab needs considerable prior preparation. You need to learn how to work with AMD Xilinx's design tool Vivado 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 part should have a proper heading. The contents of the report should be as follows:

**[2]** A cover page including: course code, course name and section, the number of the lab, your name-surname, student ID, date.

**[3]** Circuit schematic for full adder using 2-input XOR, AND and OR gates (refer to Fig. 1a and Table 1a).

**[3]** Circuit schematic for full subtractor using 2-input XOR, AND and OR gates (refer to Fig. 1b and Table 1b).

**[5]** Circuit schematic for a 2-bit adder made from two full adders (as ready black-boxes, you don't need to draw the same things again).

**[3]** Behavioral SystemVerilog module for the full adder and a testbench for it.

**[3]** Structural SystemVerilog module for the full adder and a testbench for it.

**[3]** Behavioral SystemVerilog module for the full subtractor and a testbench for it.

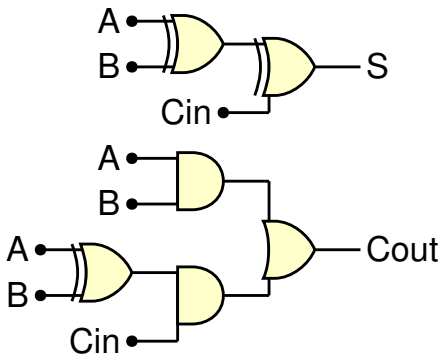
**[3]** Structural SystemVerilog module for the full subtractor and a testbench for it.

**[5]** Structural SystemVerilog module for the 2-bit adder and a testbench for it. Use the structural full adder module you wrote previously.

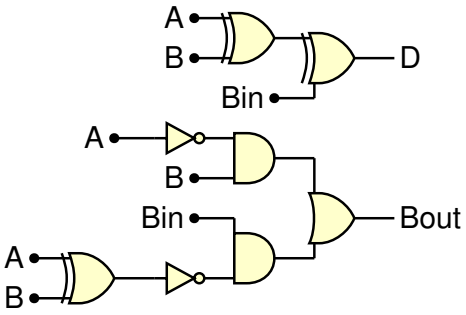
Note that behavioral model describes the function of a module using Boolean equations and continuous assignment statements; whereas structural modeling refers to using and combining simpler pieces of modules (it is an application of hierarchy). You can refer to the slides of Chapter 4 of your textbook while preparing your modules and testbenches. Additionally, remember that circuit schematic and logic diagram are different concepts.

### Additional pre-lab work

You should read the related documents (available on Moodle) to familiarize yourself with steps of digital 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 a free license. It is recommended to install the 2017 version or newer to avoid problems that may arise from bugs present in older versions. Lab computers are installed with the 2019.2 version which you can also use.




(a) Full-adder circuit.




(b) Full-subtractor circuit.

Figure 1: Full-adder and subtractor circuit logic diagrams.

Table 1: Full-adder and subtractor circuit truth tables.

(a) Full-adder circuit truth table.

| $A$ | $B$ | $C_{in}$ | $S$ | $C_{out}$ |
|---|---|---|---|---|
| 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-subtractor circuit truth table.

| $A$ | $B$ | $B_{in}$ | $D$ | $B_{out}$ |
|---|---|---|---|---|
| 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 |

**Important:** It is highly recommended to get started with the lab assignment after you are done with the preliminary work to ensure timely completion of your work in the lab.

# Part 1: Implement Preliminary Designs on FPGA [50]

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.

- ***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 if you are using lab computers).***

**[3]** **Simulation:** Implement the full adder module in behavioral style. Then, using the SystemVerilog testbench code you wrote, verify in simulation that your circuit works correctly.

**[3]** **Simulation:** Implement the full adder module in structural style. Then, using the SystemVerilog testbench code you wrote, verify in simulation that your circuit works correctly.

**[3]** **Simulation:** Implement the full subtractor module in behavioral style. Then, using the SystemVerilog testbench code you wrote, verify in simulation that your circuit works correctly.

**[3]** **Simulation:** Implement the full subtractor module in structural style. Then, using the SystemVerilog testbench code you wrote, verify in simulation that your circuit works correctly.

**[3]** **Simulation:** Implement the 2-bit adder module in structural style using two full adders you already created. Then, using the SystemVerilog testbench code you wrote, verify in simulation that your circuit works correctly.

- 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.

**[7×5]** **Program the FPGA:** Now, follow the Xilinx Vivado design flow to synthesize, implement, generate bitstream file, and program all five modules to Basys 3 FPGA board.

- Test your designs using the switches and LEDs (on Basys 3) that you have assigned in constraint file (`.xdc`). 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.

## Part 2: Lab Multiplier [20]

Suppose that we need a circuit that multiplies a two-bit unsigned number by 3. Let $A = a_1a_0$ denote the number and $P = p_3p_2p_1p_0$ denote the product $P = 3A$. Note that 4 bits are needed to represent the product. A simple approach to design the required circuit is to use two ripple-carry adders to add three copies of the number $A$. The first adder produces $A + A = 2A$. Its result is represented as two sum bits and the carry from the most-significant bit. The second adder produces $2A + A = 3A$. It has to be a three-bit adder to be able to handle the three bits of 2A, which are generated by the first adder. Because $A$ is a two-bit unsigned number and the second adder inputs have to be driven by three bits, the most significant bit of the input corresponding to $A$ of the second adder is connected to a constant zero. This approach is straightforward, but not very efficient.

- Create a new multiply by 3 circuit using **only one** ripple-carry adder.

**[5]** Design a SystemVerilog module.

**[5]** Simulate your design.

**[10]** Implement your design on the FPGA.

- Show your simulation and physical implementation results to the TA when you are convinced that everything works correctly. Be prepared to answer questions that you may be asked.

**Hint:** Think about what happens when you shift the bits of a binary number.

## 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.
- Be sure to read and follow the Policies and other related material for CS223 labs, posted in Moodle.

## 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. If you come to lab later than 20 minutes, you will lose that session completely.
8. When you are done, DO NOT return IC parts into the IC boxes where you've taken them first. Just put them inside your Lab-board box. Lab coordinator will check and return them later.

## Recommendations

When building circuits using IC's and FPGAs in CS223 labs, it is important to follow some simple guidelines to prevent damage to electronic parts or confusion during debugging. By following these rules, you can ensure that your circuit functions correctly and avoid potential problems. Here are some key points to keep in mind:

- Avoid touching IC or FPGA pins directly by your hand. Static electricity from your body can permanently damage them. If you have to touch the pins, first ground yourself by touching a nearby grounded surface.
- The white board which you setup your circuit on it, is called "breadboard". Research online and find out how its pins are connected internally and use this knowledge when building your circuit.
- Postpone connecting power pins ($V_{cc}$ and ground) until the last step. Check all other connections first, then connect the power pins if everything seems correct.
- Use a consistent wire color convention for easier debugging of circuits. For example, always use black or white wires for ground and red wires for $V_{cc}$. This will help you quickly identify any issues that may arise during testing.
- If an LED's light is weak or the IC's package feels hot to touch (you can safely touch the plastic part), there might be a problem with power pin connections, such as a short circuit or connecting $V_{cc}$ wire to ground pin.