CS223 Laboratory Assignment 5

Designing a 4x4 Binary Multiplier

Section 1:	9.4.2018 Monday 08:40-12:25
Section 2:	11.4. 2018 Wednesday 08:40-12:25
Section 3:	10/4.2018 Tuesday 08:40-12:25
Section 4:	12.4.2018 Thursday 08:40-12:25

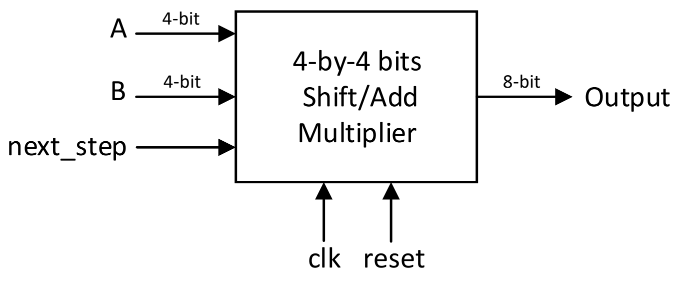
Location: EA Z04 (in the EA building, straight ahead past the elevators)
Groups: Each student will do the lab individually. Group size = 1

Preliminary Report (40 points)

In today's lab you design and implement a 4x4 binary multiplier. These advance designs and SystemVerilog models should be prepared in advance, and assembled neatly into a Preliminary Report with a printed cover page and printed pages for the schematics and SystemVerilog codes. Each page should have a proper heading. The contents of the report should be as follows:

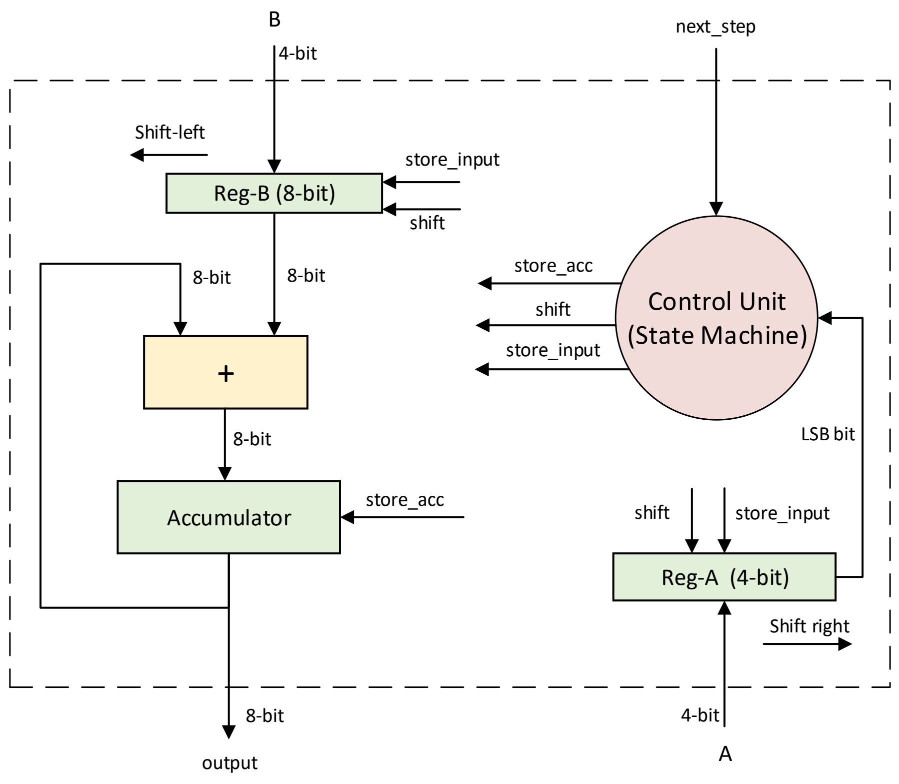
a) A cover page which includes the following: course name and code number, the number of the lab, your name and student ID, date, number of your trainer pack (remember lab policies. You must always use same pack number).
b) Design of the 4x4 binary multiplier with shift&add algorithm: The traditional multiplication algorithm which all we remember from shcool time has following steps:
    1. Check multiplier's digits one by one.
    2. Calculate partial product from multipliying multiplier's current digit by multiplicand.
    3. Shift the partial product, according to multiplier's current digit position.
    4. Add all partial products to calculate final product.

```
Multiplicand:   1 0 1 0     10
Multiplier:     0 1 0 1     5
              ---------
                1 0 1 0  }
              0 0 0 0    }  4 partial products
            1 0 1 0      }
          0 0 0 0        }
        ---------------
          0 1 1 0 0 1 0     50
```



Şekil 1: block diagram of 4-by-4 Multiplier

In this lab, we are going to implement this algorithm for an unsigned 4-by-4 binary multiplier. For being able to check intermediate steps, user should assign a 'next_step' input. In this way, everytime that user presses 'next_step' input, next partial product is calculated and added to the final product (accumulator). When all bits of multiplier are checked, final product register holds the final result.
Data path of the design is given to you in Figure 2. You need to design the control unit by yourself with a FSM. All signals are also given to you according to the Figure. First you need to capture the input A and B into two registers, Reg-A and Reg-B. Note that size of Reg-B is double of size of B. After that, in four cosecutive steps, system waits for user to press 'next_step' button and then, Reg-A is shifted to the right and at the same time, Reg-B is shifted to the left. In this way, the next digit of input A to check is always on the LSB bit location of Reg-A. Shifting the Reg-B to left implements the partial product shifting. After each shifting, if Reg-A's LSB is one, a command is sent to perform the addition. Otherwise, addition is ignored. You ignore the adder's carry-out bit, as it will be always '0' for any inputs. Everytime for starting next step, controller should wait for 'next_step' to change to '0' and then to '1' again (i.e. user shoud release the button and press it again).
Draw the controller's state machine in your report.



Şekil 2: Control unit and Data-path of 4-by-4 multiplier

c) Write the SystemVerilog code for the full design of multiplier Module. Prepare a testbench to your circuit.

The Preliminary Report will be turned in at the <u>start</u> of lab. You may need a copy of your designs and SystemVerilog programs with you in the lab to refer to or possibly correct and change it.

Implementation on FPGA (60 points)

a) Create a Vivado project and enter the SystemVerilog module into Vivado you prepared in preliminary work.

b) Simulate the multiplier for a pair of fixed input binary numbers; A=1001, B=1101; using the testbench code you prepared in preliminary. Test your circuit, if correct show it to your TA. Your simulation should clearly show the intermediate shifting and adding steps, so your TA can confirm them.

c) Implement your code on FPGA (you do not need Beti board in this lab). The 4-bit inputs, A and B, should be applied using the switches on BASYS3 board. The 8-bit output should be connected to two digits of 7-segment module as a 2-digit hexadecimal number (use SystemVerilog code for 7-segment module from lab4). Assign a pushbutton (not switch) on BASYS3 to 'next_step' input pin and another one to system reset.
Note: pushbuttons on BASYS3 give you a clean pulse while you keep them pressed without any vibration. Then, you do not need to implement any additional module like debouncer for them. Therefore, for 'next_step' and reset inputs, use pushbuttons.

d) When you are ready, show your implementation on the BASYS3 board to TA. After resetting, he/she will give arbitrary inputs, and then press 'next_step' pushbutton four times to see the result on 7-segment display.

Submit your code for MOSS similarity testing

Finally, when you are done and before leaving the lab, you need to upload the file *StudentID_SVerilog.txt* created in the Implementation with FPGA part. Be sure that the file contains exactly and only the codes which are specifically detailed above. If you have multiple files, just copy and paste them in order, one after another inside text file. Check the specifications! Even if you didn't finish, or didn't get the SystemVerilog part working, you must submit your code to the Unilica Assignment 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 an indication of plagiarism). So be sure that the code you submit is code that you actually wrote yourself! All students must upload their code to the 'Unilica>Assignment' specific for your section. Check submission time and don't miss it before leaving the lab. After taking a backup of your work, don't forget to delete it from computer. Because students of other sections will work with your system too.

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 this Lab 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 for CS223 labs, posted in Unilica.

LAB POLICIES

1. There are three computers in each row in the lab. <u>Don't use middle computers</u>, 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 <u>must</u> 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, <u>DO NOT</u> 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.