CS223 Laboratory Assignment 4
Multifunction Register

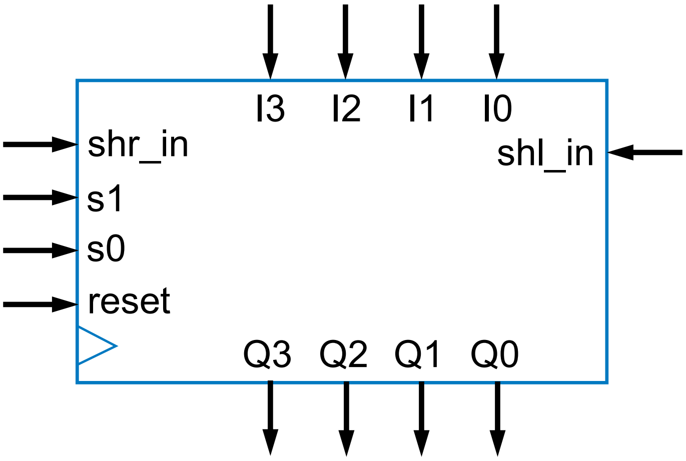
Lab dates and times:

Section 1,2,3,4: 23.04.2020 Thursday 08:40-12:25

Location: EA-Z04 (Zoom Meeting)
Groups: Each student will do the lab individually. Group size = 1

Lab Assignment

In this lab, you are going to design a 4-bit shift register with parallel load by using synchronously resettable D flip-flops. This register will perform a variety of operations and the desired operation will be achieved by setting the control inputs of the register (*You will design a multifunction register capable of Parallel Load, Shift Right, and Shift Left operations*). The block symbol showing the component's inputs and outputs is shown below. The operation table that defines the desired operation for each combination of control inputs is also given below.



reset	s1	s0	Operation
1	X	X	Clear (load all 0's)
0	0	0	Maintain present value
0	0	1	Parallel load
0	1	0	Shift right
0	1	1	Shift left

Multifunction Register:

In this experiment four resettable D flip-flops with inputs $\boldsymbol{D_3\ D_2\ D_1\ D_0}$ and outputs $\boldsymbol{Q_3\ Q_2\ Q_1\ Q_0}$ will be used. All flip-flops will be loaded on every clock cycle.

- `Reset` operation will clear the outputs of all flip-flops on the rising edge of the clock ($\boldsymbol{Q_{3:0} = 0000}$). Reset input has priority over the control inputs (`s1s0`).

- When reset=0, if s1s0=00 and the clock signal rises, each flip-flop stores its own value and the register will retain its present content ($\boldsymbol{Q_3 = Q_3},\ \boldsymbol{Q_2 = Q_2},\ \boldsymbol{Q_1 = Q_1},\ \boldsymbol{Q_0 = Q_0}$).
- When reset=0, if s1s0=01 and the clock signal rises, the register will perform a parallel load operation. Parallel load operation causes the register to get loaded with the external data inputs $\boldsymbol{I_3\ I_2\ I_1\ I_0}$ on the next rising edge of the clock ($\boldsymbol{Q_3 = I_3},\ \boldsymbol{Q_2 = I_2},\ \boldsymbol{Q_1 = I_1},\ \boldsymbol{Q_0 = I_0}$).
- When reset=0, if s1s0=10 and the clock signal rises, the register will perform a shift right operation. Shift right operation causes a right shift on the rising edge of the clock and shr_in input is shifted into the leftmost bit (msb) of the register ($\boldsymbol{shr\_in \rightarrow Q_3},\ \boldsymbol{Q_3 \rightarrow Q_2},\ \boldsymbol{Q_2 \rightarrow Q_1},\ \boldsymbol{Q_1 \rightarrow Q_0}$).
- When reset=0, if s1s0=11 and the clock signal rises, the register will perform a shift left operation. Shift left operation causes a left shift on the rising edge of the clock and shl_in input is shifted into the rightmost bit (lsb) of the register ($\boldsymbol{Q_3 \leftarrow Q_2},\ \boldsymbol{Q_2 \leftarrow Q_1},\ \boldsymbol{Q_1 \leftarrow Q_0},\ \boldsymbol{Q_0 \leftarrow shl\_in}$).

Prepare circuit schematics, SystemVerilog models and testbenches to be included in a Lab Report that has a cover page and pages for the schematics and SystemVerilog codes. You can ask your questions about the lab assignment to TA's during Zoom Meeting. The **Lab Report** must be uploaded to Unilica after 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) Write a SystemVerilog module for synchronously resettable D flip-flop.

(c) Draw a circuit schematic (block diagram) for the internal design of the multifunction register by using 4:1 multiplexers and synchronously resettable D flip-flops.

(d) Write a *Structural* SystemVerilog module for the multifunction register you designed in part (c) and a testbench for it.

Note that structural modeling refers to using and combining simpler components to create more sophisticated building blocks (it is an application of hierarchy). You can refer to the slides of Chapter 4 of your textbook in Unilica while preparing your SystemVerilog modules and testbenches.

Additional pre-lab work:

You should read the following documents (available on Unilica) 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.

- Suggestions for Lab Success.
- Basys 3 Vivado Decoder Tutorial.
- Vivado Tutorial.
- Basys 3 FPGA Board Reference Manual.

Implementation on FPGA

In this step, you will 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 to test your designs.

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

(a) Simulation: Using the SystemVerilog module for the multifunction register and testbench code you wrote in preliminary part (d), verify in simulation that your circuit works correctly.

(b) Program the FPGA: Now, follow the Xilinx Vivado design flow to synthesize, implement, generate bitstream file, and download your multifunction register to Basys 3 FPGA board.

(c) Test your design: Using the switches and LEDs (on Basys 3) that you have assigned in constraint file (.xdc), test your multifunction register. When you are convinced that it works correctly, upload your files to Unilica.

Submit your code for MOSS similarity testing

Finally, when you are done and before leaving the lab, you need to upload the file *StudentID_SectionNumber.txt* created in the Implementation on 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 the code that you actually wrote yourself! All students must upload their code to the 'Unilica Assignment' specific for their sections. Check submission time and don't miss it before leaving the lab.