CEG3155: Lab 5

Chapter 8 (Using Finite State Machines as Controllers)

Circuit description

In this chapter, we will add some modifications to the simple calculator with registers designed in chapter 6. The aim is to reduce the number of inputs used for operands (A and B) and have FSM control the operation sequence of the calculator. Therefore, we'll add an 8-bit register to store one operand before the functional blocks and the finite state controller to sequentially control the calculator as described below.

Description of the new modifications to be added are as follows:

- 8-bit register connected to the input A (SW(7 downto 0).
- Enter KEY(1) to load the SW(7 downto 0) to the 8-bit register.
- B input will be directly taken from SW(7 downto 0).
- A FSM controller which will be discussed below.
- Operation selection by SW(17 downto 16).

Our end design should function in the following steps:

1. Selecting the operation using SW(17 downto 16) .
2. Provide operand A with SW(7 downto 0).
3. Press Enter KEY(1). (A will be loaded to the 8-bit register from switches).
4. For add, xor and or provide the second operand using SW(7 downto 0).
5. Press Enter KEY(1) again.
6. Result will be presented on the 7-segments.

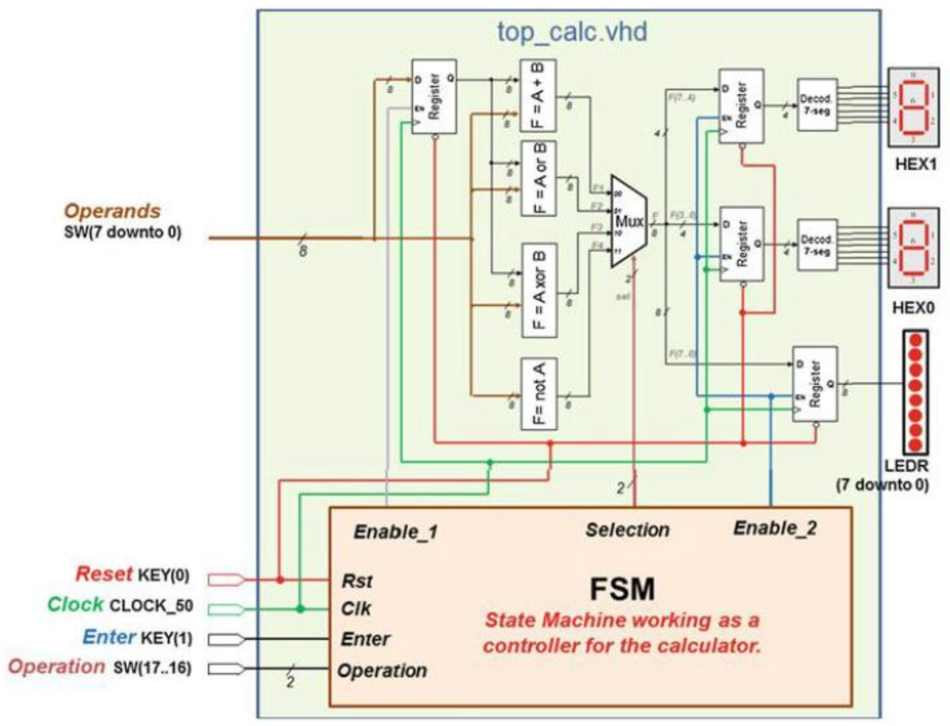
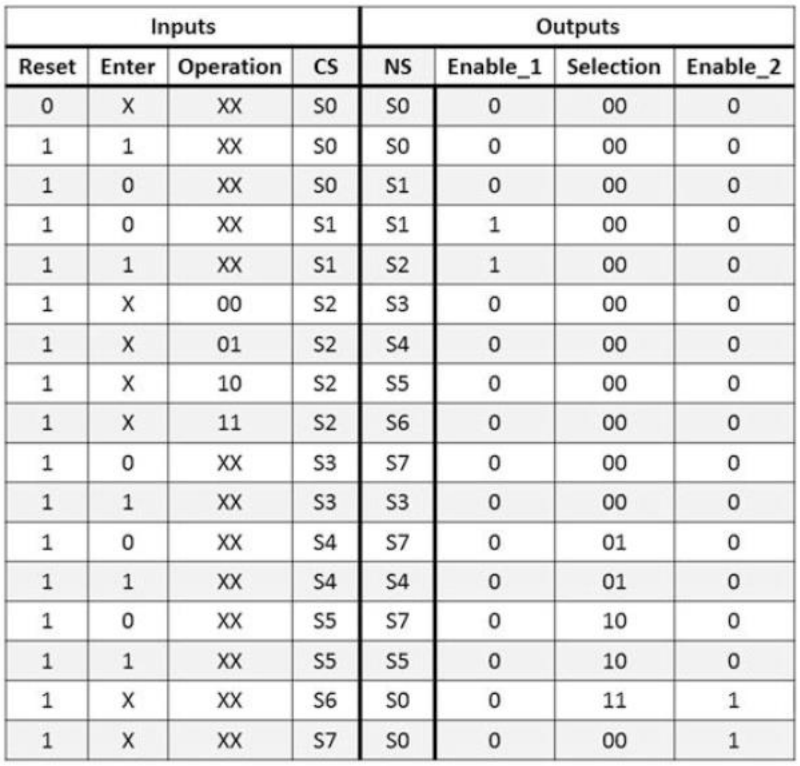


Figure 1. block diagram for the final design.

FSM Controller Design:

To achieve the functional steps explained above we are going to implement the FSM described by the following state transition table (Table 1) in the FSMctrl.vhd file. FSM changes its states S0, S1, S2, S3, S4, S5, S6, S7 according to the inputs Reset (KEY(0)), Enter (KEY(1)) and Operation (SW(17 downto 16)) and gives outputs Enable_1 (connected to the input register), Selection (connected to the mux) and Enable_2 (connected to output registers).



Inputs				Outputs			
Reset	Enter	Operation	CS	NS	Enable_1	Selection	Enable_2
0	X	XX	S0	S0	0	00	0
1	1	XX	S0	S0	0	00	0
1	0	XX	S0	S1	0	00	0
1	0	XX	S1	S1	1	00	0
1	1	XX	S1	S2	1	00	0
1	X	00	S2	S3	0	00	0
1	X	01	S2	S4	0	00	0
1	X	10	S2	S5	0	00	0
1	X	11	S2	S6	0	00	0
1	0	XX	S3	S7	0	00	0
1	1	XX	S3	S3	0	00	0
1	0	XX	S4	S7	0	01	0
1	1	XX	S4	S4	0	01	0
1	0	XX	S5	S7	0	10	0
1	1	XX	S5	S5	0	10	0
1	X	XX	S6	S0	0	11	1
1	X	XX	S7	S0	0	00	1

Table 1: state transition for the FSM controller

Demonstration on FPGA using Quartus:

In this part, we will run the circuit on FPGA.

1. Download the codes from Brightspace under “Lab5”.
2. Unzip the file.
3. Open Quartus.
4. Create a new project in Quartus: Name it “top_calc”.
    - Create the project in the folder “Ch8_Codes”.
    - FPGA Family: Cyclone IV E
    - FPGA number: EP4CE115F29C7
5. From Quartus project menu, add all the files under Ch8_Codes to your project (This should be done using Quartus).
6. Import the pin assignment file “DE2_PinAssignment.qsf”.
    - Assignments -> Import assignments -> Browse to select the file.
7. Complete the FSMctrl.vhd to have the state transition and output functionality described in the Table 1.

8. Complete the top_calc.vhd to wire the components, inputs, outputs and the internal signals as illustrated on Figure 1.
9. Compile
10. Program the FPGA.
11. Test different case on the board and show your work to the TA.

Chapter 9 (Adding multiplication/division by 2 using a shift register)

In this section we will replace XOR and NOT (C3 and C4) components in our calculator with two components to divide and multiply the input operand. we will be using the shift registers to achieve these operations. An illustration of the operation of a shift register is given in Figure 2. Figure 4 shows the whole circuit, where the modified blocks are shown in red.

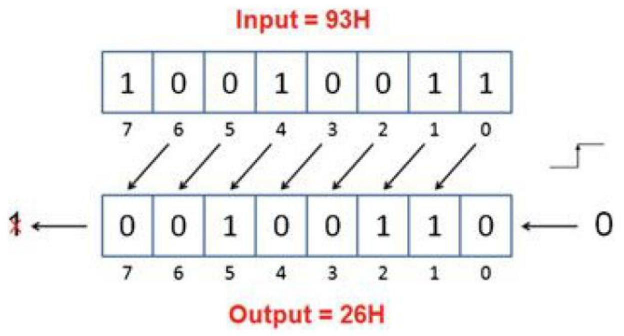


figure 2: functionality of a shift register

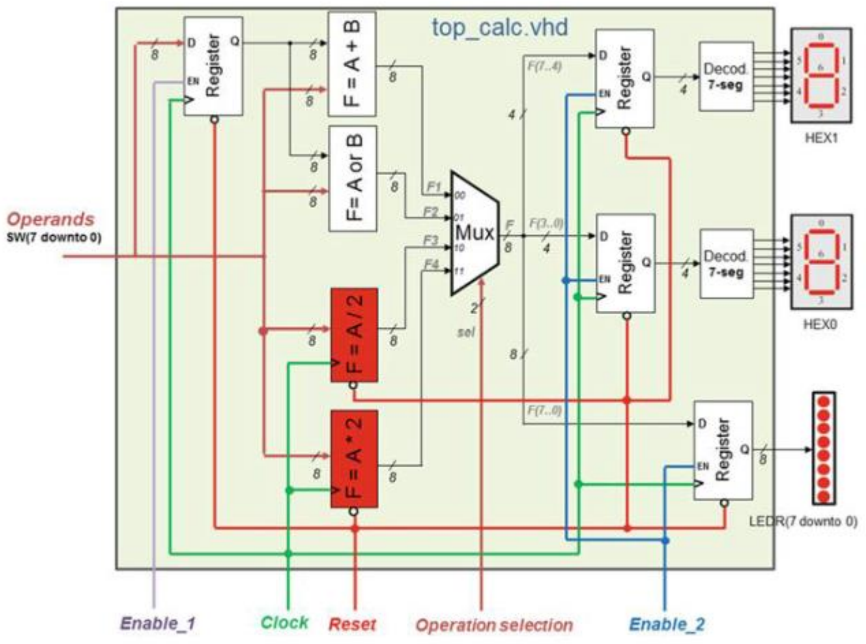


Figure 4: modified calculator with multiplication operations.

Do the following modifications in the codes provided in "Ch9_Codes" folder:

1. Replace the XOR and NOT (c3 and c4) components of the calculator with shift registers on the top module.
2. Use the component shiftreg and shiftregR implemented in shiftreg.vhd and shiftregR.vhd which is given along with the other codes.
3. Wire the circuit appropriately in top_calc.vhd to accommodate for both left shift and right shift with the shiftreg components referring to Figure 4.
4. Complete the FSM in the file "FSMctrl.vhd" as you did in the previous chapter.
5. Demonstrate on the board and show the work to the TA.