ECE 272 Lab #4

Counters

Hyunjae Kim

07/28/2021

Matthew Shuman

1. Introduction

The general purpose of this lab is to understand the finite state machine (FSM), and applicate the FSM with materials in previous lab sections. By using adders from the section 2 and D Flip-Flop, it is required to implement the counter. Moreover, the output of the counter will be indicated by seven segment displays from the last lab section.

The design of the counter with seven segment displays will be implemented by using Quartus software. Before starting the design, understanding about FSM, combinational logic, and sequential logic is crucial.

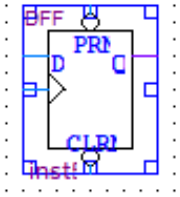
In the process of design, using buses will reduce the complexity of the design and consuming time. Furthermore, checking the input bits of seven segment displays and the number of bits of the buses are vital aspect of implementing the design. The feature of the hardware, DE10-Lite, has to be considered when it is implementing the design. The reason is that there are some pins which are active low.

After implementing the counter with seven segment displays, the implemented design will be verified by checking the actual outputs from the hardware, DE10-Lite. and the simulation via ModelSim software. By comparing the expected result and the actual result, the verification by hardware will be held, and by comparing the actual result of hardware and the result of simulation, the verification by simulation will be held.

2. Design

1) 4-bit Register

In Quartus software, one-bit D Flip-Flop is provided.

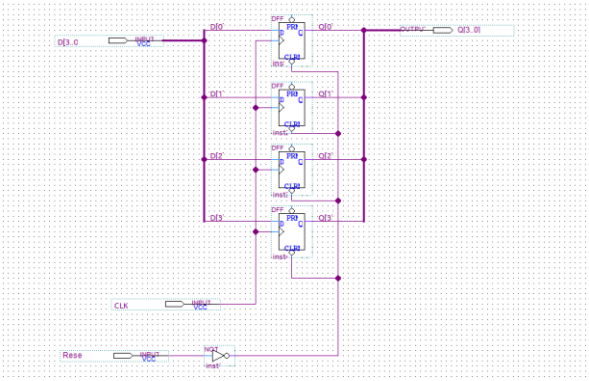


The CLRN acts as Reset, and it is active low. By connecting this one-bit D Flip-Flop, 4-bit register will be implemented. Moreover, this Flip-Flop has a logic feature like below. When the clock is rising edge, the output is same as the input D value, and when the clock is falling edge, the output is equal to the previous output which is before the falling edge. Thus, D Flip-Flop has memory.

D	Clk	Q
0	0 -> 1	0
0	1 -> 0	Unchanged
1	0 -> 1	1
1	1 -> 0	Unchanged

Table 4.1(D flip-flop Truth Table)

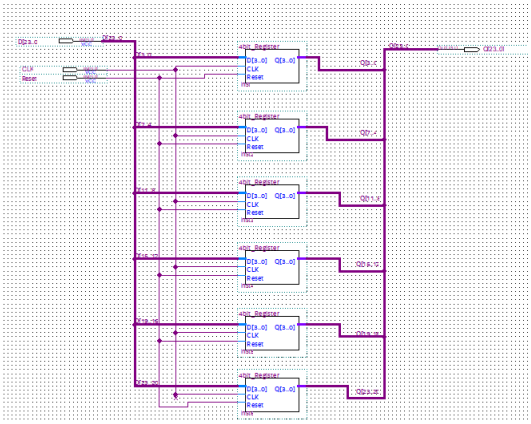
By using this feature, 4-bit register will be implemented by using D Flip-Flops.



In this design, four D Flip-Flops are connected. The input D values are connected by using bus. Because CLRN is active low, the Reset will be connected to CLRN with NOT gate, and the clock will be connected to the clock input in the D Flip-Flop.

2) 24-bit Register

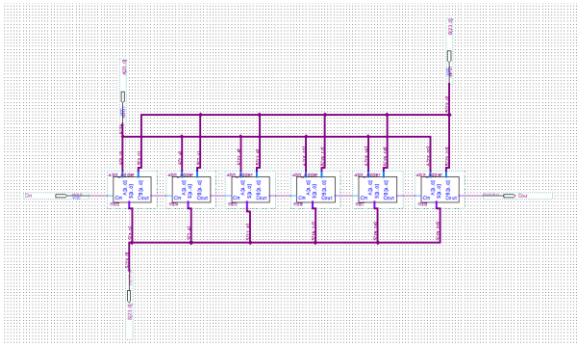
In order to reduce the labor in the design process, 24-bit register will be implemented by connecting six 4-bit registers. The 24 bit of D inputs and outputs are connected to six 4-bit registers by using buses.



The clock and reset will be connected to the entire 4 -bit registers. In 4-bit register, reset is already connected to CLRN with NOT gate so that it is okay to connect the reset to the entire 4-bit register.
Thus, 24-bit inputs will be stored depending on the state of the clock input.

3) 24-bit Adder

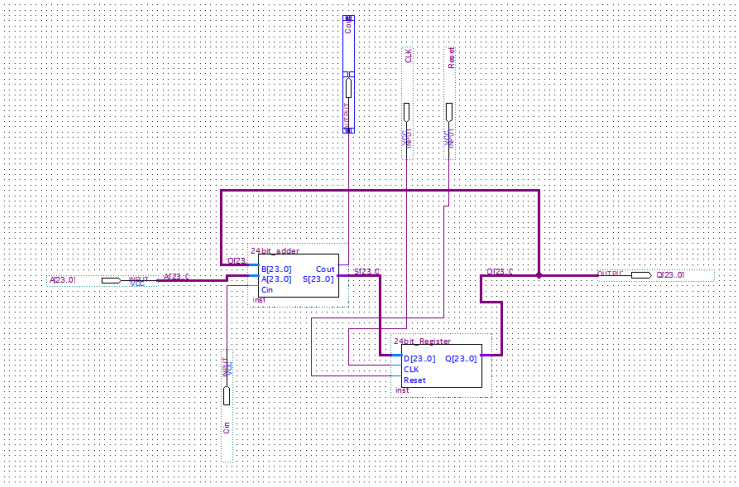
By using 4-bit adders from the section 2 lab, 24-bit adders are implemented.



It is similar to making 4-bit adders. By connecting six 4-bit adders, 24-bit adder is implemented.

4) 24-bit Counter

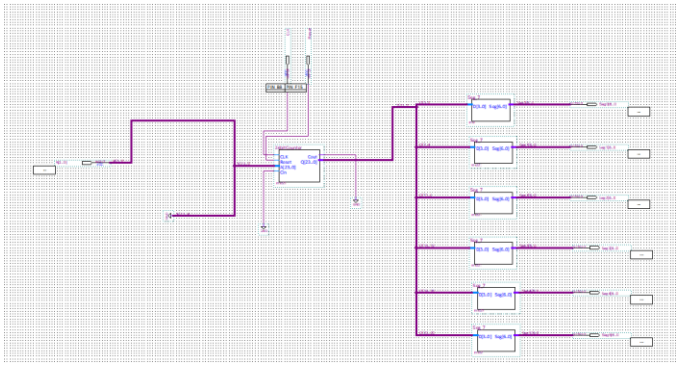
To implement 24-bit counter, 24-bit adder and 24-bit register will be applicated. By connecting the output of 24-bit register to one of the inputs of 24-bit adder, and connecting the output of 24-bit adder to the D input of 24-bit register, 24-bit counter is implemented.



To be specific, because of the memory of 24-bit register, the output of 24-bit register is stored if the clock is falling edge. The stored output goes to the input of 24-bit adder. If the clock is rising edge, the D input and the output of 24-bit register are renewed which value is sum of the previous output of 24-bit register and the A input of 24-bit adder because the output of 24-bit adder is connected to D input of 24-bit register. Moreover, if the reset is 1, the output of 24-bit register is always 0.

5) 24-bit Counter which output is displayed by seven segment displays

The final result of this lab is 24-bit counter connected to six of seven segment displays. By connecting six of seven segment displays, it is able to count the numbers until millions.



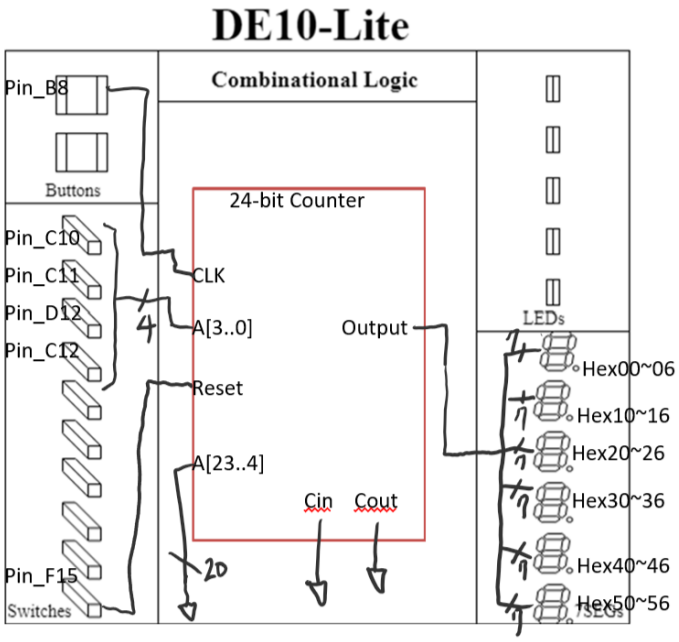
In the schematic design above, Cin input and Cout output are connected to ground to avoid from making floating values in both hardware testing and simulation testing. Moreover, only 4 of A inputs have assign pins and the others are connected to ground. The reason is that in the hardware, there were not enough pins to assign 24-bit input values so that only 4 bits of the input is assign to the pins.

Each of the seven segment displays represents one of the six digits numbers. The counting is in hexadecimal numbers with using the truth table below.

Input (Hexadecimal)	Input (4-bit Binary)	Seg_A	Seg_B	Seg_C	Seg_D	Seg_E	Seg_F	Seg_G
0	0000	0	0	0	0	0	0	1
1	0001	1	0	0	1	1	1	1
2	0010	0	0	1	0	0	1	0
3	0011	0	0	0	0	1	1	0
4	0100	1	0	0	1	1	0	0
5	0101	0	1	0	0	1	0	0
6	0110	0	1	0	0	0	0	0
7	0111	0	0	0	1	1	1	1
8	1000	0	0	0	0	0	0	0
9	1001	0	0	0	1	1	0	0
a	1010	0	0	0	1	0	0	0
b	1011	1	1	0	0	0	0	0
c	1100	0	1	1	0	0	0	1
d	1101	1	0	0	0	0	1	0
e	1110	0	1	1	0	0	0	0
f	1111	0	1	1	1	0	0	0

Furthermore, the clock and the reset are controlled by switches.

6) Block Diagram



The block diagram above is 24-bit counter connected to six of seven segment displays. Seven segment displays will represent the output of 24-bit counter in hexadecimal numbers like the shape below.

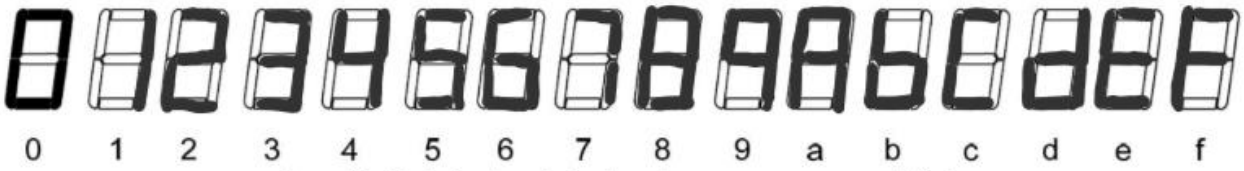


Figure 3.2: Displaying hexadecimal numbers on a seven-segment display

7) Pin Assignment

I. Inputs

Interface Name	Interface Purpose	Input or Output	Number of Pins	Board Label	FPGA Pin
A0	Gate input	Input	1	Switch 0	C10
A1	Gate input	Input	1	Switch 1	C11
A2	Gate input	Input	1	Switch 2	D12
A3	Gate input	Input	1	Switch 3	C12
A[23..4]	Gate input	Input	0	Ground	None
Cin	Gate input	Input	0	Ground	None
Reset	Gate input	Input	1	Switch 9	F15
CLK	Gate input	Input	1	Key 0	B8

II. Outputs-Cout and Seg0

Interface Name	Interface Purpose	Input or Output	Number of Pins	Board Label	FPGA Pin
Cout	Gate output	Output	0	Ground	None
Seg0[0]	Gate output	Output	1	Hex00	C14
Seg0[1]	Gate output	Output	1	Hex01	E15
Seg0[2]	Gate output	Output	1	Hex02	C15
Seg0[3]	Gate output	Output	1	Hex03	C16
Seg0[4]	Gate output	Output	1	Hex04	E16
Seg0[5]	Gate output	Output	1	Hex05	D17
Seg0[6]	Gate output	Output	1	Hex06	C17

III. Outputs-Seg1

Interface Name	Interface Purpose	Input or Output	Number of Pins	Board Label	FPGA Pin
Seg1[0]	Gate output	Output	1	Hex10	C18
Seg1[1]	Gate output	Output	1	Hex11	D18
Seg1[2]	Gate output	Output	1	Hex12	E18
Seg1[3]	Gate output	Output	1	Hex13	B16
Seg1[4]	Gate output	Output	1	Hex14	A17
Seg1[5]	Gate output	Output	1	Hex15	A18
Seg1[6]	Gate output	Output	1	Hex16	B17

IV. Outputs-Seg2

Interface Name	Interface Purpose	Input or Output	Number of Pins	Board Label	FPGA Pin
Seg2[0]	Gate output	Output	1	Hex20	B20
Seg2[1]	Gate output	Output	1	Hex21	A20
Seg2[2]	Gate output	Output	1	Hex22	B19
Seg2[3]	Gate output	Output	1	Hex23	A21
Seg2[4]	Gate output	Output	1	Hex24	B21
Seg2[5]	Gate output	Output	1	Hex25	C22
Seg2[6]	Gate output	Output	1	Hex26	B22

V. Outputs-Seg3

Interface Name	Interface Purpose	Input or Output	Number of Pins	Board Label	FPGA Pin
Seg3[0]	Gate output	Output	1	Hex30	F21
Seg3[1]	Gate output	Output	1	Hex31	E22
Seg3[2]	Gate output	Output	1	Hex32	E21
Seg3[3]	Gate output	Output	1	Hex33	C19
Seg3[4]	Gate output	Output	1	Hex34	C20
Seg3[5]	Gate output	Output	1	Hex35	D19
Seg3[6]	Gate output	Output	1	Hex36	E17

VI. Outputs-Seg4

Interface Name	Interface Purpose	Input or Output	Number of Pins	Board Label	FPGA Pin
Seg4[0]	Gate output	Output	1	Hex40	F18
Seg4[1]	Gate output	Output	1	Hex41	E20
Seg4[2]	Gate output	Output	1	Hex42	E19
Seg4[3]	Gate output	Output	1	Hex43	J18
Seg4[4]	Gate output	Output	1	Hex44	H19
Seg4[5]	Gate output	Output	1	Hex45	F19
Seg4[6]	Gate output	Output	1	Hex46	F20

VII.Outputs-Seg5

Interface Name	Interface Purpose	Input or Output	Number of Pins	Board Label	FPGA Pin
Seg5[0]	Gate output	Output	1	Hex50	J20
Seg5[1]	Gate output	Output	1	Hex51	K20
Seg5[2]	Gate output	Output	1	Hex52	L18
Seg5[3]	Gate output	Output	1	Hex53	N18
Seg5[4]	Gate output	Output	1	Hex54	M20
Seg5[5]	Gate output	Output	1	Hex55	N19
Seg5[6]	Gate output	Output	1	Hex56	N20

3. Results

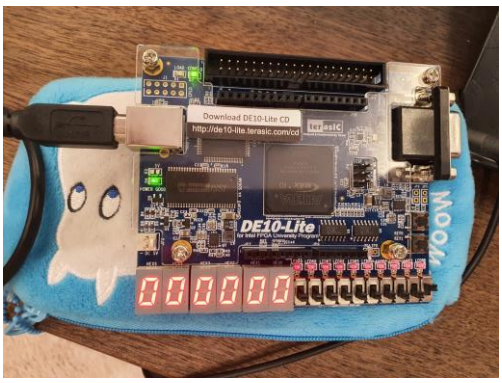
1) Hardware Results

Since there are millions of hardware results, there will be 7 randomly selected results in this report.

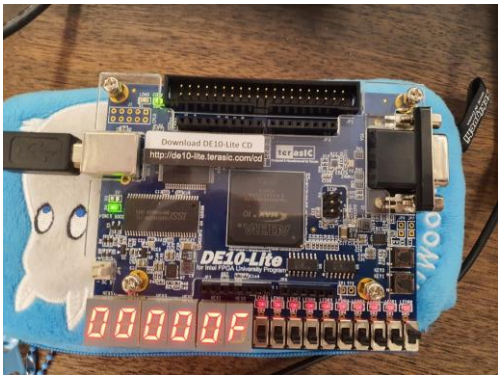
Input A (4-bit)	Number Of CLK pressed	Reset	Expected Result (Hexadecimal)	Actual Result (Hexadecimal)
0001	15	0	F	F
1010	7	1	0	0
1010	7	0	46	46
0101	20	0	64	64
1111	10	0	96	96
0000	15	0	0	0
1011	30	0	14A	14A

(Note: The row of the table starts from row 0 to row 7)

- Row#1

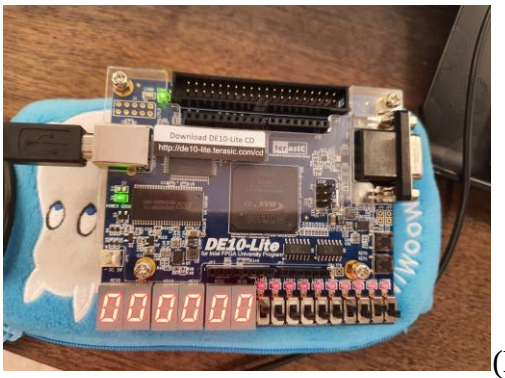


(Before)

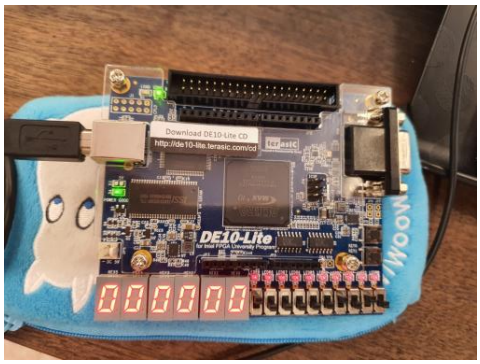


(After)

- Row#2



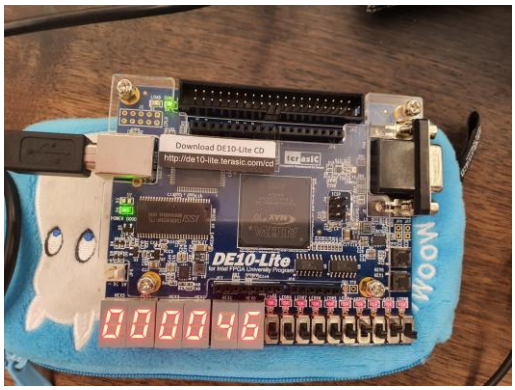
(Before)



(After)

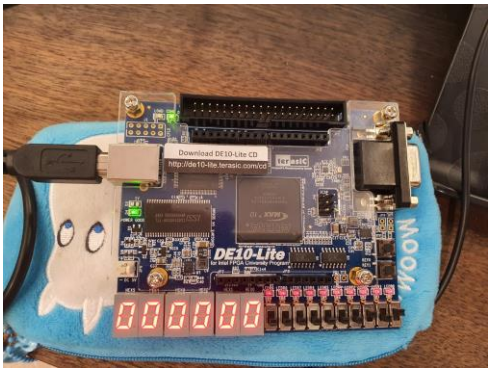
- Row#3

(Before)

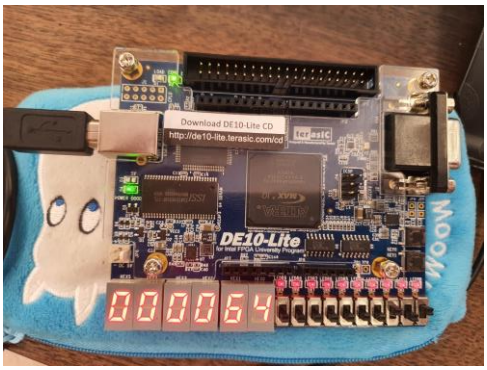


(After)

- Row#4

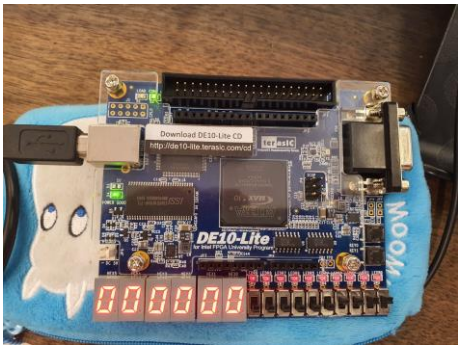


(Before)

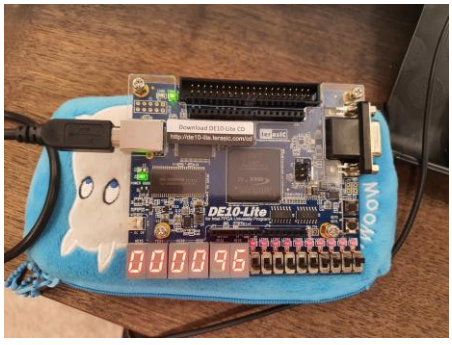


(After)

- Row#5



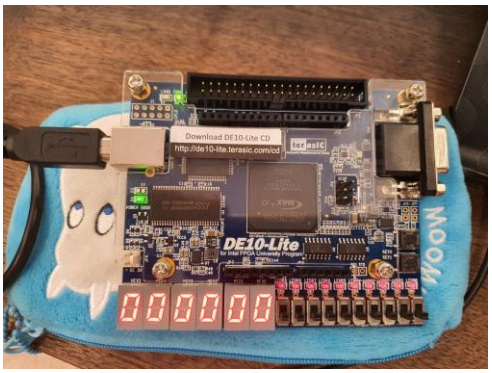
(Before)



(After)

- Row#6

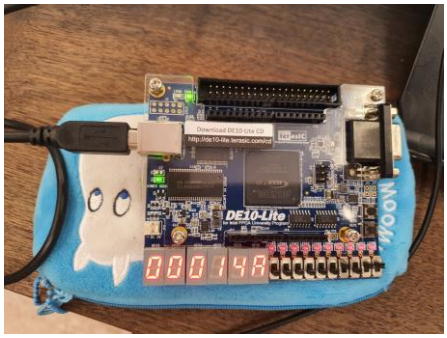
(Before)



(After)

- Row#7

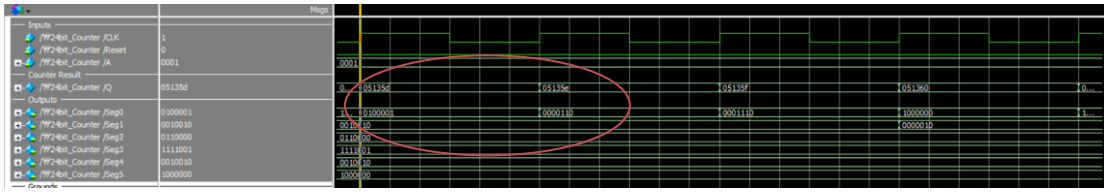
(Before)



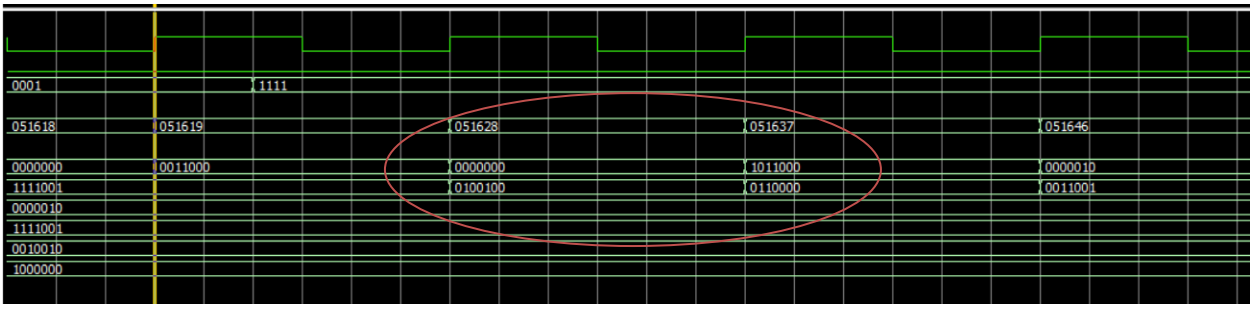
(After)

2) Simulation Result (ModelSim)

After verifying the implementation of the design, to verify the design much more precisely, the verification by simulation is required.



When the value of input A[3..0] is 0001 in binary and the clock is rising edge, the output of 24-bit is increasing by 1 in decimal. Moreover, the seven segment displays represent the output in hexadecimal. The red oval indicates that the output of 24-bit counter is counted up by 1 in decimal.



Moreover, when the value of input A is 1111in binary and the clock is rising edge, the output of 24-bit counter increased by 15 in decimal. Moreover, the seven segment displays represent the output in hexadecimal. The red oval indicates that the output of 24-bit counter is counted up by 15 in decimal.

3) Conclusion

The result of the hardware, DE10-Lite, and the result of simulation matched the expected result. Both of the results demonstrates that the implementation of the design of 24-bit counter is correct. Therefore, implementing 24-bit counter connected to six of seven segment displays is successful.

4. Experiment Notes

In this section, most of the parts were challenged to me. The reason is that whenever I program the FPGA with my design, the output was random six digits of hexadecimal numbers. Moreover, this was my first time to use buses so that it was quite hard to connecting multi-bits of input and output in the design.

I solved these problems by asking professor and reading my design thoroughly. When I asked my problem to professor, I realized that I did my counter design wrong, and get solution from professor. After solving the problem of counter, I also realized that there is some connection problem by watching RTL viewer. The problem was that one of the inputs of the adder was connected to seven segment displays. I solved this by making another symbol block of 24-bit counter.

After solving the problems in hardware, there were huge problems in simulation. The problems were that whenever there is a change in the clock input, the outputs were only showing X and Z values. I also asked this problem to professor, and professor gave me a hint that the problem is related to Cin. Thus, I connected Cin and Cout to the ground in my design and solved this problem. However, there were another problem that is resulting same output as the previous problem. I observed the input and outputs in ModelSim. The reason was all of the pins which I connected to the ground were having Z values. So, before I run the simulation, I forced the pins which were connected to the ground to have a value of 0. This solved the problem and the simulation worked well.

Unfortunately, I solved the problems mentioned after 5pm. Moreover, in the office hours, I couldn't get a check off because I was solving the problems. Hence, I realized that I need to start the lab early, and always be thorough in the process of the design and the simulation.

5. Study Questions

1. What is an alternate method of designing a counter with digital logic?

   An alternate method of designing a counter with digital logic is connecting adder and Synchronous Flip-Flop as the design in this section. The difference is adding AND gate to the D input of Synchronous Flip-Flop [1].

2. What functions do the DFF ports PRN and CLRN perform?

   PRN presets the output. This means that if PRN is turned on, then the output is the previous state of the output. CLRN clears the output. When CLRN is turned on the output is zero or low [2], [3].

3. How would this functionality be used to implement a reset?

   Using this functionality, reset can be implemented by turning off the PRN and turning on the CLRN. Reset is active high, and PRN and CLRN are active low. By connecting NOT gate to both input of PRN and CLRN, active high of reset is implemented. Therefore, reset is implemented by turning off the PRN with an input connected NOT gate, and turning on the CLRN with an input connected NOT gate.

6. Citation

[1] Digital Design and Computer Architecture, 2nd Edition,
Chapter 3 Slide show, Slide 24. Accessed 07/28/2021.

[2] DFF Primitive
https://www.intel.com/content/www/us/en/programmable/quartushelp/13.0/mergedProjects/hdl/prim/prim_file_dff.htm

[3] Using Quartus II Logic Function
http://www.altera.co.kr/_altera/html/_sw/q2help/source/vhdl/vhdl_pro_logic_funct.htm