ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 242E
DIGITAL CIRCUITS LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	5
EXPERIMENT DATE	:	16.04.2021
LAB SESSION	:	FRIDAY - 14.00
GROUP NO	:	G14

GROUP MEMBERS:

150180112	:	ÖMER MALİK KALEMBAŞI
150190014	:	FEYZA ÖZEN
150190108	:	EKİN TAŞYÜREK

SPRING 2021

Contents

1 INTRODUCTION [10 points]

In this experiment, we aimed to implement JK Flip Flop, T Flip Flop, and a circuit that combination of these flip flops and logic gates. Then we aimed to implement a 4 bit counter circuit and a 16-bit up-down counter. In the and we aimed to implement different counter modules using the 16-bit up-down counter.

2 MATERIALS AND METHODS [40 points]

2.1 PART 1

In this part, we implemented the circuit in Figure 2 by using Verilog. You used different type flip flop modules, logic gates and clock signal to synchronize the circuit. Firstly we created a state transition table and a state chart. To do this, we determined the expression for the F function that derives the JK and T Flip Flops.

1st Step:
These are the functions that derive flip flops:
$J_0 = Q_0 + A' + B'$
$K_0 = Q_1' + A' + B'$
$T_1 = (A + B) \cdot Q_1'$

2nd Step:
a) Next state equation for JK Flip Flop:
$Q_0^+ = J \cdot Q_0' + K' \cdot Q_0$
$Q_0^+ = Q_0' \cdot A' + Q_0' \cdot B' + Q_1 \cdot A \cdot B \cdot Q_0$

b) Next state equation for T Flip Flop:
$T_1^+ = T' \cdot Q_1 + T \cdot Q_1'$
$T_1^+ = A' \cdot B' \cdot Q_1 + Q_1 + A \cdot Q_1' + B \cdot Q_1'$
$T_1^+ = A + B + Q_1$

3rd Step:
Output equation Z: $Z = (A' \cdot Q_0') + (B' \cdot Q_0') + Q_1'$

$Q_1^+Q_0^+, Z$	AB			
$Q_1^+Q_0^+$	00	01	10	11
00	01, 1	11, 1	11, 1	10, 1
01	00, 1	10, 1	10, 1	10, 1
10	11, 1	11, 1	11, 1	10, 0
11	10, 0	10, 0	10, 0	11, 0

Table 1: State Table of Figure 1

	AB			
S^+	00	01	10	11
A	B, 1	D, 1	D, 1	C, 1
B	A, 1	C, 1	C, 1	C, 1
C	D, 1	D, 1	D, 1	C, 0
D	C 0	C, 0	C, 0	D, 0

Table 2: State Table(II) of Figure 1

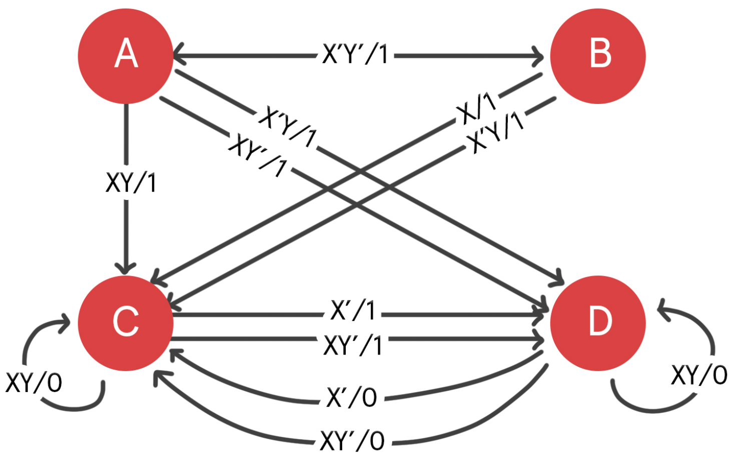

Figure 1: State Chart of Figure 1

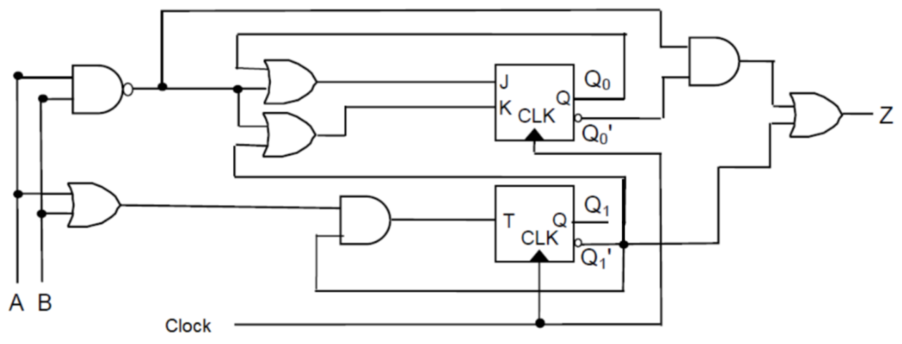

Figure 2: Circuit

After finding the equations, we implemented a JK Flip Flop using one SR Latch. Figure 3 shows the inside of the JK Flip Flop. We used the SR Latch that we implemented before in Experiment 4.

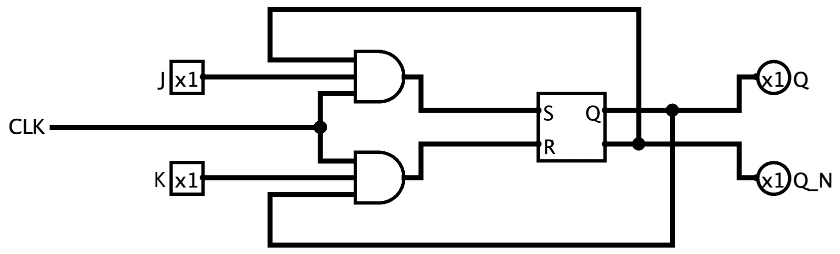

Figure 3: JK Flip Flop

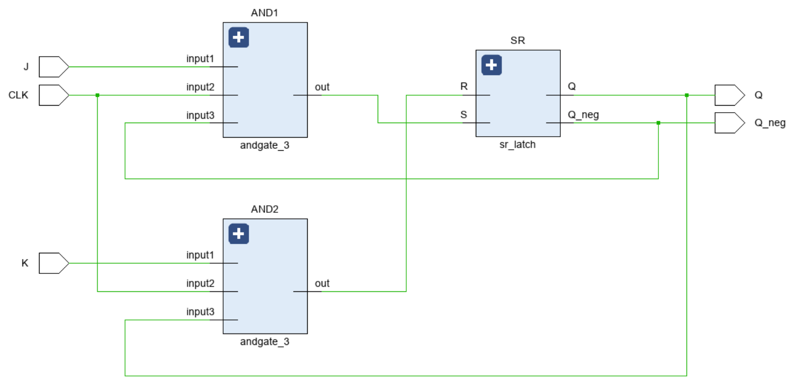

Figure 4: JK Flip Flop RTL Schematic

Then, we implemented T Flip Flop using D Flip Flop that we used before in Experiment 4. Figure 4 shows the inside of the T Flip Flop.

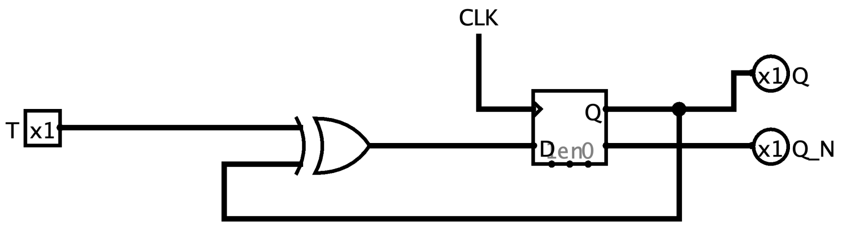

Figure 5: T Flip Flop

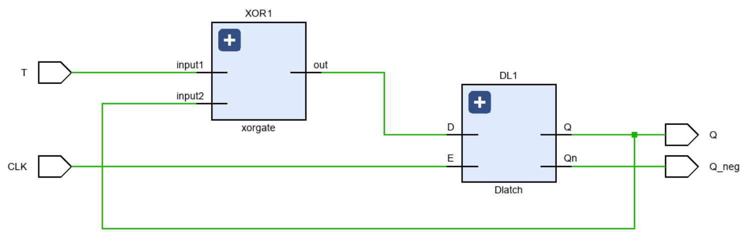

Figure 6: T Flip Flop RTL Schematic

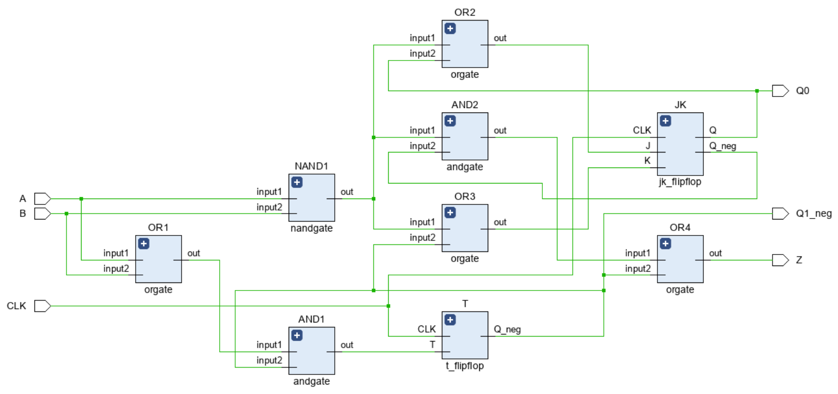

Figure 7: Part 1 RTL Schematic

2.2 PART 2

In Part 2, we implemented a 4 bit counter circuit that counts from 0 to 14 in a circular way. At each active clock signal, the counter counts. We designed our counter as a synchronus sequential circuit in Moore model, we took outputs as states. We firstly created the state table, then we separated it to 4 different tables according to output. Since D flip-flops are equal to Q(t+1), the equations that we found from our state tables are equal to D3, D2, D1 and D0. The equations that represents D3, D2 D1 and D0 are like below: D3 = Q0Q3 + Q0Q1'X + Q0Q1Q2' + Q0Q2X' + Q1Q2Q3X + Q0'Q1'Q2'Q3'X' D2 = Q1Q2'X + Q1Q2'Q3 + Q1Q2X' + Q1'Q2'Q3'X' + Q1'Q2Q3X + Q0'Q1Q3'X D1 = Q2'Q3'X' + Q2'Q3X + Q2Q3X' + Q0'Q2Q3'X + Q1'Q2Q3'X D0 = Q0'Q3'X + Q2Q3'X' + Q1Q2'Q3' + Q0Q1'Q3' Then, we implemented our circuit using AND, OR gates and D flip-flops.

Q$_3Q_2Q_1Q_0$	X=0	X=1
0000	1110	0001
0001	0000	0010
0010	0001	0011
0011	0010	0100
0100	0011	0101
0101	0100	0110
0110	0101	0111
0111	0110	1000
1000	0111	1001
1001	1000	1010
1010	1001	1011
1011	1010	1100
1100	1011	1101
1101	1100	1110
1110	1101	0000
1111	Φ	Φ

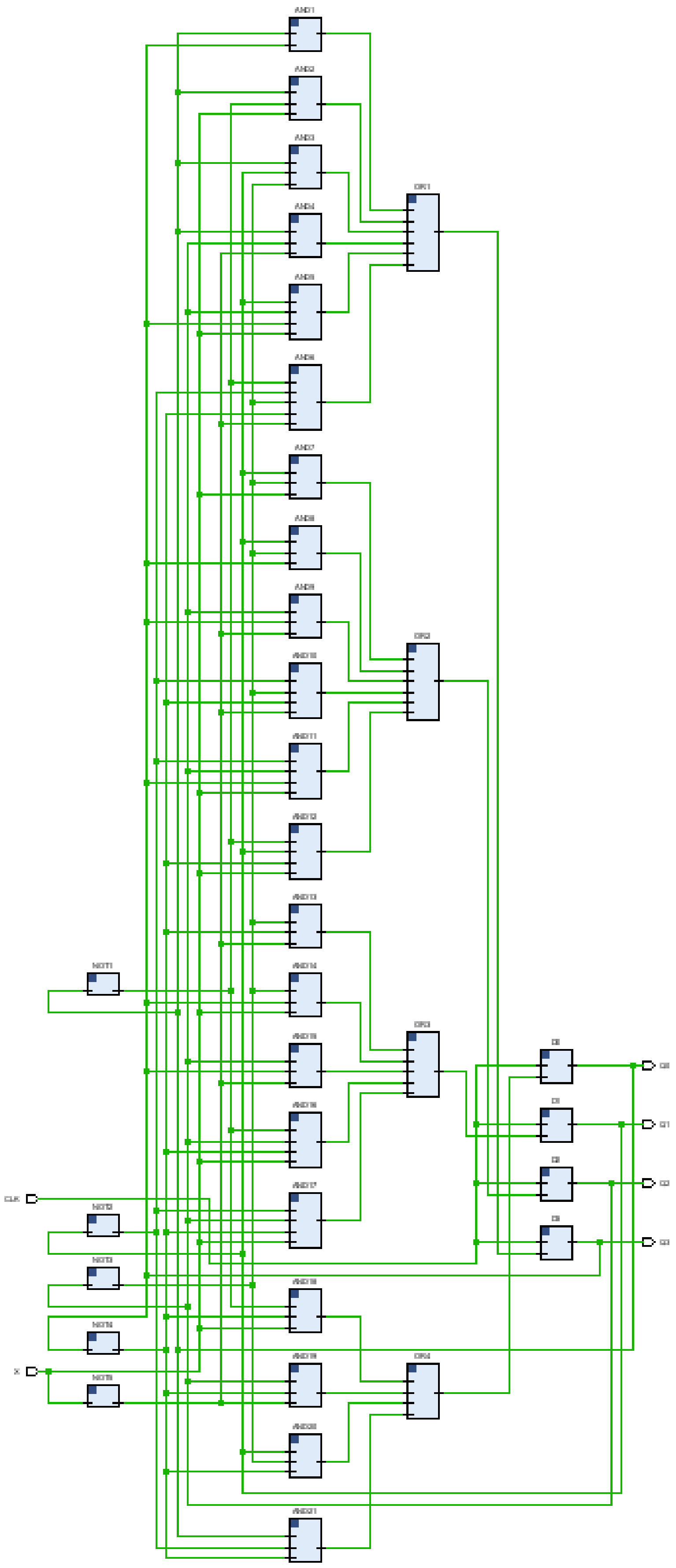

Figure 8: Part 2 RTL Schematic

2.3 PART 3

In this part, we implemented a 16-bit up-down counter that has 16-bit input data, load, clock, direction, 3-bit increment/decrement value, and clear inputs. An up-down counter is a circuit that can count in both directions, increasing and decreasing. In this part, we also add an increment decrement value that can provide us to count two each, three each, etc. We used always block to make the circuit work the only positive edge of the clock signal and used switch-case block to implement different increment/decrement values. The circuit decides when to increase or decrease by using direction input (X). We used the up-down counter as a reg-type parameter. Also, we used a Clear input that determines to load. When the clear input is zero, it clears the current value of the register. When Clear input is one, it provides the input values load into the register and makes counter active.

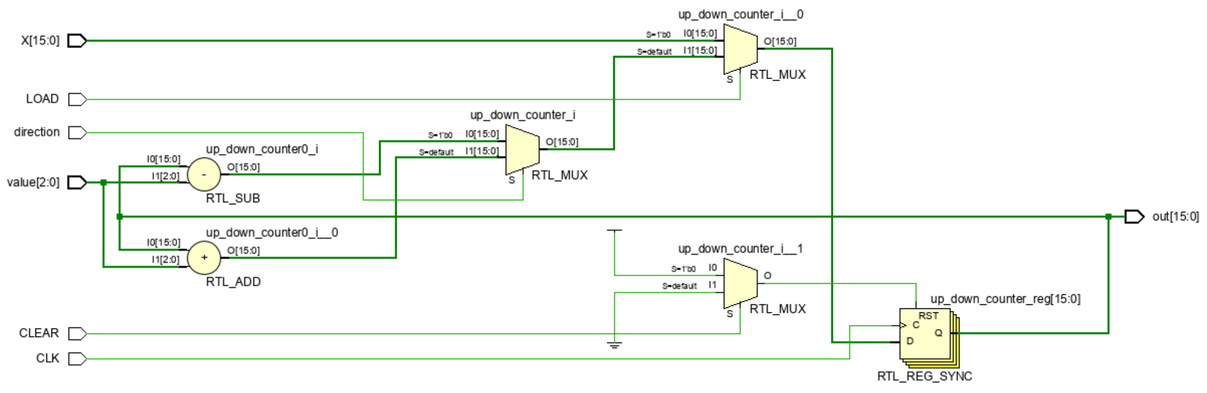

Figure 9: 16 Bit Up-Down Counter RTL Schematic

2.4 PART 4

In this part, we implemented 4 different counter modules using a 16-bit up-down counter module that is implemented in Part 3. In module Part4_1, we made a counter that counts 0 to 40 circular way with increment value 2. In module Part4_2, we made a counter that counts 350 to 371 circular way with increment value 3. In module Part4_3, we made a counter that counts 93 to 5 circular way with decrement value 4. In module Part4_4, we made a counter that counts 22525 to 22535 circular way with decrement value 1.

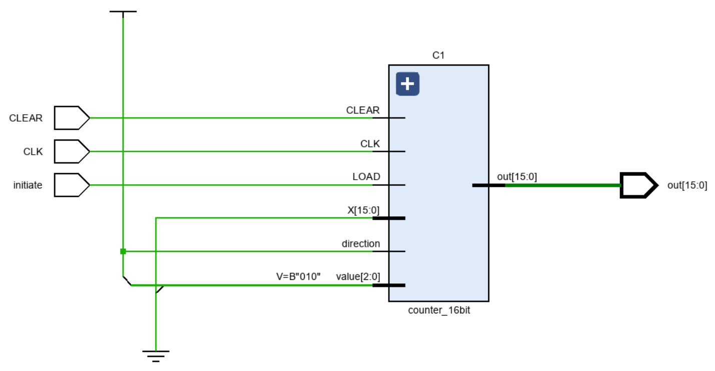

Figure 10: Part 4_1

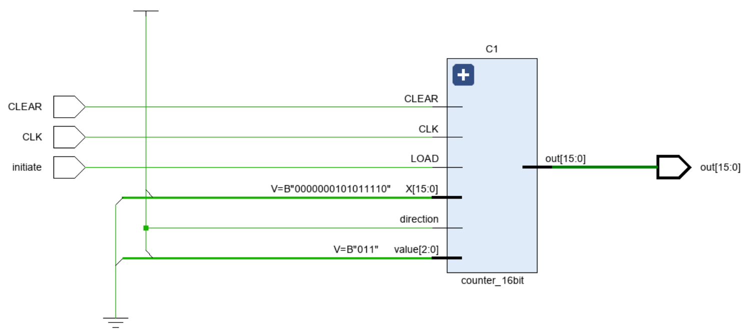

Figure 11: Part 4_2

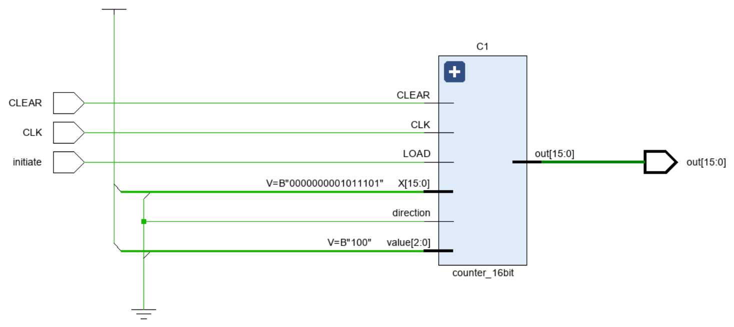

Figure 12: Part 4_3

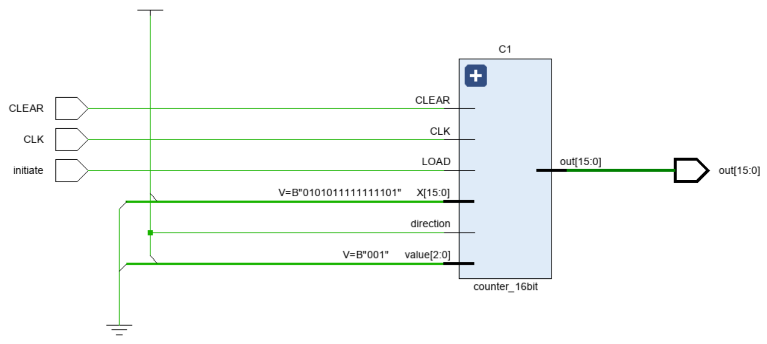

Figure 13: Part 4_4

3 RESULTS [15 points]

3.1 PART 1

JK Flip Flop and T Flip Flop simulation results are wrong, therefore Part1 Circuit does not work correctly.

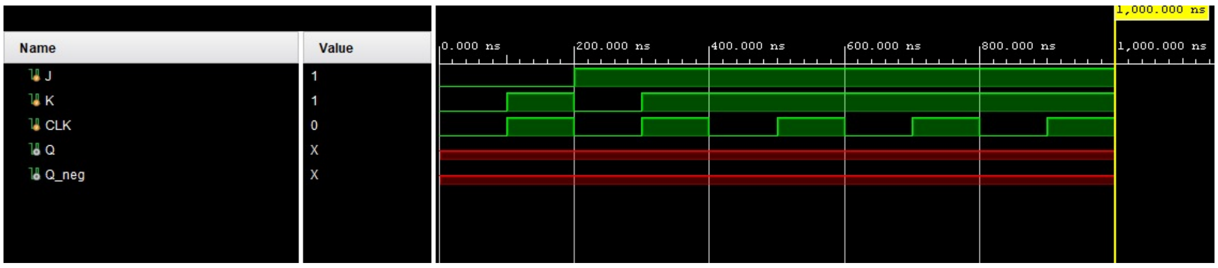

Figure 14: JK Flip Flop Simulation

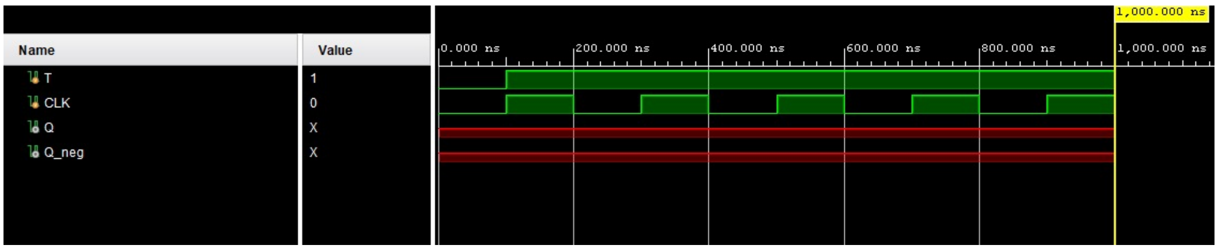

Figure 15: T Flip Flop Simulation

3.2 PART 2

3.3 PART 3

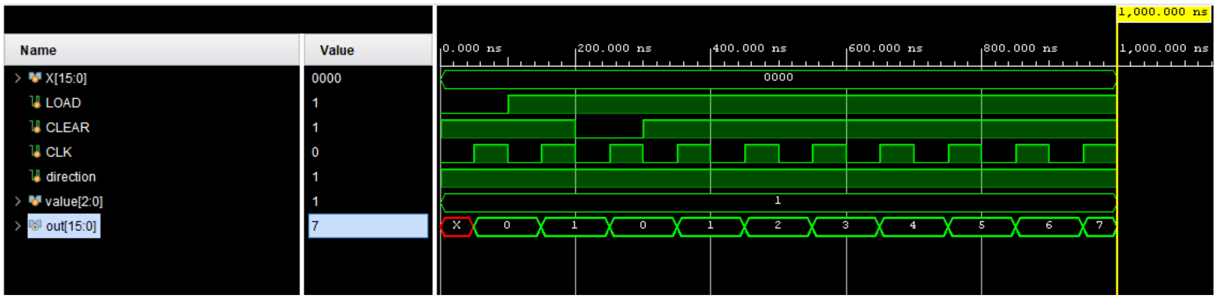

Figure 16: Simulation

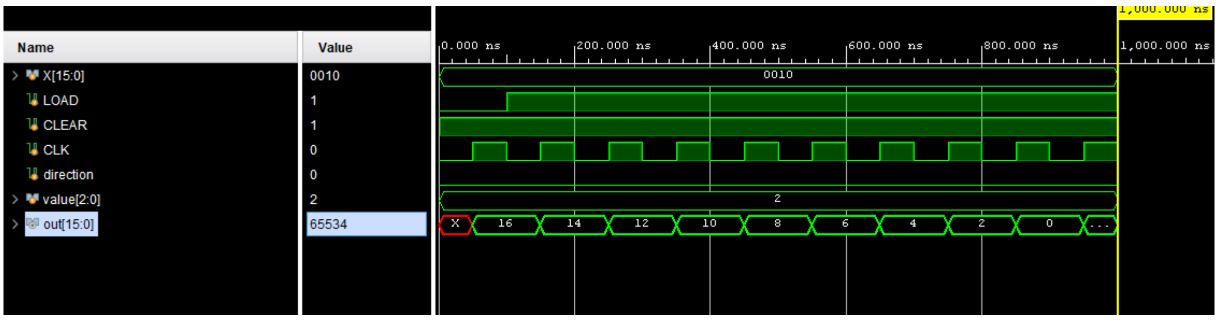

Figure 17: Simulation

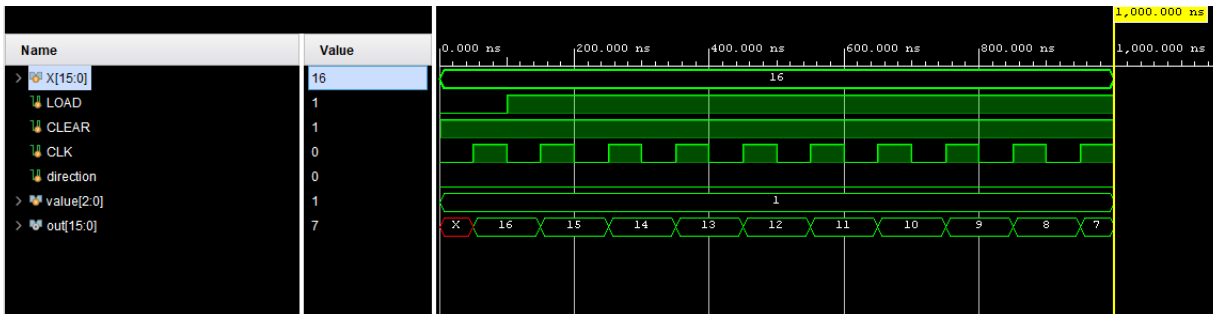

Figure 18: Simulation

3.4 PART 4

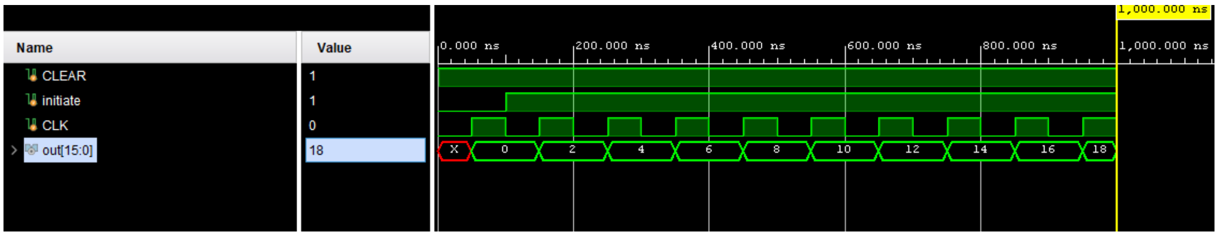

Figure 19: Part 4_1 Simulation

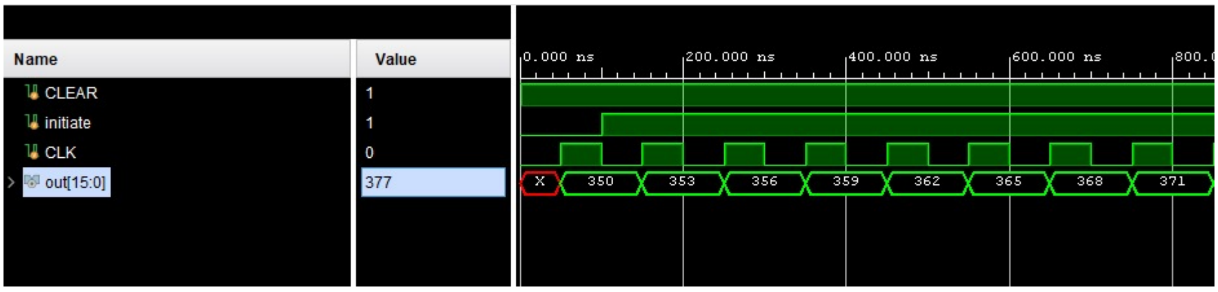

Figure 20: Part 4_2 Simulation

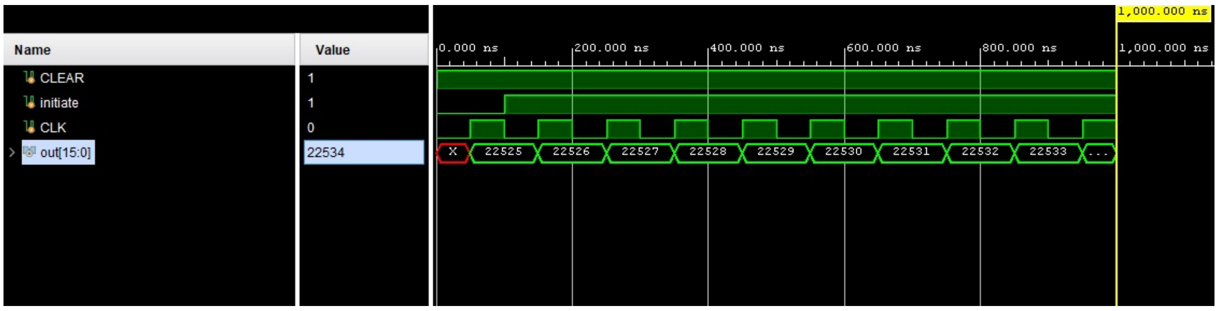

Figure 21: Part 4_4 Simulation

4 DISCUSSION [25 points]

In Part 1, we coded the circuit in Figure 1 in Verilog using logic gates and Flip Flops that we made. We had two inputs, A and B. NAND1 and OR1 take A and B as inputs. AND1 takes the output(araKablo2) of OR1 and Q_1'(araKablo4). T-flip flop (T) takes the output(araKablo3) of the AND1, clock signal, and Q_1(araKablo6) as inputs. OR2 takes the output(araKablo1) of NAND1 and Q_0(araKablo5). OR3 takes the output(araKablo1) of NAND1 and Q_1'(araKablo4). JK-flip flop (JK) takes the output(araKablo7) of OR2 and output(araKablo8) of OR3, a clock signal as inputs. AND2 takes the output(araKablo1) of NAND1 and Q_0'(araKablo9). At the end, OR4 takes the output of AND2(araKablo10) and Q_1'(araKablo4) as inputs. Then gives the output Z which is the output of our circuit. When we look at the simulation results, T-Flip Flop and JK Flip Flop did not work correctly. Therefore the circuit in Part1 does not work correctly.

In Part 2, we implemented a 4 bit counter circuit that counts from 0 to 14 in a circular way. Our inputs were X which represents down or up and CLK which represents clock. We had 4 outputs Q0, Q1, Q2 and Q3 representing the 4 bits of our output. We implemented our code in 5 parts: D3, D2, D1, D0 and lastly Q3, Q2, Q1, Q0. While implementing D3, we used 6 AND gates for Q0Q3, Q0Q1'X, Q0Q1Q2', Q0Q2X', Q1Q2Q3X and Q0'Q1'Q2'Q3'X'. Then we summed them using an OR gate. While implementing D2, we again used 6 AND gates and an OR gate. For D1, we used 5 AND gates and an OR gate. For D0, we used 4 AND gates and an OR gate. While implementing our outputs, we used 4 D flip-flops that take D3, D2, D1, D0 and CLK as input and gives Q1, Q2, Q3, Q4 as output.

In Part 3, we implemented a 16-bit up-down counter. We had 6 inputs: 16-bit X representing our data, LOAD representing load flag, CLEAR representing reset, CLK representing clock, direction representing up or down and value representing the increment/decrement value. We defined a register -up-down-counter- to use in our always block. Our always block works only when CLK is on positive edge. After that, we have 3 different conditions to decide what our circuit will do. First one is that if if CLEAR is 0, counter will reset and will be equal to. Second one is that if LOAD is 0, the X value will be uploaded to counter. Third one happens when both LOAD and CLEAR are 1. According to our direction -0 is down and 1 is up- and increment/decrement value, the counter starts to count and the value of up-down-counter changes. In the end, up-down-counter will be assigned to output.

In Part 4, we implemented 4 different modules using the 16-bit up down counter we implemented in the previous part: a counter that counts up from 0 to 40 with increment value 2, a counter that counts up from 350 to 371 with increment value 3, a counter that counts down from 93 to 5 with decrement value 4 and a counter that counts up from 22525 to 22535 with increment value 1. We have 3 inputs: CLEAR representing reset, initiate that sets the counter to any value and CLK representing clock. Direction, value and data are already given to us. For example, in the first counter; the direction is 1 meaning up, value is 2 because it increments with 2 and the data is 0 in the first place. We use these informations and our input to form our 16-bit counter.

5 CONCLUSION [10 points]

In conclusion, we learned how to implement counters using always blocks and if-else blocks. Unfortunately, we could not make JK and T flip flops work. We tried to implement different kinds of JK flip flops that consist D flip flop, SR Flip Flop, or only NAND gates. But, each time they did not work even though their components (D Flip Flop, SR Flip Flop, NAND gates) work correctly.