ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 242E
DIGITAL CIRCUITS LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	4
EXPERIMENT DATE	:	09.04.2021
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 implemented and examined data storage elements: latches and flip-flops using Verilog.

A flip flop is an electronic sequential circuit that stores data with its two stable states. They have inputs that can change the stored data. A flip flop works with the control of a clock signal. The clock signal decides when the flip flop changes its state. Flip flops are used in designing storage devices, registers, and counters.

Flip flops work with a clock signal, unlike latches. Latches can change their output immediately when an input is applied. Also, a latch can be considered as the building block of a flip flop.

2 MATERIALS AND METHODS [40 points]

2.1 PART 1

In this part, we implemented an SR Latch module with S and R inputs and with Q and (Q_N) outputs without an Enable input using NAND Gates. An S-R (Set-Reset) latch is a latch consists of 2 NOR or 2 NAND gates. It has two inputs, one set (S), and one reset (R). It has two outputs, one is output (Q) and one is complemented output (Q_N). S input sets the output, and R input resets output. When S is 1 and R is 0, the output is 1. The output will not change when the value of S changes from 1 to 0. The value of R must be set to 1 for output to be 0. Again, the output will not change when the value of R changes from 1 to 0. S and R can not be 1 at the same time. Because this condition causes errors such as Q and N being 1 or being 0 at the same time. This is impossible as they are supposed to be opposite. Figure 1 shows the truth tables of an SR latch.

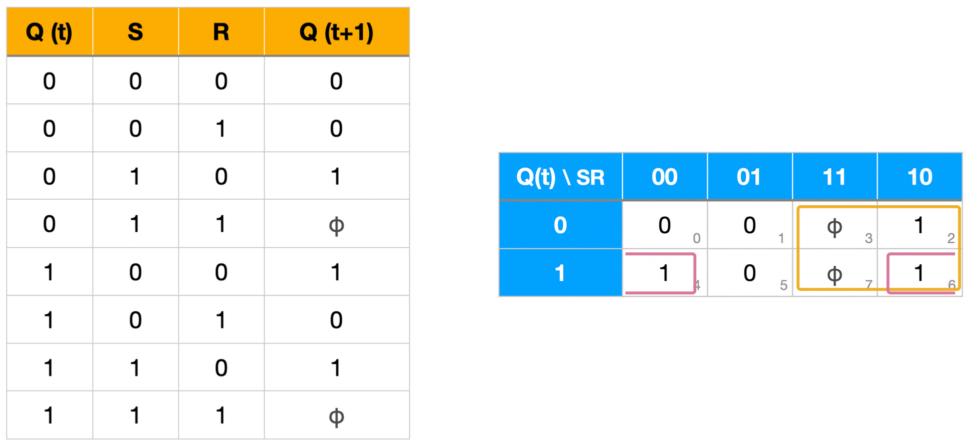
S	R	Q(t+1)	Q'(t+1)
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1
1	1	forbidden	forbidden

Table 1:

Karnaugh Map Results:

2,3,6,7: S

4,6: $R'Q(t)$



Q (t)	S	R	Q (t+1)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	ϕ
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	ϕ

Q(t) \ SR	00	01	11	10
0	0 $_{0}$	0 $_{1}$	ϕ $_{3}$	1 $_{2}$
1	1 $_{4}$	0 $_{5}$	ϕ $_{7}$	1 $_{6}$

Figure 1: SR Latch Truth Table and Karnaugh Map

Characteristics equation of SR Latch:

$Q(t+1) = S + R'Q(t)$

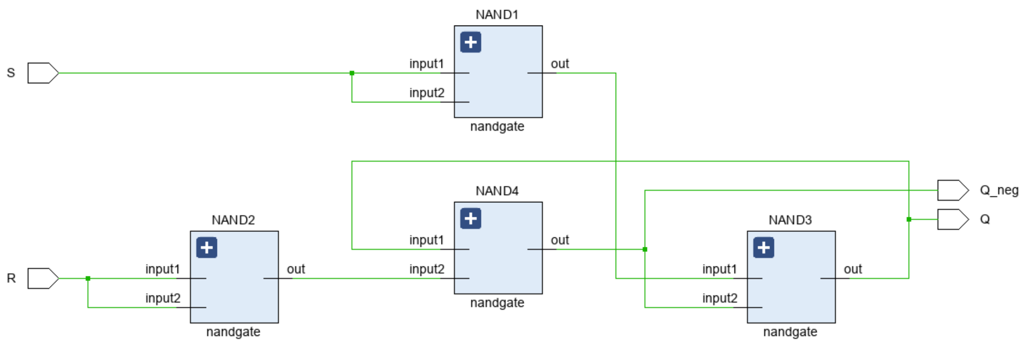


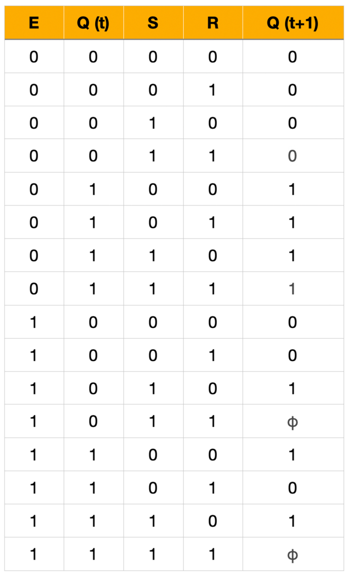
Figure 2: RTL Schematic of SR Latch

2.2 PART 2

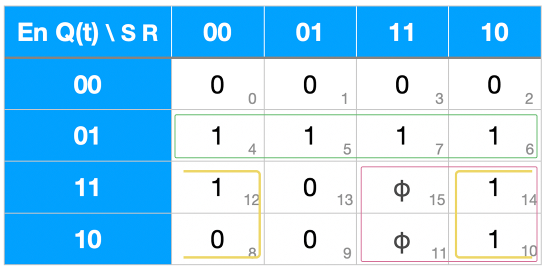
In this part, we implemented an SR Latch module with an Enable input using 2 NAND Gates. When Enable is 0, the output is the same as the previous output. When Enable is 1, outputs are the same as an SR Latch that has no Enable input. Figure 4 shows the truth tables of an SR latch with Enable.

E	S	R	Q(t+1)	Q'(t+1)
0	X	X	Q(t)	Q'(t)
1	1	0	1	0
1	0	0	1	0
1	0	1	0	1
1	0	0	0	1
1	1	1	forbidden	forbidden

Table 2:



E	Q (t)	S	R	Q (t+1)
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	1
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	ϕ
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	ϕ



En Q(t) \ S R	00	01	11	10
00	0 0	0 1	0 3	0 2
01	1 4	1 5	1 7	1 6
11	1 12	0 13	ϕ 15	1 14
10	0 8	0 9	ϕ 11	1 10

Figure 3: SR Latch with Enable Input Truth Table and Karnaugh Map

Karnaugh Map Results:

4,5,6,7: $En' \cdot Q(t)$

4,6,12,14: $Q(t) \cdot R'$

10,11,14,15: $En \cdot S$

Characteristics equation of SR Latch with Enable:

$Q(t+1) = En' \cdot Q(t) + Q(t) \cdot R' + En \cdot S$

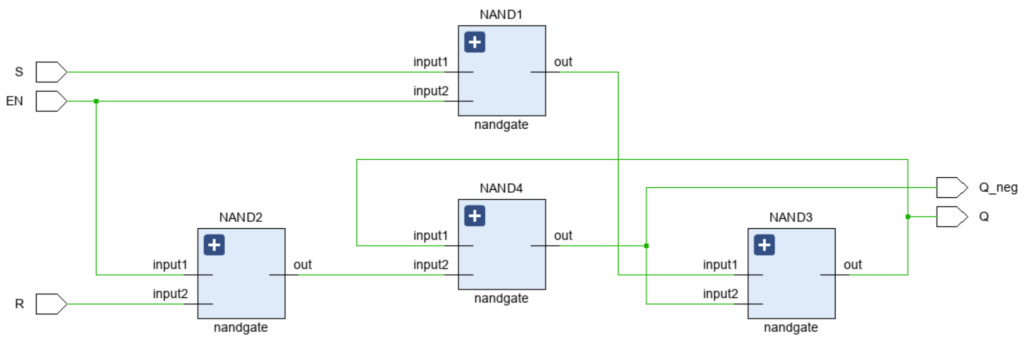


Figure 4: RTL Schematic of SR Latch with Enable

2.3 PART 3

In this part, we implemented a positive edge triggered D flip-flop module with D input and Q - Q' outputs. We used a clock signal to make D Flip Flop positive edge triggered. A D Flip Flop takes the input when an edge rises and delays it by one clock cycle. The output is the same as the input in the rising edge. D means delay.

CLK	D	Q(t+1)	Q'(t+1)
Rising Edge	0	0	1
Rising Edge	1	1	0
0	X	Q(t)	Q'(t)
1	X	Q(t)	Q'(t)

Table 3:

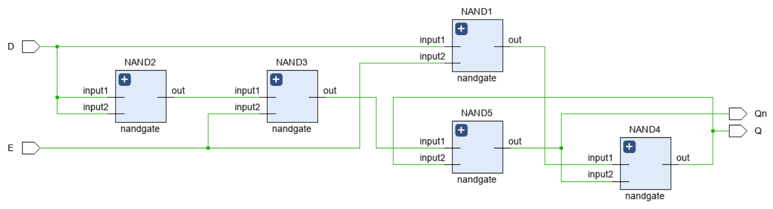


Figure 5: RTL Schematic of D Latch

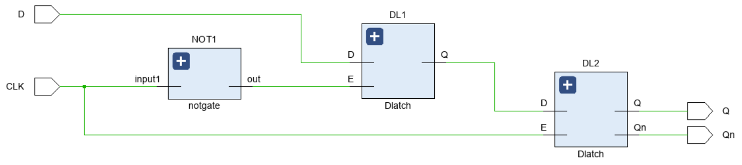


Figure 6: RTL Schematic of D Flip Flop

2.4 PART 4

In this part, we implemented a positive edge triggered pulse generator using a circular shift register. The circuit takes 16-bit input for the loaded value, 1-bit input for the clock signal, 1-bit input for the load flag and gives 1-bit output. Shift registers are sequential logic circuits that can be used when transfering or storing binary data. It loads the data present on its inputs and then moves it to the left or right once every clock cycle. If LOAD is 1, the inputs are loaded into flip-flops. If LOAD is 0, the data in the flip-flops are shifted.

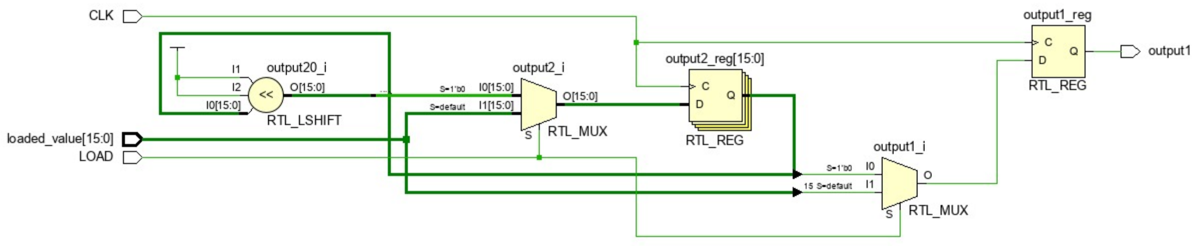


Figure 7: RTL Schematic of Part 4

3 RESULTS [15 points]

3.1 PART 1

The simulation results provide the truth table Table 1.

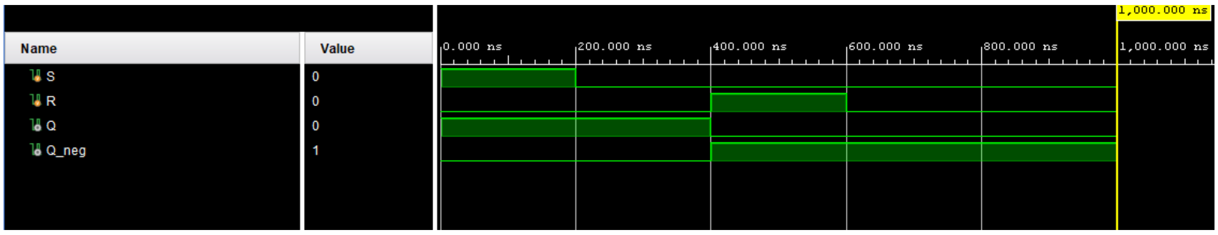


Figure 8: Part 1 Simulation

3.2 PART 2

The simulation results provide the truth table Table 2.

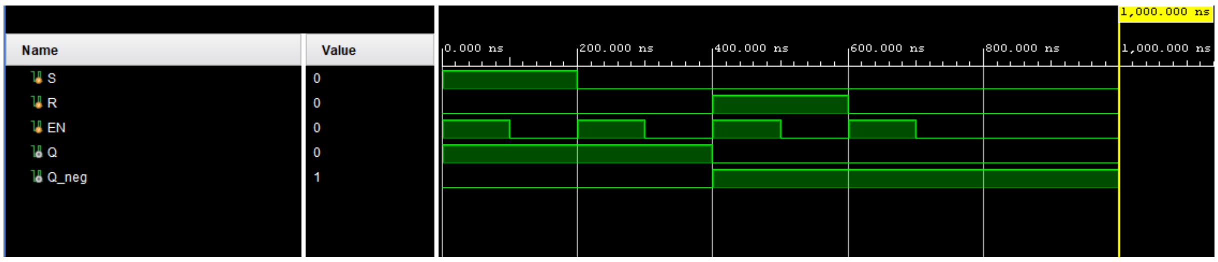


Figure 9: Part 2 Simulation

3.3 PART 3

The simulation results provide the truth table Table 3. It works with clock signal as we see in 100.000th ns. Clock signal is not rising there. Therefore Q does not change even D changes. Same situation can be seen in 200.000th ns and 300.000th ns.

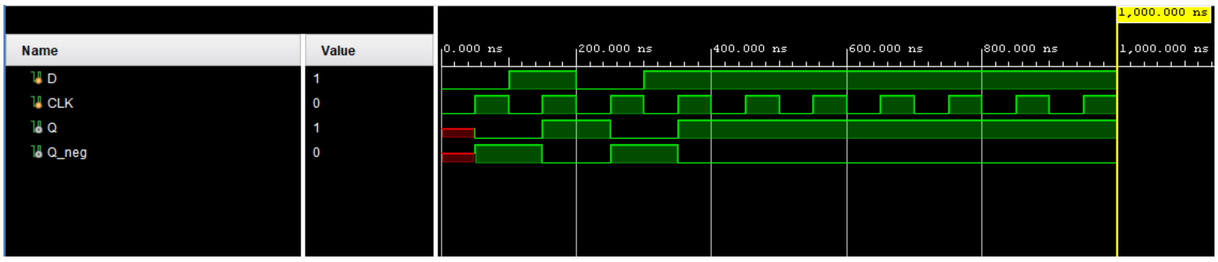


Figure 10: Part 3 Simulation

3.4 PART 4

For 1/7 pulse-gap duration rate, our input was 1000000010000000.

For 1/15 pulse-gap duration rate, our input was 1000000000000000.

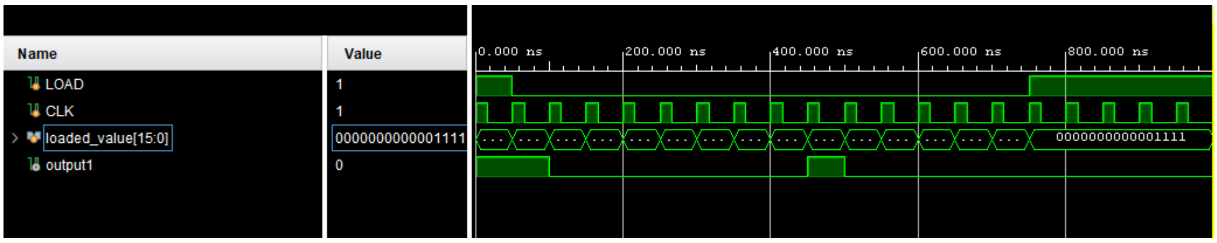
For 3/13 pulse-gap duration rate, our input was 1110000000000000.

For 11/5 pulse-gap duration rate, our input was 1111111111100000.

For the 1/2 frequency of clock signal, the active transitions of our clock was 16.6ns and 33.3ns.

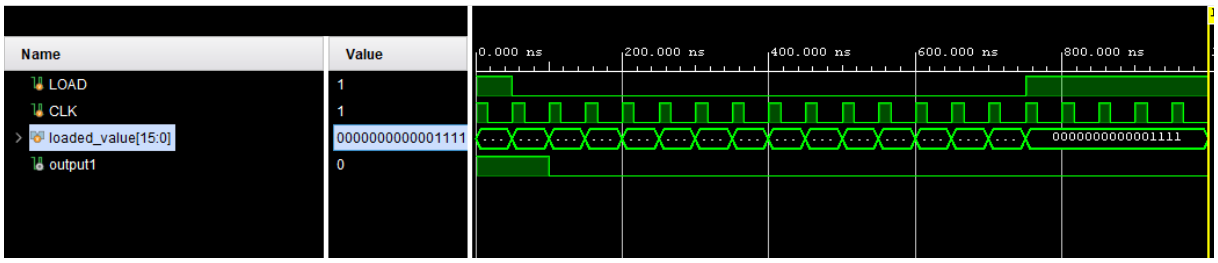
For the 1/4 frequency of clock signal, the active transitions of our clock was 10ns and 40ns.

For the 1/8 frequency of clock signal, the active transitions of our clock was 5.5ns and 44.4ns.



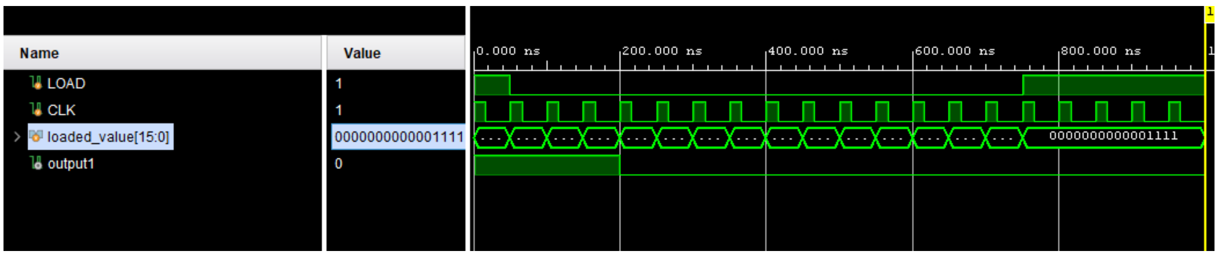
Name	Value
LOAD	1
CLK	1
loaded_value[15:0]	0000000000001111
output1	0

Figure 11: 1/2 frequency of clock signal,1/7 pulse-gap duration rate Simulation



Name	Value
LOAD	1
CLK	1
loaded_value[15:0]	0000000000001111
output1	0

Figure 12: 1/2 frequency of clock signal,1/15 pulse-gap duration rate Simulation



Name	Value
LOAD	1
CLK	1
loaded_value[15:0]	0000000000001111
output1	0

Figure 13: 1/2 frequency of clock signal,3/13 pulse-gap duration rate Simulation

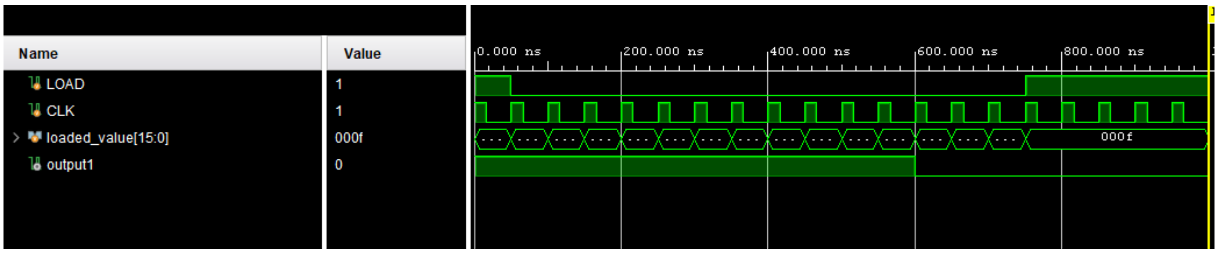


Figure 14: 1/2 frequency of clock signal,11/5 pulse-gap duration rate Simulation

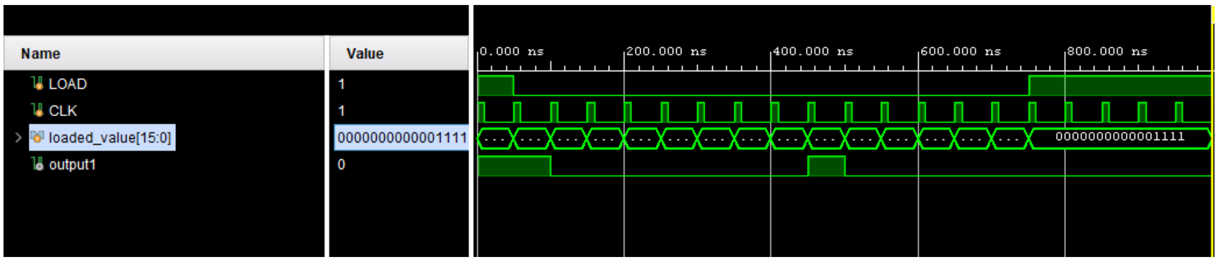


Figure 15: 1/4 frequency of clock signal, 1/7 pulse-gap duration rate Simulation

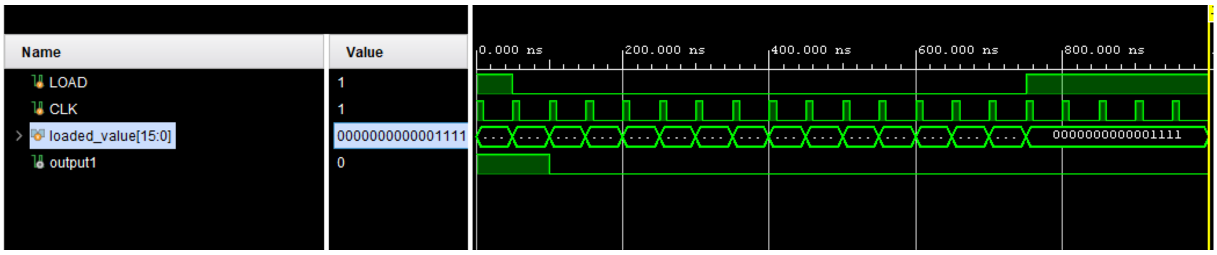


Figure 16: 1/4 frequency of clock signal, 1/15 pulse-gap duration rate Simulation

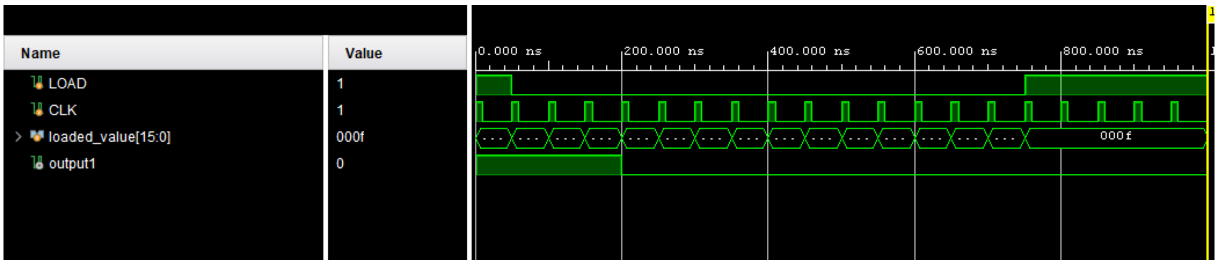


Figure 17: 1/4 frequency of clock signal, 3/13 pulse-gap duration rate Simulation

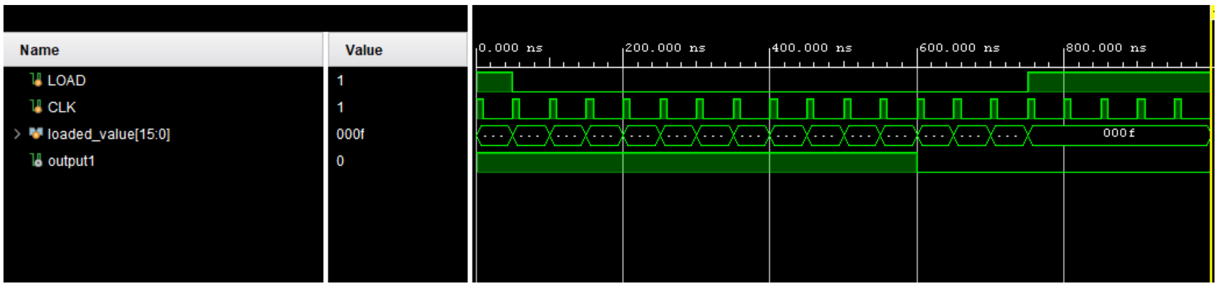


Figure 18: 1/4 frequency of clock signal, 11/5 pulse-gap duration rate Simulation

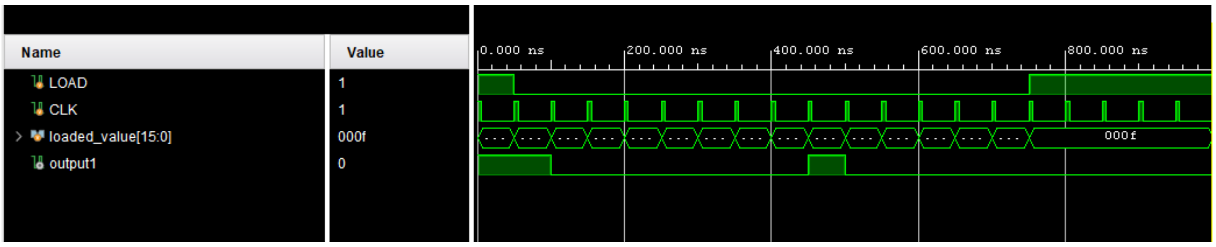


Figure 19: 1/8 frequency of clock signal, 1/7 pulse-gap duration rate Simulation

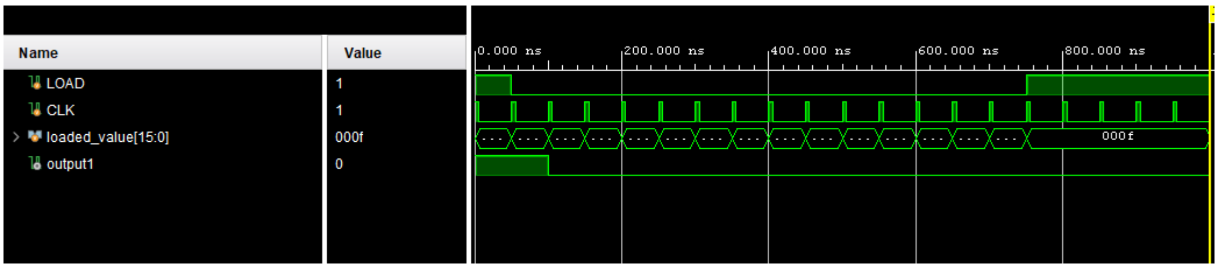


Figure 20: 1/8 frequency of clock signal, 1/15 pulse-gap duration rate Simulation

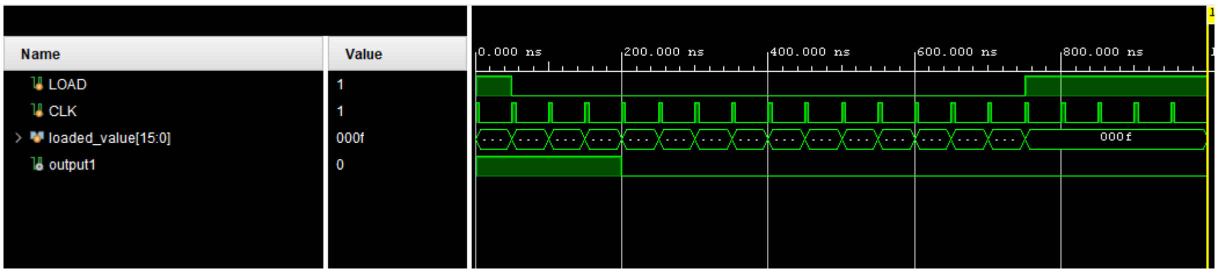


Figure 21: 1/8 frequency of clock signal, 3/13 pulse-gap duration rate Simulation

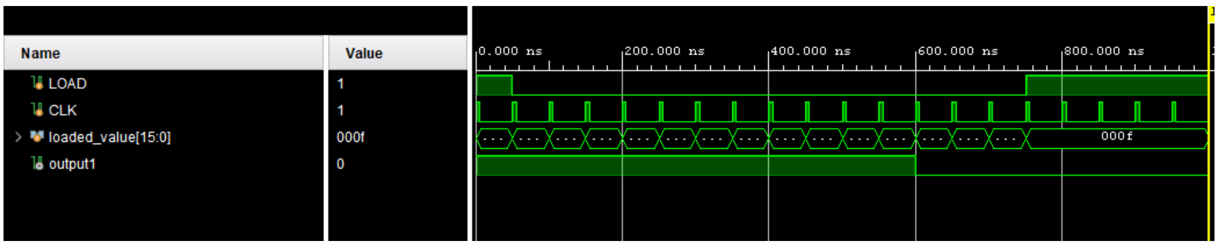


Figure 22: 1/8 frequency of clock signal, 11/5 pulse-gap duration rate Simulation

4 DISCUSSION [25 points]

In Part 1, we implemented an SR Latch module with S and R inputs and with Q and (Q_N) outputs without an Enable input using NAND Gates. We used 4 NAND Gates. In our code, their names are NAND1, NAND2, NAND3, NAND4. NAND 1 and NAND 2 are for complementing inputs. NAND3 takes (Q_N) and S' as inputs, gives Q as output. NAND4 takes Q and R' as inputs, gives (Q_N) as output.

In Part 2, we implemented an SR Latch module with S and R inputs and with Q and (Q_N) outputs with an Enable input using NAND Gates. We used 4 NAND Gates. In our code, their names are NAND1, NAND2, NAND3, NAND4. NAND 1 is for combining S with Enable and NAND2 is for combining R with Enable. NAND3 takes (Q_N) and the output of NAND1 as inputs, then gives Q as output. NAND4 takes Q and the output of NAND2 as inputs, then gives (Q_N) as output.

In Part 3, we firstly implemented a D latch. We had 2 inputs: D respresenting delay and E representing enable. We used 5 NAND gates. In NAND1, we took D and E as inputs. Output of NAND1 was one of the inputs of NAND4, other input was (Q_N). NAND4's output is Q. In NAND2, we took the complement of D: D'. D' and E was the inputs for NAND3. NAND3's output was the input for NAND5, other input was Q. NAND5's output is (Q_N). Then, we implemented a positive edge triggered D flip-flop using 2 D latches and 1 NOT gate. We had 2 inputs, D representing delay and CLK representing clock. In the NOT gate, we took the complement of CLK: CLK'. In our first D latch, our inputs were D and CLK'. From this latch we obtained (Q_m), which we used as an input in the second D latch. The other input in the second D latch was CLK, and the outputs are Q and (Q_N).

In Part 3, we implemented a positive edge triggered pulse generator using a circular shift register. We had 3 inputs: LOAD representing load flag, CLK representing clock, and 16-bit input for the loaded value. In this module, we used an always block sensitive to the rising edge of the clock. When the load flag is 1, the 16-bit input value is loaded to a register, output 2. Output 2 exists because the module's final output should be only 1-bit whereas we work with a 16-bit input. Then the most significant bit of the input is loaded into output1. When the load flag is 0, the module does the circular shift operation. Another register, most_significant_bit is defined as the last bit of output 2. Then, output 2 shifts to the left logically (¡¡ operator is used). The first bit of output 2 and the output 1 is now the most significant bit. The reason we defined output1, output2 and most_significant_bit as register is because only the reg data type can be updated in always blocks.

5 CONCLUSION [10 points]

In this experiment, we learned how to implement and examine data storage elements such as latches and flip-flops using Verilog. We learned implementing SR Latch, D Flip Flop, pulse generator, and shift register. Also, we learned to use ”if” block and ”always” block in Verilog.