ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 242E

DIGITAL CIRCUITS LABORATORY HOMEWORK REPORT

HOMEWORK NO	:	2
HOMEWORK DATE	:	06.04.2023
LAB SESSION	:	FRIDAY - 10.30
GROUP NO	:	G3

GROUP MEMBERS:

150200097 : Ahmet Emre Buz

150200010 : Mustafa Can Çalışkan

SPRING 2023

Contents

1 INTRODUCTION

In this experiment, we implemented and simulated desired designs using Verilog.

2 PRELIMINARY

2.1

Flip flops are useful in digital electronics because they allow us to store and process information. We can use them to remember and recall data, to sequence events, and to generate timing signals. Flip flops are used in many different applications, such as computers, telecommunications, and control systems.

2.2

Latches and flip flops are both circuits used for storing and processing information. The main difference is that latches respond to input changes whenever the clock is enabled, while flip flops respond only on clock transitions. Latches are transparent to input changes when the clock is enabled, while flip flops are not. Flip flops consume less power than latches. The choice of which one to use depends on the specific application and design requirements.

2.3

An SR-latch is an electronic circuit that has two inputs (S and R) and two outputs (Q and Q-bar). It can be used for storing and processing digital signals. When both S and R inputs are low, the circuit holds its previous state. When S is high and R is low, Q goes high, and when R is high and S is low, Q goes low. When both inputs are high, the circuit is in an undefined state. The S and R inputs allow the user to set or reset the latch, respectively.

2.4

S	R	Q(t)	$\overline{Q(t)}$	Q(t+1)	$\overline{Q(t+1)}$
0	0	0	1	0	1
0	0	1	0	1	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	0
1	0	1	0	0	1
1	1	0	1	0	1
1	1	1	0	0	1

Table 1: Truth table for an SR latch without an Enable input

2.5

E	S	R	Q(t)	$\overline{Q(t)}$	Q(t+1)	$\overline{Q(t+1)}$
0	-	-	0	1	0	1
1	0	0	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	1
1	0	1	1	0	0	1
1	1	0	0	1	1	0
1	1	0	1	0	1	0
1	1	1	0	1	0	1
1	1	1	1	0	0	1

Table 2: Truth table for an SR latch with an Enable input

2.6

D	CLK	Q(t)	Q(t+1)
0	0	0	0
0	1	0	0
1	0	0	0
1	1	0	1
1	0	1	1
1	1	1	1
0	0	1	1
0	1	1	1

Table 3: Truth table for a D flip flop

2.7

J	K	CLK	Q(t)	$\overline{Q(t)}$	Q(t+1)	$\overline{Q(t+1)}$
0	0	-	0	1	0	1
0	0	-	1	0	1	0
0	1	0	0	1	0	1
0	1	0	1	0	1	0
1	0	0	0	1	1	0
1	0	0	1	0	0	1
1	1	0	0	1	1	0
1	1	0	1	0	0	1
0	1	1	0	1	1	0
0	1	1	1	0	0	1
1	0	1	0	1	0	1
1	0	1	1	0	1	0
1	1	1	0	1	0	1
1	1	1	1	0	1	0

Table 4: Truth table for a JK flip flop

3 EXPERIMENT

3.1 SR Latch

SR Latch (only NAND Gates)

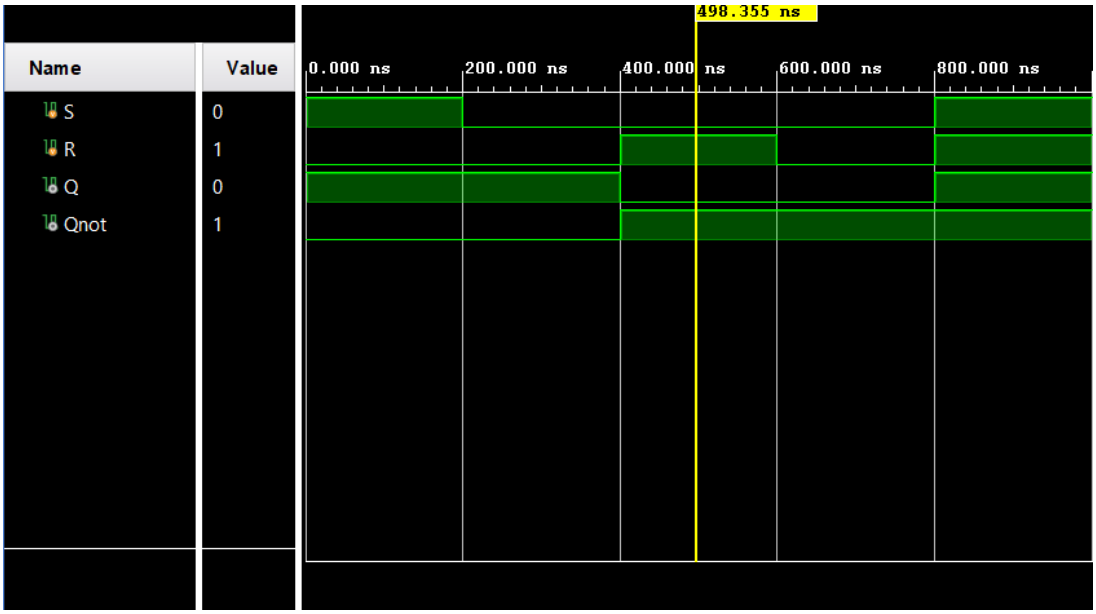

Figure 1: SR Latch

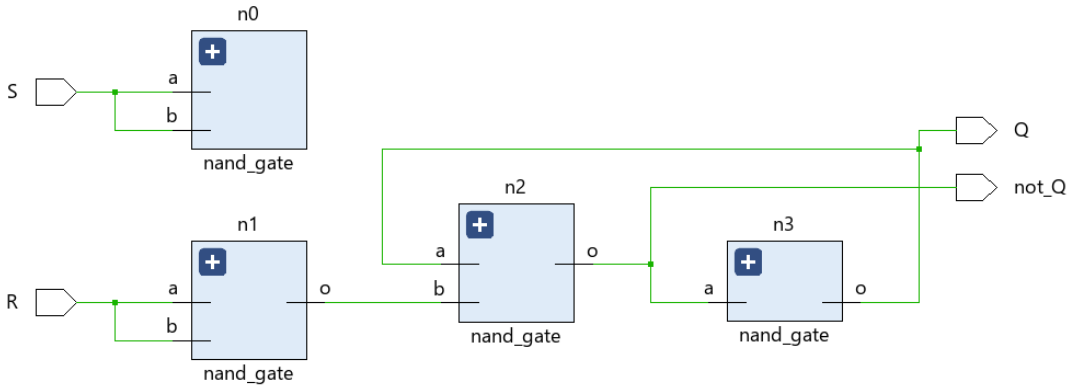

Figure 2: SR Latch RTL

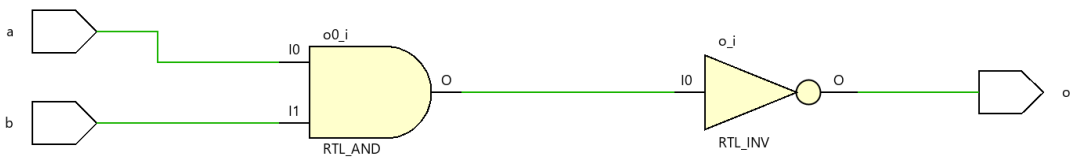

Figure 3: NAND Gate RTL

When both S and R inputs are active at the same time (i.e., S=1 and R=1), the SR latch enters an undefined state. This happens because the NAND gates in the circuit are not designed to handle this situation, and the outputs of the gates may oscillate or settle into unpredictable states.

3.2 SR Latch With Enable Input

SR Latch with Enable input (only NAND Gates)

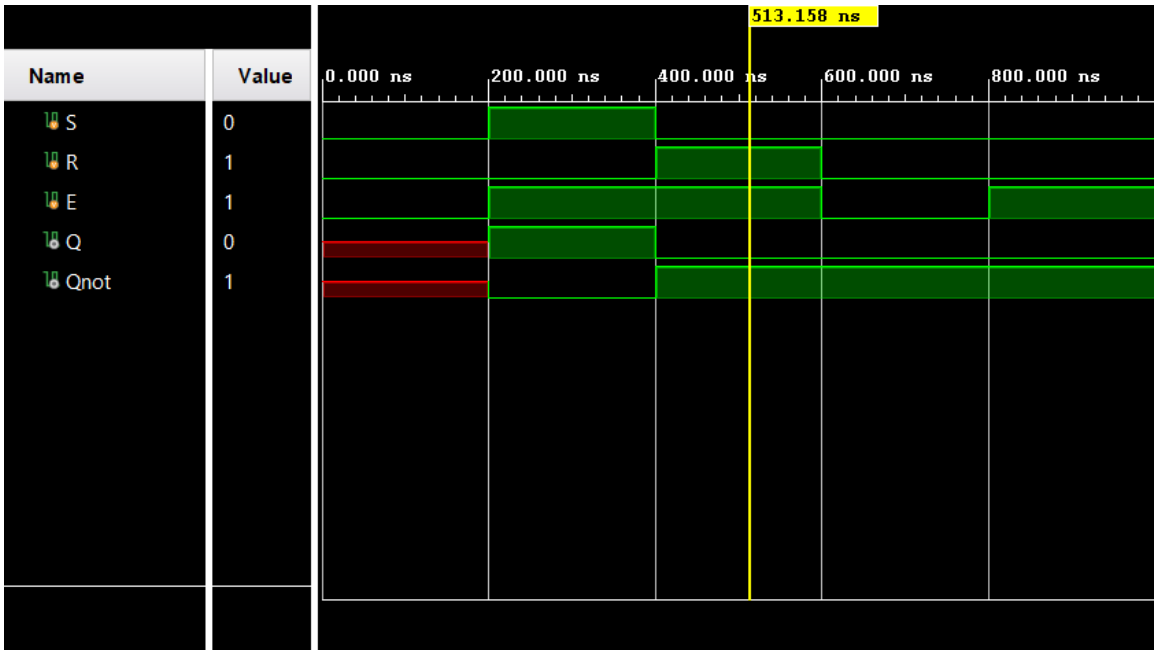

Figure 4: SR Latch with Enable Input

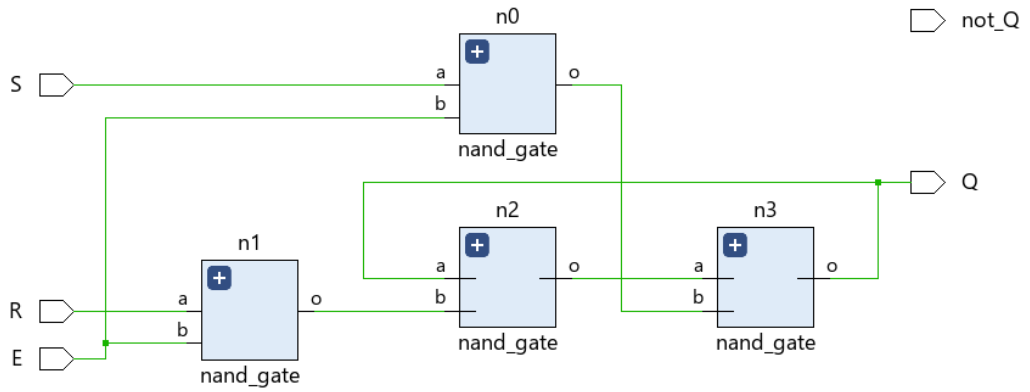

Figure 5: SR Latch with Enable Input RTL

In an SR latch with an enable input, the latch is only active when the enable input is set to a certain state (for our implementation 1). When the enable input is low or 0, the

latch is disabled and does not respond to changes in the S and R inputs.

On the other hand, an SR latch without an enable input is always active and responds to changes in the S and R inputs at all times. This means that the latch can change state even if it's not intended to do so, which can lead to unpredictable behavior.

3.3 D Flip-Flop from D-Latches

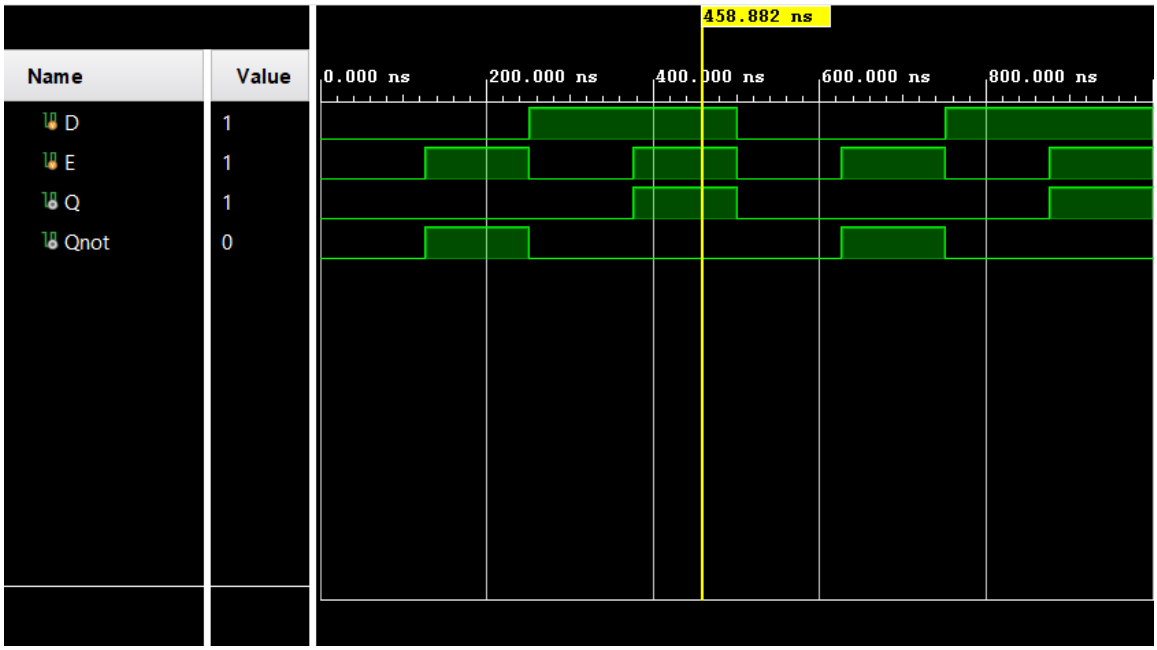

Figure 6: D Latches

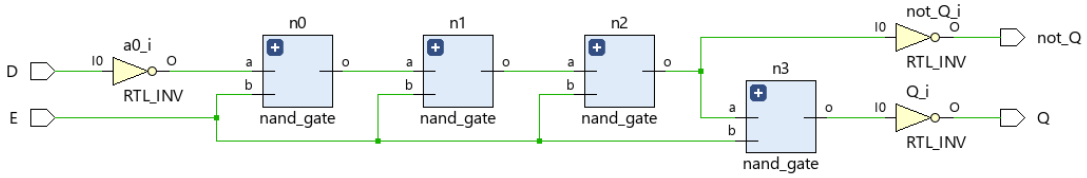

Figure 7: D Latch RTL

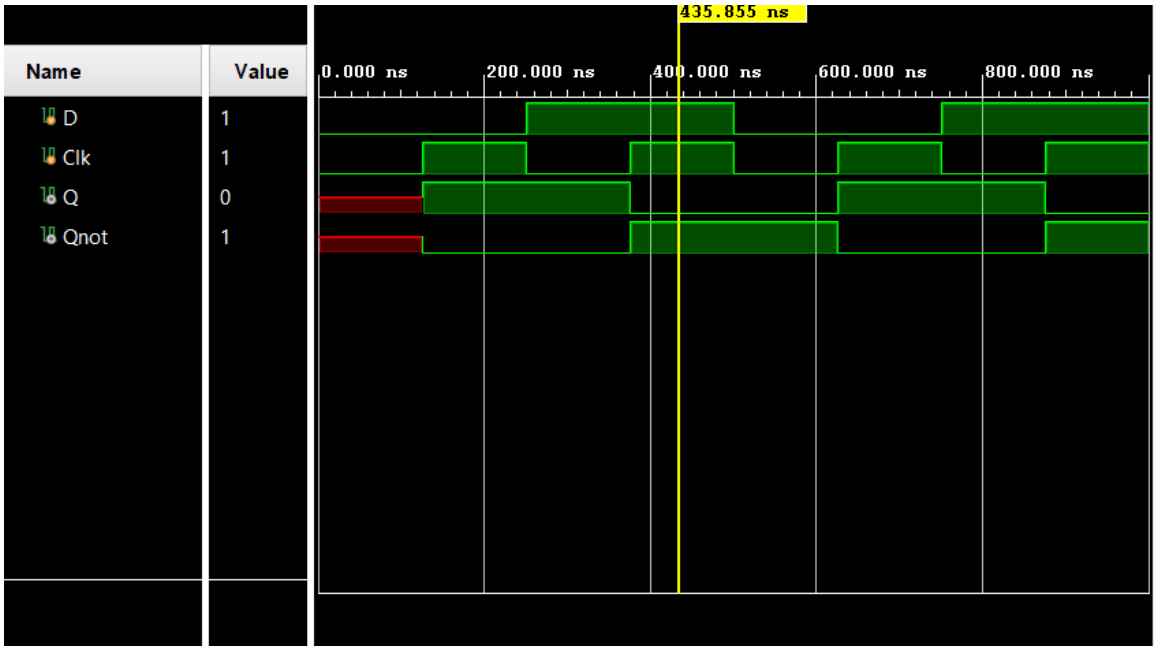

Figure 8: D Flip-Flop

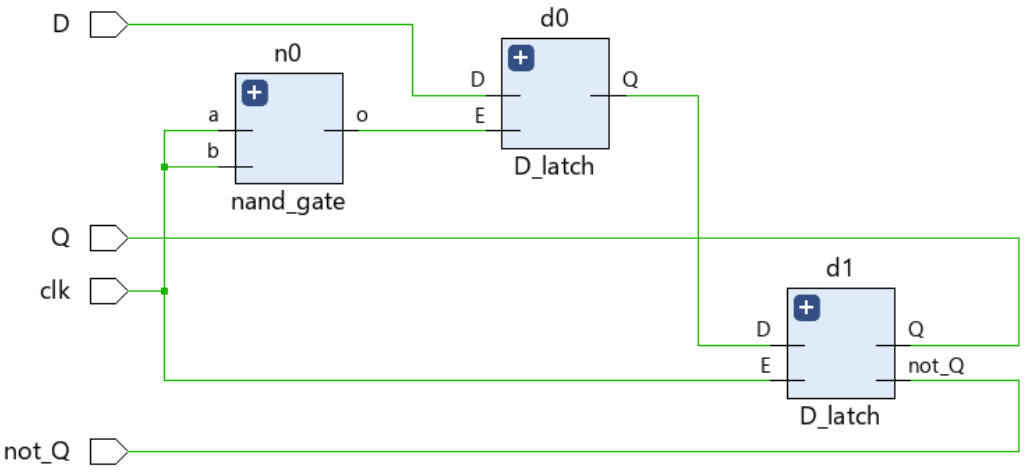

Figure 9: D Flip-Flop RTL

3.4 JK Flip-Flop

JK Flip-Flop (only NAND gates)

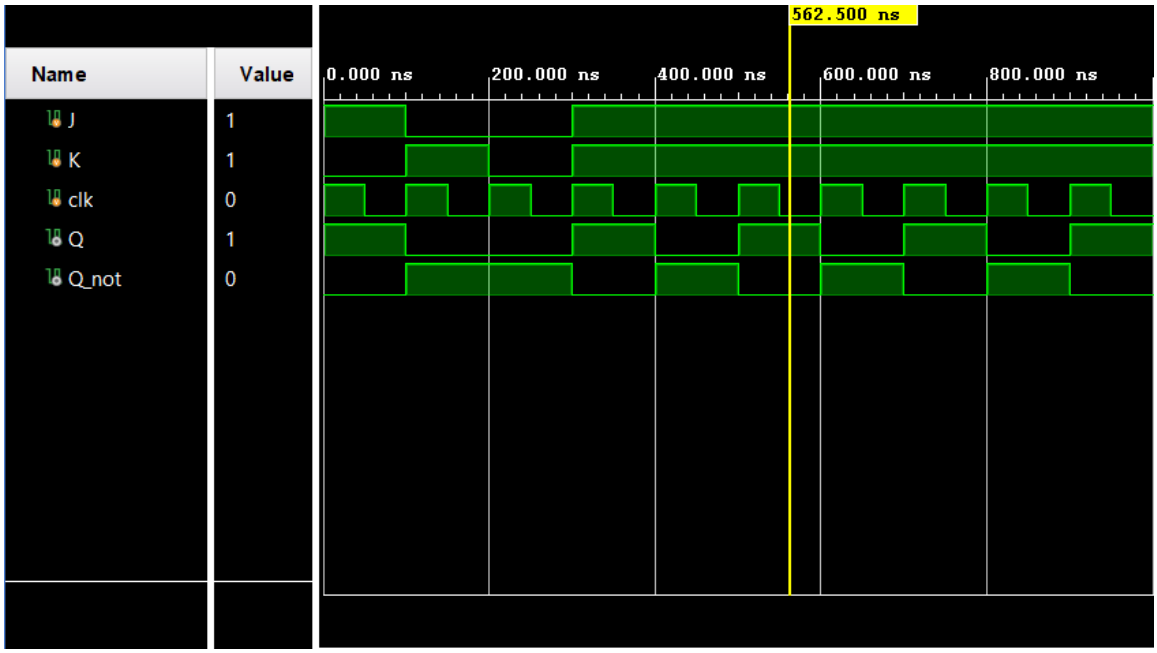

Figure 10: JK Flip-Flop

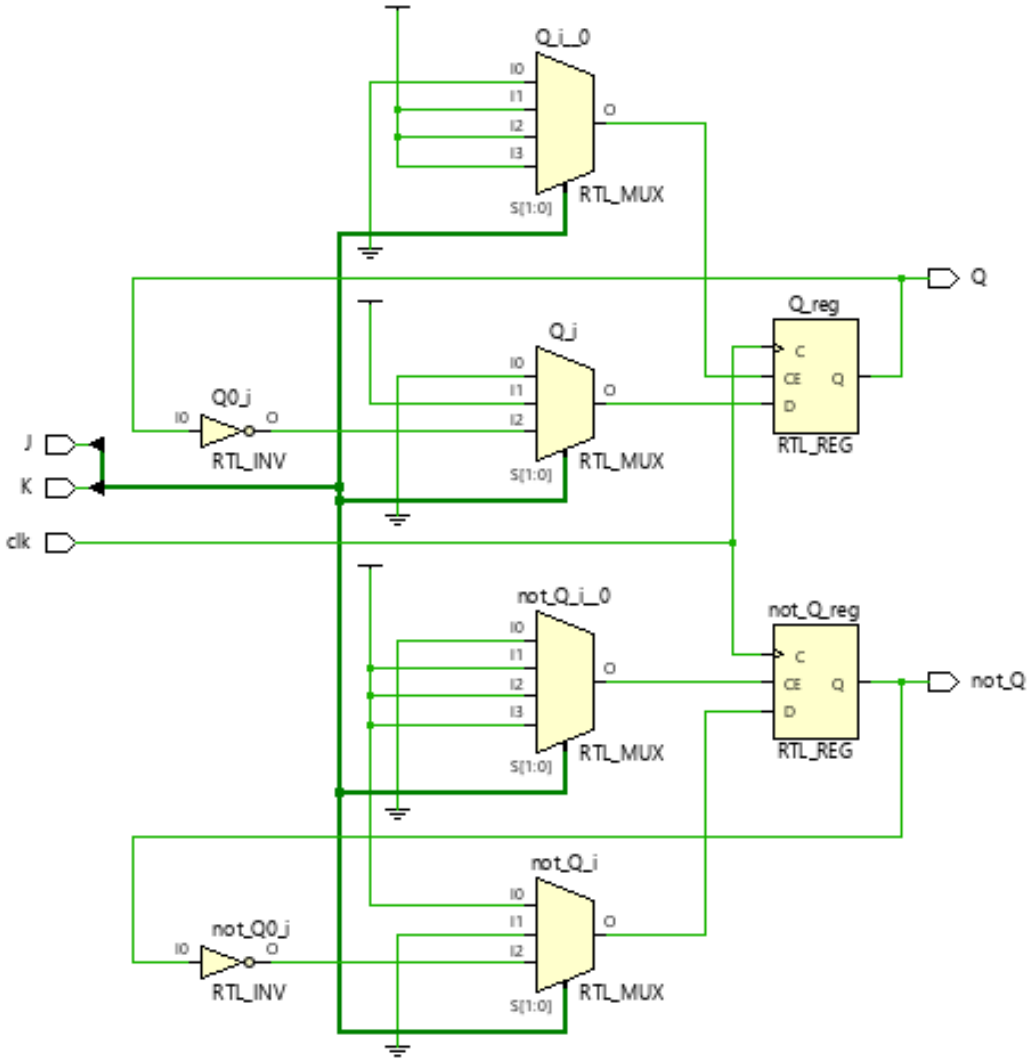

Figure 11: JK Flip-Flop RTL

3.5 Asynchronous Up Counter

Clock	J	K	Q_3	Q_2	Q_1	Q_0	Q_3^{next}	Q_2^{next}	Q_1^{next}	Q_0^{next}
0	0	1	0	0	0	0	0	0	0	1
1	1	1	0	0	0	1	0	0	1	0
2	1	1	0	0	1	0	0	0	1	1
3	1	1	0	0	1	1	0	1	0	0
4	1	1	0	1	0	0	0	1	0	1
5	1	1	0	1	0	1	0	1	1	0
6	1	1	0	1	1	0	0	1	1	1
7	1	1	0	1	1	1	1	0	0	0
8	1	1	1	0	0	0	1	0	0	1
9	1	1	1	0	0	1	1	0	1	0
10	1	1	1	0	1	0	1	0	1	1
11	1	1	1	0	1	1	1	1	0	0
12	1	1	1	1	0	0	1	1	0	1
13	1	1	1	1	0	1	1	1	1	0
14	1	1	1	1	1	0	1	1	1	1
15	1	1	1	1	1	1	0	0	0	0
16	1	1	0	0	0	0	0	0	0	1

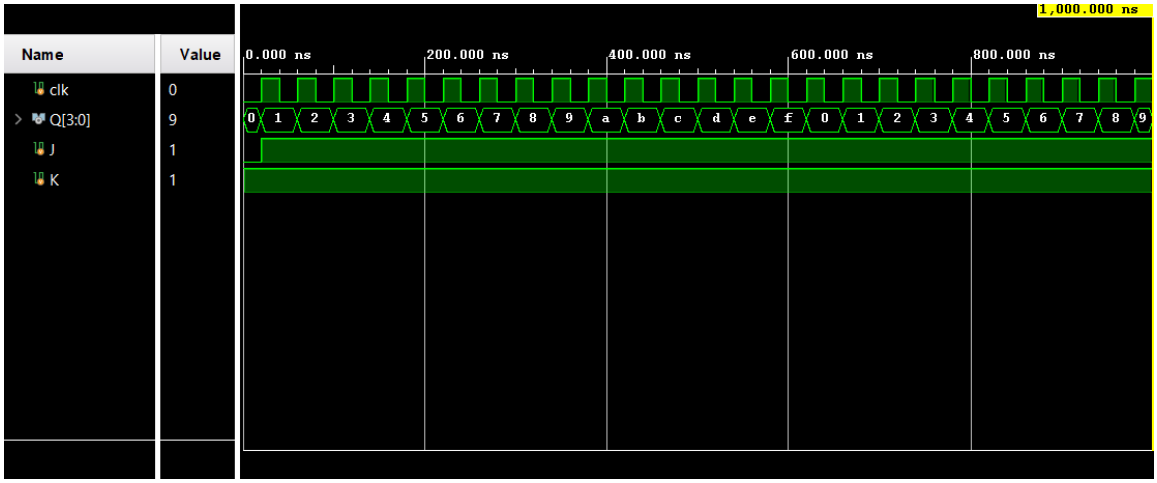

Figure 12: Asynchronous Up Counter

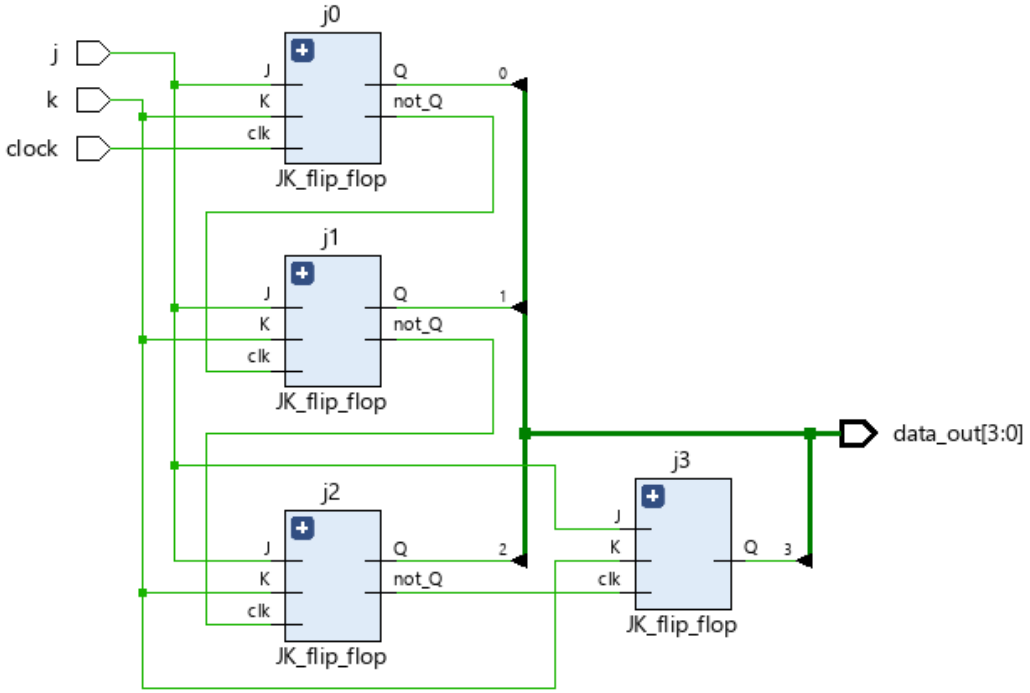

Figure 13: Asynchronous Up Counter RTL

3.6 Synchronous Up Counter

Clock	Q_3	Q_2	Q_1	Q_0	Q_3^+	Q_2^+	Q_1^+	Q_0^+
0	0	0	0	0	0	0	0	1
1	0	0	0	1	0	0	1	0
0	0	0	1	0	0	0	1	1
1	0	0	1	1	0	1	0	0
0	0	1	0	0	0	1	0	1
1	0	1	0	1	0	1	1	0
0	0	1	1	0	0	1	1	1
1	0	1	1	1	1	0	0	0
0	1	0	0	0	1	0	0	1
1	1	0	0	1	1	0	1	0
0	1	0	1	0	1	0	1	1
1	1	0	1	1	1	1	0	0
0	1	1	0	0	1	1	0	1
1	1	1	0	1	1	1	1	0
0	1	1	1	0	1	1	1	1
1	1	1	1	1	0	0	0	0

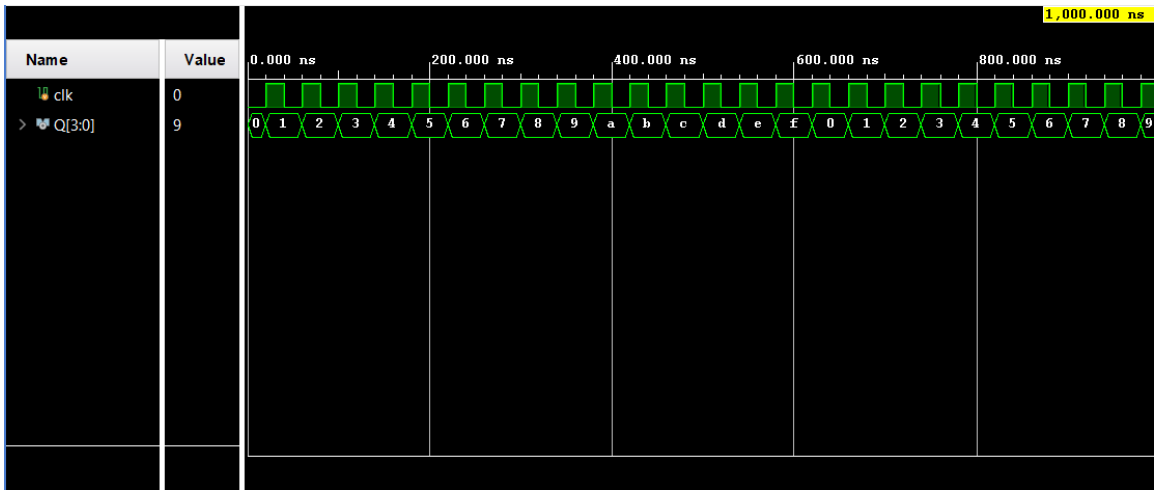

Figure 14: Synchronous Up Counter

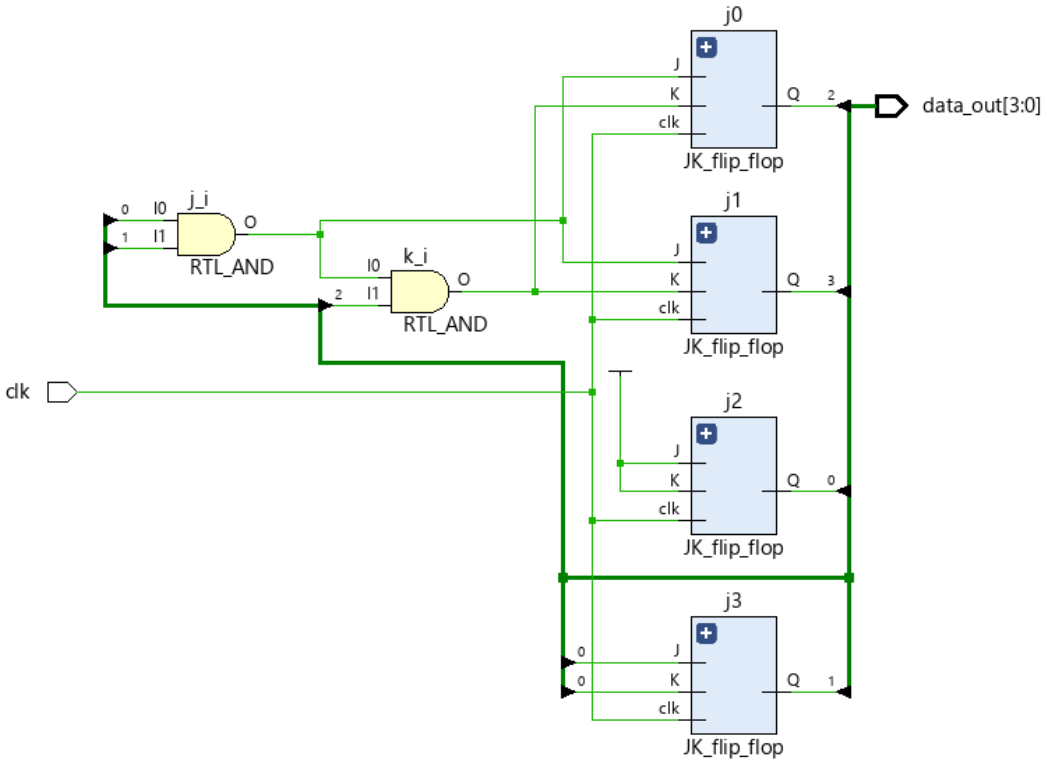

Figure 15: Synchronous Up Counter RTL

3.7 Pulse Generator Using Shift Register

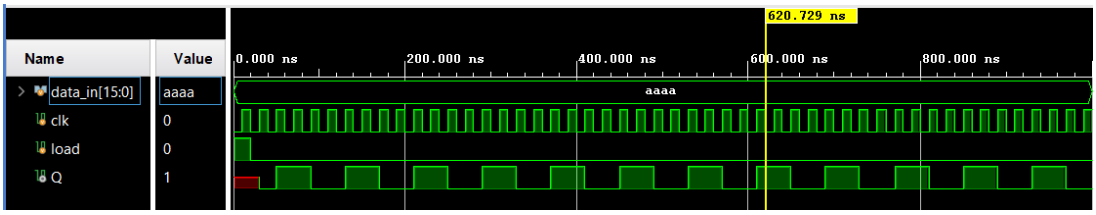

Figure 16: 1/2 Pulse-Gap Duration Rate

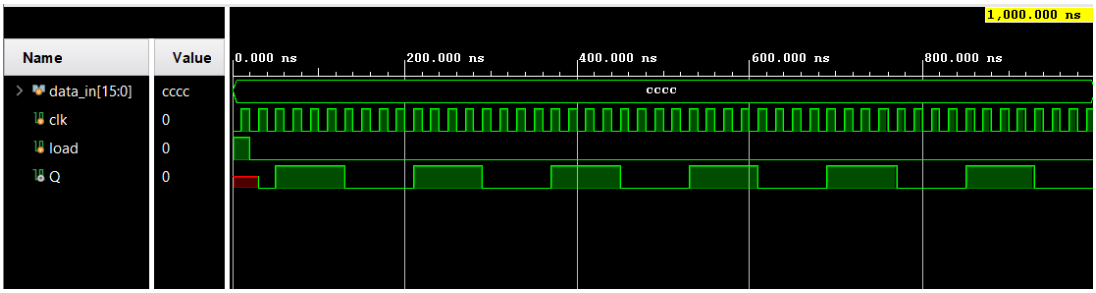

Figure 17: 1/4 Pulse-Gap Duration Rate

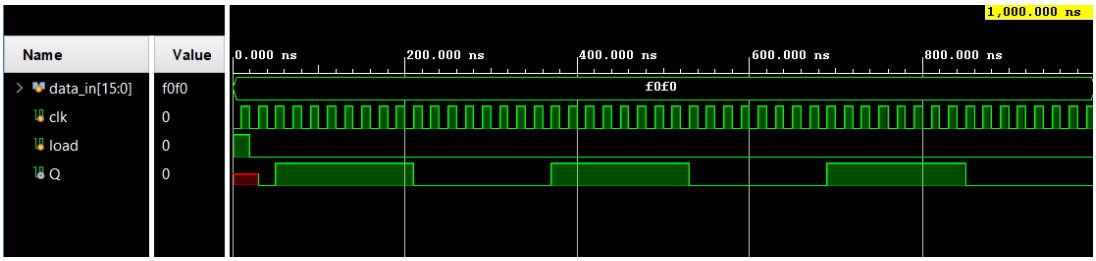

Figure 18: 1/8 Pulse-Gap Duration Rate

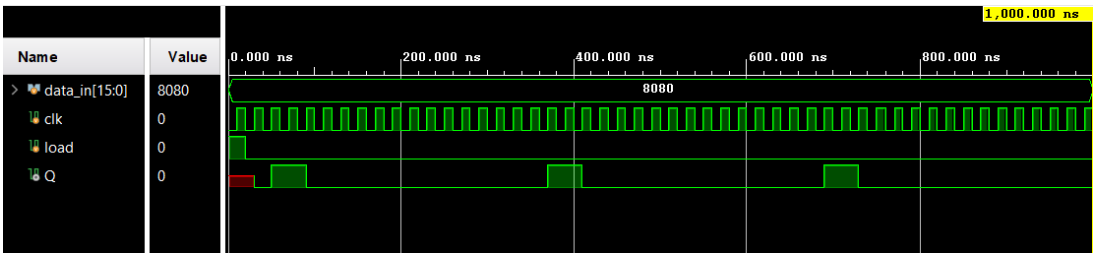

Figure 19: 1/7 Pulse-Gap Duration Rate

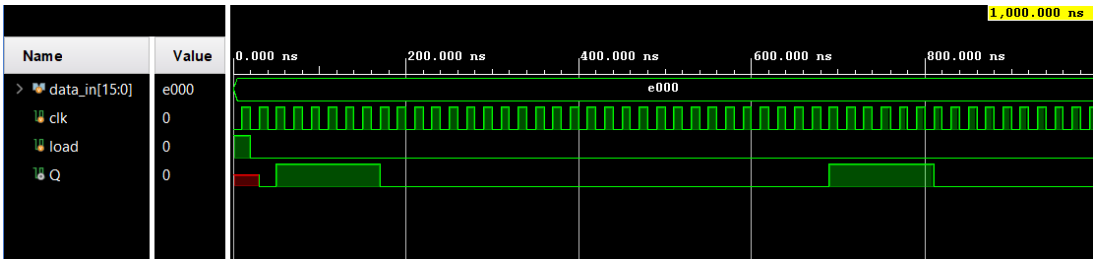

Figure 20: 3/13 Pulse-Gap Duration Rate

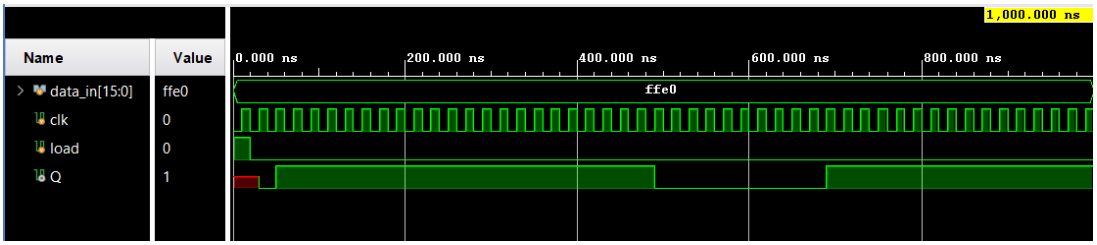

Figure 21: 11/5 Pulse-Gap Duration Rate

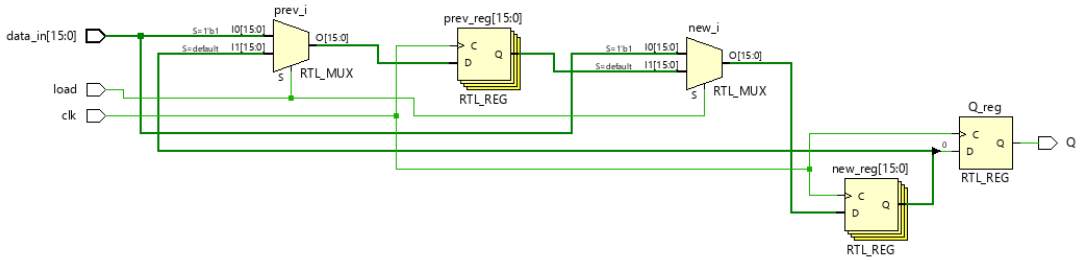

Figure 22: Shift Register RTL

4 CONCLUSION

In this experiment, we implemented and simulated desired designs using Verilog.

REFERENCES