ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 242E

DIGITAL CIRCUITS LABORATORY HOMEWORK REPORT

HOMEWORK NO	:	2
HOMEWORK DATE	:	27.04.2022
LAB SESSION	:	FRIDAY - 14.00
GROUP NO	:	G25

GROUP MEMBERS:

150210710 : FATİH BASKIN

150190102 : ELVAN TEKE

SPRING 2022

Contents

1 INTRODUCTION

In this homework we have implemented counters, cyclic shift registers with load inputs to be able to load certain signals and we implemented flip-flops and latches.
We have learned the internal structure of latches and flip-flops then we implemented them. Also we have gained some fundamental knowledge about PWM signals.

2 PRELIMINARIES

2.1 Flip-flop

A flip flop is a 1-bit memory unit. Output of a flip flop is a function of its inputs and current state. Its output can be changed only when the clock signal is active. It can be used to store state information.

2.2 Latches vs Flip-flops

Output of a latch can be changed at anytime, whereas output of a flip flop can be changed only when the clock signal is active.

2.3 SR-latch

S-R Latch is built by two NAND or two NOR gates. Outputs of each gates are the inputs of the other gates. When S input is equal to 1, it sets the output to 1. When R input is equal to 1, it resets the output to 0. When both inputs are 0, output does not change. Both inputs cannot be 1 at the same time, that input is forbidden.

2.4 Truth Table of SR-latch without Enable

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

2.5 Truth Table of SR-latch with Enable

E	Q(t)	S	R	Q(t+1)
0	0	X	X	0
0	1	X	X	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	invalid
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	invalid

2.6 Truth Table of JK flip flop

Clock	Q(t)	J	K	Q(t+1)
1	0	0	0	0
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

2.7 Truth Table of D flip flop

Clock	Q(t)	D	Q(t+1)
0	0	X	0
0	1	X	1
1	X	0	0
1	X	1	1

3 HOMEWORK

3.1 Part I

In this part, we have been asked to implement SR latch without enable input, using two input NOR gates. We also had to implement those NOR gates.

As can be seen above, our designed NOR gate works as expected, giving high output

Figure 1: RTL design of NOR gate

Figure 2: Simulation results of NOR gate

(C) when both inputs (A and B) are low.

With using NOR gates, we can implement bistable SR latch.

The latch is acting as intended, when S = 1, latch sets to 1, when R = 1, latch sets to zero. S, R = 1 is forbidden case. When S and R is 0, latch holds its previous value. This is the backbone of computer memory systems.

3.2 Part II

In this part we have been asked to implement a SR latch with enable using NAND gates. Firstly we needed to implement the NAND gates. Then using them we implemented

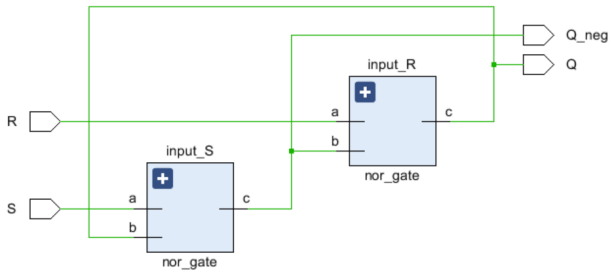

Figure 3: RTL design of no-enable-SR-latch using NOR gates

Figure 4: Simulation results of no-enable-SR-latch

SR latch with enable input.

It is viable to say that the NAND gate is working as expected. It is giving Low output

Figure 5: RTL implementation of NAND gate

in the case of both inputs (A, B) are high.

Figure 6: Simulation of the NAND gate

3.3 Part III

In this part, we have been asked to implement a positive edge triggered D flip-flop. To do that, we first implemented a D latch with enable input.

It can be seen in the simulation result that output is only changing at the rising edge.

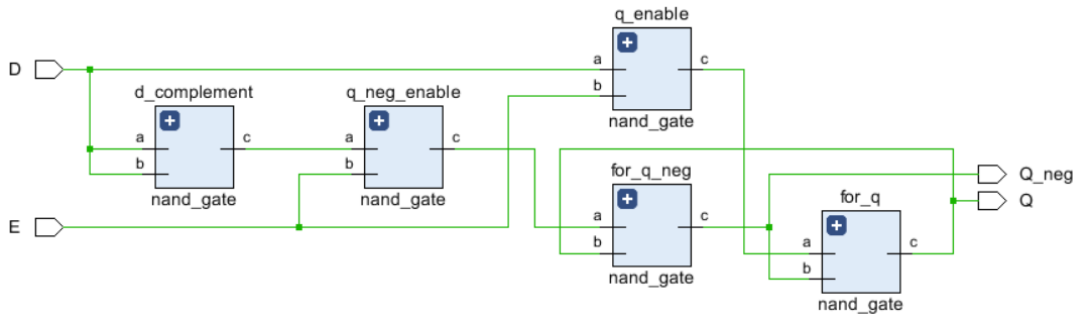

Figure 7: RTL implementation of D latch

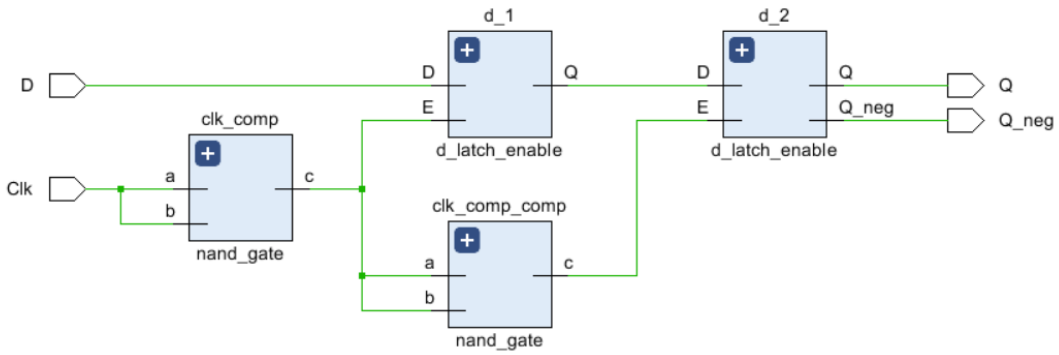

Figure 8: RTL implementation of D flip-flop

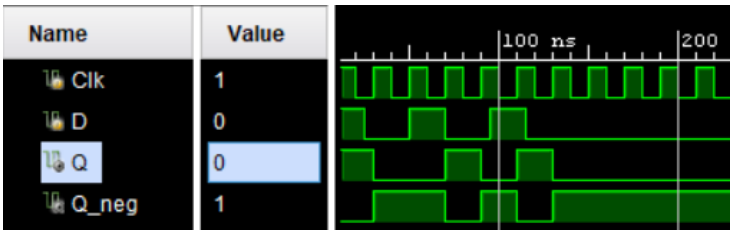

Figure 9: Simulation of D flip-flop

3.4 Part IV

In this part, we have been asked to implement a edge triggered JK flip-flop from SR flip-flop. Firstly, we implemented SR flip-flop using NAND gates and SR latches.
Then, using this SR flip-flop and a couple more NAND gates, we implemented JK flip-flop.

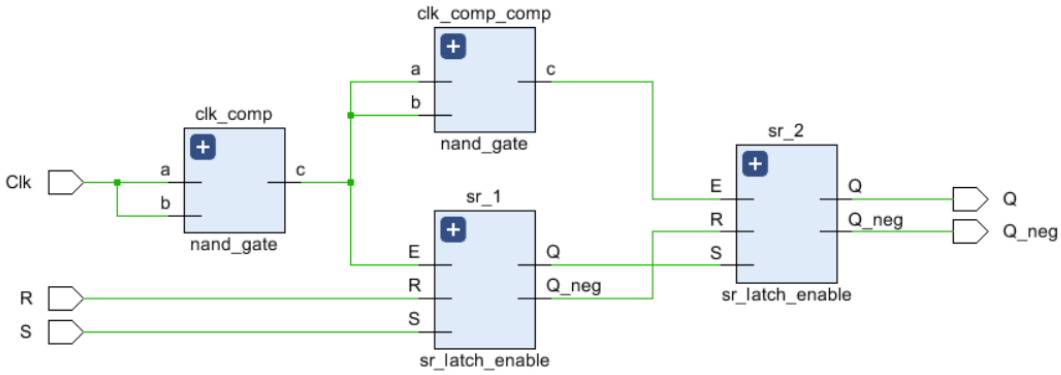

Figure 10: RTL implementation of SR flip-flop

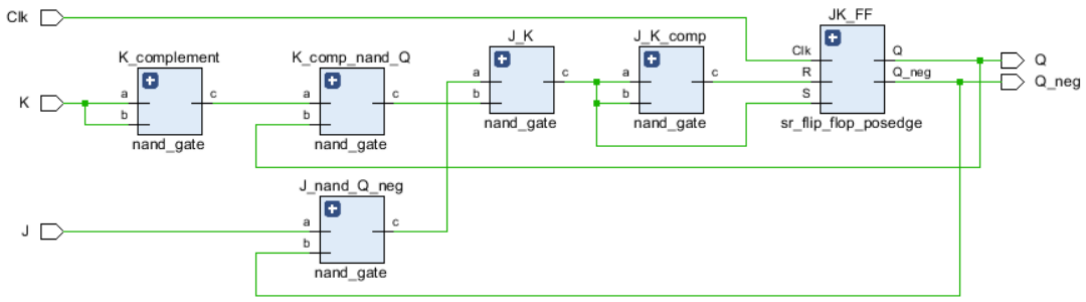

Figure 11: RTL implementation of JK flip-flop

Figure 12: Simulation of JK flip-flop

3.5 Part V

In this part, we have been asked to implement a 4-bit asynchronous up counter using JK flip-flops. We already designed JK flip-flop at the previous part. This counter should count between 0 and 14.

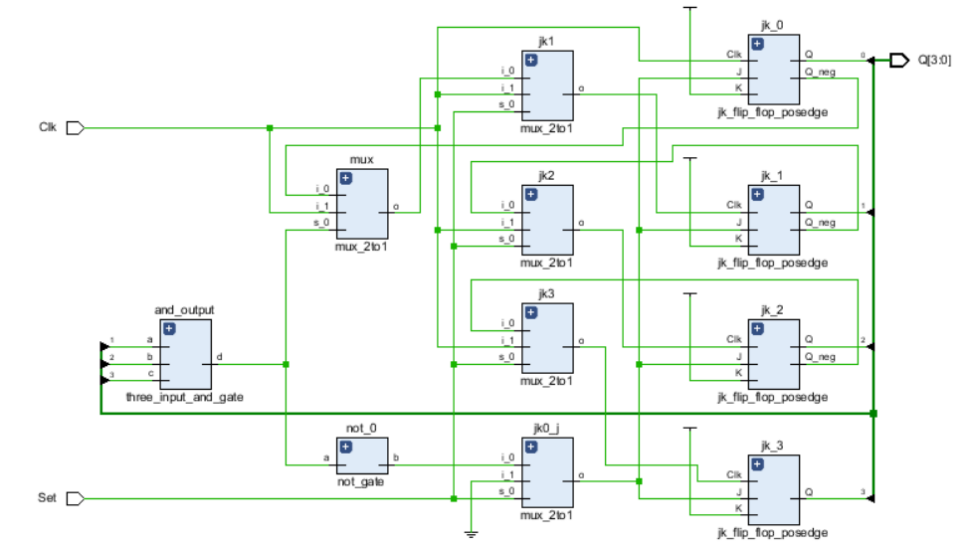

Figure 13: RTL implementation of Asynchronous Up Counter

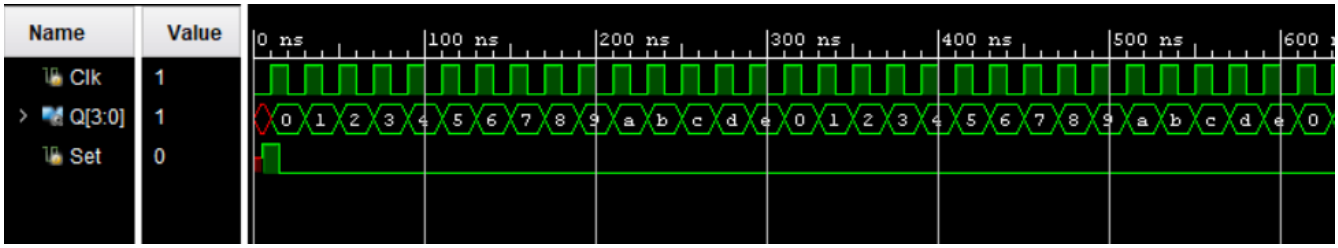

Figure 14: Simulation of Asynchronous Up Counter

Clock	Set	Q3	Q2	Q1	Q0
0	1	-	-	-	-
1	0	0	0	0	0
2	0	0	0	0	1
3	0	0	0	1	0
4	0	0	0	1	1
5	0	0	1	0	0
6	0	0	1	0	1
7	0	0	1	1	0
8	0	0	1	1	1
9	0	1	0	0	0
10	0	1	0	0	1
11	0	1	0	1	0
12	0	1	0	1	1
13	0	1	1	0	0
14	0	1	1	0	1
15	0	1	1	1	0
16	0	0	0	0	0

Truth table of the counter.

3.6 Part VI

In this part, we have been asked to implement a 4-bit synchronous up counter using JK flip-flops. This counter should count between 0 and 14.

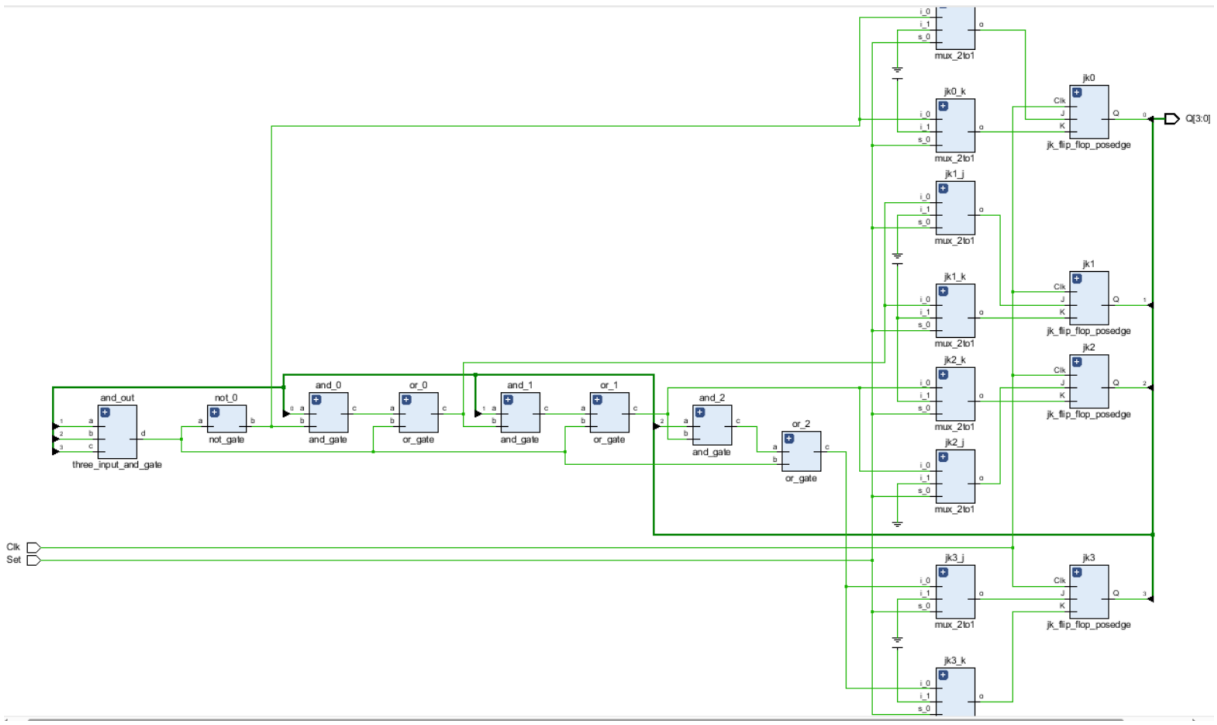

Figure 15: RTL implementation of Synchronous Up Counter

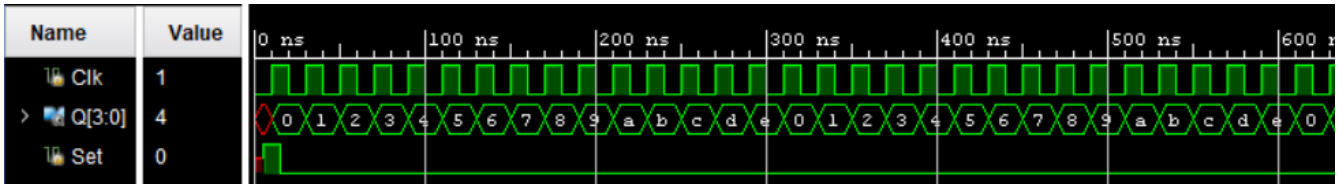

Figure 16: Simulation of Synchronous Up Counter

Clock	Set	Q3	Q2	Q1	Q0
0	1	-	-	-	-
1	0	0	0	0	0
2	0	0	0	0	1
3	0	0	0	1	0
4	0	0	0	1	1
5	0	0	1	0	0
6	0	0	1	0	1
7	0	0	1	1	0
8	0	0	1	1	1
9	0	1	0	0	0
10	0	1	0	0	1
11	0	1	0	1	0
12	0	1	0	1	1
13	0	1	1	0	0
14	0	1	1	0	1
15	0	1	1	1	0
16	0	0	0	0	0

Truth table of the counter.

3.7 Part VII

In part 7, we have been asked to implement a circuit with 8 registers which can do circular shift with load ability so we would able to implement certain signals. We were liberal about using reg structure in Vivado so we used it to implement flip flops. We were asked to implement six different signals using these registers.

1. with the 1/2 frequency of clock signal
2. with the 1/4 frequency of clock signal
3. with the 1/8 frequency of clock signal
4. with 1/3 pulse-gap duration rate
5. with 1/7 pulse-gap duration rate
6. with 4/13 pulse-gap duration rate

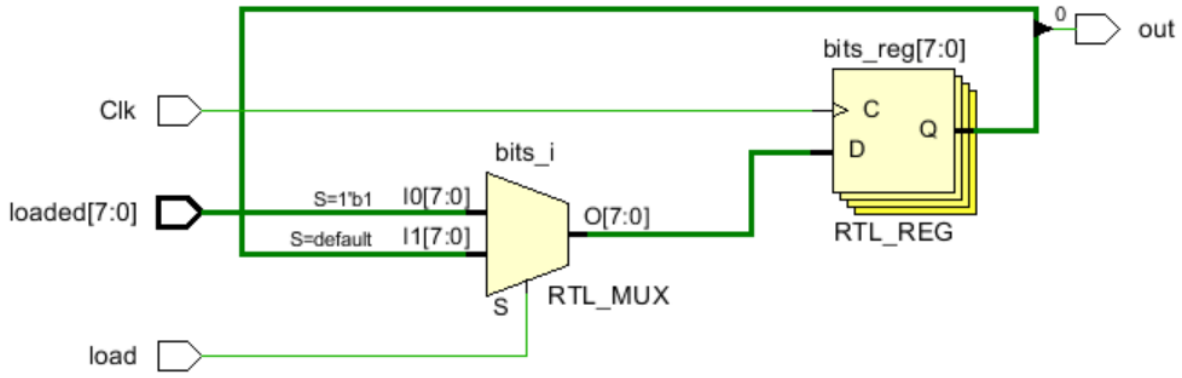

Figure 17: RTL design of Part 7

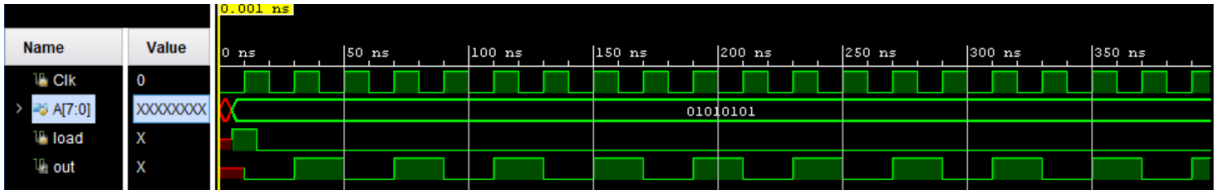

Figure 18: Simulation of a signal that we created with 1/2 frequency of clock signal

To be able to generate a signal with 1/2 frequency of clock signal, we had to load registers with bit values 01010101. Output would be high in one clock cycle and low in next clock cycle, making the signal's period two clock cycle therefore frequency 1/2 of the clock cycle.

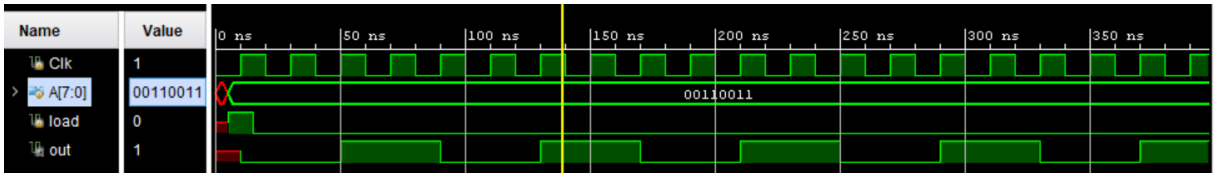

Figure 19: Simulation of a signal that we created with 1/4 frequency of clock signal

To be able to generate a signal with 1/4 frequency of clock signal, we had to load registers with bit values 00110011. Output would be low in two consecutive clock cycles and then high in next two consecutive clock cycles, making the signal's period four clock cycle therefore frequency 1/4 of the clock cycle.

To be able to generate a signal with 1/8 frequency of clock signal, we had to load registers

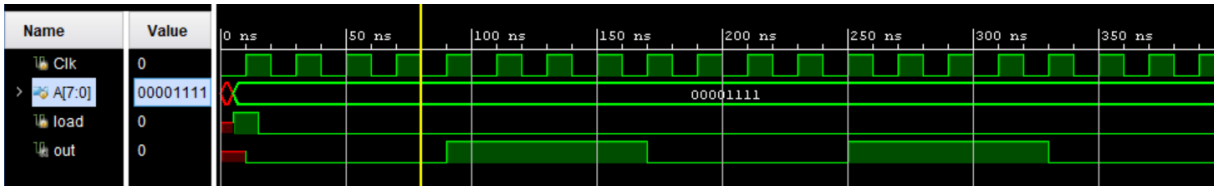

Figure 20: Simulation of a signal that we created with 1/8 frequency of clock signal

with bit values 00001111. Output would be low in four consecutive clock cycles and then high in next four consecutive clock cycles, making the signal's period eight clock cycle therefore frequency 1/8 of the clock cycle.

By default, PWM (Pulse Width Modulation) is used to send different levels of energy

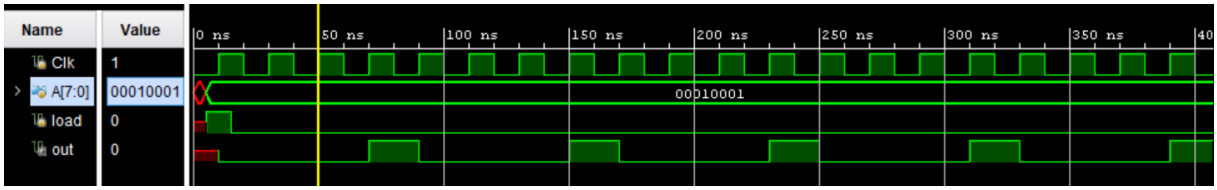

Figure 21: Simulation of a signal that we created with 1/3 pulse-gap duration rate

without changing the voltage. For PWM signal, for sending certain amount of power, they send intermittent power. In this case we were asked to implement a signal with 1/3 pulse-gap ratio. To achieve this we have sent load signal as 00010001 so for every three consecutive clock signals we have sent output as low then for the next clock cycle we have sent output as high, so the pattern was like three low one high for every four clock cycles.

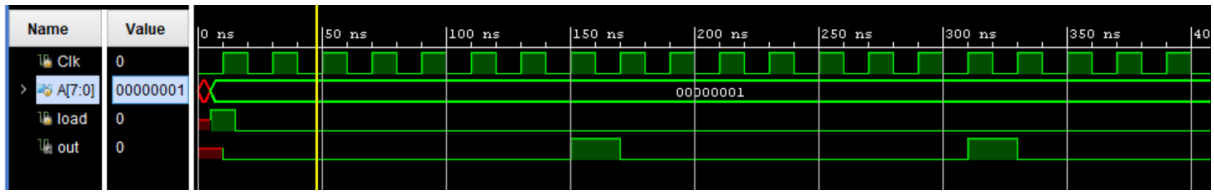

Figure 22: Simulation of a signal that we created with 1/7 pulse-gap duration rate

We were asked to implement a signal with 1/7 pulse-gap ratio. To achieve this we have sent load signal as 00000001 so for every seven consecutive clock signals we have sent output as low then for the next clock cycle we have sent output as high, so the pattern was like seven low one high for every eight clock cycles.

Unfortunately for the 4/13 pulse-gap ratio signal, we were not able to formulate a solution using eight registers, it would require seventeen registers since it was a signal consisting of 17 clock cycles.

4 DISCUSSION

In this homework assignment we learned the design of latches and flip-flops, how do they work and also we implemented them. Then with that knowledge on hand, we implemented synchronous and asynchronous counters. Then lastly we have learned about PWM signals and implemented some PWM signals using cyclic shifting registers.

5 CONCLUSION

In the end, we had a profound knowledge about latches and flip-flops. We had difficulties with implementing counters in Part 5 and Part 6 since our flip flops didn't have reset input so when we implemented them, they were acting indeterminate because we didn't have a set value initially. So we have solved this by using multiplexers, we sent necessary inputs (JK = 01) and clock cycle when Set was high so our flip flops would set their values to zero and start counting without acting indeterminate.

Also in the last part, we had difficulty with implementing 4/13 pulse-gap ratio signal since to be able to implement it we would require seventeen registers but our circuit only had eight so we weren't able to implement that signal with our current circuit. If we had a circuit with 17 registers, we would be able to.

REFERENCES