ISTANBUL TECHNICAL UNIVERSITY

COMPUTER ENGINEERING DEPARTMENT

BLG 242E

DIGITAL CIRCUITS LABORATORY EXPERIMENT REPORT

EXPERIMENT NO	:	4
EXPERIMENT DATE	:	07.04.2021
LAB SESSION	:	FRIDAY - 10.30
GROUP NO	:	G1

GROUP MEMBERS:

150170908 : MUHAMMAD RIZA FAIRUZZABADI

070170364 : AİŞE HÜMEYRA BOZ

SPRING 2021

Contents

1 INTRODUCTION

In this project, we were required to complete two distinct tasks, preliminary and experiment. First in preliminary part, we recalled latches and flip-flops and their differences respectively. Then, we revised SR Latch, SR latch with Enable input and D Flip-Flop alongside with their truth tables. In the experiment part, we implemented all modules. Then we tested them with the given values.

2 MATERIALS AND METHODS

2.1 MATERIALS

Tools Used

- Vivado Design Suite - Xilinx
- Latex (overleaf.com)
- Logisim

Firstly we recalled the aforementioned topics and noted the important characteristics we have to keep in mind in the implementation of the modules. We then implement and program the logic circuits alongside their NOT,AND,OR gates modules and specific modules that constructed for this experiment in Vivado Design Suite, lastly we used overleaf.com to prepare the report document in LaTex.

2.2 PRELIMINARY

In the preliminary part, we recalled and revised what we learnt in Digital Circuits course. There are some features we have to keep in mind in the implementation of the latches and flip-flops:

- Basic memory unit keeps on-bit of memory. Flip-flop is a memory unit that is controlled by a clock signal. It is useful because clock determines and controls the circuits structure. It decides when the next output occurs and accordingly output may change.
- While both latches and flip-flops being sequential circuits, they vary in some aspects. Latch do not have a control input which controls the circuit. Its next state only depends on its current state. However, for flip-flops same do not apply.It has two stable states with feedback and control input (clock).

- SR latch has two stable states. The input S 'sets' the output as 1, on the other hand the input 'R' 'resets' the output to 0.

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

-

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

-

D	CLK	Q(t+1)	Q'(t+1)
0	RE*	0	1
1	RE*	1	0
X	0	Q(t)	Q'(t)
X	1	Q(t)	Q'(t)

2.3 PART 1

In the first part, we were asked to design an SR Latch module using only NAND Gates. Since we were allowed to use the default operations, so we thought firstly that we did not have to use NAND gate module we designed and used in previous experiments, but eventually we realized we had to. The design is rather simple, we have two inputs of S (Set) and R (Reset), and two outputs of Q and Q_neg. Our initial design in Logisim is as shown in Figure 1. The NAND operation between S' and Q_neg gives Q, and the NAND operation between R' and Q gives Q_neg. The fact that both NAND take the Q's values cause the first outputs of this NAND to be X, expectedly. Truth table is as shown in the preliminary part. As can be seen in the simulation result in Figure 3, when both S and R values are 1, both Q and Q_neg outputs become 1, indicating forbidden inputs are applied. Note that the inputs in the Logisim Initial Design of the SR Latch are

supposedly negates, unlike the Elaborated Design (Figure) which values are negated in the process, not before. Lastly, it can be seen in the simulation result that the expected Q values are produced according to the truth table, hence proving the correctness of our implementation.

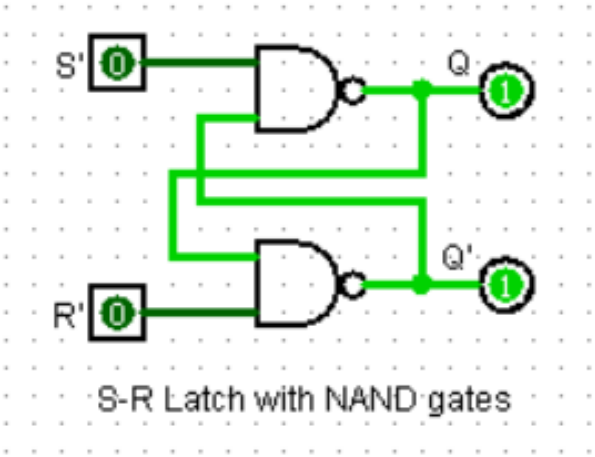

Figure 1: Logisim Initial Design of SR Latch

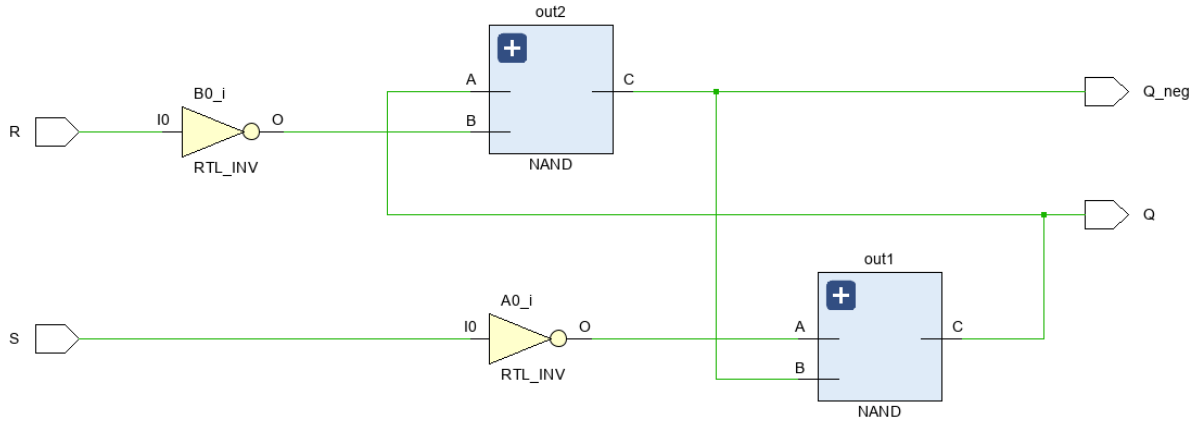

Figure 2: Elaborated Design Schematic of SR Latch

2.4 PART 2

In the second part, we were asked to design an SR Latch module using only NAND Gates, this time with Enable inputs. Nowwe have three inputs of S, R, and E, and two outputs of Q and Q_neg. Our initial design in Logisim is as shown in Figure 4. The NAND operation between S' and E gives Qi1, and the NAND operation between E and R gives Qi2. Then we did the same operations we did in previous S-R Latch but this time with Qi1 and Qi2 inputs instead of S and R. Truth table is as shown in the preliminary

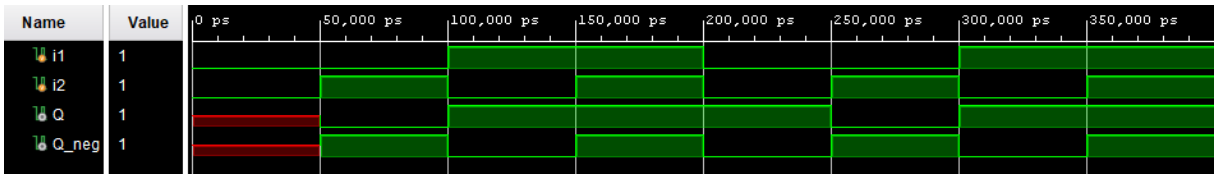

Figure 3: Simulation of SR Latch

part. As can be seen in the simulation result in Figure 6, when both S and R values are 1, both Q and Q_neg outputs become 1, indicating forbidden inputs are applied. The first outputs are again X due to the nonexistence of previous Q values. The Elaborated Design Schematic is as shown in Figure 5. Lastly, it can be seen in the simulation result that the expected Q+1 and Q values are produced according to the truth table, hence proving the correctness of our implementation.

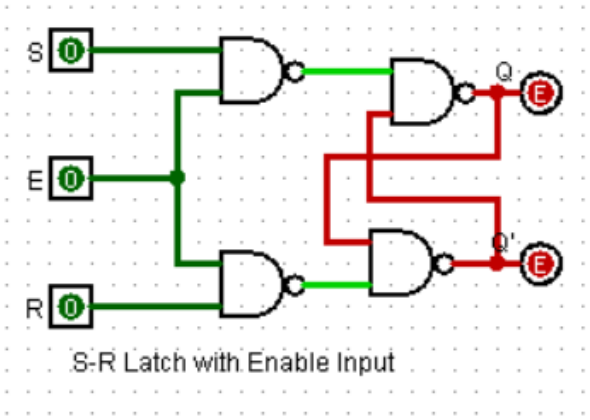

Figure 4: Logisim Initial Design of SR Latch with EN

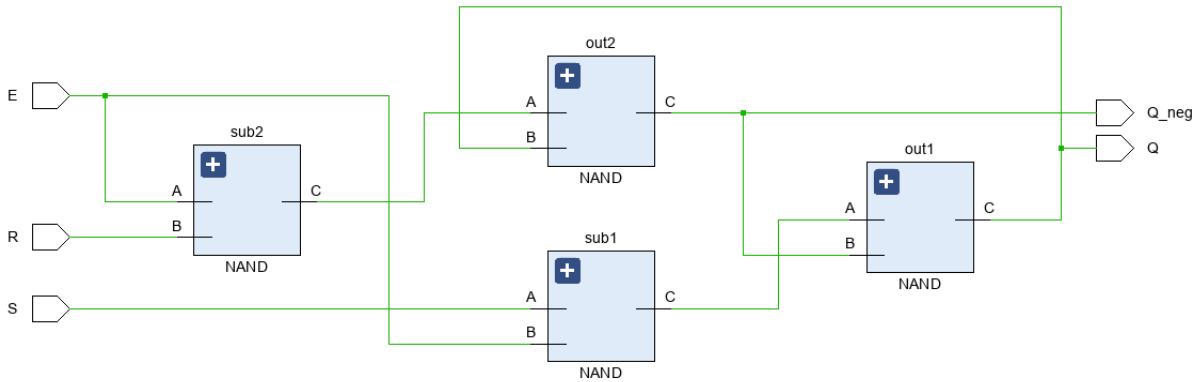

Figure 5: Elaborated Design Schematic of SR Latch with Enable

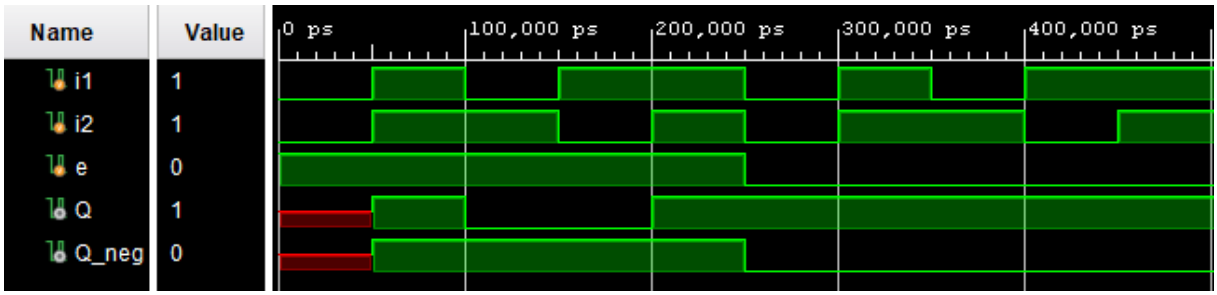

Figure 6: Simulation of SR Latch with Enable

A	B	S	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

2.5 PART 3

In the third part, we were asked to design a D Flip Flop from D-Latches. Since we have not designed a D-Latch in the previous part, we at first designed an D-Latch before implementing the flip-flop. The initial logisim design of the D-Latch and the Flip Flop is as shown in Figure 7 and 8. The design of the D Latch is similar to the previous S-R Latch with EN, with the only different being instead of R, we have a negated D as the third output of the module. Meanwhile the D Flip Flop is implemented by using two of D Latch, which the output of the first being the input D of the second Latch, and with a clock (negated once for the first and twice for the second latch) as Enable inputs. The code implementation of the D Flip FLop is simply two instantiations of the D Latch module. Truth table is as shown in the preliminary part. It is worth noting that for this part in particular, we did not implement a simulation according merely to the truth table, but we try to simulate it with such inputs that it would show its validity in different conditions (ex. that the Q value is actually unchanged unless it is supposed to). As can be seen in the simulation result in Figure 9, in the first 150 ns the Q is only implemented and resetted to 0. Within the 4th 50 ns delay, it can be seen that the Q value isn't set to 1 despite the D(i1) input, due to the clock hasn't been HIGH yet. Though in the next 50 ns, Q value is set to 1, indicating the implemenation is true in that case as well. Lastly, it can be seen in the simulation result that the expected Q+1 values are produced after a positive rising edge, indicating the correctness of our implementation in that regard. The Elaborated Design Schematic of D Latch and D Flip Flop are as shown in Figure 9 and 10. Lastly, it can be seen in the simulation result (Figure 11) that the expected Q values are produced in positive rising edge, indicating the correctness of our implementation.

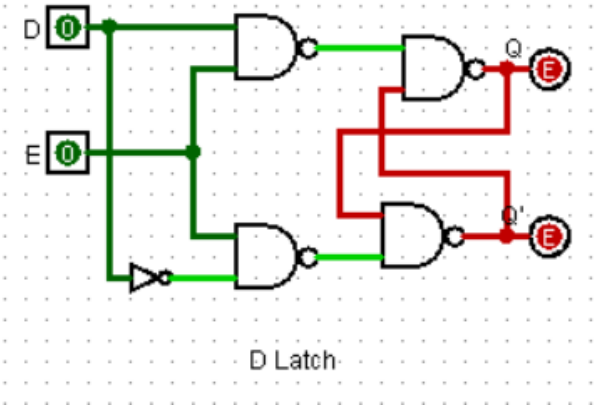

Figure 7: Logisim Initial Design of D Latch

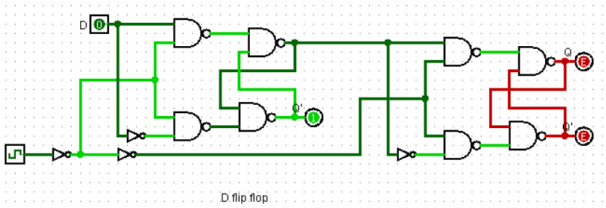

Figure 8: Logisim Initial Design of D Flip Flop

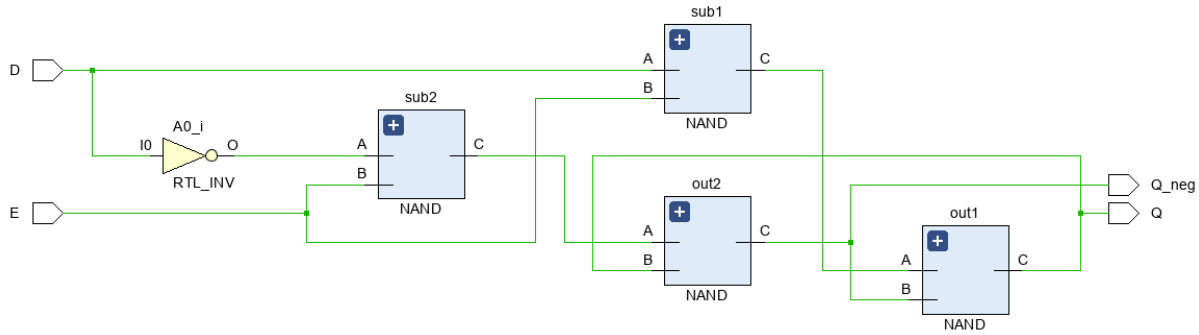

Figure 9: Elaborated Design Schematic of D Latch

3 RESULTS

We completed the first three tasks we were asked in the experiment. Unfortunately we couldn't do the last part due to lack of time caused by the high amount of other works we have for other lectures this week. However, we plan to do part 4 before the demo at least to ask questions and learn more about it. At first we drew the circuits in Logism, then we got to see their results clearly through simulations. Since we draw and design the expressions in Logisim, it was not hard to validate their correctness through simulations. We added required tables, images, circuits to the specific places materials methods part of the experiment. Lastly, we supplemented our images namely elaborated design, simulations and truth tables to the report for each part.

4 DISCUSSION

In preliminary part, we recalled the concept of flip flop and latches. In the same part we also constructed the truth tables of SR Latches and Flip Flop we were going to implement in the parts after that. We reused the NAND module we designed in the second experiment. We implemented SR Latch with and without EN, and D Flip Flop alongside D Latch. All are done using only NAND modules, with operations like negates are done with the õperation. The experiment was rather simple, with the only issue being the time we have this week is too limited, therefore Part 4 is neglected.

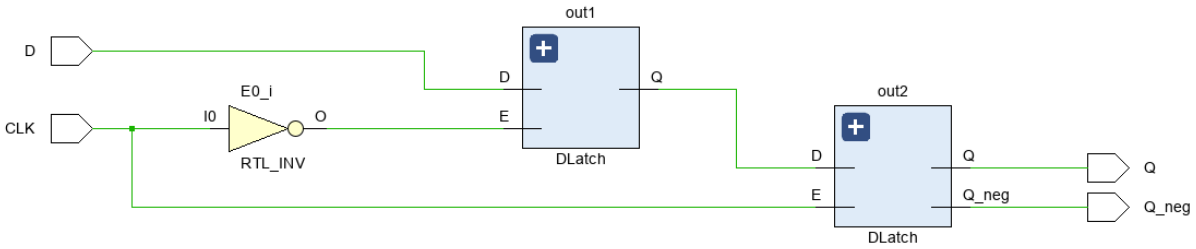

Figure 10: Elaborated Design Schematic of D Flip Flop with D Latches

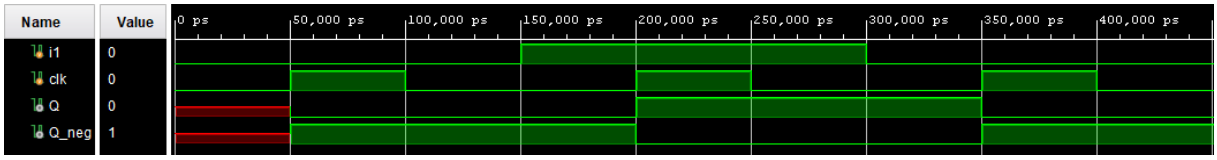

Figure 11: Simulation of D Flip Flop

5 CONCLUSION

We successfully implemented the modules and simulated them, but apart from last weeks we learnt how to implement adders which are more than one bits using only allowed operators. We learnt how to implement more than one bit adders specifically 4-bits and 16-bits. In 16-bit adder, we learnt how to call bits of a number. Also in order to implement adder-substractor in part 6, we learnt for loop's detail. Lastly, it was a great chance for us to implement flags as overflow, borrow and validity signs.

REFERENCES

[1] LogicLab. Logic lab. *An example journal*, 22(4):10–16, February 2020.

[2] Overleaf documentation https://tr.overleaf.com/learn.