

Lab 6 Latches and Flip-Flops

CS211 Summer 2017

July 17, 2017

Xu Mingzhi

## Contents

1. Objective .....	3
2. SR Latch.....	4
2.1 Functionality and Specification .....	4
2.2 Simulation .....	6
3. Control SR Latch .....	8
3.1 Functionality and Specification .....	8
3.2 Simulation .....	10
4. D Latch.....	11
4.1 Functionality and Specification .....	11
4.2 Simulation .....	12
5. Positive Edge Triggered Master Slave D Flip Flop .....	14
5.1 Functionality and Specification .....	14
5.2 Simulation .....	15
6. Negative Edge Triggered Master Slave D Flip Flop.....	17
6.1 Functionality and Specification .....	17
6.2 Simulation .....	18
7. JK Flip Flop .....	20
7.1 Functionality and Specification .....	20
7.2 Simulation .....	22
8. T Flip Flop .....	23
8.1 Functionality and Specification .....	23
8.2 Simulation .....	25
9. Analysis.....	26
10. Conclusion .....	28

# 1. Objective

In this lab, we are introduced to latches and flip flops and to under the difference between the two and their functionalities. Latches and flip flops are the basic elements for storing information and one latch or flip flop can store one bit of information and they are sequential device that will remember the previous value inputs and outputs to compute the next output. In order to understand how latches and flip flop works, we will create several latches and flip flops using block diagrams and VHDL code then run waveform simulations and observe the behavior of these devices.

The circuit we will be designing in this lab are:

- a. SR Latch
- b. Control SR Latch
- c. D Latch
- d. Positive Edge Triggered Master Slave D Flip Flop
- e. Negative Edge Triggered Master Slave D Flip Flop
- f. JK Flip Flop
- g. T Flip Flop

## 2. SR Latch

### 2.1 *Functionality and Specification*

The SR Latch is a basic storage device used to store a binary bit. This latch has two functions which are Set and Reset which are defined by S and R. Below is the characteristic table that describes the behavior of the SR Latch.

S	R	Q	Q'	State
0	0	-	-	No Change
0	1	0	1	Reset
1	0	1	0	Set
1	1	?	?	Undefined

Based on the characteristic table, when input S and R are 0 there are no change for the output Q and Q' which means whatever the previous outputs were then the current outputs will be the same. When S is 0 and R is 1 then it will conduct the function Reset which resets Q back to 0 and since Q' stands for not Q it will always be the opposite of Q so Q' will be 1. When S is 1 and R is 0 then it will conduct the function Set which sets Q to 1 and Q' will be 0. When both S and R is 1, it is considered as the forbidden input that will put the output in a metastable state where it's undefined.

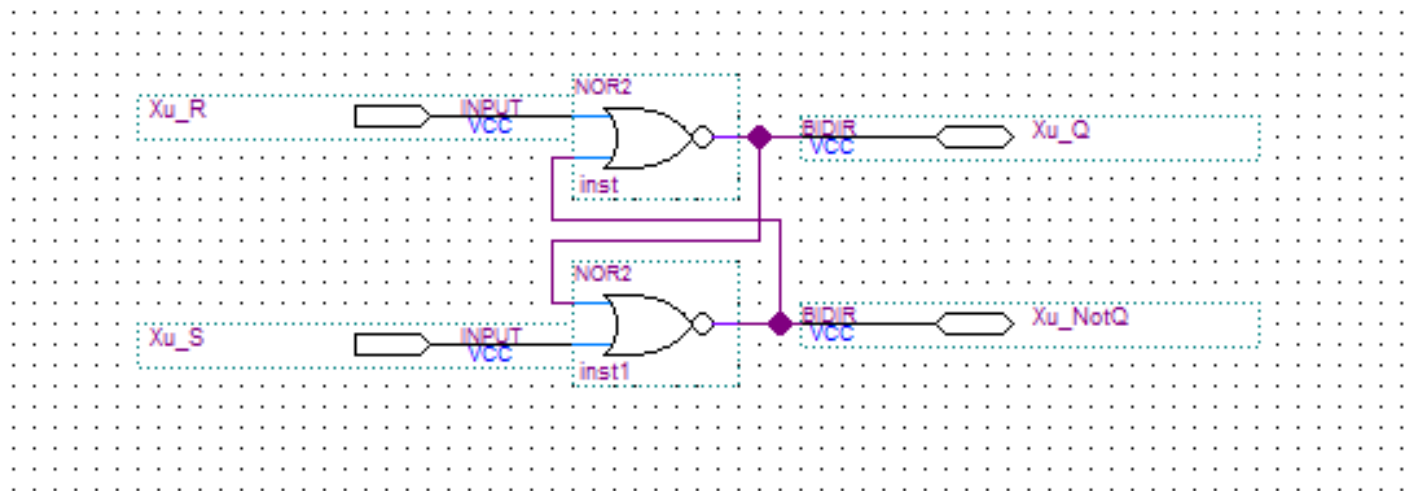


Figure 1: Block diagram for SR Latch.

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use work.Xu_package.all;
4
5  entity SR_Latch is
6  port( R, S: in std_logic;
7        Q, NotQ: inout std_logic );
8  end SR_Latch;
9
10 architecture arch of SR_Latch is
11 begin
12     Q <= R nor NotQ;
13     NotQ <= S nor Q;
14 end arch;

```

Figure 2: VHDL code for SR Latch.

This is the block diagram and VHDL code for SR Latch, the functionality of these two will be the same. It takes in R and S as inputs and bidirectional outputs which can also function as inputs Q and NotQ. The Boolean function for the outputs will be

$$Q = R \text{ nor } \text{Not}Q$$

$$\text{Not}Q = S \text{ nor } Q$$

Based on the last lab on VHDL language, we learn that unlike other coding languages which executes line by line, VHDL codes are about to execute parallelly which means all the lines are executed at the same time and alternate changes of the values.

## 2.2 *Simulation*

Now we will run the waveform simulation for both block diagram and VHDL code for SR Latch and compare the results that it'll produce.

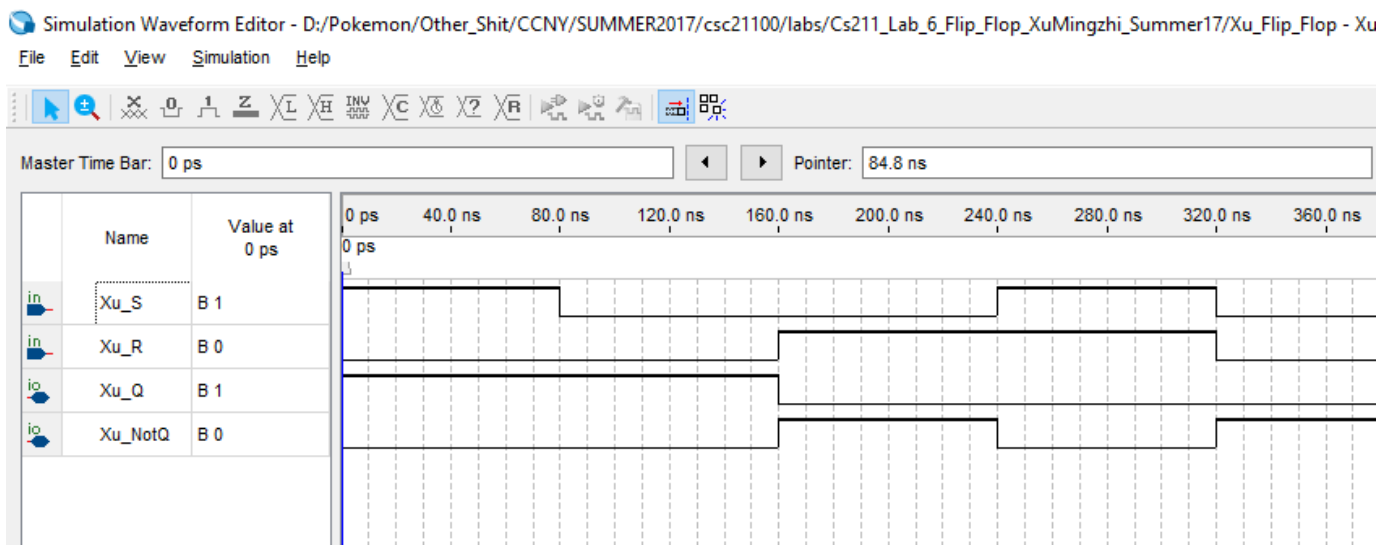


Figure 3: Vector waveform simulation for block diagram of SR Latch.

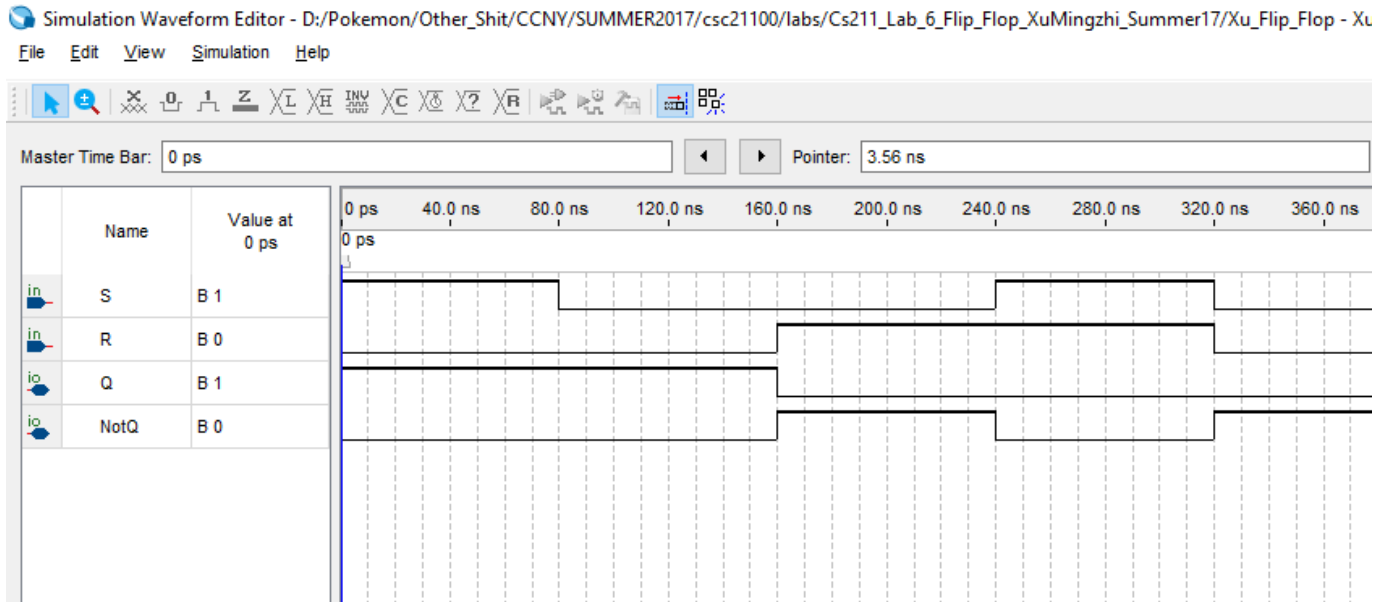


Figure 4: Vector waveform simulation for VHDL code of SR Latch.

We can observe that both simulations produce the same result and it corresponds to characteristic table of SR Latch.

### 3. Control SR Latch

#### 3.1 *Functionality and Specification*

The Control SR Latch is similar to the SR Latch that was designed in the last section, but this time it will have an extra input of C which stands for Control which means the user will have control when there will be a state change. Below is the characteristic table of the Control SR Latch.

C	S	R	$Q_n$	$Q_n'$	State
0	X	X	Q	Q'	No Change
1	0	0	Q	Q'	No Change
1	0	1	0	1	Reset
1	1	0	1	0	Set
1	1	1	?	?	Undefined

Based on the characteristic table, when C is 0, no change will be made for whatever S or R are.

Which means state changes will occur when C is 1. When C is 1, it will function the same as SR Latch.



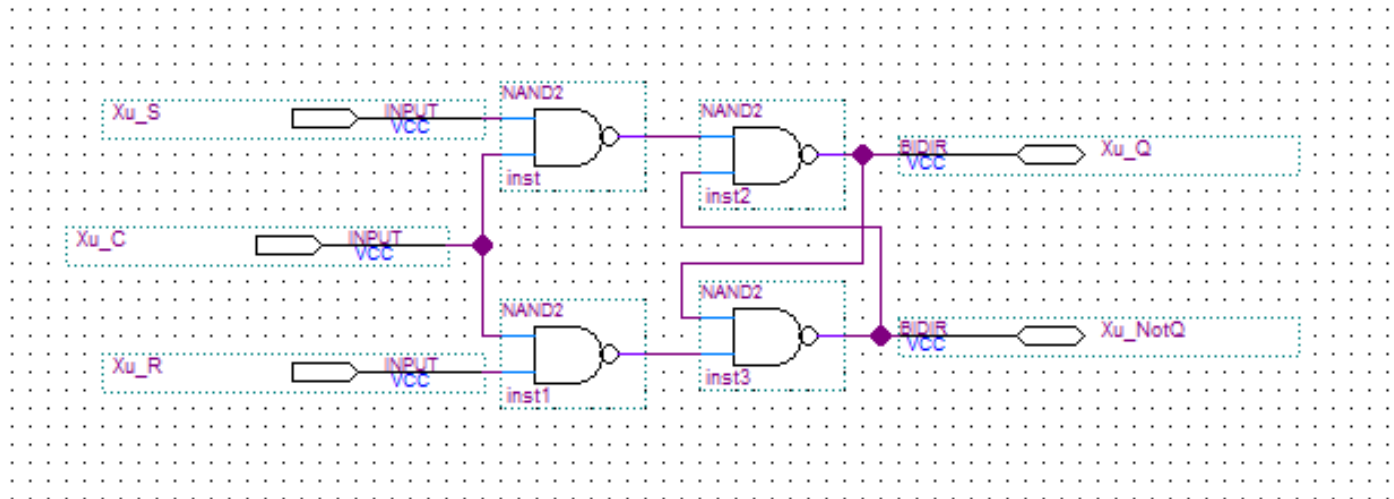


Figure 5: Block diagram of Control SR Latch.

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use work.Xu_package.all;
4
5  entity Control_SR_Latch is
6  port( C, S, R: in std_logic;
7        Q, NotQ: inout std_logic );
8  end Control_SR_Latch;
9
10 architecture arch of Control_SR_Latch is
11 begin
12     Q <= ( S nand C ) nand NotQ;
13     NotQ <= ( R nand C ) nand Q;
14 end arch;

```

Figure 6: VHDL code of Control SR Latch.

This is the block diagram and VHDL code for Control SR Latch, this time we will use four NAND gates to construct the block diagram and the Boolean function are

$$Q = ( S \text{ nand } C ) \text{ nand } \text{Not}Q$$

$$\text{NotQ} = (\text{R nand C}) \text{ nand Q}$$

### 3.2 Simulation

Now we will run the waveform simulation for both block diagram and VHDL code for Control SR Latch and compare the results that it'll produce.

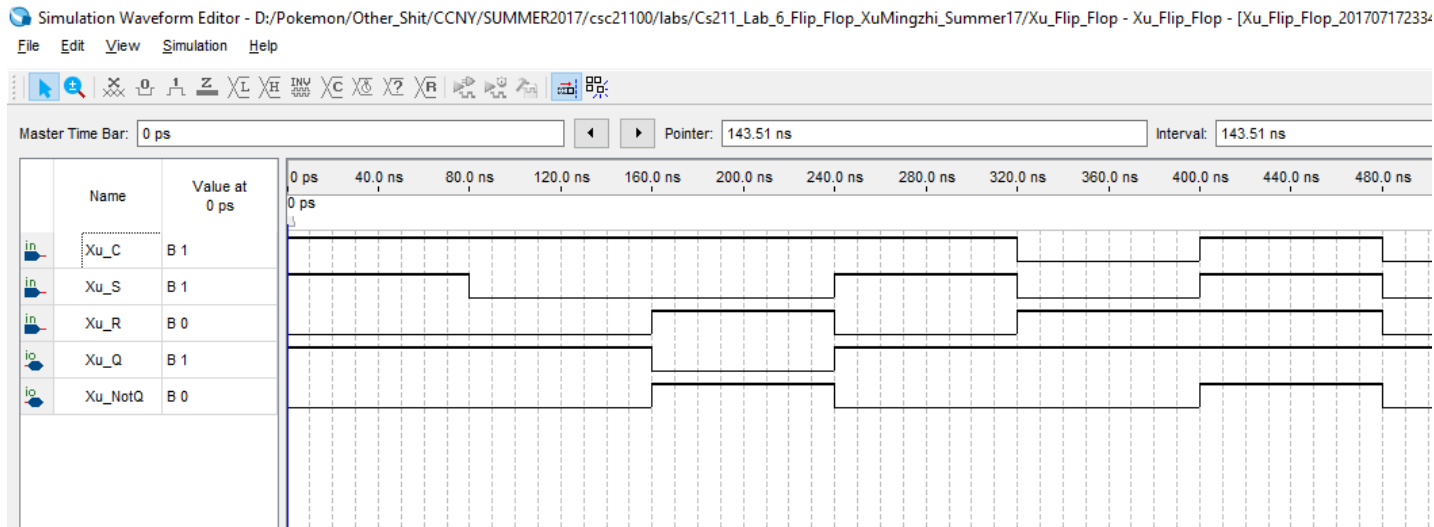


Figure 7: Simulation for block diagram of Control SR Latch.

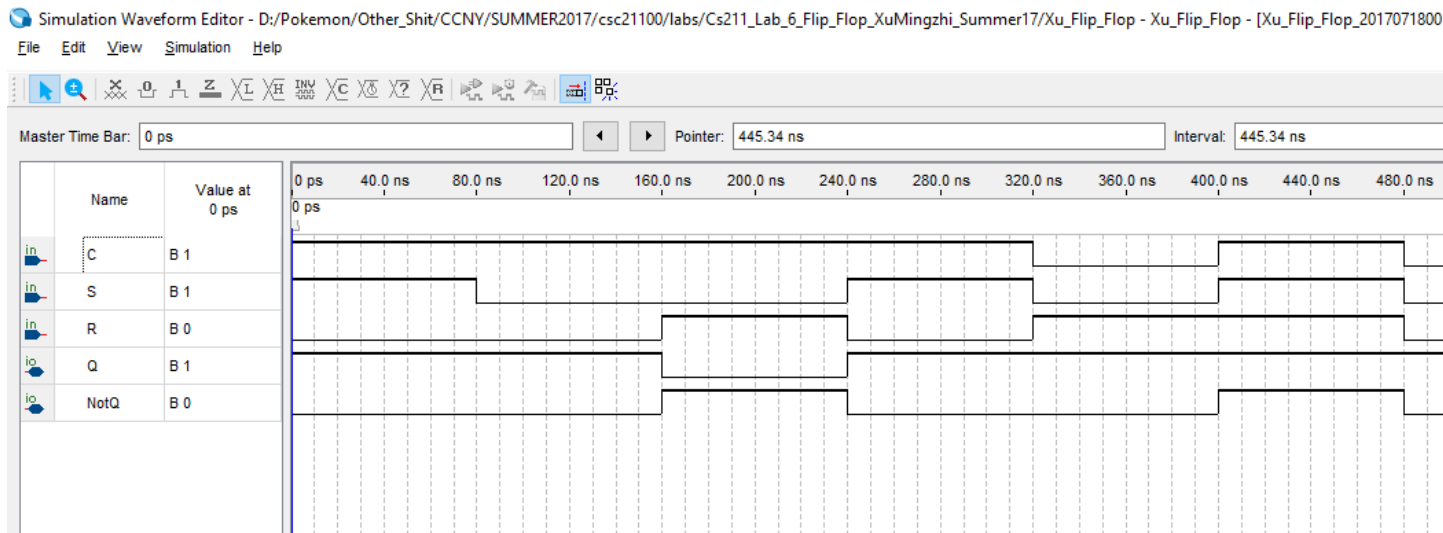


Figure 8: Simulation for VHDL code of Control SR Latch.

We can observe that both simulations produce the same result and it corresponds to characteristic table of Control SR Latch.

## 4. D Latch

### 4.1 *Functionality and Specification*

The D Latch is a modified version of the Control SR Latch which only uses input D to replace Set and Reset. This will make sure that it will never go into metastable state since S and R input will always be opposite of each other due to the use of a NOT gate. Below is the characteristic table of the D Latch.

C	D	$Q_n$	$Q_n'$	State
0	X	Q	Q'	No Change
1	0	0	1	Reset
1	1	1	0	Set

Based on the characteristic table we can see that the functionality of the D Latch is almost the same the Control SR Latch. When C is 0, there will be no change and will output the previous outputs. When C is 1 and D is 0 it will reset and Q will be 0 and Q' will be 1. When C is 1 and D is 1 it will set Q to 1 and Q' to 0.

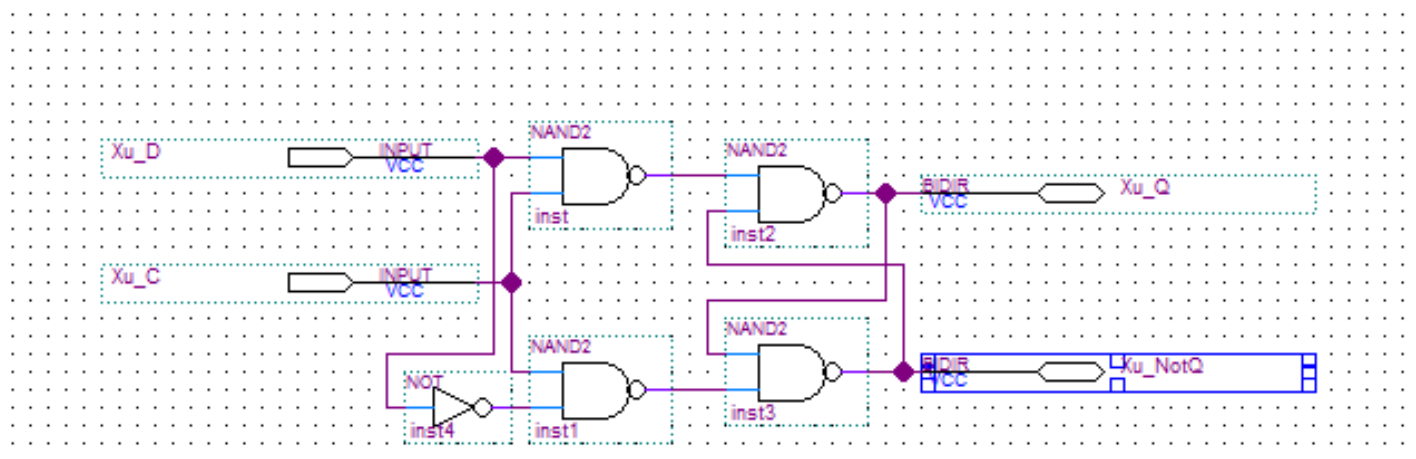
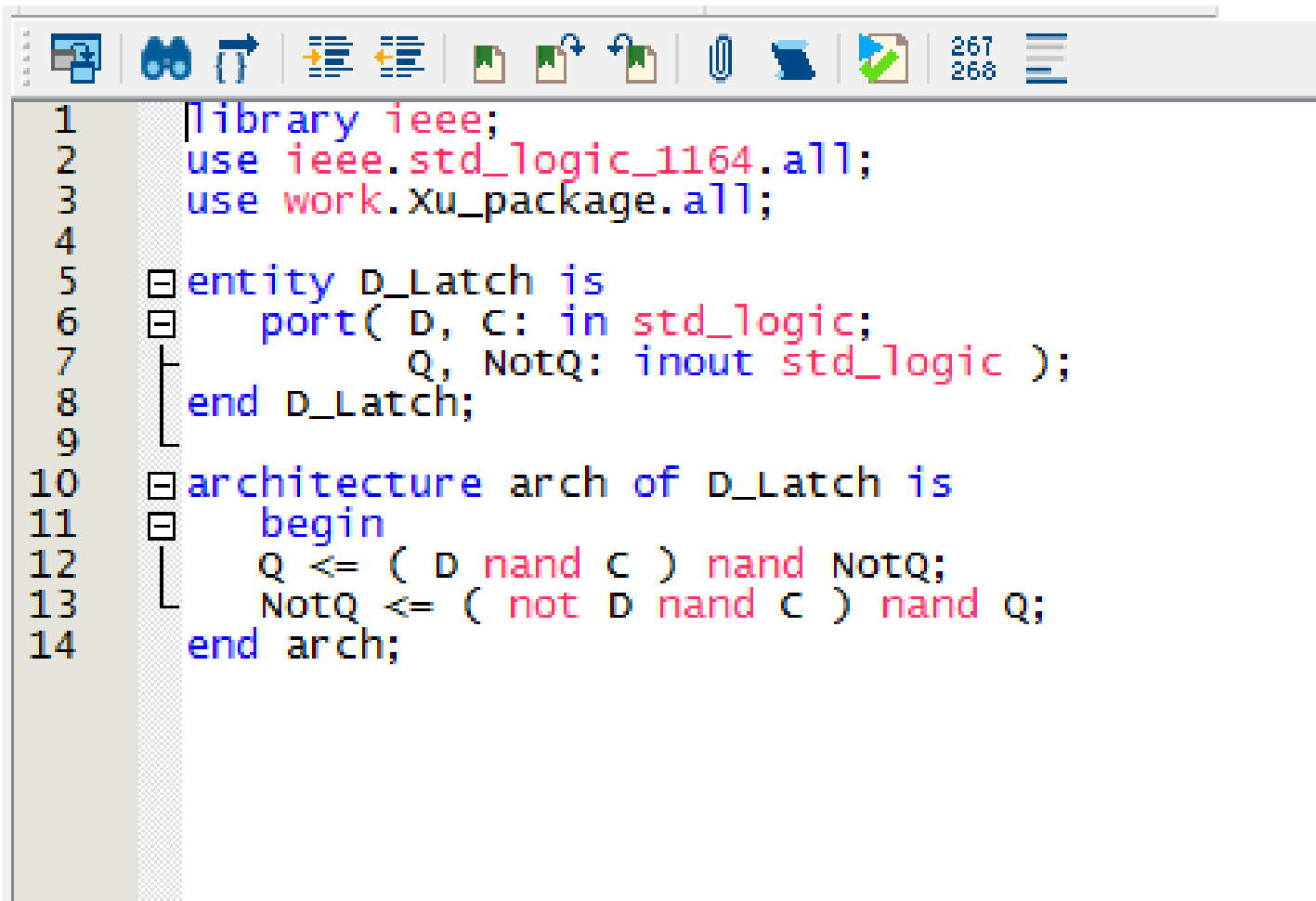


Figure 9: Block diagram of D Latch.



```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use work.Xu_package.all;
4
5  entity D_Latch is
6  port( D, C: in std_logic;
7        Q, NotQ: inout std_logic );
8  end D_Latch;
9
10 architecture arch of D_Latch is
11 begin
12     Q <= ( D nand C ) nand NotQ;
13     NotQ <= ( not D nand C ) nand Q;
14 end arch;

```

Figure 10: VHDL code of D Latch.

This is the block diagram and VHDL code for the D Latch. We will use a NOT gate and input D to replace the SR input in the Control SR Latch so there the metastable state will never occur.

The Boolean function will be

$$Q = ( D \text{ nand } C ) \text{ nand } \text{Not}Q$$

$$\text{Not}Q = ( \text{Not } D \text{ nand } C ) \text{ nand } Q$$

## 4.2 *Simulation*

Now we will run the waveform simulation for both block diagram and VHDL code for D Latch and compare the results that it'll produce.

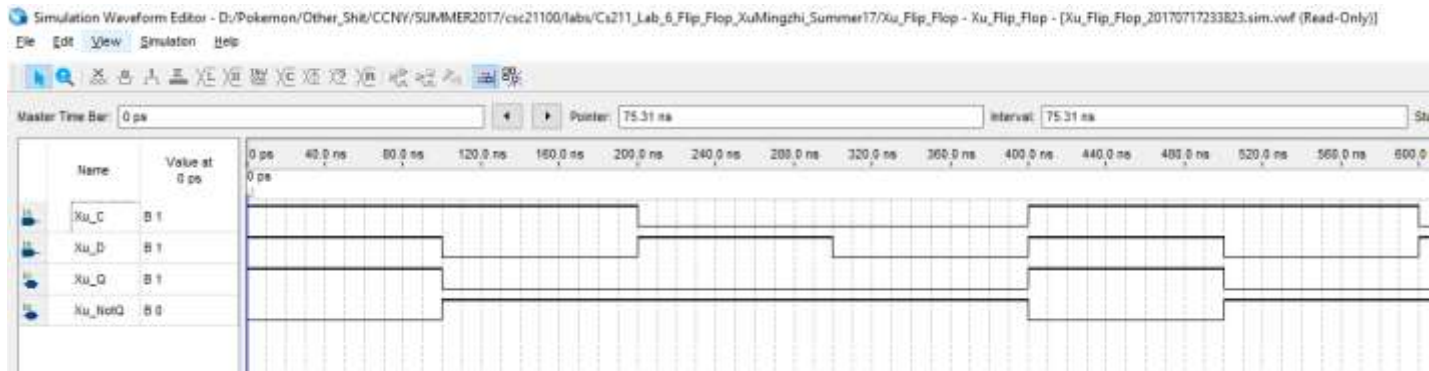


Figure 11: Simulation for block diagram of D Latch.



Figure 12: Simulation for VHDL code of D Latch.

We can observe that both simulations produce the same result and it corresponds to characteristic table of D Latch.

## 5. Positive Edge Triggered Master Slave D Flip Flop

### 5.1 *Functionality and Specification*

This is a Positive Edge Triggered Master Slave D Flip Flop, the Master Slave D Flip Flop is made up of two D Latch component. It is different than the D Latch because it triggers the event to change state when the clock signal is going from 0 to 1 or 1 to 0. Since this is a positive edge triggered which means it will change state when the clock signal goes from 0 to 1. Below is the characteristic table for Positive Edge Triggered Master Slave D Flip Flop.

$Clk$	$D$	$Q$	$Q_{next}$	$Q_{next}'$
0	$\times$	0	0	1
0	$\times$	1	1	0
1	$\times$	0	0	1
1	$\times$	1	1	0
$\uparrow$	0	$\times$	0	1
$\uparrow$	1	$\times$	1	0

Based on the characteristic table we can see that whenever Clock signal is 0 or 1, the D input doesn't affect the output  $Q_{next}$  and will output the previous output  $Q$ . When the clock signal goes from 0 to 1, the  $Q_{next}$  will change based on D at that moment.

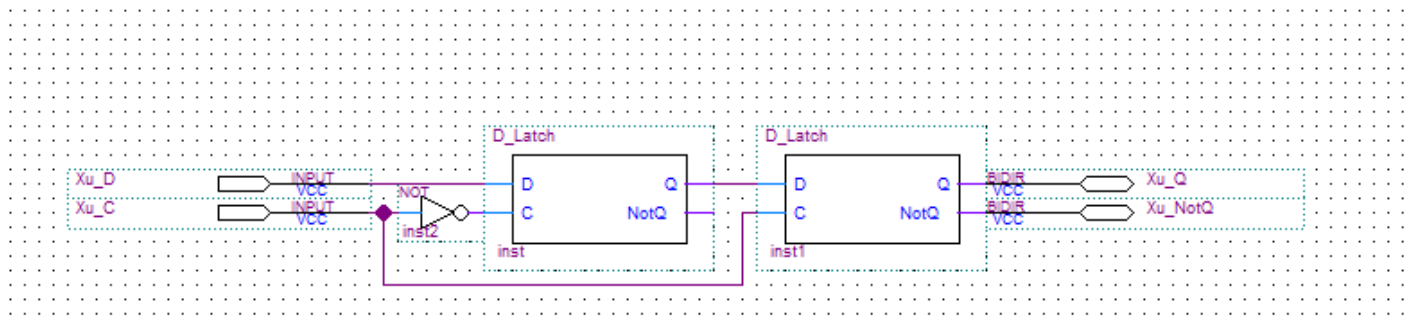
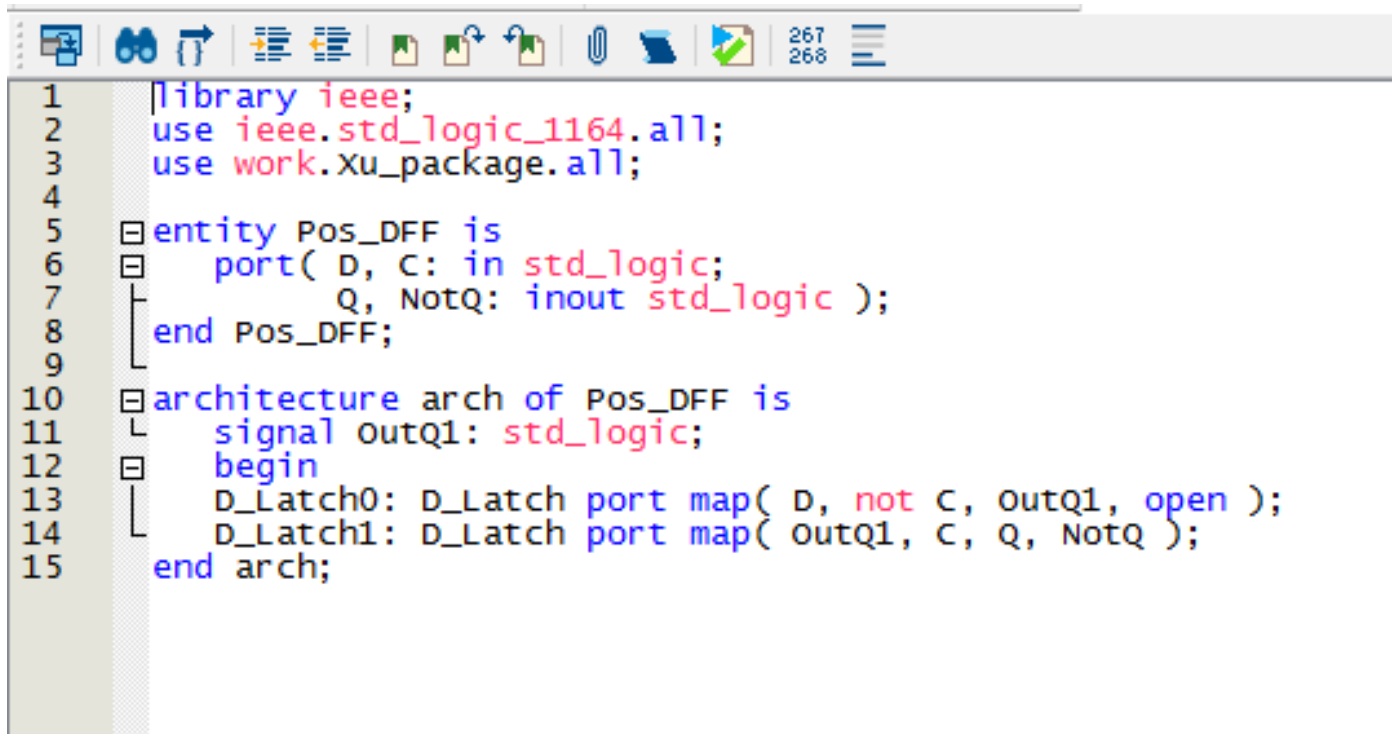


Figure 13: Block diagram for Positive D Flip Flop.

The image shows a screenshot of a VHDL code editor. The editor has a toolbar at the top with various icons for file operations, editing, and simulation. The code is written in a syntax-highlighted font. The code defines an entity named 'Pos\_DFF' with two input ports 'D' and 'C' of type 'std\_logic', and two output ports 'Q' and 'NotQ' of type 'std\_logic'. The architecture 'arch' of 'Pos\_DFF' uses two 'D\_Latch' components. The first latch, 'D\_Latch0', has 'D' as its data input, 'not C' as its clock input, and 'outQ1' as its output. The second latch, 'D\_Latch1', has 'outQ1' as its data input, 'C' as its clock input, and 'Q' as its output. The 'NotQ' output is simply the complement of 'Q'.

```
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use work.Xu_package.all;
4
5 entity Pos_DFF is
6     port( D, C: in std_logic;
7           Q, NotQ: inout std_logic );
8 end Pos_DFF;
9
10 architecture arch of Pos_DFF is
11     signal outQ1: std_logic;
12     begin
13         D_Latch0: D_Latch port map( D, not C, outQ1, open );
14         D_Latch1: D_Latch port map( outQ1, C, Q, NotQ );
15     end arch;
```

Figure 13: VHDL code for Positive D Flip Flop.

This is the block diagram and VHDL code for Positive Edge Triggered Master Slave D Flip Flop. It uses two D Latch as components and for it to have positive edge trigger, the first clock signal will have a NOT gate.

## 5.2 Simulation

Now we will run the waveform simulation for both block diagram and VHDL code for Positive Edge Triggered Master Slave D Flip Flop and compare the results that it'll produce.

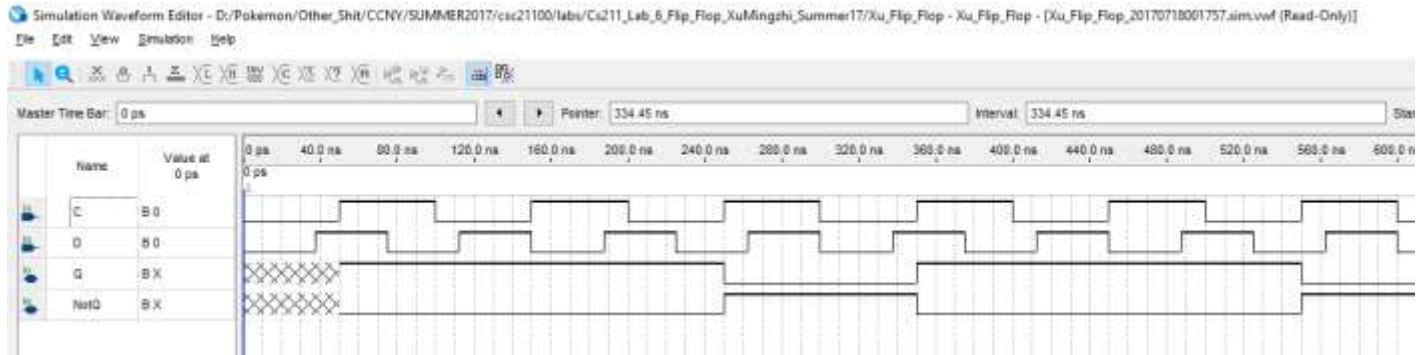


Figure 14: Simulation for block of Positive D Flip Flop.

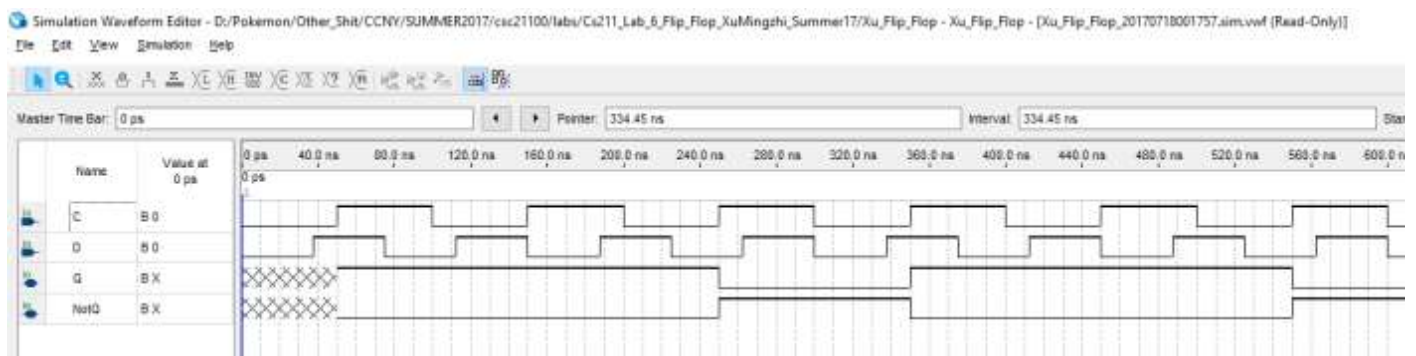


Figure 15: Simulation for VHDL of Positive D Flip Flop.

We can observe that both simulations produce the same result and it corresponds to characteristic table of Positive D Flip Flop.



## 6. Negative Edge Triggered Master Slave D Flip Flop

### 6.1 *Functionality and Specification*

The Negative Edge Triggered Master Slave D Flip Flop functions similarly to the Positive Edge Triggered Master Slave D Flip Flop. The only difference is that it will change state and the clock signal goes from 1 to 0.

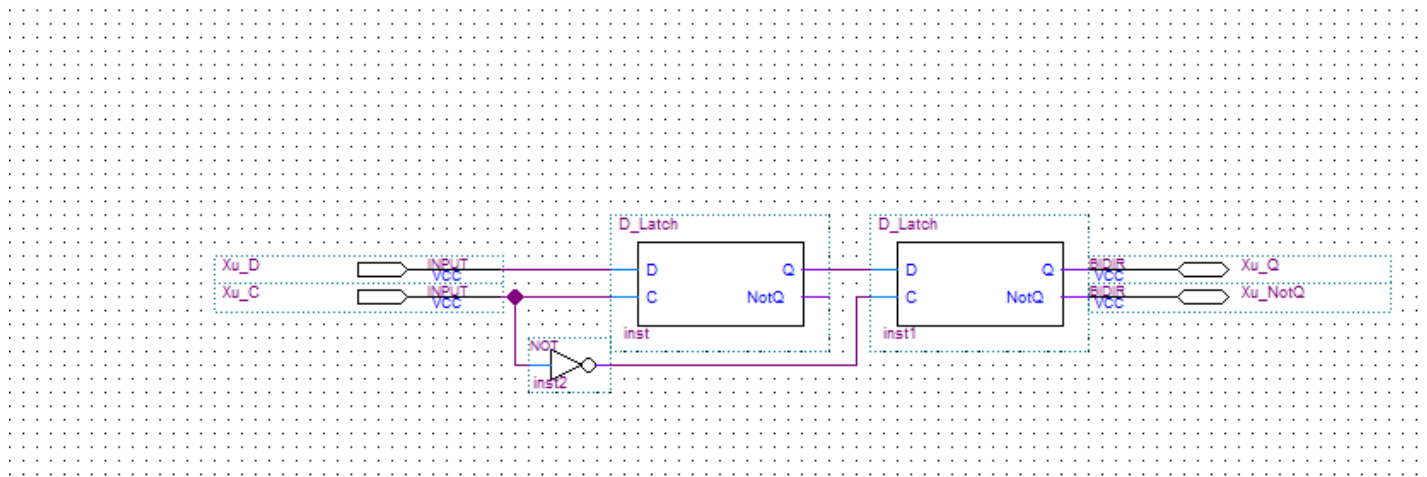


Figure 16: Block diagram for Negative D Flip Flop.

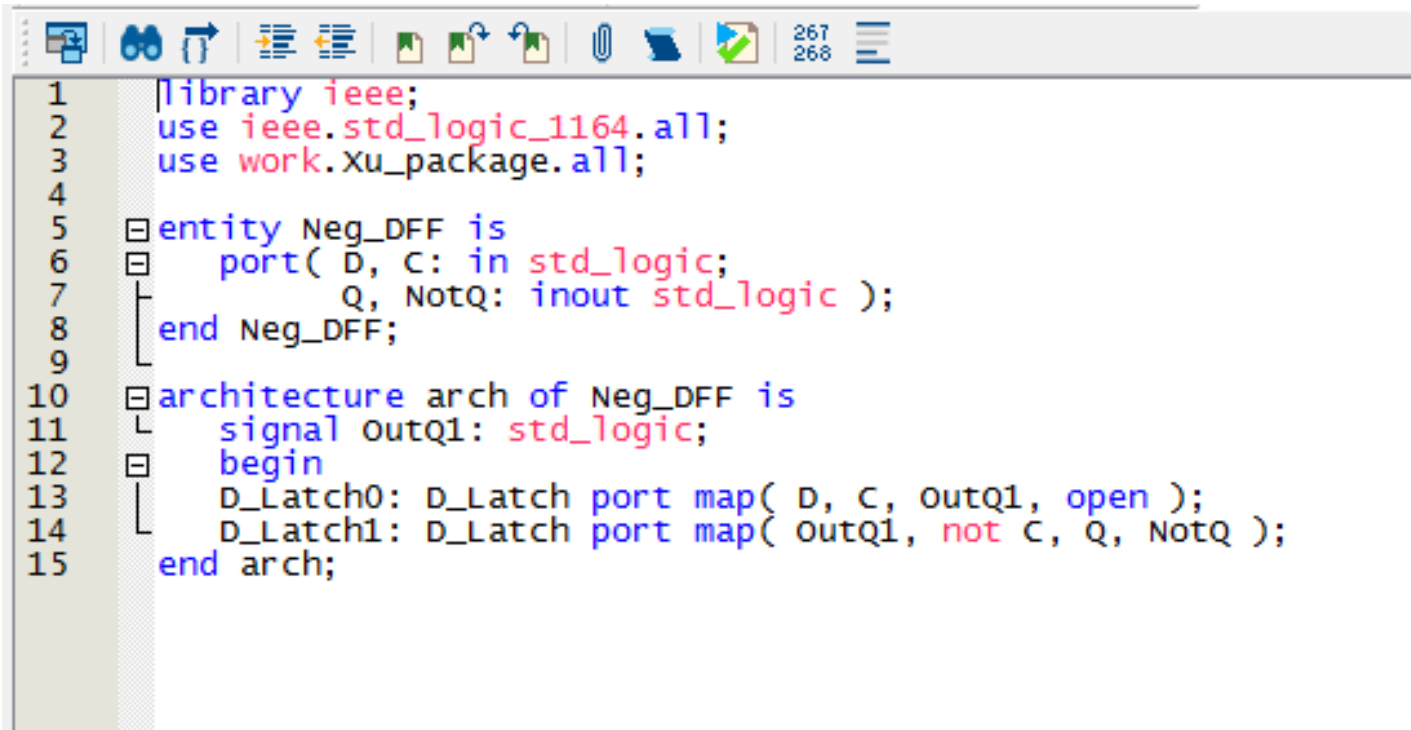


Figure 17: VHDL code for Negative D Flip Flop.

This is the block diagram and VHDL code for the Negative Edge Triggered Master Slave D Flip Flop. We can see that it is very similar to the Positive D Flip Flop, but in order for the clock signal to triggered the event to change state from 1 to 0 we will have the NOT gate connect to the clock input in the slave D Latch unlike the Positive D Flip Flop where the NOT gate goes to the Master D Latch clock input.

## 6.2 Simulation

Now we will run the waveform simulation for both block diagram and VHDL code for Negative Edge Triggered Master Slave D Flip Flop and compare the results that it'll produce

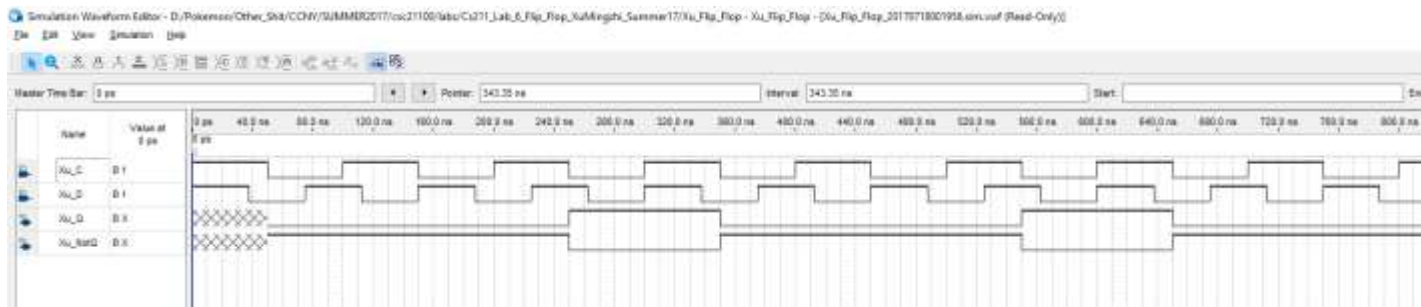


Figure 18: Simulation for block of Negative D Flip Flop.

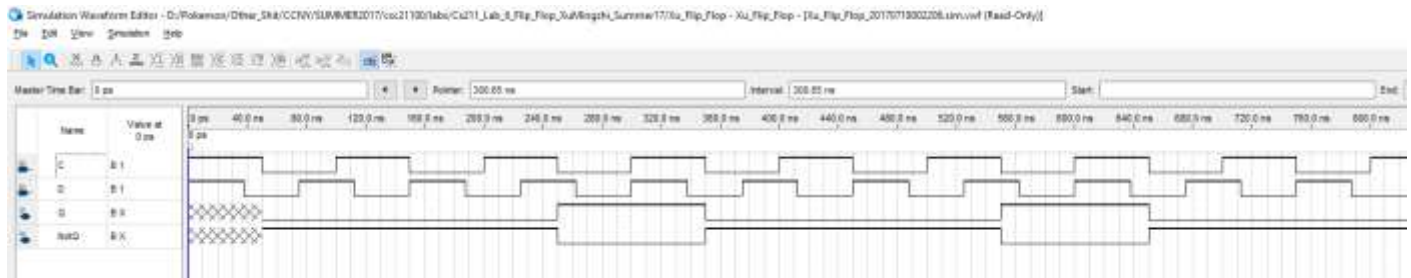


Figure 19: Simulation for VHDL of Negative D Flip Flop.

We can observe that both simulations produce the same result and it corresponds to characteristic table of Negative D Flip Flop.

## 7. JK Flip Flop

### 7.1 *Functionality and Specification*

The JK Flip Flop is very similar to the Control SR Latch but this is a flip flop which means it is a edge triggered device and the JK flip flop uses the Positive D Flip Flop as component. Below is the characteristic table for the JK Flip Flop.

$J$	$K$	$Q$	$Q_{next}$	$Q_{next}'$
0	0	0	0	1
0	0	1	1	0
0	1	0	0	1
0	1	1	0	1
1	0	0	1	0
1	0	1	1	0
1	1	0	1	0
1	1	1	0	1

Based on the characteristic table we can observe that when both J and K is 0 there will be no change and  $Q_{next}$  will output the previous output  $Q$ . When J is 0 and K is 1  $Q_{next}$  will Reset to 0. When J is 1 and K is 0  $Q_{next}$  will Set to 1. When both J and K are 1, instead of going into metastable state like the Control SR Latch, it will instead output the NotQ of the previous  $Q$ .

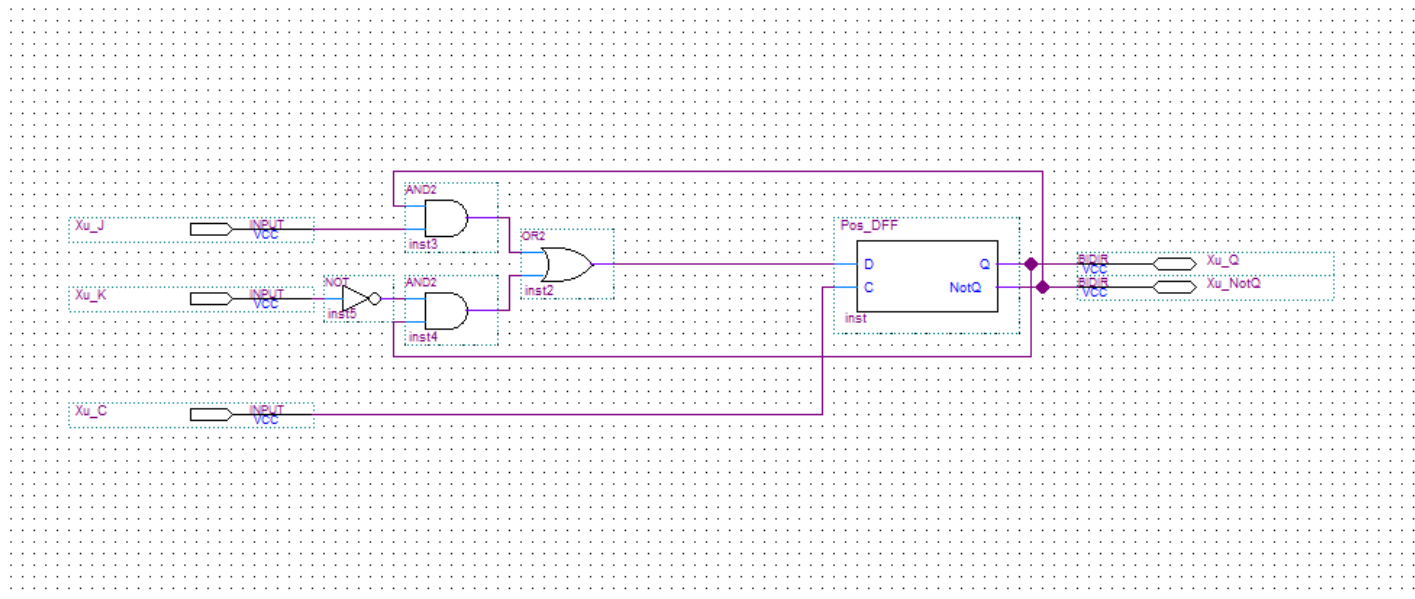


Figure 20: Block diagram of JK Flip Flop.

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use work.Xu_package.all;
4
5  entity JK_FF is
6  port( J, K, C: in std_logic;
7        Q, QNot: inout std_logic );
8  end JK_FF;
9
10 architecture arch of JK_FF is
11     signal X: std_logic;
12     begin
13         X <= ( J and QNot ) or ( not K and Q );
14         DFF0: Pos_DFF port map( X, C, Q, QNot );
15     end arch;

```

Figure 21: VHDL code of JK Flip Flop.

## 7.2 Simulation

Now we will run the waveform simulation for both block diagram and VHDL code for JK Flip Flop and compare the results that it'll produce.

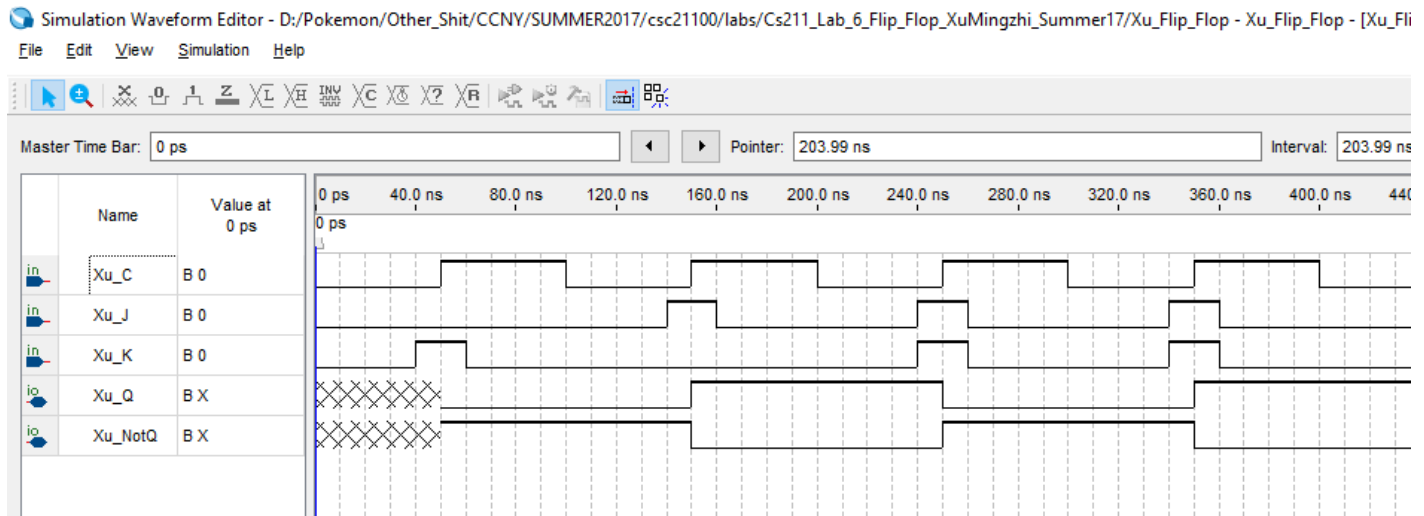


Figure 22: Simulation for block of JK Flip Flop.

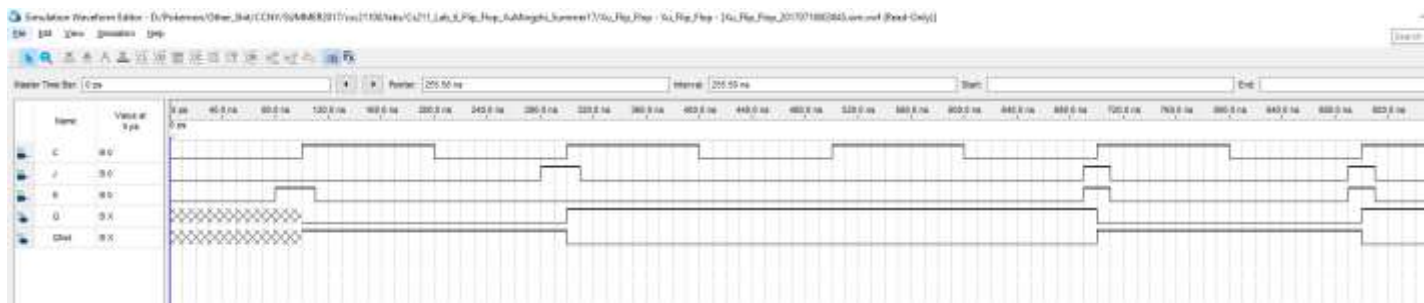


Figure 23: Simulation for VHDL of JK Flip Flop.

We can observe that both simulations produce the same result and it corresponds to characteristic table of JK Flip Flop.

## 8. T Flip Flop

### 8.1 *Functionality and Specification*

The T Flip Flop has the same function as the JK Flip Flop. Below is the characteristic table for the T Flip Flop.

$T$	$Q$	$Q_{next}$	$Q_{next}'$
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Based on the characteristic table, we can observe that when T is 0  $Q_{next}$  will output the previous Q. When T is 1  $Q_{next}$  will output the QNot of the previous Q.

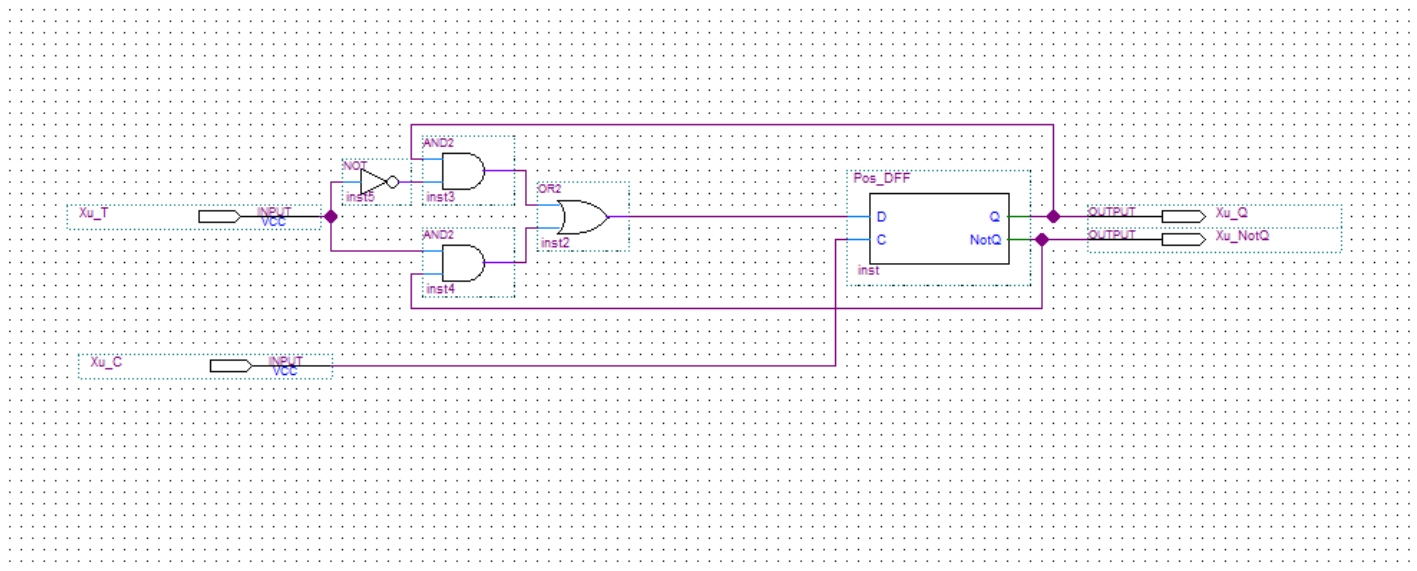
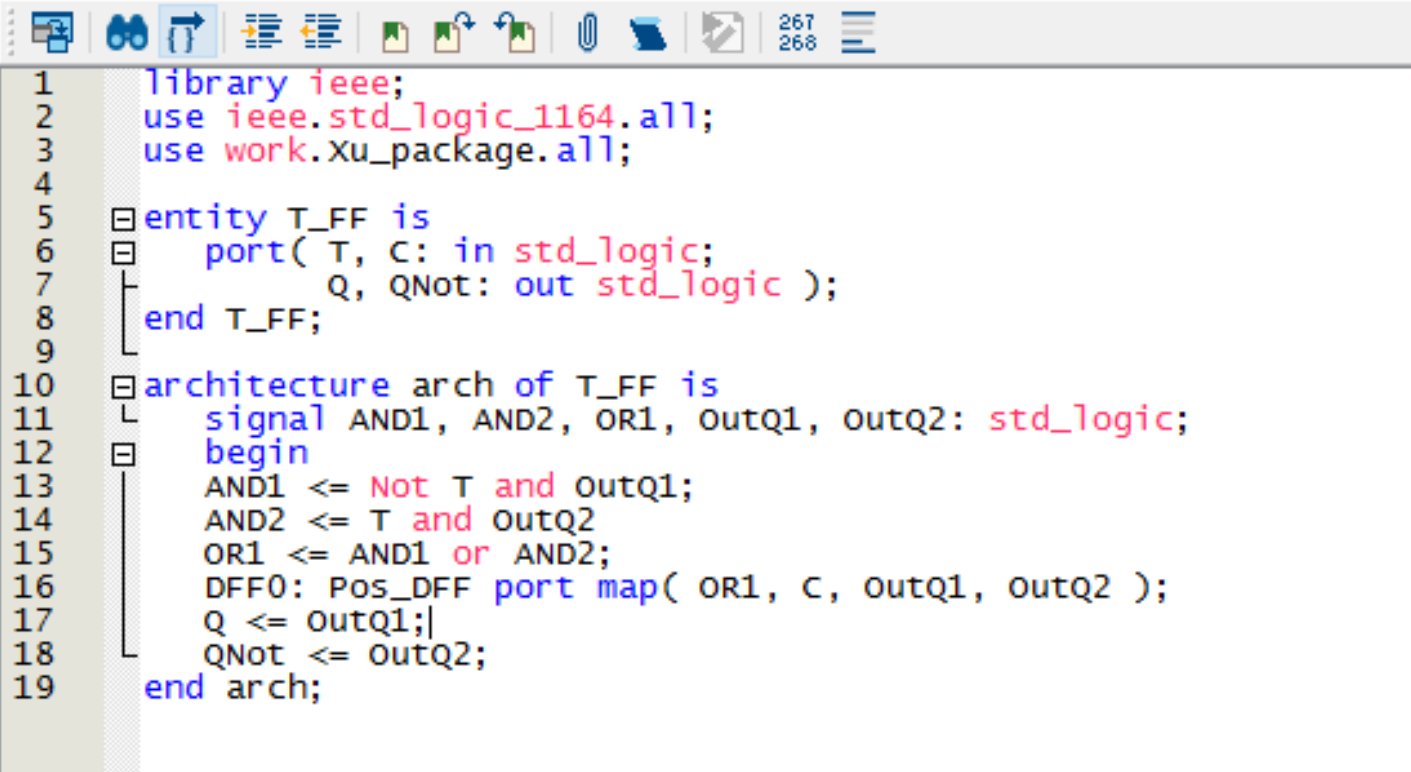


Figure 24: Block diagram of T Flip Flop.



```
1  library ieee;
2  use ieee.std_logic_1164.all;
3  use work.Xu_package.all;
4
5  entity T_FF is
6  port( T, C: in std_logic;
7        Q, QNot: out std_logic );
8  end T_FF;
9
10 architecture arch of T_FF is
11     signal AND1, AND2, OR1, OutQ1, OutQ2: std_logic;
12     begin
13         AND1 <= Not T and OutQ1;
14         AND2 <= T and OutQ2;
15         OR1 <= AND1 or AND2;
16         DFF0: Pos_DFF port map( OR1, C, OutQ1, OutQ2 );
17         Q <= OutQ1;
18         QNot <= OutQ2;
19     end arch;
```

Figure 25: VHDL code of T Flip Flop.

This is the block diagram and VHDL code for the T Flip Flop. The T Flip Flop functions the same as the JK Flip Flop when both J and K is 1 which will output the QNot of the previous Q.



## 8.2 Simulation

Now we will run the waveform simulation for both block diagram and VHDL code for T Flip Flop and compare the results that it'll produce.



Figure 26: Simulation for block of T Flip Flop.

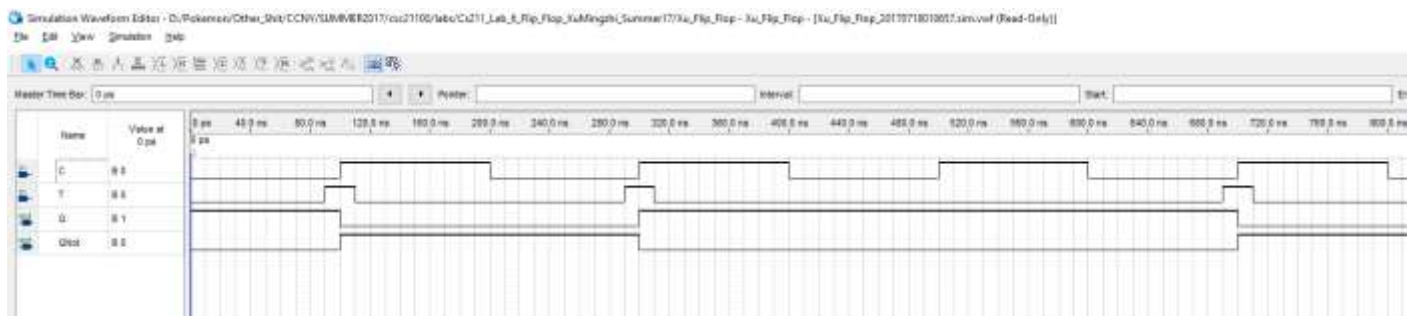


Figure 27: Simulation for VHDL of T Flip Flop.

We can observe that both simulations produce the same result and it corresponds to characteristic table of T Flip Flop.

## 9. Analysis

The difference between latches and flip-flops is that they have different event triggers for the clock signal. For latches state change will occur either when clock is 0 or 1 depending if it is a positive or negative D Latch. For flip-flops state change will occur when clock signal changes from 0 to 1 or from 1 to 0. It matters because Flip-Flops can be more flexible than Latches.

We want to avoid the scenario when S and R is 1 in an SR Latch because it will lead to metastable state which means there will be no indeterminate state, it could be avoided using a D Latch since the input D will replace S and R along with a NOT gate which means S and R will always be opposite of each other so metastable state will never occur.

The differences between edge-triggered and level triggered device is the same as flip-flops and latches. Latches are level triggered device which state change will occur when the clock is either 0 or 1. Flip-Flops are edge-triggered device which state change will occur when the clock goes from 0 to 1 or from 1 to 0. The T Flip Flop is an edge-triggered device in this lab.

Flip-Flops are always required to be clocked or else it will ignore the other inputs and there won't be any outputs.

Now we will compare the results of a D Latch with Positive D Flip Flop and Negative D Flip Flop.

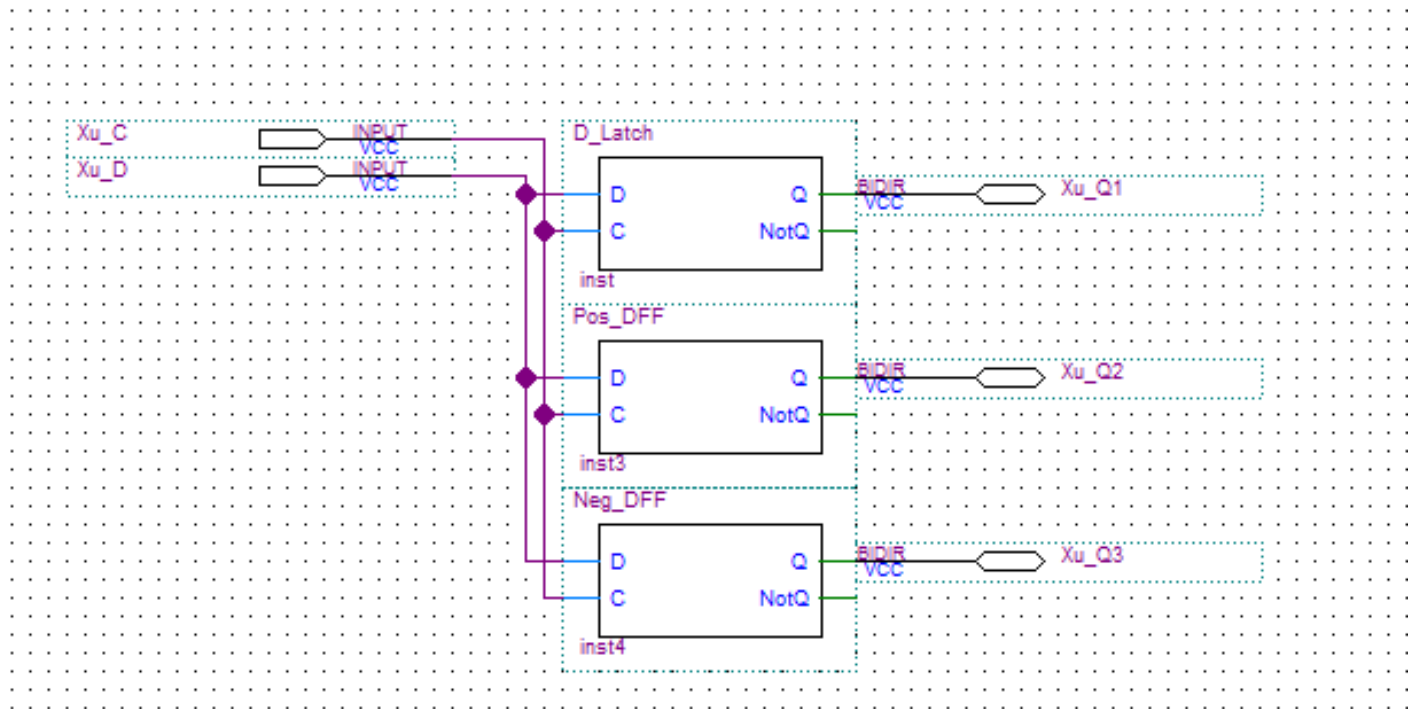


Figure 28: Block diagram to compare D Latch and D Flip Flops.

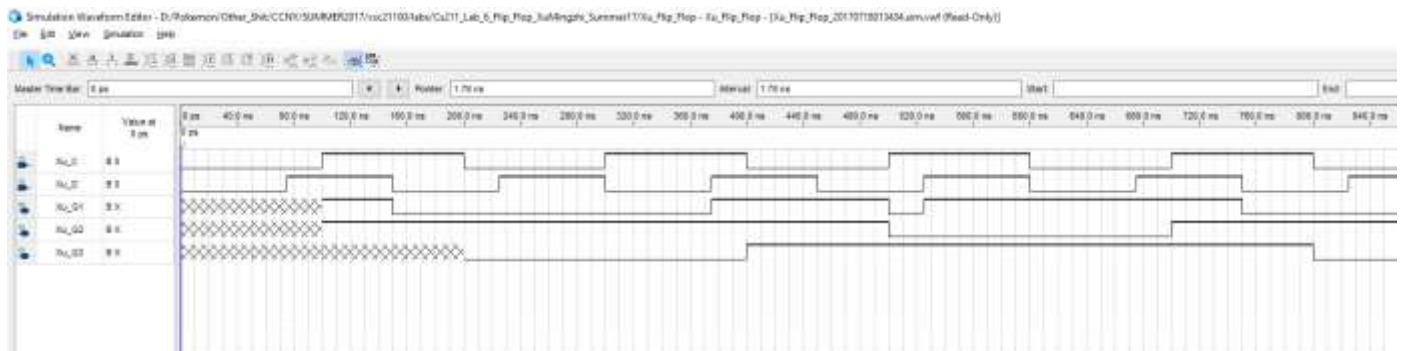


Figure 29: Simulation for compare D Latch and D Flip Flops.

We can observe from the simulation that for D Latch outputs 1 for Q1 when C is 1 and D is 1 and outputs 0 for Q1 when C is 1 and D is 0 and Q1 will output the previous Q1 when C is 0 and D can be either 0 or 1. The Positive and Negative Flip Flop functions the same as the D Latch but the state change occurs at the edge from 0 to 1 for Positive and from 1 to 0 for Negative.

## **10. Conclusion**

In this lab, we learned how latches and flip-flops work as sequential devices which send feedbacks from output back to input. We also learned the difference between latches and flip-flops which have different event triggers. For latches, it is a level-triggered device which will have a state change occur to output when the clock signal is either 0 or 1. For flip-flops, it is an edge-triggered device which will have a state change occur to output when the clock signal changes from 0 to 1 or from 1 to 0. These devices will also remember previous states which means it is storing 1-bit binary information.