Latches & Flip Flops

Name: Jonathan J. Rodriguez

CSC343 – Computer Organization Lab, Spring 2020

Instructor: Izidor Gertner

The City College of New York

Table of Contents

Objective

The objective of this laboratory exercise is to understand the difference between both latches and flip flops. Their differences would be critical whenever circuits are being built. They will be differentiated using block diagrams, VHDL, and waveform simulations to understand their behavior.

Software Used

1. Quartus II 13.0 sp1
2. Quartus Prime 16.0)

Design Specifications

SR Latch

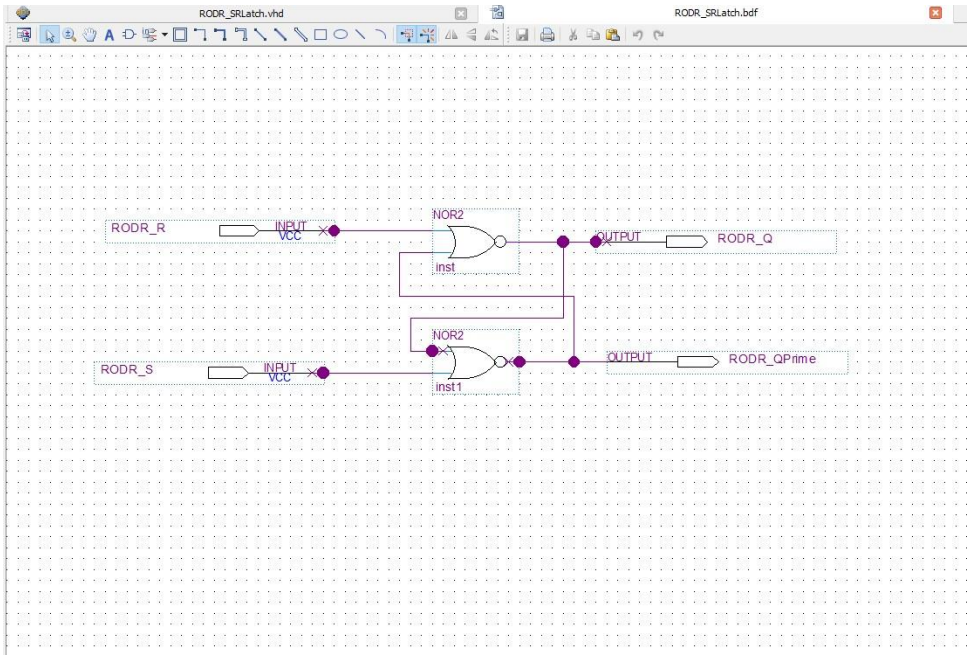

Figure 1: SR Latch Schematic

This electronic device is called a Set/Reset Latch (or SR Latch). It will convert the output of *RODR_Q* and its counterpart *RODR_QPrime* based on the inputs bits from *RODR_R* (for reset) and *RODR_S* (for setting input).

Control SR Latch

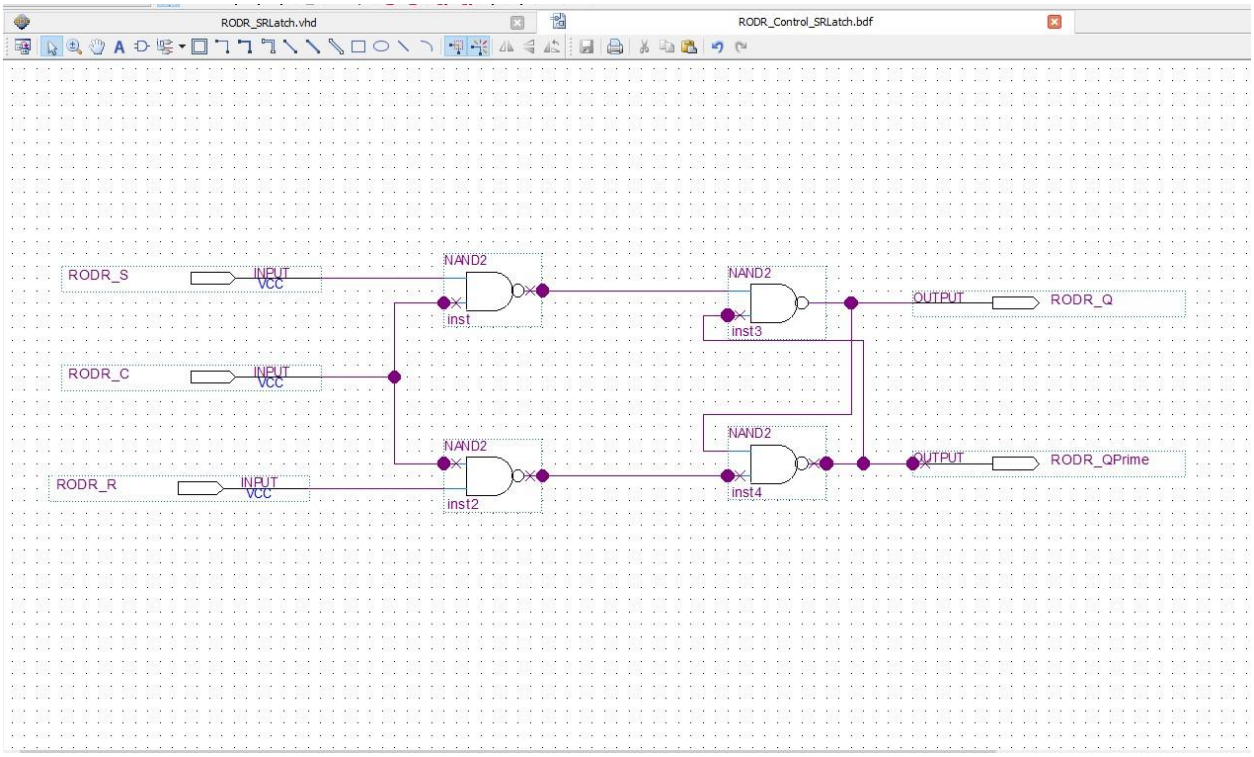

Figure 2: Control SR Latch Schematic created in Quartus 13.0

This electronic device is called a Control SR Latch. It is very similar to an SR latch, but now its outputs are determined by a clocked signal. The main difference is that there is a new input signal called *RODR_C*.

D Latch

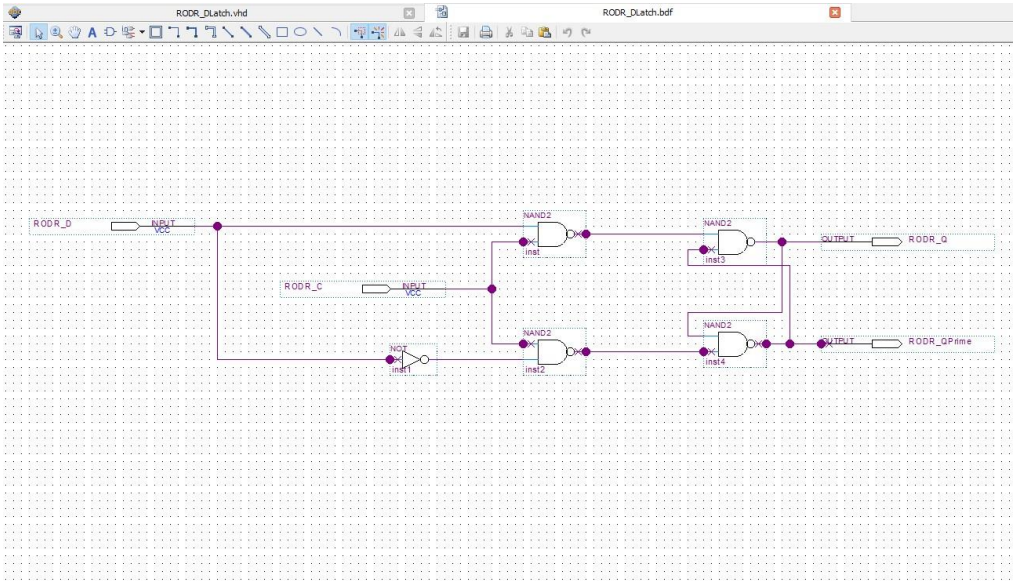

Figure 3: Schematic for a D Latch

This electronic device is called a D latch. It only takes in one input *RODR_D* and uses it to create outputs *RODR_Q* and *RODR_QPrime*. Using the NOT gate for the second latch would ensure that the SR latch undefined condition doesn't occur for the latch.

Master Slave Positive-Edge D Flip Flop

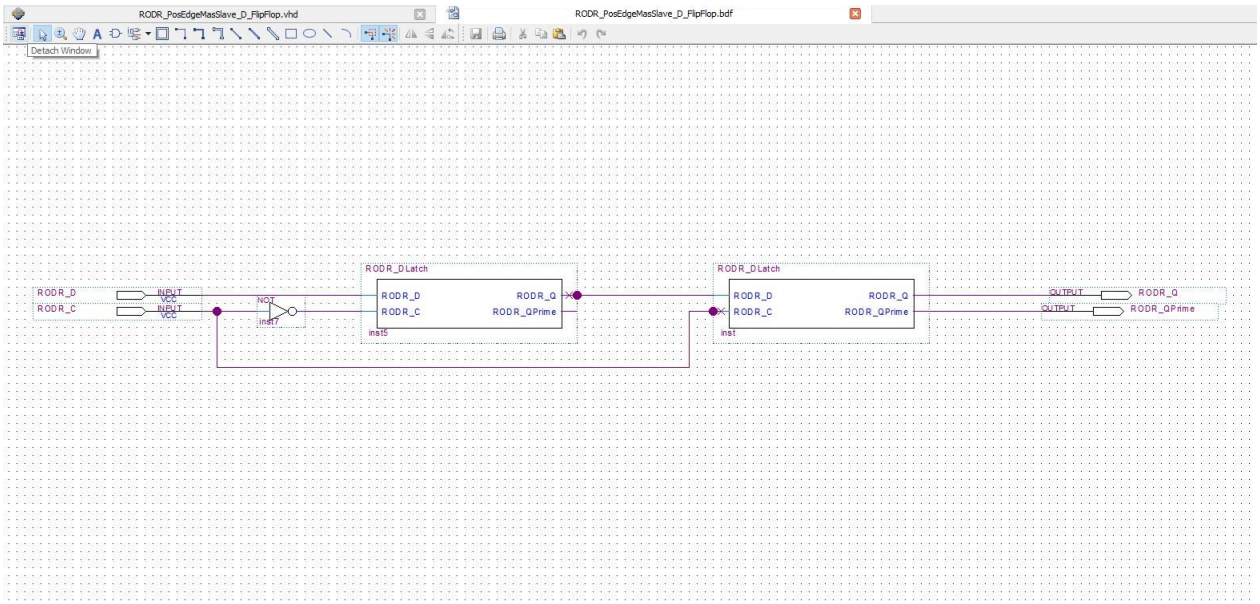

Figure 4: Schematic for Positive Edge Master Slave D Flip Flop

For a Master Slave D Flip Flop, we are using two D Latches with the output of the first from *RODR_Q* into the input of the next latch *RODR_D.* Also, they will take in the same clock signal, but one latch will have it inverted.

Master Slave Negative-Edge D Flip Flop

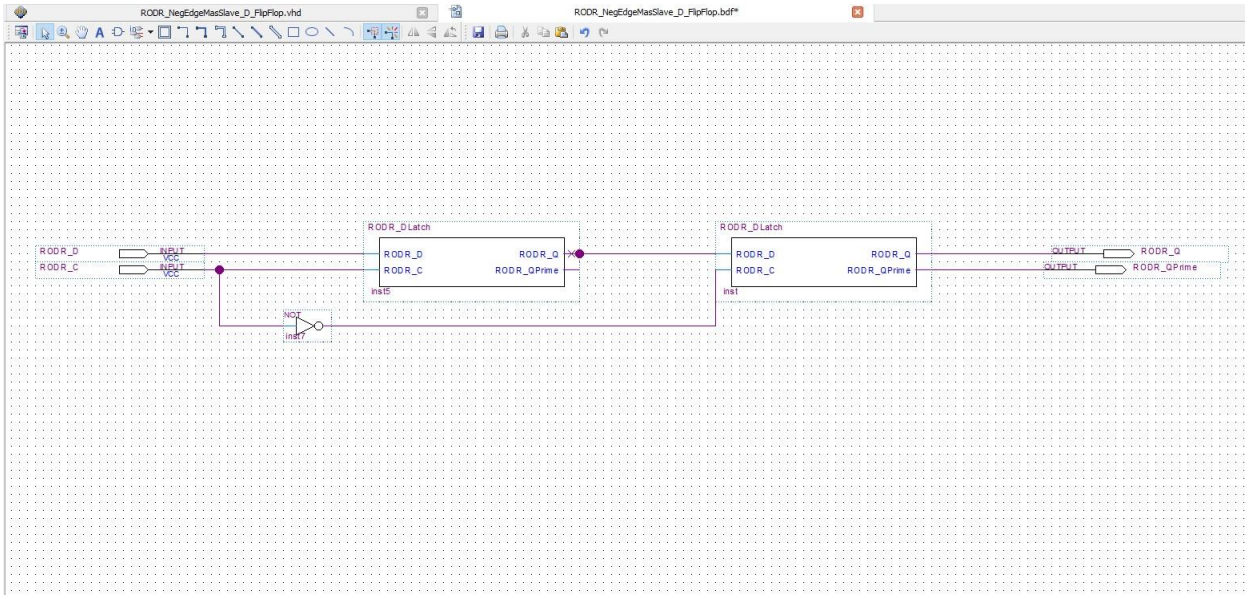

Figure 5: Schematic for Negative Edge Master Slave D Flip Flop

This electronic is the exact same as a positive edge master slave D flip flop, except that the second latch's clock signal is inverted. This means that changes to the output will only be made at the falling edge of the clock cycle.

Master Slave SR Flip Flop

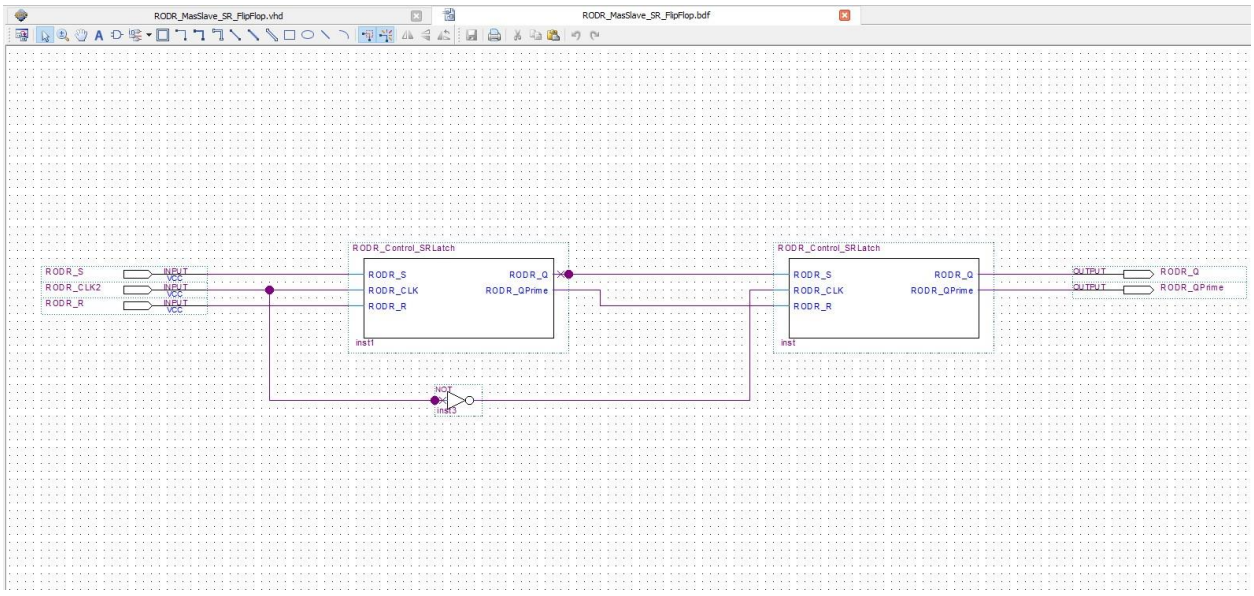

Figure 6: Schematic for Master Slave SR Flip Flop (Positive Edge)

In this electronic device, we are using two SR control latches. Like a Master Slave D Flip Flop, we will have shared connections between the outputs of the first control latch to the inputs of the next control latch. Also, the clock will be inverted for the second latch to ensure that on the positive edge of the clock signal, the output would change.

JK Flip Flop

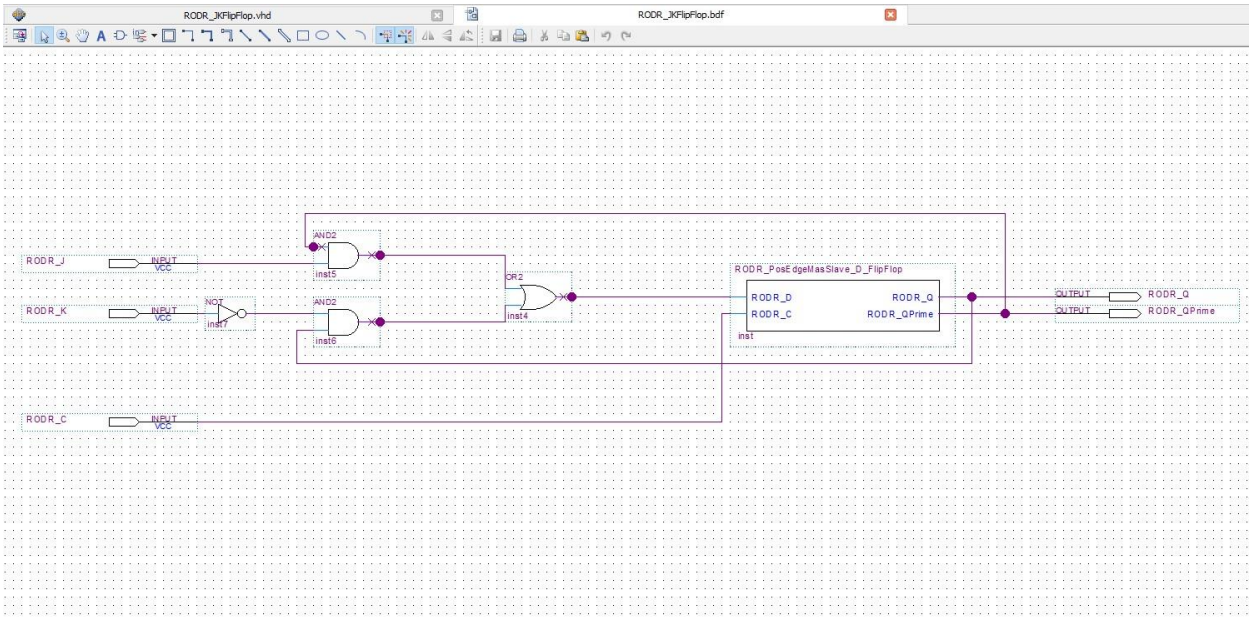

Figure 7: Schematic for JK Flip Flop

In this electronic device, we have created a JK Flip Flop. It uses a D flip flop creating Q and Q' output, and their outputs go to their respective AND gates with either a J or K input for each of them. K is also inverted to ensure that a reset is possible. A Clock signal is used to determine whether or not the JK will change its output.

T Flip Flop

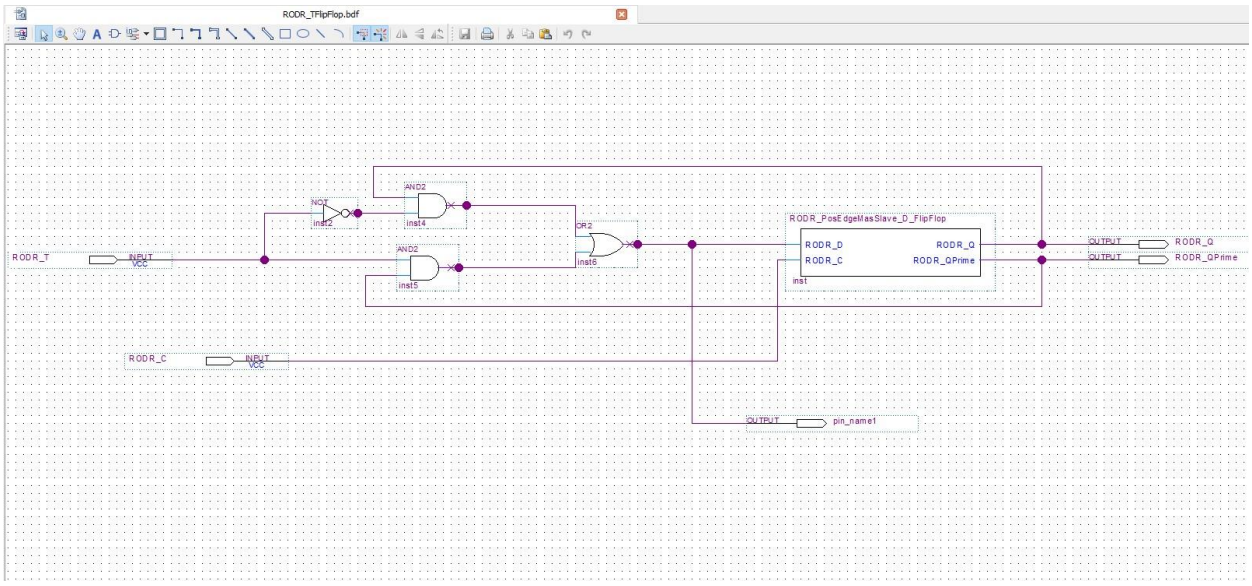

Figure 8: Schematic for T Flip Flop

This electronic device is constructed using a D flip flop with both outputs going back into a multiplexer controlled by the T input. A clock signal is also supplied to know when to change state.

Functionality

SR Latch

If both the reset switch were binary '0', it would continue to hold its previous state for both outputs. If the set switch was set to binary '0' but the reset switch was set to a binary '1', then *RODR_Q* would be set to '0' while *RODR_QPrime* would be it's opposite, '1'. If the set switch was set to binary '1' and the reset switch was off, then *RODR_Q* would be '1' while the other output is set to '0'. But if both reset and set switches are binary '1', then it is an undefined state. It is unpredictable to know what the output would be as it could take in garbage data or no data at all.

S	**R**	**Q**	**Q'**
0	0	No change	No Change
0	1	0	1
1	0	1	0
1	1	Undefined	Undefined

Table 1: SR Latch truth table

This table shows a simplified version of how an SR latch would work. As this electronic device is only level triggered, only binary inputs are needed to define its next state (besides the undefined state).

Control SR Latch

If a clock signal (C) goes through both the S and R terminals of a NAND gate as a binary '0', the previous state will be preserved as is. However, if the signal is a binary '1', it will then work the same as an SR Latch. Again, the outputs would always be the complement of each other as it is the functionality of the electronic device.

C	**S**	**R**	**Q**	**Q'**	**State**
0	X	X	Q	Q'	No Change
1	0	0	Q	Q'	
1	0	1	0	1	
1	1	0	1	0	
1	1	1	Undef.	Undef.	Undefined

Table 2: Truth Table for Control SR Latch

D latch

In this case, it will behave like an SR latch except that the Data (D) signal will determine whether a "Set" or "Reset" will occur. A binary '0' from the clock will not change the current state of the outputs. Meanwhile, if both are binary '1', then *RODR_Q* will be set to '1' and *RODR_QPrime* will be set to '0', still taking the principle of complementing each other. Also, there will be a reset if the clock is on but the data input is at '0'.

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

Table 3: Truth Table for D Latch

Master Slave D Flip Flop

The inversion of the clock signal will depend on whether the flip flop will be positive edge or negative edge. In this case, because the inversion is for the first latch, it will become a positive edge flip flop as the data for the next rising edge of the clock is stored in the first latch. Once the rising edge comes, the output from the first latch would then be transferred into the next latch and output as either a set or reset. This is what makes a latch "level-triggered" and a flip flop "edge-triggered" as the output will only change every rising edge (or falling edge) of a clock signal.

Clock	***D***	***Q***	***Q (Output)***	***Q' (Output)***
0	*X*	*0*	*0*	*1*
0	*X*	*1*	*1*	*0*
1	*X*	*0*	*0*	*1*

1	*x*	*1*	*1*	*0*
Rising Edge	*0*	*X*	*0*	*1*
Rising Edge	*1*	*X*	*1*	*0*

Table 4: Truth table for Positive Edge Master Slave D Flip Flop

Clock	***D***	***Q***	***Q (Output)***	***Q' (Output)***
0	*X*	*0*	*0*	*1*
0	*X*	*1*	*1*	*0*
1	*X*	*0*	*0*	*1*
1	*x*	*1*	*1*	*0*
Falling Edge	*0*	*X*	*0*	*1*
Falling Edge	*1*	*X*	*1*	*0*

Table 5: Truth table for Negative Edge Master Slave D Flip Flop

SR Flip Flop

The SR Flip flop behaves the same as a their own latches. So because of their nature, the SR Flip Flop will enter an undefined state when both SET and RESET are both logical '1' as either one latch or the other will pass in the undefined state.

S	**R**	**Q**	**Q (Output)**	**Q (Output)**
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	x	X
1	1	1	x	x

Table 6: Truth table for a positive edge Master Slave SR Flip Flop

JK Flip Flop

A JK Flip Flop works the exact same as an SR flip flop. The J work exactly like a set and the K works exactly like a reset. However, if both J and K inputs both equal a binary '1', they will then toggle the previous output to be inverted.

J	**K**	**Clock**	**Q**
0	0	1	**Q (no change)**
1	0	1	1
0	1	1	0
1	1	1	**Q' (Toggle)**

Table 7: Truth Table for an JK Flip Flop

T Flip Flop

A T flip flop works only based on the toggling of the T switch. Whenever the switch is a logical '0', it will keep its current state. However, if it is a logical '1', a toggle is then done and thus inverts the output.

T	Q (Current State)	Q' (Next State)
0	0	1
0	1	1
1	0	1
1	1	0

Table 8: Truth Table for T Flip Flop

Simulations

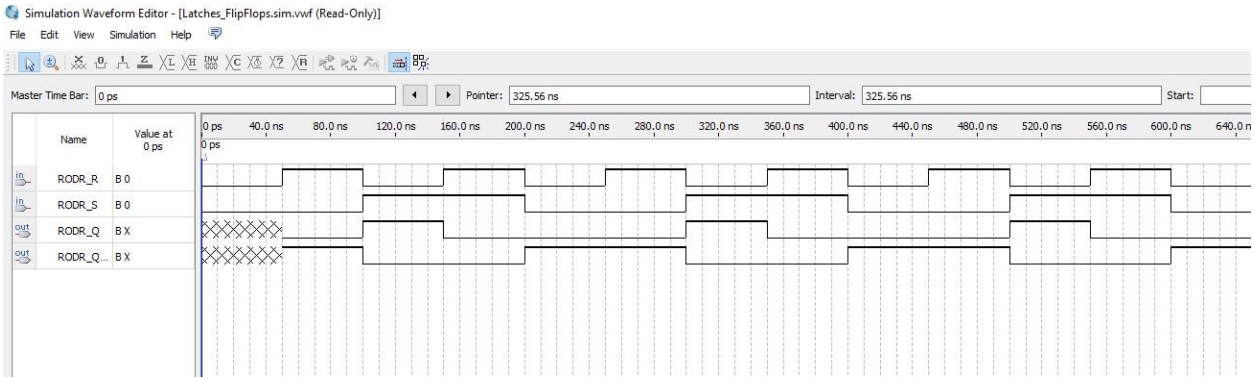

Figure 9: Output Waveform for SR Latch

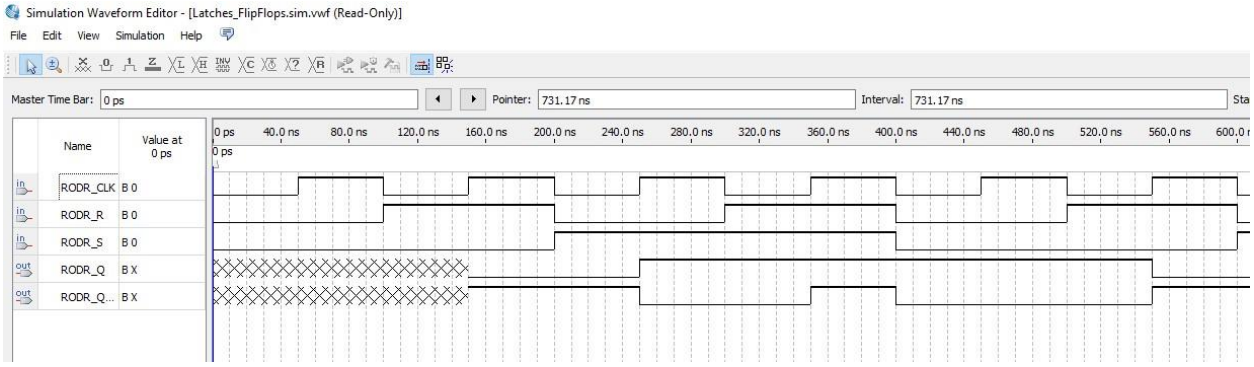

Figure 10: Output Waveform for Control SR Latch

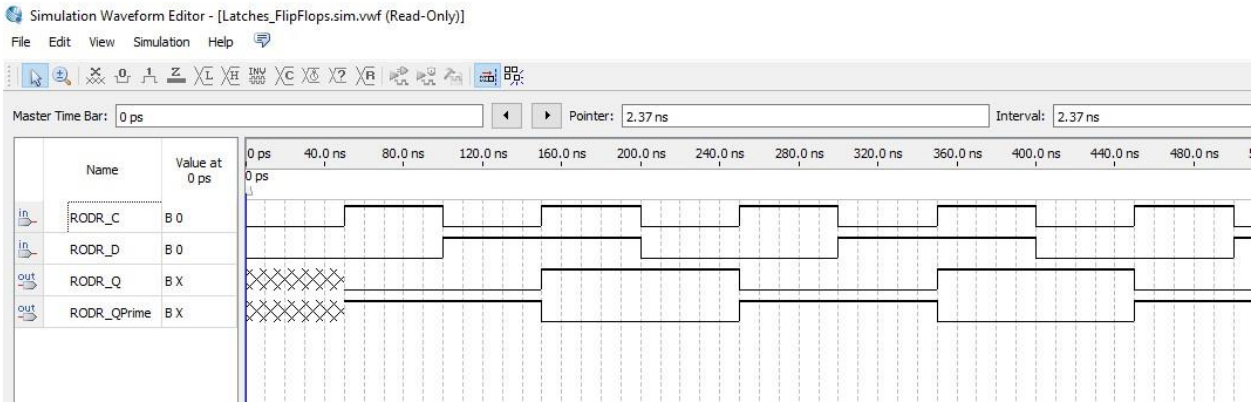

Figure 11: Output Waveform for D Latch

Figure 12: Output Waveform for Master Slave Positive Edge D Flip Flop

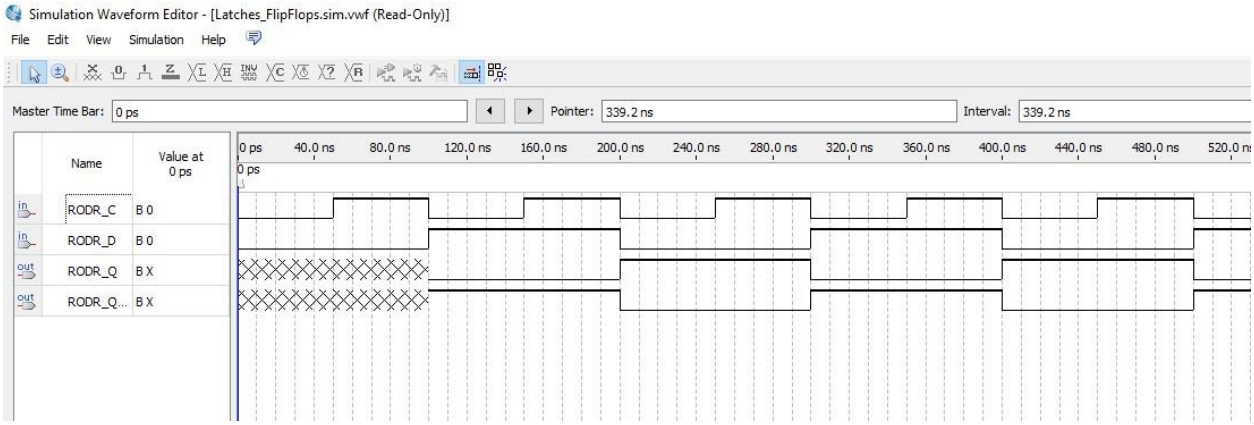

Figure 13: Output Waveform for Master Slave Negative Edge D Flip Flop

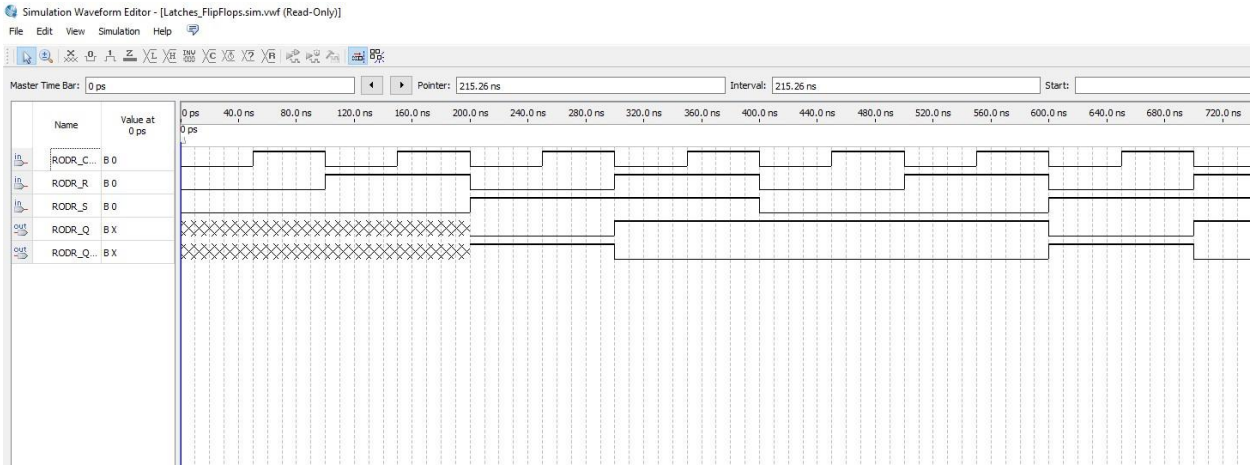

Figure 14: Output Waveform of Positive Edge Master Slave SR Flip Flop

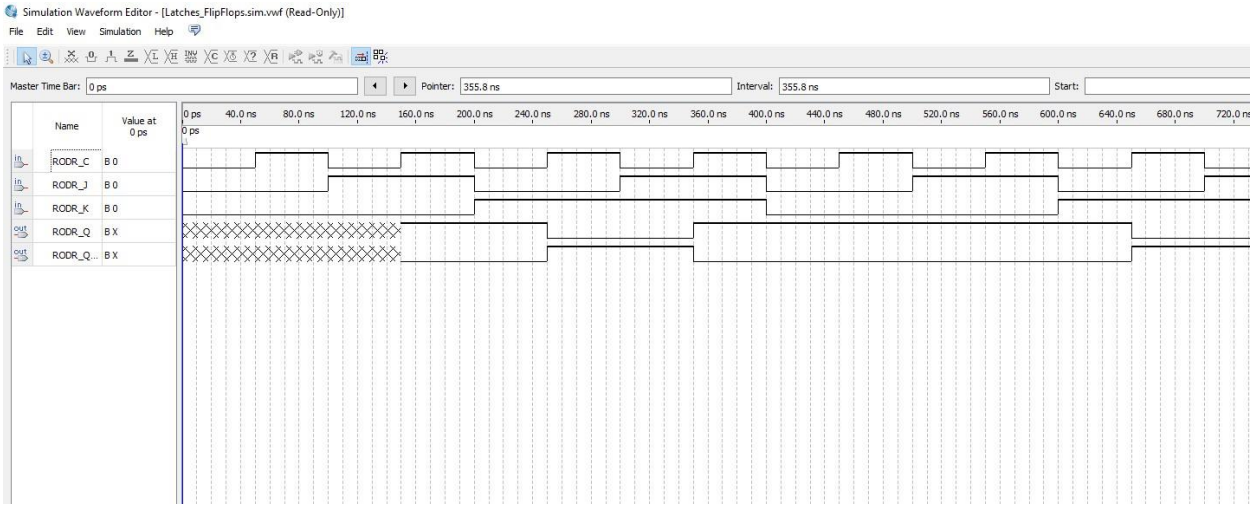

Figure 15: Output Waveform for JK Flip Flop

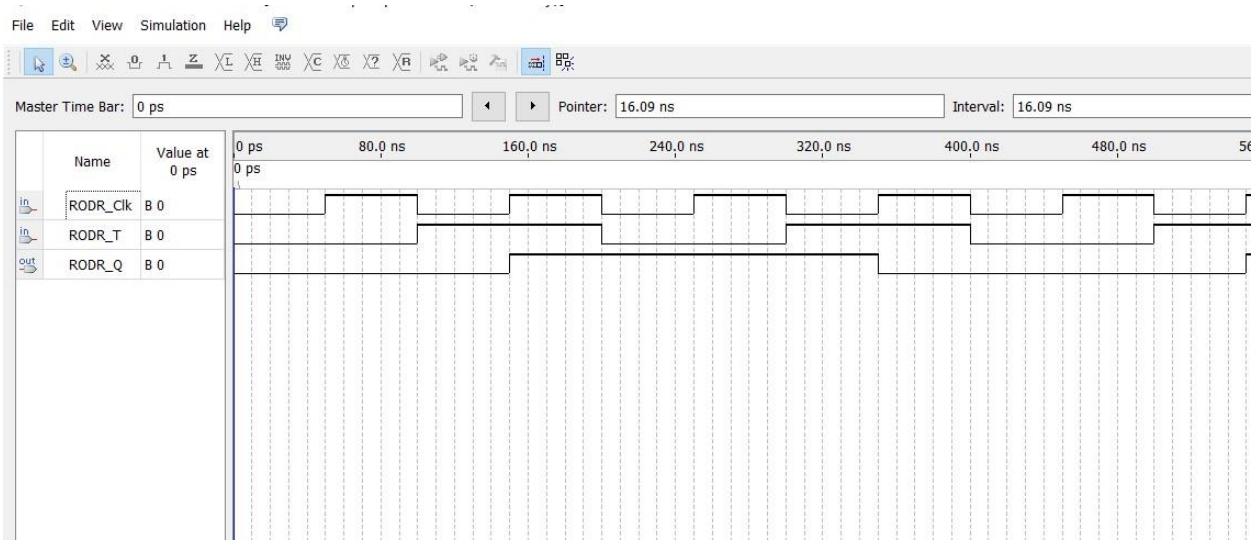

Figure 16: Output Waveform for T Flip Flop

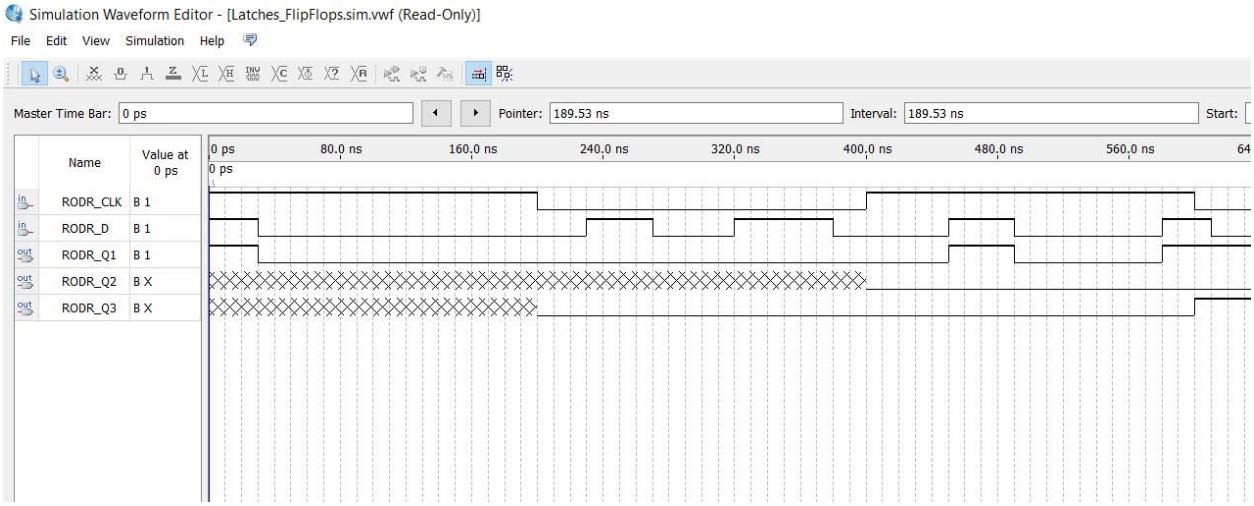

Figure 17: Output for Figure 10 with a Positive D Latch, Positive Edge Triggered D Flip Flop, and Negative Triggered D Flip Flop

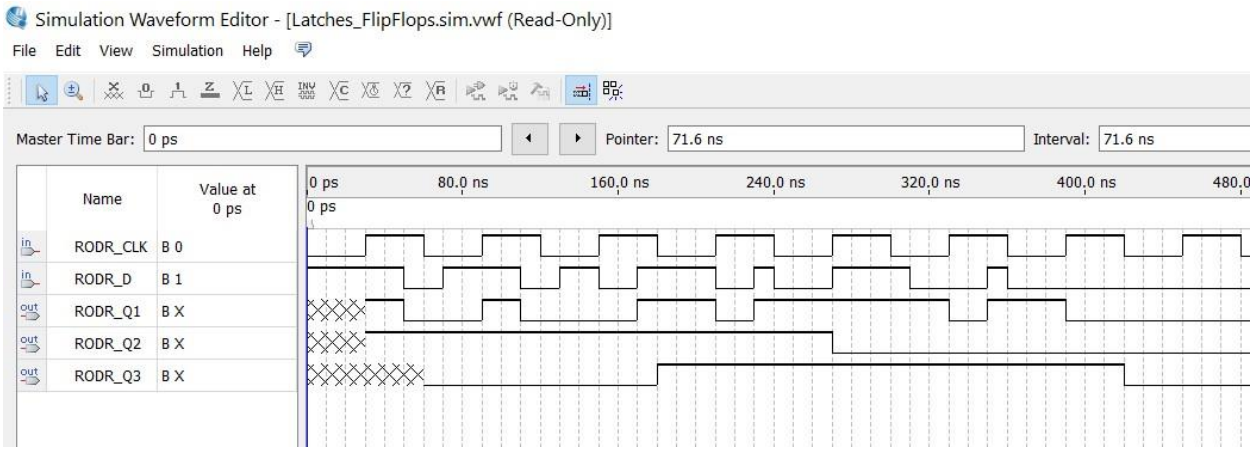

Figure 18: Output Waveform for figure 15 comparing a Positive D Latch, Positive Edge Master Slave D Flip Flop, and Negative Edge Master Slave D Flip Flop

Analysis

- What is the difference between latches and flip-flops? Why does it matter?
  - The difference between latches and flip flops is that latches are level-triggered while flip flops are edge-triggered. This means that latches are dependent on the inputs of their circuits ONLY, while flip flops are dependent on the rising or falling edge of the clock signal. This matters because we can control and save inputs when the clock is low and push the desired output out at will.
- Why do we want to avoid the scenario when S=1 and R=1 in an SR Latch? Could this scenario happen in a D Latch? Why or Why not?
  - We want to avoid the scenario when S=1 and R=1 in an SR latch because it is an unknown state. This means that we don't know what the output would be and could potentially corrupt any data flow within it. This scenario cannot happen in a D latch because the D input controls both AND gate control with the clock. Meaning that the output will only either set or reset when the clock is at an active high.
- What is the difference between edge-triggered and level-triggered devices? Is the T Flip flop you built in this lab level or edge triggered, for instance?
  - The difference between edge-triggered and level-triggered devices is that the former is mainly based on master slave flip flop devices where the clock's rising or falling edge determines if the outputs are to be changed or not. On the other hand, the latter changes automatically when the output changes (and the clock is active high). The T flip flop in this lab is an edge-triggered flip flop as it changes at the moment in which the clock changes state.
- Are flip flops always required to be clocked? Why or why not?
  - Flip flops are always required to be clocked because then there would be no storage and movement of data. Also, it would be a problem if we want a desired

outcome for a longer period of time but aren't able to because the data continuously changes, and the user has no real control of that. So there would be potential chaos when talking between registers.

- Compare the behavior of the D Latch and D Flip Flop (positive and negative) outputs in figure 15
    - The D latch and D Flip flop change states on different points of the waveform. For the D latch, the output changes whenever the clock is at an active high. It simply changes based on the Data levels. But once the clock is at an active low, it retains its previous state. For the positive D flip flop, it changes at the rising edge of the clock and outputs the state in which the clock instantaneously lands on. After that occurs, it retains that state until the next rising edge of the clock. The negative-edge D flip flop is the exact same as the positive-edge D flip flop, except it changes on every negative edge of the clock.
    -

Conclusion

From this assignment, I learned a lot about the differences between latches and flip flops. At the core, flip flips are just a combination of either their respective latches or can use a D flip flop to create SR or T flip flops using something like a multiplexer. With latches and flip flops, you can build registers that are able to store information based on a single clock signal. Specifically, for D flip flops, If we cascade multiples of them, we can create a register that stores bit sequences from the switches or an array of binary numbers and store them for later use for a different operation. We can also use them to communicate memory address locations to the microcontroller for programming purposes (mainly in assembly).