

Simulation Exercise 6  
Introduction to Sequential Circuits  
Latches and Flip Flops  
ELEE 2640

Benyamain YACOOB

March 17, 2024

Date Performed: March 17, 2024

Partners: Ara OLADIPO  
Andre PRICE

Instructor: Professor PAULIK



## Contents

<b>1 Objectives</b>	<b>3</b>
<b>2 Problem Statement</b>	<b>3</b>
<b>3 Materials</b>	<b>3</b>
<b>4 Requirements</b>	<b>3</b>
<b>5 Sequential Systems</b>	<b>9</b>
5.1 NOR Gate Chain . . . . .	9
5.2 Set-Reset Latch . . . . .	10
5.2.1 Constraints File . . . . .	10
5.2.2 Set-Reset Testbench Code . . . . .	11
5.2.3 Set-Reset Latch Dataflow Implementation . . . . .	11
5.2.4 Set-Reset Latch Gate-Level Implementation . . . . .	13
5.3 Gated Set-Reset Latch . . . . .	15
5.4 Results and Analysis Discussion . . . . .	18
<b>6 D Latches and Flip Flops</b>	<b>19</b>
6.1 Three Flip Flops Comparison . . . . .	19
6.2 D Flip Flop with Synchronous Reset and Clock Enable . . . . .	22
6.3 Results and Analysis Discussion . . . . .	25
<b>7 Results and Conclusion</b>	<b>25</b>
<b>8 Group Contributions</b>	<b>26</b>

# 1 Objectives

## First Objective

Explore feedback in digital circuits using an oscilloscope to visualize behavior.

## Second Objective

Build a basic NOR gate latch and implement in both MultiSIM and physical hardware.

## Third Objective

Code in SystemVerilog for introductory sequential systems.

## Fourth Objective

Model latches and flip flops.

# 2 Problem Statement

Sequential circuits are digital circuits in which the output depends not only on the present input (like combinatorial circuits) but also on the past sequence of inputs. In effect, these circuits must be able to remember something about the past history of the inputs. Thus the concept of timing is introduced, with the clock signal providing the timing structure for sequential circuits. Latches and flip flops are commonly used memory devices in sequential circuits and both will be explored in this simulation exercise.

# 3 Materials

NI MultiSIM

Xilinx integrated synthesis environment (ISE)

# 4 Requirements

## NOR Gate Chain

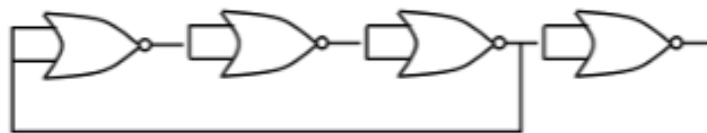


Figure 1: A Sequence of NOR Gate Inverters with Feedback

## SR Latch

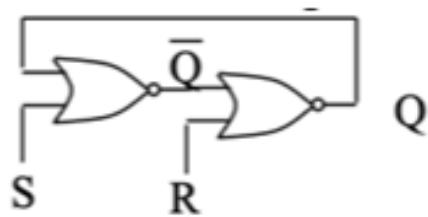


Figure 2: SR Latch Linear Layout with Feedback (NOR Gate Implementation)

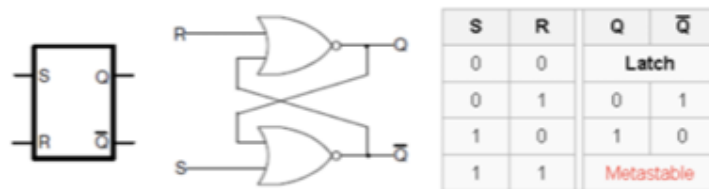


Figure 3: SR Latch Symbol, NOR Gate Implementation, and Truth Table

### Gated SR Latch

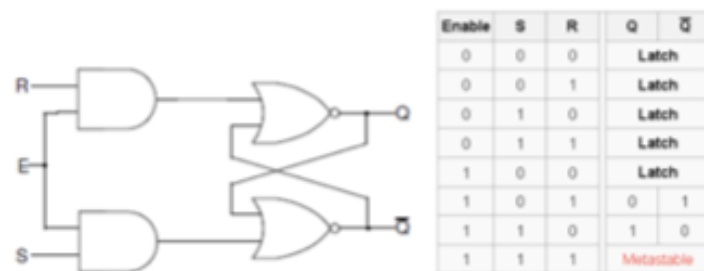


Figure 4: Gated SR Latch and Truth Table



Figure 5: Illustration of a Testbench Post Implementation Simulation Result for a Gated SR Latch

## Gated D Latch

Gated D latch truth table				
E/C	D	Q	$\bar{Q}$	Comment
0	X	$Q_{prev}$	$\bar{Q}_{prev}$	No change
1	0	0	1	Reset
1	1	1	0	Set

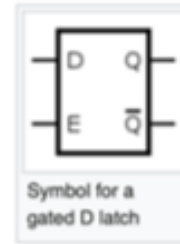


Figure 6: Truth Table and Symbol

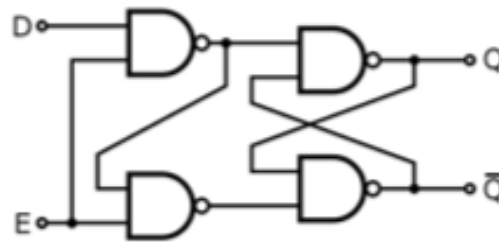


Figure 7: A Gated D Latch Based on a SR NAND Latch

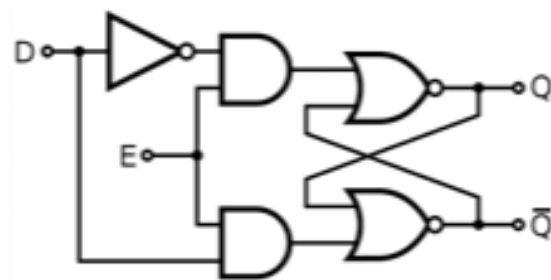


Figure 8: A Gated D Latch Based on an SR NOR Latch

## Gate Level Edge Triggered D Flip Flop

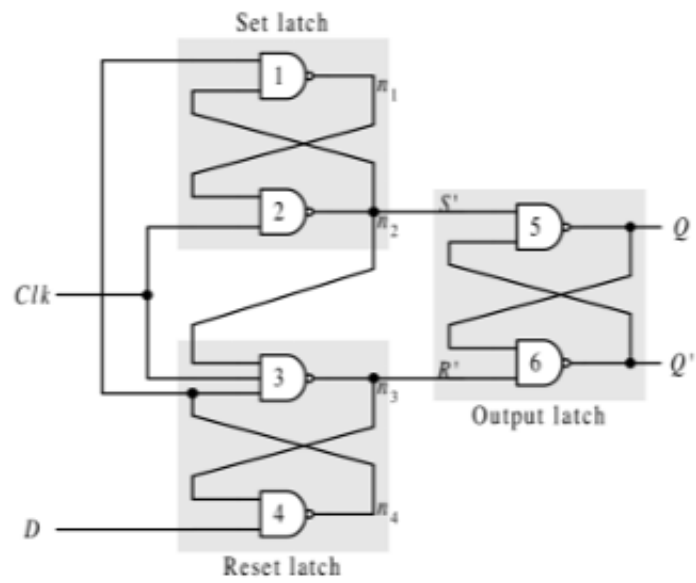


Figure 9: Positive Edge Triggered D Flip Flop at the Gate Level

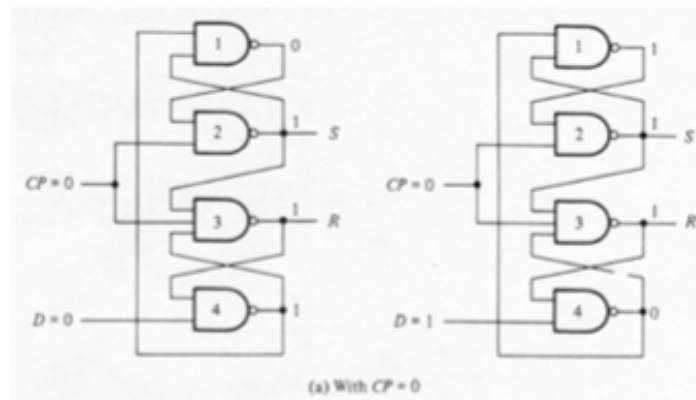


Figure 10: Clock = 0, D = 0/1, SR Fixed at 1 & 1

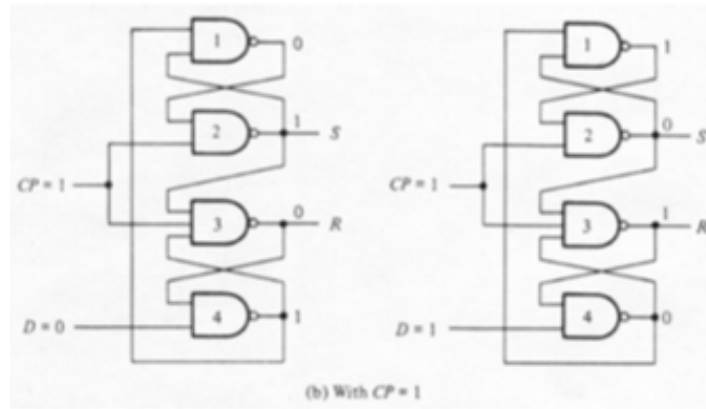


Figure 11: Clock = 1,  $D = 0/1$ , SR Fixed at 1 & 0 or 0 & 1

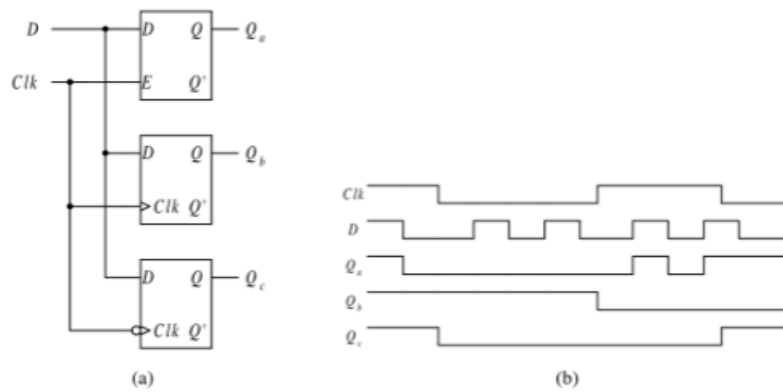


Figure 12: Comparison of a gated latch, a positive-edge-triggered flip-flop, and a negative-edge-triggered flip-flop: (a) circuit; (b) timing diagram.

Figure 12: Comparison of a Gated Latch, a Positive Edge Triggered Flip Flop, and a Negative Edge Triggered Flip Flop for the Given Circuit and Timing Diagram

### Flip Flop Based Digital Delay Line (Sequential Design)

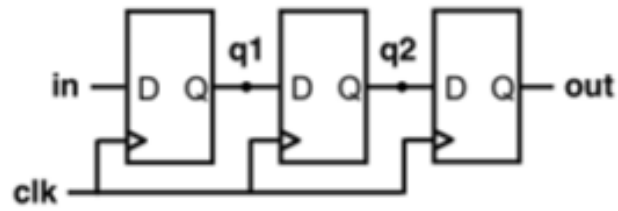


Figure 13: Three Edged Triggered D Flip Flops In a Delay Line Configuration

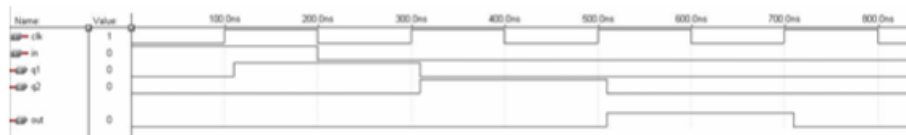


Figure 14: Nonblocking Delay Line Simulation Desired Outcome

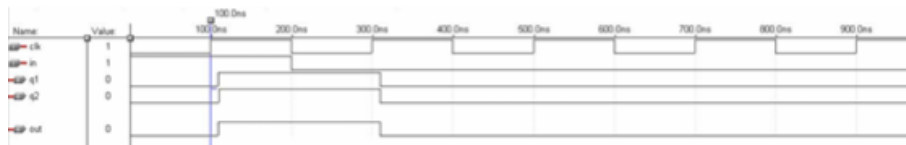


Figure 15: Blocking Simulation Undesired Outcome

### Combinational Logic Blocking/Nonblocking Code Example

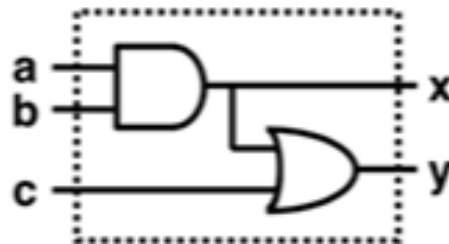


Figure 16: AND OR Gate Circuit for Blocking/Nonblocking Combinational Logic Study



## Synchronous D Flip Flop with Set and Reset

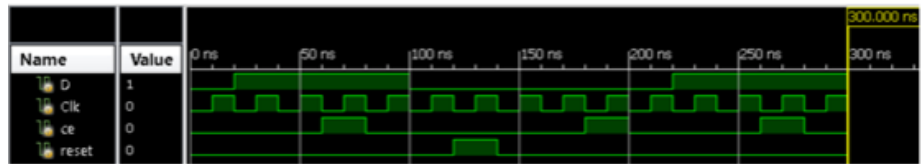


Figure 17: D Flip Flop with Synchronous Reset and Clock Enable Partial Waveform

## 5 Sequential Systems

### 5.1 NOR Gate Chain

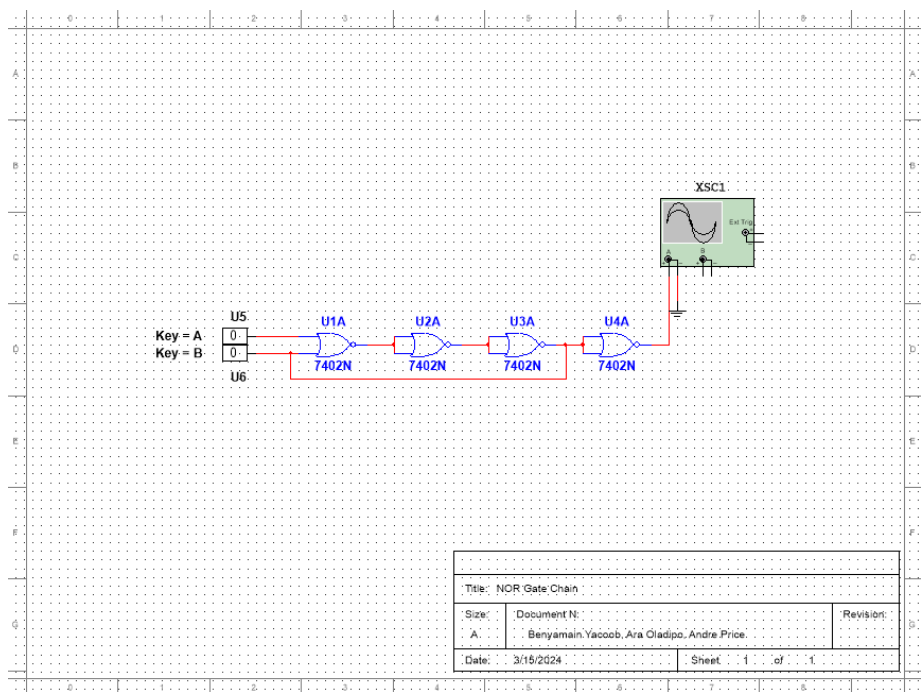


Figure 18: NOR Gate Chain

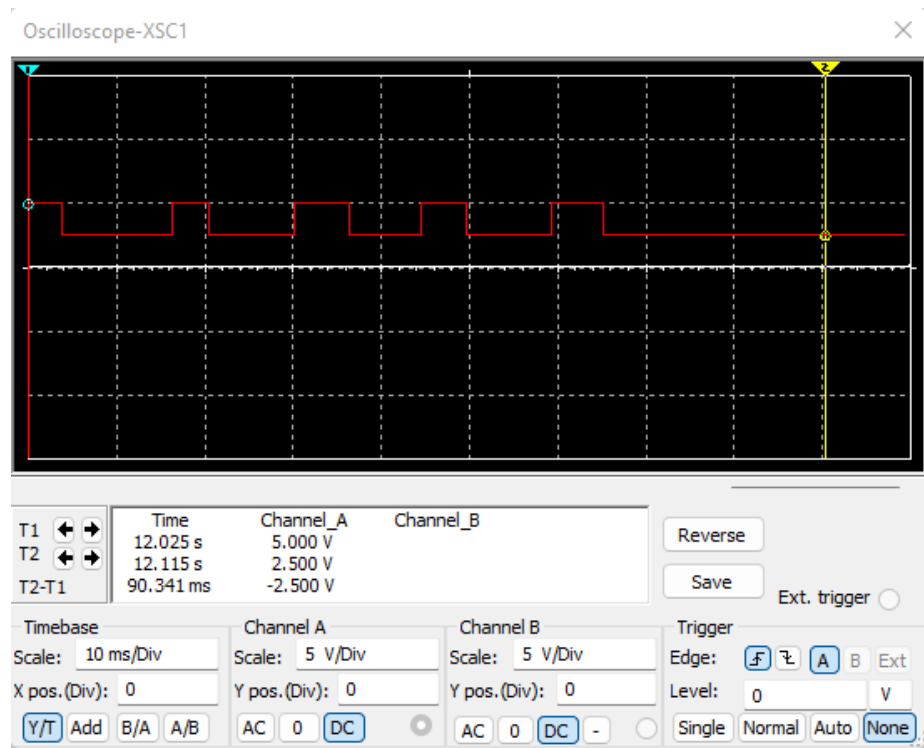


Figure 19: NOR Gate Scope Waveform

## 5.2 Set-Reset Latch

### 5.2.1 Constraints File

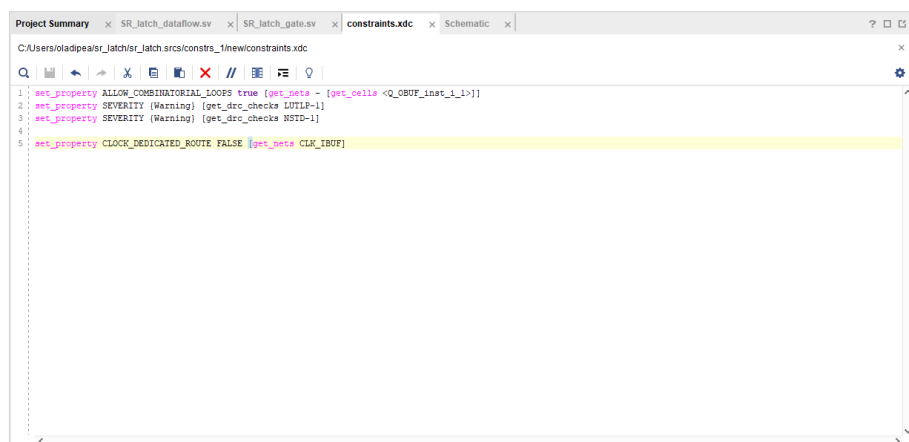
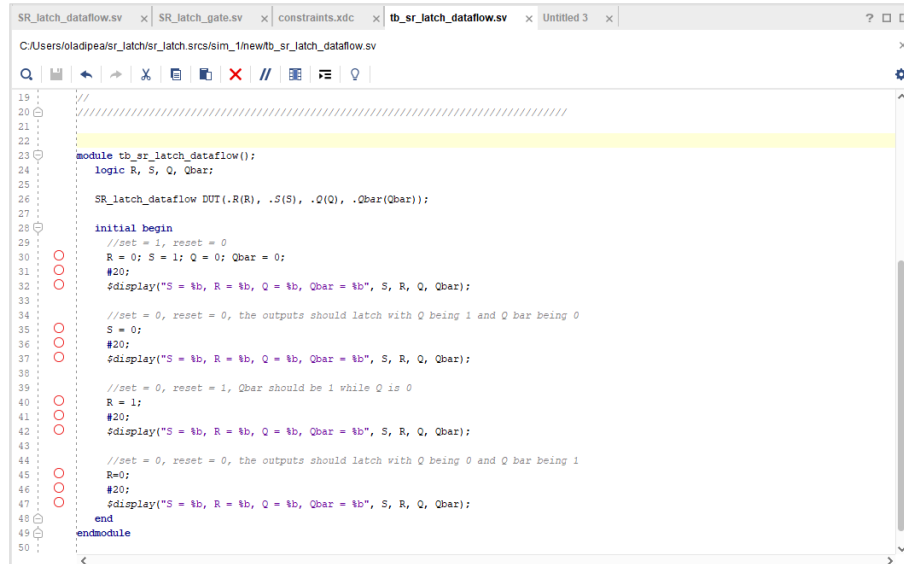


Figure 20: Constraints File

### 5.2.2 Set-Reset Testbench Code

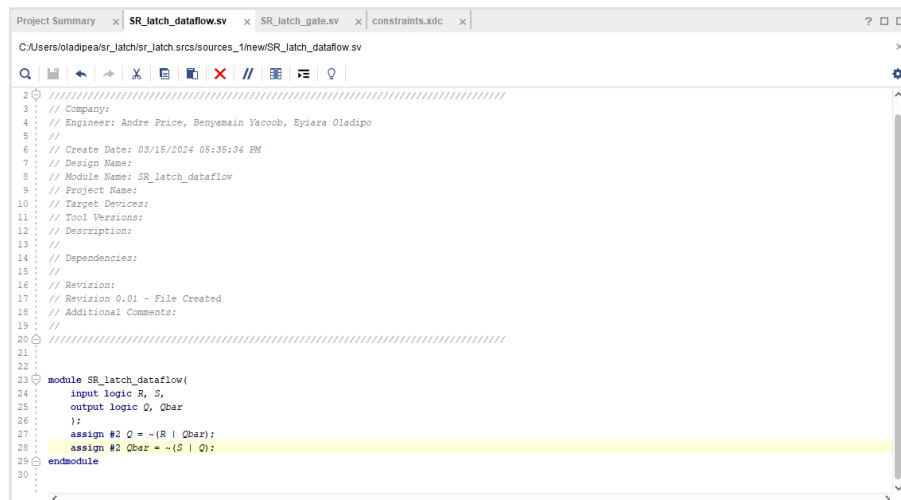


```
SR_latch_dataflow.vv | SR_latch_gate.vv | constraints.xdc | tb_sr_latch_dataflow.vv | Untitled 3 |
C:/Users/oladipea/sr_latch/sr_latch/srcs/sim_1/new/tb_sr_latch_dataflow.vv

19 //
20 ///////////////////////////////////////////////////////////////////
21
22
23 module tb_sr_latch_dataflow()
24     logic R, S, Q, Qbar;
25
26     SR_latch_dataflow DUT(.R(R), .S(S), .Q(Q), .Qbar(Qbar));
27
28     initial begin
29         //set = 1, reset = 0
30         R = 0; S = 1; Q = 0; Qbar = 0;
31         #20;
32         $display("S = %b, R = %b, Q = %b, Qbar = %b", S, R, Q, Qbar);
33
34         //set = 0, reset = 0, the outputs should latch with Q being 1 and Q bar being 0
35         S = 0;
36         #20;
37         $display("S = %b, R = %b, Q = %b, Qbar = %b", S, R, Q, Qbar);
38
39         //set = 0, reset = 1, Qbar should be 1 while Q is 0
40         R = 1;
41         #20;
42         $display("S = %b, R = %b, Q = %b, Qbar = %b", S, R, Q, Qbar);
43
44         //set = 0, reset = 0, the outputs should latch with Q being 0 and Q bar being 1
45         R = 0;
46         #20;
47         $display("S = %b, R = %b, Q = %b, Qbar = %b", S, R, Q, Qbar);
48     end
49 endmodule
50
```

Figure 21: Set-Reset Testbench Code

### 5.2.3 Set-Reset Latch Dataflow Implementation



```
Project Summary | SR_latch_dataflow.vv | SR_latch_gate.vv | constraints.xdc |
C:/Users/oladipea/sr_latch/sr_latch/srcs/sources_1/new/SR_latch_dataflow.vv

2 //
3 // Company:
4 // Engineer: Andre Price, Benyamin Yacoub, Eyiara Oladipo
5 //
6 // Create Date: 03/15/2024 06:35:34 PM
7 // Design Name:
8 // Module Name: SR_latch_dataflow
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22
23 module SR_latch_dataflow(
24     input logic R, S,
25     output logic Q, Qbar
26 );
27     assign #2 Q = ~(R | Qbar);
28     assign #2 Qbar = ~(S | Q);
29 endmodule
30
```

Figure 22: Set-Reset Dataflow Implementation Code

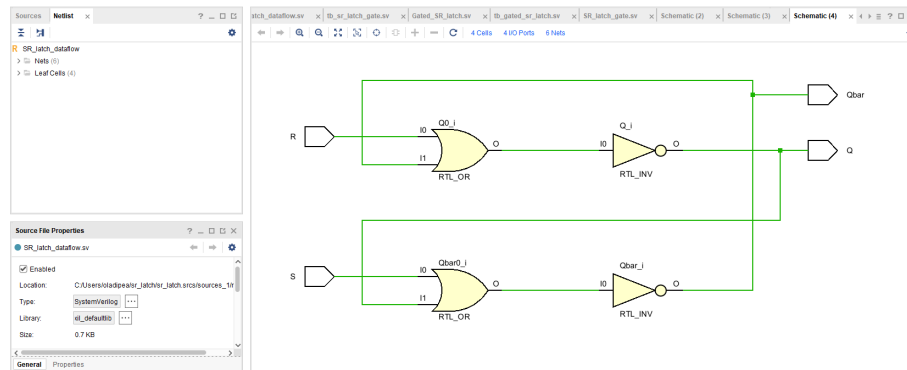


Figure 23: Set-Reset Dataflow Implementation RTL Schematic

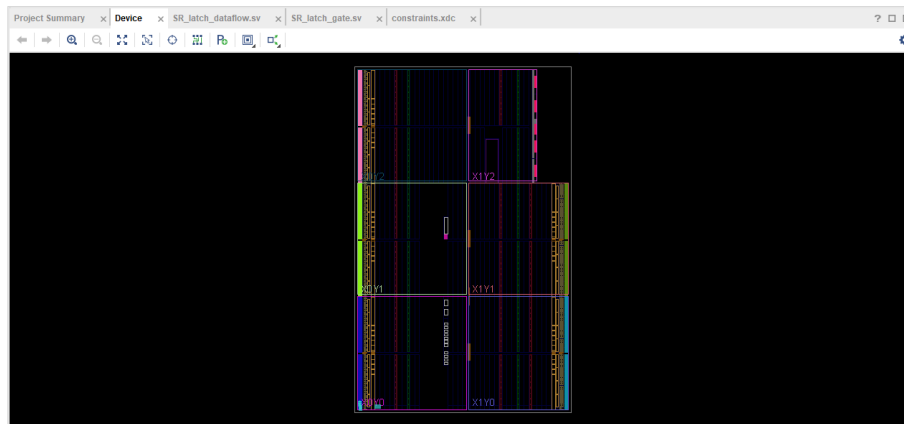


Figure 24: Set-Reset Dataflow Implementation Synthesized Design

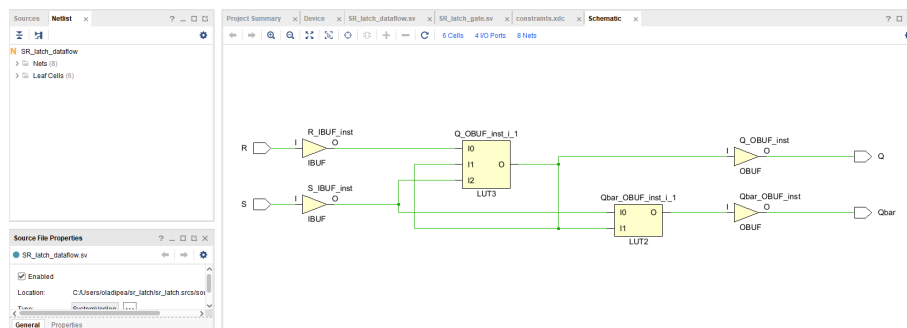


Figure 25: Set-Reset Dataflow Implementation Synthesized Schematic

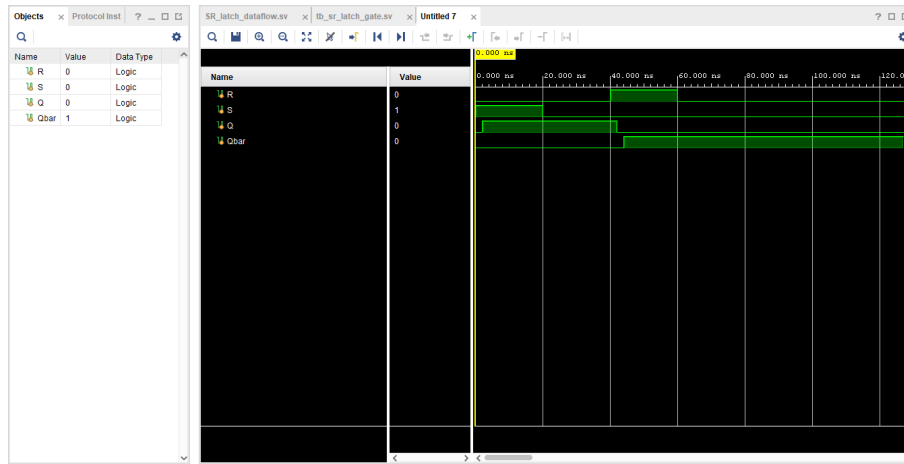


Figure 26: Set-Reset Dataflow Testbench Behavioral Simulation

```

# run 100ns
S = 1, R = 0, Q = 1, Qbar = 0
S = 0, R = 0, Q = 1, Qbar = 0
S = 0, R = 1, Q = 0, Qbar = 1
S = 0, R = 0, Q = 0, Qbar = 1
INFO: [DPF-K11a-14] XSim completed. Design snapshot "1b_sr_dataflow_behav" loaded.
INFO: [DPF-K11a-15] XSim simulation ran for 100ns

```

Figure 27: Set-Reset Dataflow Testbench TCL Console

## 5.2.4 Set-Reset Latch Gate-Level Implementation

```

Project Summary | Device | SR_latch_dataflow.v | SR_latch_gate.v | constraints.xdc
C:\Users\oladipealr_latch\srcs\sources_1\new\SR_latch_gate.v

8 // Module Name: SR_latch_gate
9 // Project Name:
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22 module SR_latch_gate(
23     input R,
24     input S,
25     output Q,
26     output Qbar
27 );
28
29     nor(Q, R, Qbar);
30     nor(Qbar, S, Q);
31
32 endmodule
33
34

```

Figure 28: Set-Reset Gate-Level Implementation Code

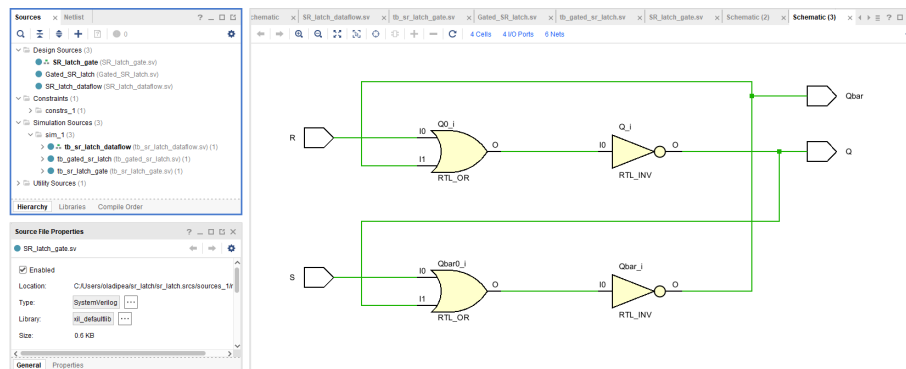


Figure 29: Set-Reset Gate-Level Implementation RTL Schematic

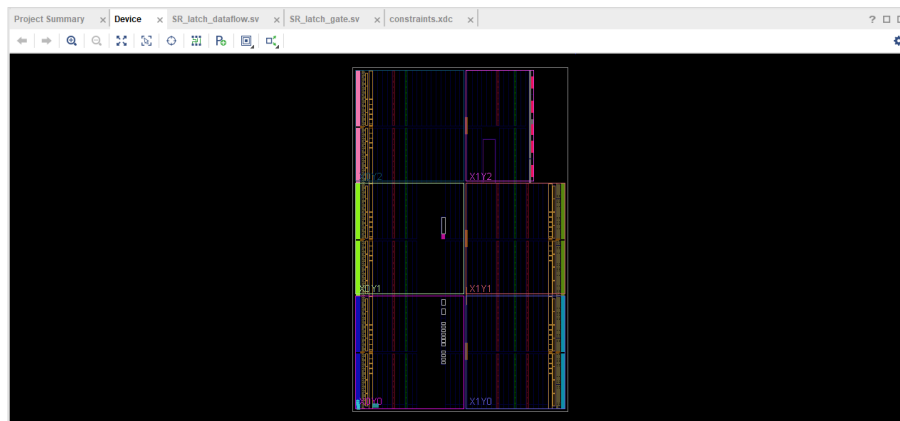


Figure 30: Set-Reset Gate-Level Implementation Synthesis Design

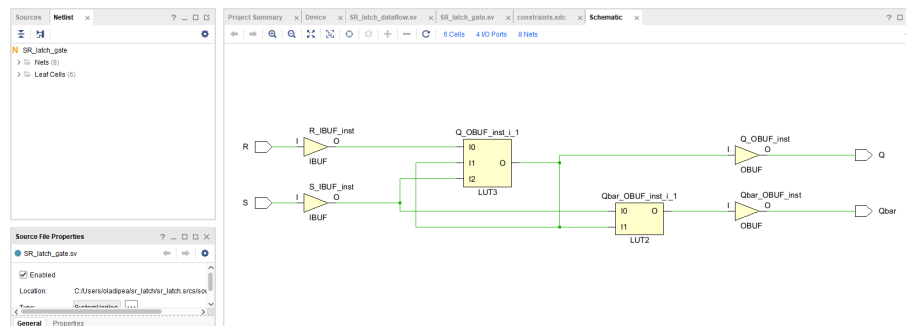


Figure 31: Set-Reset Gate-Level Implementation Synthesized Schematic

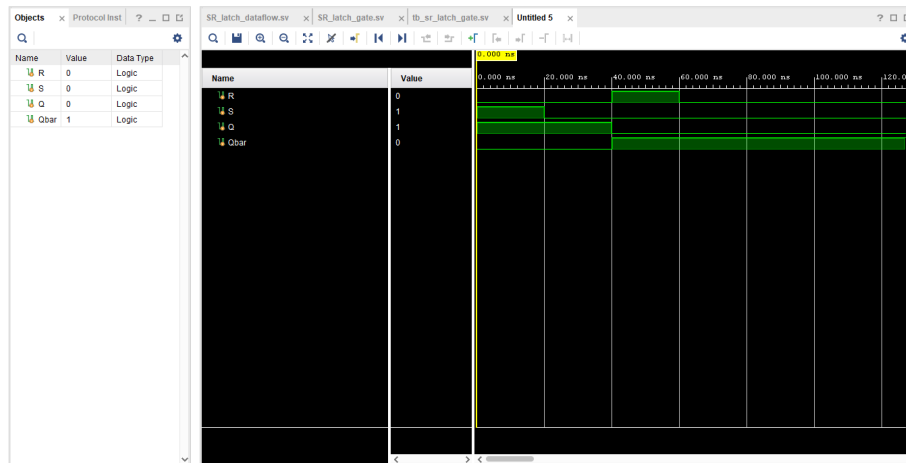


Figure 32: Set-Reset Gate-Level Testbench Behavioral Simulation

```

1 //
2 //
3 // run 1000ns
4 S = 1, R = 0, Q = 1, Qbar = 0
5 S = 0, R = 0, Q = 1, Qbar = 0
6 S = 0, R = 1, Q = 0, Qbar = 1
7 S = 0, R = 0, Q = 0, Qbar = 1
8 INFO: (USF-XS1a-97) XSim compilation. Design mapsheet 'tb_sr_latch_gate_behav' loaded.
9 INFO: (USF-XS1a-97) XSim simulation ran for 1000ns

```

Figure 33: Set-Reset Gate-Level Testbench TCL Console

## 5.3 Gated Set-Reset Latch

```

12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22
23 module Gated_SR_latch(
24     input logic R, S,
25     output logic Q, Qbar
26 );
27
28     logic internalS, internalR;
29
30     assign internalS = S & S;
31     assign internalR = S & R;
32
33     assign #2 Q = ~(internalR | Qbar);
34     assign #2 Qbar = ~(internalS | Q);
35 endmodule
36

```

Figure 34: Gated Set-Reset Latch Implementation Code

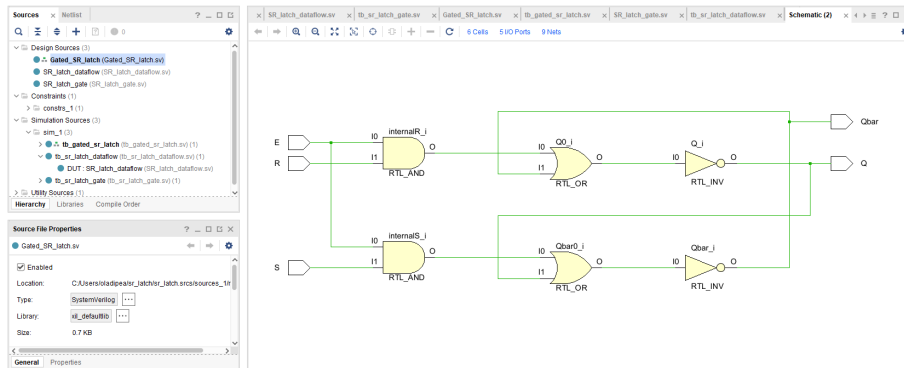


Figure 35: Gated Set-Reset Latch RTL Schematic

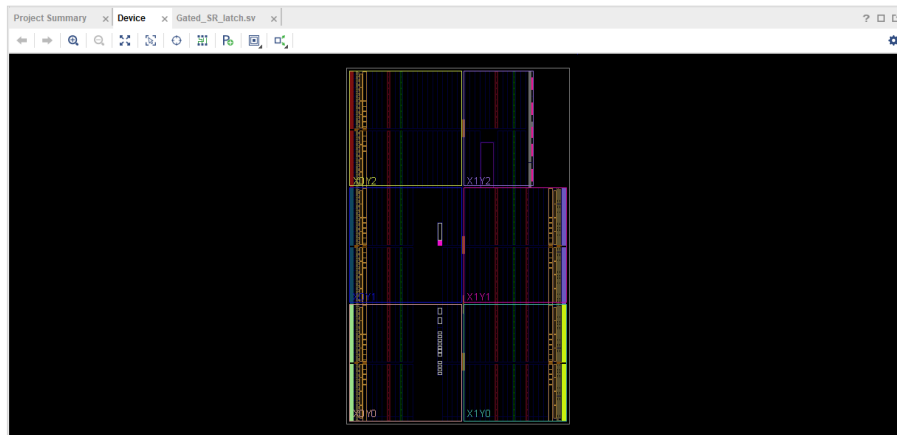
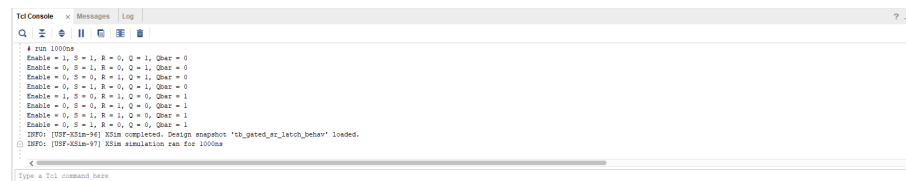
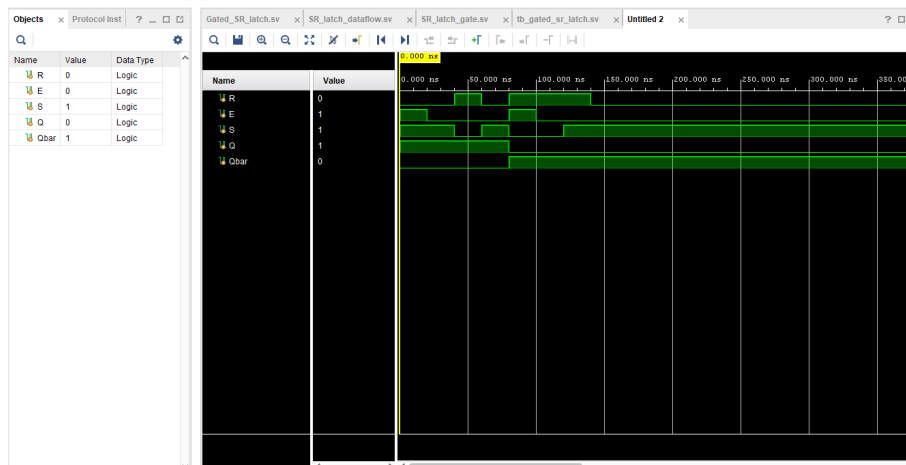
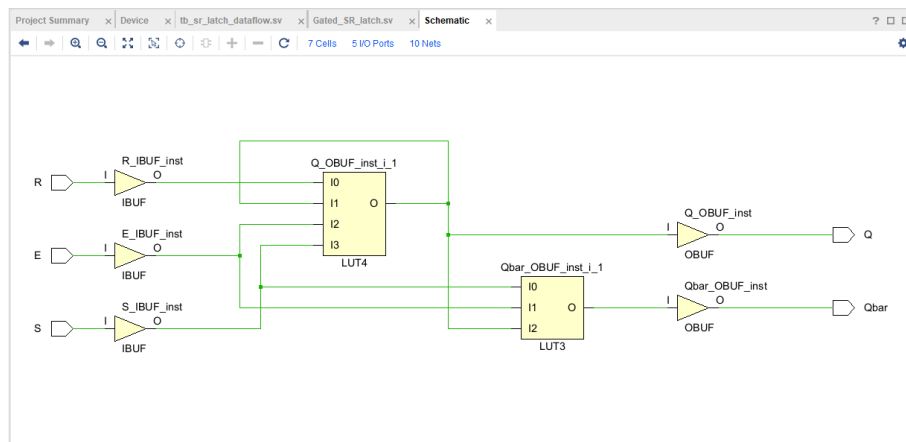


Figure 36: Gated Set-Reset Latch Synthesized Design





## 5.4 Results and Analysis Discussion

The purpose of this section was to create a NOR-gate circuit within the MultiSIM software while also utilizing a feedback path, implement a SR Latch with feedback, and to implement a Gated SR Latch.

Going over the results for the NOR-gate scope, we can see many different attributes. The vertical axis consists of *Channel A* which is measured in voltage. The horizontal axis consists of the *timebase* and is measured in seconds. By looking at the scope, we can see a constant loop of voltage going from positive to negative. This can also be highlighted in the console of the scope, where we can see a positive voltage reading of 5 and a negative voltage reading of 2.5 for the same measurement type. To note, the scale for the *timebase* was 10 *ms/Div* while the scale for the *Channel A* was 5 *V/Div*.

As it pertains to the difficulty of the task, at least when it comes to the NOR-gate chain, granted the same characteristic can be applied to the other parts of this section, the original rubric for the assignment was inherently correct in stating outright that the exercise would be relatively simple. Given our collective proficiency in MultiSIM, it took mere minutes for the group to completely finish the chain and feedback. However, that was because of comprehensive conceptual familiarity of the subject matter at hand.

The ideas of both the SR Latch and the Gated SR Latch were new, but that does not mean that the theoretical concepts of these topics were difficult; in fact, the opposite was apparent. The introductory analysis displayed in the assignment rubric provided a thorough explanation of what the SR latches encompassed. The only notable highlight or “struggle” with this section was that it felt like “busy” work, which also is not inherently a negative attribute of the simulation assignment. It felt more or less like your usual simulation assignment with an added guise of repetition and debilitation.

## 6 D Latches and Flip Flops

### 6.1 Three Flip Flops Comparison

```
tb_d_bb.v  D_ff_behavior.v
C:\Users\ben\sim_assg_7\sim_assg_7\srcs\sources_1\imports\Downloads\D_ff_behavior.v

12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22
23 module D_ff_behavior(
24     input logic D, Clk,
25     output reg Q_lat, Q_pos, Q_neg
26 );
27
28     always_ff @(Clk, D) begin
29         if (Clk == 1)
30             Q_lat <= D;
31     end
32
33     always_ff @(posedge Clk) begin
34         Q_pos <= D;
35     end
36
37     always_ff @(negedge Clk) begin
38         Q_neg <= D;
39     end
40
41 endmodule
```

Figure 40: Three Flip Flops Comparison Code Implementation

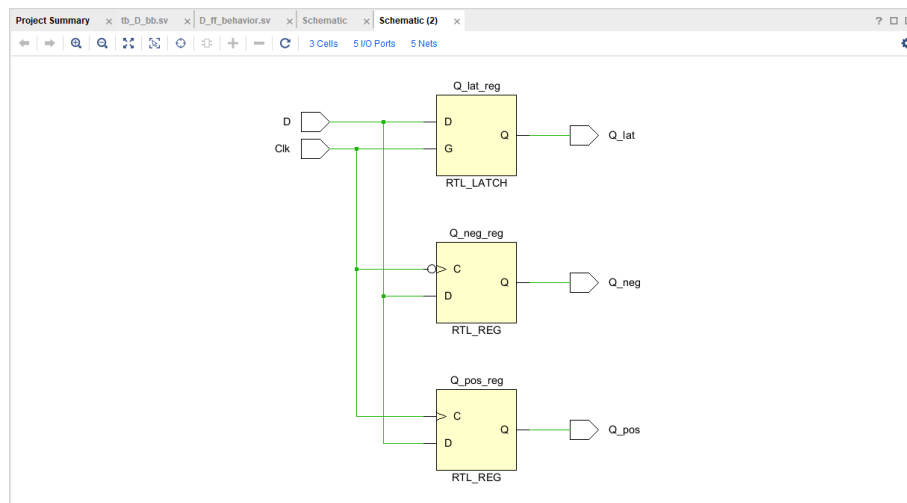


Figure 41: Three Flip Flops Comparison RTL Schematic

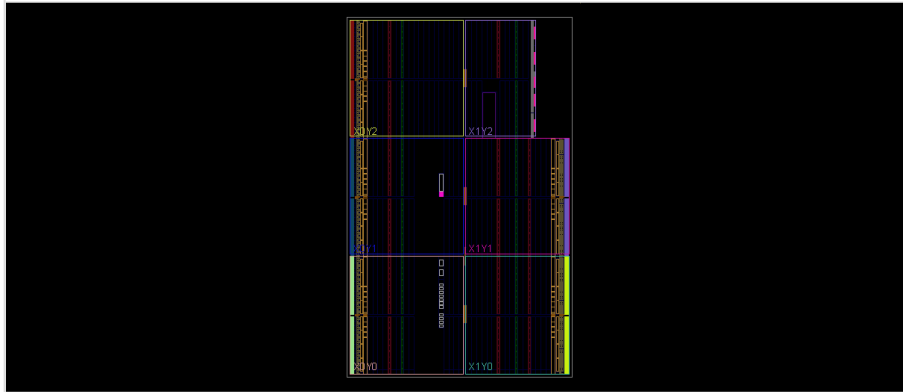


Figure 42: Three Flip Flops Comparison Synthesis Design

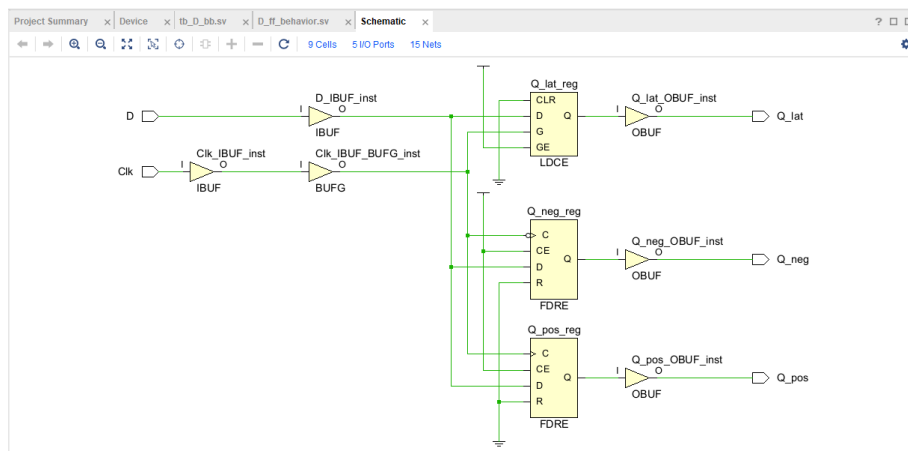


Figure 43: Three Flip Flops Comparison Synthesized Design Schematic

```

tb_D_bb sv x D_ff_behavior sv x Untitled 1 x
C:/Users/ben/sim_assg_7/sim_assg_7/srcs/sim_1/imports/Downloads/tb_D_bb sv

22
23 module tb_D_bb();
24
25     // Should always be on for purposes of tb (as shown in the figure)
26     logic D = 1;
27     logic Clk = 1;
28
29     reg Q_lat, Q_pos, Q_neg;
30
31     D_ff_behavior dffb(
32         .D(D),
33         .Clk(Clk),
34         .Q_lat(Q_lat),
35         .Q_pos(Q_pos),
36         .Q_neg(Q_neg)
37     );
38
39
40     // Reverse the clock to be on or off
41     always #30 Clk = ~Clk;
42

```

Figure 44: Three Flip Flops Testbench Code

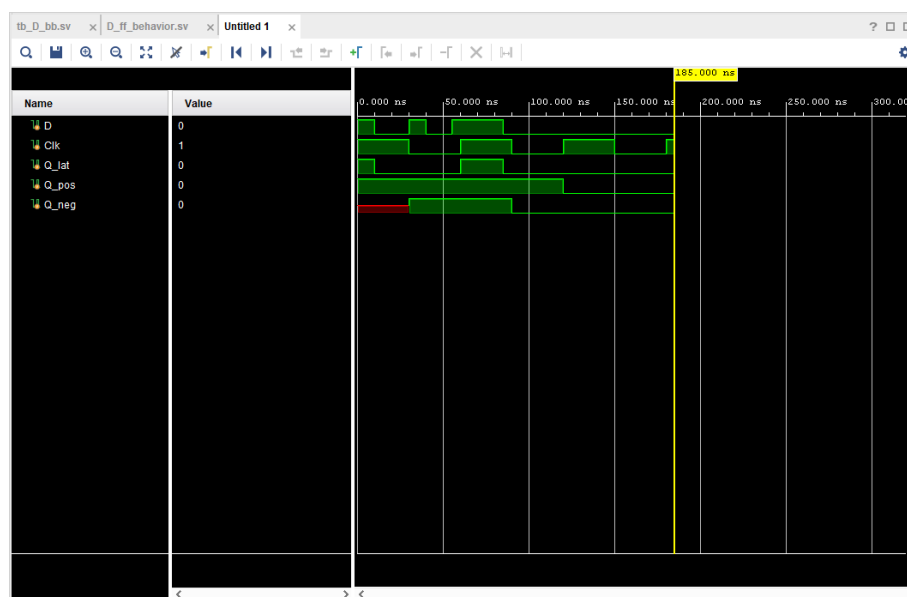


Figure 45: Three Flip Flops Testbench Behavioral Simulation

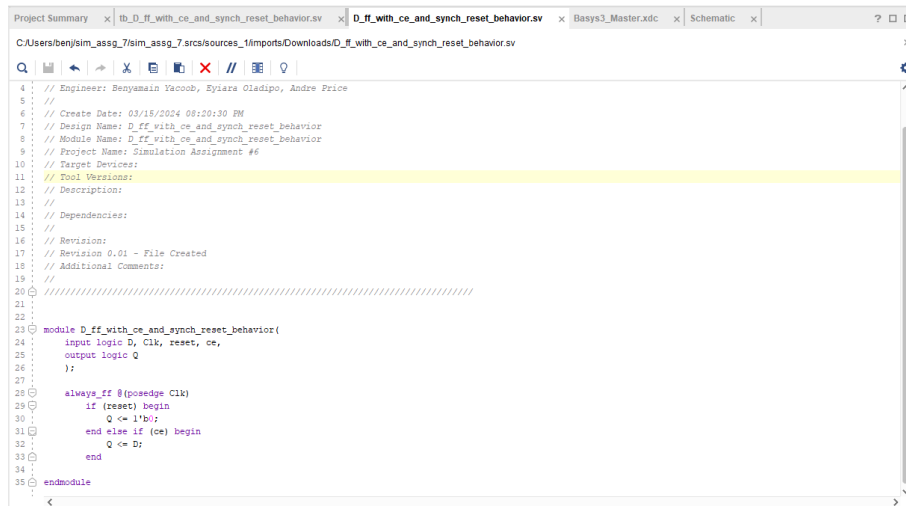
```

# run 1000ns
D = 0, Clk = 1, Q_latch = 0, Q_posedge = 1, Q_negedge = 1
D = 0, Clk = 0, Q_latch = 0, Q_posedge = 1, Q_negedge = 0
D = 0, Clk = 1, Q_latch = 0, Q_posedge = 0, Q_negedge = 0
D = 0, Clk = 0, Q_latch = 0, Q_posedge = 0, Q_negedge = 0
D = 0, Clk = 1, Q_latch = 0, Q_posedge = 0, Q_negedge = 0

```

Figure 46: Three Flip Flops Testbench TCL Console

## 6.2 D Flip Flop with Synchronous Reset and Clock Enable



```

4 // Engineer: Benyamin Yacoub, Eyiara Oladipo, Andre Price
5 //
6 // Create Date: 03/15/2024 08:20:30 PM
7 // Design Name: D_ff_with_ce_and_sync_reset_behavior
8 // Module Name: D_ff_with_ce_and_sync_reset_behavior
9 // Project Name: Simulation Assignment #6
10 // Target Devices:
11 // Tool Versions:
12 // Description:
13 //
14 // Dependencies:
15 //
16 // Revision:
17 // Revision 0.01 - File Created
18 // Additional Comments:
19 //
20 ///////////////////////////////////////////////////////////////////
21
22
23 module D_ff_with_ce_and_sync_reset_behavior(
24     input logic D, Clk, reset, ce,
25     output logic Q
26 );
27
28     always_ff @(posedge Clk)
29     if (reset) begin
30         Q <= 1'b0;
31     end else if (ce) begin
32         Q <= D;
33     end
34
35 endmodule

```

Figure 47: D Flip Flop with Clock-Enable and Sync Reset Behavior Code Implementation

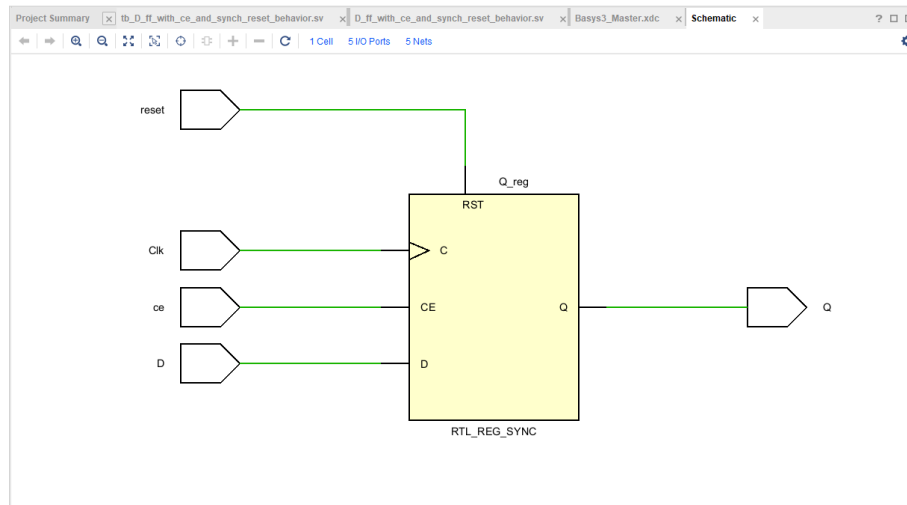


Figure 48: D Flip Flop with Clock-Enable and Sync Reset Behavior RTL Schematic

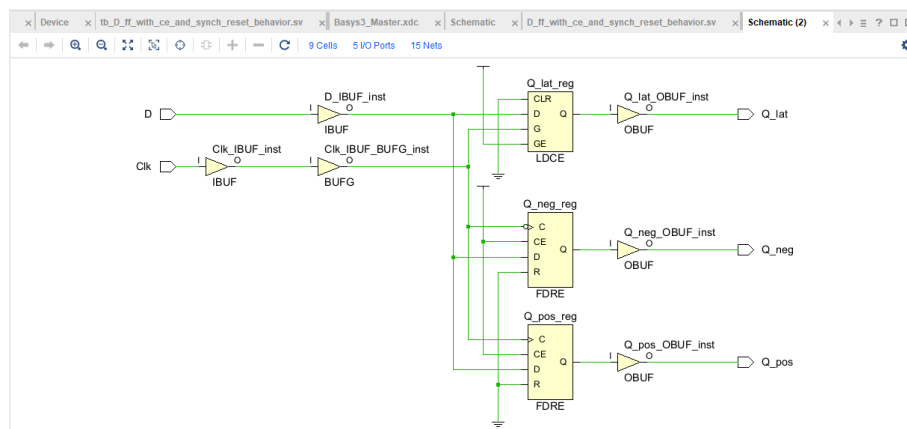


Figure 49: D Flip Flop with Clock-Enable and Sync Reset Behavior Synthesized Design

```

tb_D_ff_with_ce_and_sync_reset_behavior sv  x  D_ff_with_ce_and_sync_reset_behavior sv  x  Untitled 10  x
C:/Users/benj/sim_assg_7/sim_assg_7/srcs/sim_1/imports/Downloads/tb_D_ff_with_ce_and_sync_reset_behavior sv

20 ///////////////////////////////////////////////////////////////////
21
22
23 module tb_D_ff_with_ce_and_sync_reset_behavior();
24
25     logic D = 0, Clk = 0, reset = 0, ce = 0;
26     logic Q = 0;
27
28     D_ff_with_ce_and_sync_reset_behavior dffcsrb(
29         .D(D),
30         .Clk(Clk),
31         .reset(reset),
32         .ce(ce),
33         .Q(Q)
34     );
35
36     // Reverse the clock to be on or off
37     always #10 Clk = ~Clk;
38
39     initial begin
40         #20;
41         D = 1;
42
43         #40
44         ce = 1;
45         #20
46         ce = 0;
47         #60
48         D = 0;
49         #20
50         ce = 1;
51         #10;
52         D = 0;
53     end

```

Figure 50: D Flip Flop with Clock-Enable and Sync Reset Behavior Testbench Code

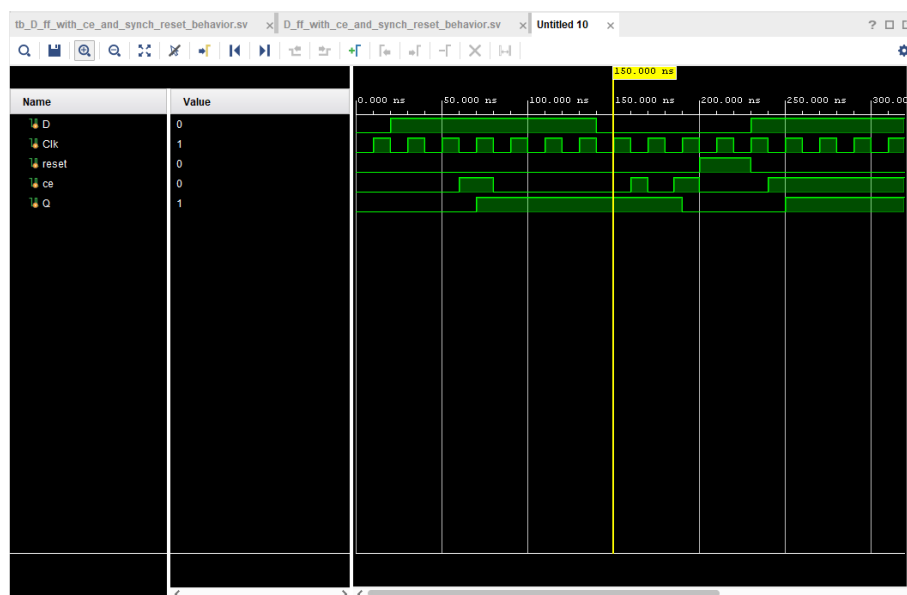


Figure 51: D Flip Flop with Clock-Enable and Sync Reset Behavior Testbench Behavioral Simulation



```

# run 1000ns
D = 1, Clk = 0, reset = 0, ce = 0, Q = 1
D = 1, Clk = 1, reset = 0, ce = 0, Q = 1
D = 1, Clk = 0, reset = 0, ce = 0, Q = 1
D = 1, Clk = 1, reset = 0, ce = 0, Q = 1
D = 1, Clk = 0, reset = 0, ce = 0, Q = 1
D = 1, Clk = 1, reset = 0, ce = 0, Q = 1
D = 1, Clk = 0, reset = 0, ce = 0, Q = 1
D = 1, Clk = 1, reset = 0, ce = 0, Q = 1
D = 1, Clk = 0, reset = 0, ce = 0, Q = 1
D = 1, Clk = 1, reset = 0, ce = 0, Q = 1

```

Figure 52: D Flip Flop with Clock-Enable and Sync Reset Behavior Testbench TCL Console

### 6.3 Results and Analysis Discussion

This section was the harder of the sections. The reading of the concepts was incredibly convoluted, trying to understand it a task of its own. This led to group members having to read this section over and over, seemingly resembling that of an infinite for loop. This obvious notion was more easily recognizable with the sections related to flip-flops rather than the Gated D latches, although that's not to say that the sections involving latches were any less difficult. The biggest problems with flip-flops were with the modular hierarchy. Despite conceptually making sense, the code required a lot of trial and error to fully understand; and even then, we still felt the need to go back and re-read, even after finishing the code, to try and understand all of the smaller nuances present within this topic.

## 7 Results and Conclusion

This section discusses the results of the Vivado and the MultiSIM simulation exercise. The discussion provides in-depth explanations of the results and any discrepancies between the results and theoretical expectations. It also explores potential sources of error and discusses the accuracy of the tests, the debugging process, and what was learned from these experiences.

Overall, the processes and challenges displayed in this simulation exercise completely juxtaposed that of the previous Simulation Exercise #5.5. As we highlighted in the previous multi-page simulation assignment report, the previous simulation exercise emerged as a convoluted document leading to many mental discrepancies and encompassing perplexing directions and intricate coding complexities. All in all, it was incredibly infuriating.

On the other hand, Simulation Exercise #6 was the complete and utter opposite,

from instructions to tone. The most complicated part of the exercise was the boredom displayed in comprehending the guidelines and understanding the concepts, something more noticeable in the D Latch and Flip-Flop sections.

The NOR gates, the SR Latch and the Gated SR Latch were all concepts which were relatively easy to understand both theoretically and practically. The NOR gates had the benefit of being familiar to all members of the group while the latches were explained in a cohesive way that made their introduction seem as a smooth new edition to the course, the lesson plan, and the simulation exercises overall.

The D Latch and Flip-Flop sections of the simulation provided more difficult than the former section, however. The way these areas were worded made understanding the material presented more complicated of a task. Since the instructions were convoluted, it became a task to try and grasp the concept in its entirety, especially when it came to modular hierarchy. We were able to complete these sections through trial and error and a lot of re-reading.

## **8 Group Contributions**

The work for this simulation assignment was divided among three individuals: Benyamain Yacoob, Andre Price Jr, and Eyiara Oladipo. Eyiara and Benyamain collaborated using Vivado software and writing in SystemVerilog to code the circuits. Benyamain worked on the flip-flops, and Eyiara worked on the SR Latches. It should be noted, however, that this was a group effort and all members were present to answer any questions related to the circuits. Andre led the effort in organizing the document, formatting it correctly, and writing the analysis for this report.