Sam Lee

4/26/18

CPE 64

Professor Kiran

Lab Report #4

Part 1:

For this part of the lab, I had to first create the truth table for a RS latch and use it as a reference for the other steps in Part 1.I had to create the test bench for the RS latch code provided to us by the lab instructions. The code compiled, and the waveform was generated after I had set up all the conditions for the waveform to display. In order to make sure that the code was displaying the correct output I had to create the circuit for a RS latch on Multisim. Luckily the schematic for a RS latch was given to us in part one therefore all I had to do was recreate it. Then I was given the truth table example of a NAND RS latch by the instructor in order to read my waveform properly. Finally, I was able to demo it to my instructor and get this part check mark and start on

Part 2.

|  |  |  |  |
| --- | --- | --- | --- |
| S | R | Q(present) | Q(new) |
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | Undefined |
| 1 | 1 | 1 | Undefined |

Truth Table:

A close up of a clock

Description generated with high confidence

Verilog Code:

module Part1(R,S,Q, NQ);

input R, S;

output Q, NQ;

assign Q = ~( S & NQ); // S (SET) is active low

assign NQ = ~(R & Q); // R (RESET) is active low

endmodule

Test Bench:

`timescale 1ns / 1ps

module Part1TB();

reg R, S;

wire Q, NQ;

Part1 uut(.R(R), .S(S), .Q(Q), .NQ(NQ));

initial begin

R = 1'b1;

S = 1'b0;

#20;

R = 1'b1;

S = 1'b1;

#20;

R = 1'b0;

S = 1'b1;

#20;

R = 1'b1;

S = 1'b0;

#20;

R = 1'b0;

S = 1'b1;

#20;

R = 1'b1;

S = 1'b1;

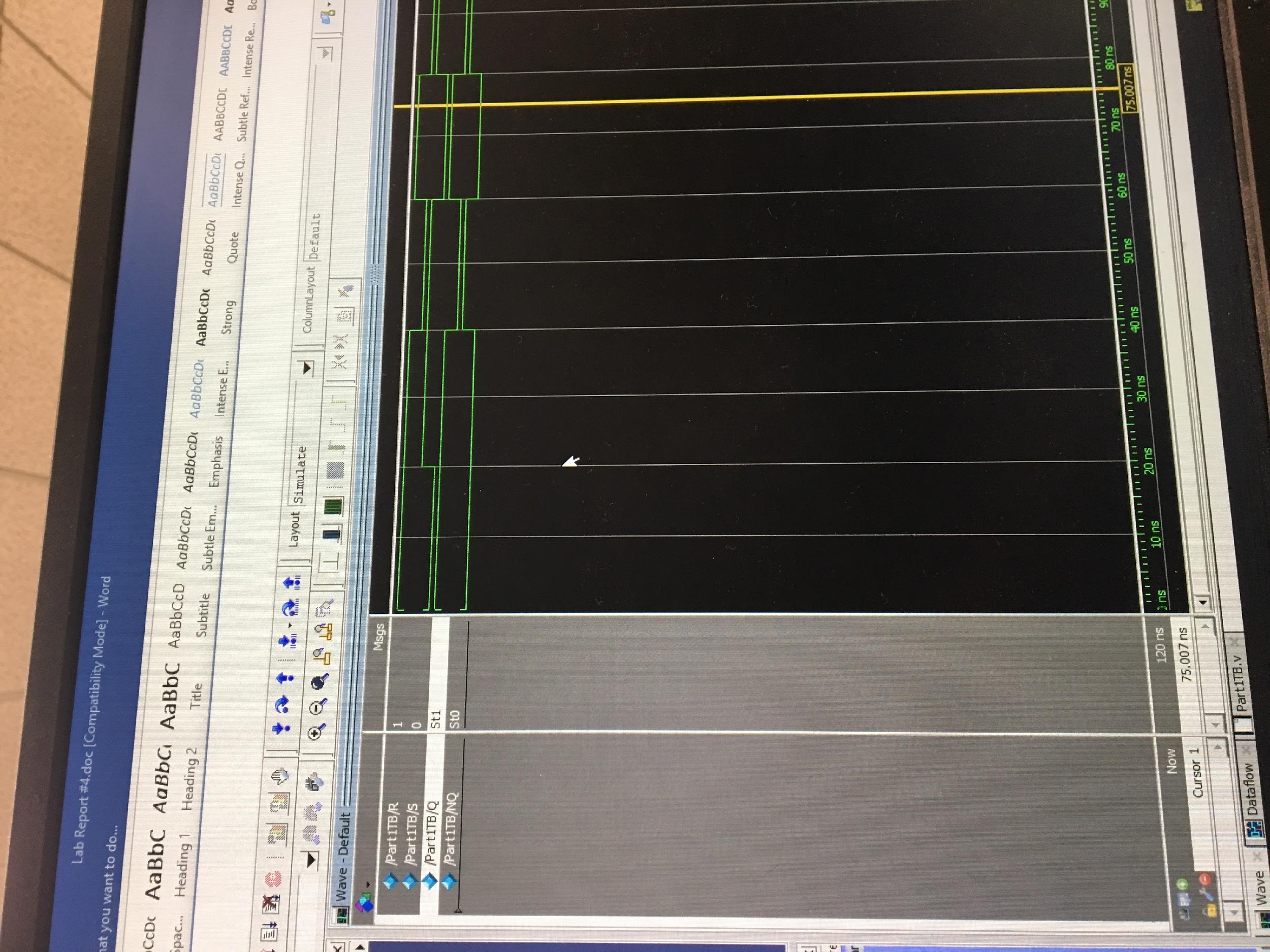
#20;

$stop;

end

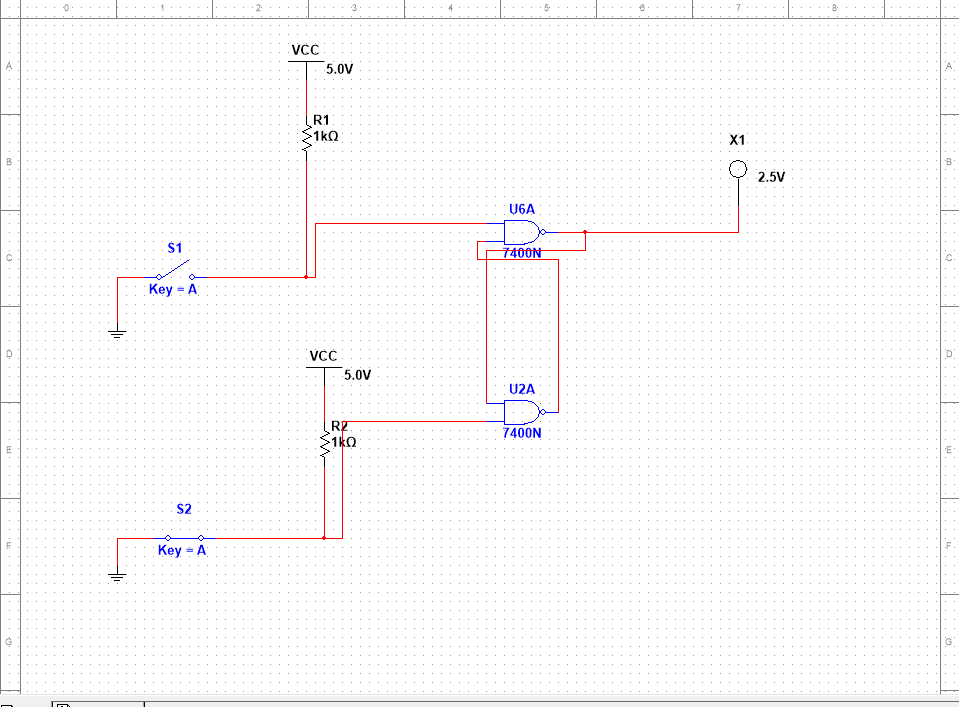
endmodule

WaveForm:



\*SR = 1,0 and Q/Qnew = 1,0

MultiSim:



\*SR = 1,0 and Q/Qnew = 0,1

Part 2:

For this part of the lab, it follows the same process as part one with the only difference being that were making a NOR RS latch. The NOR RS latch follow the standard truth table for a RS latch therefore making it more understandable to read. I had to create the Verilog code and test bench for the NOR SR latch. Then I created the Schematic on Multisim in order to verify my test bench results after they have generated the waveform. I then created my waveforms and made sure I was getting the correct output. Finally, I demoed it to my instructor and proceeded to Part 3.

Verilog Code:

module Part2(R,S,Q,NQ);

input R, S;

output Q, NQ;

assign Q = ~(R | NQ); // R (RESET) is active high

assign NQ = ~(S | Q); // S (SET) is active high

endmodule

Test Bench:

`timescale 1ns / 1ps

module Part2TB();

reg R, S;

wire Q, NQ;

Part2 uut(.R(R), .S(S), .Q(Q), .NQ(NQ));

initial begin

R = 1'b1;

S = 1'b0;

#20;

R = 1'b0;

S = 1'b0;

#20;

R = 1'b0;

S = 1'b1;

#20;

R = 1'b1;

S = 1'b0;

#20;

R = 1'b0;

S = 1'b1;

#20;

R = 1'b0;

S = 1'b0;

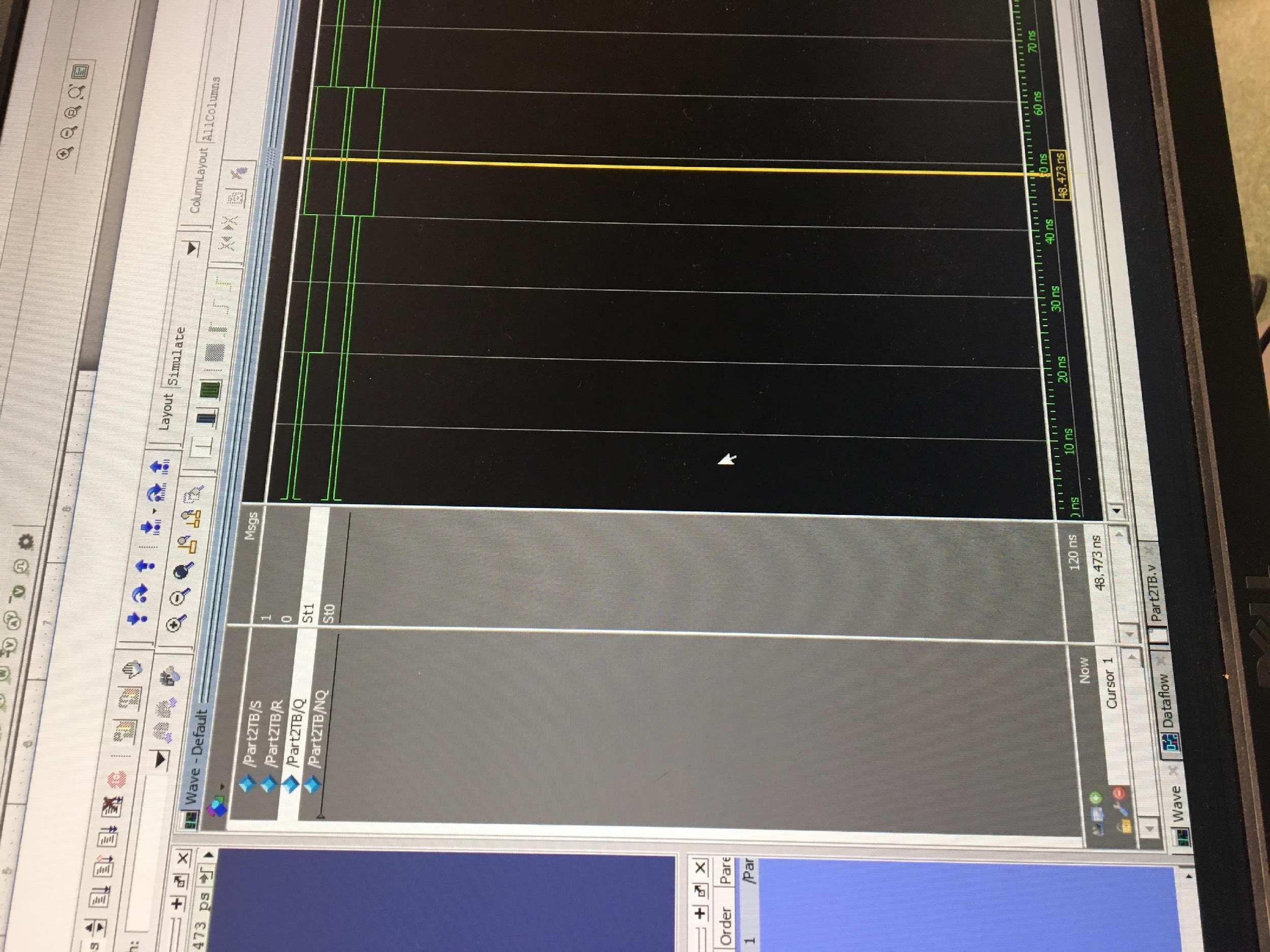
#20;

$stop;

end

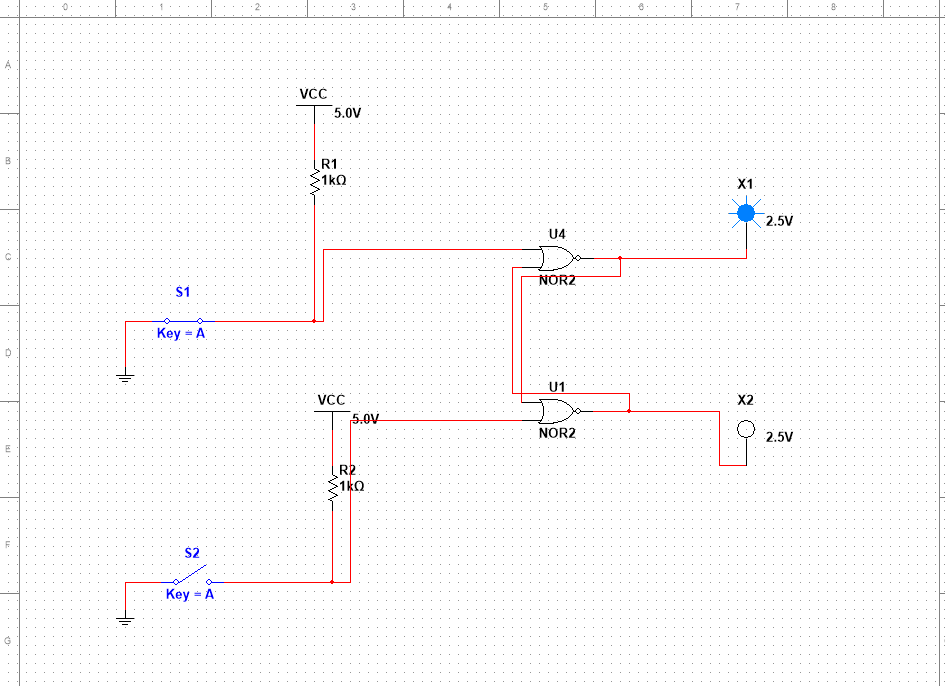
endmodule

WaveForm:



\*S/R = 1,0 and Q/Qnew = 1,0

MultiSim:



\*S/R = 1,0 and Q/Qnew = 1,0

Part 3:

For Part 3 of this lab, I had to create a NOR version of a D flip flop so I first had to create the circuit on Multisim. The circuit only had two switches for clock and D which dictate the result of Q and QN since S and R are part of the circuit itself. After I had created my schematic I created the Verilog for the NOR D Latch. After I had created the Verilog I created the test bench for the code and observed the results generated by the waveform. Then I made sure I was getting the correct output from my waveform by comparing to the results from my circuit schematic. Finally, I demoed it to my instructor and proceeded to my next part.

Verilog Code:

module Part3(CLK,D,R,S,Q,QN);

input CLK,D;

output R, S, Q, QN;

wire CLK,D;

assign Q = ~ ( R | QN );

assign QN = ~ ( S | Q );

assign S = ~(CLK | ~(D | S) | R);

assign R = ~(CLK | ~(R | ~(D | S)));

endmodule

Test Bench:

`timescale 1ns / 1ps

module Part3TB();

reg CLK, D;

wire R, S, Q, QN;

Part3 uut(.CLK(CLK), .D(D), .R(R), .S(S), .Q(Q), .QN(QN));

initial begin

CLK = 1'b1;

D = 1'b0;

#5;

CLK = 1'b0;

//D = 1'b1;

#5;

CLK = 1'b0;

D = 1'b1;

#5;

CLK = 1'b1;

D = 1'b1;

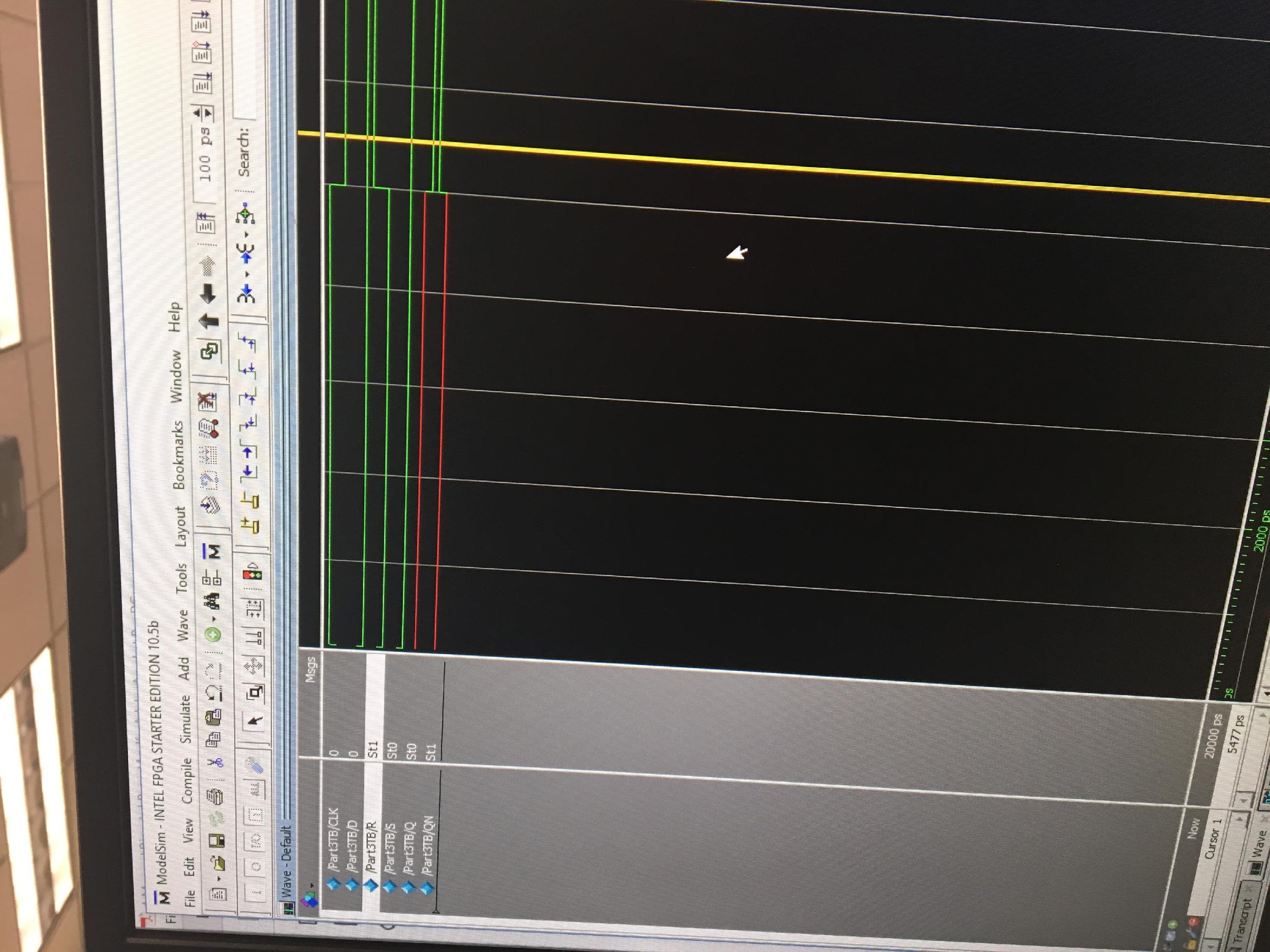
#5;

$stop;

end

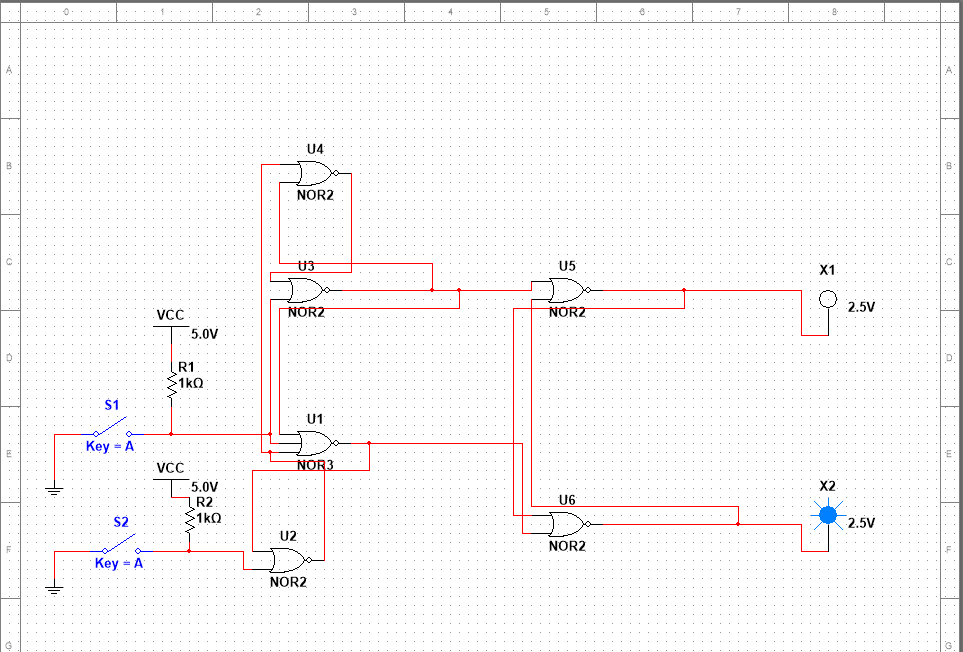
endmodule

WaveForm:



\*Input 0,0 and Output 0,1

MultiSim:



\*Input 0,0 and Output 0,1

Part 4:

The NAND gate version of a D flip flop is different than the NOR gate version because it introduces two new input which are Preset and Clear. For part 4 I created the Verilog by basing my code on the circuit diagram on the instructions and later creating the test bench. After I had made sure that my code was compiling and generating a waveform I was ready to demo it to my instructor. I demoed it to my instructor and proceed to the next part.

Verilog Code:

`timescale 1ns / 1ps

module Part4TB();

reg PR,CR,CLK,D;

wire Q,QN;

Part4 uut(.PR(PR), .CR(CR), .CLK(CLK), .D(D), .Q(Q), .QN(QN));

initial begin

PR = 1'b0;

CR= 1'b0;

CLK = 1'b0;

D = 1'b0;

#10;

PR = 1'b0;

CR= 1'b0;

CLK = 1'b0;

D = 1'b1;

#10;

PR = 1'b0;

CR= 1'b0;

CLK = 1'b0;

D = 1'b1;

#10;

PR = 1'b1;

CR= 1'b0;

CLK = 1'b0;

D = 1'b1;

#10;

PR = 1'b0;

CR= 1'b1;

CLK = 1'b0;

D = 1'b1;

#10;

PR = 1'b1;

CR= 1'b1;

CLK = 1'b1;

D = 1'b1;

#10;

$stop;

end

endmodule

Test Bench:

`timescale 1ns / 1ps

module Part4TB();

reg PR,CR,CLK,D;

wire Q,QN;

Part4 uut(.PR(PR), .CR(CR), .CLK(CLK), .D(D), .Q(Q), .QN(QN));

initial begin

PR = 1'b0;

CR= 1'b0;

CLK = 1'b0;

D = 1'b0;

#10;

PR = 1'b0;

CR= 1'b0;

CLK = 1'b0;

D = 1'b1;

#10;

PR = 1'b0;

CR= 1'b0;

CLK = 1'b0;

D = 1'b1;

#10;

PR = 1'b1;

CR= 1'b0;

CLK = 1'b0;

D = 1'b1;

#10;

PR = 1'b0;

CR= 1'b1;

CLK = 1'b0;

D = 1'b1;

#10;

PR = 1'b1;

CR= 1'b1;

CLK = 1'b1;

D = 1'b1;

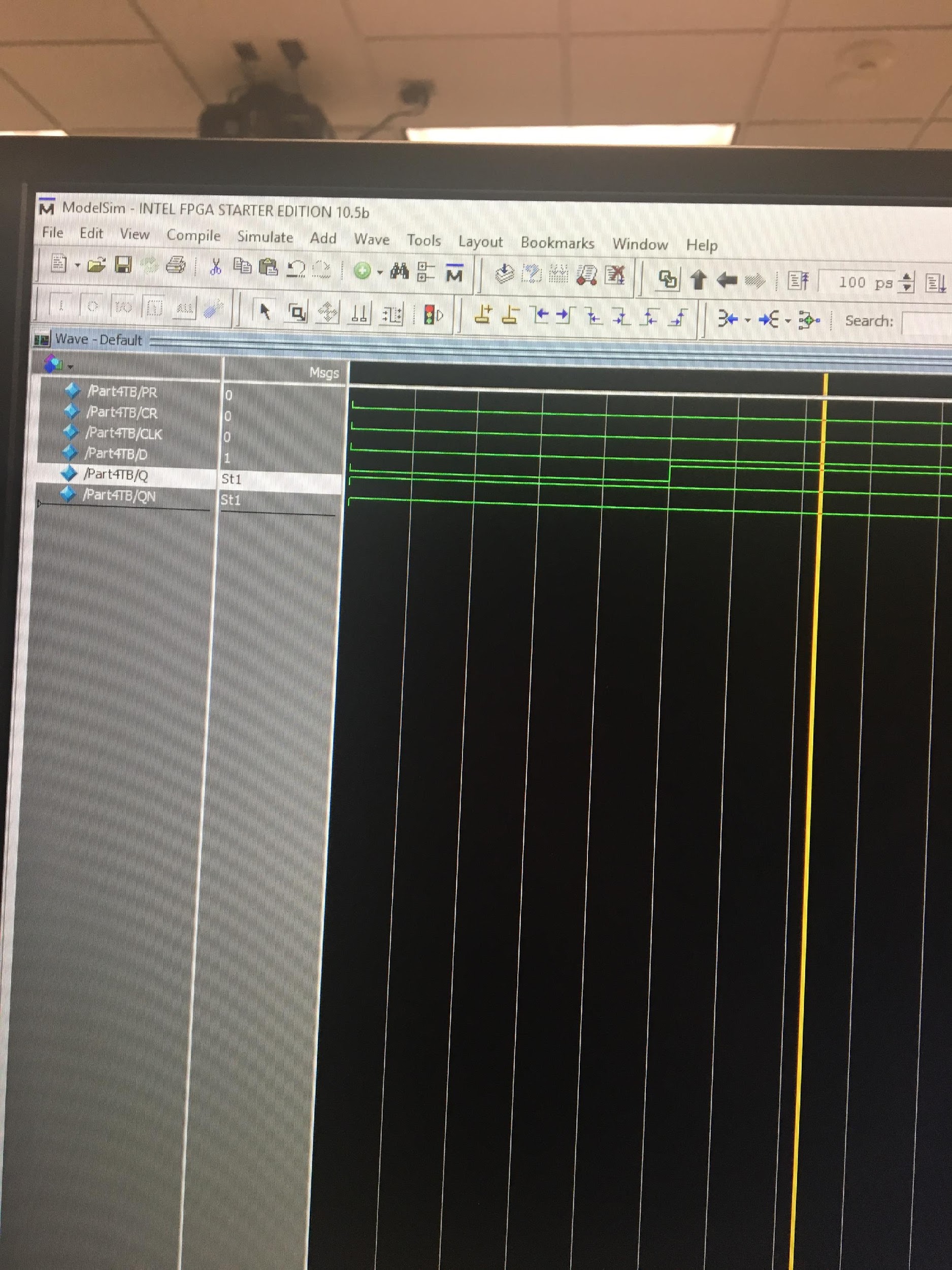
#10;

$stop;

end

endmodule

WaveForm:



Input 0,0,0,1 and Output 1,1

Part 5:

For this part of the lab, it required me to combine four d flip flops into one circuit in Verilog and generate the waveform from the test bench. When creating the Verilog, I used the schematic provided as reference when creating the code for it. Then I had to make sure that my test bench was working properly by compiling it then generating the waveform. The code was very time specific because we were going to use a bunch of clocks in combination.

Verilog Code:

module Part5(CLK, q0, q1, q2, q3);

input CLK;

wire d0, d1, d2, d3;

output reg q0, q1, q2, q3;

assign d0 = ~q0;

assign d1 = (~q1 & q0) | (~q0 & q1);

assign d2 = (q1 & q0 & ~q2) | (~q1 & q2) | (~q0 & q2);

assign d3 = (~q3 & q2 & q0 & q1) | (~q2 & q3) | (~q0 & q3) | (~q1 & q3);

initial

begin

q0 <= 0;

q1 <= 0;

q2 <= 0;

q3 <= 0;

end

always @(posedge CLK)

begin

q0 <= d0;

q1 <= d1;

q2 <= d2;

q3 <= d3;

end

endmodule

Test Bench:

`timescale 1ns / 1ps

module Part5TB();

reg CLK;

wire q0, q1, q2, q3;

Part5 uut(.CLK(CLK), .q0(q0), .q1(q1), .q2(q2), .q3(q3));

initial begin

CLK = 1'b0;

forever #5 CLK = ~CLK;

end

initial begin

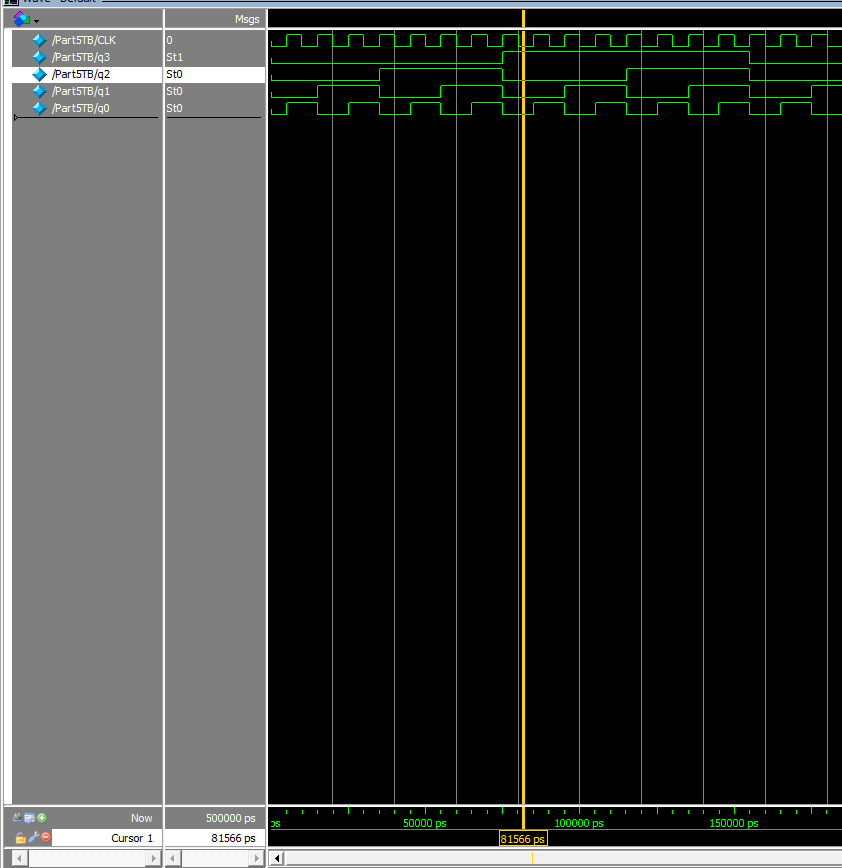
#500;

$stop;

end

endmodule

WaveForm:



\*Input 0 and Output 1,0,0,0

Part 6:

In this part of the lab, I had to design a three stage shift register which delay the signal of the waveform for “n” number of shift register stages. I used the schematic given to me on the instructions to create the Verilog code then I created the test bench by reference of my code from previous parts of the lab to help guide in making the waveform. The waveforms seems to be delayed by the three shift register stages that I made in the code.

Verilog Code:

module Part6(clock,i,reset,q0,q1,q2,q3,q4,q5,q6,q7);

wire d0,d1,d2,d3,d4,d5,d6,d7;

input clock,i,reset;

output reg q0,q1,q2,q3,q4,q5,q6,q7;

assign d0=i;

assign d1=q0;

assign d2=q1;

assign d3=q2;

assign d4=q3;

assign d5=q4;

assign d6=q5;

assign d7=q6;

always@(posedge clock or negedge reset)

begin

if(reset == 0)

begin

q0 <= 0;

q1 <= 0;

q2 <= 0;

q3 <= 0;

q4 <= 0;

q5 <= 0;

q6 <= 0;

q7 <= 0;

end

else

begin

q0 <= d0;

q1 <= d1;

q2 <= d2;

q3 <= d3;

q4 <= d4;

q5 <= d5;

q6 <= d6;

q7 <= d7;

end

end

endmodule

Test Bench:

`timescale 1ns / 1ps

module Part6TB();

reg clock, i, reset;

wire q0,q1,q2,q3,q4,q5,q6,q7;

Part6 uut(.clock(clock), .i(i), .reset(reset), .q0(q0), .q1(q1), .q2(q2), .q3(q3), .q4(q4), .q5(q5), .q6(q6), .q7(q7));

initial begin

clock = 1'b0;

forever #5 clock = ~clock;

end

initial begin

reset = 1'b0;

#20;

i = 1'b1;

#20;

reset = 1'b1;

#20;

end

initial begin

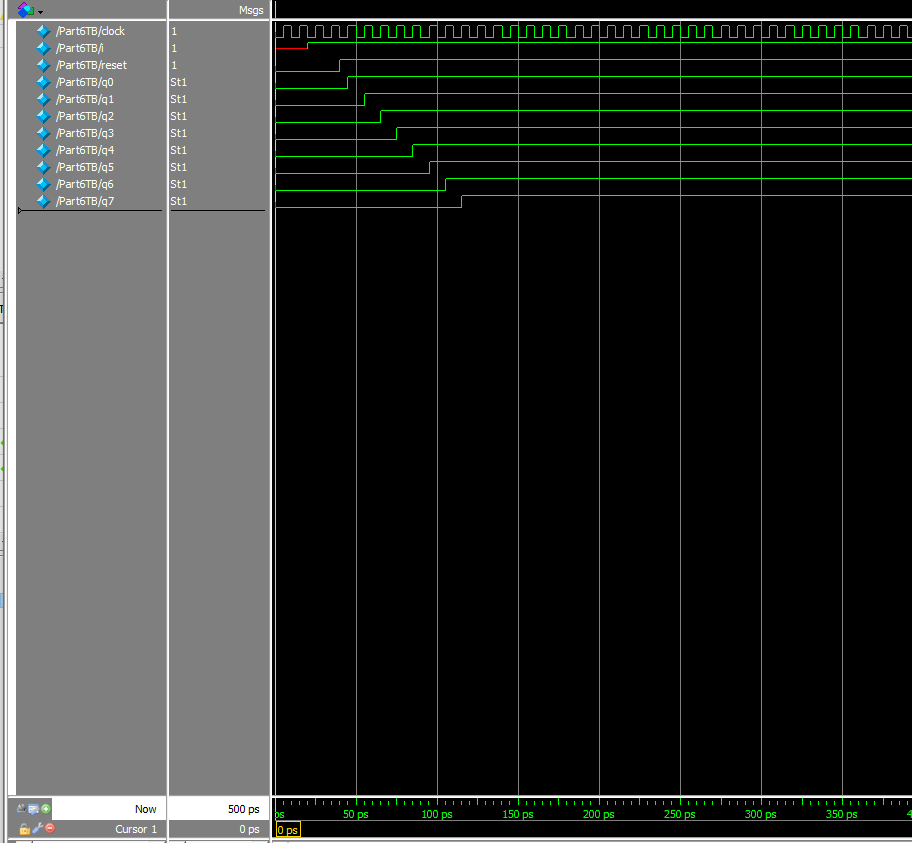
#500;

$stop;

end

endmodule

WaveForm:



Part 7:

In this last part of the lab, I had to design an eight bit shift register that can be synchronously pre-loaded via a control line called “PL” (Parallel Load). I needed to write equations for each of the D F/F inputs as a function of the previous flip-flop output, the “PL” signal and the parallel load data signals (call them P7, P6, P5, P4, P3, P2, P1, P0). The D F/F signals (D7 – D0) should be displayed on the LEDs next to the Q outputs (Q7 – Q0). For this part I needed to use eight assign statements for D7 – D0. I recorded the results and show it to my lab instructor.

Verilog Code:

module Part7(clk, si, pl, sl, sr, rr, rl, Q);

input clk, si, pl, sl, sr, rr, rl;

output reg [7:0]Q = 8'b10101010;

wire [7:0]P;

assign P = 8'b10011001;

always @(posedge clk)

case ({pl, sl, sr, rr, rl})

5'b10000: begin Q<=P; end

5'b01000: begin Q<=Q<<1; Q[0]<=si; end

5'b00100: begin Q<=Q>>1; Q[7]<=si; end

5'b00010: begin Q<=Q>>1; Q[7]<=Q[0]; end

5'b00001: begin Q<=Q<<1; Q[0]<=Q[7]; end

endcase

endmoduless

Test Bench:

`timescale 1ns / 1ps

module Part7TB();

reg clk=1'b0, si=1'b0, pl=1'b1, sl=1'b0, sr=1'b0, rr=1'b0, rl=1'b0;

wire [7:0] Q;

integer i,j,k,l,m,n,o;

Part7 uut(.clk(clk), .si(si), .pl(pl), .sl(sl), .sr(sr), .rr(rr), .rl(rl), .Q(Q));

initial begin

for(i=0;i<160;i=i+1) begin

#20

clk<=~clk;

end

$stop;

end

initial begin

for(j=0;i<107;j=j+1) begin

#30;

si<=~si;

end

$stop;

end

initial begin

for(k=0;k<80;k=k+1) begin

#40;

pl<=~pl;

end

$stop;

end

initial begin

for(l=0;l<64;l=l+1) begin

#50;

sl<=~sl;

end

$stop;

end

initial begin

for(m=0;m<54;m=m+1) begin

#60;

sr<=~sr;

end

$stop;

end

initial begin

for(n=0;n<46;n=n+1) begin

#70;

rr<=~rr;

end

$stop;

end

initial begin

for(o=0;o<40;o=o+1) begin

#80;

rl<=~rl;

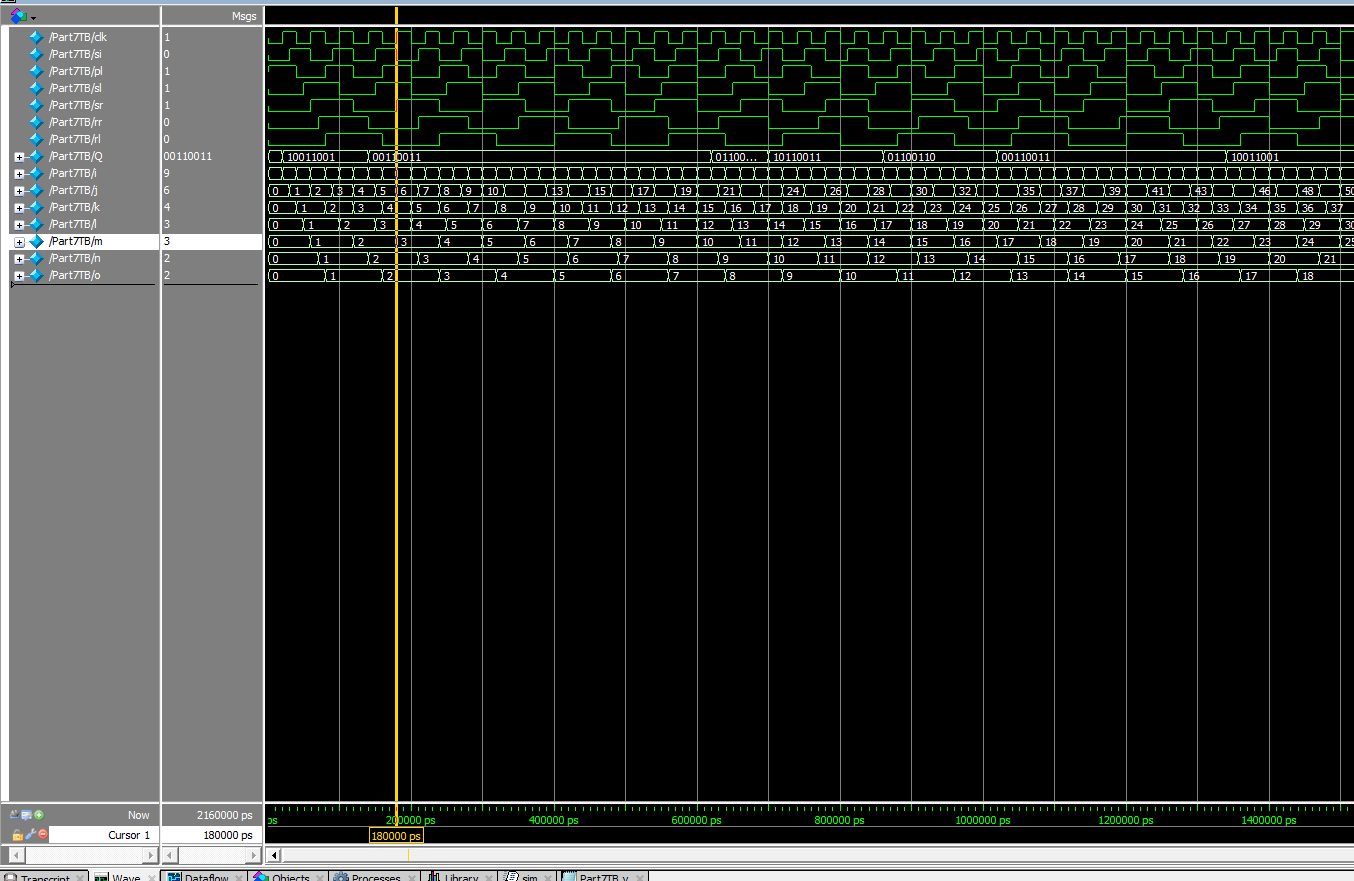
end

$stop;

end

endmodule

WaveForm:



Conclusion:

This lab was a thorough introduction to RS latches, flip-flops, shift registers, and positive/negative clock triggers. There were a number of important concepts covered in lecture that were explored during the assignments, such as state transition tables and sequential logic. Other important lab assignments included gaining familiarity with waveform simulations of Verilog code and understanding the differences between several different types of inputs for flip-flops and latches. My experience with this lab was generally positive but somewhat frustrating at times. There were a few setbacks during thelab due to my unfamiliarity with the concepts that were used in this assignment and my misinterpretation of certain instructions. There was quite a significant amount of different concepts covered in this lab which resulted in me needing lot of time to complete the individual lab assignments. However, I was eventually able to complete the assignments and clear up my misunderstanding of the instructions with the help of the instructor and other classmates.

Question 1: What is the purpose of the two resistors in the schematic below?

The purpose of the two resistors is to keep the input active high until the switch connect and it turns to active low.  Both of the resistors are connected to +5V which tie to the line to keep it high. It prevent the line to have the input to be something that we do not want or somewhere between 0-5 volts.

Question 2: What is the difference between NOR gate version and NAND gate version when using the S and R inputs?

The difference between the two kind is that for the NOR gate, the input is active high and for the NAND gate, the input is active low.

Question 3: Which edge of the clock can cause the Q output of the D Flip-Flop (NOR version) to change, the positive / rising edge of the clock (from 0 to 1 transition) or the negative / falling edge transition) of the clock?

For this particular D flip flop, the Q output changed on the falling edge of the clock (going from 1 to 0).  From the provided wave form, the Q and QN changed every time the clock changed from 1 to 0.

Question 4: Which edge of the clock can change the Q output of our D Flip-Flop (NAND version), the positive / rising edge (from 0 to 1 transition), or the negative / falling edge transition)?

The Q output of the NAND gate D flip flop changed in the positive edge of the clock (going from 0 to 1).

Question 5: What are the logic levels of Q and QN when PR and CR are BOTH active at the same time? (NAND version)

When both CR and PR are active high at the same time, the output Q and QN will be the same as the previous state because of the way the latches are designed.

Question 6:What is the difference between the NAND version and NOR version of the D Flip- Flop when using the Preset (PR) and Clear (CR)?

For NAND gate version, when CR = 1 and PR = 0, the output will be Q =1 and QN = 0.  On the contradictory, for NOR gate version, when CR = 1 and PR =0, the output will be Q= 0 and QN=1.  The reverse will happen for both gate when CR=0 and PR=1. For NAND gate, it will be Q=0 and QN=1 and NOR gate will be Q=1 and QN=0.