Lab 4: LatchesFF- ShiftRegisters-Counters

CpE Section 1

Instructor: Telles, Eric

Friday: 4:30 pm – 6:50pm

Thao, Shammah

Part 1: Nand gate version of the RS latch

Description:

We will have to write a data flow Verilog, which is basically assigning the formula and then make a waveform. To do so you need multisim and a truthtable.

Procedure:

Write a Data Flow model description in Verilog HDL for a NAND gate version of the RS latch as shown in Figure 4-1. Follow the Introduction to Simulation of Verilog Designs Using ModelSim Graphical Waveform Editor document on moodle, as well as the How to create and run testbenches in Quartus II version 13.0 video, also found on moodle.

Engineering Data:

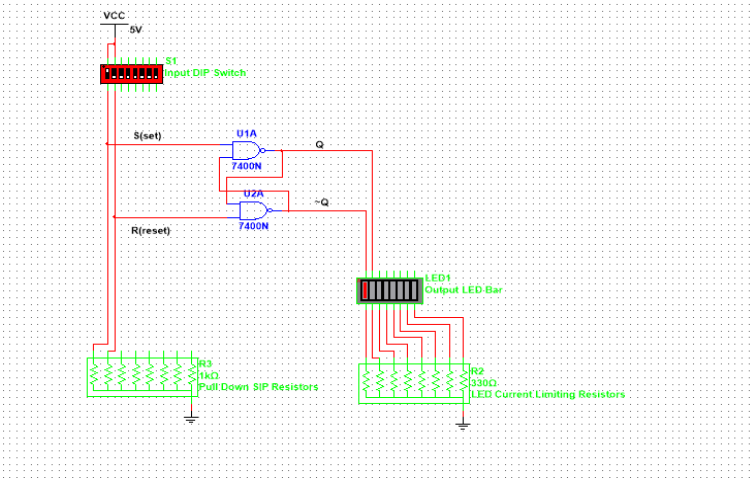

Figure 1

Figure 1 is the multisim of and nand rs latch. As should in figure 2, the truth table, the rs latch will be invalid when both output is 0. Rs latch will memorize one of the output and re output it when input is 1.

D10

	A	B	C	D	E
1	S	R	Qold	Qnew	
2	0	0	0	invalid	
3	0	0	1	invalid	
4	0	1	0	1	
5	0	1	1	1	
6	1	0	0	0	
7	1	0	1	0	
8	1	1	0	0	
9	1	1	1	1	

Figure 2: Truth table

Figure 3 is the Verilog code using data flow, as we assign a output with an equation we will get a wave form with the correct testbench that was given to us.

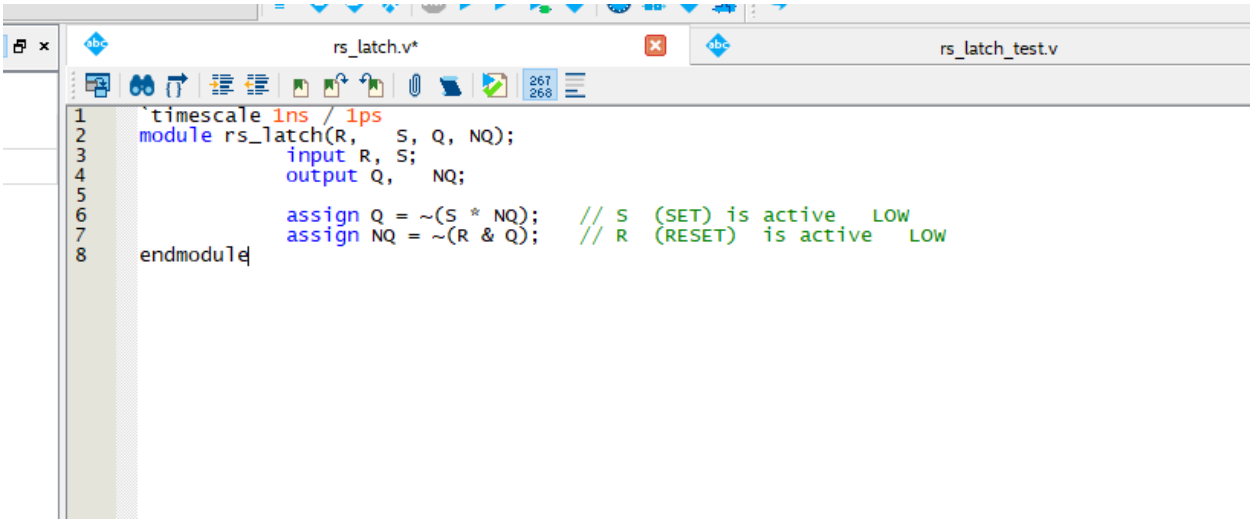

Figure 3: Verilog

For figure 4, it shows the output in waveform mode. It shows what was shown in the truth table in figure 2, invalid at 0 and as it goes on everything else is valid.

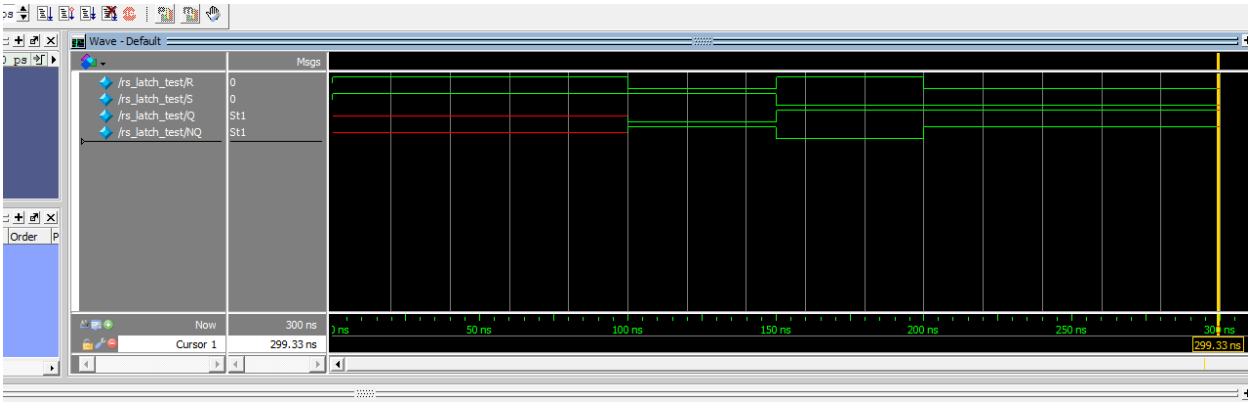

Figure 4:Waveform

Part 2: Nor gate version of RS latch

Description:

For this part of the lab, we had to also write another data flow Verilog along with the truth table. Which includes a truth table and a multisim

Procedure:

Write a Data Flow model description in Verilog HDL for a NOR gate version of the RS latch. Again, you should write the Verilog before lab class using a TEXT EDITOR

Engineering Data:

L1

A	B	C	D	E
S	R	Qold	Qnew	
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	invalid	
1	1	1	invalid	

Figure 5

In figure 5, it shows the truth table of the nor rs latch, using figure 6 to show the output, we know that when both input is 1 it is invalid. But everything else is okay since it is a low output

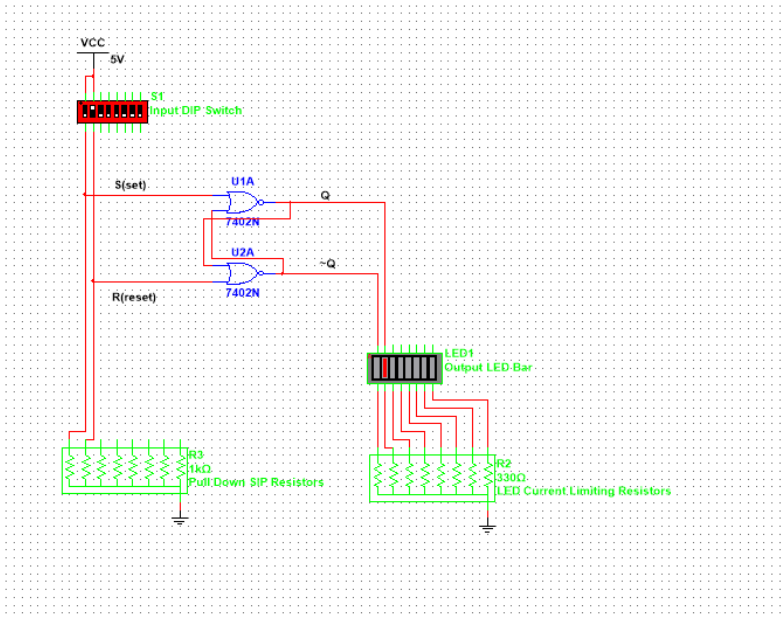

Figure 6

Figure 7, is basically the same as figure 3, the only changes is that we are nor – ing the equation s we change the sign.

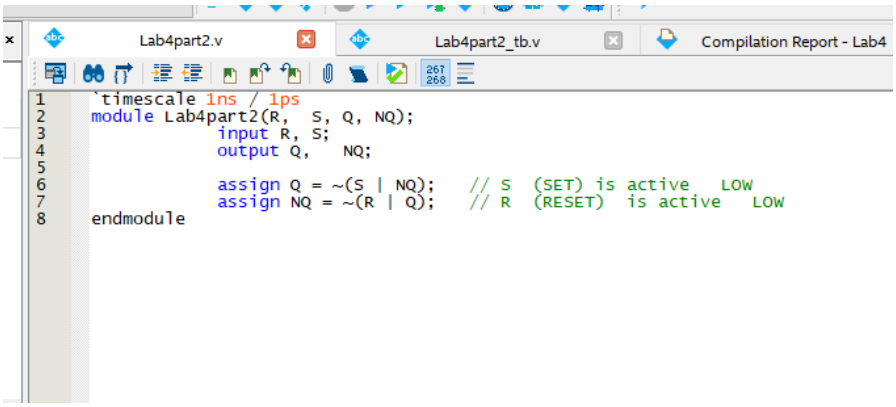

Figure 7

Figure 8, is the wave form that I got, the only issue is that I got the q and nq backward, so it turn out all green. But it is suppose to be red toward the end showing that 1 is invalid.

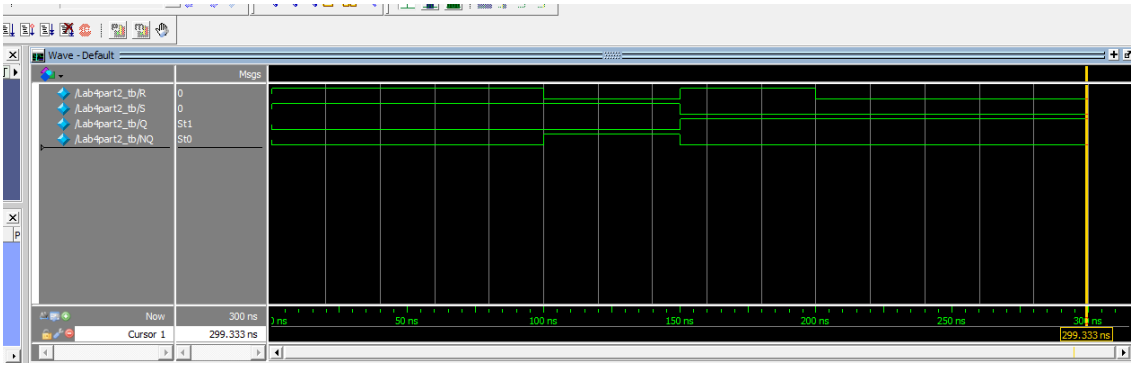

Figure 8

Part 3: Nor gate version of D Flip-Flops

Description:

We are also using another data flow Verilog design , that will be assigning 4 different equation to test D flip flop that is connected to a RS latch.

Procedure:

Write a Data Flow model description in Verilog HDL for a NOR gate version of the D Flip-Flop. The logic diagram for the NOR gate version is on Figure 4-2. Draw a diagram for your report with the signal names you used in your Hardware Description. Again, you should write the Verilog before class using a TEXT EDITOR. Bring the file to lab class and compile and download your Verilog solution to the CPLDFPGA board.

Engineering Data:

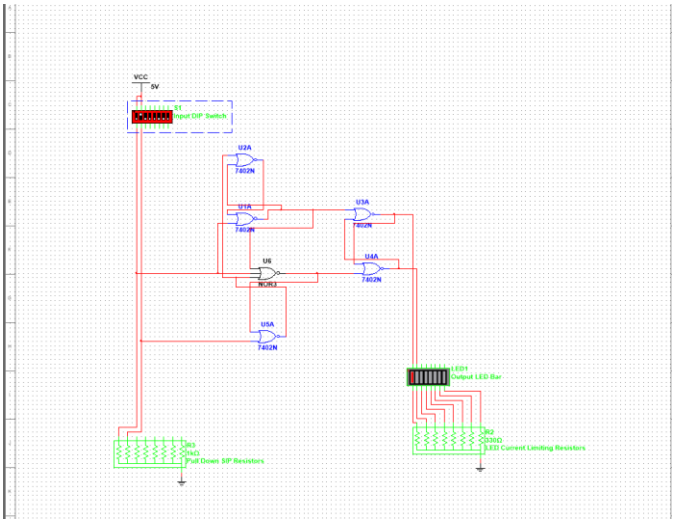

Figure 9

Figure 9 is the multism that was created. Using the multisim, we create 2 additional equation with the first 2 given to us already. We use the 4 equation and plug them into figure 10, where the Verilog is at.

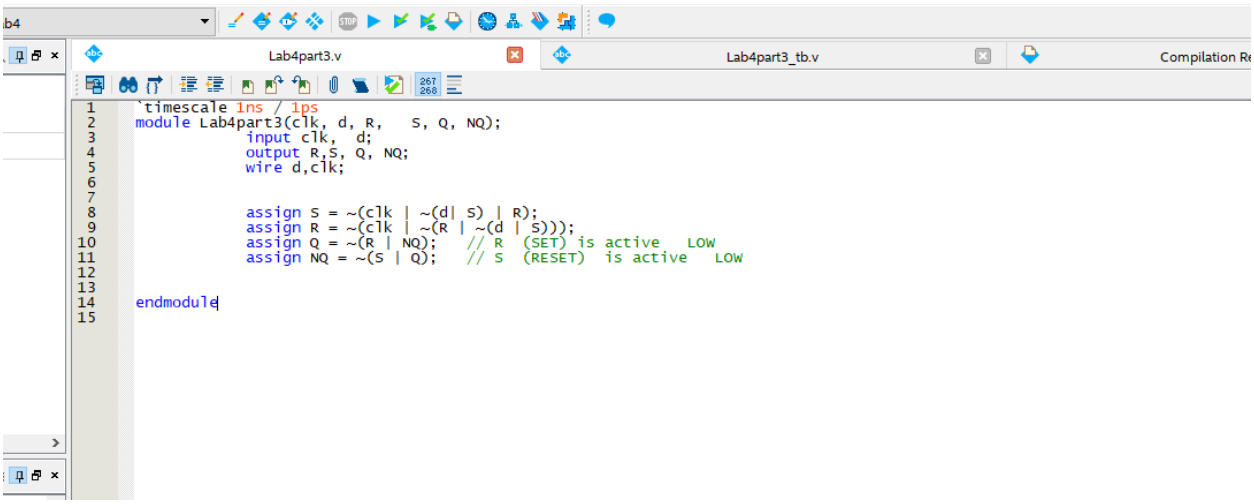

Figure 10

Figure 11, will show the waveform where it tell us that St1 and st0 is invalid since it had no memory to begin with so they cant save anything so its invalid, until later on.

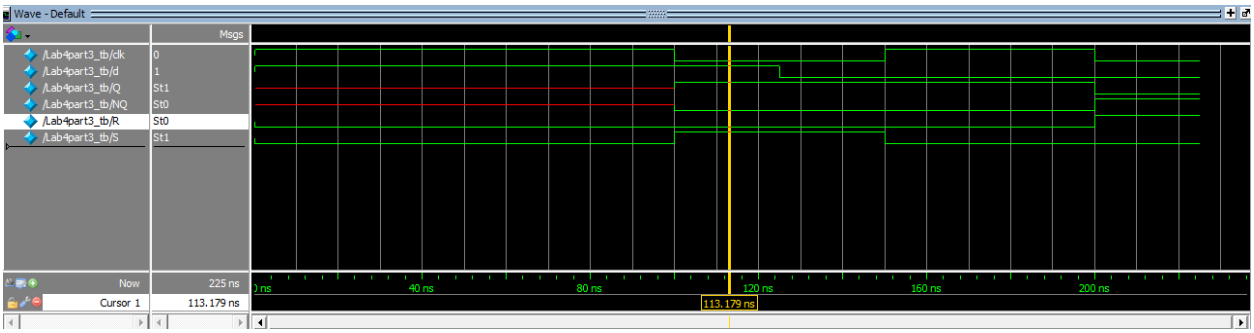

Figure 11

Part 4: Nand gate version of D flip-flop with Preset and Clear

Description:

Using data flow Verilog coding, we are to create a nand gate that works along a D flip-flop.

Procedure:

Write a Data Flow model description for a NAND gate version of the D Flip-Flop with Preset (PR) and Clear (CR). The logic diagram for the NAND gate version is on Figure 4-4. Draw a Block diagram for your report with the signal names you used in your Hardware Description.

Engineering Data:

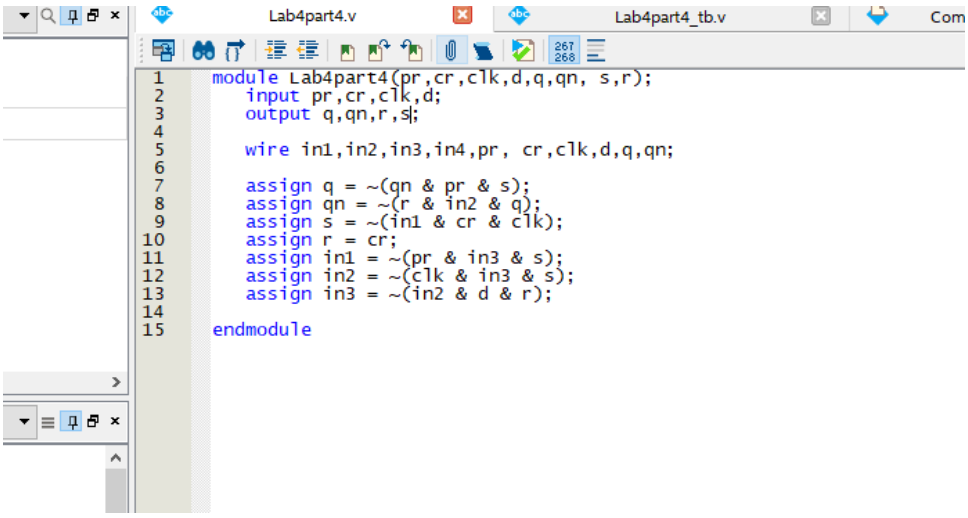

Figure 12

Figure 12 is the equation that was Verilog that shows the 7 equation. Each equation will be working together to be combine into each other using nands. It will be used to make figure 13, waveform.

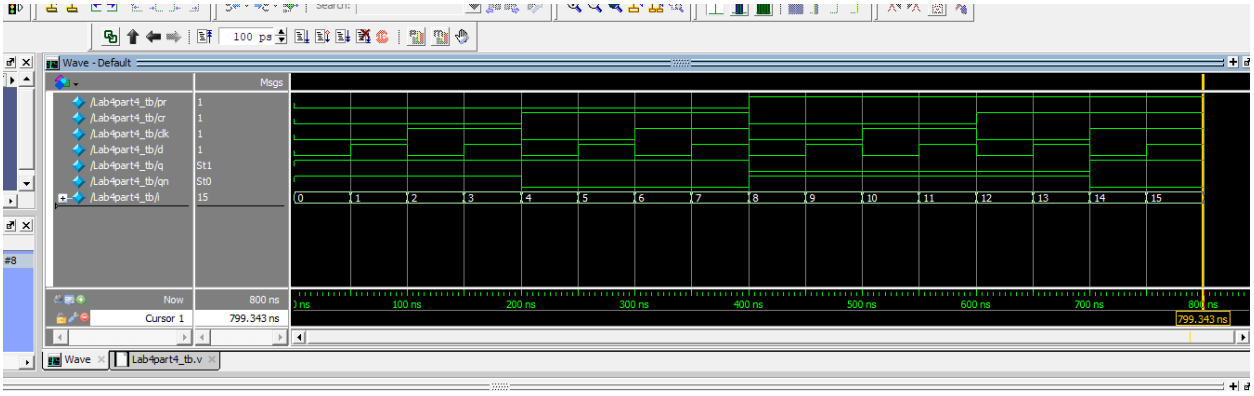

Figure 13

Part 5: Circuit using four D-type flip-flops

Description:

We are to make a Verilog equation that involves 4 different flip flops. Following a Circuit that is given to us.

Procedure:

The diagram in Figure 4-5 shows a circuit using four D-type flip-flops. Write the Verilog equation for each of the four "D" inputs. For example, in FF_0 the equation for it's D input is D0 = ~Q0; In a similar manner, in FF_1 the equation for it's D input is: D1 = (~Q1 & Q0) | (~Q0 & Q1); Compile and then simulate. Notice that the only input is the "clock" for the flip-flops. Describe in your report the behavior of this design. Use the simulator and create a timing waveform with at least 20 clock pulses; observe the flip-flop outputs in the bit order Q3, Q2, Q1, Q0.

Engineering Data:

Lab4part5.v | Lab4part5_tb.v* | Lab4part6.v | Lab4part6_tb.v | Lab4part7.v | Lab4part7_tb.v | Compilation Repo

```
1   module Lab4part5 (reset,CLK,Q);
2
3      input wire CLK,reset;
4      output reg[3:0] Q;
5      wire [3:0] D;
6
7      assign D[0] = ~Q[0]; //FF_0 flip-flop
8      assign D[1] = (Q[1] & ~Q[0])|(Q[0] & ~Q[1]); //FF_1 flip-flop
9      assign D[2] = (~Q[2] & Q[1] & Q[0])|(Q[2] & ~Q[1])|(Q[2] & ~Q[0]); //FF_2 flip-flop
10     assign D[3] = (~Q[3] & Q[2] & Q[0] & Q[1])|(~Q[2] & Q[3])|(~Q[0] & Q[3])|(~Q[1] & Q[3]); //FF_3 flipflop
11     always @ (posedge CLK)Q <= D; //shows the D values
12
13     endmodule
14
```

Figure 14

Figure 14, is equation that was acquire reading a circuit that was given to us about the 4 D flip flop. Each equation Is assigned to a different box, since there are 4 boxes, each had its own gates before a number can go through them, making figure 15, the waveform. As a number goes through the box it get more complicated.

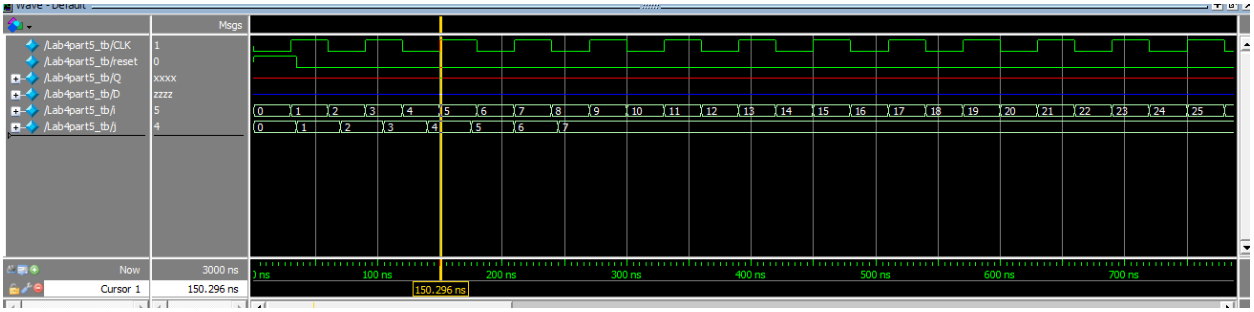

Figure 15

Part 6: Three-stage shift register

Description:

Using a three stage shift, we got to make an 8 stage shifting register

Procedure:

The logic diagram in Figure 4-6 illustrates a three-stage shift register. Using this idea, expand the circuit into a eight stage shift register. Design Verilog code to model your design. Assign outputs to LEDs, compile, simulate, and download and test. Record your results for you lab report with your simulation waveform.

Engineering Data:

Lab4part5_tb.v | Lab4part6.v | debounce.v | Lab4part6_tb.v | Lab4part7.v | Lab4part7_tb.v

```
1  module Lab4part6 (pin12out, CLK,SDin,SDout);
2     input CLK, SDin,pin12out;
3     output reg [7:0] SDout;
4
5     debounce g0(pin12out,CLK, clean);
6
7     always @ (posedge clean) //begins on the positive edge of CLK
8     begin
9     SDout <= SDout << 1; //cycles through all 8 registers
10    SDout[0] <= SDin; //assigns the first register
11
12 end
13 endmodule
```

Figure 16

Figure 16, uses figure 17, as they bounces from each other to figure out a 8 stage shifting register. It goes through a for look checking if something will happen, if something happen the number changes.

```
1   module debounce(clk, dirty, clean);
2      input clk; // onboard 64 MHz clock *pin 12
3      input dirty; // "dirty" is the glitched, asynchronous, active low push-button signal
4      output reg clean; // 1 while the push-button is active (down)
5      reg [15:0] cnt;
6      wire mcount = &cnt; // true when all bits of cnt are 1's
7
8      always @(posedge clk)
9      if(clean != dirty) cnt <= 0; // nothing's going on
10     else
11     begin
12        cnt <= cnt + 1; // something happened, start counting
13        if(mcount) clean <= ~clean; // if the counter is maxed, dirty changed!
14  end
15  endmodule
```

Figure 17

Figure 18, is the waveform that was made through figure 17 and 16.

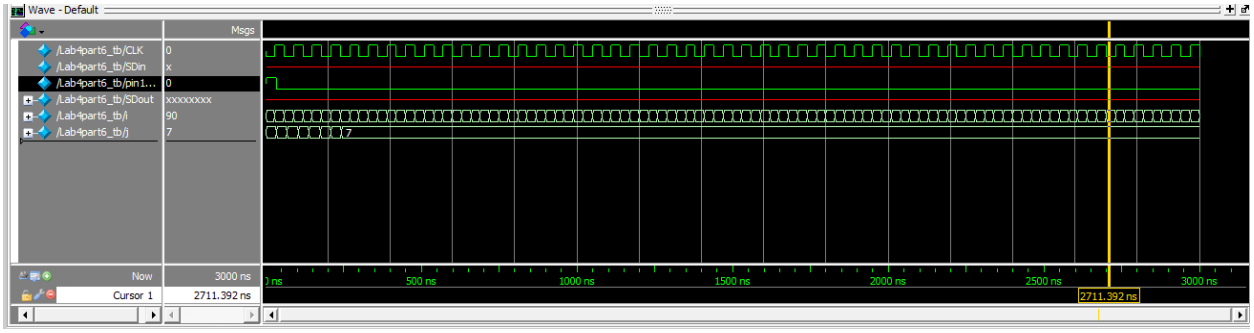

Figure 18

Part 7: Eight Bit Shift Register Design

Description:

Using an equation from each shift from a register, we will me making a Verilog equation using the 8 different assign statement

Procedure:

Design an eight bit shift register that can be synchronously pre-loaded via a control line called “PL” (Parallel Load). You need 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 you need to use eight assign statements for D7 – D0. Record your results and show the simulation waveform in your report.

Engineering Data:

Lab4part7.v | Lab4part7_tb.v | Compilation Report - Lab4 | Lab4part6_

```
1  module Lab4part7(clk, dirty, serial_in, p_load, SHL, SHR, ROR, ROL, Q);
2     input clk, dirty, serial_in, p_load, SHL, SHR, ROR, ROL;
3     output reg [7:0]Q;
4     wire [7:0]P;
5     wire clean;
6
7     assign P = 8'b01010101; //assigns a value to each input
8     debounce (clk,dirty,clean);
9     always @(posedge clean)
10    case ({p_load, SHL, SHR, ROR, ROL})
11    5'b10000: begin Q<=P; end
12    5'b01000: begin Q<=Q<<1; Q[0]<=serial_in; end
13    5'b00100: begin Q<=Q>>1; Q[7]<=serial_in; end
14    5'b00010: begin Q<=Q>>1; Q[7]<=Q[0]; end
15    5'b00001: begin Q<=Q<<1; Q[0]<=Q[7]; end
16
17    default Q<=Q;
18    endcase
19 endmodule
20
```

Figure 19

Figure 19 is the, Verilog since there is 8 different boxes that the input had to go through, we used Q[0] - Q[7] which represent the boxes. The boxes will have its own binary code that it will start out with. Figure 20 was the output of the code that was tested.

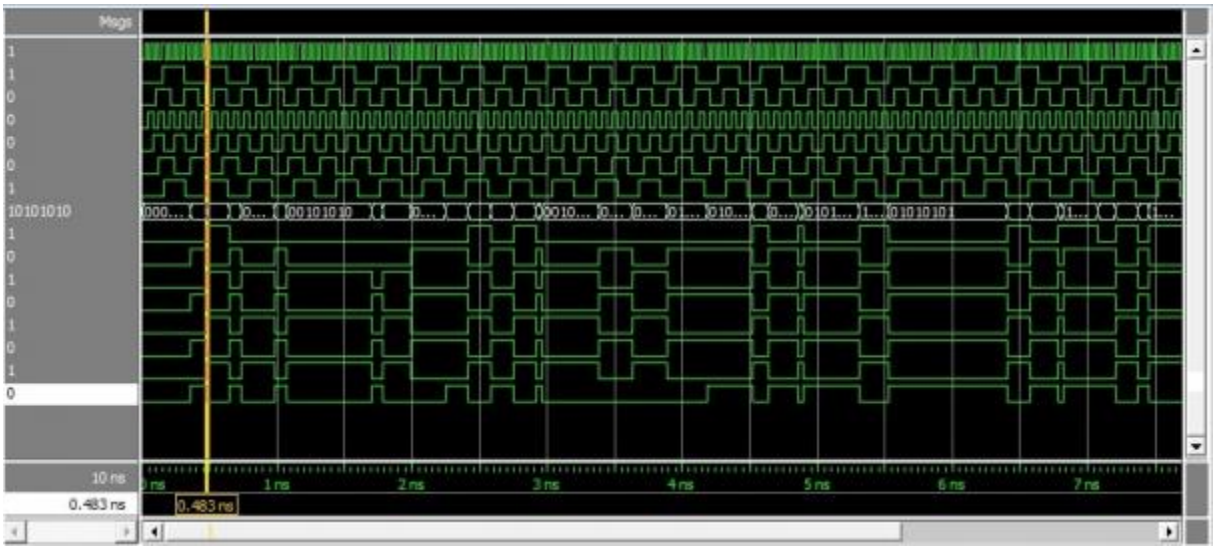

Figure 20

Conclusion:

In conclusion, this lab feels a bit rushed and majority of the stuff was blazed through and not really learn as clearly. This lab was quite difficult since some of the part require you to write up Verilog and testbenches that we can't even write, since it wasn't taught. The Verilog might be easier but when it come to the testbench, majority of the people aren't even sure how to begin it, but at the end it made some sense after the professor went over some of it when we demo. Overall this lab does help student learn more about flip flop and rs latchs but under a time crunch it is kinda hard to understand and memorize.

Questions:

QUESTION#1: What is the purpose of the two resistors in the schematic below? {Be sure to put all the answers to the Questions in your conclusion. Please number each question}

- The purpose of the resistor is to create and allow the gates to be pulled to high voltage and pulled to low voltage when connected to ground.

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

- In the NAND gate is active low which is when S and R are both zeros. In the NOR gate, S and R are active high since they both are 1.

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?

- Since the NOR version was active low, it implies that is due to the falling edge.

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)?

- Since the nor version is active low, the NAND gate has to be active high, which is caused by the positive edge of the clock.

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 the outputs are both high, it means the PR and Cr are active, this is an invalid state since it's not possible.

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)?

- They continue to carry the same characters of NAND gate being active low and NOR gate being active high.