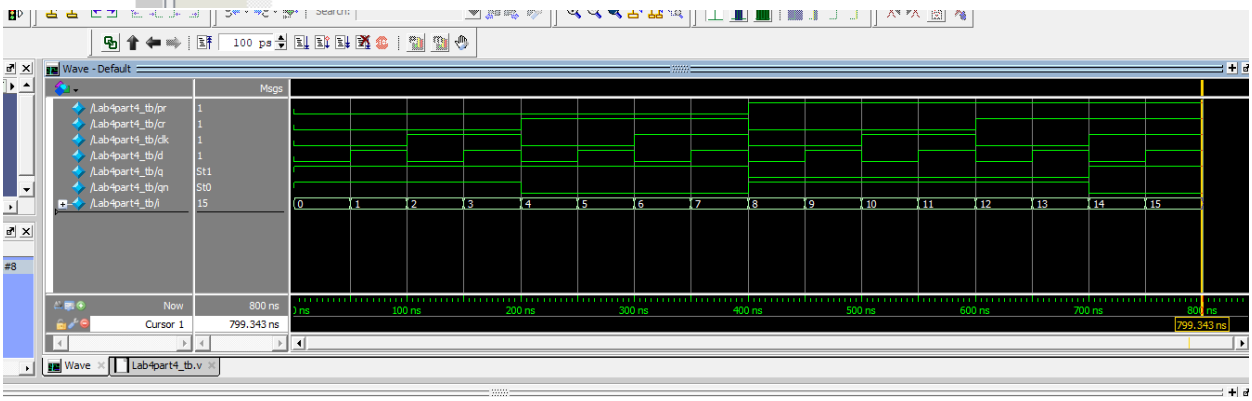
Part 4

Lab4part2.v | Lab4part2_tb.v | Lab4part3.v | Lab4part3_tb.v | Lab4part4.v | Lab4part4_tb.v | Lab4part5_tb

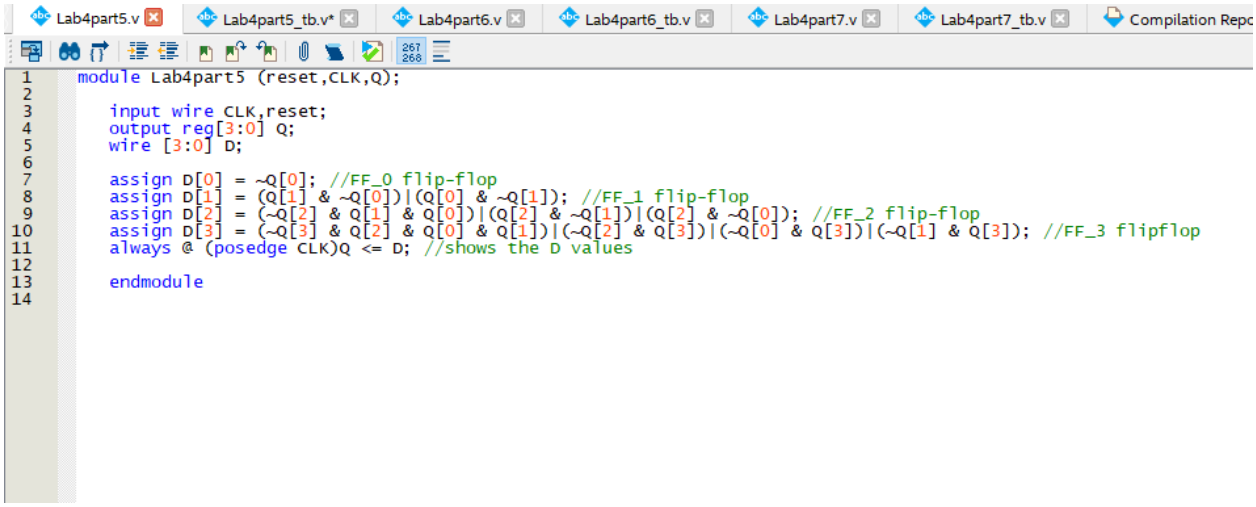
```
1  `timescale 1ns / 1ps
2  module Lab4part4_tb();
3              reg pr,cr,clk,d;
4              wire q, qn;
5              integer i;
6              Lab4part4 uut (pr,cr,clk,d,q,qn);
7
8              initial begin
9                 for(i = 0; i<= 15; i= i +1)
10                begin
11                   {pr,cr,clk,d} <= i;
12                   #50;
13                   #25;
14                   #100;
15                end
16
17                $stop;
18             end
19
20 endmodule
```

Lab4part4.v | Lab4part4_tb.v

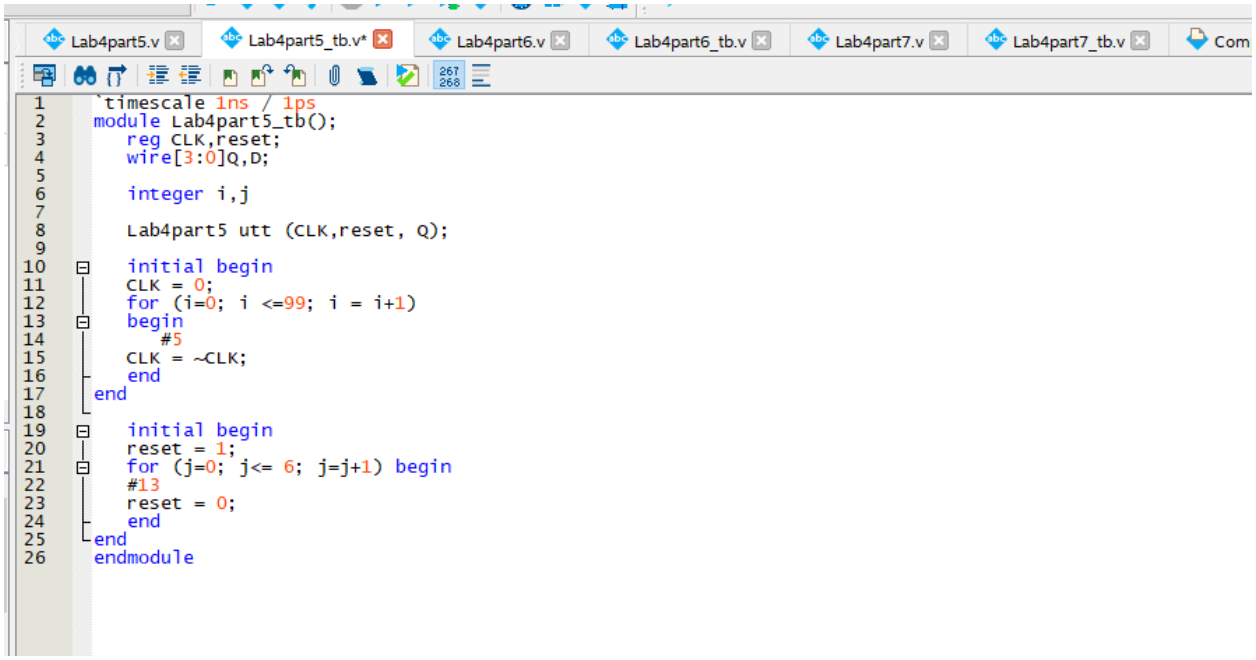
```
1  module Lab4part4(pr,cr,clk,d,q,qn, s,r);
2     input pr,cr,clk,d;
3     output q,qn,r,s;
4
5     wire in1,in2,in3,in4,pr, cr,clk,d,q,qn;
6
7     assign q = ~(qn & pr & s);
8     assign qn = ~(r & in2 & q);
9     assign s = ~(in1 & cr & clk);
10    assign r = cr;
11    assign in1 = ~(pr & in3 & s);
12    assign in2 = ~(clk & in3 & s);
13    assign in3 = ~(in2 & d & r);
14
15 endmodule
```



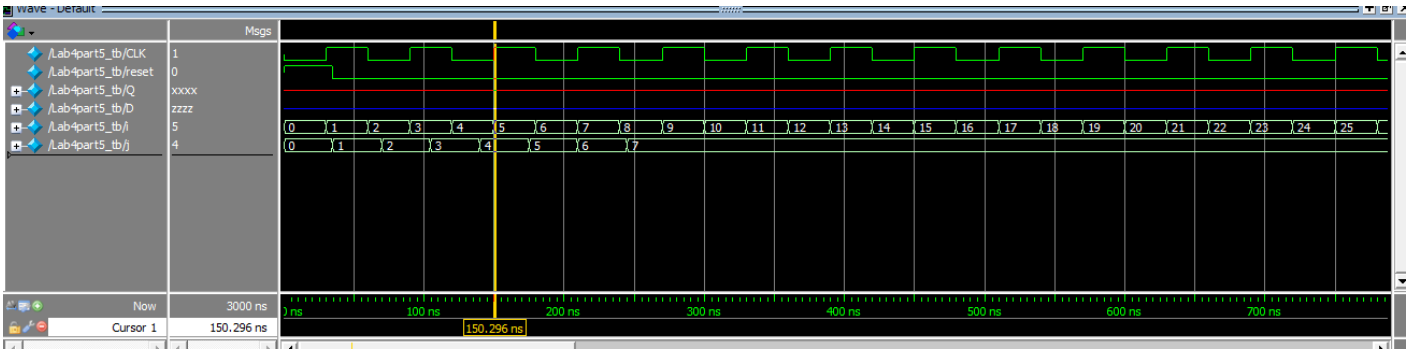
Part5



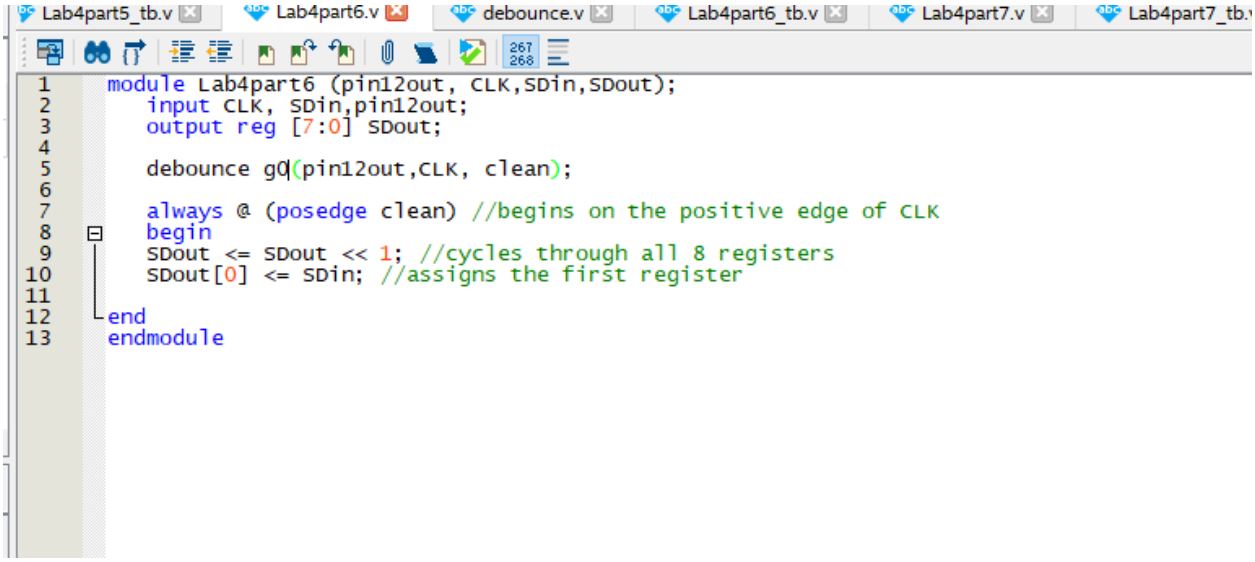
```
module Lab4part5 (reset,CLK,Q);

   input wire CLK,reset;
   output reg[3:0] Q;
   wire [3:0] D;

   assign D[0] = ~Q[0]; //FF_0 flip-flop
   assign D[1] = (Q[1] & ~Q[0])|(Q[0] & ~Q[1]); //FF_1 flip-flop
   assign D[2] = (~Q[2] & Q[1] & Q[0])|(Q[2] & ~Q[1])|(Q[2] & ~Q[0]); //FF_2 flip-flop
   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
   always @ (posedge CLK)Q <= D; //shows the D values

   endmodule
```



```
`timescale 1ns / 1ps
module Lab4part5_tb();
   reg CLK,reset;
   wire[3:0]Q,D;

   integer i,j

   Lab4part5 utt (CLK,reset, Q);

   initial begin
   CLK = 0;
   for (i=0; i <=99; i = i+1)
   begin
      #5
   CLK = ~CLK;
   end
end

   initial begin
   reset = 1;
   for (j=0; j<= 6; j=j+1) begin
   #13
   reset = 0;
   end
end
endmodule
```

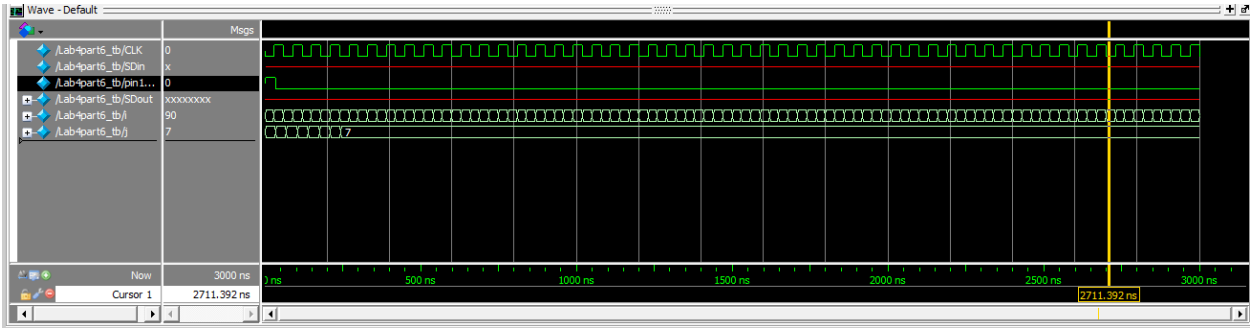

Wave - Default
Msgs
/Lab4part5_tb/CLK
/Lab4part5_tb/reset
/Lab4part5_tb/Q
/Lab4part5_tb/D
/Lab4part5_tb/i
/Lab4part5_tb/j
1
0
xxxx
zzzz
5
4
Now
3000 ns
Cursor 1
150.296 ns
100 ns
200 ns
300 ns
400 ns
500 ns
600 ns
700 ns

Part 6


Lab4part5_tb.v
Lab4part6.v
debounce.v
Lab4part6_tb.v
Lab4part7.v
Lab4part7_tb.v
module Lab4part6 (pin12out, CLK,SDin,SDout);
input CLK, SDin,pin12out;
output reg [7:0] SDout;
debounce g0(pin12out,CLK, clean);
always @ (posedge clean) //begins on the positive edge of CLK
begin
SDout <= SDout << 1; //cycles through all 8 registers
SDout[0] <= SDin; //assigns the first register
end
endmodule

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

```
1   `timescale 1ns / 1ps
2   module Lab4part6_tb();
3      reg CLK, SDin, pin12out;
4      wire[7:0] SDout;
5
6      integer i,j;
7
8      Lab4part6 utt (pin12out, CLK,SDin,SDout);
9
10     initial begin
11     CLK = 0;
12     for (i=0; i <=99; i = i+1)
13     begin
14        #5;
15        #25;
16     CLK = ~CLK;
17     end
18  end
19
20     initial begin
21      pin12out = 1;
22     for (j=0; j<= 6; j=j+1) begin
23     #15;
24     #20;
25     pin12out = 0;
26     end
27  end
28  endmodule
```



Part 7

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
```