University of Tehran
College of Engineering
School of Electrical and Computer Engineering

Digital Systems 1

Dr.Navabi

Computer Assignment 4

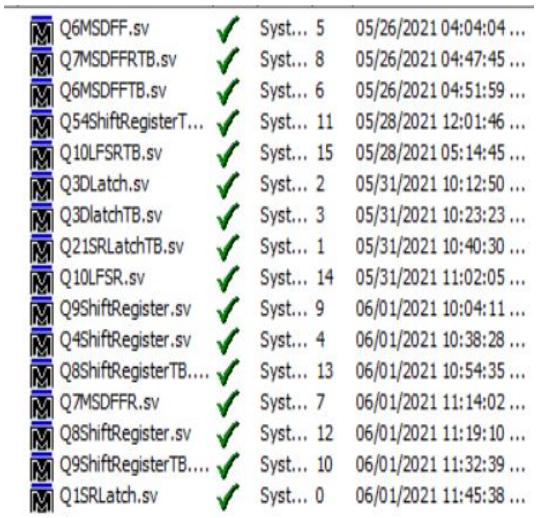
Soroush Mesforush Mashhad

SN:810198472

Khordad 00

1 Project generated files

A full view of the generated and used SystemVerilog files in my project can be seen below:



Q6MSDFF.sv	✓	Syst... 5	05/26/2021 04:04:04 ...
Q7MSDFFRTB.sv	✓	Syst... 8	05/26/2021 04:47:45 ...
Q6MSDFFTB.sv	✓	Syst... 6	05/26/2021 04:51:59 ...
Q54ShiftRegisterT...	✓	Syst... 11	05/28/2021 12:01:46 ...
Q10LFSRTB.sv	✓	Syst... 15	05/28/2021 05:14:45 ...
Q3DLatch.sv	✓	Syst... 2	05/31/2021 10:12:50 ...
Q3DlatchTB.sv	✓	Syst... 3	05/31/2021 10:23:23 ...
Q21SRLatchTB.sv	✓	Syst... 1	05/31/2021 10:40:30 ...
Q10LFSR.sv	✓	Syst... 14	05/31/2021 11:02:05 ...
Q9ShiftRegister.sv	✓	Syst... 9	06/01/2021 10:04:11 ...
Q4ShiftRegister.sv	✓	Syst... 4	06/01/2021 10:38:28 ...
Q8ShiftRegisterTB....	✓	Syst... 13	06/01/2021 10:54:35 ...
Q7MSDFFR.sv	✓	Syst... 7	06/01/2021 11:14:02 ...
Q8ShiftRegister.sv	✓	Syst... 12	06/01/2021 11:19:10 ...
Q9ShiftRegisterTB....	✓	Syst... 10	06/01/2021 11:32:39 ...
Q1SRLatch.sv	✓	Syst... 0	06/01/2021 11:45:38 ...

Figure 1: Generated and Compiled .sv files

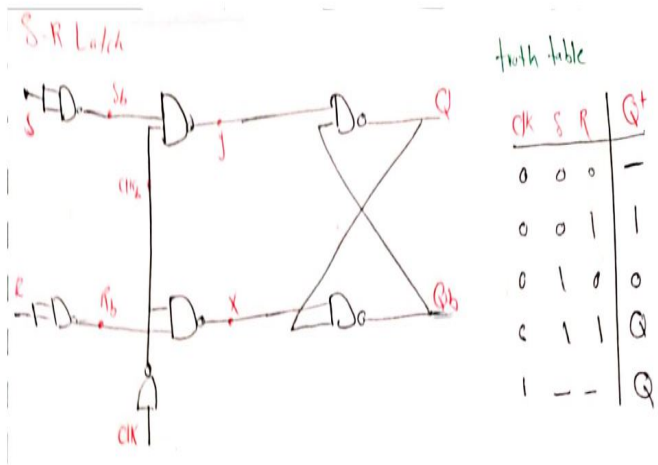


Figure 2: Hand Simulation

2 Questions

2.1 Q1 & Q2

Each of the nand gates have an 8ns delay respectively.

2.1.1 SystemVerilog Code & Testbench

```
`timescale 1ns/1ns
module Q1SRLatch(input S,R,CLK ,output Q,Qb);
    wire x,y,Sb,Rb,CLKb;
    nand #8 (Sb, S , S);
    nand #8 (Rb , R ,R);
    nand #8 (CLKb , CLK , CLK);
    nand #8 (x , Rb , CLKb);
    nand #8 (y , Sb , CLKb);
    nand #8 (Qb,Q,x);
    nand #8(Q , Qb , y);
endmodule
```

```
`timescale 1ns/1ns
module Q21SRLatchTB();
  logic CLK=0;
  logic S=0;
  logic R=0;
  wire Qb,Q;
  Q1SRLatch UUT (S,R,CLK,Q,Qb);
always #65 CLK<=~CLK;
  initial begin
    #100 R = 1;
    #100 R = 0;
    #100 S = 1;
    #100 R = 1;
    #100 R = 0;
    #100 S = 0;
    #100 $stop;
  end
endmodule
```

Figure 3: SystemVerilog Code & Testbench

2.1.2 Waveform

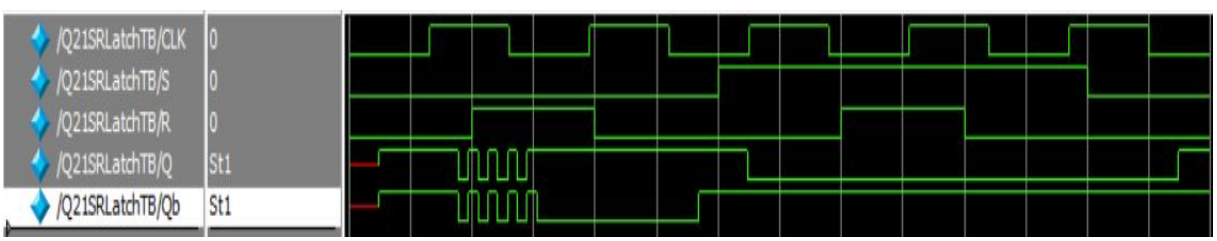


Figure 4: SRLatch Waveform

2.2 Q3

2.2.1 Hand-Simulation

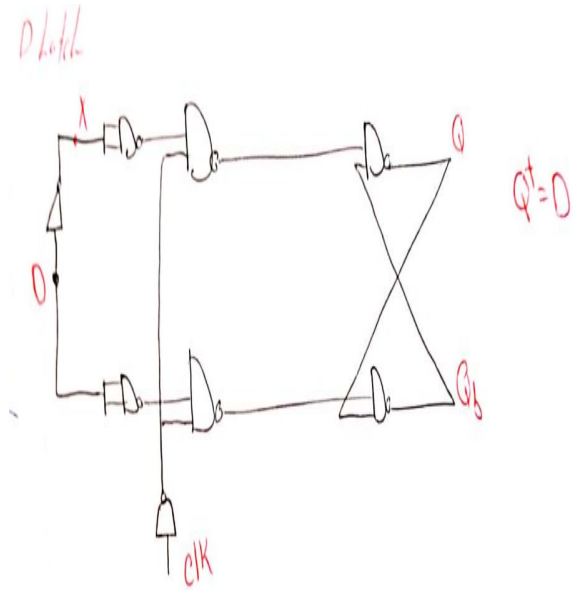


Figure 5: Hand Simulation

We fullfil our need to make a DLatch by using a not gate with a 6ns delay and by utilizing the previous SRLatch.

2.2.2 SystemVerilog Code & Testbench

```
`timescale 1ns/1ps
module Q3DLatch(input D ,CLK ,output Q,Qb);
    wire X;
    not #6 (X,D);
    Q1SRLatch G1 (.S(X),.R(D),.CLK(CLK),.Q(Q),.Qb(Qb));
endmodule
```

```
`timescale 1ns/1ps
module Q3DLatchTB();
  logic CLK=0;
  logic D=0;
  wire Qb,Q;
  Q3DLatch UUT (D,CLK,Q,Qb);
  always #50 CLK<=~CLK;
  initial begin
    #50 D = 0;
    #50 D = 1;
    #50 D = 0;
    #50 D = 1;
    #50 D = 1;
    #50 D = 0;
    #50 $stop;
  end
endmodule
```

Figure 6: SystemVerilog Code & Testbench

2.2.3 Waveform

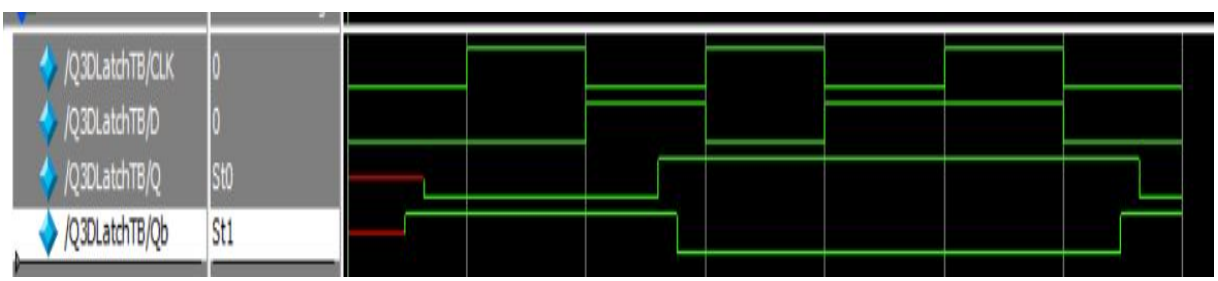


Figure 7: DLatch Waveform

2.3 Q4 & Q5

2.3.1 Hand-Simulation

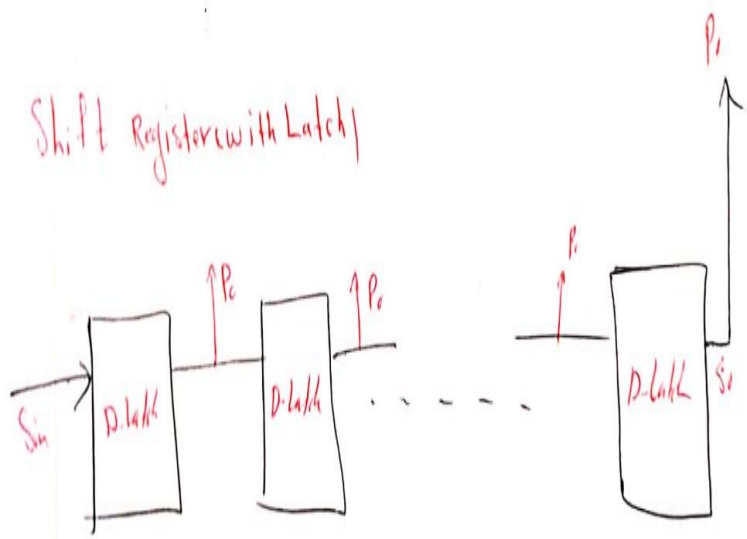


Figure 8: Hand Simulation

In this part we connect the DLatches respectively, but this design shall not work properly due to the fact that we don't have a Master-Slave here hence the input jumps right onto the output.

2.3.2 SystemVerilog Code & Testbench

```
`timescale 1ns/1ns
module Q4ShiftRegister(input si,CLK,output [7:0]PO);
        wire [8:0]Con;
        assign Con[8]=si;
        genvar i;
        generate
                for(i=0;i<8;i=i+1)begin:DFFs
                        Q3DLatch Latches(Con[8-i],CLK,Con[8-i-1]);
                end
        endgenerate
        assign PO=Con[7:0];
endmodule
```

```
`timescale 1ns/1ns
module Q54ShiftRegisterTB();
logic si=0,CLK=0;
wire [7:0] PO;
Q4ShiftRegister UUT(si,CLK,PO);
always #40 CLK<=~CLK;
initial begin
  repeat(8) #50 si=$random();
  #800 $stop;
 end
endmodule
```

Figure 9: Verilog Code & Testbench

2.3.3 Waveform

Figure 10: Q4ShiftRegister Waveform

2.4 Q6

Here by connecting two of the DLatches consecutively we create the desired MSDFF.

2.4.1 SystemVerilog Code & Testbench

```
`timescale 1ns/1ps
module Q6MSDFF(input D ,CLK ,output Q,Qb);
    wire x,y;
    Q3DLatch M(.D(D),.CLK(CLK),.Q(x),.Qb(y));
    Q3DLatch S(.D(x),.CLK(~CLK),.Q(Q),.Qb(Qb));
endmodule
```

```
`timescale 1ns/1ps
module Q6MSDFFTB();
  logic CLK=0;
  logic D=0;
  wire Qb,Q;
  Q6MSDFF UUT (D,CLK,Q,Qb);
  always #40 CLK<=~CLK;
  initial begin
    #100 D = 0;
    #100 D = 1;
    #100 D = 0;
    #100 D = 1;
    #100 D = 1;
    #100 D = 0;
    #100 $stop;
  end
endmodule
```

Figure 11: SystemVerilog Code & Testbench

2.4.2 Waveform

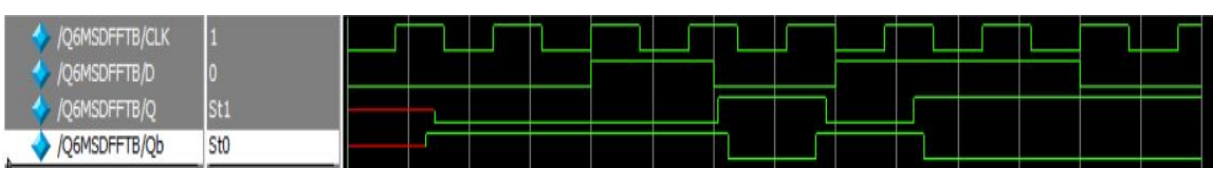


Figure 12: MSDFF Waveform

2.5 Q7

To create the MSDFFR we use the exact design of part 6 we merely add an if statement and we shall have the MSDFFR.

2.5.1 SystemVerilog Code & Testbench

```
`timescale 1ns/1ns
module Q7MSDFFR(input D ,CLK,RST,output logic Q,Qb);
    wire x,y;
    Q3DLatch M(.D(D),.CLK(CLK),.Q(x),.Qb(y));
    Q3DLatch S(.D(x),.CLK(~CLK),.Q(Q),.Qb(Qb));
    always@(RST)begin
     if(RST)
       Q<=1'b0;
       Qb<=1'b1;
    end
endmodule
```

```
`timescale 1ns/1ns
module Q7MSDFFTB();
  logic CLK=0;
  logic D=0;
  logic RST=0;
  wire Qb,Q;
  Q7MSDFFR UUT (D,CLK,RST,Q,Qb);
  always #40 CLK<=~CLK;
  initial begin
    #100 D = 0;
    #100 D = 1;
    #100 D = 0;
    #100 D = 1;
    #100 D = 1;
    #50 RST=1;
    #100 D = 0;
    #100 RST=1;
    #100 $stop;
  end
endmodule
```

Figure 13: SystemVerilog Code & Testbench

2.5.2 Waveform

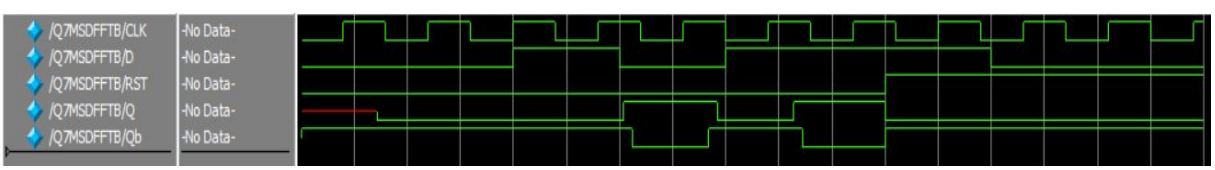


Figure 14: MSDFFR Waveform

2.6 Q8

2.6.1 Hand-Simulation

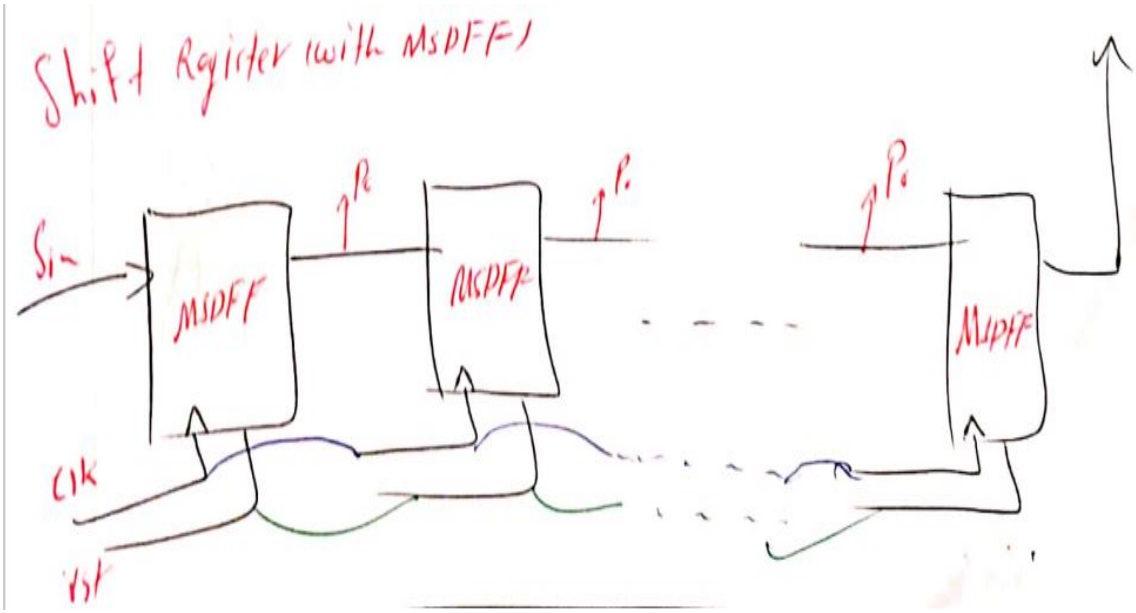


Figure 15: Hand Simulation

Here we connect 8 of the MSDFFRs to make a shift register, here this shift register shall work properly because with the utilization of the MSDFFR we solve the timing problem.

2.6.2 SystemVerilog Code & Testbench

```
`timescale 1ns/1ns
module Q8ShiftRegister(input si,CLK,RST,output [7:0]PO);
        wire [8:0]Con;
        assign Con[0]=si;
        genvar i;
        generate
                for(i=0;i<8;i=i+1)begin:DFFs
                        Q7MSDFFR Latches(Con[8-i],CLK,RST,Con[8-i-1]);
                end
        endgenerate
        assign PO=Con[7:0];
endmodule
```

```
`timescale 1ns/1ns
module Q8ShiftRegisterTB();
logic si=0,CLK=0,RST=0;
wire [7:0] PO;
Q8ShiftRegister UUT(si,CLK,RST,PO);
always #40 CLK<=~CLK;
initial begin
  repeat(8) #50 si=$random();
  #800 $stop;
 end
endmodule
```

Figure 16: SystemVerilog Code & Testbench

2.6.3 Waveform

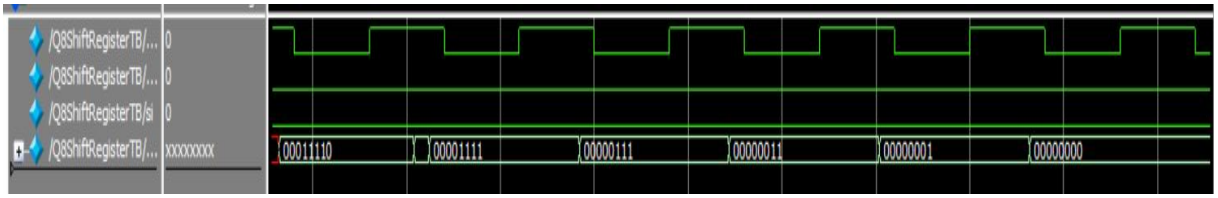


Figure 17: Q8ShiftRegister Waveform

2.7 Q9

Here somewhat like Cprogramming we use the help of a temporary wire and with the help of the always statement we create the shift register.

2.7.1 SystemVerilog Code & Testbench

```
`timescale 1ns/1ns
module Q9ShiftRegister(input [7:0]PI, input CLK,si,output logic[7:0]PO);
    logic[7:0]temp;
    integer cnt=0;
    always@(negedge CLK)begin
      if(cnt==0)begin
          temp<=PI;
      end
      else begin
          temp<=PO;
      end
      cnt=cnt+1;
    end
      assign PO = {si, temp[7:1]};
endmodule
```

```
`timescale 1ns/1ns
module Q9ShifRegisterTB();
  logic [7:0] PI = 8'b10010100;
  logic si = 0 , CLK = 0;
  wire [7:0] PO;
  Q9ShiftRegister UUT(PI, CLK, si, PO);
  always #40 CLK = ~CLK;
 initial begin
  #800 $stop;
 end
endmodule
```

Figure 18: SystemVerilog Code & Testbench

2.7.2 Waveform

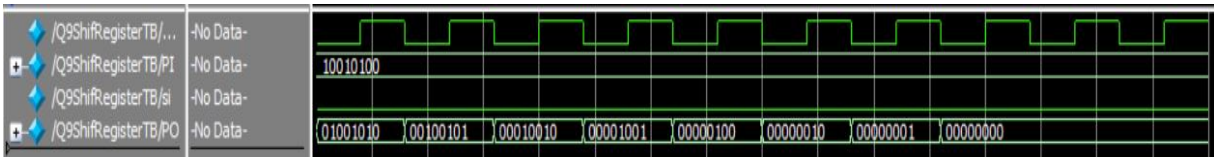


Figure 19: Q9ShiftRegister Waveform

2.8 Q10

2.8.1 Hand-Simulation

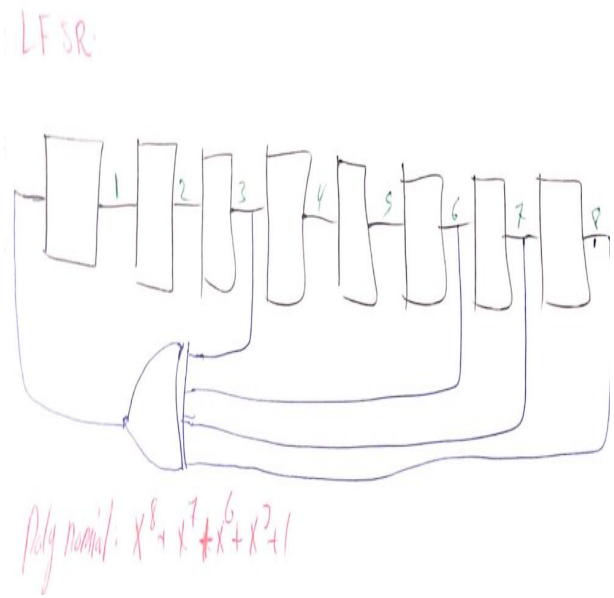


Figure 20: Hand Simulation

Here we create an LFSR(Linear Feedback Shift Register), these shift registers are used to generate pseudo-random numbers and so forth, we create this register by xoring the wires which are shown to us by the following polynomial and feeding it back in as the serial input.

$$x^8 + x^7 + x^6 + x^3 + 1$$

2.8.2 SystemVerilog Code & Testbench

```
`timescale 1ns/1ns
module Q10LFSR(input[7:0]PI,input CLK,output [7:0]PO);
        logic z;
        xor(z,PO[0],PO[1],PO[2],PO[5]);
       Q9ShiftRegister SR1 (.PI(PI),.CLK(CLK),.si(z),.PO(PO));
endmodule
```

```
`timescale 1ns/1ns
module Q10LFSRTB();
  logic [7:0] PI = 8'b10010100;
  logic CLK = 0;
  wire [7:0] PO;
  Q10LFSR UUT(PI,CLK,PO);
  always #40 CLK = ~CLK;
 initial begin
  #20500 $stop;
 end
endmodule
```

Figure 21: SystemVerilog Code & Testbench

2.8.3 Waveform

Figure 22: LFSR Waveform