University of Tehran

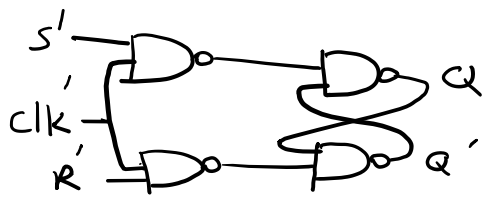
DLD computer assignment 4

Mohammad Abaeiani

Std: 810198432

Part 1&2

Shematics:



Nand delays :

Max(2*nmos,pmos) = 2*4 = 8ns

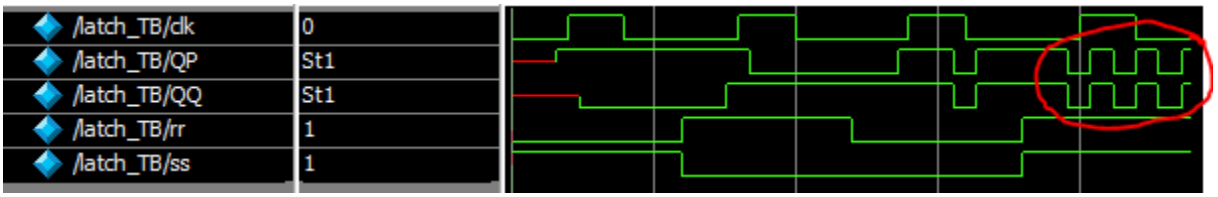
Verilog:

```
`timescale 1ns/1ns

module sr_latch(input clk,s,r,output Q,Q_p);
    not(si,s);
    not(ri,r);
    not(clki,clk);
    nand #8 N1(o1,si,clki);
    nand #8 N2(o2,ri,clki);
    nand #8 N3(Q,o1,Q_p);
    nand #8 N4(Q_p,o2,Q);
endmodule
```

TestBench:

```
`timescale 1ns/1ns

module SR_latch_TB();
    logic ss,rr,clk=0;
    wire QQ,QP;
    sr_latch L1(clk,ss,rr,QQ,QP);
    initial begin
        #0 ss=1;rr=0;
        #20 clk = 1;
        #20 clk = 0;
        #20 ss =0; rr=1;
        #20 clk = 1;
        #20 clk = 0;
        #20 ss = 0;rr=0;
        #20 clk = 1;
        #20 clk = 0;
        #20 ss=1; rr=1;
        #20 clk = 1;
        #20 clk = 0;
        #20 $stop;
    end
endmodule
```

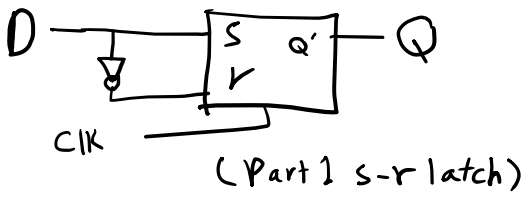
WaveForm:



The highlighted part is the memory loss; in which both s and r become zero and the nand gates alternate between 10 and 11 inputs constantly, while both Q and Q' are the same.

Part 3

Schematics:



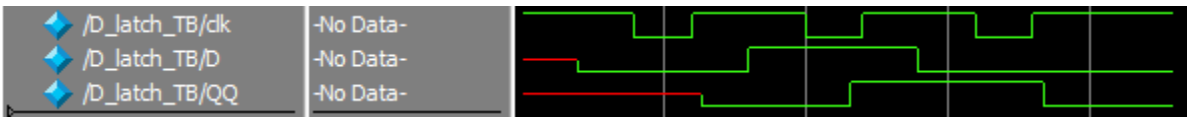
Verilog:

```
`timescale 1ns/1ns

module D_latch(input clk,D,output Q);
    not N1(Di,D);
    wire Q_p;
    sr_latch L1(clk,D,Di,Q_p,Q);
endmodule
```

TestBench:

```
`timescale 1ns/1ns
module D_latch_TB();
    logic D,clk =1;
    wire QQ;
    D_latch D1(clk,D,QQ);
    initial begin
        #20 D = 0;
        #20 clk =0;
        #20 clk =1;
        #20 D = 1;
        #20 clk = 0;
        #20 clk = 1;
        #20 D = 0;
        #20 clk =0;
        #20 clk =1;
        #50 $stop;
    end
endmodule
```

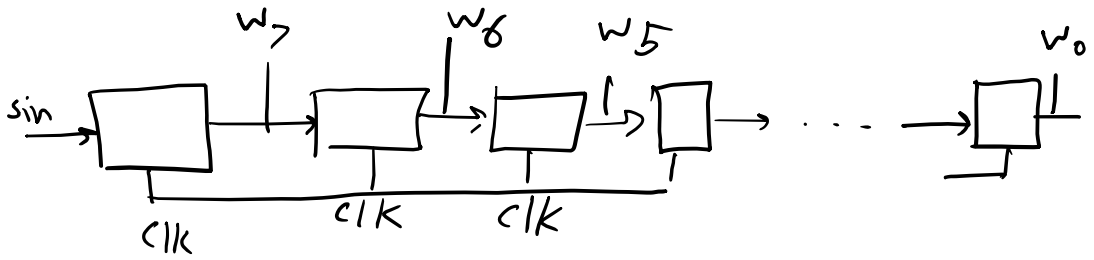
Waveform:



note that the output is Q' since the inputs of the latch are active low.

Part 4&5

Schematics:



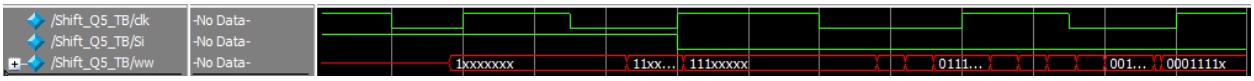
Verilog:

```
`timescale 1ns/1ns
module shift_reg8(input clk,Sin,output [7:0]W);
    wire [8:0]g;
    assign g[8] = Sin;
    genvar k;
    generate
        for(k=0;k<8;k = k+1)begin: regs
            D_latch XX(clk,g[8-k],g[7-k]);
        end
    endgenerate
    assign W =g;
endmodule
```

TestBench:

```
`timescale 1ns/1ns

module Shift_Q5_TB();
    logic Si = 1;
    wire [7:0]ww;
    logic clk =1;
    shift_reg8 S1(clk,Si,ww);
    initial begin
        #20 clk =0;
        #20 clk =1;
        #30 clk =0;
        #30 clk =1; Si =0;
        #40 clk =0;
        #40 clk =1;
        #30 clk =0;
        #30 clk =1;
        #20 $stop;
    end
endmodule
```

Wave Form:

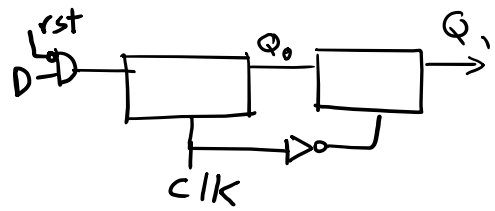


It is concluded from the wave form that by extending the clock active time(from 20 to 30 to 40) the transparency emerges in the wave form and during each cycle there will be more than 1 shift(the more active time the more bit shifts); so this design is not so functional overall.

The reason of this is because the shifter turns on not on the edge of the clock but during the whole clock active time.

Part 6&7

Schematics:



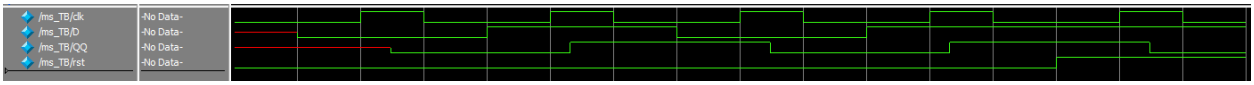
Verilog:

```
`timescale 1ns/1ns

module ms_ff(input clk,D,rst,output Q);
    assign clki = ~clk;
    wire rsti,Do;

    //active high reset
    not(rsti,rst);
    and(Do,D,rsti);
    //master slave
    D_latch D1(clk,Do,Q1);
    D_latch D2(clki,Q1,Q);
endmodule
```

Test Bench:

```
`timescale 1ns/1ns
module ms_TB();
    logic D,clk =0,rst=0;
    wire QQ;

    ms_ff D1(clk,D,rst,QQ);
    initial begin
        #50 D = 0;
        #50 clk =1;
        #50 clk =0;
        #50 D = 1;
        #50 clk = 1;
        #50 clk = 0;
        #50 D = 0;
        #50 clk =1;
        #50 clk =0;
        #50 D = 1;
        #50 clk =1;
        #50 clk =0;
        #50 rst=1;
        #50 clk =1;
        #50 clk =0;
        #50 $stop;
        end
endmodule
```

Wave Form:



It could be dedicated that at the positive edge of clock the output saves the D input unless reset is active, if it is, the output becomes 0 at the posedge of clock even if the D input is one.

Part 8

Verilog:

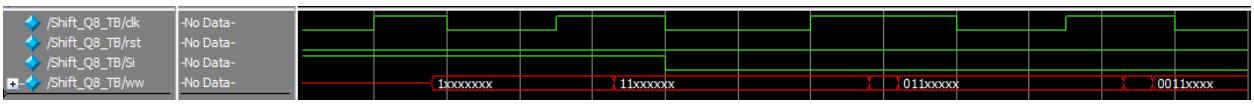
```
module sr_shift_reg8(input clk,Sin,rst,output [7:0]W);
    wire [8:0]g;
    assign g[8] = Sin;
    genvar k;
    generate
        for(k=0;k<8;k = k+1)begin: regs
            ms_ff XX(clk,g[8-k],rst,g[7-k]);
        end
    endgenerate
    assign W =g;
endmodule
```

Test Bench:

```
`timescale 1ns/1ns

module Shift_Q8_TB();
    logic Si = 1;
    wire [7:0]ww;
    logic clk =0;
    logic rst =0;
    sr_shift_reg8 S1(clk,Si,rst,ww);
    initial begin
        #20 clk =1;
        #20 clk =0;
        #30 clk =1;
        #30 clk =0; Si =0;
        #40 clk =1;
        #40 clk =0;
        #30 clk =1;
        #30 clk =0;
        #20 $stop;
    end
endmodule
```

(Here the test bench is the same as the former shift register in order to compare them)

Wave Form:



Here it is dedicated that with each clock the output is shifted only once no matter how long the clock active time is, unlike the former shift register.

There are some tiny glitches on the shift to 0 from left which are because of time difference between the latch output To0 and To1 time.

Part 9

Verilog:

```
`timescale 1ns/1ns

module Shift_Q9(input clk,Sin,rst,output logic [7:0]W);
    always@(posedge clk)begin
        if(rst)
            W<= 8'b0;
        else
            W<={Sin,W[7:1]};
    end
endmodule
```

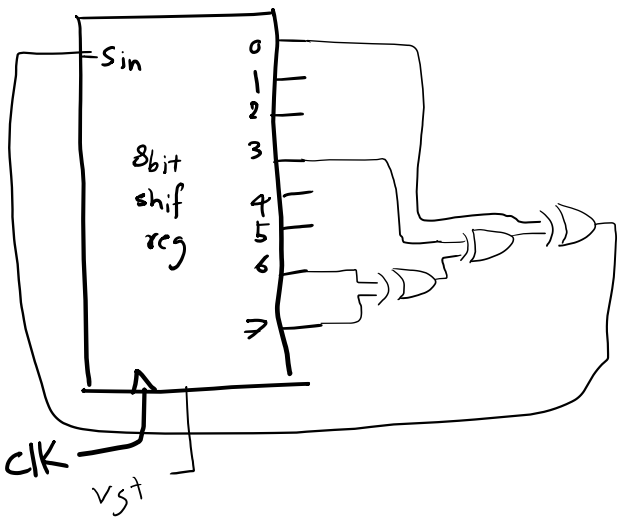
Test Bench:

```
`timescale 1ns/1ns

module Shift_Q9_TB();
    logic Si = 1;
    wire [7:0]ww;
    logic clk =0;
    logic rst =0;
    Shift_Q9 S1(clk,Si,rst,ww);
    initial begin
        #20 clk =1;
        #20 clk =0;
        #30 clk =1;
        #30 clk =0; Si =0;
        #40 clk =1;
        #40 clk =0;
        #30 clk =1;
        #30 clk =0;
        #20 $stop;
    end
endmodule
```

Wave Form:

Part 10

Schematics:



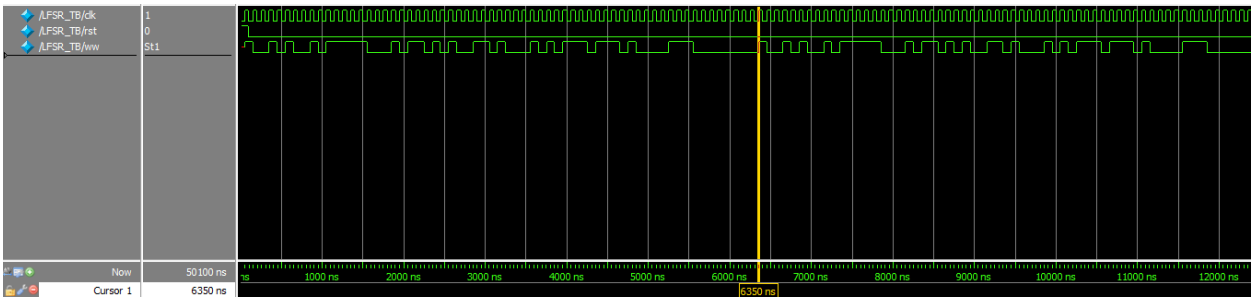
Verilog:

```
module LFSR(input clk,rst,output w);
    logic Si;
    wire [7:0]Po;
    assign Si = ~(Po[7] ^ Po[6] ^ Po[3] ^ Po[0]);
        Shift_Q9 S1(clk,Si,rst,Po);
    assign w = Si;
endmodule
```

Test bench:

```
`timescale 1ns/1ns

module LFSR_TB();
    logic clk=0,rst=1;

    wire ww;
    LFSR L1(clk,rst,ww);
    always #50 clk = ~clk;
        initial begin;
            #100 rst=0;
            #50000 $stop;
    end
endmodule
```

Wave form:



As we can conclude from wave form , considering the clocking started at 50ns(0 of rst) knowing each clock cycle is 100ns the period of the pattern is 63 clock cycles.

As I found out every polynomial has its own period no matter what the initial condition is, and by finding the best polynomial we could get 2^n -1 period-length in an n bit shifter.