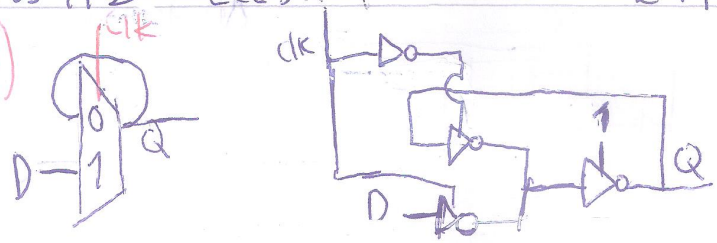
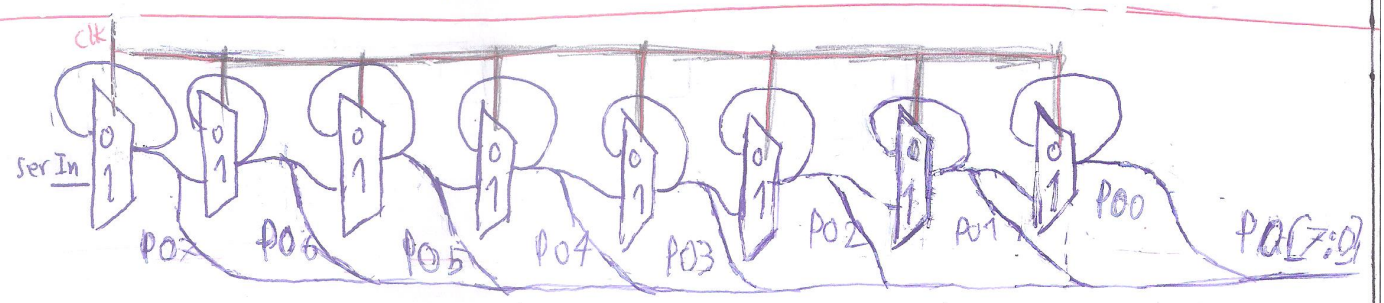
810 99 442 ECE367 | Digital Logic System LA4 | Arash Shahin

1) b)



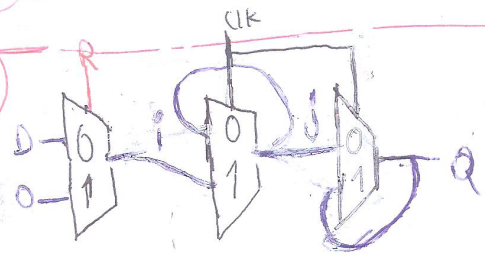
2) a)



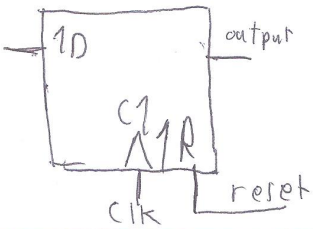
2) b) It doesn't shift the number like we wanted to but insteads, when clock is 1, it turns all of our bits into the number of SerIn. this is because we don't "hold" information the same way we do with a master-slave flip flop. In the master slave, the information goes to the "next level" Once we change our clock and doesn't go to the next level untill we change our clock again. In our latch though once our clock is 1 we keep going to the next level untill we reach the end and only stop if the clock becomes 0 then we will hold the current information untill the clock becomes 1 again

4) a)

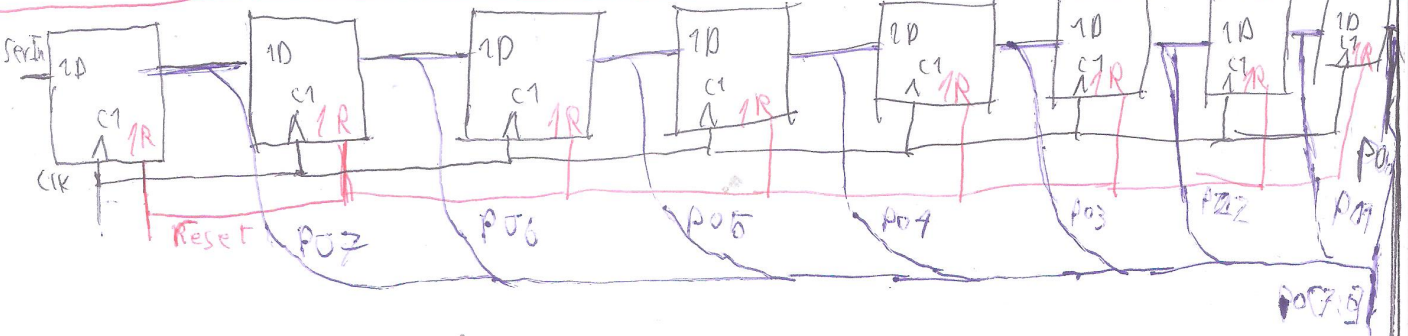
3) a)



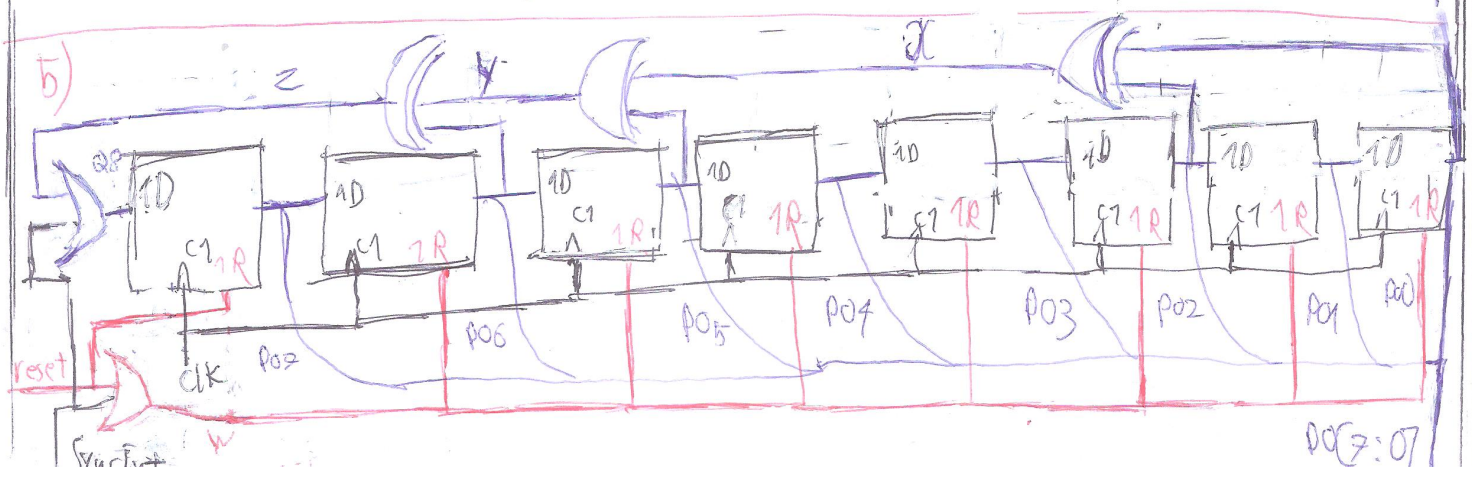
Block Diagram:



4) a)

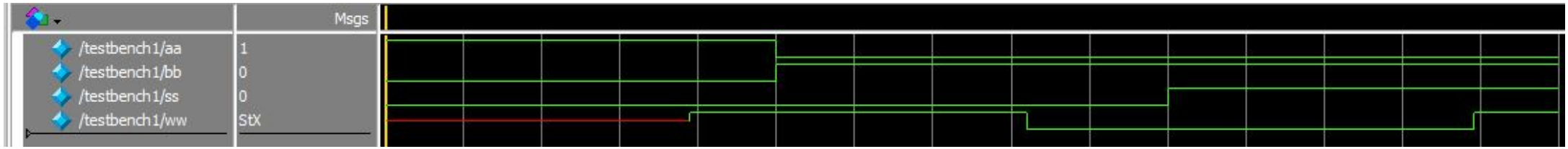


b)



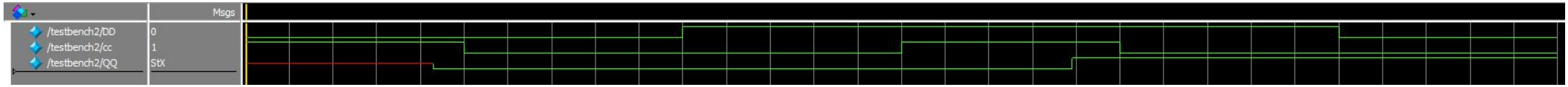
```
1   `timescale 1ns/1ns
2   module mymulti(input a,b,s, output w);
3           wire i,j;
4           not #(7,9) N1(i,s);
5           notif1 #(14,18,16) NT1(j,a,i);
6           notif1 #(14,18,16) NT2(j,b,s);
7           notif1 #(14,18,16) NT3(w,j,1);
8   endmodule
```

```
1   `timescale 1ns/1ns
2   module testbenchl ();
3           logic aa=1,bb=0,ss=0;
4           wire ww;
5           mymulti MMT(.a(aa),.b(bb),.s(ss),.w(ww));
6           initial begin
7           #50 aa=0;bb=1;
8           #50 ss=1;
9           #50 $stop;
10          end
11  endmodule
```


Msgs
/testbench1/aa
/testbench1/bb
/testbench1/ss
/testbench1/ww
1
0
0
StX

```
1   `timescale lns/lns
2   module mydlatch(input D,clk, output Q);
3           mymulti MT(Q,D,clk,Q);
4   endmodule
```

```
1   `timescale 1ns/1ns
2   module testbench2 ();
3           logic DD=0,cc=1;
4           wire QQ;
5           mydlatch MDL(.D(DD),.clk(cc),.Q(QQ));
6           initial begin
7           #50 cc=0;
8           #50 DD=1;
9           #50 cc=1; DD=1;
10          #50 cc=0;
11          #50 DD=0;
12          #50 $stop;
13          end
14  endmodule
```

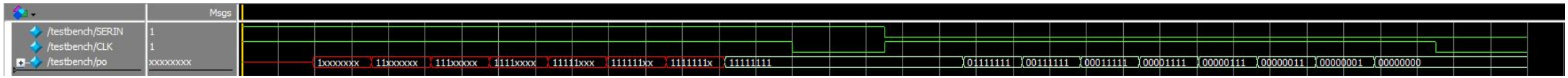

Msgs
/testbench2/DD
/testbench2/cc
/testbench2/QQ
0
1
StX

```
1   `timescale lns/lns
2   module testbench3 ();
3           logic DD=0,cc=1;
4           wire QQ;
5           mydlatch MDL(.D(DD),.clk(cc),.Q(QQ));
6           initial begin
7           #50 DD=1;
8           #20 cc=0;
9           #80 cc=1;
10          #50 DD=0;
11          #20 DD=1;
12          #40 cc=0;
13          #10 cc=1;DD=0;
14          #20 cc=0;
15          #30 $stop;
16          end
17  endmodule
```


Msgs
/testbench3/DD
/testbench3/cc
/testbench3/QQ
0
1
StX

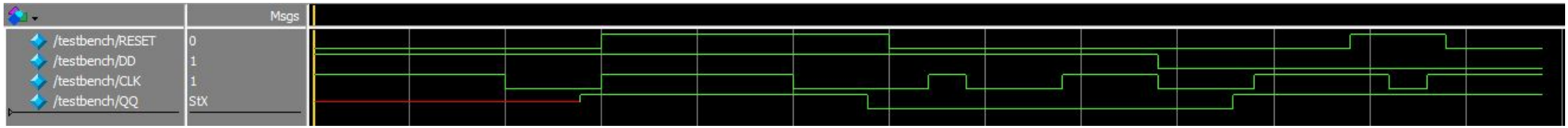
```
1   `timescale 1ns/1ns
2   module mydshift (input serIn,clk, output [7:0] PO);
3           wire [8:0] Q;
4           assign Q[8] = serIn;
5           genvar i;
6           generate
7               for (i=8; i > 0; i=i-1) begin
8                   mydlatch MDL(Q[i],clk,Q[i-1]);
9               end
10          endgenerate
11          assign PO = Q[7:0];
12  endmodule
```

```
1   `timescale 1ns/1ns
2   module testbench ();
3           logic SERIN=1,CLK=1;
4           wire [7:0] po;
5           mydshift MDS(.serIn(SERIN),.clk(CLK),.PO(po));
6           initial begin
7           #300 CLK=0;
8           #50 CLK=1;SERIN=0;
9           #300 CLK=0;
10          #50 $stop;
11          end
12  endmodule
```


Msgs
/testbench/SERIN
/testbench/CLK
/testbench/po
1
1
xxxxxxxx
1xxxxxxx
11xxxxxx
111xxxxx
1111xxxx
11111xxx
111111xx
1111111x
11111111
01111111
00111111
00011111
00001111
00000111
00000011
00000001
00000000

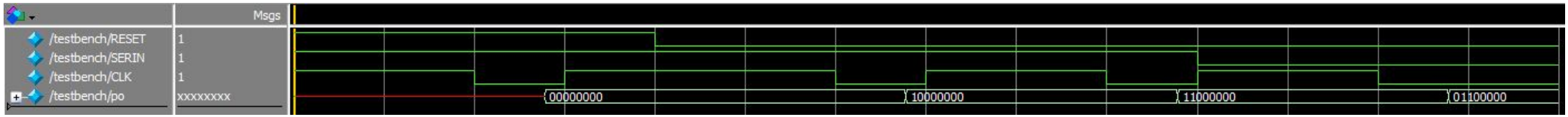
```
1   `timescale 1ns/1ns
2   module myMSDFF (input D,reset,clk, output Q);
3           wire i,j;
4           mymulti MT1(D,0,reset,i);
5           mymulti MT2(j,i,clk,j);
6           mymulti MT3(j,Q,clk,Q);
7   endmodule
```

```
1   `timescale 1ns/1ns
2   module testbench ();
3           logic RESET=0,DD=1,CLK=1;
4           wire QQ;
5           myMSDFF MDS(.D(DD),.reset(RESET),.clk(CLK),.Q(QQ));
6           initial begin
7           #100 CLK=0;
8           #50 CLK=1;RESET=1;
9           #100 CLK=0;
10          #50 RESET=0;
11          #20 CLK=1;
12          #20 CLK=0;
13          #50 CLK=1;
14          #50 CLK=0;DD=0;
15          #50 CLK=1;
16          #50 RESET=1;
17          #20 CLK=0;
18          #20 CLK=1;
19          #10 RESET=0;
20          #50 $stop;
21          end
22  endmodule
```


Msgs
/testbench/RESET
/testbench/DD
/testbench/CLK
/testbench/QQ
0
1
1
StX

```
1   `timescale 1ns/1ns
2   module mydshift (input serIn,clk,reset, output [7:0] PO);
3           wire [8:0] Q;
4           assign Q[8] = serIn;
5           genvar i;
6           generate
7               for (i=8; i > 0; i=i-1) begin
8                   myMSDFF MMSDFF(Q[i],reset,clk,Q[i-1]);
9               end
10          endgenerate
11          assign PO = Q[7:0];
12  endmodule
```

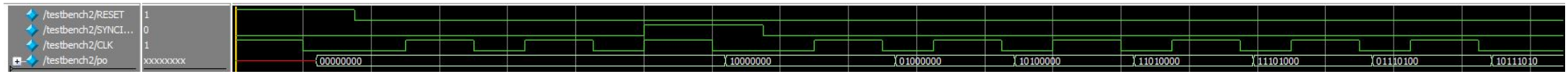
```
1   `timescale 1ns/1ns
2   module testbench ();
3           logic RESET=1,SERIN=1,CLK=1;
4           wire [7:0] po;
5           mydshift MDS(.serIn(SERIN),.reset(RESET),.clk(CLK),.PO(po));
6           initial begin
7           #100 CLK=0;
8           #50 CLK=1;
9           #50 RESET=0;CLK=1;
10          #100 CLK=0;
11          #50 CLK=1;
12          #100 CLK=0;
13          #50 SERIN=0;CLK=1;
14          #100 CLK=0;
15          #100 $stop;
16          end
17  endmodule
```


Msgs
/testbench/RESET
/testbench/SERIN
/testbench/CLK
/testbench/po
1
1
1
xxxxxxxx
00000000
10000000
11000000
01100000

```
1   `timescale 1ns/1ns
2   module dshift (input serIn,clk,reset, output logic [7:0] PO);
3           always #43 @ (negedge clk) begin
4                   if(reset)
5                           PO <= 8'b0;
6                   else
7                           PO <= {serIn, PO[7:1]};
8           end
9   endmodule
```

```
1   `timescale 1ns/1ns
2   module LSFR (input syncInit,clk,reset, output [7:0] PO);
3           wire [8:0] Q;
4           wire x,y,z;
5           xor #(23,25) X1(x,Q[0],Q[2]);
6           xor #(23,25) X2(y,x,Q[5]);
7           xor #(23,25) X3(Q[8],y,Q[6]);
8           assign Q[8] = syncInit | Q[8];
9           genvar i;
10          generate
11              for (i=8; i > 0; i=i-1) begin
12                  myMSDFF MMSDFF(Q[i],reset,clk,Q[i-1]);
13              end
14          endgenerate
15          assign PO = Q[7:0];
16  endmodule
```

```
1   `timescale 1ns/1ns
2   module testbench2 ();
3           logic RESET=1,SYNCINIT=0,CLK=1;
4           wire [7:0] po;
5           LSFR LSFRTB(.syncInit(SYNCINIT),.reset(RESET),.clk(CLK),.PO(po));
6           initial begin
7           #200 CLK=0;
8           #150 RESET=0;
9           #150 CLK=1;
10          #200 CLK=0;
11          #150 CLK=1;
12          #200 CLK=0;
13          #150 SYNCINIT=1; CLK=1;
14          #200 CLK=0;
15          #150 SYNCINIT=0;
16          #150 CLK=1;
17          #200 CLK=0;
18          #150 CLK=1;
19          #200 CLK=0;
20          #150 CLK=1;
21          #200 CLK=0;
22          #150 CLK=1;
23          #200 CLK=0;
24          #150 CLK=1;
25          #200 CLK=0;
26          #150 CLK=1;
27          #200 CLK=0;
28          #250 $stop;
29          end
30  endmodule
```


/testbench2/RESET
1
/testbench2/SYNCI...
0
/testbench2/CLK
1
/testbench2/po
xxxxxxxx
00000000
10000000
01000000
10100000
11010000
11101000
01110100
10111010