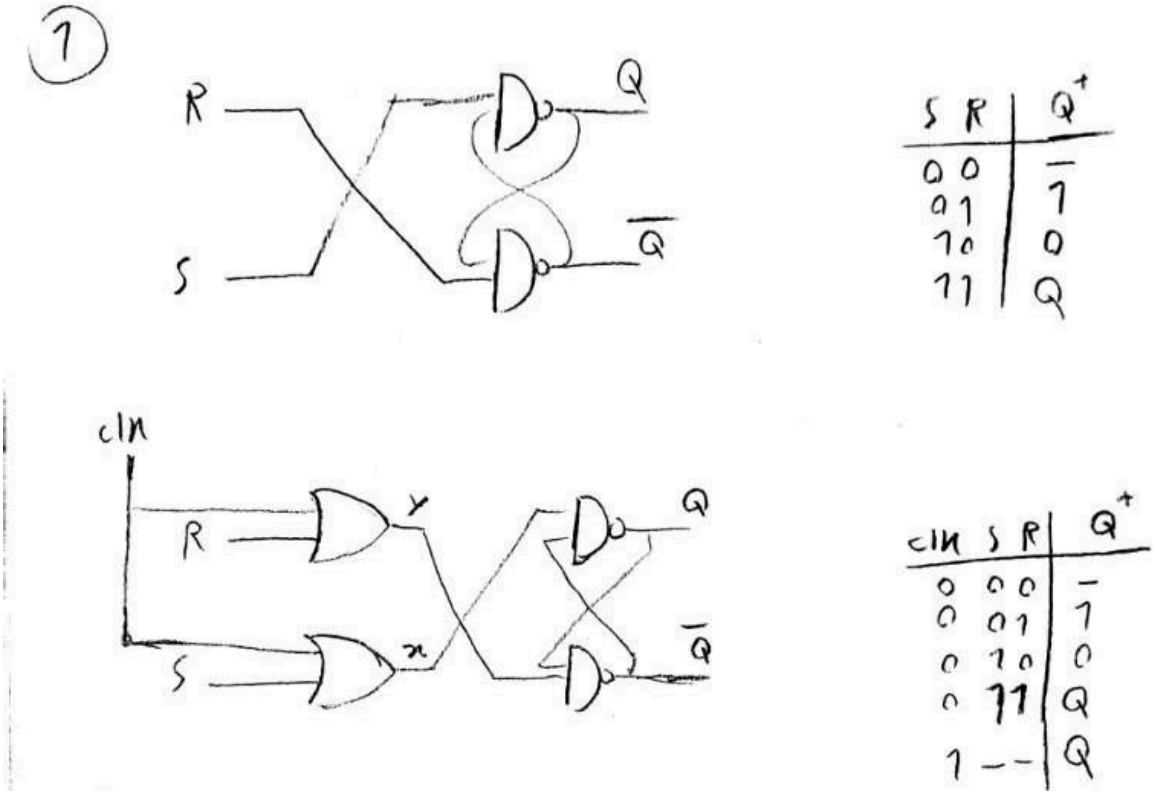
Erfan Falahati 810102491

CA3

ECE-Fall403

Course id: 4031810189401

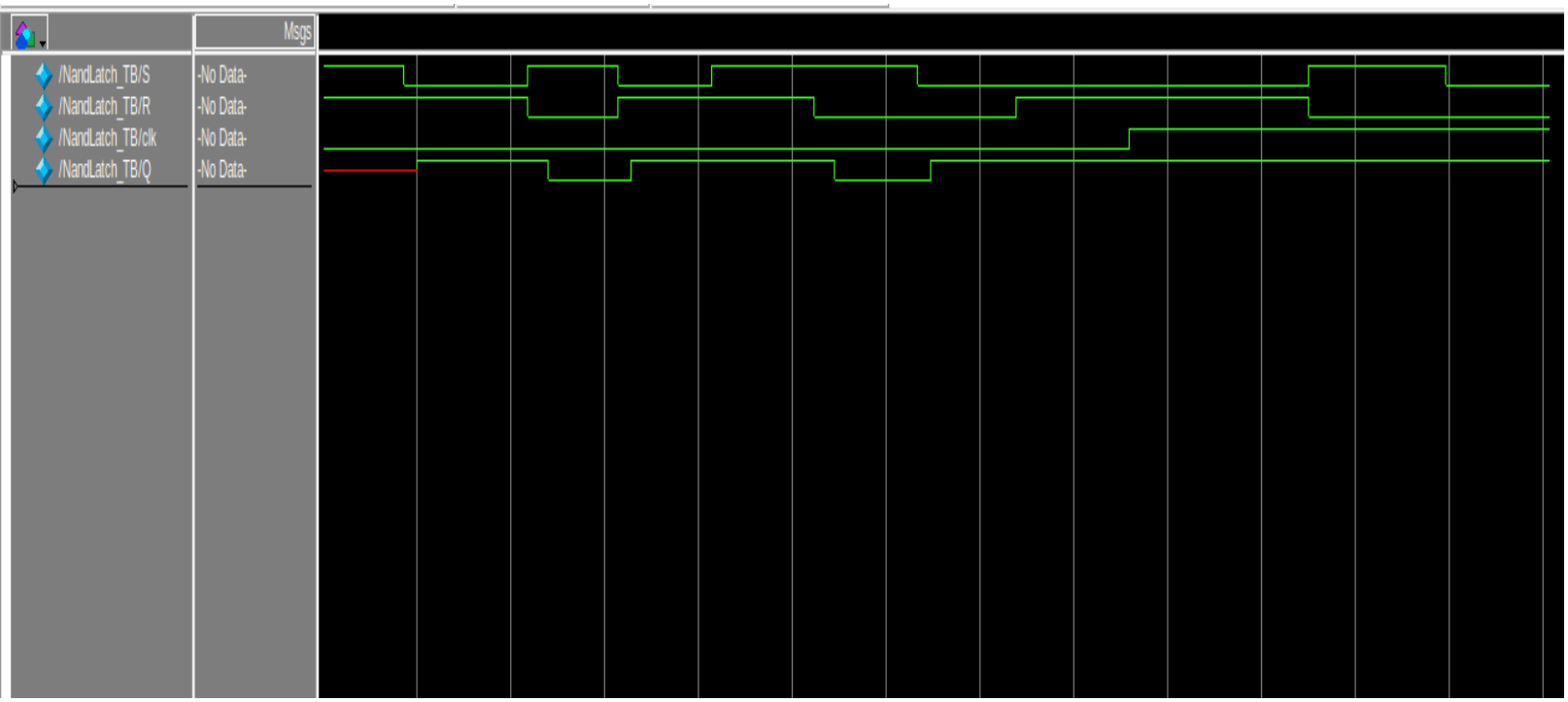
1.



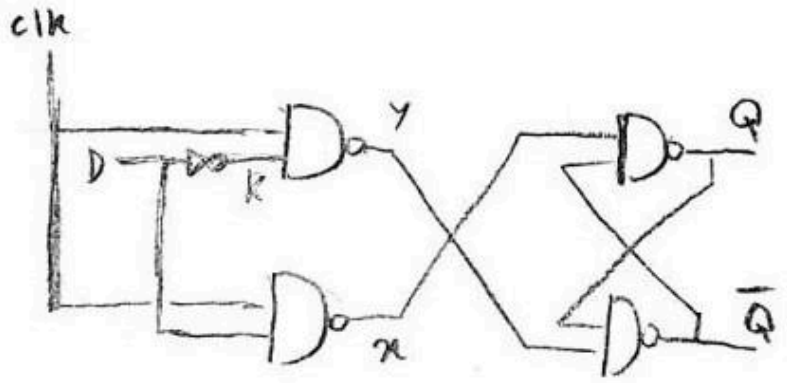
2.

```
module SRLatchNand(input clk,S,R, output Q);

    wire x,y,Qbar;

    or #(7) OR1(y, clk, R);
    or #(7) OR2(x, clk, S);
    nand #(7) N1(Q, x, Qbar);
    nand #(7) N2(Qbar, y, Q);

endmodule
```

```
module NandLatch_TB();

    logic S = 1,R = 1,clk = 0;
    wire Q;

    SRLatchNand Latch(clk, S, R, Q);

    initial begin
        #86 S = 0;
        #60
        #73 S = 1; R = 0;
        #96 S = 0; R = 1;
        #100 S = 1;
        #109 R = 0;
        #110 R = 0; S = 0;
        #56
        #49 R = 1;
        #43
        #78 clk = 1;
        #65
        #125 R = 0; S= 1;
        #90
        #57 S = 0;
        #110 $stop;
    end

endmodule
```


Msgs
/NandLatch_TB/S
/NandLatch_TB/R
/NandLatch_TB/clk
/NandLatch_TB/Q
-No Data-
-No Data-
-No Data-
-No Data-

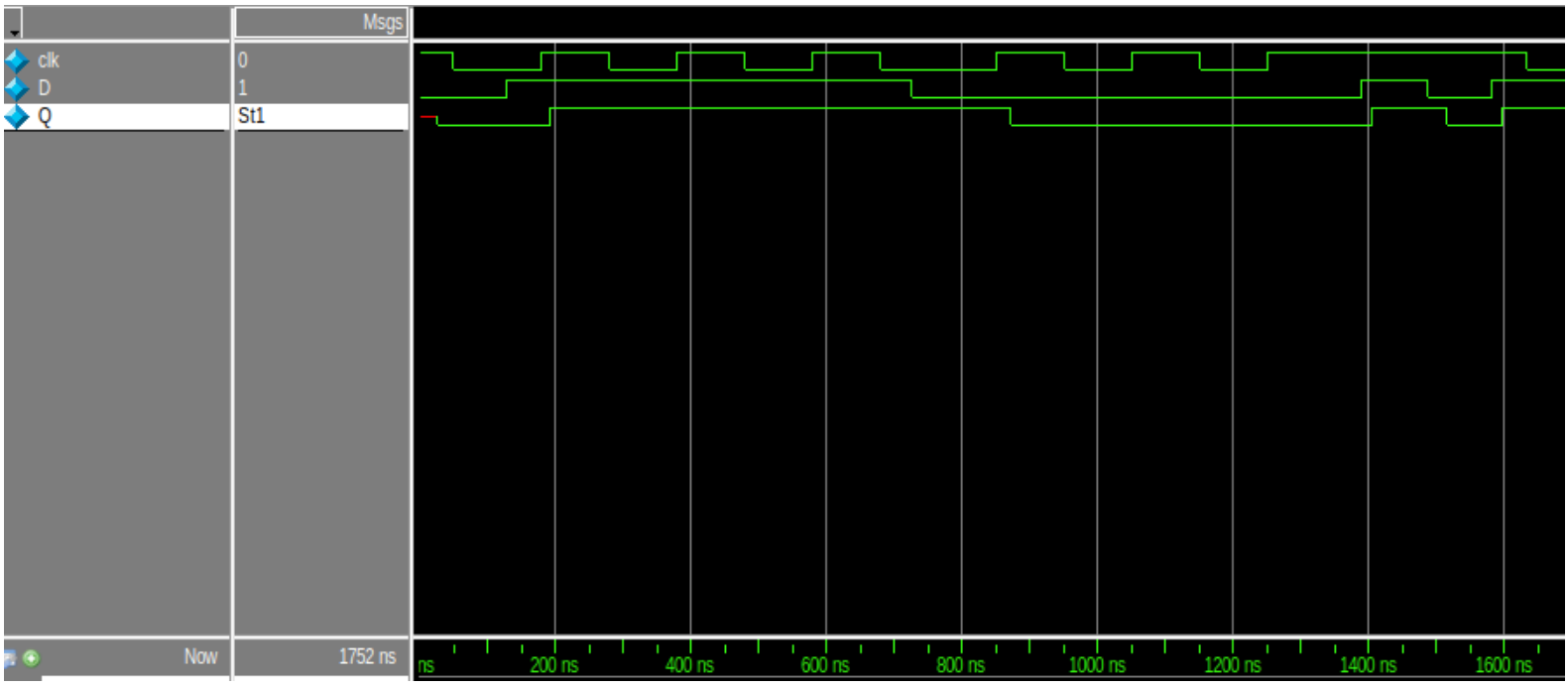
3. & 4.



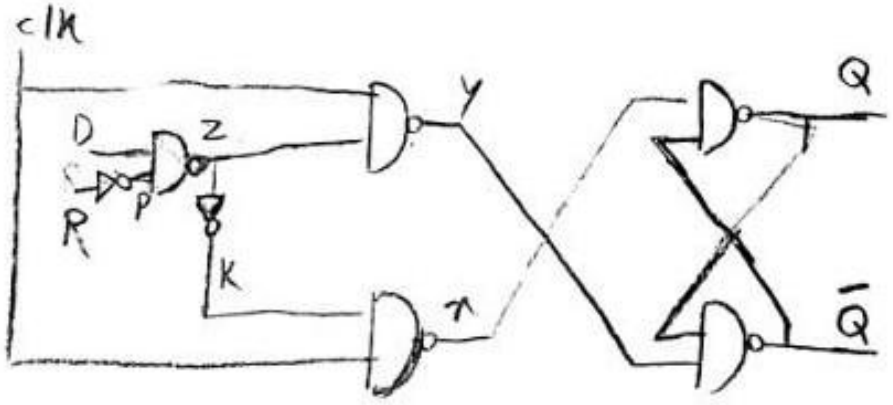
clk	D	Q^+
1	0	0
1	1	1
0	–	Q

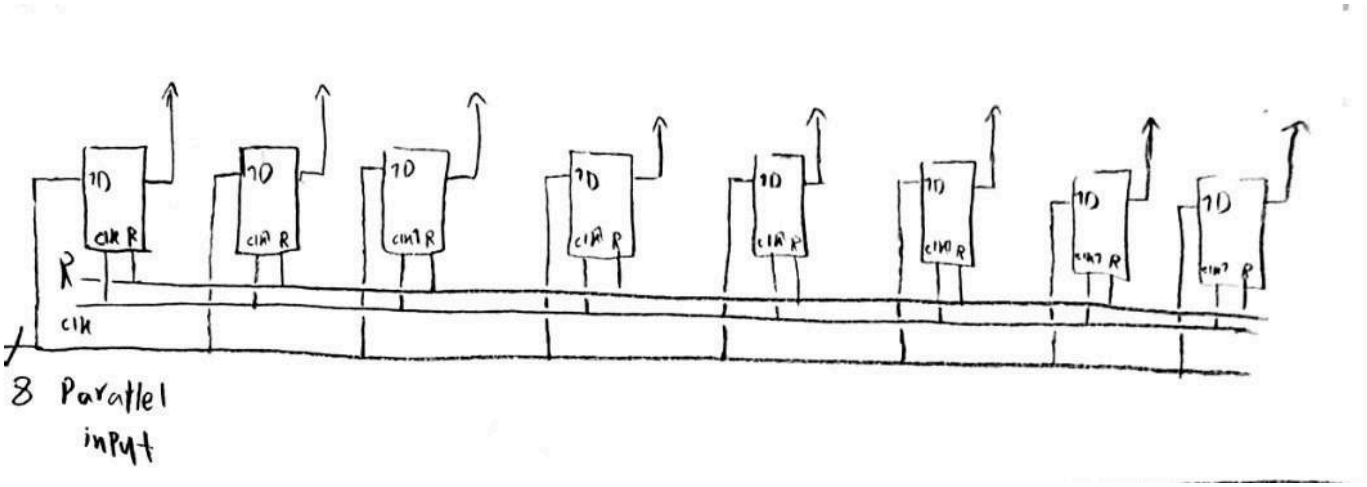
```
module DLatch1(input clk,D, output Q);

    wire k,x,y, Qbar;

    not #(7) NOT1(k,D);
    nand #(7) NAND1(y, k, clk);
    nand #(7) NAND2(x, D, clk);
    nand #(7) NAND3(Q, x, Qbar);
    nand #(7) NAND4(Qbar, y, Q);

endmodule
```

```
module Dlatch_TB();

    logic clk = 1, D = 0;
    wire Q;

    DLatch1 Latch(clk, D, Q);

    initial begin
        #50 clk = 0;
        #20
        #60 D = 1;
        #50 clk = 1;
        repeat(5) #100 clk = ~clk;
        #47 D = 0;
        #25
        repeat(5) #100 clk = ~clk;
        #42
        repeat(3) #96 D = ~D;
        #50 clk = 0;
        #120 $stop;
    end

endmodule
```



5.

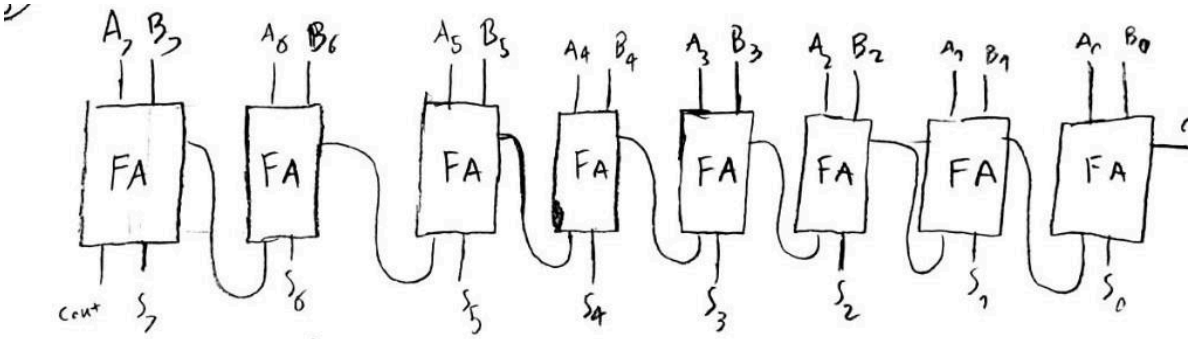


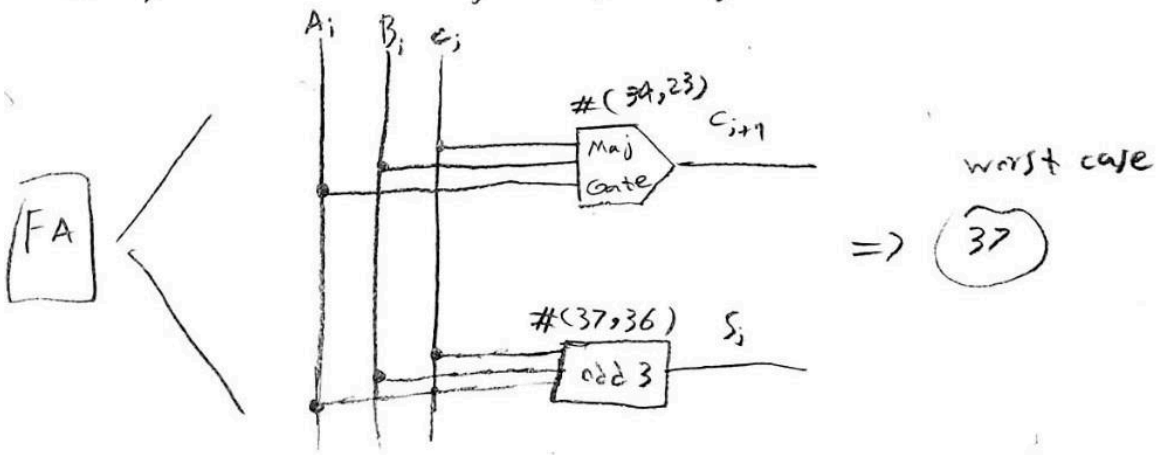


```
28  module DLatch2(input clk,R,D, output Q);
29
30      wire k,x,y,z,p, Qbar;
31
32      not #(7) NOT1(p,R);
33      not #(7) NOT2(k,z);
34      nand #(7) NAND1(z,D,p);
35      nand #(7) NAND2(y, z, clk);
36      nand #(7) NAND3(x, k, clk);
37      nand #(7) NAND4(Q, x, Qbar);
38      nand #(7) NAND5(Qbar, y, Q);
39
40  endmodule
```

```
4   module Register8Bit(input clk,R, input [7:0] PI,output [7:0] PO);
5
6       genvar k;
7       generate
8           for (k = 0; k<8; i=i+1) begin : dlatch
9               DLatch2 dd (clk, R, PI[k], PO[k]);
10          end
11      endgenerate
12
13  endmodule
14
```

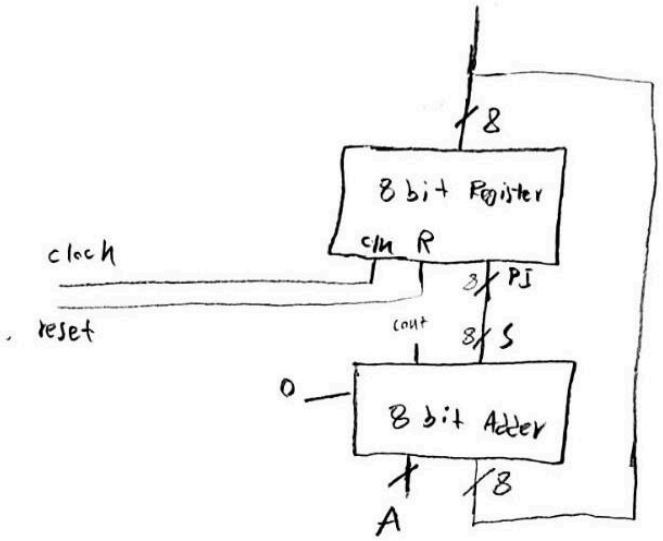
6.





8x37 = 296 ns —, worst case

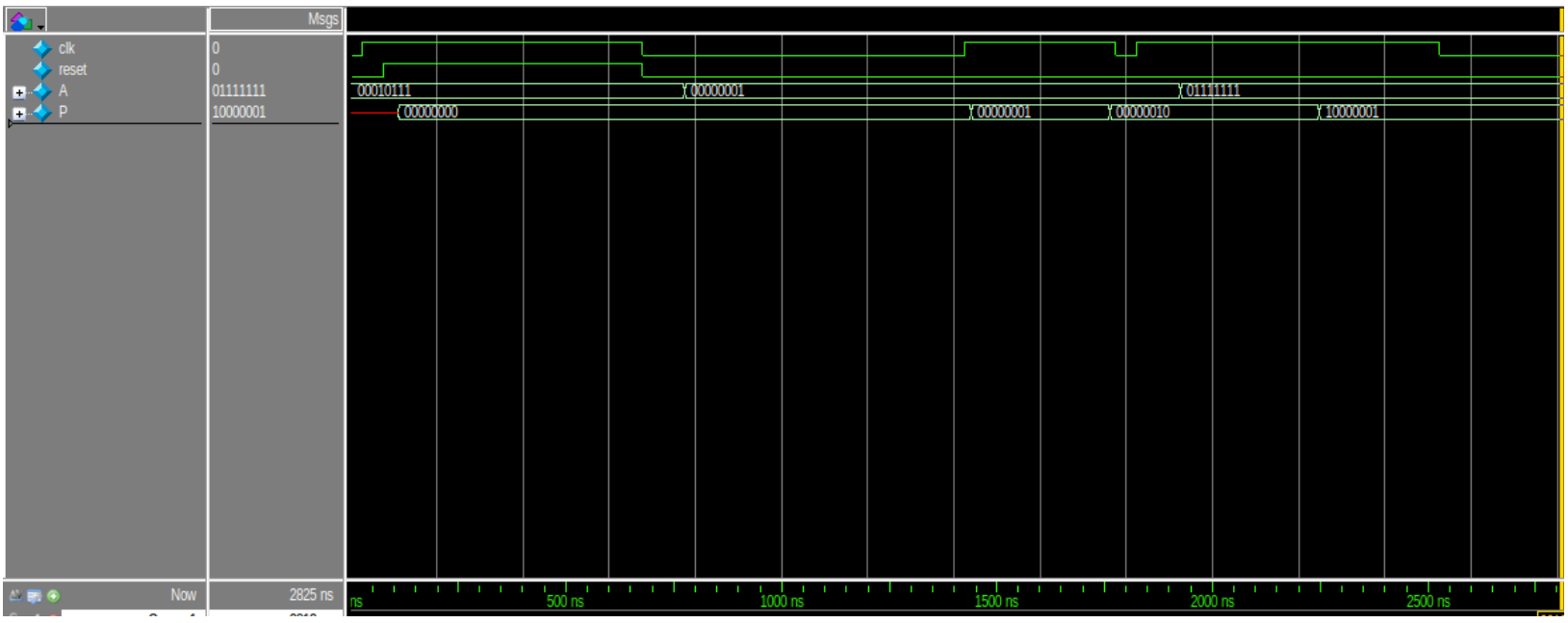
sequential adder with D-latch



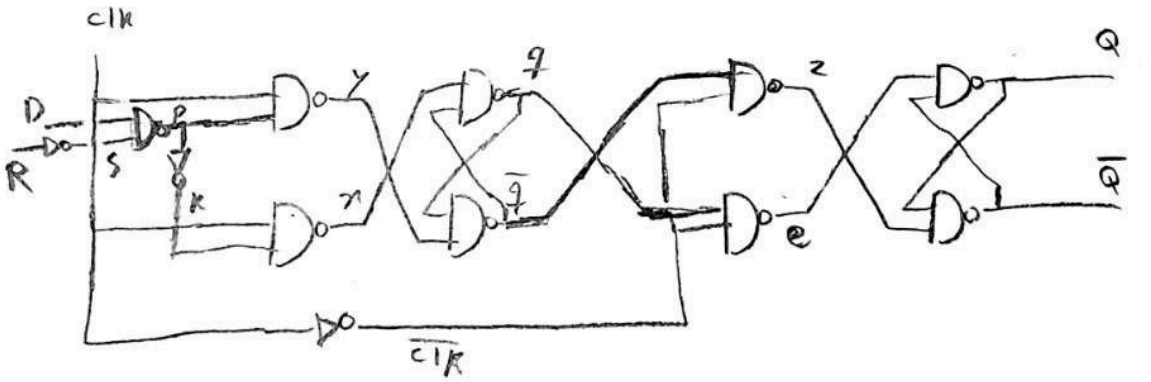
```
5  module Add8Bit(input [7:0]A,B,input cin, output [7:0] S, output cout);
6
7   assign #(296) {cout,S} = A + B + cin;
8
9  endmodule
0
1
2  module SequenceAdder(input clock, reset, input [7:0] A, output [7:0] P);
3
4      wire [7:0] PI;
5      wire cout;
6      Add8Bit ADD(A ,P ,0 ,PI, cout);
7      Register8Bit Rei(clock,reset,PI,P);
8
9  endmodule
```

7.

```
4   module seqAdder_TB();
5
6       logic clk = 0,reset = 0;
7       logic [7:0] A = 8'b00010111;
8       wire [7:0] P;
9
10      SequenceAdder SA(clk,reset, A, P);
11
12      initial begin
13          #25 clk = 1;
14          #50 reset = 1;
15          #600 reset = 0; clk = 0;
16          #100 A = 8'b00000001;
17          #300
18          #350 clk = 1;
19          #350 clk = 0;
20          #50  clk = 1;
21          #100 A = 8'b01111111;
22          #600 clk = 0;
23          #300 $stop;
24      end
25
26  endmodule
```



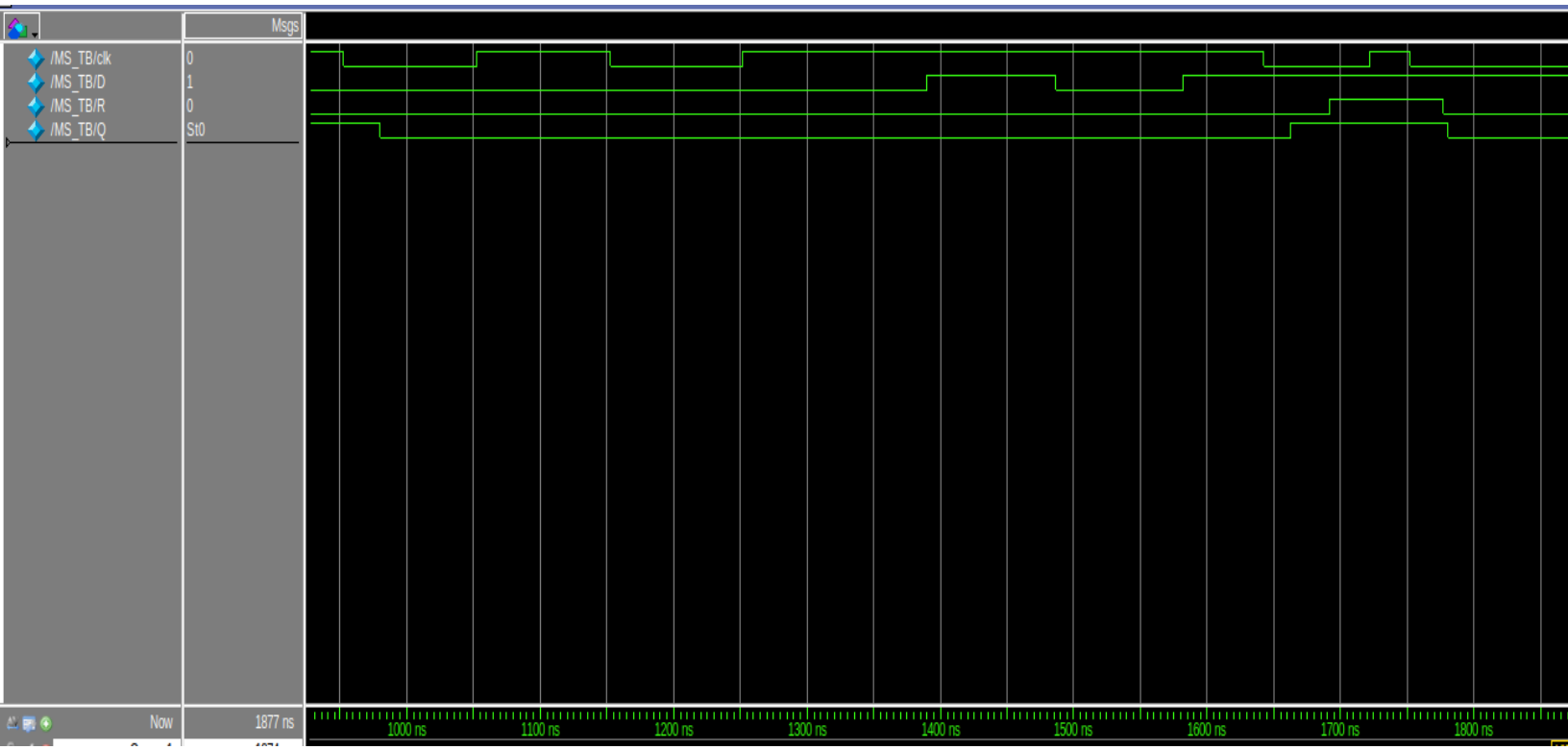
8.



Master-slave D Flip-Flop

```
46  module MSDFF(input clk,R,D, output Q);
47
48      wire s,p,k,y,x,q,qbar,clkbar,z,e,Qbar;
49      not #(7) NOT1(s, R);
50      not #(7) NOT2(clkbar, clk);
51      not #(7) NOT3(k, p);
52      nand #(7) NAND0(p, D, s);
53      nand #(7) NAND1(y, clk, p);
54      nand #(7) NAND2(x, clk, k);
55      nand #(7) NAND3(q, x, qbar);
56      nand #(7) NAND4(qbar, y, q);
57      nand #(7) NAND5(z, qbar, clkbar);
58      nand #(7) NAND6(e, q, clkbar);
59      nand #(7) NAND7(Q, e, Qbar);
60      nand #(7) NAND8(Qbar, z, Q);
61
62  endmodule
```

```
3   module MS_TB();
4
5       logic clk = 0, D = 0, R = 0;
6       wire Q;
7
8       MSDFF MSD(clk, R, D, Q);
9
10      initial begin
11          #50 clk = 0;
12          #20
13          #60 D = 1;
14          #50 clk = 1;
15          repeat(5) #100 clk = ~clk;
16          #47 D = 0;
17          #25
18          repeat(5) #100 clk = ~clk;
19          #42
20          repeat(3) #96 D = ~D;
21          #40 D = 1;
22          #20 clk = 0;
23          #50 R = 1;
24          #30 clk = 1; #30 clk = 0;
25          #25 R = 0;
26          #100 $stop;
27
28      end
29
30  endmodule
```

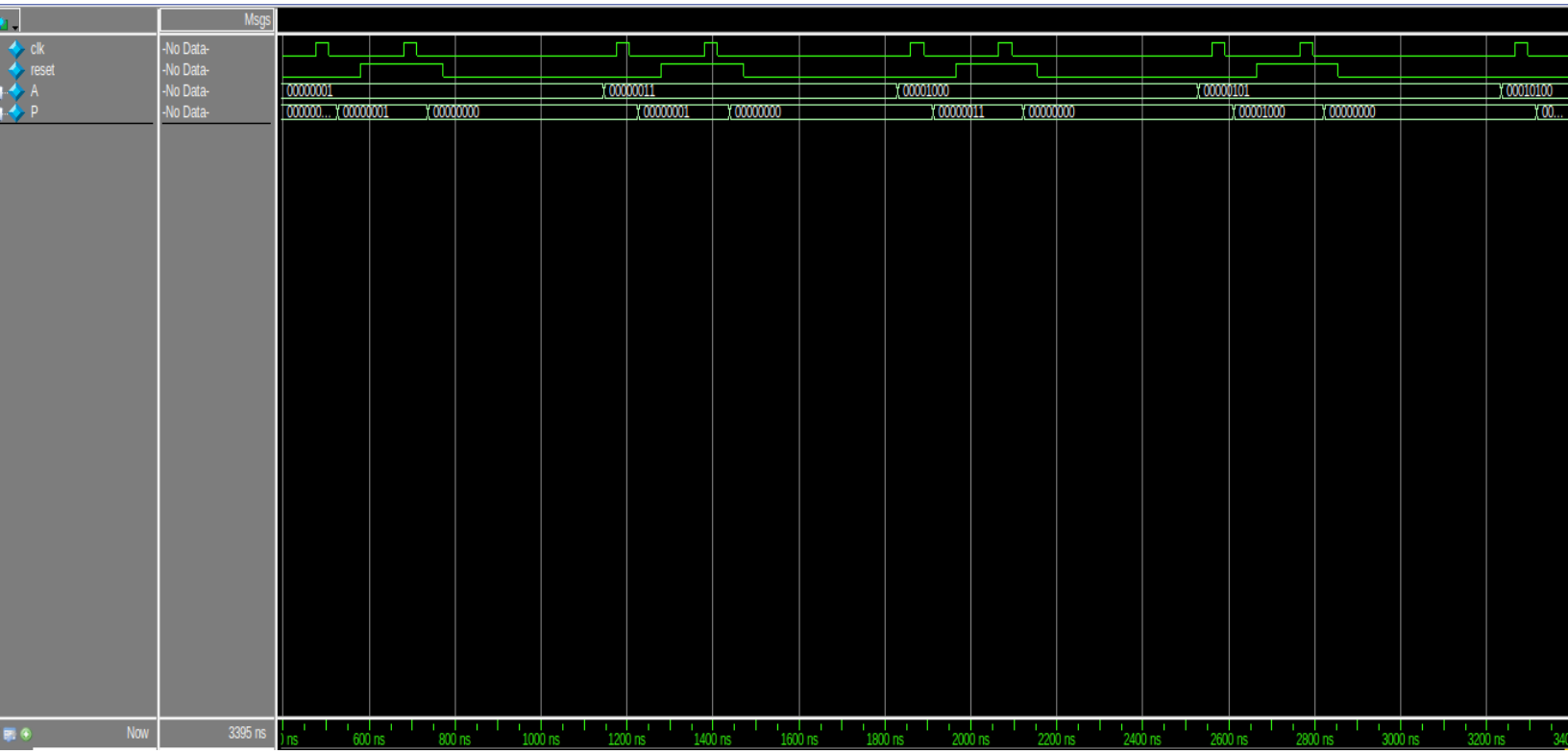


9.

```
16  module Register8Bit_MSDFF(input clk,R, input [7:0] PI,output [7:0] PO);
17
18      genvar k;
19      generate
20          for (k = 0; k<8; k=k+1) begin : ms_dff
21              MSDFF msdff (clk, R, PI[k], PO[k]);
22          end
23      endgenerate
24
25  endmodule
```

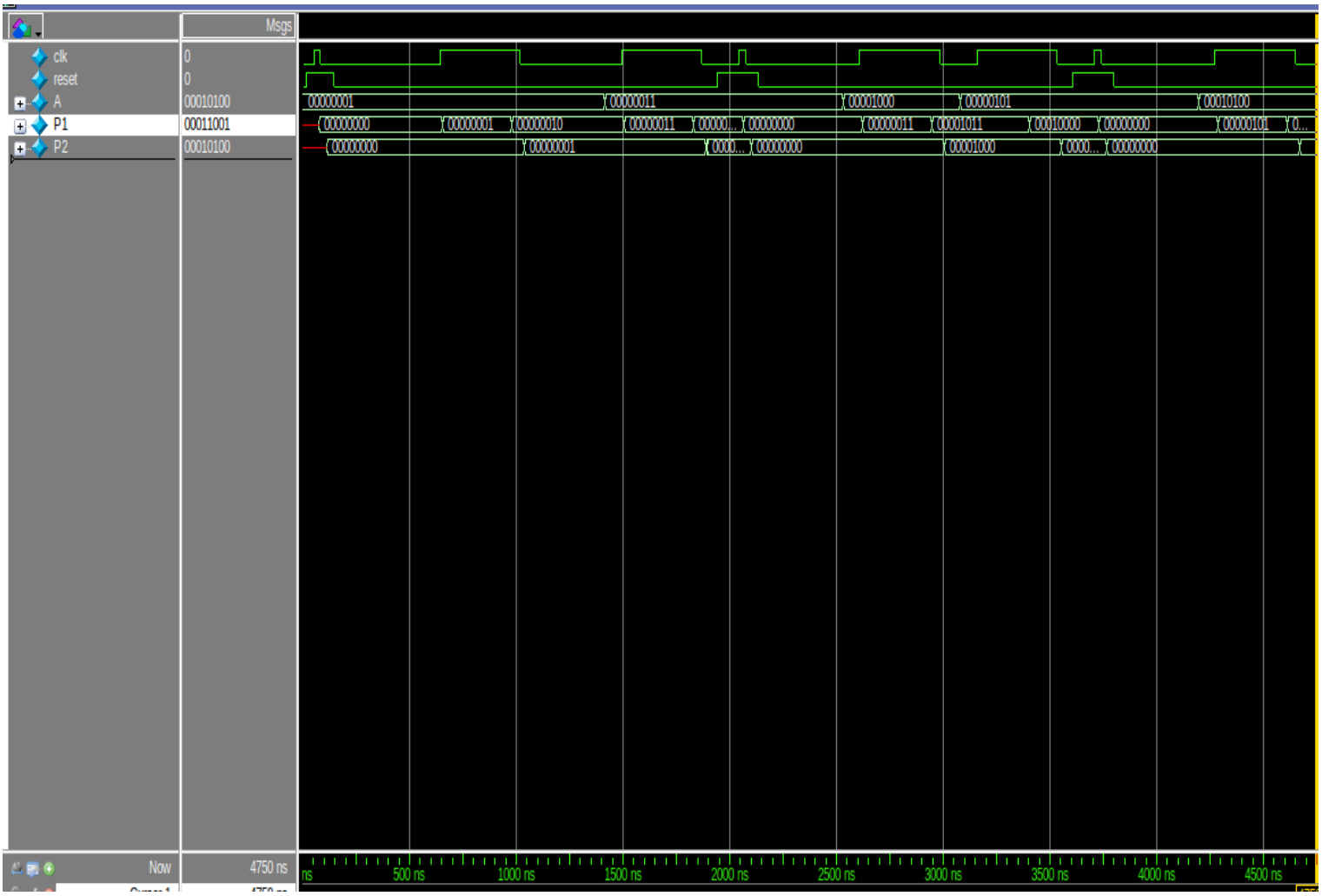
```
21  module SequenceAdder_MSDFF(input clock, reset, input [7:0] A, output [7:0] P);
22
23      wire [7:0] PI;
24      wire cout;
25      Add8Bit ADD(A ,P ,0 ,PI, cout);
26      Register8Bit_MSDFF Rei(clock,reset,PI,P);
27
28  endmodule
```

```
module seqAdder_MSDFF_TB();

    logic clk = 0, reset = 0;
    logic [7:0] A = 8'b00000001;
    wire [7:0] P;

    SequenceAdder_MSDFF SA(clk, reset, A, P);

    initial begin
        #20 reset = 1; #35 clk = 1; #30 clk = 0; #60 reset = 0;
        #300
        #30 clk = 1; #30 clk = 0;
        #75 reset = 1; #100 clk = 1; #30 clk = 0;#60 reset = 0;
        #375 A = 8'b00000011;
        #30 clk = 1; #30 clk = 0;
        #75 reset = 1; #100 clk = 1; #30 clk = 0;#60 reset = 0;
        #360 A = 8'b00001000;
        #30 clk = 1; #30 clk = 0;
        #75 reset = 1; #100 clk = 1; #30 clk = 0;#60 reset = 0;
        #375 A = 8'b00000101;
        #30 clk = 1; #30 clk = 0;
        #75 reset = 1; #100 clk = 1; #30 clk = 0;#60 reset = 0;
        #380 A = 8'b00010100;
        #30 clk = 1; #30 clk = 0;
        #100 $stop;
    end

endmodule
```


Msgs
clk
reset
A
P
-No Data-
-No Data-
-No Data-
-No Data-
00000001
00000011
00001000
00000101
00010100
000000...
00000001
00000000
00000001
00000000
00000011
00000000
00001000
00000000
00...
Now
3395 ns
1 ns
600 ns
800 ns
1000 ns
1200 ns
1400 ns
1600 ns
1800 ns
2000 ns
2200 ns
2400 ns
2600 ns
2800 ns
3000 ns
3200 ns

10.

```
61  module MS_VS_LATCH_TB();
62
63      logic clk = 0, reset = 0;
64      logic [7:0] A = 8'b00000001;
65      wire [7:0] P1,P2;
66
67      SequenceAdder SA1(clk,reset, A, P1);
68      SequenceAdder_MSDFF SA2(clk, reset, A, P2);
69
70      initial begin
71          #20 reset = 1; #35 clk = 1; #30 clk = 0; #60 reset = 0;
72          #400
73          #100 clk = 1; #375 clk = 0;
74          #400 A = 8'b00000011;
75          #75 clk = 1; #375 clk = 0;
76          #75 reset = 1; #100 clk = 1; #30 clk = 0;#60 reset = 0;
77          #400 A = 8'b00001000;
78          #75 clk = 1; #375 clk = 0;
79          #100 A = 8'b00000101;
80          #75 clk = 1; #375 clk = 0;
81          #75 reset = 1; #100 clk = 1; #30 clk = 0;#60 reset = 0;
82          #400 A = 8'b00010100;
83          #75 clk = 1; #375 clk = 0;
84          #100 $stop;
85      end
86
87
88  endmodule
```


Msgs
clk
reset
A
P1
P2
0
0
00010100
00011001
00010100
00000001
00000011
00001000
00000101
00010100
00000000
00000001
00000010
00000011
00000011
00001011
00010000
00000101
Now
4750 ns
ns
500 ns
1000 ns
1500 ns
2000 ns
2500 ns
3000 ns
3500 ns
4000 ns
4500 ns