

Assignment 3  
By Amr Mohamed Ahmed Ebrahim

## Question 1

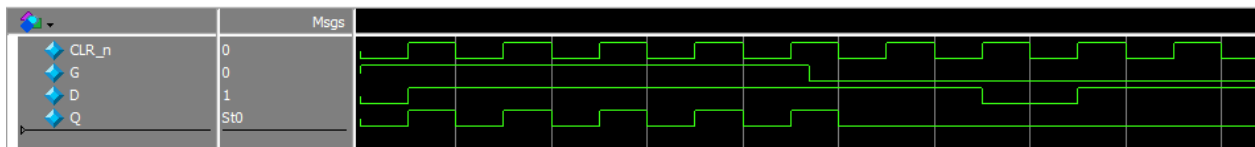
### latch Code

```
1  `timescale 1ns/1ps
2  module latch
3  (
4      input CLR_n, D, G,
5      output reg Q
6  );
7
8      always @(*)
9          if(~CLR_n)
10             Q <= 1'b0;
11          else if(G)
12             Q <= D ;
13
14  endmodule
```

### Testbench Code

```
16 ▼ module latch_tb ();
17
18     reg CLR_n, D, G ;
19     wire Q ;
20     latch latch (CLR_n, D, G, Q) ;
21
22 ▼ initial begin
23     CLR_n=1'b0;
24     forever
25         #10 CLR_n=~CLR_n ;
26     end
27
28 ▼ initial begin
29     G=1'b1;
30 ▼ repeat(5) begin
31     D=$random ;
32     @(posedge CLR_n) ;
33     end
34     #4 G=1'b0;
35 ▼ repeat(5) begin
36     D=$random ;
37     @(posedge CLR_n) ;
38     end
39     $stop ;
40
41     end
42 endmodule
```

**TestBench Wave**



## Question 2 (A)

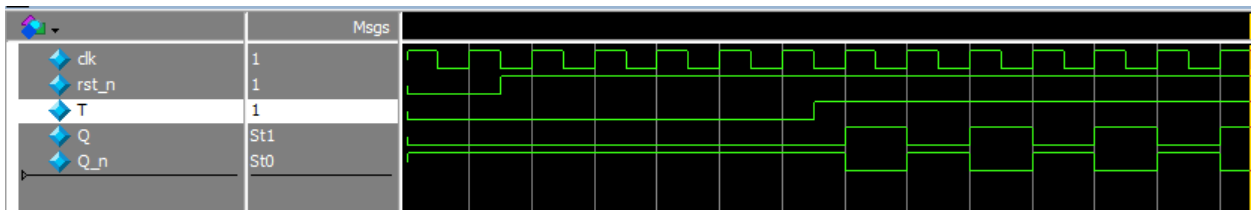
### T Flipflop Code

```
1  `timescale 1ns/1ps
2  module T_FF (
3      input T, clk, rst_n,
4      output Q, Q_n
5  );
6
7      reg Q_reg, Q_next ;
8
9      always @(posedge clk or negedge rst_n)
10         if(~rst_n)
11             Q_reg <= 1'b0;
12         else
13             Q_reg <= Q_next ;
14
15         assign Q_next = T ? ~Q_reg: Q_reg ;
16         assign Q = Q_reg;
17         assign Q_n = ~Q_reg;
18
19     endmodule
```

### T Flipflop Testbench Code

```
21 ▼ module T_FF_tb ();
22
23     reg T, clk, rst_n ;
24     wire Q, Q_n ;
25
26     T_FF T_FF (T, clk, rst_n, Q, Q_n) ;
27
28 ▼     initial begin
29         clk=1'b1;
30         forever
31             #10 clk=~clk;
32     end
33
34 ▼     initial begin
35         rst_n=0;
36         T=0;
37         repeat(2) @(negedge clk) ;
38         rst_n=1;
39         repeat(5) @(negedge clk) ;
40         T=1;
41         repeat(7) @(negedge clk) ;
42         $stop;
43     end
44
45     endmodule
```

**T Flipflop Testbench Wave**



## Question 2 (B)

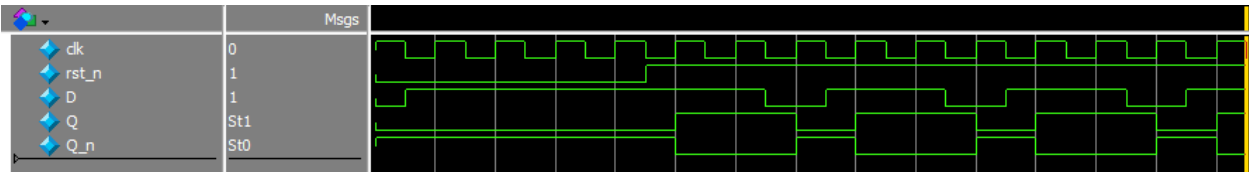
### D Flipflop Code

```
1  `timescale 1ns/1ps
2  module D_FF (
3      input D, clk, rst_n,
4      output Q, Q_n
5  );
6
7      reg Q_reg, Q_next;
8
9      always@(posedge clk or negedge rst_n)
10         if(~rst_n)
11             Q_reg<=0;
12         else
13             Q_reg<=Q_next;
14
15         assign Q_next = D;
16         assign Q = Q_reg;
17         assign Q_n = ~Q_reg;
18
19     endmodule
```

### D Flipflop Testbench Code

```
21 module D_FF_tb () ;
22
23     reg D, clk, rst_n ;
24     wire Q, Q_n ;
25     D_FF D_FF (D, clk, rst_n, Q, Q_n) ;
26
27     initial begin
28         clk=1;
29         forever
30             #10 clk=~clk;
31     end
32
33     initial begin
34         rst_n=0;
35         repeat(5) begin
36             D=$random;
37             @(negedge clk) ;
38         end
39         rst_n=1;
40         repeat(10) begin
41             D=$random;
42             @(negedge clk) ;
43         end
44         $stop;
45     end
46 endmodule
```

D Flipflop Testbench Wave



## Question 2 (C)

```
1  module Flipflop #(parameter FF_TYPE="DFF")
2      (
3          input D, clk, rst_n,
4          output Q, Q_n
5      );
6
7      generate
8          if(FF_TYPE=="TFF")
9              T_FF T_FF (D, clk, rst_n, Q, Q_n) ;
10         else
11             D_FF D_FF (D, clk, rst_n, Q, Q_n) ;
12         endgenerate
13     endmodule
```

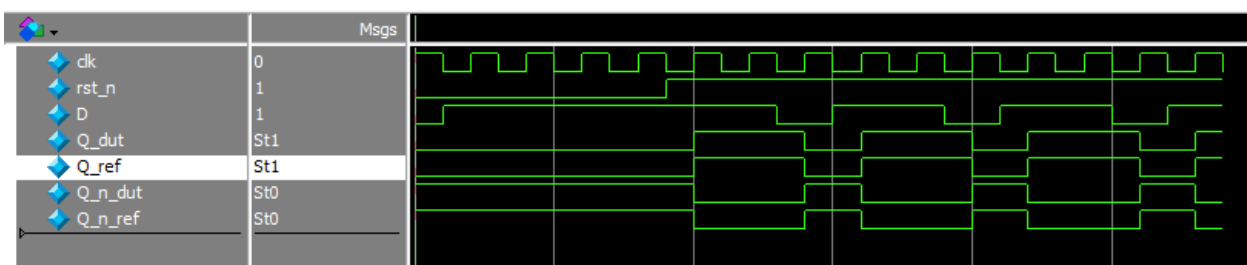


## Question 2 (D)

### D Flipflop testbench

```
1  `timescale 1ns/1ps
2  module Flipflop_tb_1 () ;
3      reg D, clk, rst_n ;
4      wire Q_dut, Q_n_dut ;
5      wire Q_ref, Q_n_ref ;
6
7      Flipflop #(.FF_TYPE("DFF")) dut (D, clk, rst_n, Q_dut, Q_n_dut) ;
8      D_FF ref (D, clk, rst_n, Q_ref, Q_n_ref) ;
9
10     initial begin
11         clk=1;
12         forever
13             #10 clk=~clk;
14     end
15
16     initial begin
17         rst_n=0;
18         repeat(5) begin
19             D=$random;
20             @(negedge clk) ;
21         end
22         rst_n=1;
23         repeat(10) begin
24             D=$random;
25             @(negedge clk) ;
26         end
27         $stop;
28     end
29
30     always@(*)
31         if(Q_dut != Q_ref) begin
32             $display("Error") ;
33             $stop;
34         end
35 endmodule
```

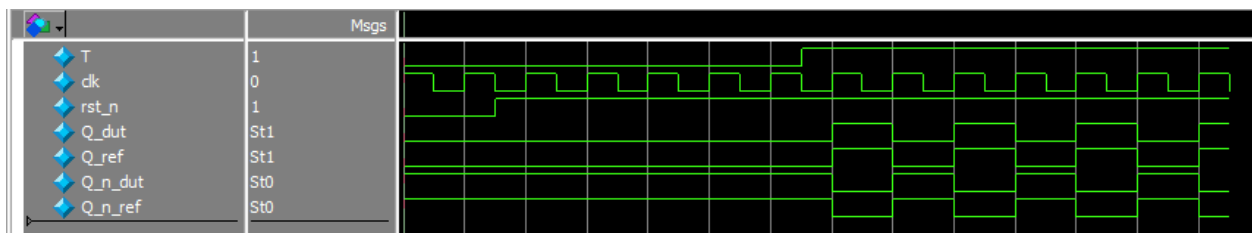
### D Flipflop testbench wave



## T Flipflop testbench

```
38 module Flipflop_tb_2 ( ) ;
39
40     reg T, clk, rst_n ;
41     wire Q_dut, Q_n_dut ;
42     wire Q_ref, Q_n_ref ;
43
44     Flipflop #(.FF_TYPE("TFF")) dut (T, clk, rst_n, Q_dut, Q_n_dut) ;
45     T_FF ref (T, clk, rst_n, Q_ref, Q_n_ref) ;
46
47     initial begin
48         clk=1'b1;
49         forever
50             #10 clk=~clk;
51     end
52
53     initial begin
54         rst_n=0;
55         T=0;
56         repeat(2) @(negedge clk) ;
57         rst_n=1;
58         repeat(5) @(negedge clk) ;
59         T=1;
60         repeat(7) @(negedge clk) ;
61         $stop;
62     end
63
64     always@(*)
65     if(Q_dut != Q_ref) begin
66         $display("Error") ;
67         $stop;
68     end
69
70 endmodule
```

## T Flipflop Testbench Wave



### Question 3

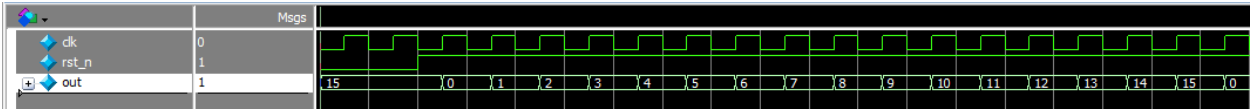
#### Design Code

```
1  `timescale 1ns/1ps
2  ▼ module ripple_counter (
3      input clk, rst_n,
4      output [3:0] out
5  ▼ );
6
7      wire [3:0] qn ;
8      wire clk1, clk2, clk3, clk4 ;
9
10     D_FF ff1 (qn[0], clk, rst_n, clk1, qn[0]);
11     D_FF ff2 (qn[1], clk1, rst_n, clk2, qn[1]);
12     D_FF ff3 (qn[2], clk2, rst_n, clk3, qn[2]);
13     D_FF ff4 (qn[3], clk3, rst_n, clk4, qn[3]);
14
15     assign out = qn;
16
17 endmodule
```

#### Testbench Code

```
20 module ripple_counter_tb ();
21     reg clk, rst_n ;
22     wire [3:0] out ;
23     ripple_counter dut (clk, rst_n, out) ;
24
25     initial begin
26         clk=0;
27         forever
28             #10 clk=~clk;
29     end
30
31     initial begin
32         rst_n=0;
33         repeat(2) @(negedge clk) ;
34         rst_n=1;
35         repeat(50) @(negedge clk) ;
36         $stop;
37     end
38 endmodule
```

Testbench Wave



## Question 4

### Design Code

```
1  ▼ module shift_register
2      #(parameter LOAD_AVALUE=1, LOAD_SVALUE=1,
3  ▼  SHIFT_DIRECTION="LEFT",SHIFT_WIDTH=8)
4  ▼  (input sclr, sset, shiftin, load,
5      input aclr, aset,clk, enable,
6      input [SHIFT_WIDTH-1:0] data,
7      output [SHIFT_WIDTH-1:0] q,
8      output reg shiftout
9  ▼  );
10     reg [SHIFT_WIDTH-1:0] Q_reg, Q_next ;
11     reg temp;
12
13 ▼  always @(posedge clk or posedge aclr or posedge aset)
14 ▼      if(aclr)
15          Q_reg<=0;
16 ▼      else if (aset)
17          Q_reg<=LOAD_AVALUE;
18 ▼      else begin
19          Q_reg<=Q_next;
20          shiftout=temp;
21      end
22
23 ▼  always @(*)
24 ▼      if(enable)
25 ▼          if(sclr)
26              Q_next=0;
27 ▼          else if(sset)
28              Q_next=LOAD_SVALUE ;
29 ▼          else if(load)
30              Q_next = data;
31 ▼          else if(!load && SHIFT_DIRECTION=="LEFT")
32              {temp,Q_next}={Q_reg,shiftin};
33 ▼          else if(!load && SHIFT_DIRECTION=="RIGHT")
34              {Q_next,temp}={shiftin,Q_reg};
35 ▼          else
36              Q_next=Q_reg;
37 ▼          else
38              Q_next=Q_reg;
39
40     assign q = Q_reg;
41 endmodule
```

## Testbench Code

```
45 ▼ module shift_register_tb #(parameter LOAD_AVALUE=1, LOAD_SVALUE=1,
46 ▼   SHIFT_DIRECTION="LEFT",SHIFT_WIDTH=8) ();
47     reg sclr, sset, shiftin, load ;
48     reg aclr, aset,clk, enable ;
49     reg [SHIFT_WIDTH-1:0] data ;
50     wire [SHIFT_WIDTH-1:0] q ;
51     wire shiftout ;
52     shift_register #(LOAD_AVALUE,LOAD_SVALUE,SHIFT_DIRECTION,SHIFT_WIDTH)
53     dut (sclr, sset, shiftin, load, aclr, aset,clk, enable, data, q, shiftout);
54 ▼   initial begin
55       clk=0;
56 ▼     forever
57       #10 clk=~clk;
58   end
59 ▼   initial begin
60       data=8'b1100_1010;
61       {sclr,sset,shiftin,load,enable}=0;
62 ▼     aset=0;
63       aclr=1;
64 ▼     @(negedge clk);
65       aset=1;
66       aclr=0;
67       @(negedge clk);
68       aset=1;
69       aclr=1;
70       @(negedge clk);
71       aset=0;
72 ▼     aclr=0;
73       @(negedge clk);
74       sset=0;
75 ▼     sclr=1;
76       @(negedge clk);
77       enable=1;
78       sset=1;
79       sclr=0;
80       @(negedge clk);
81       sset=1;
82 ▼     sclr=1;
83       @(negedge clk);
84       sset=0;
85       sclr=0;
86       load=1;
87       @(negedge clk);
88       load=0;
89       repeat(10) @(negedge clk);
90       $stop;
91   end
92 endmodule
```

Testbench Wave



## Question 5

### Design Code

```
1  module SLE (
2      input D,CLK,EN,
3      input ALn, ADn,
4      input SLn, SD,
5      input LAT,
6      output Q
7  );
8
9      reg Q_reg_ff,Q_reg_latch;
10     reg Q_next ;
11
12     always @(posedge CLK or negedge ALn)
13     if(~ALn)
14         Q_reg_ff <= ~ADn;
15     else
16         Q_reg_ff <= Q_next;
17
18     always @(*)
19     if(~ALn)
20         Q_reg_latch<=~ADn;
21     else if(CLK)
22         Q_reg_latch<=Q_next;
23
24     always @(*)
25     if(EN)
26         if(~SLn)
27             Q_next=SD;
28         else
29             Q_next=D;
30     else
31         Q_next= (LAT) ? Q_reg_latch:Q_reg_ff;
32
33     assign Q= (LAT) ? Q_reg_latch:Q_reg_ff;
34
35 endmodule
36
```



## Testbench Code

```
37 `timescale 1ns/1ps
38 module SLE_tb();
39
40     reg D,CLK,EN;
41     reg ALn, ADn;
42     reg SLn, SD;
43     reg LAT;
44     wire Q;
45     SLE dut (D, CLK, EN, ALn, ADn, SLn, SD, LAT, Q) ;
46
47     initial begin
48         CLK=0;
49         forever
50             #10 CLK=~CLK;
51     end
52
53     initial begin
54         D=0;
55         ALn=0;
56         ADn=0;
57         EN=0;
58         SLn=0;
59         SD=1;
60         LAT=0;
61         repeat(2) @(negedge CLK);
62         ALn=1;
63         @(negedge CLK);
64         EN=1;
65         @(negedge CLK);
66         SLn=1;
67         repeat(2) @(negedge CLK);
68         LAT=1;
69         @(negedge CLK);
70         D=0;
71         ALn=0;
72         EN=0;
73         SLn=0;
74         @(negedge CLK);
75         ALn=1;
76         @(negedge CLK);
77         EN=1;
78         @(negedge CLK);
79         SLn=1;
80         repeat(2) @(negedge CLK);
81
82         $stop;
83     end
84 endmodule
```

Simulation Wave

