Assignment 3

By Amr Mohamed Ahmed Ebrahim

Question 1

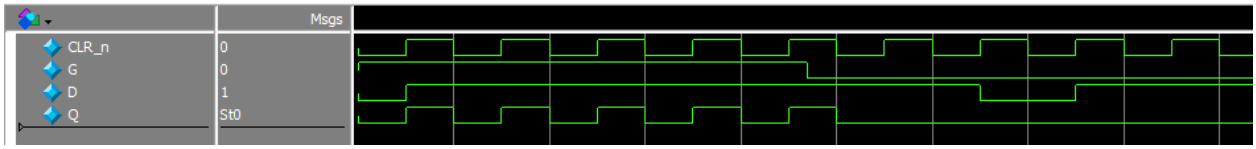
latch Code

```
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)

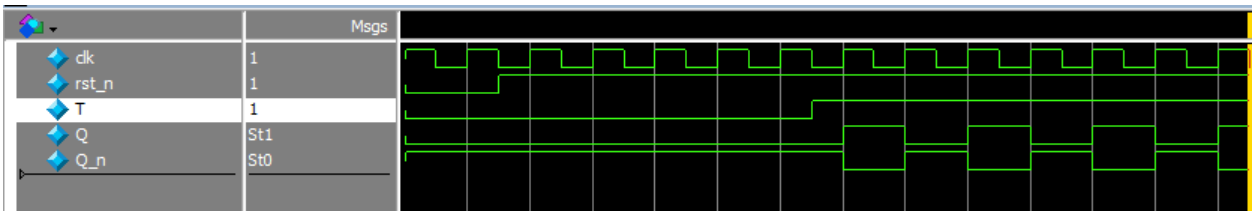
T Flipflop Code

```
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)

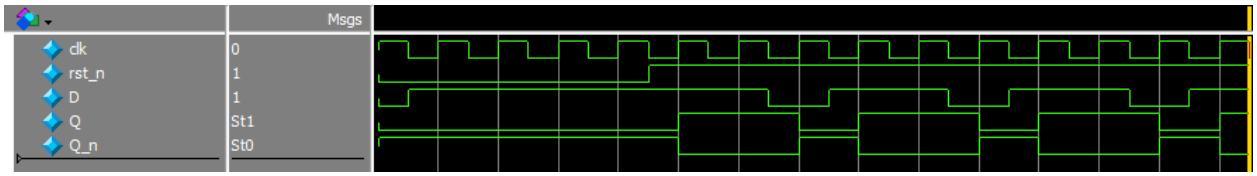
D Flipflop Code

```
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)

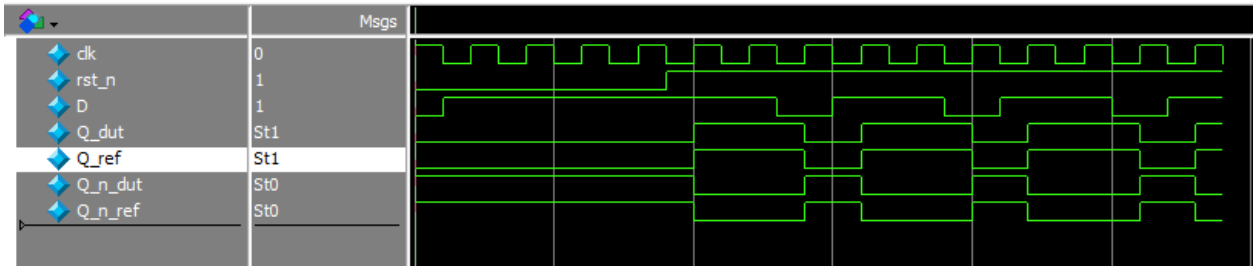
```
1   module Flipflop #(parameter FF_TYPE="DFF")
2       (
3       input D, clk, rst_n,
4       output Q, Q_n
5       );
6
7       generate
8           if(FF_TYPE=="TFF")
9             T_FF T_FF (D, clk, rst_n, Q, Q_n) ;
10          else
11            D_FF D_FF (D, clk, rst_n, Q, Q_n) ;
12
13      endgenerate
14  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
```

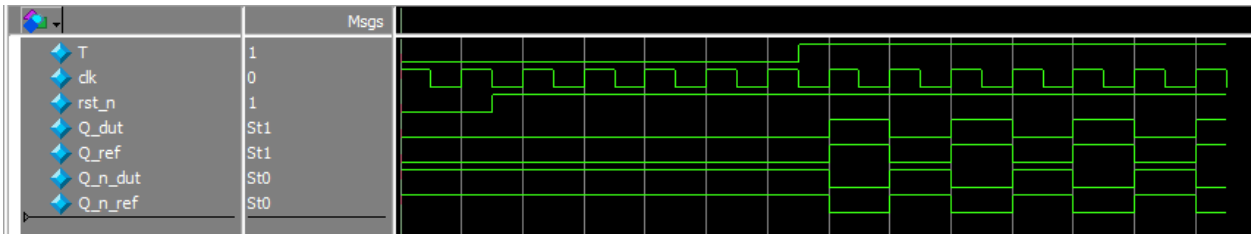
D Flipflop testbench wave



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

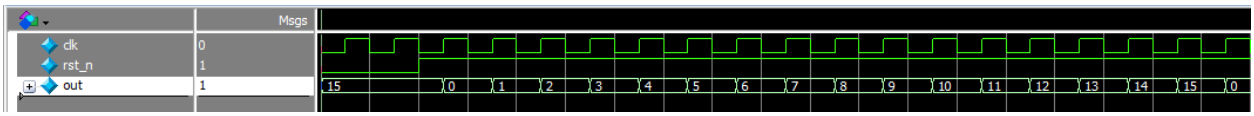
Design Code

```
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

```
module shift_register
  #(parameter LOAD_AVALUE=1, LOAD_SVALUE=1,
 SHIFT_DIRECTION="LEFT",SHIFT_WIDTH=8)
  (input sclr, sset, shiftin, load,
   input aclr, aset,clk, enable,
   input [SHIFT_WIDTH-1:0] data,
   output [SHIFT_WIDTH-1:0] q,
   output reg shiftout
   );
  reg [SHIFT_WIDTH-1:0] Q_reg, Q_next ;
  reg temp;

  always @(posedge clk or posedge aclr or posedge aset)
    if(aclr)
      Q_reg<=0;
    else if (aset)
      Q_reg<=LOAD_AVALUE;
    else begin
      Q_reg<=Q_next;
      shiftout=temp;
    end

   always @(*)
     if(enable)
       if(sclr)
         Q_next=0;
       else if(sset)
         Q_next=LOAD_SVALUE ;
       else if(load)
         Q_next = data;
       else if(!load && SHIFT_DIRECTION=="LEFT")
         {temp,Q_next}={Q_reg,shiftin};
       else if(!load && SHIFT_DIRECTION=="RIGHT")
         {Q_next,temp}={shiftin,Q_reg};
       else
         Q_next=Q_reg;
     else
       Q_next=Q_reg;

   assign q = Q_reg;
endmodule
```

Testbench Code

```
module shift_register_tb #(parameter LOAD_AVALUE=1, LOAD_SVALUE=1,
  SHIFT_DIRECTION="LEFT",SHIFT_WIDTH=8) ();
    reg sclr, sset, shiftin, load ;
    reg aclr, aset,clk, enable ;
    reg [SHIFT_WIDTH-1:0] data ;
    wire [SHIFT_WIDTH-1:0] q ;
    wire shiftout ;
    shift_register #(LOAD_AVALUE,LOAD_SVALUE,SHIFT_DIRECTION,SHIFT_WIDTH)
    dut (sclr, sset, shiftin, load, aclr, aset,clk, enable, data, q, shiftout);
    initial begin
      clk=0;
      forever
        #10 clk=~clk;
    end
    initial begin
        data=8'b1100_1010;
      {sclr,sset,shiftin,load,enable}=0;
      aset=0;
        aclr=1;
      @(negedge clk);
        aset=1;
        aclr=0;
      @(negedge clk);
      aset=1;
      aclr=1;
      @(negedge clk);
      aset=0;
      aclr=0;
        @(negedge clk);
      sset=0;
      sclr=1;
        @(negedge clk);
        enable=1;
      sset=1;
      sclr=0;
      @(negedge clk);
      sset=1;
      sclr=1;
        @(negedge clk);
      sset=0;
      sclr=0;
      load=1;
      @(negedge clk);
      load=0;
      repeat(10) @(negedge clk);
      $stop;
    end
endmodule
```

Testbench Wave

Question 5

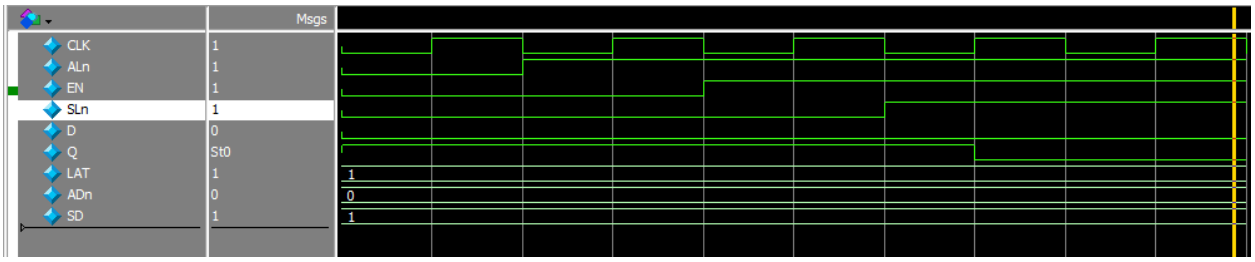
Design Code

```
module SLE #(parameter LAT=0, ADn=0, SD=1)(
  input D,CLK,EN,
  input ALn,
  input SLn,
  output Q
  );

      reg Q_reg, Q_next ;
      generate
        if(~LAT) begin
          always @(posedge CLK or negedge ALn)
            if(~ALn)
              Q_reg <= ~ADn;
            else
              Q_reg <= Q_next;

          always @(*)
            if(EN)
              if(~SLn)
                Q_next=SD;
             else
                Q_next=D;
            else
              Q_next=Q_reg;
          assign Q=Q_reg;
        end
        else begin
          always @(*)
            if(~ALn)
              Q_reg<=~ADn;
            else if(CLK)
              Q_reg<=Q_next;

          always @(*)
            if(EN)
              if(~SLn)
                Q_next=SD;
             else
                Q_next=D;
            else
              Q_next=Q_reg;
          assign Q=Q_reg;
        end
      endgenerate

endmodule
```

Testbench Code

```
48  `timescale 1ns/1ps
49  module SLE_tb();
50
51    reg D,CLK,EN;
52    reg ALn;
53    reg SLn;
54    wire Q;
55    SLE  #(0,0,1)dut (D, CLK, EN, ALn, SLn,Q) ;
56
57    initial begin
58      CLK=0;
59      forever
60      #10 CLK=~CLK;
61    end
62    initial begin
63      D=0;
64      ALn=0;
65      EN=0;
66      SLn=0;
67      @(negedge CLK);
68      ALn=1;
69      @(negedge CLK);
70      EN=1;
71      @(negedge CLK);
72      SLn=1;
73      repeat(2) @(negedge CLK);
74      $stop;
75    end
76  endmodule
```

Latch Simulation Wave



Flipflop Simulation Wave