Assignment_3

(Under supervision of Eng: Karim Wassem)

Q1]

question1.v | question1_tb.v

Assignment_3 > q1 > question1.v

```
1   module q1(E,D,CLK,Q);
2
3   input E,D,CLK;   // E is an enable signal
4   output reg Q;
5
6   always @(posedge CLK) begin
7
8     if(E)
9     Q<=D;
10
11  end
12
13  endmodule
14
15
```

Figure (1) : Verilog code

```
VSIM 46> run -all
# AT time= 0  D=0  enable=0  Q=x
# AT time= 50  D=0  enable=1  Q=x
# AT time= 75  D=0  enable=1  Q=0
# AT time= 100  D=1  enable=1  Q=0
# AT time= 125  D=1  enable=1  Q=1
# AT time= 200  D=1  enable=0  Q=1
# AT time= 250  D=1  enable=1  Q=1
# AT time= 300  D=0  enable=1  Q=1
# AT time= 325  D=0  enable=1  Q=0
# AT time= 350  D=1  enable=0  Q=0
# ** Note: $stop    : D:/VS_code/Verilog codes/Assignment_3/q1/question1_tb.v(29)
#    Time: 500 ns  Iteration: 1  Instance: /q1_tb
```

Figure (2) : transcript

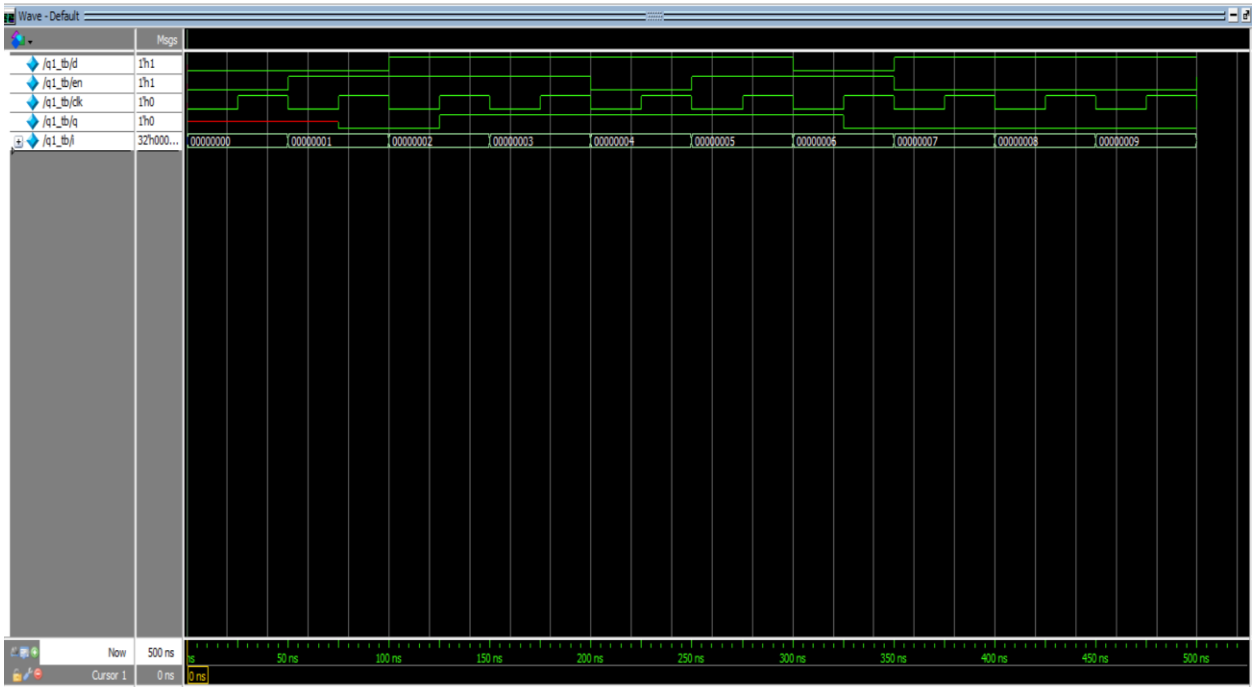

Figure (3): waveform

question1.v | question1_tb.v

Assignment_3 > q1 > question1_tb.v

```
1   module q1_tb();
2
3   reg d,en,clk;
4   wire q;
5   integer i;
6
7   q1 DUT (.D(d),.E(en),.CLK(clk),.Q(q));
8
9   initial begin
10  clk=0;
11  forever
12  #25 clk=~clk;     // this mean that time period of clock=50 ns
13
14  end
15
16  initial begin
17
18  d=0;
19  en=0;
20
21  for(i=0;i<10;i=i+1) begin
22
23  @(negedge clk);
24  d=$random;
25  en=$random;
26
27  end
28
29  $stop;
30
31  end
32
33  initial
34  $monitor("AT time= %g\t D=%b \t enable=%b \tQ=%b",$time,d,en,q);
35
36  endmodule
```

Figure(4) : testbench

➔ **As we see from these figures that we use randomized values for the input to produce the output of the D flip flop with monitoring it's values as shown in figure (2) "transcript".**

Q2]

question2.v X | question2_tb.v

Assignment_3 > q2 > question2.v

```
1  module q2(clear,G,D,Q);
2
3  input clear,G,D;   // G is an enable signal
4  output reg Q;
5
6  always @(clear or G or D) begin
7
8   if(~clear)
9   Q<=0;
10  else if(G)
11  Q<=D;
12
13 end
14
15 endmodule
```

Figure(5): Verilog code

question2.v | question2_tb.v X

Assignment_3 > q2 > question2_tb.v

```
1  module q2_tb();
2
3  reg CLR,D,G;
4  wire q;
5  integer i;
6
7  q2 DUT (.D(D),.clear(CLR),.G(G),.Q(q));
8
9  initial begin
10
11 for(i=0;i<10;i=i+1) begin
12 D=$random;
13 CLR=$random;
14 G=$random;
15 #50;
16 end
17
18 $stop;
19
20 end
21
22 initial
23 $monitor("AT time= %g\t D=%b \tclear=%b\tenable(G)=%b\tQ=%b",$time,D,CLR,G,q);
24
25 endmodule
```

Figure (6): testbench code

➔ **This design represent a D latch with active enable(G) signal and active low (clear) where as shown in figure (6) we test the design using randomized values of inputs.**

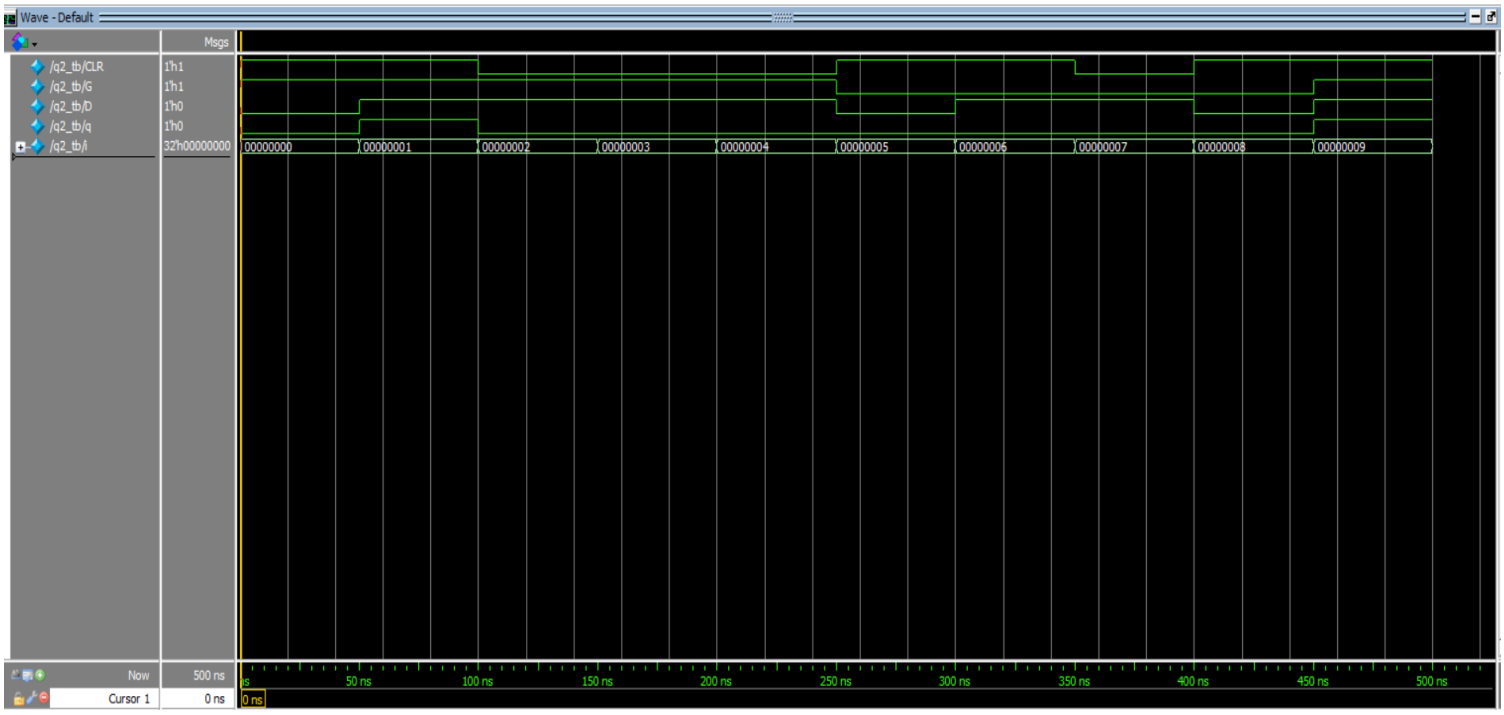

Figure (7): waveform

Q3]

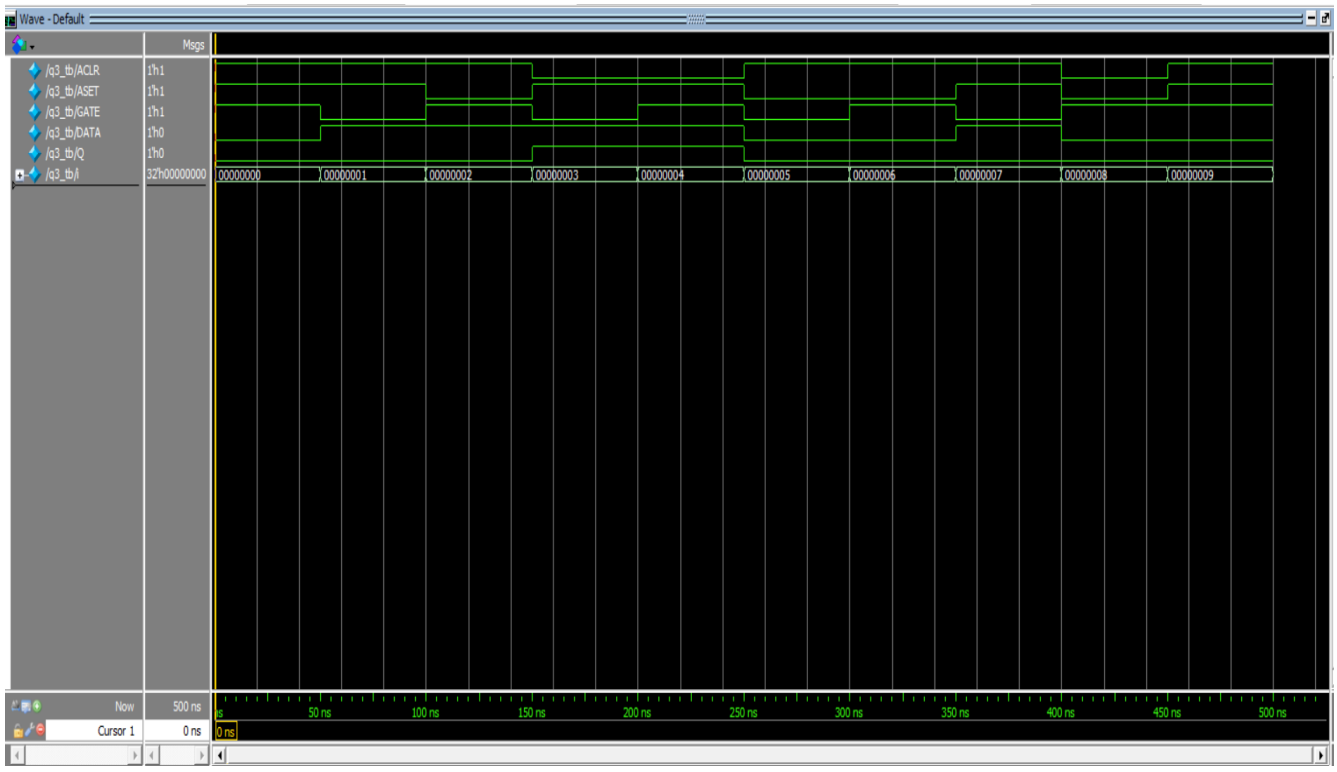

Figure (8): waveform

question3.v question3_tb.v

Assignment_3 > q3 > question3.v

```
1   module q3(data,gate,aclr,aset,q);
2
3   parameter LAT_WIDTH=1;
4   input [LAT_WIDTH-1:0]data;
5   input gate,aclr,aset;
6   output reg [LAT_WIDTH-1:0] q;
7
8   always @(*) begin
9
10  if(aclr)
11  q<=0;
12  else begin
13  if(aset)
14  q<=1;
15  else if(gate)
16  q<=data;
17
18  end
19
20  end
21
22  endmodule
```

Figure (9): Verilog code

question3.v X question3_tb.v X

Assignment_3 > q3 > question3_tb.v

```
1   module q3_tb();
2
3   reg DATA,ACLR,ASET,GATE;
4   wire Q;
5   integer i;
6
7   q3 DUT (.data(DATA),.gate(GATE),.aclr(ACLR),.aset(ASET),.q(Q));
8
9   initial begin
10
11  for(i=0;i<10;i=i+1) begin
12  DATA=$random;
13  ACLR=$random;
14  ASET=$random;
15  GATE=$random;
16  #50;
17  end
18
19  $stop;
20
21  end
22
23  initial
24  $monitor(" Data=%b \tclear=%b\tset=%b\tgate(enable)=%b\tQ=%b",DATA,ACLR,ASET,GATE,Q);
25
26  endmodule
```

Figure (10) : testbench code

➔ **Figure (8&9&10) shows the waveform and codes for question3 .Also as shown in figure (10) we use the randomized values for inputs to detect the output of the D latch.**

Q4]

tff.v | dff.v | parameterized_ff.v

Assignment_3 > q4 > dff.v

```
1   module dff(d,clk,rst_n,q,qbar);
2
3   input d,clk,rst_n;
4   output reg q;
5   output qbar;
6
7   always @(posedge clk or negedge rst_n) begin
8
9   if(~rst_n)
10  q<=0;
11  else
12  q<=d;
13
14  end
15
16  assign qbar=~q;
17
18  endmodule
```

Figure (11): D flipflop code

tff.v | dff.v | parameterized_ff.v

Assignment_3 > q4 > tff.v

```
1   module tff(t,clk,rst_n,q,qbar);
2
3   input t,clk,rst_n;
4   output reg q;
5   output qbar;
6
7   always @(posedge clk or negedge rst_n) begin
8
9   if(~rst_n)
10  q<=0;
11  else if(t)
12  q<=~q;
13
14  end
15
16  assign qbar=~q;
17
18  endmodule
```

Figure (12): T flipflop code

tff.v | dff.v | parameterized_ff.v | test1_tb.v | test2_tb

Assignment_3 > q4 > parameterized_ff.v

```
1   module para_ff(d,clk,rst_n,q,qbar);
2
3   parameter FF_TYPE="DFF";
4   input d,clk,rst_n;
5   output q,qbar;
6
7   generate
8       case(FF_TYPE)
9
10      "DFF": dff f1(.d(d),.clk(clk),.rst_n(rst_n),.q(q),.qbar(qbar));
11
12      "TFF": tff f2(.t(d),.clk(clk),.rst_n(rst_n),.q(q),.qbar(qbar));
13
14      endcase
15
16  endgenerate
17
18
19  endmodule
```

Figure (13): parameterized flip flop code

tff.v | dff.v | parameterized_ff.v | test1_tb.v | test2_tb.v | Genera

Assignment_3 > q4 > test1_tb.v

```
1  module test1_tb();
2  parameter N1="DFF";
3  reg D,CLK,RST_N;
4  wire Q,QBAR;
5  wire q_expected,qbar_expected;
6  integer i;
7
8  para_ff #(.FF_TYPE(N1)) DUT1 (.d(D),.clk(CLK),.rst_n(RST_N),.q(Q),.qbar(QBAR));
9  dff  DUT2 (.d(D),.clk(CLK),.rst_n(RST_N),.q(q_expected),.qbar(qbar_expected));
10
11 initial begin
12 CLK=0;
13 forever
14 #25 CLK=~CLK;     // this mean that the period of the clock =50ns
15 end
16
17 initial begin
18 RST_N=0;
19 D=0;
20 #50;
21 RST_N=1;
22 for(i=0;i<10;i=i+1) begin
23 @(negedge CLK);
24 D=$random;
25 if(Q!=q_expected||QBAR!=qbar_expected) begin
26 $display("Error,this design is incorrect!!");
27 $stop;
28 end
29 end
30 $stop;
31 end
32
33 initial
34 $monitor("D=%b\treset=%b\tQ=%b\tQ_expected=%b\tQbar=%b\tQbar_expected=%b",D,RST_N,Q,q_expected,QBAR,qbar_expected);
35
36 endmodule
37
```

Figure (14): first testbench for the first design (DFF)

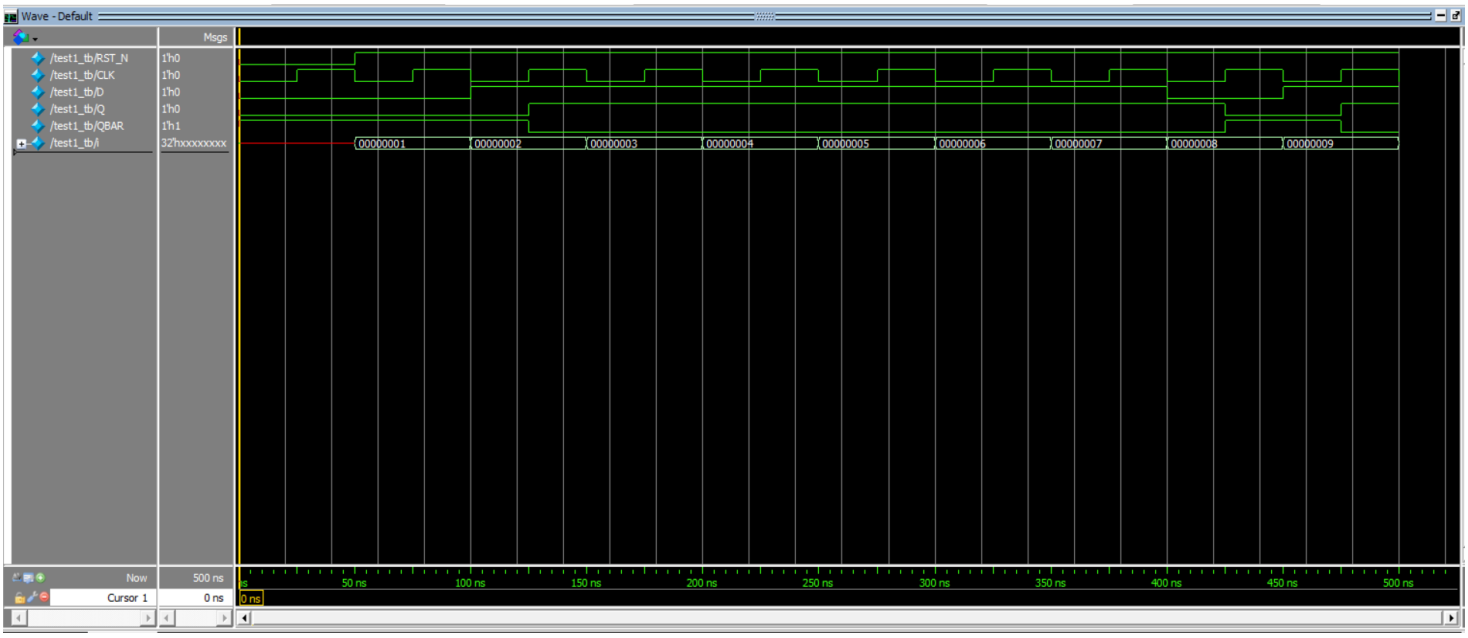

Figure (15) : waveform for the first testbench

```
VSIM 96> run -all
# D=0   reset=0 Q=0     Q_expected=0 Qbar=1 Qbar_expected=1
# D=0   reset=1 Q=0     Q_expected=0 Qbar=1 Qbar_expected=1
# D=1   reset=1 Q=0     Q_expected=0 Qbar=1 Qbar_expected=1
# D=1   reset=1 Q=1     Q_expected=1 Qbar=0 Qbar_expected=0
# D=0   reset=1 Q=1     Q_expected=1 Qbar=0 Qbar_expected=0
# D=0   reset=1 Q=0     Q_expected=0 Qbar=1 Qbar_expected=1
# D=1   reset=1 Q=0     Q_expected=0 Qbar=1 Qbar_expected=1
# D=1   reset=1 Q=1     Q_expected=1 Qbar=0 Qbar_expected=0
# ** Note: $stop    : D:/VS_code/Verilog codes/Assignment_3/q4/test1_tb.v(40)
#    Time: 500 ns  Iteration: 1  Instance: /test1_tb
# Break in Module test1_tb at D:/VS_code/Verilog codes/Assignment_3/q4/test1_tb.v line 40
```

Figure (16): transcript for the first design using DFF

tff.v | dff.v | parameterized_ff.v | test1_tb.v | test2_tb.v | Generate

Assignment_3 > q4 > test2_tb.v

```
1  module test2_tb();
2  parameter N2="TFF";
3  reg D,CLK,RST_N;
4  wire Q,QBAR;
5  wire q_expected,qbar_expected;
6  integer i;
7
8  para_ff #(.FF_TYPE(N2)) DUT3 (.d(D),.clk(CLK),.rst_n(RST_N),.q(Q),.qbar(QBAR));
9  tff  DUT4 (.t(D),.clk(CLK),.rst_n(RST_N),.q(q_expected),.qbar(qbar_expected));
10
11 initial begin
12 CLK=0;
13 forever
14 #25 CLK=~CLK;
15 end
16
17 initial begin
18 RST_N=0;
19 D=0;
20 #50;
21 RST_N=1;
22 for(i=0;i<10;i=i+1) begin
23 @(negedge CLK);
24 D=$random;
25 if(Q!=q_expected||QBAR!=qbar_expected) begin
26 $display("Error,this design is incorrect!!");
27 $stop;
28 end
29 end
30 $stop;
31 end
32
33 initial
34 $monitor("T=%b\treset=%b\tQ=%b\tQ_expected=%b\tQbar=%b\tQbar_expected=%b",D,RST_N,Q,q_expected,QBAR,qbar_expected);
35
36 endmodule
```

Figure (17) : second testbench for the second design (TFF)

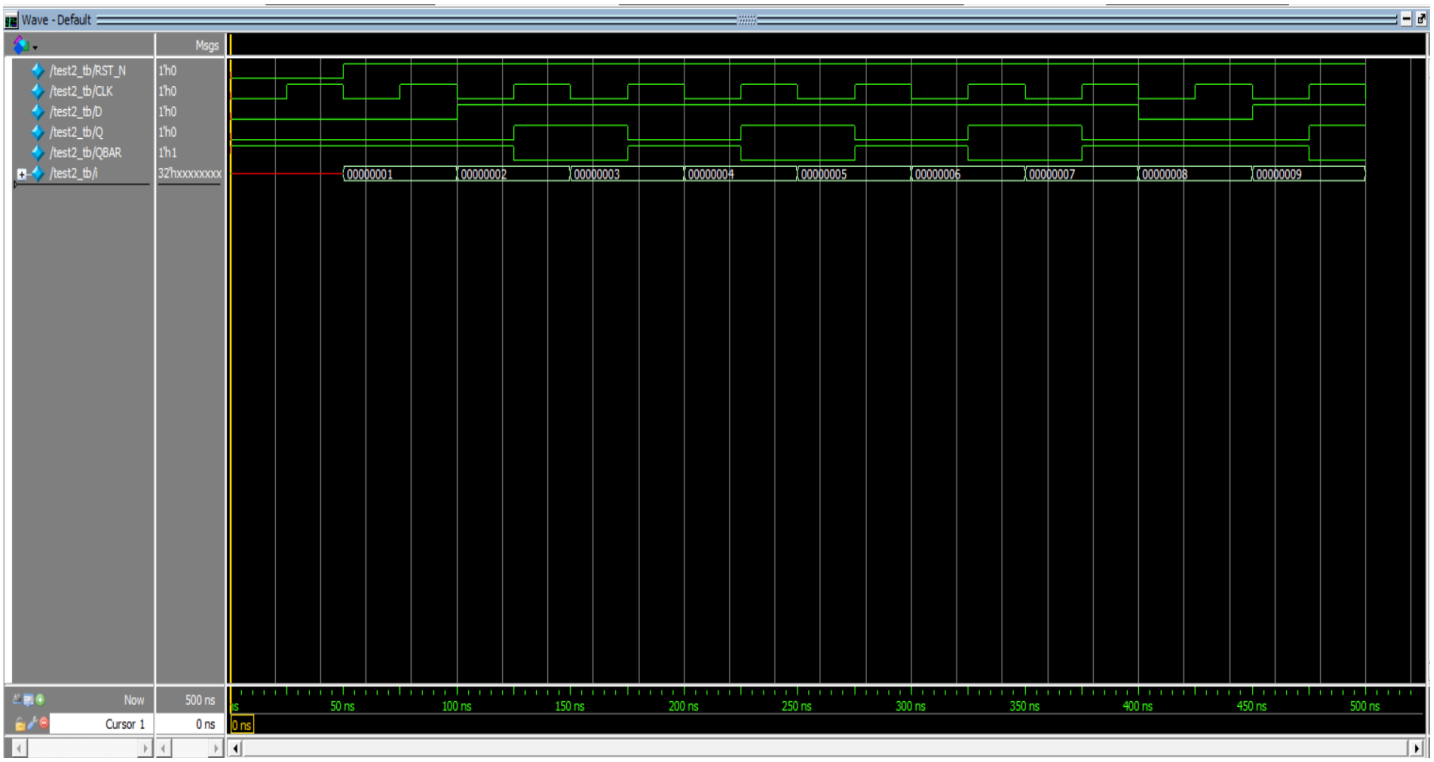

Figure (18) : waveform for the second design

```
VSIM 102> run -all
# T=0    reset=0 Q=0     Q_expected=0 Qbar=1 Qbar_expected=1
# T=0    reset=1 Q=0     Q_expected=0 Qbar=1 Qbar_expected=1
# T=1    reset=1 Q=0     Q_expected=0 Qbar=1 Qbar_expected=1
# T=1    reset=1 Q=1     Q_expected=1 Qbar=0 Qbar_expected=0
# T=1    reset=1 Q=0     Q_expected=0 Qbar=1 Qbar_expected=1
# T=1    reset=1 Q=1     Q_expected=1 Qbar=0 Qbar_expected=0
# T=1    reset=1 Q=0     Q_expected=0 Qbar=1 Qbar_expected=1
# T=1    reset=1 Q=1     Q_expected=1 Qbar=0 Qbar_expected=0
# T=1    reset=1 Q=0     Q_expected=0 Qbar=1 Qbar_expected=1
# T=0    reset=1 Q=0     Q_expected=0 Qbar=1 Qbar_expected=1
# T=1    reset=1 Q=0     Q_expected=0 Qbar=1 Qbar_expected=1
# T=1    reset=1 Q=1     Q_expected=1 Qbar=0 Qbar_expected=0
# ** Note: $stop    : D:/VS_code/Verilog codes/Assignment_3/q4/test2_tb.v(40)
#    Time: 500 ns  Iteration: 1  Instance: /test2_tb
# Break in Module test2_tb at D:/VS_code/Verilog codes/Assignment_3/q4/test2_tb.v line 40
```

Figure (19): transcript for the second design

- **As shown in the figures (14 & 17) “two testbenches” we use the randomization method but with self checking as we test our designs using two golden references (DFF & TFF).**

- **Moreover, the figures(16 & 19) “two transcript” verify that our designs operate well due to the equality between the out_dut(Q) and the expected output from the golden reference design (Q_expected).**

Q5]

dff.v | question5.v | question5_tb.v

Assignment_3 > q5 > question5.v

```
1  module counter (RST_N,CLK,out);
2
3  input RST_N,CLK;
4  output [3:0] out;
5  wire w1,w2,w3;
6
7  dff f1(.d(out[0]),.clk(CLK),.rst_n(RST_N),.q(w1),.qbar(out[0]));
8
9  dff f2(.d(out[1]),.clk(w1),.rst_n(RST_N),.q(w2),.qbar(out[1]));
10
11 dff f3(.d(out[2]),.clk(w2),.rst_n(RST_N),.q(w3),.qbar(out[2]));
12
13 dff f4(.d(out[3]),.clk(w3),.rst_n(RST_N),.q(),.qbar(out[3]));
14
15
16 endmodule
17
```

Figure (20): design code

dff.v | question5.v | question5_tb.v

Assignment_3 > q5 > question5_tb.v

```
1  module counter_tb();
2
3  reg reset,clk;
4  wire [3:0] out_dut;
5
6  counter DUT(.RST_N(reset),.CLK(clk),.out(out_dut));
7
8  initial begin
9  clk=0;
10 forever
11 #25 clk=~clk;
12 end
13
14 initial begin
15 reset=0;
16 #50;          // #50 mean one clock cycle (to reset the counter)
17 reset=1;
18 #1000;         // #1000 mean 20 clock cycles
19 $stop;
20 end
21
22 initial
23 $monitor(" reset=%b \t out_dut=%b ",reset,out_dut);
24
25 endmodule
```

Figure (21) : testbench code

dff.v | question5.v | question5_tb.v | main.do

Assignment_3 > q5 > main.do

```
1  vlib work
2  vlog question5.v question5_tb.v
3  vsim -voptargs=+acc work.counter_tb
4  add wave *
5  run -all
```

Figure (22) : do file

```
QuestaSim> do main.do
#  ** Warning: (vlib-34) Library already exists at "work".
#  Errors: 0, Warnings: 1
#  QuestaSim-64 vlog 2021.1 Compiler 2021.01 Jan 19 2021
#  Start time: 05:45:34 on Jul 20,2023
#  vlog -reportprogress 300 question5.v question5_tb.v
#  -- Compiling module counter
#  -- Compiling module counter_tb
#
#  Top level modules:
#         counter_tb
#  End time: 05:45:34 on Jul 20,2023, Elapsed time: 0:00:00
#  Errors: 0, Warnings: 0
#  vsim -voptargs="+acc" work.counter_tb
#  Start time: 05:45:36 on Jul 20,2023
#  ** Note: (vsim-3813) Design is being optimized due to module recompilation...
#  Loading work.counter_tb(fast)
#  Loading work.counter(fast)
#  Loading work.dff(fast)
#  Loading work.dff(fast__1)
#    reset=0    out_dut=1111
#    reset=1    out_dut=1111
#    reset=1    out_dut=0000
#    reset=1    out_dut=0001
#    reset=1    out_dut=0010
#    reset=1    out_dut=0011
#    reset=1    out_dut=0100
#    reset=1    out_dut=0101
#    reset=1    out_dut=0110
#    reset=1    out_dut=0111
#    reset=1    out_dut=1000
#    reset=1    out_dut=1001
#    reset=1    out_dut=1010
#    reset=1    out_dut=1011
#    reset=1    out_dut=1100
#    reset=1    out_dut=1101
#    reset=1    out_dut=1110
#    reset=1    out_dut=1111
#    reset=1    out_dut=0000
#    reset=1    out_dut=0001
#    reset=1    out_dut=0010
#    reset=1    out_dut=0011
#  ** Note: $stop    : question5_tb.v(19)
#     Time: 1050 ns  Iteration: 0  Instance: /counter_tb
#  Break in Module counter_tb at question5_tb.v line 19
```

Figure (23): transcript using do command

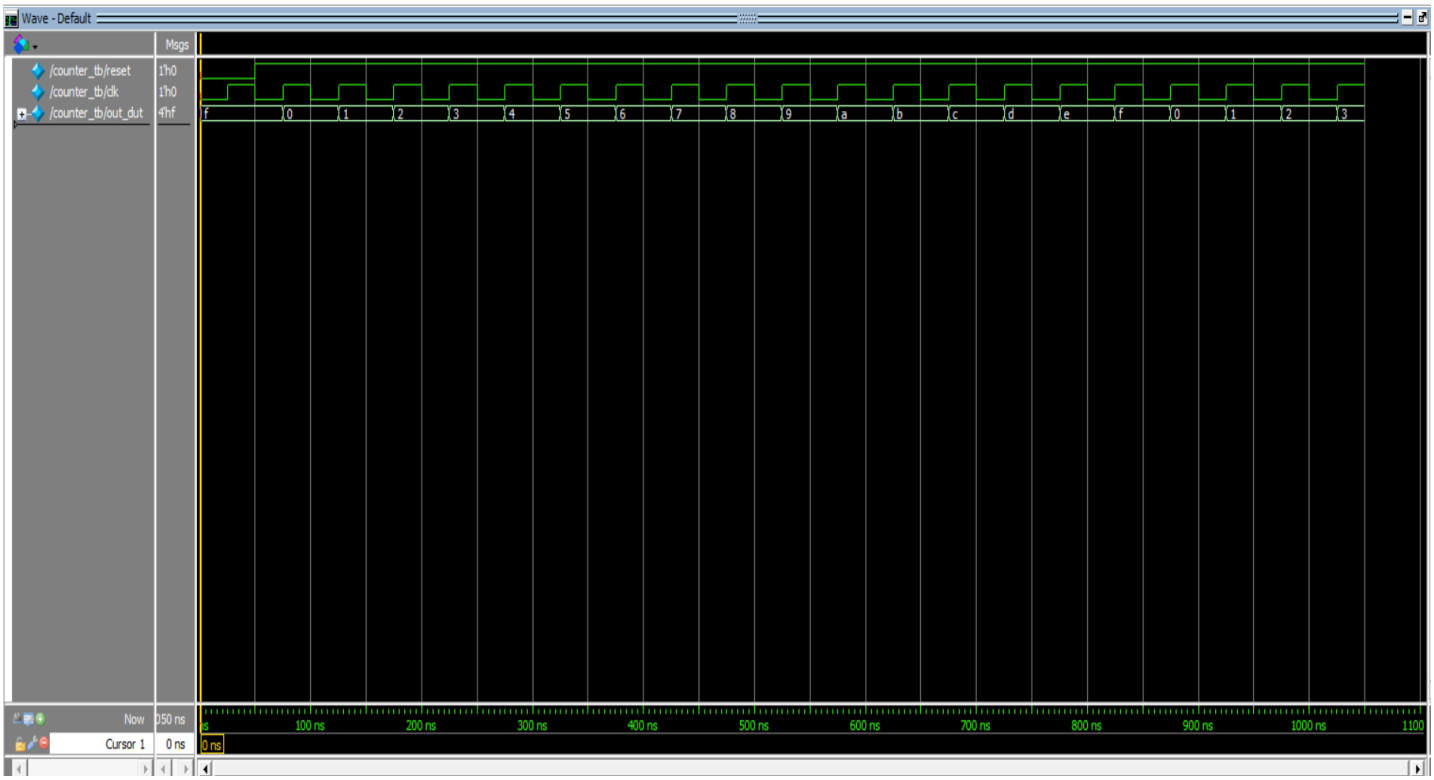

Figure (24): waveform

Q6]

question6.v × question6_tb.v

Assignment_3 > q6 > question6.v

```
1  module SLE(ALn,ADn,LAT,CLK,EN,SLn,SD,D,Q);
2
3  input ALn,ADn,LAT,CLK,EN,SLn,SD,D;
4  output reg Q;
5
6  always @(posedge CLK or negedge ALn) begin
7
8  if(~ALn)
9     Q<=~ADn;
10
11 else if(!LAT)
12 case({EN,SLn})
13 2'b10:Q<=SD;
14 2'b11:Q<=D;
15 endcase
16
17 end
18
19 always @(*) begin
20
21 if(LAT) begin
22   if(~ALn)
23     Q<=~ADn;
24 else begin
25 case({CLK,EN,SLn})
26 3'b110:Q<=SD;
27 3'b111:Q<=D;
28 endcase
29
30 end
31
32 end
33
34 end
35
36 endmodule
```

Figure (25): design verilog code

question6.v question6_tb.v ×

Assignment_3 > q6 > question6_tb.v

```
1  module SLE_tb();
2
3  reg d,clk,en,aln,adn,sln,sd,lat;
4  wire q;
5  integer i;
6
7  SLE DUT(.D(d),.CLK(clk),.EN(en),.ALn(aln),.ADn(adn),.SLn(sln),.SD(sd),.LAT(lat),.Q(q));
8
9  initial begin
10 clk=0;
11 forever
12 #25 clk=~clk;
13 end
14
15 initial begin
16
17    aln=0; adn=0; d=0; en=0; sln=0; sd=0; lat=0;
18    #40;
19    aln=1;
20 for(i=0;i<10;i=i+1) begin
21
22    d=$random;
23    adn=$random;
24    en=$random;
25    sln=$random;
26    sd=$random;
27    lat=$random;
28    #50;
29 end
30 $stop;
31 end
32
33 initial
34 $monitor(" D=%b\t enable=%b\t SLn=%b\t SD=%b \t lat=%b\t out_dut=%b",d,en,sln,sd,lat,q);
35
36 endmodule
```

Figure (26): testbench code

➔ **The above figures show the design and the testbench where in this case we use randomized input without self checking**

➔ **In addition to that the wave form snippit for this design is shown below “figure (27)”**

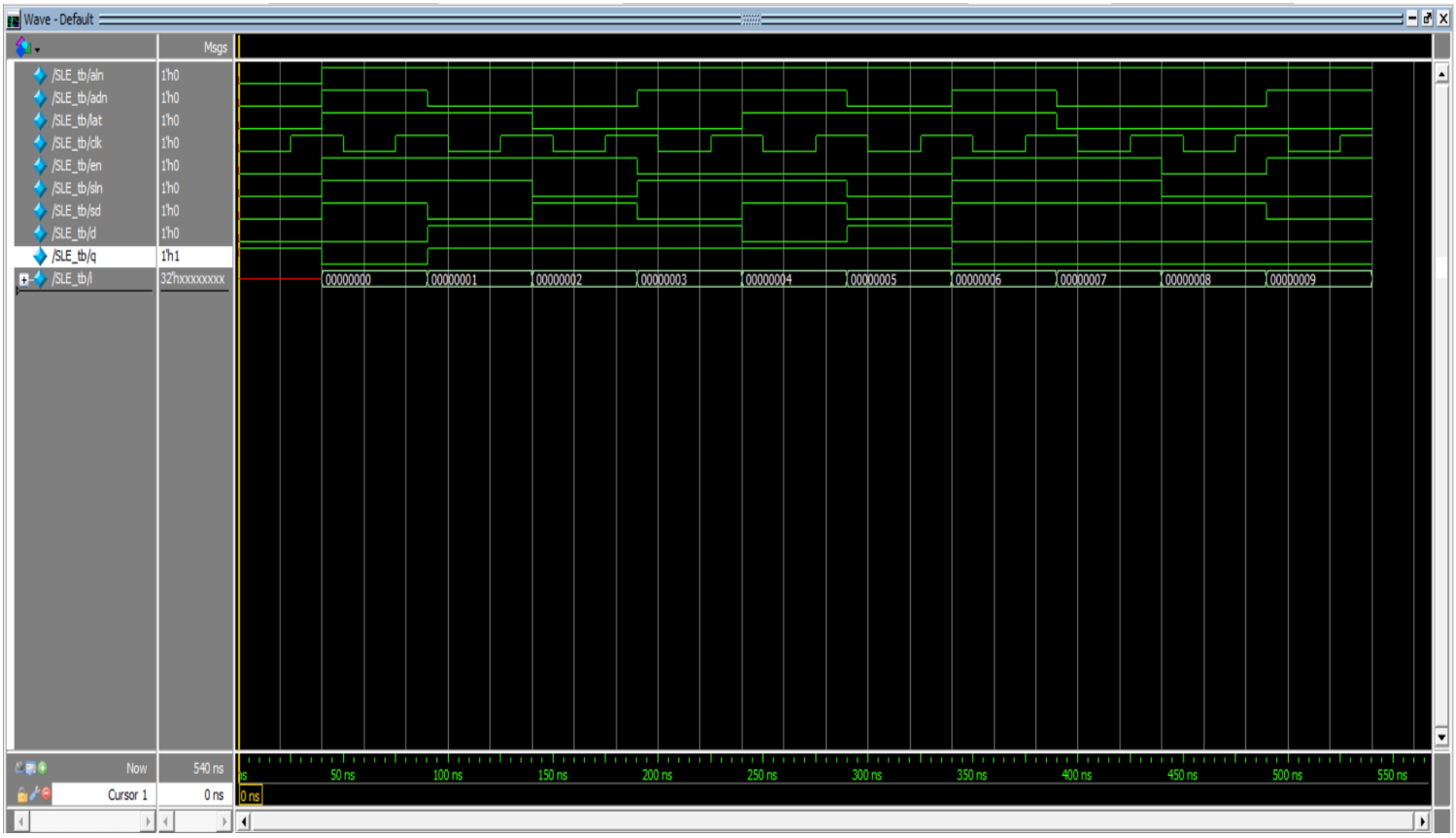

Figure (27) : waveform