

يوسف أحمد محمد ابراهيم

Assignment 1
Extra

Question 1:

1. Design :

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

2. verification plan :

1	Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
2	RESET_TEST	When the reset is asserted, outputs C should be low.	assert_reset() task that asserts rst to 1, wait then deassert rst to 0 at the beginning and end of the simulation.	-	check_result() task checks that output is 0 during reset.
3	WITH_EN_TEST	Output depends on en value, if en = 1, q = D.	Directed test at en = 0, and at en = 1	-	check_result() checks the value of q.
4	WITHOUT_EN_TEST	Output q = D whatever the value of en is.	Directed test at en = 0, and at en = 1	-	check_result() checks the value of q.

3. Testbench 1 (USE_EN = 1) :

```
1  module DFF_tb1();
2      logic clk;
3      logic rst;
4      logic d;
5      logic q;
6      logic en;
7
8      logic q_exp;
9
10     integer correct_count, error_count;
11
12     dff dut(clk, rst, d, q, en);
13
14     always @(posedge clk, posedge rst) begin
15         if(rst)
16             q_exp <= 0;
17         else if(en)
18             q_exp = d;
19     end
20
21     initial begin
22         clk = 0;
23         forever
24             #1 clk = ~clk;
25     end
26
27     initial begin
28         correct_count = 0;
29         error_count = 0;
30         d = 0;
31         en = 0;
32         //reset test
33         rst = 1;
34         check_result();
35         rst = 0;
36
37         //WITH_EN check
38         d = 1;
39         check_result();
40         en = 1;
41         check_result();
42
43         d = 0;
44         en = 0;
45         check_result();
46
47         rst = 1;
48         check_result();
49         rst = 0;
50
51         $display("*** errors: %0d, success: %0d ***", error_count, correct_count);
52         $stop;
53     end
54
55     task check_result();
56         @(negedge clk);
57         if(q != q_exp) begin
58             error_count = error_count + 1;
59             $display ("*** ERROR! D = %0d,en = %0d, rst = %0d, q = %0d ***", d, en, rst, q);
60         end
61         else
62             correct_count = correct_count + 1;
63     endtask
64 endmodule
```

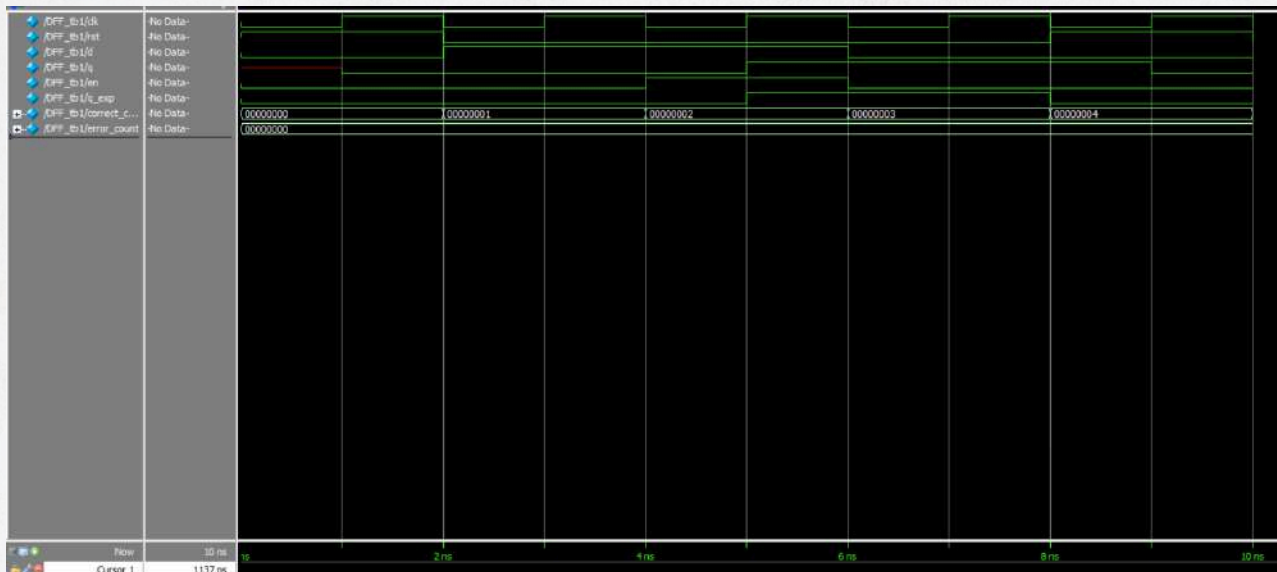
3. Testbench (USE_EN = 0):

```
68 module DFF_tb2();
69     logic clk;
70     logic rst;
71     logic d;
72     logic q;
73     logic en;
74
75     logic q_exp;
76
77     integer correct_count, error_count;
78
79     dff #(.USE_EN(0)) dut2(clk, rst, d, q, en);
80
81     always @(posedge clk, posedge rst) begin
82         if(rst)
83             q_exp <= 0;
84         else
85             q_exp = d;
86     end
87
88     initial begin
89         clk = 0;
90         forever
91             #1 clk = ~clk;
92     end
93
94     initial begin
95         correct_count = 0;
96         error_count = 0;
97         d = 0;
98         en = 0;
99         //reset test
100        rst = 1;
101        check_result();
102        rst = 0;
103
104        //WITH_EN check
105        d = 1;
106        check_result();
107        en = 1;
108        check_result();
109
110        d = 0;
111        en = 0;
112        check_result();
113
114        rst = 1;
115        check_result();
116        rst = 0;
117
118        $display("*** errors: %0d, success: %0d ***", error_count, correct_count);
119        $stop;
120    end
121
122    task check_result();
123        @(negedge clk);
124        if(q != q_exp) begin
125            error_count = error_count + 1;
126            $display ("*** ERROR! D = %0d, en = %0d, rst = %0d, q = %0d ***", d, en, rst, q);
127        end
128        else
129            correct_count = correct_count + 1;
130    endtask
131 endmodule
132
133
```


4. Do file (*changing vsim instance name and coverage save to the testbench to simulate*):

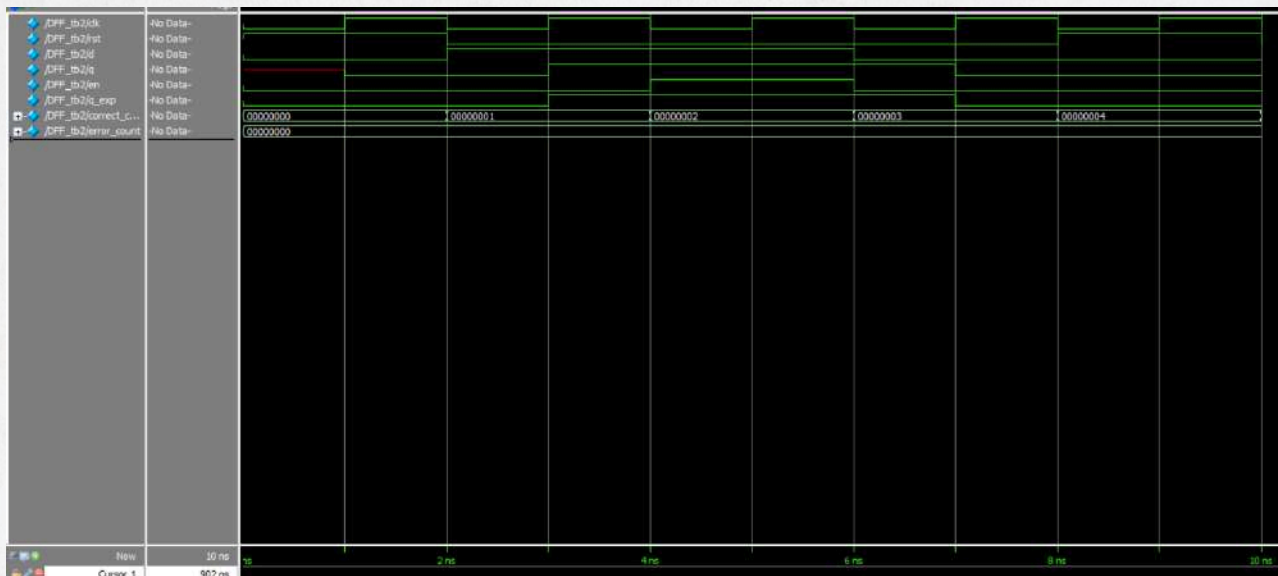
```
1  vlib work
2  vlog DFF.v DFF_tb.sv +cover -covercells
3  vsim -voptargs=+acc work.DFF_tb2 -cover
4  add wave *
5  coverage save 3_ALU_tb2.ucdb -onexit -du work.dff
6  run -all
```


5. Qesta sim wave snippets (First TB):



```
# *** errors: 0, success: 5 ***  
# ** Note: $stop      : DFF_tb.sv(53)  
#    Time: 10 ns   Iteration: 1   Instance: /DFF_tb1  
# Break in Module DFF_tb1 at DFF_tb.sv line 53
```

5. Qesta sim wave snippets (Second TB):



```
# *** errors: 0, success: 5 ***  
# ** Note: $stop      : DFF_tb.sv(120)  
#    Time: 10 ns  Iteration: 1  Instance: /DFF_tb2  
# Break in Module DFF_tb2 at DFF_tb.sv line 120
```


6. Statement coverage report :

```
31  Statement Coverage:
32  Enabled Coverage      Bins      Hits      Misses  Coverage
33  -----
34  Statements            4        4         0    100.00%
35
36  =====Statement Details=====
37
38  Statement Coverage for instance /\work.dff  --
39
40  Line      Item      Count      Source
41  ----      -
42  File DFF.v
43  1          module dff(clk, rst, d, q, en);
44
45  2          parameter USE_EN = 1;
46
47  3          input clk, rst, d, en;
48
49  4          output reg q;
50
51  5
52
53  6          1          10      always @(posedge clk) begin
54
55  7          if (rst)
56
57  8          1          4          q <= 0;
58
59  9          else
60
61  10         if(USE_EN) begin
62
63  11         if (en)
64
65  12         1          1          q <= d;
66
67  13         end
68
69  14         else
70
71  15         1          3          q <= d;
72
```


7. Branch coverage report :

```

7 Branch Coverage:
8   Enabled Coverage           Bins      Hits      Misses  Coverage
9   -----
10  Branches                   4        4          0    100.00%
11
12  =====Branch Details=====
13
14  Branch Coverage for instance /\work.dff
15
16  Line      Item                      Count      Source
17  ----      -
18  File DFF.v
19  -----IF Branch-----
20  7          Count coming in to IF
21  7          1                      4          if (rst)
22
23  11         1                      1          if (en)
24
25  9          1                      2          All False Count
26  9          1                      3          else
27
28  Branch totals: 4 hits of 4 branches = 100.00%

```

8. Toggle coverage report :

```

74  ✓ Toggle Coverage:
75      Enabled Coverage          Bins      Hits      Misses  Coverage
76      -----
77      Toggles                   10       10        0    100.00%
78
79      =====Toggle Details=====
80
81  ✓ Toggle Coverage for instance /\work.dff  --
82
83      Node      1H->0L      0L->1H  "Coverage"
84      -----
85      clk        2          2    100.00
86      d          2          2    100.00
87      en         2          2    100.00
88      q          2          2    100.00
89      rst        2          2    100.00
90
91  Total Node Count      =          5
92  Toggled Node Count   =          5
93  Untoggled Node Count =          0
94
95  Toggle Coverage      =    100.00% (10 of 10 bins)
96
97
98  Total Coverage By Instance (filtered view): 100.00%

```