

## Assignment 1 extra

Q1

## CODE TB 1 :

```
dff_tb1.v
1  module dff_tb1 ();
2  parameter USE_EN = 1;
3  logic clk, rst, d, en;
4  logic q;
5  logic q_ex;
6
7  dff r (.*);
8
9  int error_counter = 0;
10 int correct_counter = 0;
11
12 task golden_model ;
13     if (rst)
14         q_ex <= 0;
15     else
16         if(USE_EN)
17             if (en)
18                 q_ex <= d;
19         else
20             q_ex <= d;
21     endtask //automatic
22
23
24 task cheack_result ;
25     @(negedge clk);
26     if (q_ex != q) begin
27         $display("%t error", $time);
28         error_counter++;
29     end
30     else
31         correct_counter++;
32 endtask
33
34
```

```

34
35 task cheack_reset ;
36     rst = 1;
37     @(negedge clk);
38     cheack_result ;
39     rst = 0;
40
41 endtask
42
43 initial begin
44     clk = 0;
45     forever begin
46         #1 clk = ~clk;
47     end
48 end
49
50 initial begin
51     // 1
52     cheack_reset ;
53
54     //2
55     repeat (400) begin
56         d = $random;
57         en = $random;
58         golden_model ;
59         cheack_result ;
60     end
61
62     // 3
63     cheack_reset ;
64
65     $display("error_counter = %d correct_counter = %d",error_counter,correct_counter);
66     $stop;
67 end
68
69 endmodule

```

DO FILE 1 :

```

dff1.do
1  vlib work
2  vlog dff.v dff_tb.sv +cover -covercells
3  vsim -voptargs=+acc work.dff_tb1 -cover
4  add wave *
5  coverage save dff_tb1.ucdb -onexit
6  run -all

```

## DISPLAY 1 :

```
# Loading work.dff_tbl(fast)
# Loading work.dff(fast)
# error_counter =          0  correct_counter =          402
# ** Note: $stop      : dff_tb.sv(66)
#   Time: 808 ns  Iteration: 1  Instance: /dff_tbl
# Break in Module dff_tbl at dff_tb.sv line 66
```

## COVERAGE 1 :

### Branch Coverage:

Enabled Coverage	Bins	Hits	Misses	Coverage
-----	----	----	-----	-----
Branches	3	3	0	100.00%

### Statement Coverage:

Enabled Coverage	Bins	Hits	Misses	Coverage
-----	----	----	-----	-----
Statements	4	4	0	100.00%

### Toggle Coverage:

Enabled Coverage	Bins	Hits	Misses	Coverage
-----	----	----	-----	-----
Toggles	10	10	0	100.00%

## CODE TB 2 :

```

71  module dff_tb2 ();
72  parameter USE_EN = 0;
73  logic clk, rst, d, en;
74  logic q;
75  logic q_ex;
76
77  dff r (.*);
78
79  int error_counter = 0;
80  int correct_counter = 0;
81
82  task golden_model ;
83  if (rst)
84      q_ex <= 0;
85  else
86      if(USE_EN)
87          if (en)
88              q_ex <= d;
89      else
90          q_ex <= q;
91  endtask //automatic
92
93
94  task cheack_result ;
95  @(negedge clk);
96  if (q_ex != q) begin
97      $display("%t error", $time);
98      error_counter++;
99  end
100  else
101      correct_counter++;
102  endtask
103
104

```

```

105 task check_reset ;
106     rst = 1;
107     @(negedge clk);
108     check_result ;
109     rst = 0;
110
111 endtask
112
113 initial begin
114     clk = 0;
115     forever begin
116         #1 clk = ~clk;
117     end
118 end
119
120 initial begin
121     // 1
122     check_reset ;
123
124     //2
125     repeat (400) begin
126         d = $random;
127         en = $random;
128         golden_model ;
129         check_result ;
130     end
131
132     // 3
133     check_reset ;
134
135     $display("error_counter = %d  correct_counter = %d",error_counter,correct_counter);
136     $stop;
137 end
138
139 endmodule

```

DO FILE 2 :

```

dff2.do
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 dff_tb2.ucdb -onexit
6  run -all

```

DISPLAY 2 :

```

# Loading work.dff_tbl(fast)
# Loading work.dff(fast)
# error_counter = 0 correct_counter = 402
# ** Note: $stop : dff_tb.sv(66)
# Time: 808 ns Iteration: 1 Instance: /dff_tbl
# Break in Module dff_tbl at dff_tb.sv line 66

```

COVERAGE 2 :

Branch Coverage:				
Enabled Coverage	Bins	Hits	Misses	Coverage
-----	----	----	-----	-----
Branches	3	3	0	100.00%

Statement Coverage:				
Enabled Coverage	Bins	Hits	Misses	Coverage
-----	----	----	-----	-----
Statements	4	4	0	100.00%

Toggle Coverage:				
Enabled Coverage	Bins	Hits	Misses	Coverage
-----	----	----	-----	-----
Toggles	10	10	0	100.00%

VERIFICATION PLAN :

1	label	description	stimulus generation	function functionality check
2	dff_1	when the rst is asserted the output dout value must be low	directed at the start of the simulation	checker in the testbench to make sure the output is correct
3	dff_2	verifying random d and en and cheack output with golden model	directed at the start of the simulation	checker in the testbench to make sure the output is correct
4	dff_3	when the rst is asserted the output dout value must be low	directed at the start of the simulation	checker in the testbench to make sure the output is correct