# System Verilog

## Verification Plan & Subroutines

# Assignment 1 Extended

By: Magdy Ahmed Abbas Abdelhamid

# Question 1

## DFF

# Verification plan & Subroutines – Extra

1)

Verify the functionality of the following D-FF:

- **Inputs:** clk, rst (active high sync), d, en
- **Outputs:** q
- **Parameters:** USE_EN (if equals 1 then use the enable to update the q output when the reset is deasserted, else ignore the en input)

Since the design has a parameter, we need to verify the functionality of both modes when the USE_EN equals 1 and zero.

Requirements:

- You should create 2 testbenches (dff_t1_tb.sv & dff_t2_tb.sv) to verify the functionality of both modes.
- Use exhaustive test patterns where all combinations of inputs are tested using directed testing.
- Use a task to check the functionality of the design.
- Create a golden model reference (always block with to generate the correct expected functionality) inside of the testbench to compare the DUT output to your reference output.
- Use a do file
    - Compile the design and 2 testbenches
    - Run simulation for the first testbench
    - Close the simulation (quit -sim) to save the coverage database (dff_t1.ucdb) to test1
    - Run simulation for the second testbench
    - Close the simulation (quit -sim) to save the coverage database (dff_t2.ucdb) to test2
    - Use this command to merge the 2 databases generated
        - vcover merge dff_merged.ucdb dff_t1.ucdb dff_t2.ucdb -du dff
    - Generate a coverage report for the merged ucdb

# VERIFICATION PLAN

## Test Plan:

- **Do two Testbenches (USE_EN = 0) & (USE_EN = 1) Named (dff_t1_tb) & (dff_t2_tb) respectively.**
- **Golden Reference with q_expected Output in always block.**
- **Write two integer counters (error count) & (correct count).**
- **Clock Generation.**
- **Initialize counters to zero.**
- **Check Reset Values using (check reset) Task.**
- **Apply All Permutations of d & en to the DFF.**
- **In Each case do Check the DFF Output using (check result) Task.**
- **Check Reset Values using (check reset) Task.**
- **Toggling Achieved for All Interfaces (d, en, USE_EN, rst, clk, q).**

## Technique for Testing Feature:

- **Exhaustive Testing.**
- **Self - Checking.**
- **Verification IP.**
- **Block Level.**

# DFF RTL Design

## System Verilog Design Code:

```
module dff(clk, rst, d, q, en);

parameter USE_EN = 1;

input clk, rst, d, en;
output reg q;

always @(posedge clk) begin
  if (rst)
    q <= 0;
  else begin
    case(USE_EN)
      1'b0 : q <= d;
      1'b1 : begin
        if(en)
          q <= d;
      end
    endcase
  end
end

endmodule
```

## DFF Testbench tb1

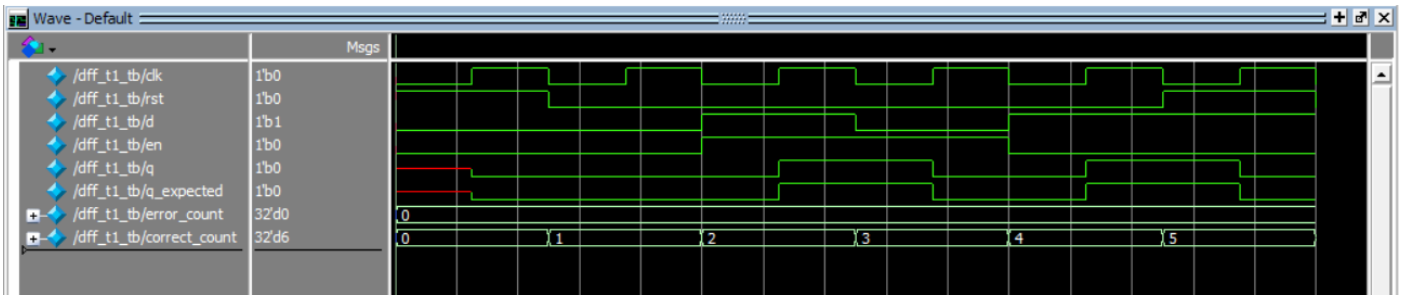
```
module dff_t1_tb;

parameter USE_EN = 0 ;

logic clk, rst, d, en , q , q_expected;

dff #(.USE_EN(USE_EN)) DUT1(.clk(clk),.rst(rst),.d(d),.en(en),.q(q));

integer error_count;   // 32 – Bit Signed
integer correct_count; // 32 – Bit Signed

initial begin
  clk = 0 ;
  forever
    #25 clk = ~clk;
end

always @ (posedge clk) begin
  if(rst)
    q_expected <= 0;
  else
    q_expected <= d;
end

initial begin
  error_count = 0;
  correct_count = 0;
  d = 0; en = 0;

  check_reset;

  d = 0; en = 0; check_result(q_expected);
  d = 1; en = 1; check_result(q_expected);
  d = 0; en = 1; check_result(q_expected);
  d = 1; en = 0; check_result(q_expected);
```

```
  // Toggle Closure for Reset, Reset Again?  maybe repeated code?
  check_reset;

  $display("%0t: At End of test error counter  =  %0d and
                  correct counter  =  %0d", $time, error_count, correct_count);
  $stop;
end

task check_result (input expected_result);
  @(negedge clk);
  if (q !==  q_expected) begin
    error_count  =  error_count  +  1;
    $display("%0t: Error: For D  =  %0b, and en  =  %0d and q should be
          equal to %0d but it is equal %0d", $time, d, en, expected_result, q);
  end
  else
    correct_count  =  correct_count  +  1;
endtask

task check_reset();
  // Assert reset
  rst  =  1 ;
  @(negedge clk);
  if (q !=  0) begin
    error_count  =  error_count  +  1;
    $display("%0t: Error: Reset Value is Asserted and the Output is
                                                    Not Tied to Low", $time);
  end
  else
    correct_count  =  correct_count  +  1;
  // desert reset
  rst  =  0 ;
endtask

endmodule
```

## Do File for (DFF tb1) Question 1

```
vlib work
vlog dff.v dff_t1_tb.sv dff_t2_tb.sv +cover -covercells
vsim -voptargs=+acc work.dff_t1_tb -cover
add wave *
coverage save dff_t1_tb.ucdb -onexit -du work.dff
run -all
# quit -sim
# vcover report dff_t1_tb.ucdb -details -annotate
-all -output dff_t1_Coverage_Rpt.txt
```

## WAVEFORM




```
# 300: At End of test error counter = 0 and correct counter = 6
```

## DFF Testbench tb2

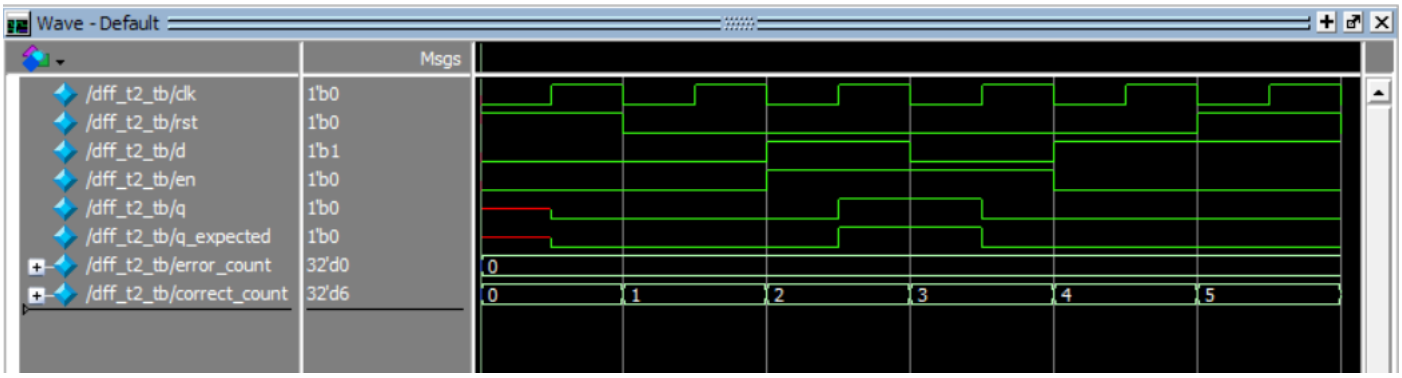
```
module dff_t2_tb;

parameter USE_EN = 1;

logic clk, rst, d, en, q, q_expected;

dff #(.USE_EN(USE_EN)) DUT2(.clk(clk),.rst(rst),.d(d),.en(en),.q(q));

integer error_count;   // 32 - Bit Signed
integer correct_count; // 32 - Bit Signed

initial begin
  clk = 0;
  forever
    #25 clk = ~clk;
end

always @ (posedge clk) begin
  if(rst)
    q_expected <= 0;
  else
  if(en)
    q_expected <= d;
end

initial begin
  error_count = 0;
  correct_count = 0;

  d = 0; en = 0;
  check_reset;
  d = 0; en = 0; check_result(q_expected);
  d = 1; en = 1; check_result(q_expected);
  d = 0; en = 1; check_result(q_expected);
  d = 1; en = 0; check_result(q_expected);
```

```
  // Toggle Closure for Reset, Reset Again?  maybe repeated code?
  check_reset;

  $display("%0t: At End of test error counter  =  %0d and
                  correct counter  =  %0d", $time, error_count, correct_count);
  $stop;
end


task check_result (input expected_result);
  @(negedge clk);
  if (q ! ==  q_expected) begin
    error_count  =  error_count  +  1;
    $display("%0t: Error: For D  =  %0b, and en  =  %0d and q should be
          equal to %0d but it is equal %0d", $time, d, en, expected_result, q);
  end
  else
    correct_count  =  correct_count  +  1;
endtask

task check_reset();
  // Assert reset
  rst  =  1 ;
  @(negedge clk);
  if (q ! =  0) begin
    error_count  =  error_count  +  1;
    $display("%0t: Error: Reset Value is Asserted and the Output is
                                                    Not Tied to Low", $time);
  end
  else
    correct_count  =  correct_count  +  1;
  // desert reset
  rst  =  0 ;
endtask

endmodule
```

## Do File for (DFF tb2) Question 1

```
vlib work
vlog dff.v dff_t1_tb.sv dff_t2_tb.sv  +cover  -covercells
vsim  -voptargs = +acc work.dff_t2_tb  -cover
add wave  *
coverage save dff_t2_tb.ucdb  -onexit  -du work.dff
run  -all
# quit  -sim
# vcover report dff_t2_tb.ucdb  -details  -annotate
-all  -output dff_t2_Coverage_Rpt.txt

# vcover merge dff_magdy.ucdb dff_t1_tb.ucdb  dff_t2_tb.ucdb  -du work.dff

# vcover report dff_magdy.ucdb  -details  -annotate
-all  -output dff_Coverage_Rpt.txt
```

## WAVEFORM




```
# 300: At End of test error counter = 0 and correct counter = 6
```

# Code Coverage

dff_Coverage_Rpt.txt

```
1   Coverage Report by instance with details
2
3   ================================================================================
4   === Instance: /\work.dff
5   === Design Unit: work.dff
6   ================================================================================
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      9                                        12     Count coming in to IF
21      9              1                          4        if (rst)
22
23      11             1                          4        else begin
24
25      15             1                          2                if(en)
26
27                                                2     All False Count
28  Branch totals: 4 hits of 4 branches = 100.00%
```

dff_Coverage_Rpt.txt

```
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
46
47      3                                                parameter USE_EN = 1 ;
48
49      4
50
51      5                                                input clk, rst, d, en;
52
53      6                                                output reg q;
54
55      7
56
57      8              1                          12     always @(posedge clk) begin
58
59      9                                                   if (rst)
60
61      10             1                          4          q <= 0;
62
63      11                                                  else begin
```

```
62
63      11                                              else begin
64
65      12                                                 case(USE_EN)
66
67      13              1                       4             1'b0 : q <= d ;
68
69      14                                                    1'b1 : begin
70
71      15                                                       if(en)
72
73      16              1                       2                   q <= d ;
74
75
76    Toggle Coverage:
77        Enabled Coverage              Bins      Hits    Misses  Coverage
78        ----------------              ----      ----    ------  --------
79        Toggles                         10        10         0   100.00%
80
```

```
81    ================================Toggle Details================================
82
83    Toggle Coverage for instance /\work.dff  --
84
85                                             Node      1H->0L      0L->1H  "Coverage"
86                                             ----------------------------------------
87                                              clk           2           2      100.00
88                                                d           2           2      100.00
89                                               en           2           2      100.00
90                                                q           2           2      100.00
91                                              rst           2           2      100.00
92
93    Total Node Count     =          5
94    Toggled Node Count   =          5
95    Untoggled Node Count =          0
96
97    Toggle Coverage      =     100.00% (10 of 10 bins)
98
99
100   Total Coverage By Instance (filtered view): 100.00%
101
102
```