Assignment 1 extra

Digital Design Verification

Contents

1 DFF "D Flip Flop"

1.1 1 Design

```
// bug : missing begin and end of if and else statements
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;
   end else begin
      if(USE_EN) begin
         if (en)
            q <= d;
      end else
         q <= d;
   end
end

endmodule
```

1.2 2. Verification Plan

Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
DFF_USE_EN_1	When USE_EN = 1, q should follow d only when en is high	Exhaustive test patterns (00, 01, 10, 11)	-	A checker in the testbench to validate q updates based on en
DFF_USE_EN_0	When USE_EN = 0, q should always follow d, regardless of en	Exhaustive test patterns (00, 01, 10, 11, x)	-	A checker in the testbench to validate q always takes d, ignoring en
DFF_RESET	When reset is asserted, q should be set to 0	Directed test — apply reset	-	A checker in the testbench to validate q is 0 after reset is asserted

Table 1: DFF Testbench Design Requirements

1.3 3 Testbench

```
// Testbench for USE_EN = 1
module dff_tb1;

    //------------------------------------------
    // Declare signals
    //------------------------------------------
    reg clk ,rst ,d ,en;
    wire q;
    parameter USE_EN = 1;

    //------------------------------------------
    // Instantiate the DUT (Device Under Test)
    //------------------------------------------
    dff #(USE_EN) dut (.*);

    //------------------------------------------
    // Clock generation
    //------------------------------------------
    parameter CLOCK_PERIOD = 10;
    always begin
        #(CLOCK_PERIOD/2) clk = ~clk;
    end

    //------------------------------------------
    // Reset task
    //------------------------------------------
    task reset_dut;
        begin
            // Assert reset
            rst = 1;   // syncgronous reset
            @(negedge clk);

            // Check output reset to zero
            if (q != 1'b0) begin
                $display("Reset test failed: rst=%0b , d=%0b , en=%0b , q=%0b", rst, d, en, q);
                $stop;
            end else begin
                $display("Reset test passed: rst=%0b , d=%0b , en=%0b , q=%0b", rst, d, en, q);
            end

            // Deassert reset
            #(CLOCK_PERIOD/2);
            rst = 0;
        end
    endtask

   //------------------------------------------
   // Functionality check task
   //------------------------------------------
    task check_functionality;
        input reg d_in, en_in;
        begin
            d = d_in;
            en = en_in;
            @(negedge clk);
            if (en) begin
```

```
                if (q != d) begin
                    $display("[USE_EN=1] Test failed: rst=%0b, d=%0b, en=%0b, q=%0b", rst, d, en, q);
                    $stop;
                end else
                    $display("[USE_EN=1] All tests passed!: rst=%0b, d=%0b, en=%0b, q=%0b", rst, d, en, q);
                #(CLOCK_PERIOD/2);
            end else begin
                if (q != q) begin
                    $display("[USE_EN=1] Test failed: rst=%0b, d=%0b, en=%0b, q=%0b", rst, d, en, q);
                    $stop;
                end else
                    $display("[USE_EN=1] All tests passed!: rst=%0b, d=%0b, en=%0b, q=%0b", rst, d, en, q);
                #(CLOCK_PERIOD/2);
            end
        end
    endtask


    initial begin
        //-------------------------------------
        // Initialize signals
        //-------------------------------------
        clk = 1'b1;
        rst = 1'b0;
        d = 1;
    en = 1;

        #(CLOCK_PERIOD);

        //---------------------------------
        // Check reset functionality
        //---------------------------------
        reset_dut;

        // Exhaustive test patterns
        check_functionality(1'b0,1'b0);
        check_functionality(1'b0,1'b1);
        check_functionality(1'b1,1'b0);
        check_functionality(1'b1,1'b1);

        $finish;
     end

endmodule

//*****************************************************************************************//
//*****************************************************************************************//
//*****************************************************************************************//

// Testbench for USE_EN = 0
module dff_tb2;

    //-----------------------------------------
    // Declare signals
    //-----------------------------------------
    reg clk ,rst ,d ,en;
    wire q;
    parameter USE_EN = 0;

    //-----------------------------------------
    // Instantiate the DUT (Device Under Test)
    //-----------------------------------------
    dff #(USE_EN) dut (.*);

    //-----------------------------------------
    // Clock generation
    //-----------------------------------------
    parameter CLOCK_PERIOD = 10;
    always begin
        #(CLOCK_PERIOD/2) clk = ~clk;
    end

    //-----------------------------------------
    // Reset task
    //-----------------------------------------
    task reset_dut;
        begin
            // Assert reset
            rst = 1;    // syncgronous reset
            @(negedge clk);

            // Check output reset to zero
            if (q != 1'b0) begin
                $display("Reset test failed: rst=%0b , d=%0b , en=%0b , q=%0b", rst, d, en, q);
                $stop;
            end else begin
                $display("Reset test passed: rst=%0b , d=%0b , en=%0b , q=%0b", rst, d, en, q);
            end

            // Deassert reset
            #(CLOCK_PERIOD/2);
            rst = 0;
        end
    endtask

   //-----------------------------------------
   // Functionality check task
   //-----------------------------------------
    task check_functionality;
        input reg d_in, en_in;
```

```
        begin
            d = d_in;
            en = en_in;
            @(negedge clk);
            // check it's completely don't care about [en] signal
            if (en || !en || en === 1'bx) begin
                if (q != d) begin
                    $display("[USE_EN=0] Test failed: rst=%0b, d=%0b, en=%0b, q=%0b", rst, d, en, q);
                    $stop;
                end else
                    $display("[USE_EN=0] All tests passed!: rst=%0b, d=%0b, en=%0b, q=%0b", rst, d, en, q);
            end
            #(CLOCK_PERIOD/2);
        end
    endtask


    initial begin
        //-------------------------------------
        // Initialize signals
        //-------------------------------------
        clk = 1'b1;
        rst = 1'b0;
        d = 1;
      en = 1;

        #(CLOCK_PERIOD);

        //---------------------------------
        // Check reset functionality
        //---------------------------------
        reset_dut;

        // Exhaustive test patterns
        check_functionality(1'b0,1'b0);
        check_functionality(1'b0,1'b1);
        check_functionality(1'b1,1'b0);
        check_functionality(1'b1,1'b1);
        check_functionality(1'b1,1'bx);
        check_functionality(1'b0,1'bx);

        $finish;
      end

endmodule
```

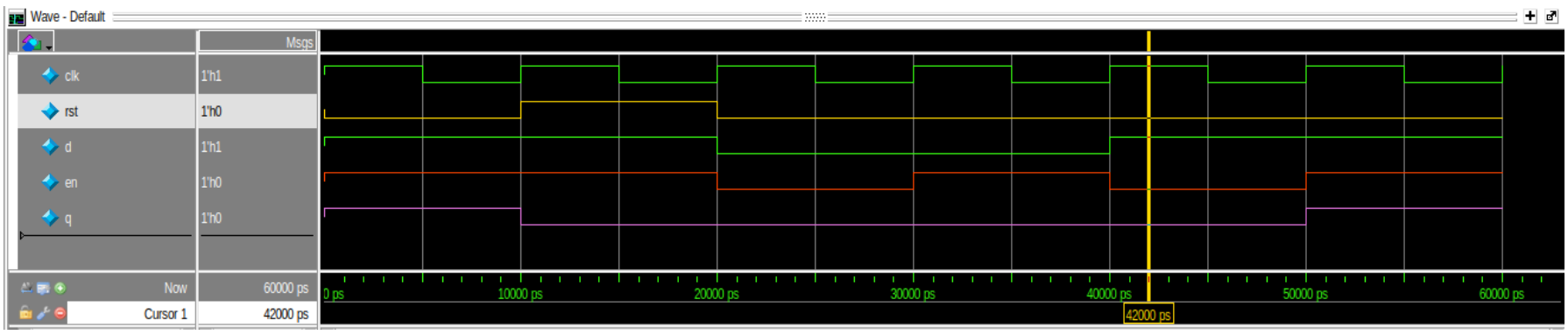

Figure 1: TB1 waveform [use_en = 1]

```
# vsim -voptargs="+acc" dff_tb1 -coverage
# Start time: 11:31:49 on Mar 06,2025
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading sv_std.std
# Loading work.dff_tb1(fast)
# Loading work.dff(fast)
# Reset test passed: rst=1 , d=1 , en=1 , q=0
# [USE_EN=1] All tests passed!: rst=0, d=0, en=0, q=0
# [USE_EN=1] All tests passed!: rst=0, d=0, en=1, q=0
# [USE_EN=1] All tests passed!: rst=0, d=1, en=0, q=0
# [USE_EN=1] All tests passed!: rst=0, d=1, en=1, q=1
# ** Note: $finish    : dff_tb1.sv(97)
#    Time: 60 ns  Iteration: 0  Instance: /dff_tb1
# 1
# Break in Module dff_tb1 at dff_tb1.sv line 97
```

Figure 2: TB1 transcript [use_en = 1]

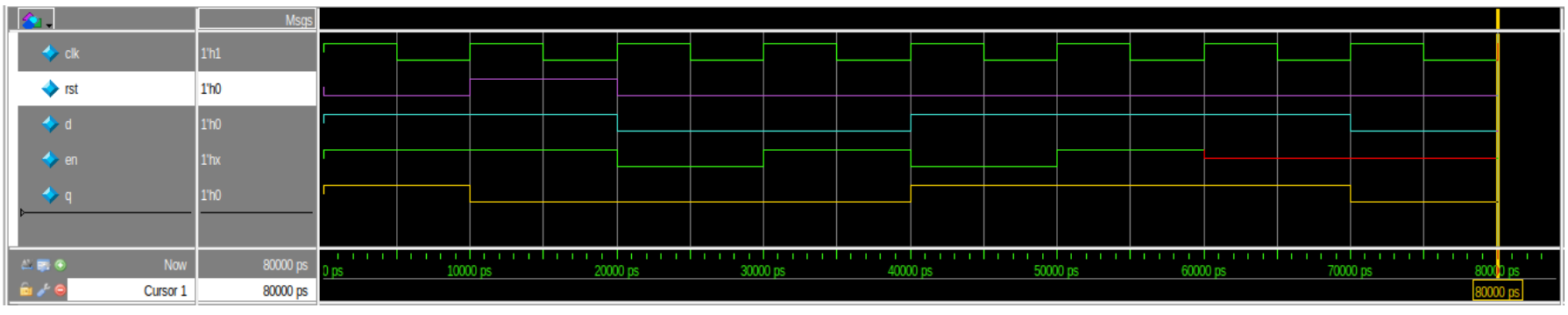

Figure 3: TB2 waveform [use_en = 0]

```
# vsim -voptargs="+acc" dff_tb2 -coverage
# Start time: 11:49:49 on Mar 06,2025
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading sv_std.std
# Loading work.dff_tb2(fast)
# Loading work.dff(fast)
# Reset test passed: rst=1 , d=1 , en=1 , q=0
# [USE_EN=0] All tests passed!: rst=0, d=0, en=0, q=0
# [USE_EN=0] All tests passed!: rst=0, d=0, en=1, q=0
# [USE_EN=0] All tests passed!: rst=0, d=1, en=0, q=1
# [USE_EN=0] All tests passed!: rst=0, d=1, en=1, q=1
# [USE_EN=0] All tests passed!: rst=0, d=1, en=x, q=1
# [USE_EN=0] All tests passed!: rst=0, d=0, en=x, q=0
# ** Note: $finish    : dff_tb2.sv(92)
#    Time: 80 ns  Iteration: 0  Instance: /dff_tb2
# 1
# Break in Module dff_tb2 at dff_tb2.sv line 92
```

Figure 4: TB2 transcript [use_en = 0]

1.4 4 Do File

```
vlib work
vlog dff.v dff_tb1.sv dff_tb2.sv +cover -covercells

# Run testbench for USE_EN = 1
proc run_tb1 {} {
    vsim -voptargs=+acc dff_tb1 -cover
    add wave *
    coverage save dff_tb1.ucdb -du dff -onexit
    run -all
}

# Run testbench for USE_EN = 0
proc run_tb2 {} {
    vsim -voptargs=+acc dff_tb2 -cover
    add wave *
    coverage save dff_tb2.ucdb -du dff -onexit
    run -all
}

# Choose testbench to run
puts "Select testbench to run: 1 for USE_EN=1, 2 for USE_EN=0"
set choice [gets stdin]
if {$choice == 1} {
    run_tb1
} elseif {$choice == 2} {
    run_tb2
} else {
    puts "Invalid choice. Exiting."
}

# note : this do file to run tb according to input argument when run command do run.txt inside questasim
```

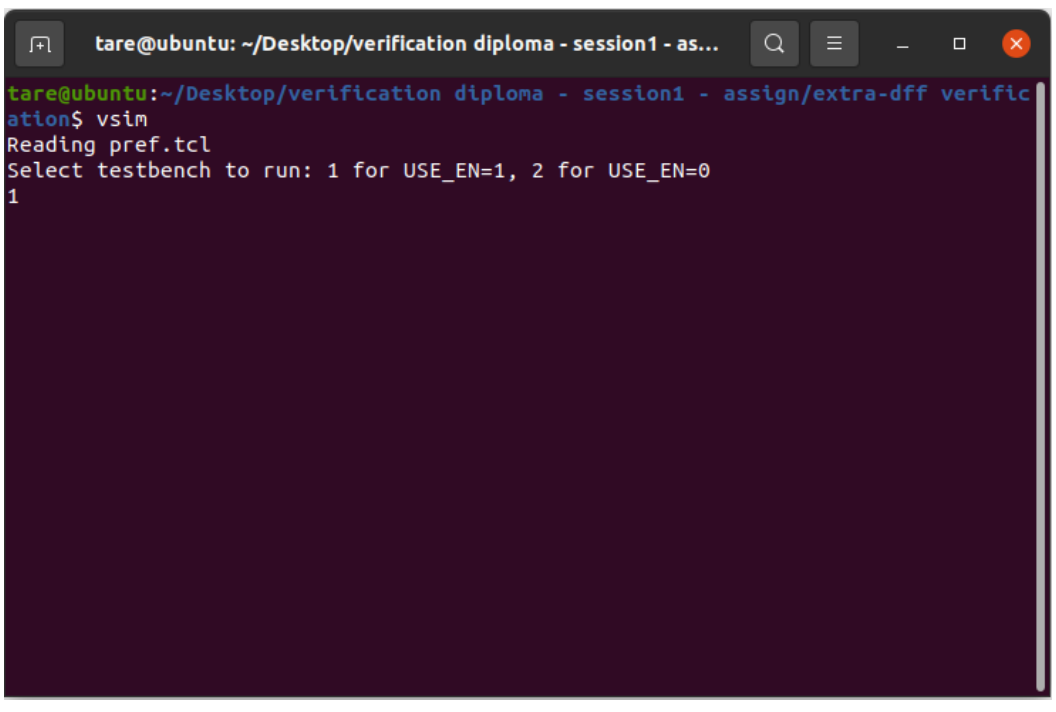

Figure 5: first run dor TB1 using ubuntu 20.04 termianl

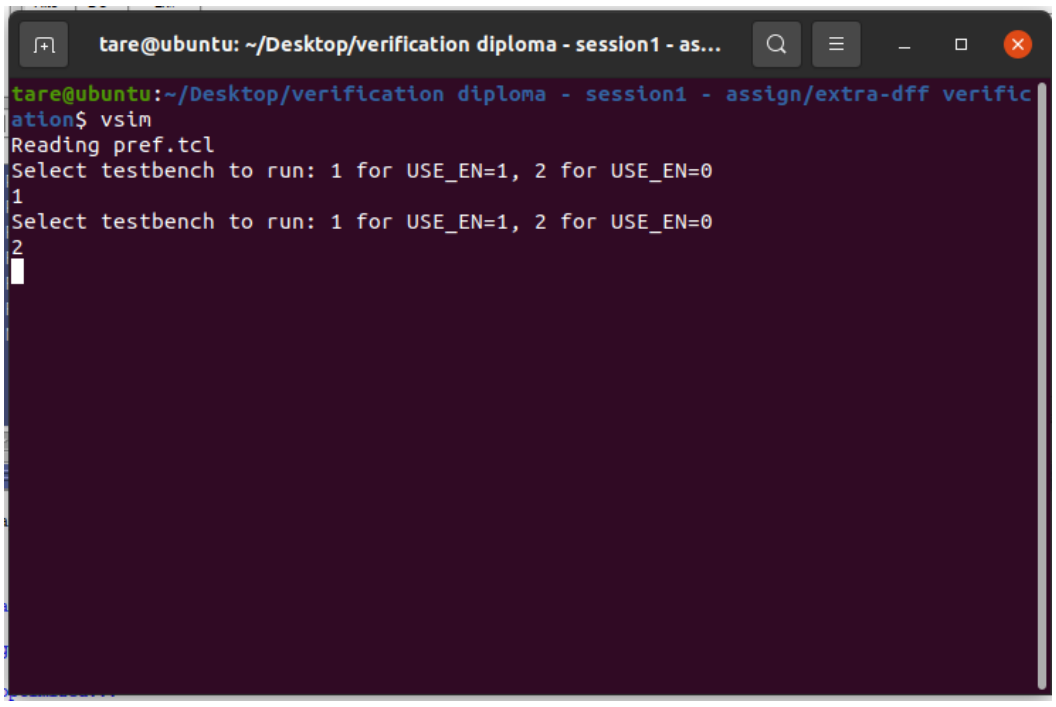

Figure 6: second run for TB2 using ubuntu 20.04 termianl

1.5 5 Coverage Report

Coverage Report by instance with details

```
================================================================================
=== Instance: /\work.dff
=== Design Unit: work.dff
================================================================================
Branch Coverage:
    Enabled Coverage                 Bins      Hits    Misses  Coverage
    ----------------                 ----      ----    ------  --------
    Branches                            4         4         0   100.00%

================================Branch Details================================

Branch Coverage for instance /\work.dff

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File dff.v
----------------------------------IF Branch-----------------------------------
    7                                        12     Count coming in to IF
    7               1                         2        if (rst) begin
    11              1                         3             if (en)
                                              2     All False Count
    9               1                         5        end else begin
Branch totals: 4 hits of 4 branches = 100.00%


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

================================Statement Details================================

Statement Coverage for instance /\work.dff  --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File dff.v
    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               1                        12     always @(posedge clk) begin
    7                                                  if (rst) begin
    8               1                         2           q <= 0;
    9                                                  end else begin
    10                                                    if(USE_EN) begin
    11                                                       if (en)
    12              1                         3                q <= d;
    13                                                    end else
    14              1                         5              q <= d;

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

================================Toggle Details================================

Toggle Coverage for instance /\work.dff  --

                                      Node      1H->0L      0L->1H  "Coverage"
                                      ----      ------      ------  ----------
```

clk	2	2	100.00
d	2	2	100.00
en	2	2	100.00
q	2	2	100.00
rst	2	2	100.00

Total Node Count = 5
Toggled Node Count = 5
Untoggled Node Count = 0

Toggle Coverage = 100.00% (10 of 10 bins)

Total Coverage By Instance (filtered view): 100.00%