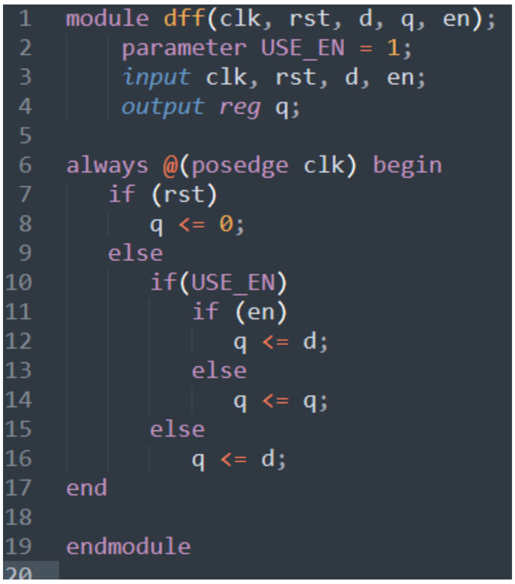
Assignment 1 Extended

By: Abdelrahman Maher Hassan

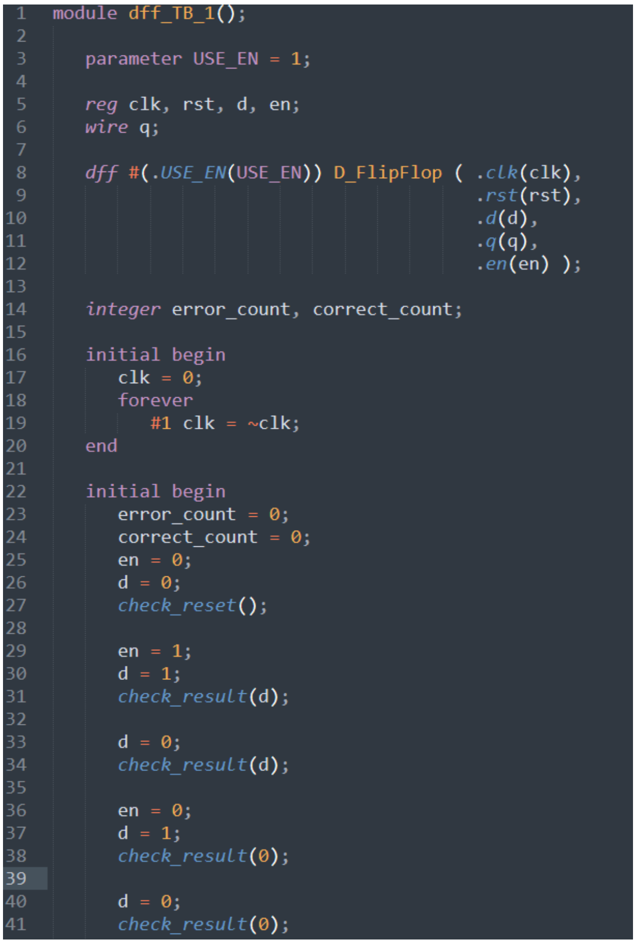
Question 1

Codes:

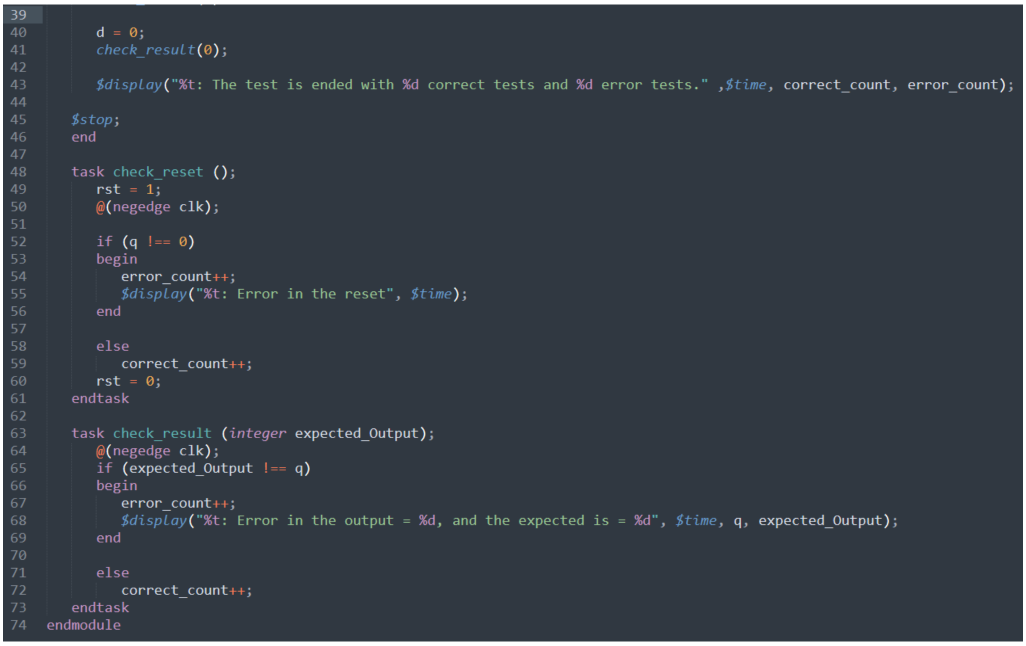


```
module dff(clk, rst, d, q, en);
    parameter USE_EN = 1;
    input clk, rst, d, en;
    output reg q;

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

endmodule
```

Test Bench 1:

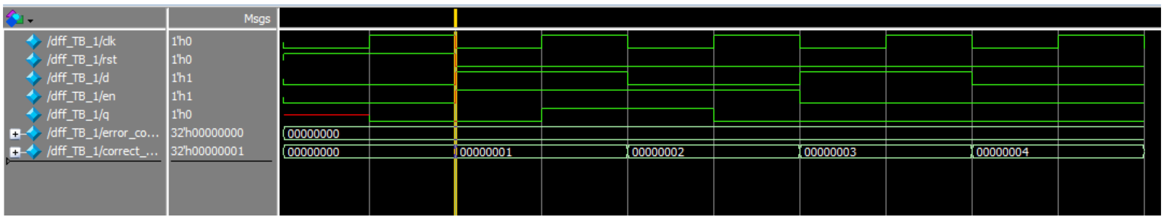


```
module dff_TB_1();

   parameter USE_EN = 1;

   reg clk, rst, d, en;
   wire q;

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

   integer error_count, correct_count;

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

   initial begin
      error_count = 0;
      correct_count = 0;
      en = 0;
      d = 0;
      check_reset();

      en = 1;
      d = 1;
      check_result(d);

      d = 0;
      check_result(d);

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

      d = 0;
      check_result(0);
```

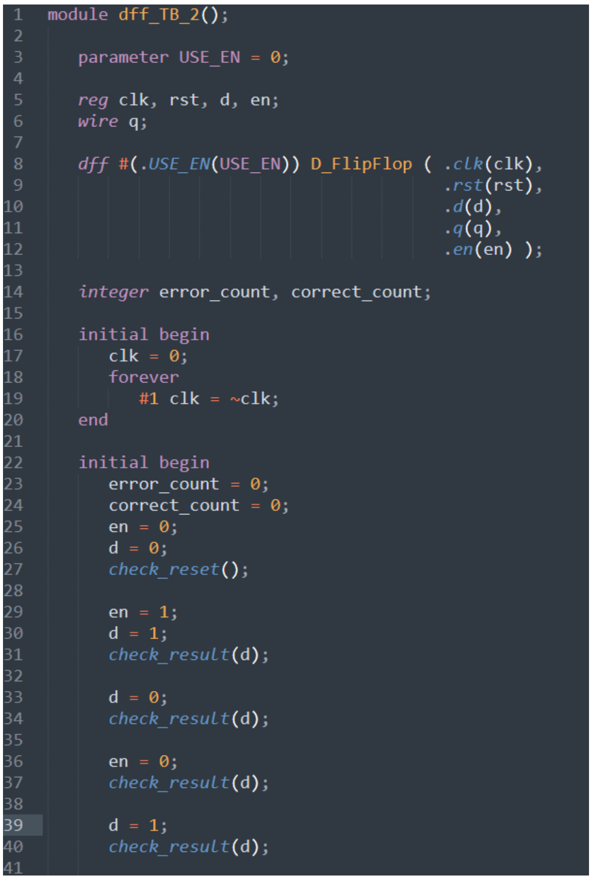


```
        d = 0;
        check_result(0);

        $display("%t: The test is ended with %d correct tests and %d error tests." ,$time, correct_count, error_count);

    $stop;
    end

    task check_reset ();
        rst = 1;
        @(negedge clk);

        if (q !== 0)
        begin
            error_count++;
            $display("%t: Error in the reset", $time);
        end

        else
            correct_count++;
        rst = 0;
    endtask

    task check_result (integer expected_Output);
        @(negedge clk);
        if (expected_Output !== q)
        begin
            error_count++;
            $display("%t: Error in the output = %d, and the expected is = %d", $time, q, expected_Output);
        end

        else
            correct_count++;
    endtask
endmodule
```

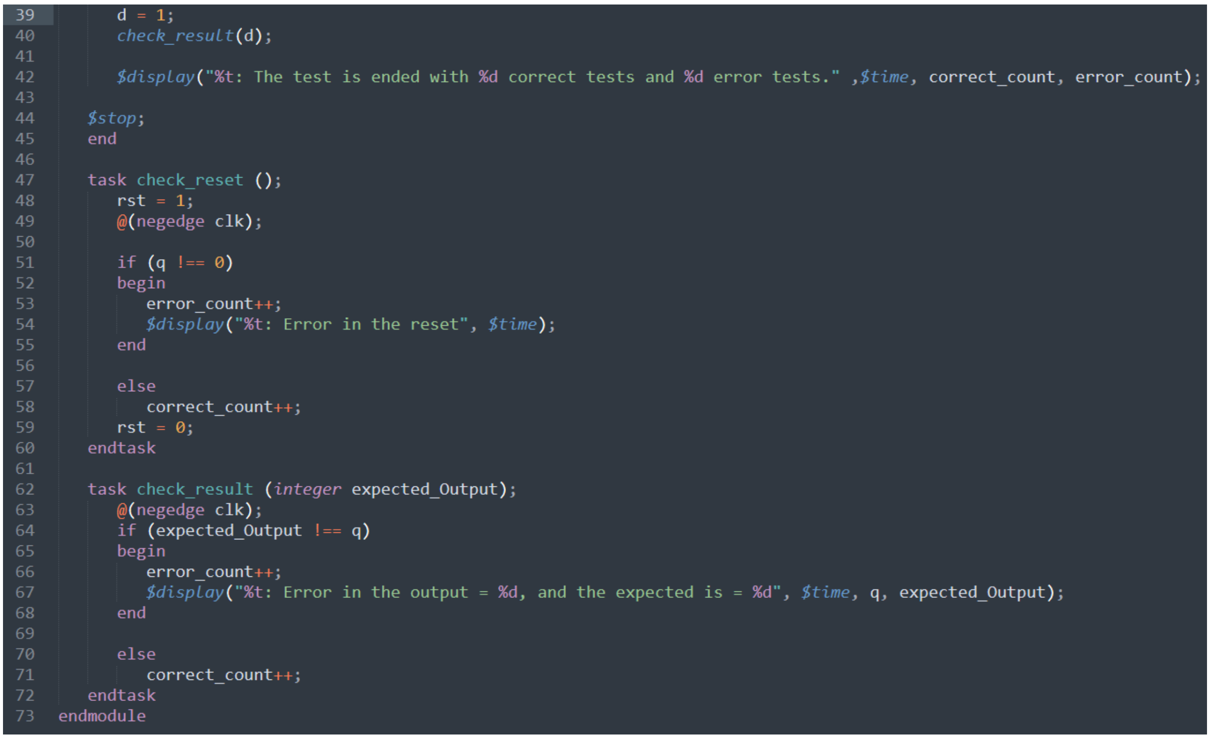
Simulation 1:



Test Bench 2:

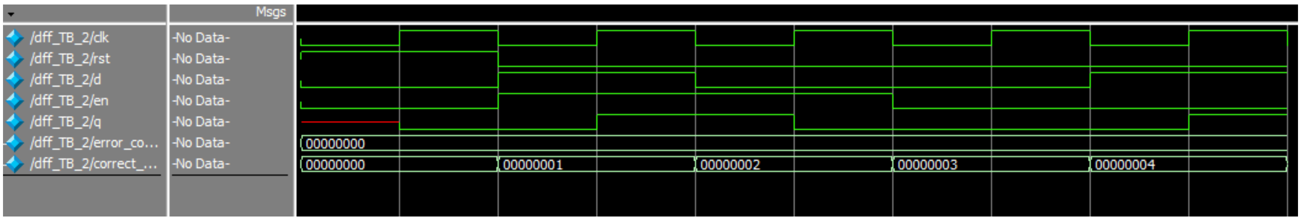


```verilog
module dff_TB_2();

    parameter USE_EN = 0;

    reg clk, rst, d, en;
    wire q;

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

    integer error_count, correct_count;

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

    initial begin
        error_count = 0;
        correct_count = 0;
        en = 0;
        d = 0;
        check_reset();

        en = 1;
        d = 1;
        check_result(d);

        d = 0;
        check_result(d);

        en = 0;
        check_result(d);

        d = 1;
        check_result(d);

```



```verilog
        d = 1;
        check_result(d);

        $display("%t: The test is ended with %d correct tests and %d error tests." ,$time, correct_count, error_count);

    $stop;
    end

    task check_reset ();
        rst = 1;
        @(negedge clk);

        if (q !== 0)
        begin
            error_count++;
            $display("%t: Error in the reset", $time);
        end

        else
            correct_count++;
        rst = 0;
    endtask

    task check_result (integer expected_Output);
        @(negedge clk);
        if (expected_Output !== q)
        begin
            error_count++;
            $display("%t: Error in the output = %d, and the expected is = %d", $time, q, expected_Output);
        end

        else
            correct_count++;
    endtask
endmodule
```

Simulation 2:



Msgs
/dff_TB_2/clk
/dff_TB_2/rst
/dff_TB_2/d
/dff_TB_2/en
/dff_TB_2/q
/dff_TB_2/error_co...
/dff_TB_2/correct_...
-No Data-
-No Data-
-No Data-
-No Data-
-No Data-
-No Data-
-No Data-
00000000
00000000
00000001
00000002
00000003
00000004