# Digital Design Diploma

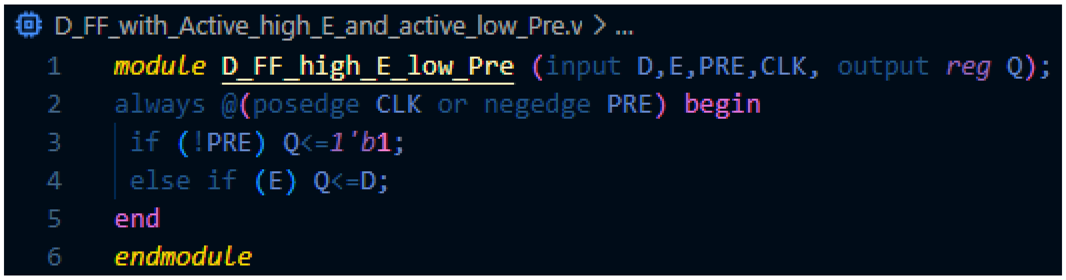
## Assignment 3

## Sequential Logic Design

| | |
|---|---|
| **Name** | كريم حسن عاطف علي |
| **Group** | G2 |

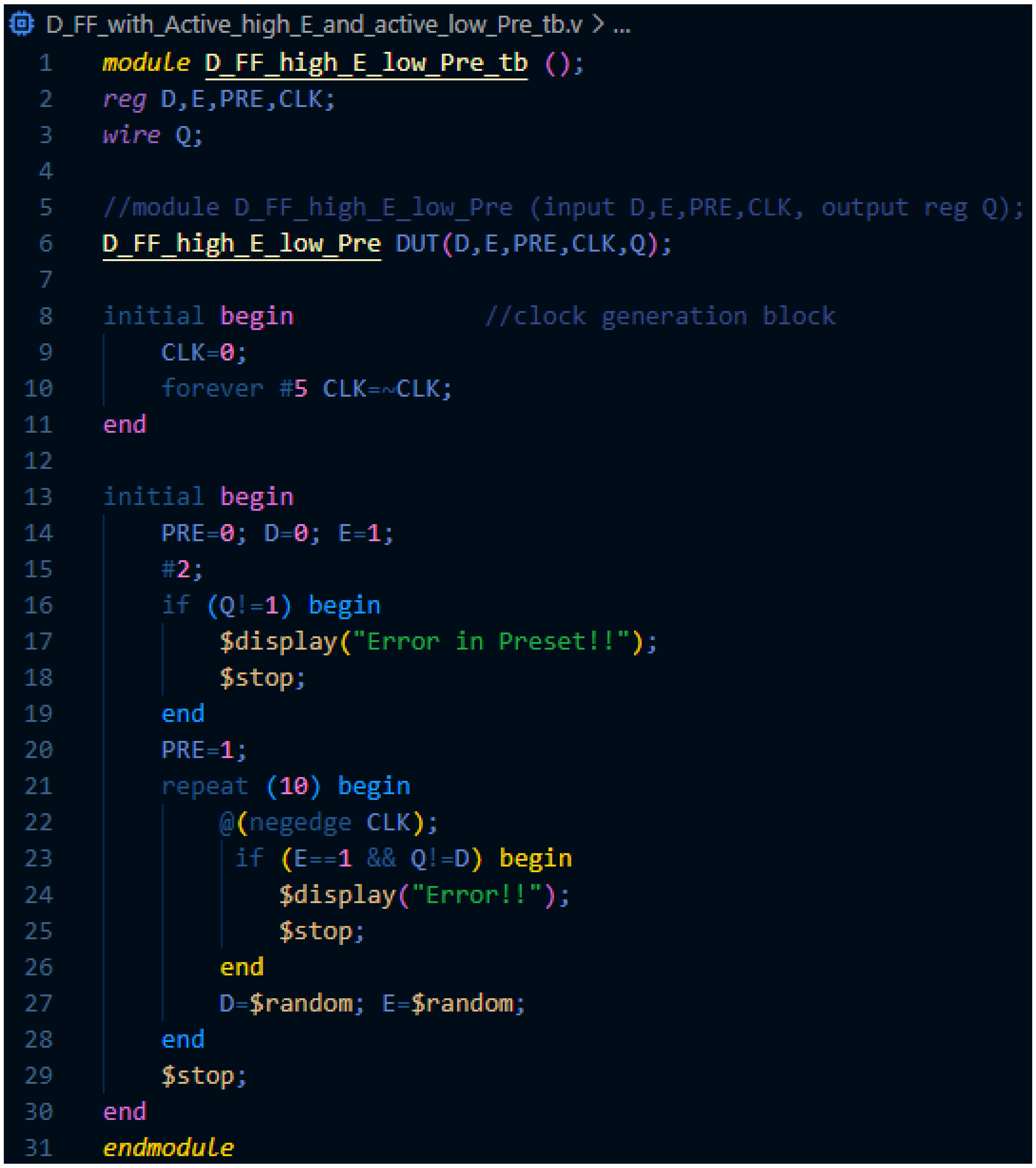
**Submitted to: Eng. Kareem Waseem**

## 1) DFF with clock enable and PRE control signal:


D_FF_with_Active_high_E_and_active_low_Pre.v > ...

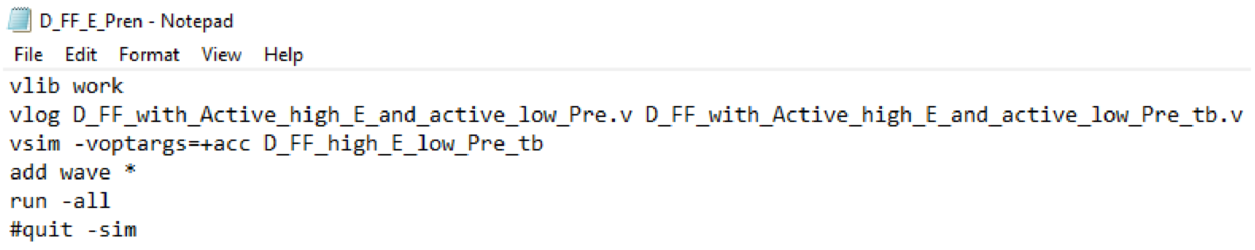
```verilog
module D_FF_high_E_low_Pre (input D,E,PRE,CLK, output reg Q);
always @(posedge CLK or negedge PRE) begin
  if (!PRE) Q<=1'b1;
  else if (E) Q<=D;
end
endmodule
```

*Figure1:  Q1 Code*


D_FF_with_Active_high_E_and_active_low_Pre_tb.v > ...

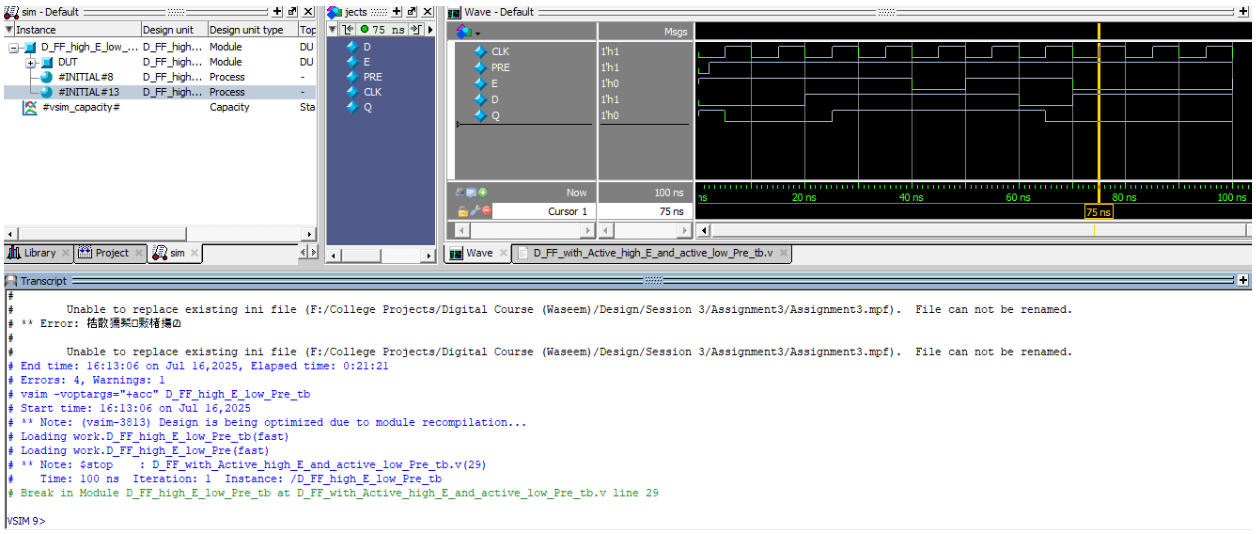
```verilog
module D_FF_high_E_low_Pre_tb ();
reg D,E,PRE,CLK;
wire Q;

//module D_FF_high_E_low_Pre (input D,E,PRE,CLK, output reg Q);
D_FF_high_E_low_Pre DUT(D,E,PRE,CLK,Q);

initial begin             //clock generation block
    CLK=0;
    forever #5 CLK=~CLK;
end

initial begin
    PRE=0; D=0; E=1;
    #2;
    if (Q!=1) begin
        $display("Error in Preset!!");
        $stop;
    end
    PRE=1;
    repeat (10) begin
        @(negedge CLK);
         if (E==1 && Q!=D) begin
            $display("Error!!");
            $stop;
        end
        D=$random; E=$random;
    end
    $stop;
end
endmodule
```

*Figure 2:  Q1 Testbench*



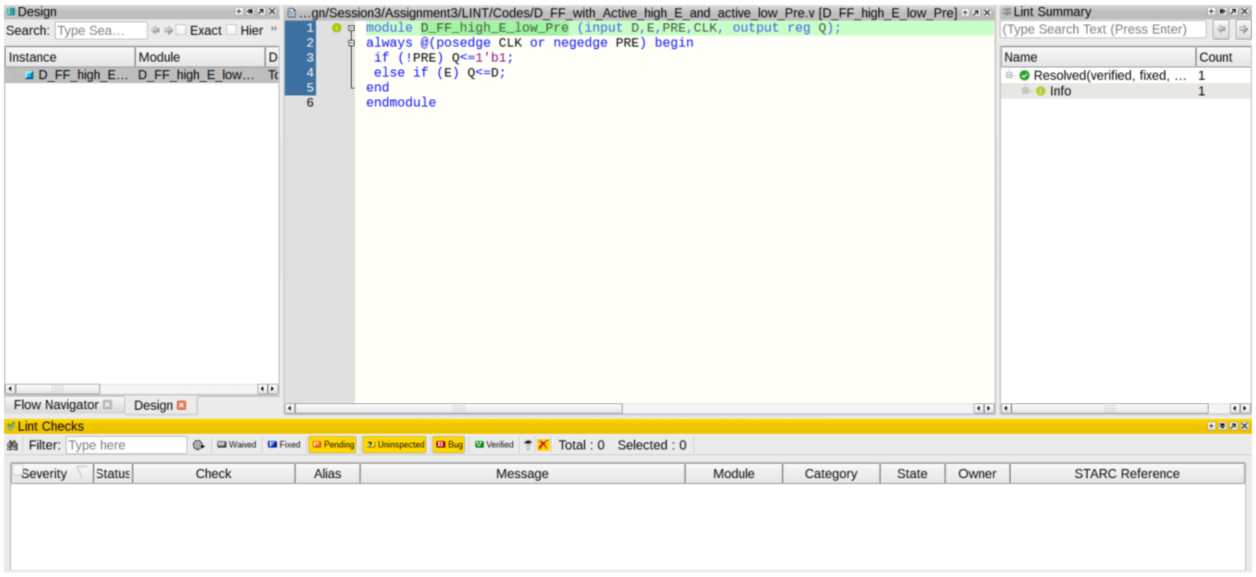
```
vlib work
vlog D_FF_with_Active_high_E_and_active_low_Pre.v D_FF_with_Active_high_E_and_active_low_Pre_tb.v
vsim -voptargs=+acc D_FF_high_E_low_Pre_tb
add wave *
run -all
#quit -sim
```

*Figure 3: Q1 DO File*



```
#
#        Unable to replace existing ini file (F:/College Projects/Digital Course (Waseem)/Design/Session 3/Assignment3/Assignment3.mpf).  File can not be renamed.
# ** Error: 梏散獯桀□敦楮擖ø
#
#        Unable to replace existing ini file (F:/College Projects/Digital Course (Waseem)/Design/Session 3/Assignment3/Assignment3.mpf).  File can not be renamed.
# End time: 16:13:06 on Jul 16,2025, Elapsed time: 0:21:21
# Errors: 4, Warnings: 1
# vsim -voptargs="+acc" D_FF_high_E_low_Pre_tb
# Start time: 16:13:06 on Jul 16,2025
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading work.D_FF_high_E_low_Pre_tb(fast)
# Loading work.D_FF_high_E_low_Pre(fast)
# ** Note: $stop    : D_FF_with_Active_high_E_and_active_low_Pre_tb.v(29)
#    Time: 100 ns  Iteration: 1  Instance: /D_FF_high_E_low_Pre_tb
# Break in Module D_FF_high_E_low_Pre_tb at D_FF_with_Active_high_E_and_active_low_Pre_tb.v line 29

VSIM 9>
```

*Figure 4: Q1 Wave*

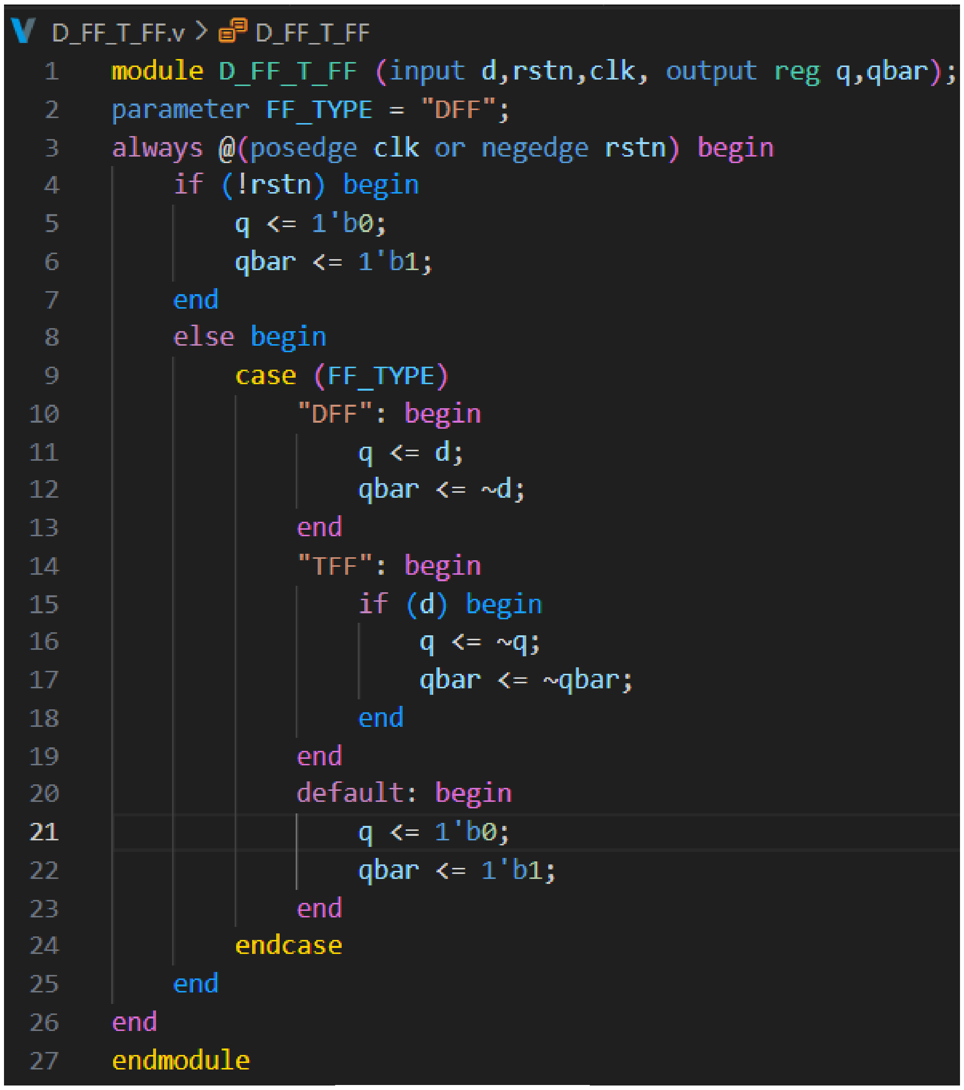


...gn/Session3/Assignment3/LINT/Codes/D_FF_with_Active_high_E_and_active_low_Pre.v [D_FF_high_E_low_Pre]

```
module D_FF_high_E_low_Pre (input D,E,PRE,CLK, output reg Q);
always @(posedge CLK or negedge PRE) begin
 if (!PRE) Q<=1'b1;
 else if (E) Q<=D;
end
endmodule
```

*Figure 5: Q1 Linting*

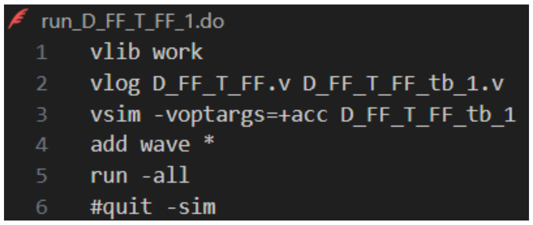
## 2) Parameterized Asynchronous FF with Active low rst (D FF & T FF):



D_FF_T_FF.v > D_FF_T_FF

```verilog
module D_FF_T_FF (input d,rstn,clk, output reg q,qbar);
parameter FF_TYPE = "DFF";
always @(posedge clk or negedge rstn) begin
    if (!rstn) begin
        q <= 1'b0;
        qbar <= 1'b1;
    end
    else begin
        case (FF_TYPE)
            "DFF": begin
                q <= d;
                qbar <= ~d;
            end
            "TFF": begin
                if (d) begin
                    q <= ~q;
                    qbar <= ~qbar;
                end
            end
            default: begin
                q <= 1'b0;
                qbar <= 1'b1;
            end
        endcase
    end
end
endmodule
```

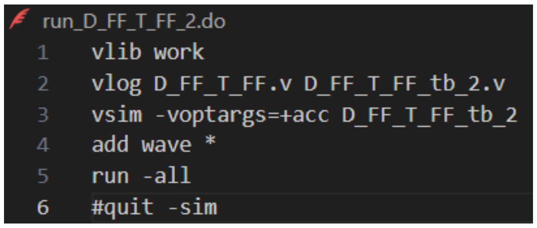
*Figure 6: Q2 Code*



run_D_FF_T_FF_1.do

```
vlib work
vlog D_FF_T_FF.v D_FF_T_FF_tb_1.v
vsim -voptargs=+acc D_FF_T_FF_tb_1
add wave *
run -all
#quit -sim
```

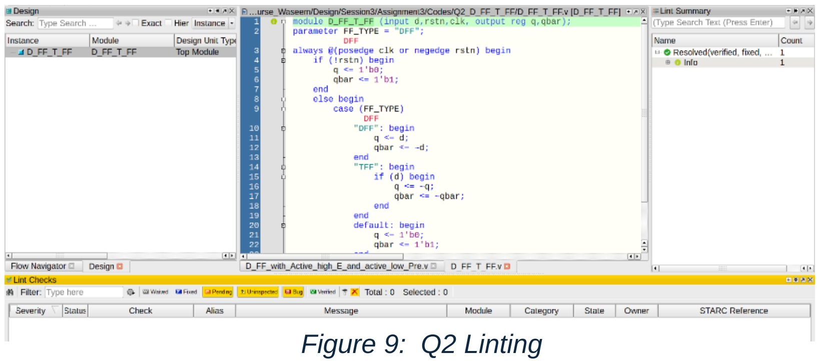
*Figure 7: Q2 DO File (DFF)*



run_D_FF_T_FF_2.do

```
vlib work
vlog D_FF_T_FF.v D_FF_T_FF_tb_2.v
vsim -voptargs=+acc D_FF_T_FF_tb_2
add wave *
run -all
#quit -sim
```

*Figure 8: Q2 DO File (TFF)*



...urse_Waseem/Design/Session3/Assignment3/Codes/Q2_D_FF_T_FF/D_FF_T_FF.v [D_FF_T_FF]

```verilog
module D_FF_T_FF (input d,rstn,clk, output reg q,qbar);
parameter FF_TYPE = "DFF";
                    DFF
always @(posedge clk or negedge rstn) begin
    if (!rstn) begin
        q <= 1'b0;
        qbar <= 1'b1;
    end
    else begin
        case (FF_TYPE)
            DFF
            "DFF": begin
                q <= d;
                qbar <= ~d;
            end
            "TFF": begin
                if (d) begin
                    q <= ~q;
                    qbar <= ~qbar;
                end
            end
            default: begin
                q <= 1'b0;
                qbar <= 1'b1;
```

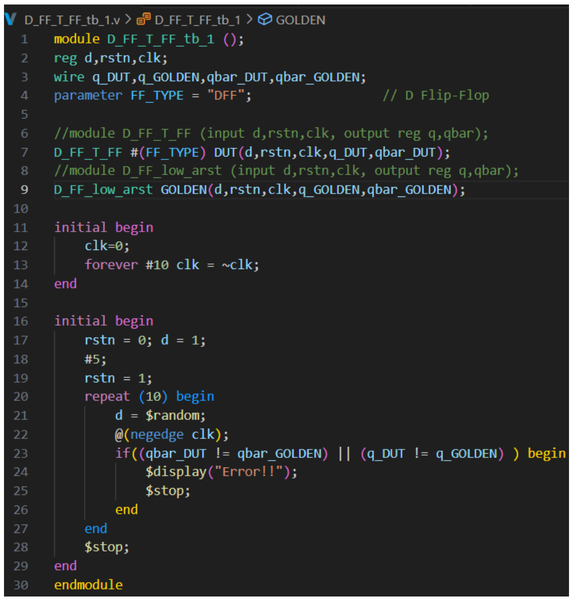
| Instance | Module | Design Unit Type |
|---|---|---|
| D_FF_T_FF | D_FF_T_FF | Top Module |

| Name | Count |
|---|---|
| Resolved(verified, fixed, ... | 1 |
| Info | 1 |

Lint Checks — Filter: Type here — Total : 0 Selected : 0

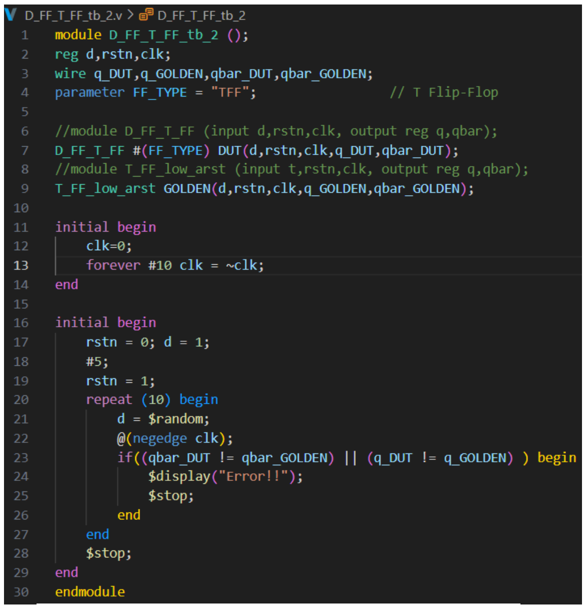
| Severity | Status | Check | Alias | Message | Module | Category | State | Owner | STARC Reference |
|---|---|---|---|---|---|---|---|---|---|

*Figure 9: Q2 Linting*



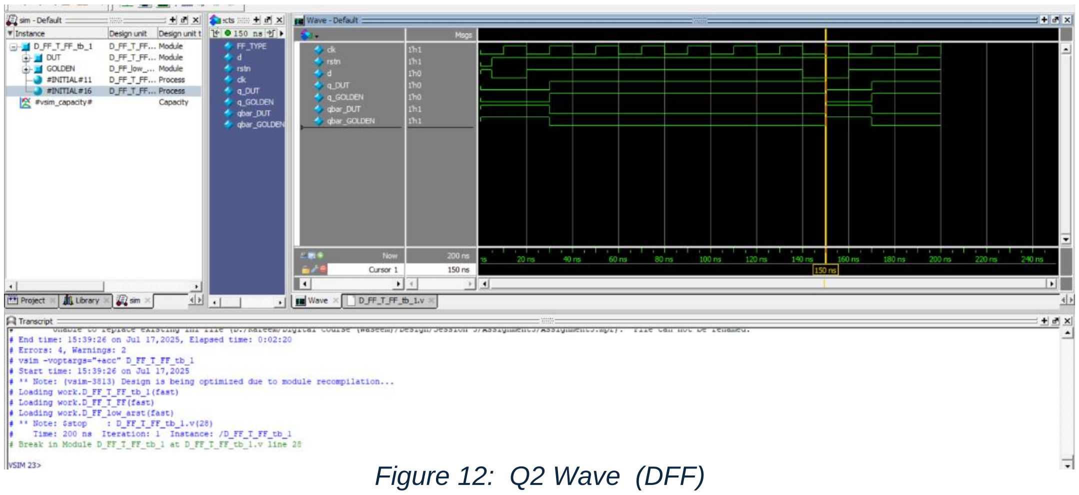
```verilog
module D_FF_T_FF_tb_1 ();
reg d,rstn,clk;
wire q_DUT,q_GOLDEN,qbar_DUT,qbar_GOLDEN;
parameter FF_TYPE = "DFF";                  // D Flip-Flop

//module D_FF_T_FF (input d,rstn,clk, output reg q,qbar);
D_FF_T_FF #(FF_TYPE) DUT(d,rstn,clk,q_DUT,qbar_DUT);
//module D_FF_low_arst (input d,rstn,clk, output reg q,qbar);
D_FF_low_arst GOLDEN(d,rstn,clk,q_GOLDEN,qbar_GOLDEN);

initial begin
    clk=0;
    forever #10 clk = ~clk;
end

initial begin
    rstn = 0; d = 1;
    #5;
    rstn = 1;
    repeat (10) begin
        d = $random;
        @(negedge clk);
        if((qbar_DUT != qbar_GOLDEN) || (q_DUT != q_GOLDEN) ) begin
            $display("Error!!");
            $stop;
        end
    end
    $stop;
end
endmodule
```

Figure 10: Q2 Testbench (DFF)



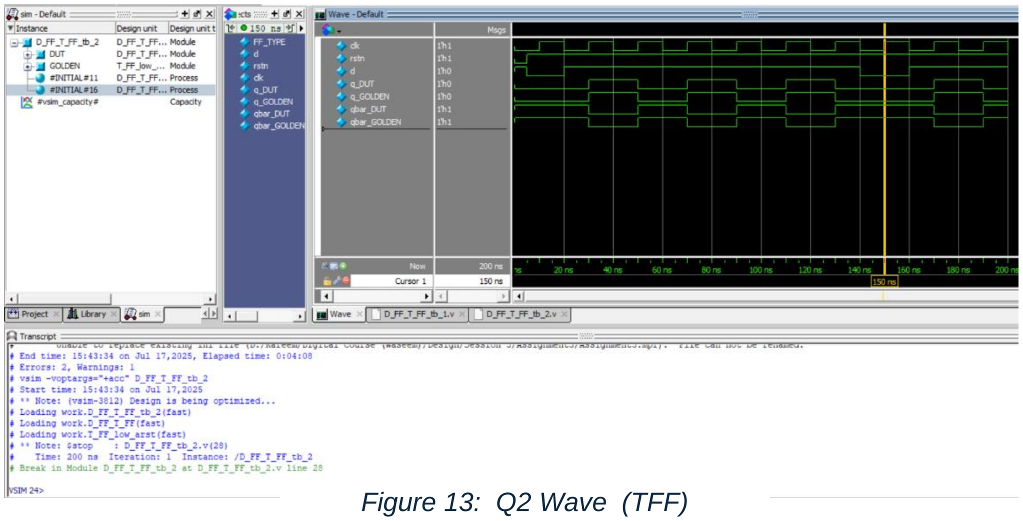
```verilog
module D_FF_T_FF_tb_2 ();
reg d,rstn,clk;
wire q_DUT,q_GOLDEN,qbar_DUT,qbar_GOLDEN;
parameter FF_TYPE = "TFF";                  // T Flip-Flop

//module D_FF_T_FF (input d,rstn,clk, output reg q,qbar);
D_FF_T_FF #(FF_TYPE) DUT(d,rstn,clk,q_DUT,qbar_DUT);
//module T_FF_low_arst (input t,rstn,clk, output reg q,qbar);
T_FF_low_arst GOLDEN(d,rstn,clk,q_GOLDEN,qbar_GOLDEN);

initial begin
    clk=0;
    forever #10 clk = ~clk;
end

initial begin
    rstn = 0; d = 1;
    #5;
    rstn = 1;
    repeat (10) begin
        d = $random;
        @(negedge clk);
        if((qbar_DUT != qbar_GOLDEN) || (q_DUT != q_GOLDEN) ) begin
            $display("Error!!");
            $stop;
        end
    end
    $stop;
end
endmodule
```

Figure 11: Q2 Testbench (TFF)



```
# End time: 15:39:26 on Jul 17,2025, Elapsed time: 0:02:20
# Errors: 4, Warnings: 2
# vsim -voptargs="+acc" D_FF_T_FF_tb_1
# Start time: 15:39:26 on Jul 17,2025
# ** Note: (vsim-3813) Design is being optimized due to module recompilation...
# Loading work.D_FF_T_FF_tb_1(fast)
# Loading work.D_FF_T_FF(fast)
# Loading work.D_FF_low_arst(fast)
# ** Note: $stop    : D_FF_T_FF_tb_1.v(28)
#    Time: 200 ns  Iteration: 1  Instance: /D_FF_T_FF_tb_1
# Break in Module D_FF_T_FF_tb_1 at D_FF_T_FF_tb_1.v line 28

VSIM 23>
```

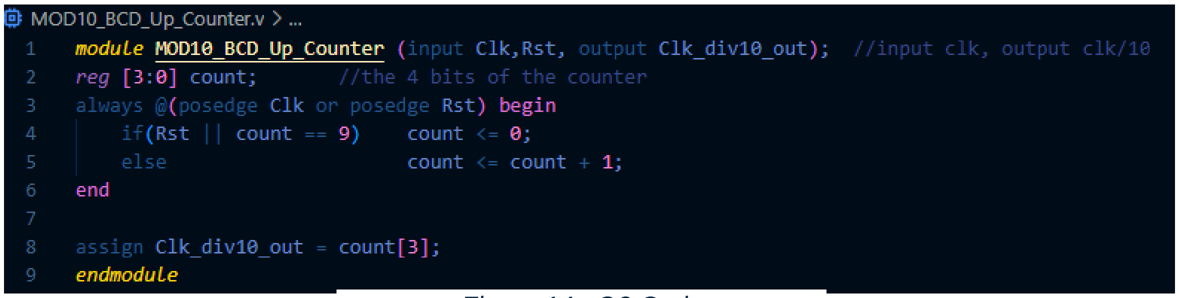
Figure 12: Q2 Wave (DFF)



```
# End time: 15:43:34 on Jul 17,2025, Elapsed time: 0:04:08
# Errors: 2, Warnings: 1
# vsim -voptargs="+acc" D_FF_T_FF_tb_2
# Start time: 15:43:34 on Jul 17,2025
# ** Note: (vsim-3812) Design is being optimized...
# Loading work.D_FF_T_FF_tb_2(fast)
# Loading work.D_FF_T_FF(fast)
# Loading work.T_FF_low_arst(fast)
# ** Note: $stop    : D_FF_T_FF_tb_2.v(28)
#    Time: 200 ns  Iteration: 1  Instance: /D_FF_T_FF_tb_2
# Break in Module D_FF_T_FF_tb_2 at D_FF_T_FF_tb_2.v line 28

VSIM 24>
```

Figure 13: Q2 Wave (TFF)

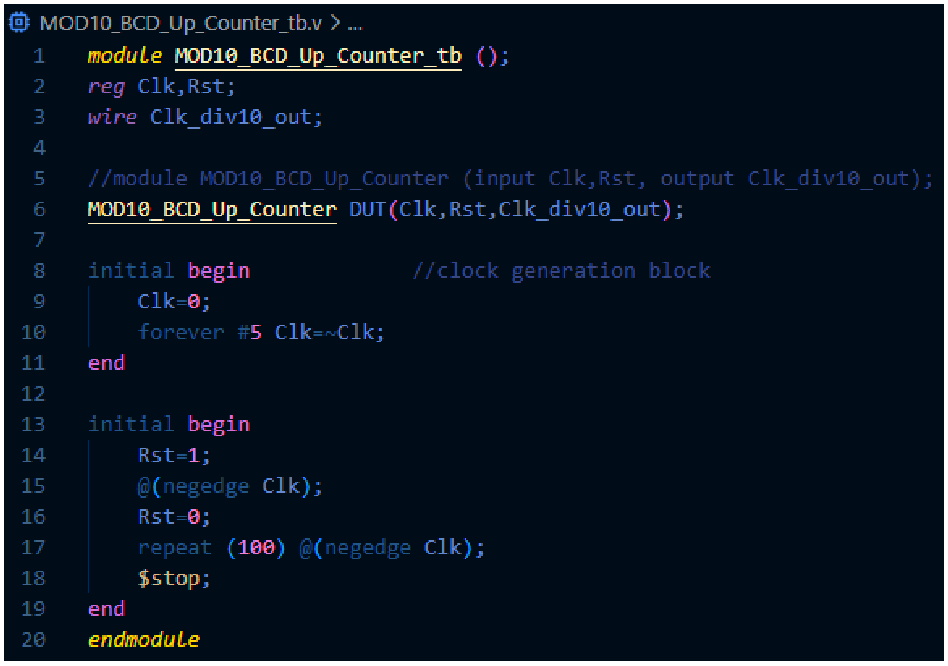
## 3) BCD up counter (MOD 10 counter):



MOD10_BCD_Up_Counter.v > ...

```verilog
module MOD10_BCD_Up_Counter (input Clk,Rst, output Clk_div10_out);  //input clk, output clk/10
reg [3:0] count;        //the 4 bits of the counter
always @(posedge Clk or posedge Rst) begin
    if(Rst || count == 9)     count <= 0;
    else                      count <= count + 1;
end

assign Clk_div10_out = count[3];
endmodule
```

Figure 14:  Q3 Code



MOD10_BCD_Up_Counter_tb.v > ...

```verilog
module MOD10_BCD_Up_Counter_tb ();
reg Clk,Rst;
wire Clk_div10_out;

//module MOD10_BCD_Up_Counter (input Clk,Rst, output Clk_div10_out);
MOD10_BCD_Up_Counter DUT(Clk,Rst,Clk_div10_out);

initial begin               //clock generation block
    Clk=0;
    forever #5 Clk=~Clk;
end

initial begin
    Rst=1;
    @(negedge Clk);
    Rst=0;
    repeat (100) @(negedge Clk);
    $stop;
end
endmodule
```

Figure 15:  Q3 Testbench

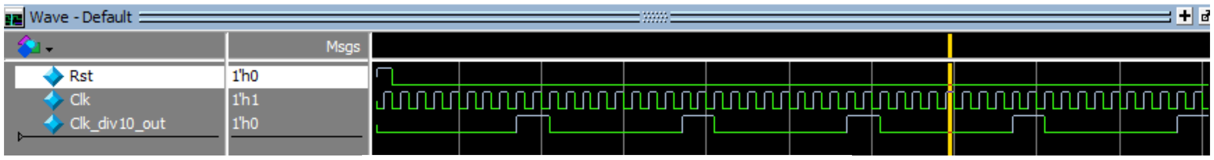
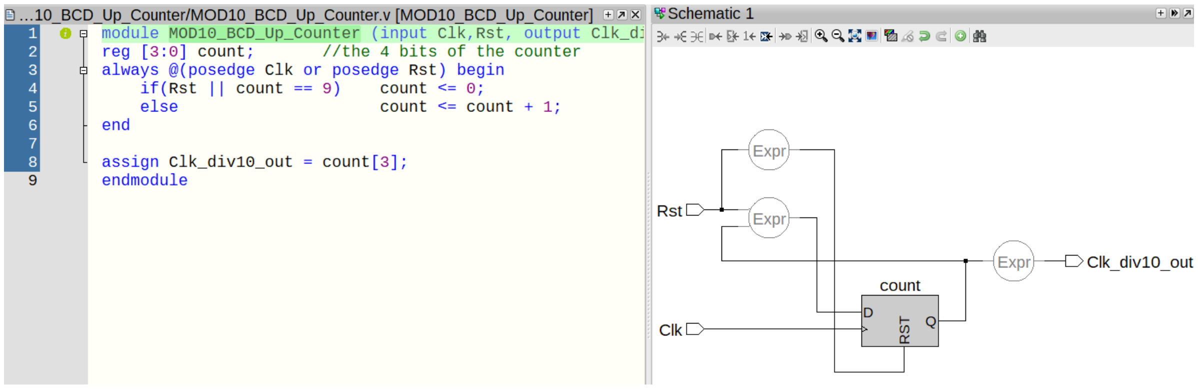


Figure 16:  Q3 Wave

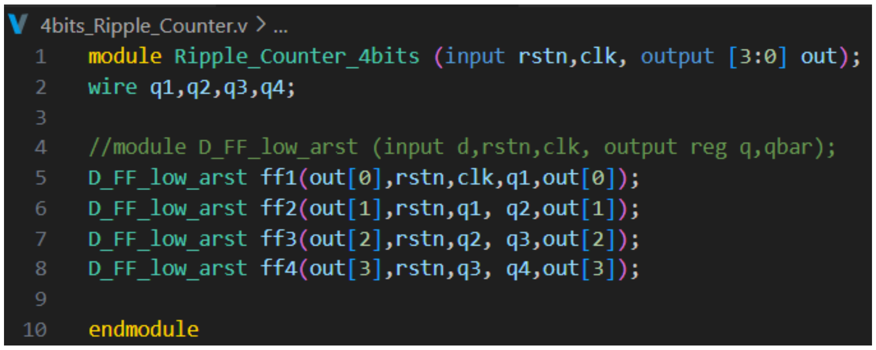


...10_BCD_Up_Counter/MOD10_BCD_Up_Counter.v [MOD10_BCD_Up_Counter]

```verilog
module MOD10_BCD_Up_Counter (input Clk,Rst, output Clk_di
reg [3:0] count;        //the 4 bits of the counter
always @(posedge Clk or posedge Rst) begin
    if(Rst || count == 9)     count <= 0;
    else                      count <= count + 1;
end

assign Clk_div10_out = count[3];
endmodule
```

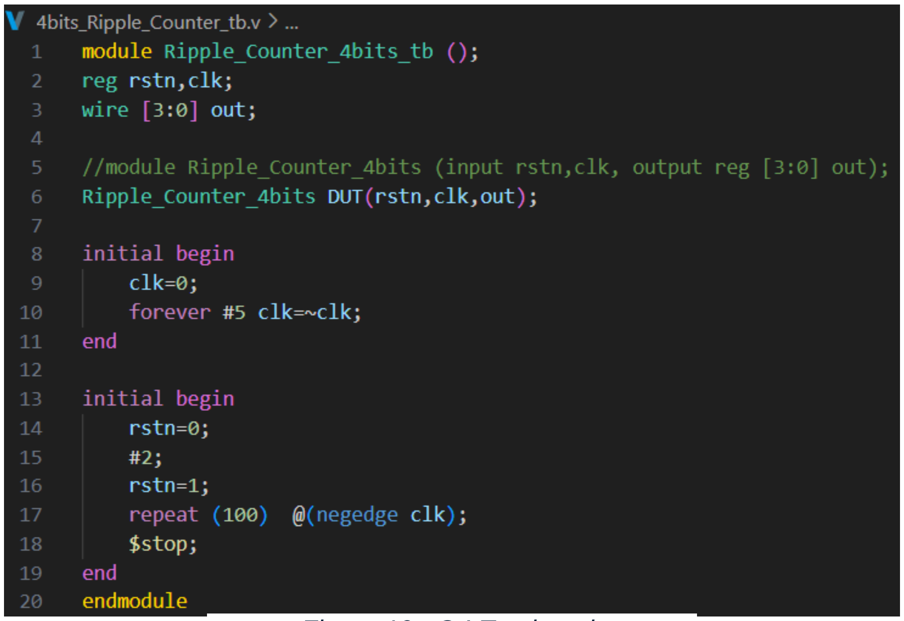
Figure 17:  Q3 Linting

## 4) 4-bit Ripple Counter:



```
4bits_Ripple_Counter.v > ...
module Ripple_Counter_4bits (input rstn,clk, output [3:0] out);
wire q1,q2,q3,q4;

//module D_FF_low_arst (input d,rstn,clk, output reg q,qbar);
D_FF_low_arst ff1(out[0],rstn,clk,q1,out[0]);
D_FF_low_arst ff2(out[1],rstn,q1, q2,out[1]);
D_FF_low_arst ff3(out[2],rstn,q2, q3,out[2]);
D_FF_low_arst ff4(out[3],rstn,q3, q4,out[3]);

endmodule
```

*Figure 18: Q4 Code*



```
4bits_Ripple_Counter_tb.v > ...
module Ripple_Counter_4bits_tb ();
reg rstn,clk;
wire [3:0] out;

//module Ripple_Counter_4bits (input rstn,clk, output reg [3:0] out);
Ripple_Counter_4bits DUT(rstn,clk,out);

initial begin
    clk=0;
    forever #5 clk=~clk;
end

initial begin
    rstn=0;
    #2;
    rstn=1;
    repeat (100)  @(negedge clk);
    $stop;
end
endmodule
```

*Figure 19: Q4 Testbench*

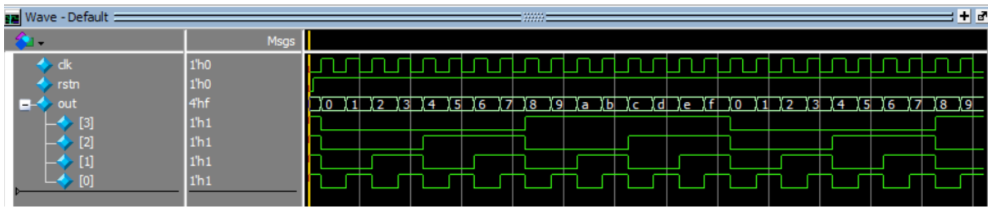


*Figure 20: Q4 Wave*

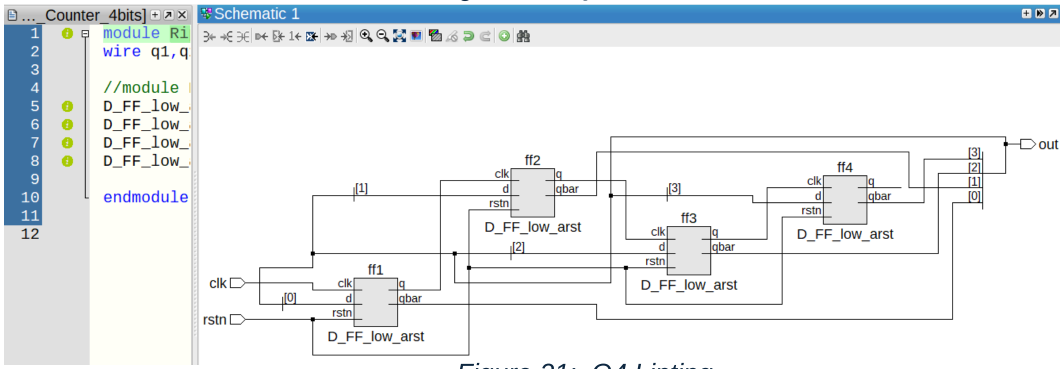
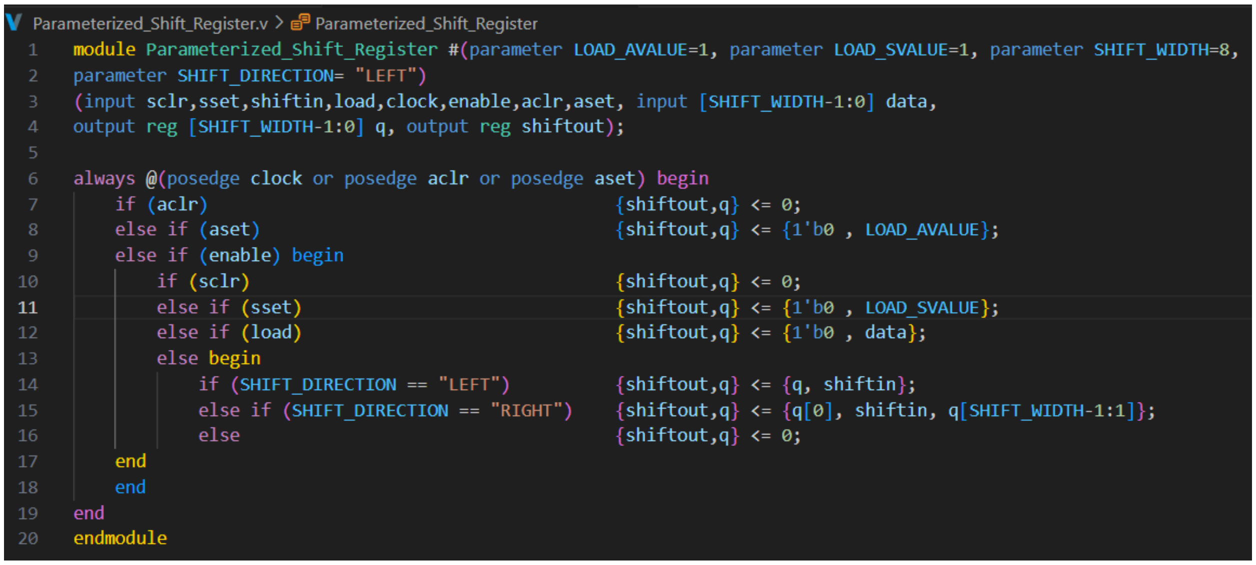


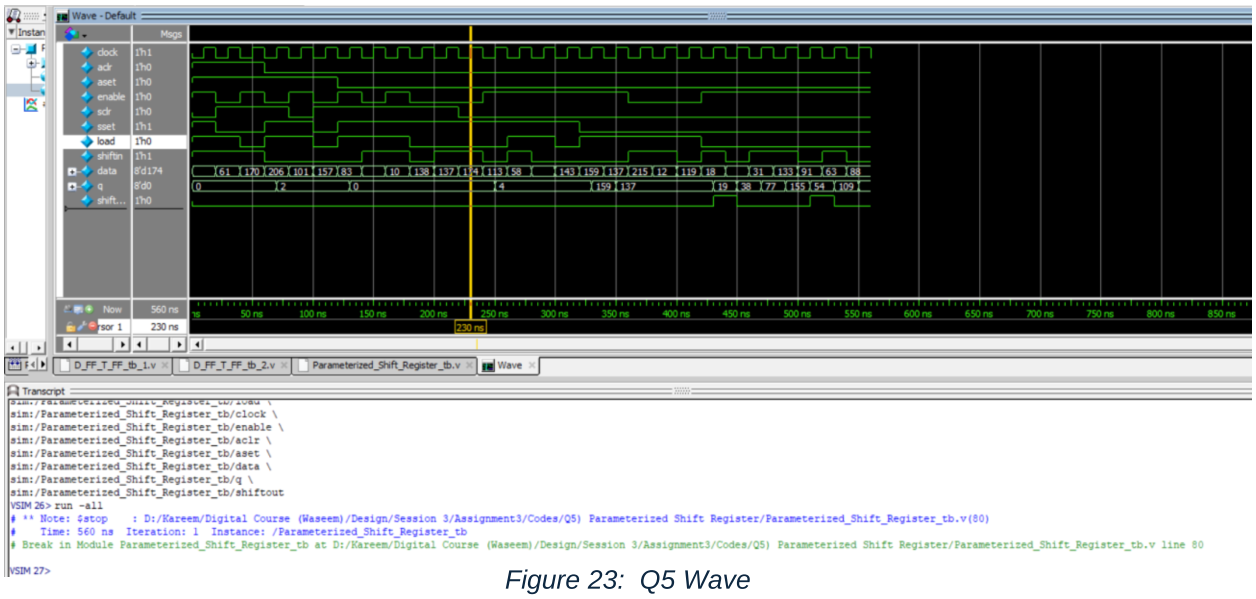
*Figure 21: Q4 Linting*

## 5) Parameterized Shift register:



```verilog
module Parameterized_Shift_Register #(parameter LOAD_AVALUE=1, parameter LOAD_SVALUE=1, parameter SHIFT_WIDTH=8,
parameter SHIFT_DIRECTION= "LEFT")
(input sclr,sset,shiftin,load,clock,enable,aclr,aset, input [SHIFT_WIDTH-1:0] data,
output reg [SHIFT_WIDTH-1:0] q, output reg shiftout);

always @(posedge clock or posedge aclr or posedge aset) begin
    if (aclr)                                     {shiftout,q} <= 0;
    else if (aset)                                {shiftout,q} <= {1'b0 , LOAD_AVALUE};
    else if (enable) begin
        if (sclr)                                 {shiftout,q} <= 0;
        else if (sset)                            {shiftout,q} <= {1'b0 , LOAD_SVALUE};
        else if (load)                            {shiftout,q} <= {1'b0 , data};
        else begin
            if (SHIFT_DIRECTION == "LEFT")        {shiftout,q} <= {q, shiftin};
            else if (SHIFT_DIRECTION == "RIGHT")  {shiftout,q} <= {q[0], shiftin, q[SHIFT_WIDTH-1:1]};
            else                                  {shiftout,q} <= 0;
    end
    end
end
endmodule
```

Figure 22: Q5 Code



```
sim:/Parameterized_Shift_Register_tb/clock \
sim:/Parameterized_Shift_Register_tb/enable \
sim:/Parameterized_Shift_Register_tb/aclr \
sim:/Parameterized_Shift_Register_tb/aset \
sim:/Parameterized_Shift_Register_tb/data \
sim:/Parameterized_Shift_Register_tb/q \
sim:/Parameterized_Shift_Register_tb/shiftout
VSIM 26> run -all
# ** Note: $stop    : D:/Kareem/Digital Course (Waseem)/Design/Session 3/Assignment3/Codes/Q5) Parameterized Shift Register/Parameterized_Shift_Register_tb.v(80)
#    Time: 560 ns  Iteration: 1  Instance: /Parameterized_Shift_Register_tb
# Break in Module Parameterized_Shift_Register_tb at D:/Kareem/Digital Course (Waseem)/Design/Session 3/Assignment3/Codes/Q5) Parameterized Shift Register/Parameterized_Shift_Register_tb.v line 80

VSIM 27>
```

Figure 23: Q5 Wave

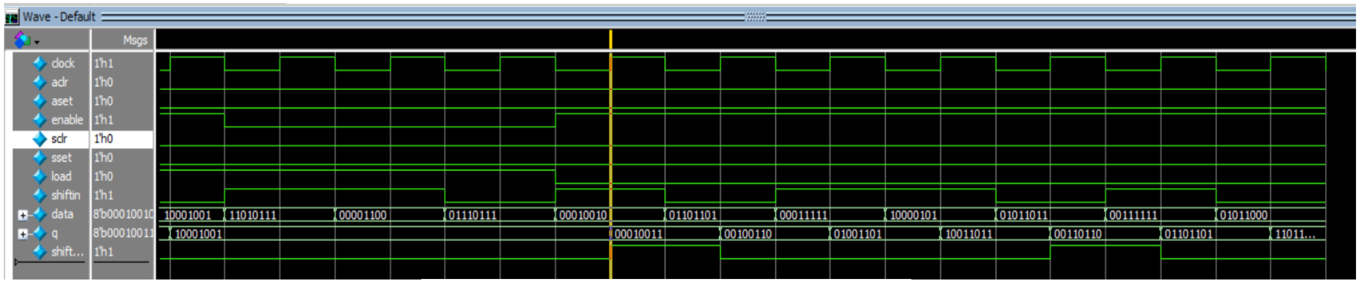
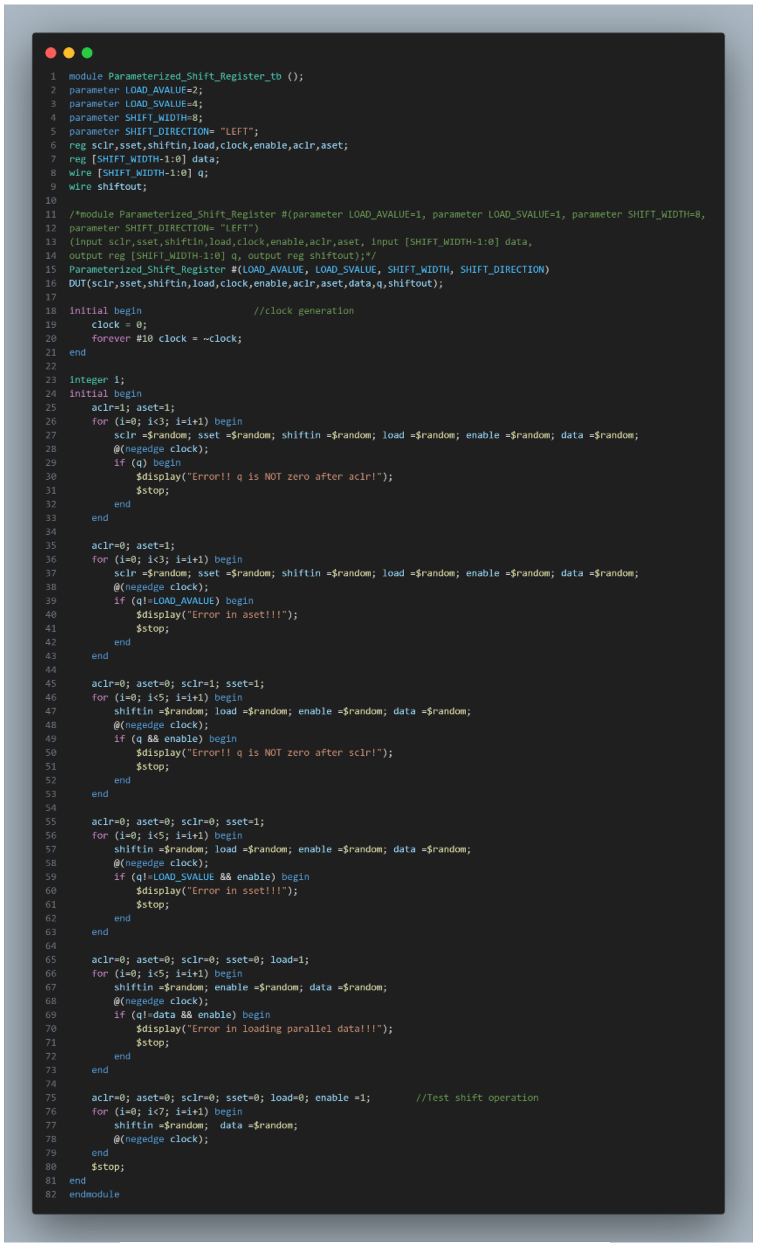
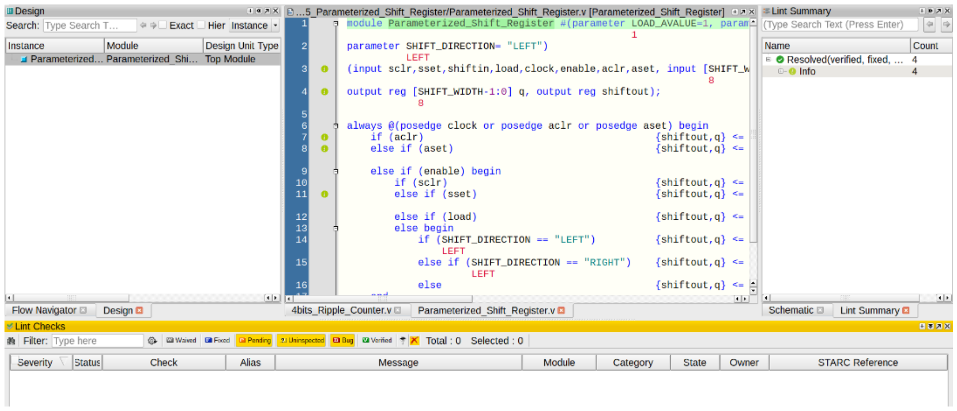


Figure 24: Q5 Wave (Shifting Part)



```verilog
module Parameterized_Shift_Register_tb ();
parameter LOAD_AVALUE=2;
parameter LOAD_SVALUE=4;
parameter SHIFT_WIDTH=8;
parameter SHIFT_DIRECTION= "LEFT";
reg sclr,sset,shiftin,load,clock,enable,aclr,aset;
reg [SHIFT_WIDTH-1:0] data;
wire [SHIFT_WIDTH-1:0] q;
wire shiftout;

/*module Parameterized_Shift_Register #(parameter LOAD_AVALUE=1, parameter LOAD_SVALUE=1, parameter SHIFT_WIDTH=8,
parameter SHIFT_DIRECTION= "LEFT")
(input sclr,sset,shiftin,load,clock,enable,aclr,aset, input [SHIFT_WIDTH-1:0] data,
output reg [SHIFT_WIDTH-1:0] q, output reg shiftout);*/
Parameterized_Shift_Register #(LOAD_AVALUE, LOAD_SVALUE, SHIFT_WIDTH, SHIFT_DIRECTION)
DUT(sclr,sset,shiftin,load,clock,enable,aclr,aset,data,q,shiftout);

initial begin                    //clock generation
    clock = 0;
    forever #10 clock = ~clock;
end

integer i;
initial begin
    aclr=1; aset=1;
    for (i=0; i<3; i=i+1) begin
        sclr =$random; sset =$random; shiftin =$random; load =$random; enable =$random; data =$random;
        @(negedge clock);
        if (q) begin
            $display("Error!! q is NOT zero after aclr!");
            $stop;
        end
    end

    aclr=0; aset=1;
    for (i=0; i<3; i=i+1) begin
        sclr =$random; sset =$random; shiftin =$random; load =$random; enable =$random; data =$random;
        @(negedge clock);
        if (q!=LOAD_AVALUE) begin
            $display("Error in aset!!!");
            $stop;
        end
    end

    aclr=0; aset=0; sclr=1; sset=1;
    for (i=0; i<5; i=i+1) begin
        shiftin =$random; load =$random; enable =$random; data =$random;
        @(negedge clock);
        if (q && enable) begin
            $display("Error!! q is NOT zero after sclr!");
            $stop;
        end
    end

    aclr=0; aset=0; sclr=0; sset=1;
    for (i=0; i<5; i=i+1) begin
        shiftin =$random; load =$random; enable =$random; data =$random;
        @(negedge clock);
        if (q!=LOAD_SVALUE && enable) begin
            $display("Error in sset!!!");
            $stop;
        end
    end

    aclr=0; aset=0; sclr=0; sset=0; load=1;
    for (i=0; i<5; i=i+1) begin
        shiftin =$random; enable =$random; data =$random;
        @(negedge clock);
        if (q!=data && enable) begin
            $display("Error in loading parallel data!!!");
            $stop;
        end
    end

    aclr=0; aset=0; sclr=0; sset=0; load=0; enable =1;        //Test shift operation
    for (i=0; i<7; i=i+1) begin
        shiftin =$random;  data =$random;
        @(negedge clock);
    end
    $stop;
end
endmodule
```

Figure 25: Q5 Testbench



| Instance | Module | Design Unit Type |
|---|---|---|
| Parameterized… | Parameterized_Shi… | Top Module |

…5_Parameterized_Shift_Register/Parameterized_Shift_Register.v [Parameterized_Shift_Register]

```verilog
module Parameterized_Shift_Register #(parameter LOAD_AVALUE=1, param
                                                              1
parameter SHIFT_DIRECTION= "LEFT")
          LEFT
(input sclr,sset,shiftin,load,clock,enable,aclr,aset, input [SHIFT_W
                                                             8
output reg [SHIFT_WIDTH-1:0] q, output reg shiftout);
            8

always @(posedge clock or posedge aclr or posedge aset) begin
    if (aclr)                                          {shiftout,q} <=
    else if (aset)                                     {shiftout,q} <=

    else if (enable) begin
        if (sclr)                                      {shiftout,q} <=
        else if (sset)                                 {shiftout,q} <=

        else if (load)                                 {shiftout,q} <=
        else begin
            if (SHIFT_DIRECTION == "LEFT")             {shiftout,q} <=
                LEFT
            else if (SHIFT_DIRECTION == "RIGHT")       {shiftout,q} <=
                    LEFT
            else                                       {shiftout,q} <=
```

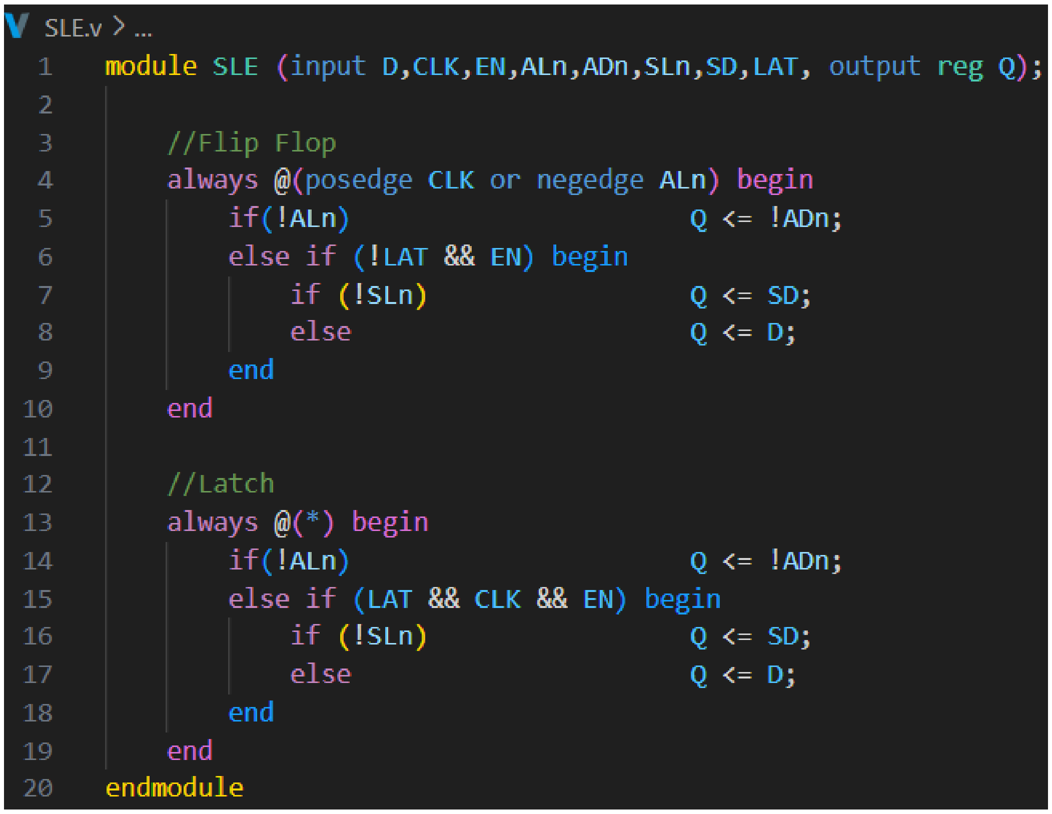
Lint Summary

| Name | Count |
|---|---|
| Resolved(verified, fixed, … | 4 |
| Info | 4 |

| Severity | Status | Check | Alias | Message | Module | Category | State | Owner | STARC Reference |
|---|---|---|---|---|---|---|---|---|---|

Figure 26: Q5 Linting

## 6) SLE (Sequential Logic Element):



SLE.v > ...

```verilog
module SLE (input D,CLK,EN,ALn,ADn,SLn,SD,LAT, output reg Q);

    //Flip Flop
    always @(posedge CLK or negedge ALn) begin
        if(!ALn)                        Q <= !ADn;
        else if (!LAT && EN) begin
            if (!SLn)                   Q <= SD;
            else                        Q <= D;
        end
    end

    //Latch
    always @(*) begin
        if(!ALn)                        Q <= !ADn;
        else if (LAT && CLK && EN) begin
            if (!SLn)                   Q <= SD;
            else                        Q <= D;
        end
    end
endmodule
```

Figure 27: Q6 Code

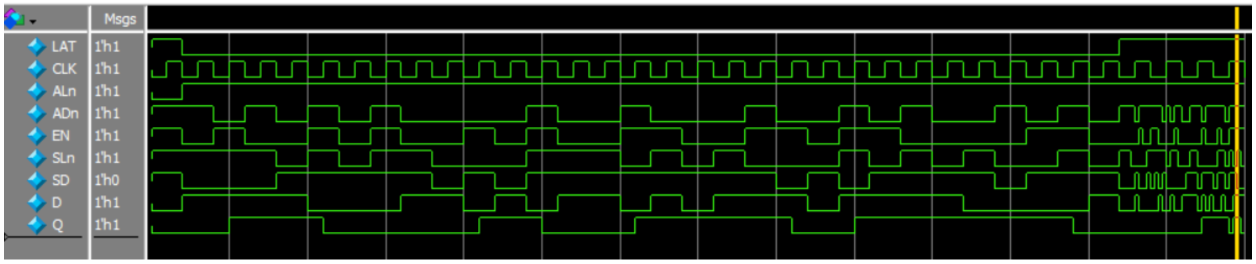
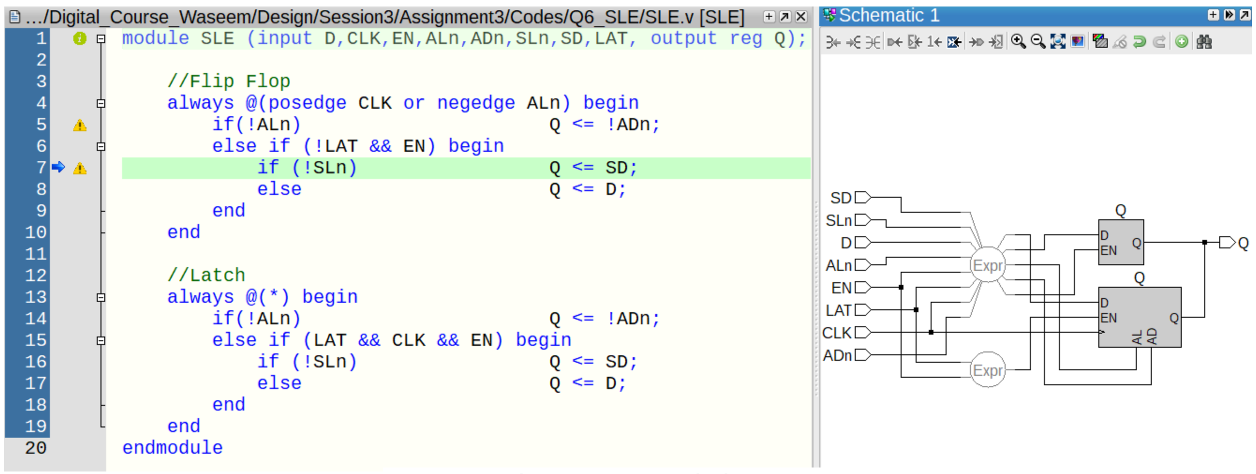


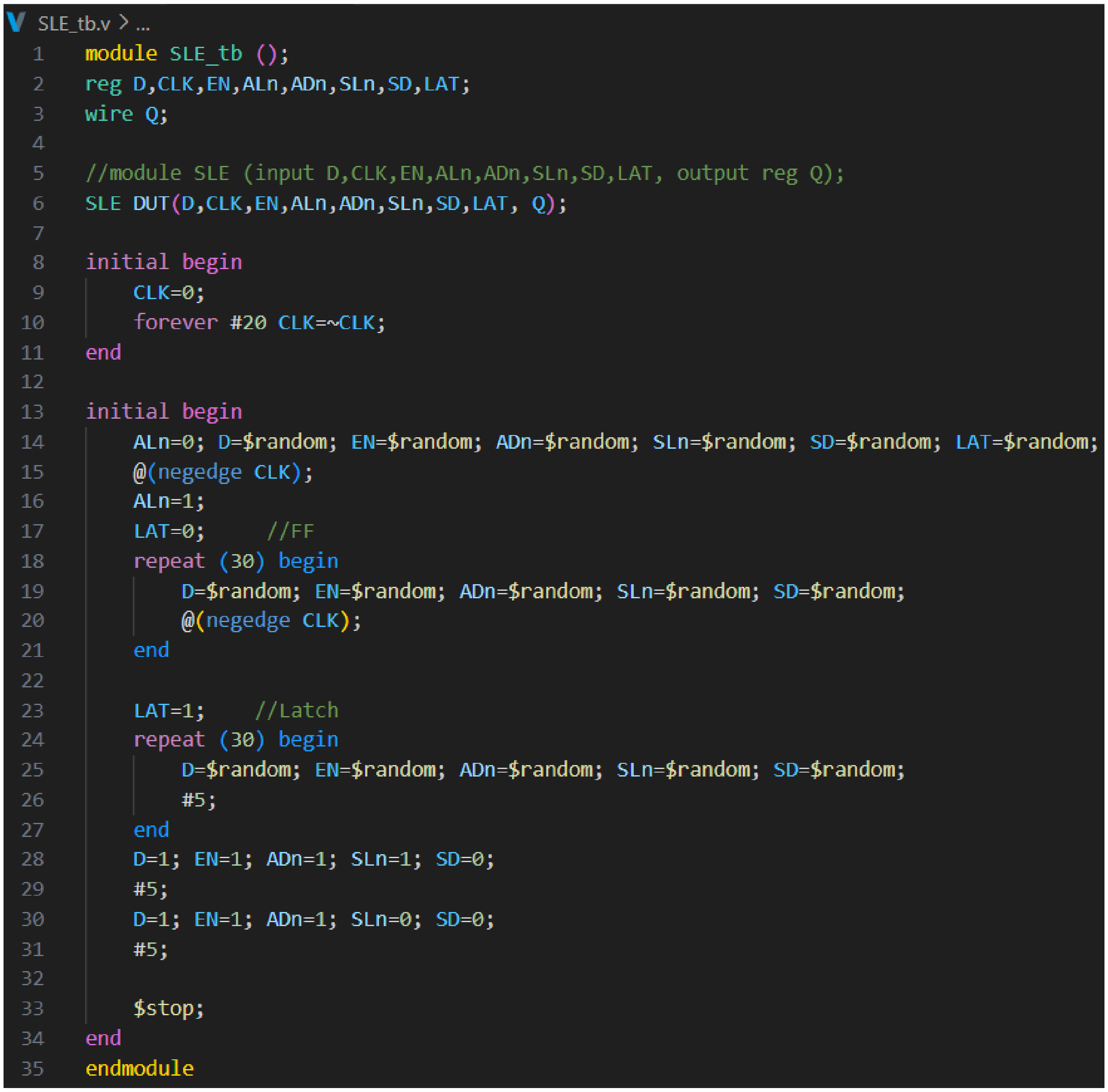
Figure 28: Q6 Wave



.../Digital_Course_Waseem/Design/Session3/Assignment3/Codes/Q6_SLE/SLE.v [SLE]

```verilog
module SLE (input D,CLK,EN,ALn,ADn,SLn,SD,LAT, output reg Q);

    //Flip Flop
    always @(posedge CLK or negedge ALn) begin
        if(!ALn)                        Q <= !ADn;
        else if (!LAT && EN) begin
            if (!SLn)                   Q <= SD;
            else                        Q <= D;
        end
    end

    //Latch
    always @(*) begin
        if(!ALn)                        Q <= !ADn;
        else if (LAT && CLK && EN) begin
            if (!SLn)                   Q <= SD;
            else                        Q <= D;
        end
    end
endmodule
```

Figure 29: Q6 Linting


SLE_tb.v > ...

```verilog
module SLE_tb ();
reg D,CLK,EN,ALn,ADn,SLn,SD,LAT;
wire Q;

//module SLE (input D,CLK,EN,ALn,ADn,SLn,SD,LAT, output reg Q);
SLE DUT(D,CLK,EN,ALn,ADn,SLn,SD,LAT, Q);

initial begin
    CLK=0;
    forever #20 CLK=~CLK;
end

initial begin
    ALn=0; D=$random; EN=$random; ADn=$random; SLn=$random; SD=$random; LAT=$random;
    @(negedge CLK);
    ALn=1;
    LAT=0;     //FF
    repeat (30) begin
        D=$random; EN=$random; ADn=$random; SLn=$random; SD=$random;
        @(negedge CLK);
    end

    LAT=1;    //Latch
    repeat (30) begin
        D=$random; EN=$random; ADn=$random; SLn=$random; SD=$random;
        #5;
    end
    D=1; EN=1; ADn=1; SLn=1; SD=0;
    #5;
    D=1; EN=1; ADn=1; SLn=0; SD=0;
    #5;

    $stop;
end
endmodule
```

*Figure 30: Q6 Testbench*