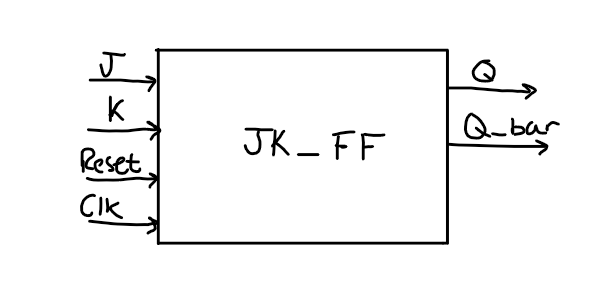
# Laboratory Report #6

**Name:** Josh Ratificar **Date Completed:** 12-08-2023

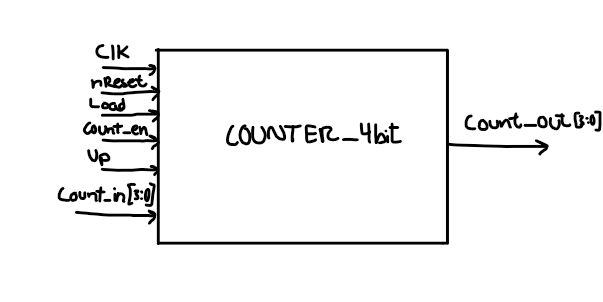
**Laboratory Exercise Title:** Behavioral Modeling of Sequential Circuits

**Block Diagrams:**

**Figure 1.0** – *Hex\_to\_7\_Seg\_Decoder Block Diagram*



**Figure 2.0 –***Counter\_4bit Block Diagram*

****

**Exercise 5A:**

**Figure 1.1** –*JK\_FF.v Script*

/\**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*   FILE:               JK\_FF.v*

*\*   AUTHOR:             Mohan Francis, Josh Ratificar*

*\*   Class:              Gr.3 CpE 3101L Introduction to HDL*

*\*   Group/Schedule      Friday, 7:30 AM to 10:30 AM*

*\*   Description:        JK\_FF.v file*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**\*/

module JK\_FF (

    input wire J,K,Reset, Clk,

    output reg Q, Q\_bar

);

always @(negedge Clk, posedge Reset)

    begin

        if(Reset)

        begin

            Q <= 1'b0;

            Q\_bar <= 1'b1;

        end

        case({J,K})

            2'b00:                          //*NO CHANGE*

                begin

                    Q <= Q;

                    Q\_bar <= Q\_bar;

                end

            2'b01:                          //*RESET*

                begin

                    Q <= 1'b0;

                    Q\_bar <= 1'b1;

                end

            2'b10:                     //*SET*

                begin

                    Q <= 1'b1;

                    Q\_bar <= 1'b0;

                end

            2'b11:                          //*TOGGLE*

                begin

                    Q <= ~Q;

                    Q\_bar <= ~Q\_bar;

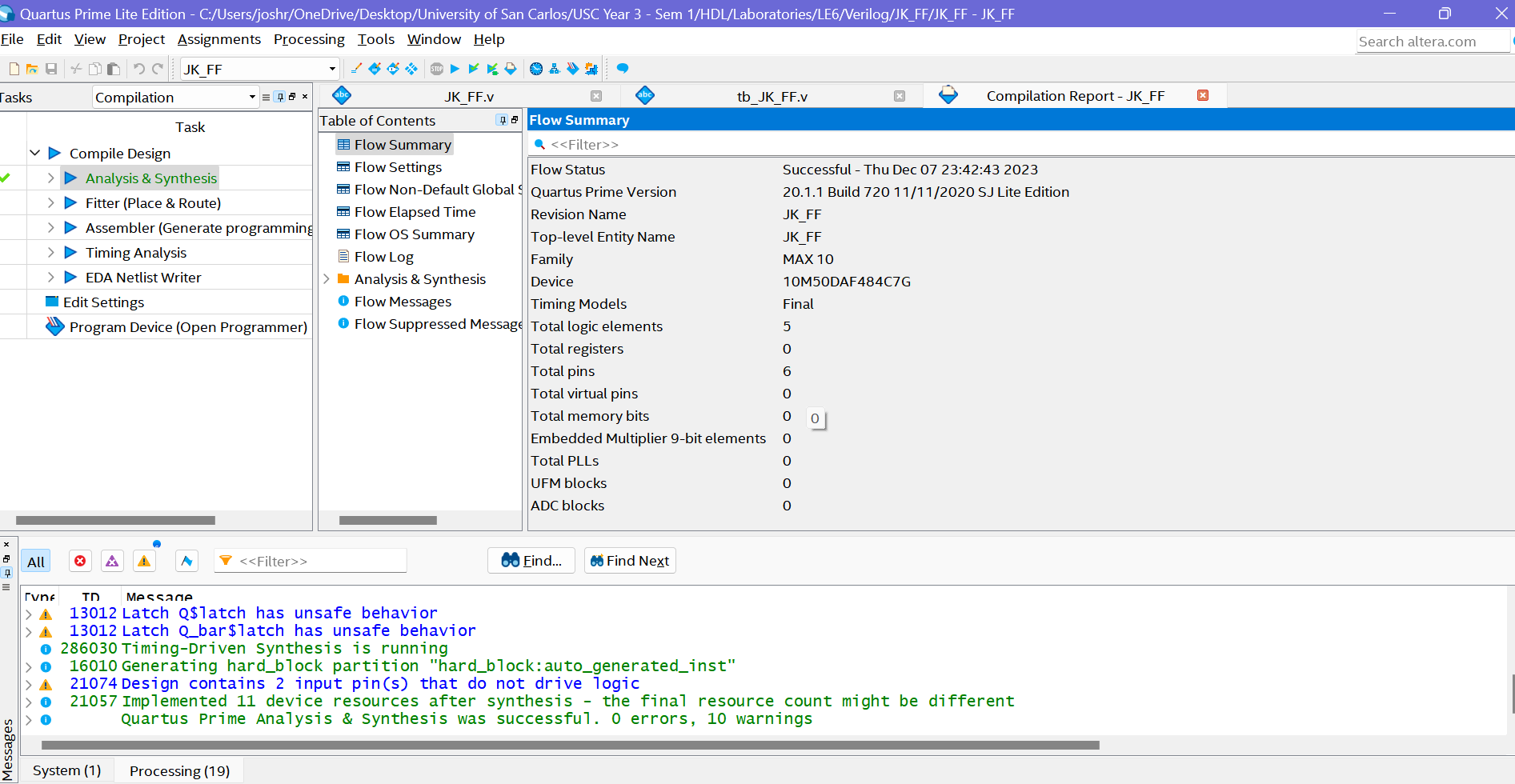
                end

        endcase

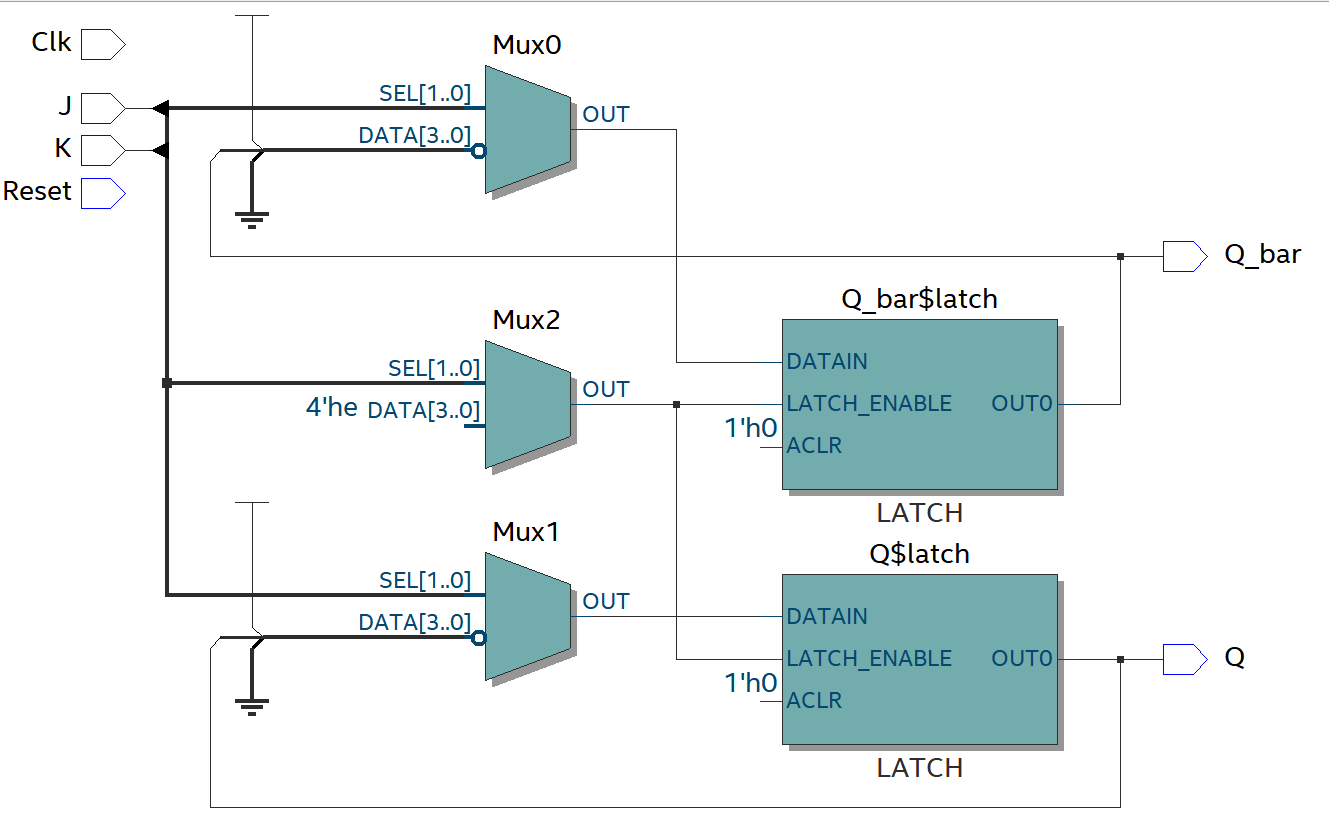
    end

endmodule

**Figure 1.2** – *JK\_FF.v Analysis and Elaboration Test Results*



**Figure 1.3** – *JK\_FF RTL View Output*



**Figure 1.4** – *tb\_* *JK\_FF.v Script for Testing Module*

/\**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*   FILE:                   tb\_JK\_FF.v*

*\*   AUTHOR:             Mohan Francis, Josh Ratificar*

*\*   Class:              Gr.3 CpE 3101L Introduction to HDL*

*\*   Group/Schedule      Friday, 7:30 AM to 10:30 AM*

*\*   Description:        Test Bench for JK\_FF.v file*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**\*/

`timescale 1 ns / 1 ps

module tb\_JK\_FF();

    reg  J,K,Reset,Clk;

    wire Q,Q\_bar;

    JK\_FF UUT(

        .J(J),

        .K(K),

        .Reset(Reset),

        .Clk(Clk),

        .Q(Q),

        .Q\_bar(Q\_bar)

    );

    initial

        Clk = 1'b0;

    always

        #5 Clk = ~Clk;

    initial begin

        Reset = 1'b1;       #10

        Reset = 1'b0;

    end

    initial begin

        $display("Test Bench | JK FLIP FLOP...");

        J = 1'b0;   K = 1'b0; #5

        J = 1'b0;   K = 1'b1; #5

        J = 1'b1;   K = 1'b0; #5

        J = 1'b1;   K = 1'b1; #5

        J = 1'b0;   K = 1'b0; #5

        J = 1'b0;   K = 1'b1; #5

        J = 1'b1;   K = 1'b0; #5

        J = 1'b1;   K = 1'b1; #5

        J = 1'b1;   K = 1'b1; #5

        $stop;

    end

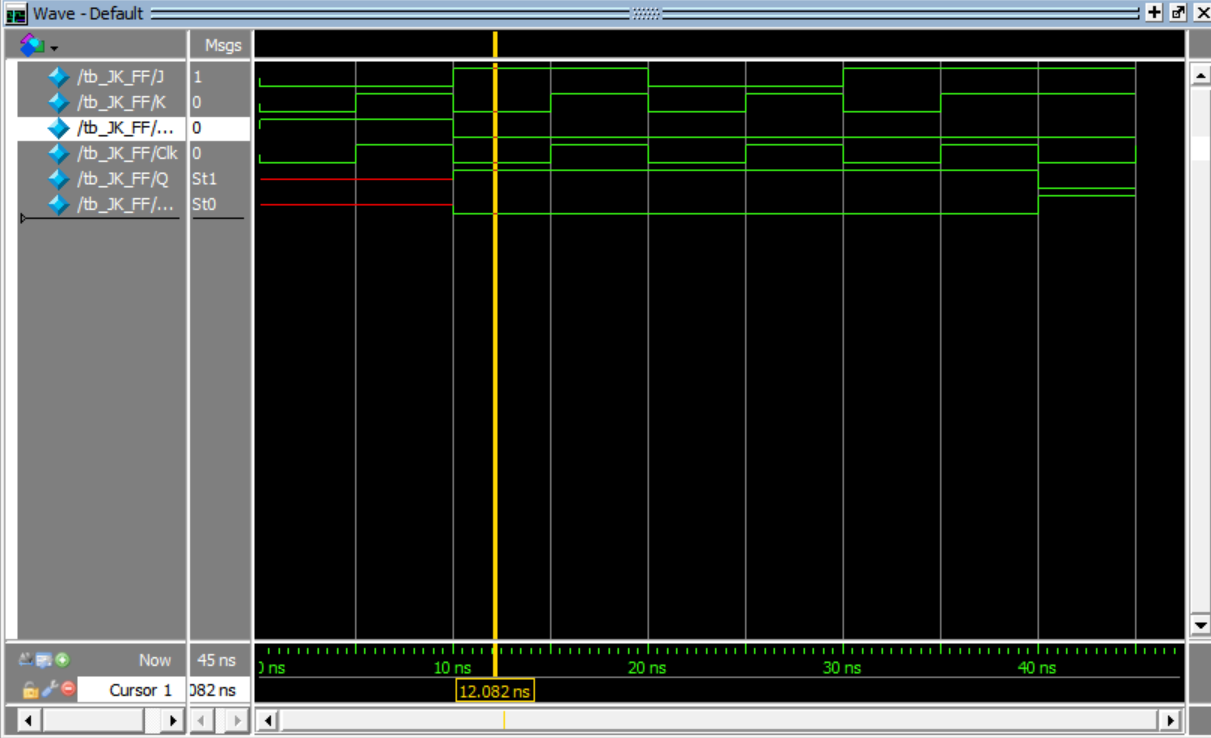
    initial begin

        $monitor("Time = %2d ns | J = %b | K = %b | CLK = %b | RESET = %b | Q = %b | Q\_bar = %b", $time, J, K, Clk, Reset, Q, Q\_bar);

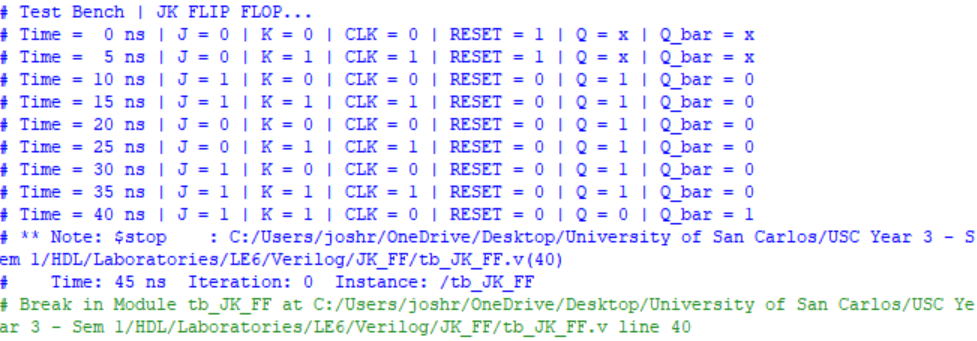
    end

endmodule

**Figure 1.5** – *JK\_FF RTL Simulation Output*

**

**Figure 1.6** – *JK\_FF Test Bench Monitor Output (Annotations to* ***Figure 1.5****)*

****

**Discussion of Results (Exercise 5A)**

It appears that the simulation is working as expected according to the JK Flip Flop’s behaviour as seen in **Figure 1.5**. The behaviour expected is that we can set, reset, no-change, and toggle. This can also be seen through **Figure 1.6** where we can see the asynchronous reset being called behaving as expected.

**Exercise 5B:**

**Figure 2.1** – *Counter\_4bit.v Script*

/\**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*   FILE:                   Counter\_4bit.v*

*\*   AUTHOR:             Mohan Francis, Josh Ratificar*

*\*   Class:              Gr.3 CpE 3101L Introduction to HDL*

*\*   Group/Schedule      Friday, 7:30 AM to 10:30 AM*

*\*   Description:        Counter\_4bit.v file*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**\*/

//*Notes:*

//*nReset   |   Clk   |  Load  |  Count\_en  |  Up  |   Counter Operation*

// *0    |    X    |    X   |      X     |  X   |    Reset to 0*

// *1    |    ®    |    1   |      X     |  X   |    Load inputs (parallel load)*

// *1    |    ®    |    0   |      0     |  X   |    No change*

// *1    |    ®    |    0   |      1     |  0   |    Count down*

// *1    |    ®    |    0   |      1     |  1   |    Count up*

module Counter\_4bit(

    input Clk, nReset, Load, Count\_en, Up,

    input [3:0] Count\_in,

    output reg [3:0] Count\_out

);

    always @(negedge Clk, negedge nReset)

        begin

            if(nReset == 0) // *Checking for Asynchronous Reset*

                Count\_out <= 4'b0000;

            else if(Load == 1) // *Status of Load Pin*

                Count\_out <= Count\_in;

            else if(Count\_en == 1) // *Counter Enable*

                begin

                    if(Up == 1) // *If up is high, then count up*

                        Count\_out <= Count\_out + 1;

                    else // *If up is low, then we should count down*

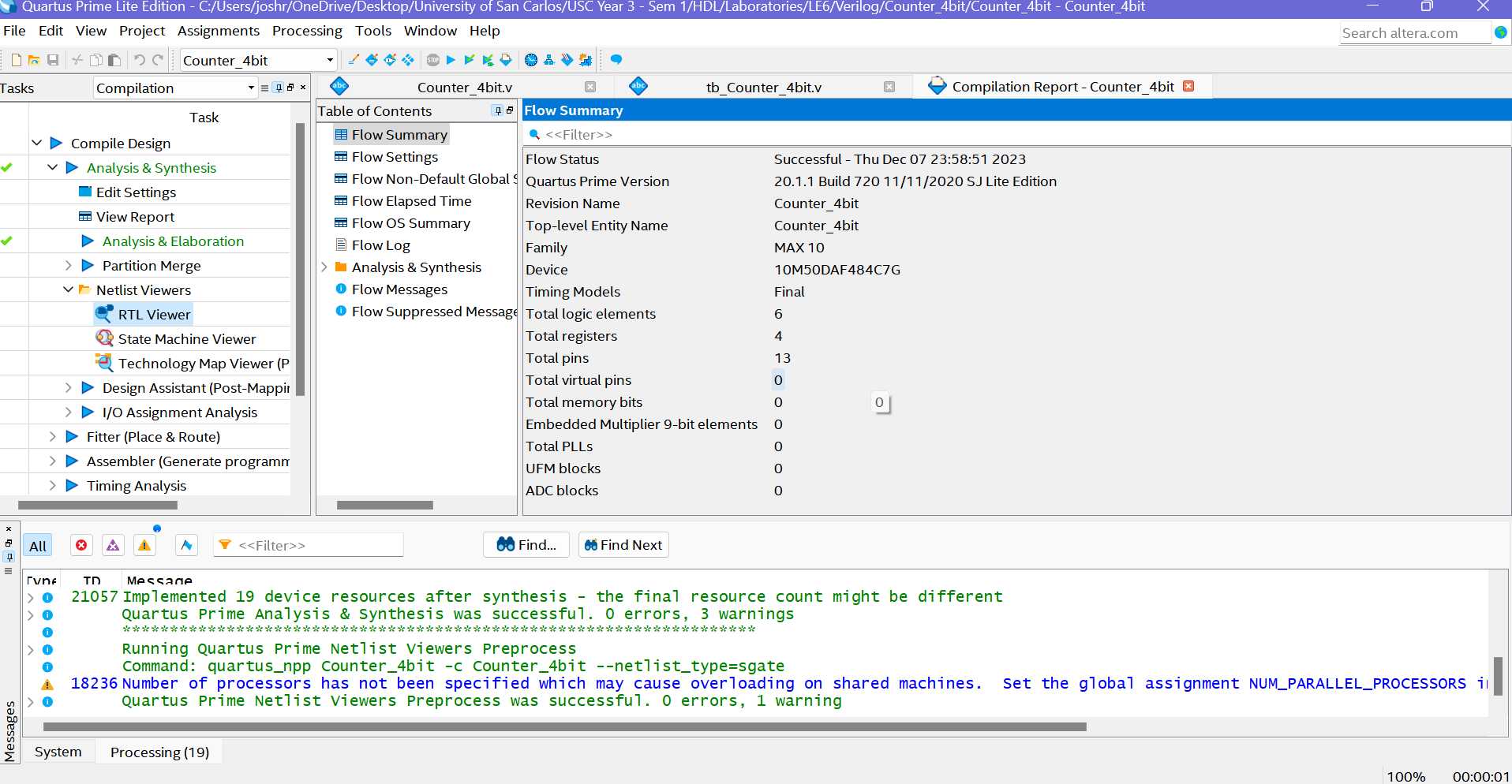
                        Count\_out <= Count\_out - 1;

                end

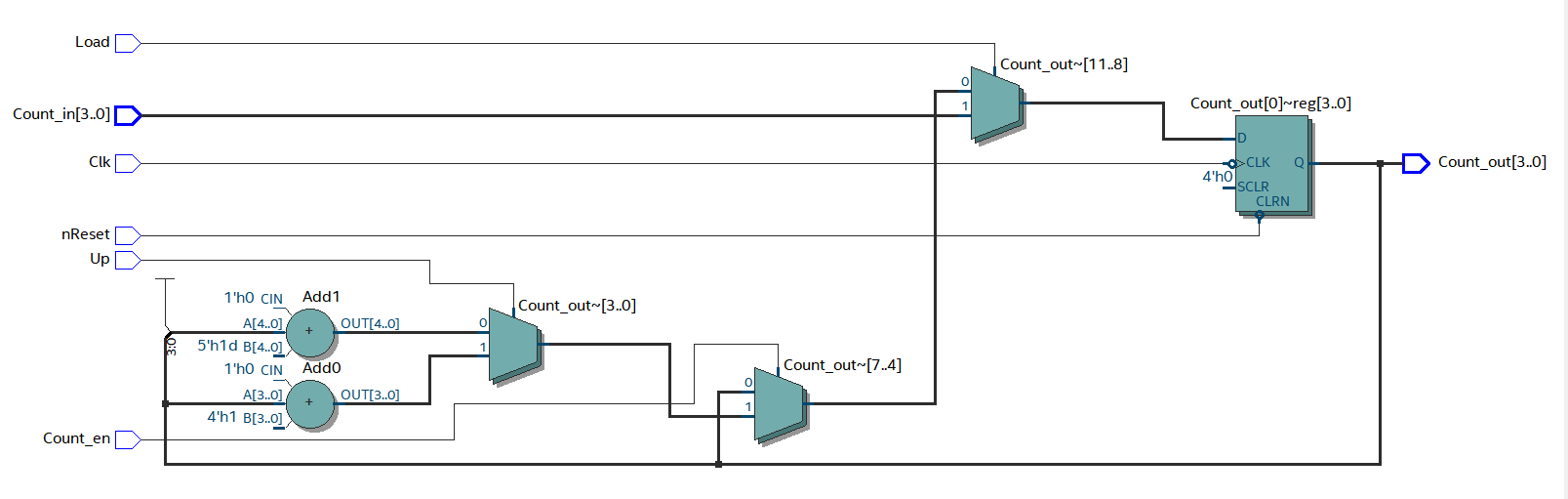
        end

endmodule

**Figure 2.2** – *Counter\_4bit.v Analysis and Elaboration Test Results*

**

**Figure 2.3** – *Counter\_4bit RTL View Output*



**Figure 2.4** – *tb\_ Counter\_4bit.v Script for Testing Module*

/\**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*   FILE:               tb\_Counter\_4bit.v*

*\*   AUTHOR:             Mohan Francis, Josh Ratificar*

*\*   Class:              Gr.3 CpE 3101L Introduction to HDL*

*\*   Group/Schedule      Friday, 7:30 AM to 10:30 AM*

*\*   Description:        Testbench for Counter\_4bit.v file*

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**\*/

`timescale 1 ns / 1 ps

module tb\_Counter\_4bit();

    reg Clk, nReset, Load, Count\_en, Up;

    reg [3:0] Count\_in;

    wire [3:0] Count\_out;

    Counter\_4bit UUT(

        .Clk(Clk),

        .nReset(nReset),

        .Load(Load),

        .Count\_en(Count\_en),

        .Up(Up),

        .Count\_in(Count\_in),

        .Count\_out(Count\_out)

    );

    // *Clock generation*

    always #5 Clk = ~Clk;

    // *Initial block for stimulus generation*

    initial begin

        $display("Test Bench | Counter\_4bit...");

        $display("Initialize inputs...");

        Clk = 0; nReset = 1; Load = 0; Count\_en = 0; Up = 0; Count\_in = 4'b0000; #10

        $display("Loading Counter with 15d");

        Load = 1; #10

        Load = 0; #10

        Count\_in = 4'b1111; #10; // *15*

        $display("Enabling Counting");

        Count\_en = 1; #10

          $display("Countdown:");

        // *Counting down*

        #10 Up = 0; Load = 0; Count\_en = 1;  // *14*

        Count\_en = 1; #10; // *13*

        Count\_en = 1; #10; // *12*

        Count\_en = 1; #10; // *11*

        Count\_en = 1; #10; // *10*

        Count\_en = 1; #10; // *09*

        Count\_en = 1; #10; // *08*

        Count\_en = 1; #10; // *07*

        Count\_en = 1; #10; // *06*

        Count\_en = 1; #10; // *05*

        Count\_en = 1; #10; // *04*

        Count\_en = 1; #10; // *03*

        Count\_en = 1; #10; // *02*

        Count\_en = 1; #10; // *01*

        Count\_en = 1; #10; // *00*

        // *Counting up*

        Up = 1; Count\_en = 1; #10;  // *01*

        Count\_en = 1; #10; // *02*

        Count\_en = 1; #10; // *03*

        Count\_en = 1; #10; // *04*

        Count\_en = 1; #10; // *05*

        Count\_en = 1; #10; // *06*

        Count\_en = 1; #10; // *07*

        $display("Resetting");

        nReset = 0; #10;

        nReset = 1; #10;

        $stop;

    end

    initial begin

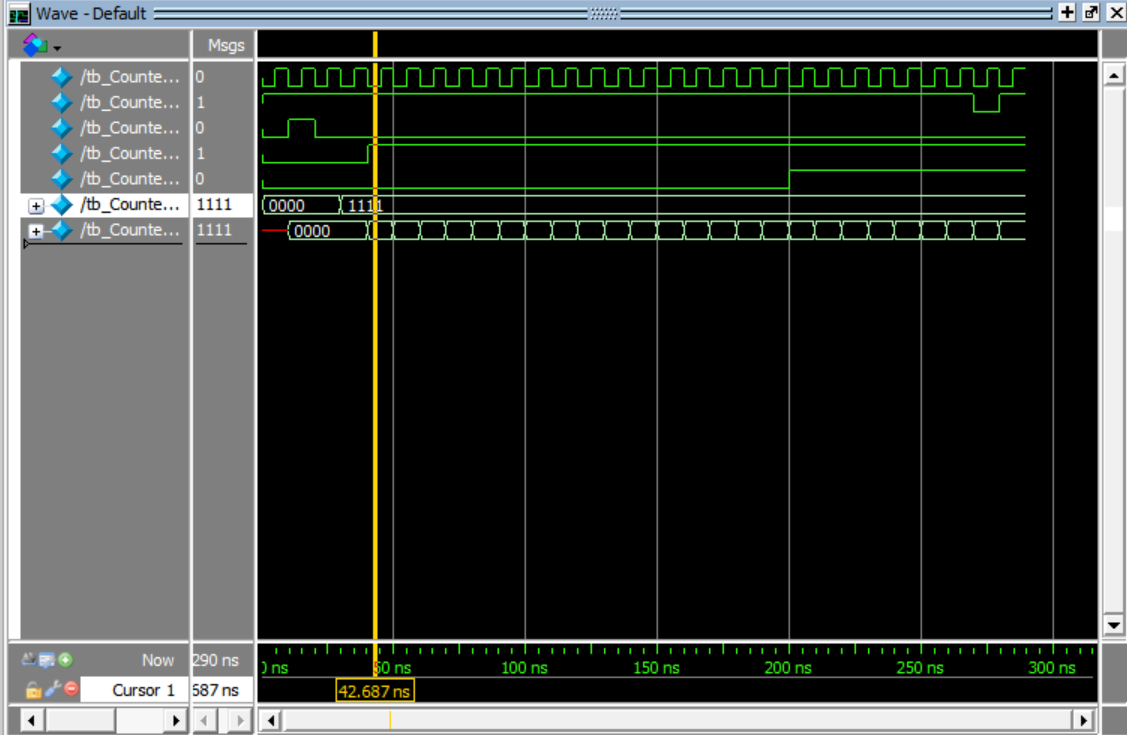
        $display("Test Bench | Counter\_4bit...");

        $monitor("Time = %2d ns | Clk = %b | nReset = %b | Load = %b | Count\_en = %b | Up = %b | Count\_in = %b | Count\_out = %d", $time, Clk, nReset, Load, Count\_en, Up, Count\_in, Count\_out);

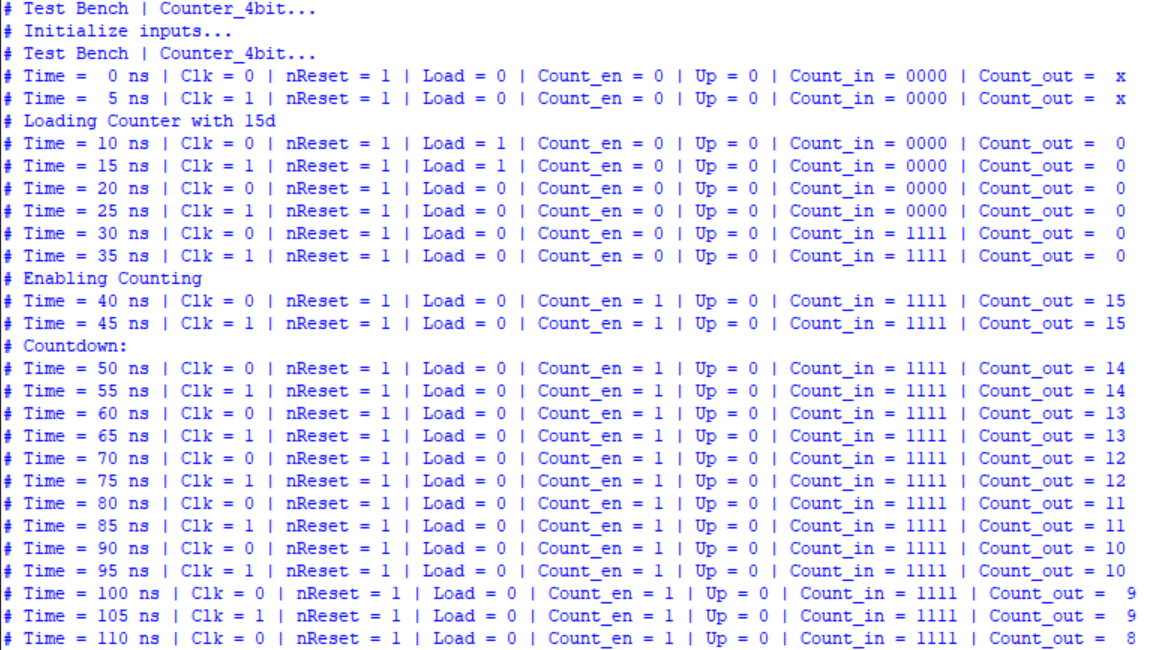
    end

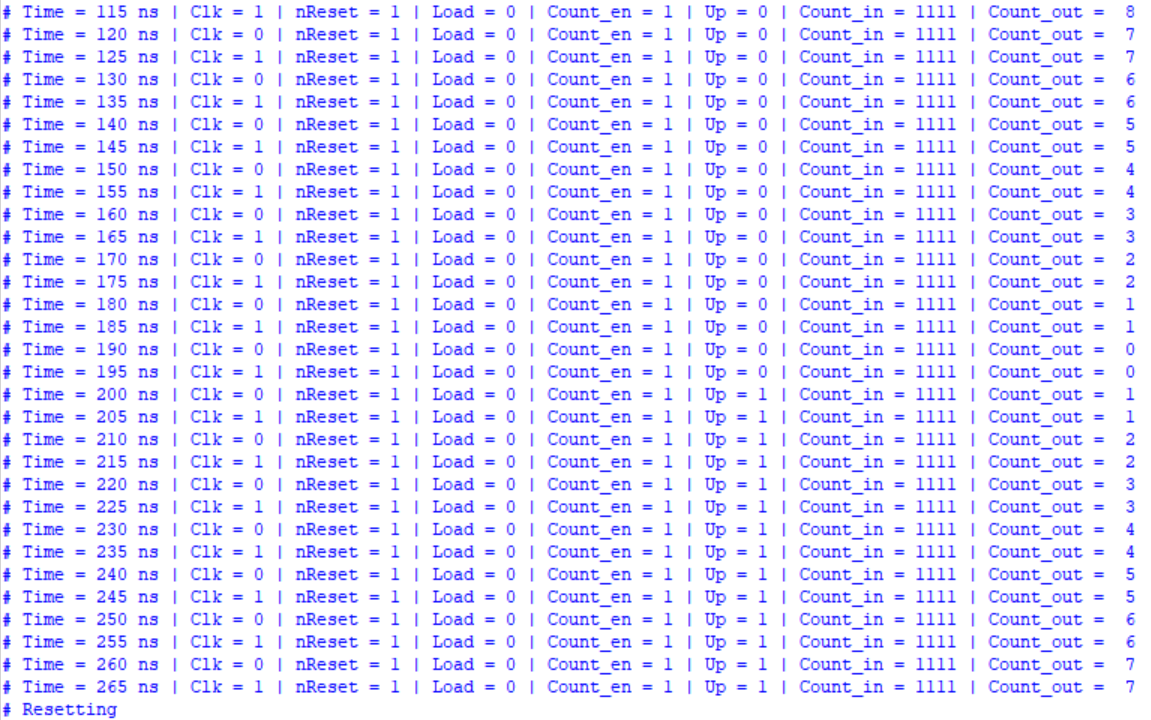
endmodule

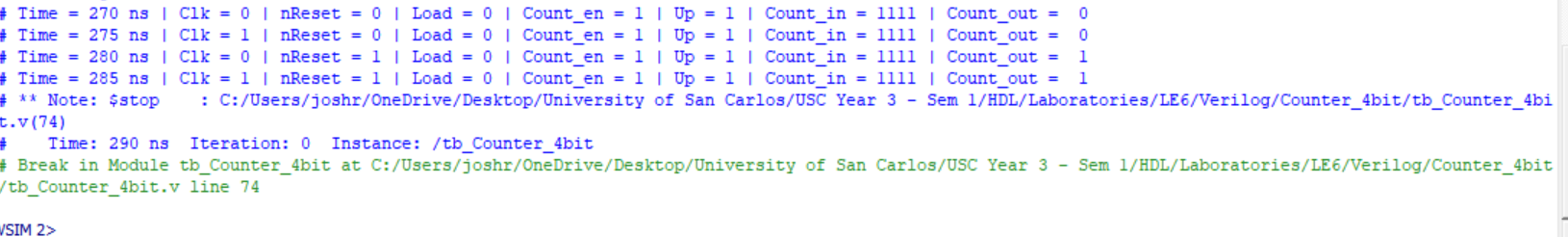
**Figure 2.5** – *tb\_ Counter\_4bit RTL Simulation Output*

**

**Figure 2.6** – *tb\_ Counter\_4bit Test Bench Monitor Output (Annotations to* ***Figure 2.5****)*

****

****

****

**Discussion of Results (Exercise 5B)**

The 4-bit counter operates as expected as observed in **Figure 2.5**. Through the test bench, we confirmed that the circuit can switch to countdown mode and as well count up mode. The asynchronous active low reset as well works as expected. There is no change when the clock is low which is also the behaviour that is desired. The **Figure 2.6** “$monitor” command really makes it easier to see that this behaviour is consistent.