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*

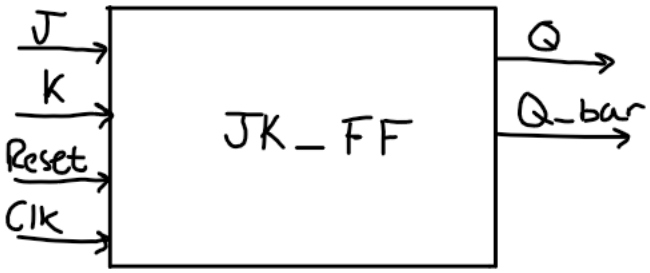

Figure 2.0 –*Counter_4bit Block Diagram*

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*

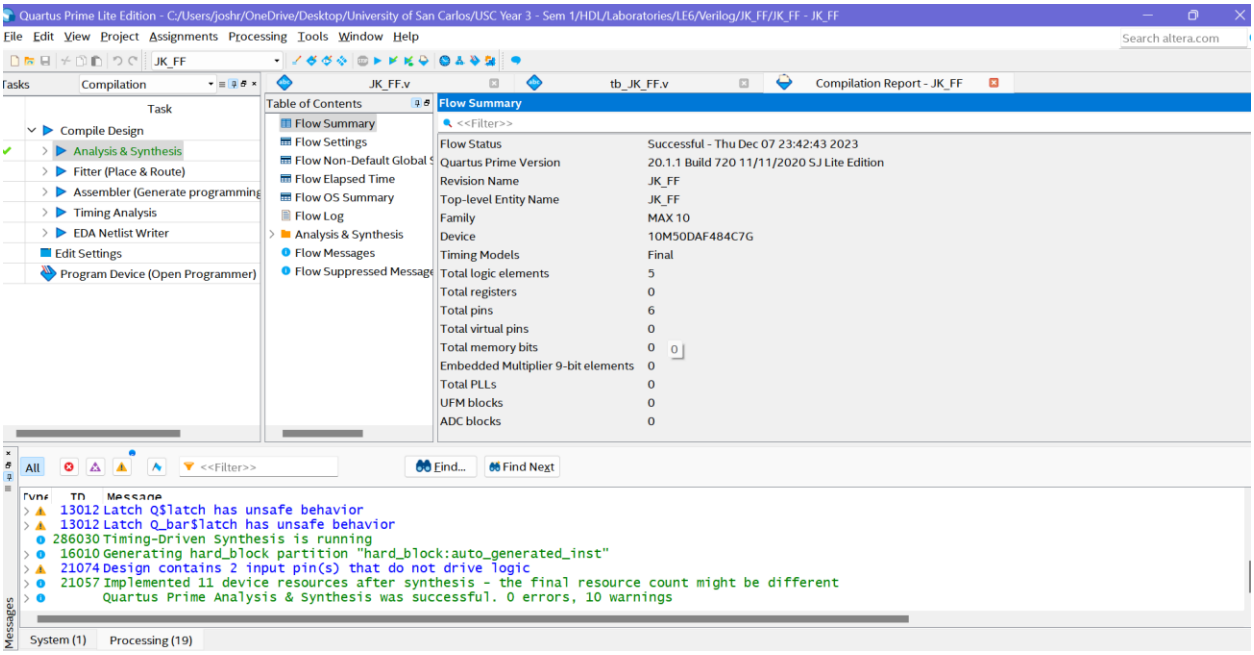

Figure 1.3 – *JK_FF RTL View Output*

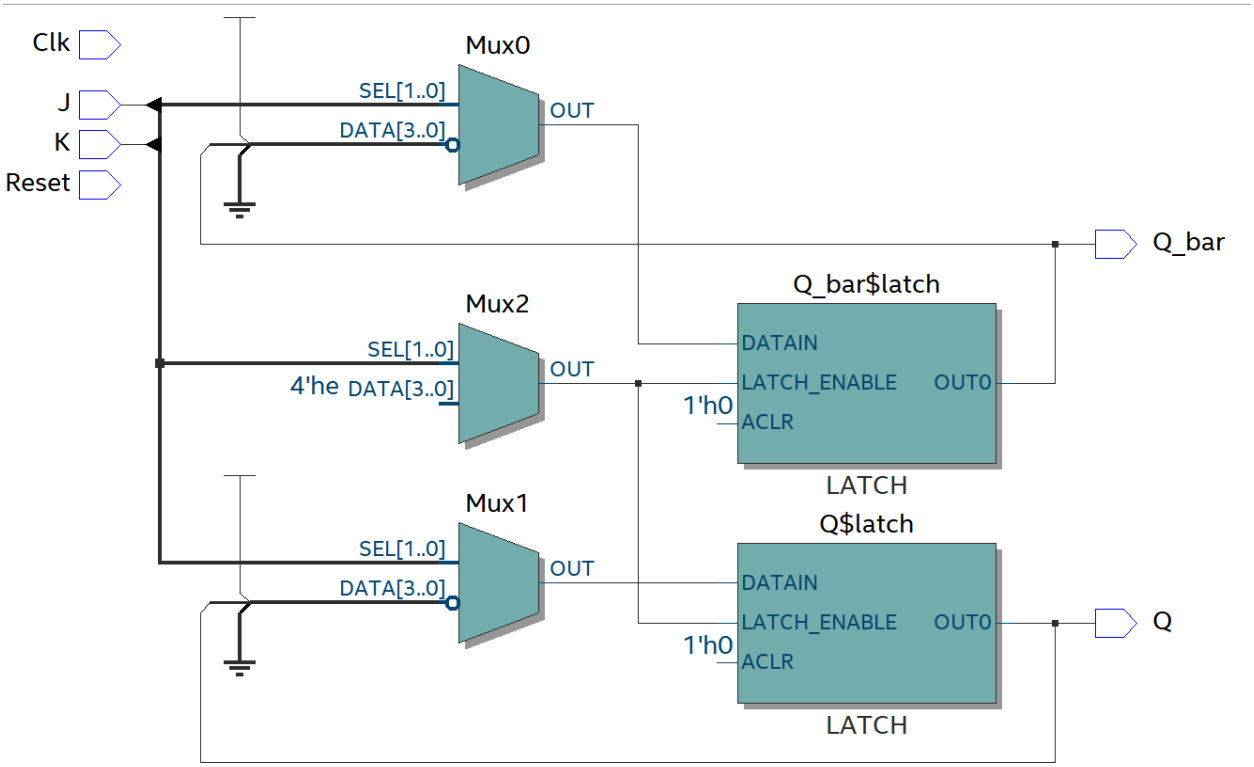

Figure 1.4 – *tb_ JK_FF.v Script for Testing Module*

```
/**********************************
*   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*

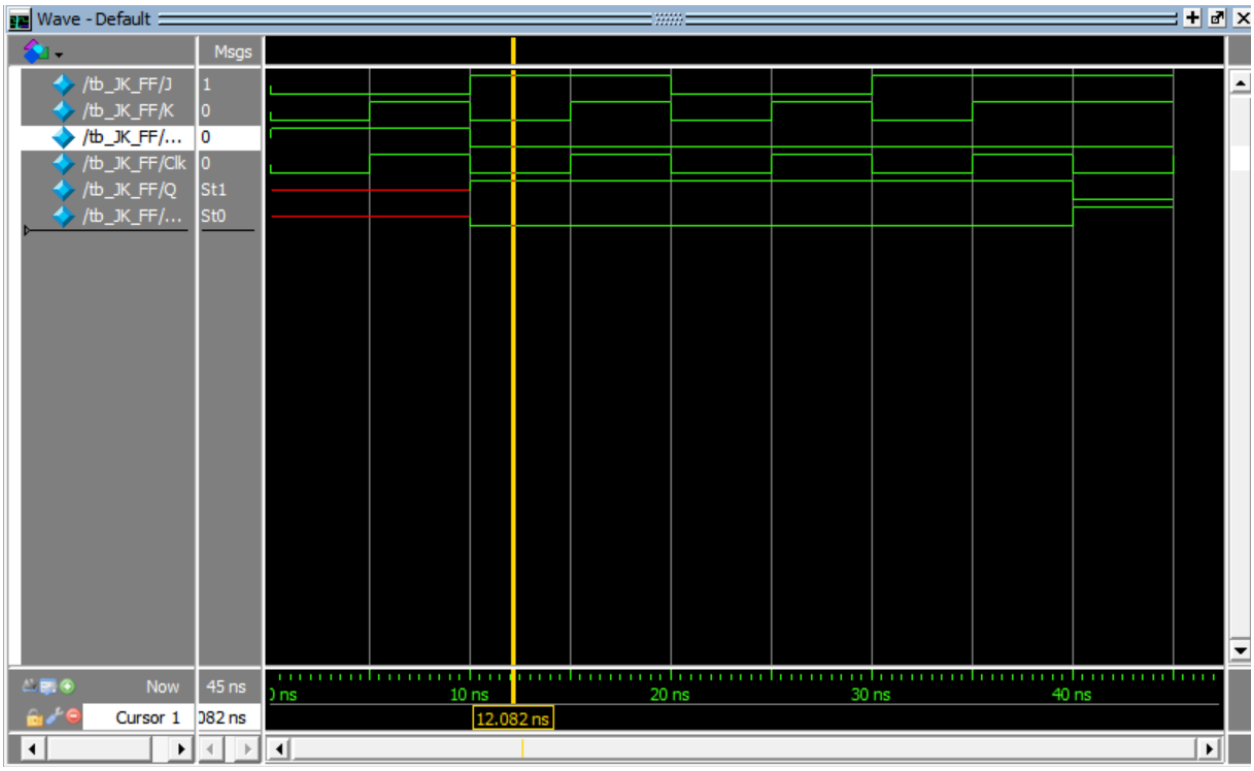

Figure 1.6 – *JK_FF Test Bench Monitor Output (Annotations to **Figure 1.5**)*

```
# Test Bench | JK FLIP FLOP...
# Time =  0 ns | J = 0 | K = 0 | CLK = 0 | RESET = 1 | Q = x | Q_bar = x
# Time =  5 ns | J = 0 | K = 1 | CLK = 1 | RESET = 1 | Q = x | Q_bar = x
# Time = 10 ns | J = 1 | K = 0 | CLK = 0 | RESET = 0 | Q = 1 | Q_bar = 0
# Time = 15 ns | J = 1 | K = 1 | CLK = 1 | RESET = 0 | Q = 1 | Q_bar = 0
# Time = 20 ns | J = 0 | K = 0 | CLK = 0 | RESET = 0 | Q = 1 | Q_bar = 0
# Time = 25 ns | J = 0 | K = 1 | CLK = 1 | RESET = 0 | Q = 1 | Q_bar = 0
# Time = 30 ns | J = 1 | K = 0 | CLK = 0 | RESET = 0 | Q = 1 | Q_bar = 0
# Time = 35 ns | J = 1 | K = 1 | CLK = 1 | RESET = 0 | Q = 1 | Q_bar = 0
# Time = 40 ns | J = 1 | K = 1 | CLK = 0 | RESET = 0 | Q = 0 | Q_bar = 1
# ** Note: $stop    : C:/Users/joshr/OneDrive/Desktop/University of San Carlos/USC Year 3 - S
em 1/HDL/Laboratories/LE6/Verilog/JK_FF/tb_JK_FF.v(40)
#    Time: 45 ns  Iteration: 0  Instance: /tb_JK_FF
# Break in Module tb_JK_FF at C:/Users/joshr/OneDrive/Desktop/University of San Carlos/USC Ye
ar 3 - Sem 1/HDL/Laboratories/LE6/Verilog/JK_FF/tb_JK_FF.v line 40
```

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*

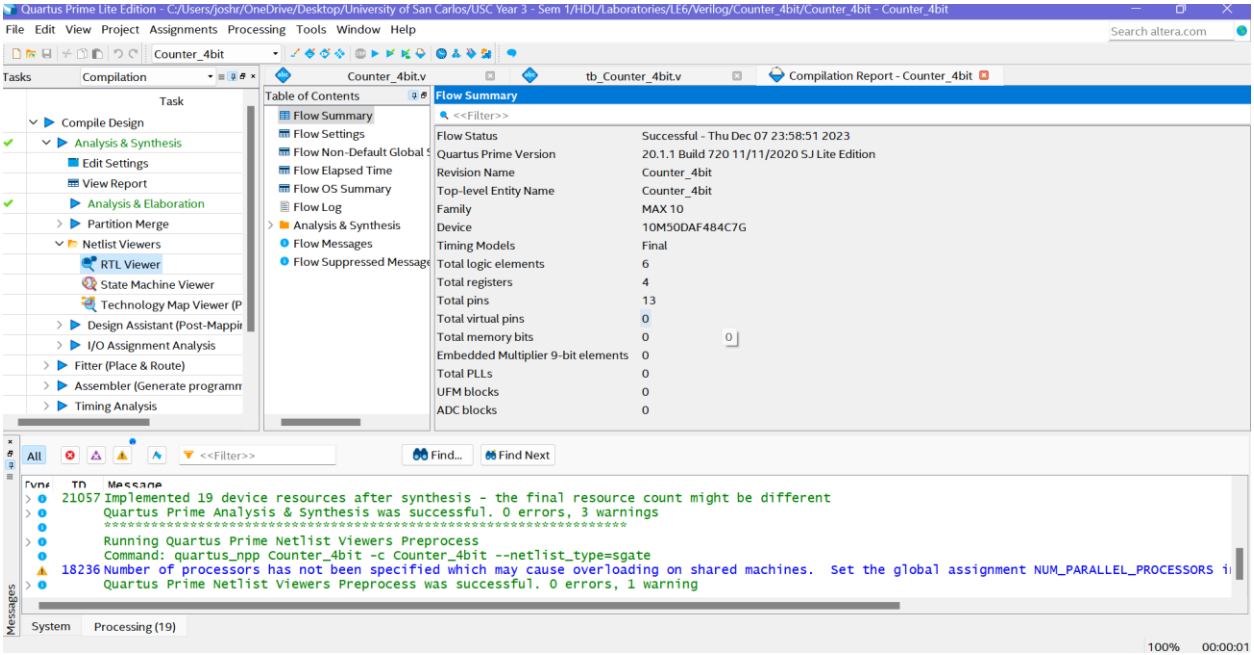

Figure 2.3 – *Counter_4bit RTL View Output*

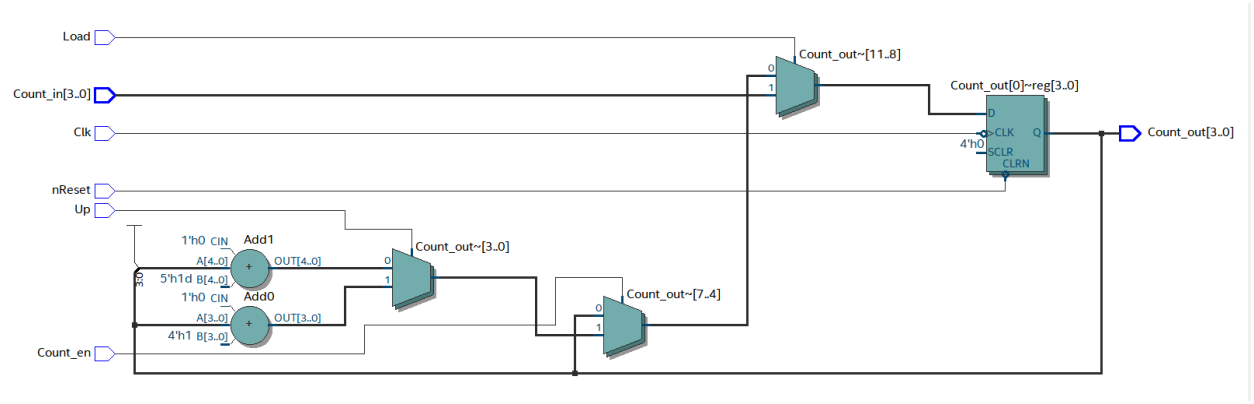

Figure 2.4 – *tb_ Counter_4bit.v Script for Testing Module*

```
/************************************
*   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*

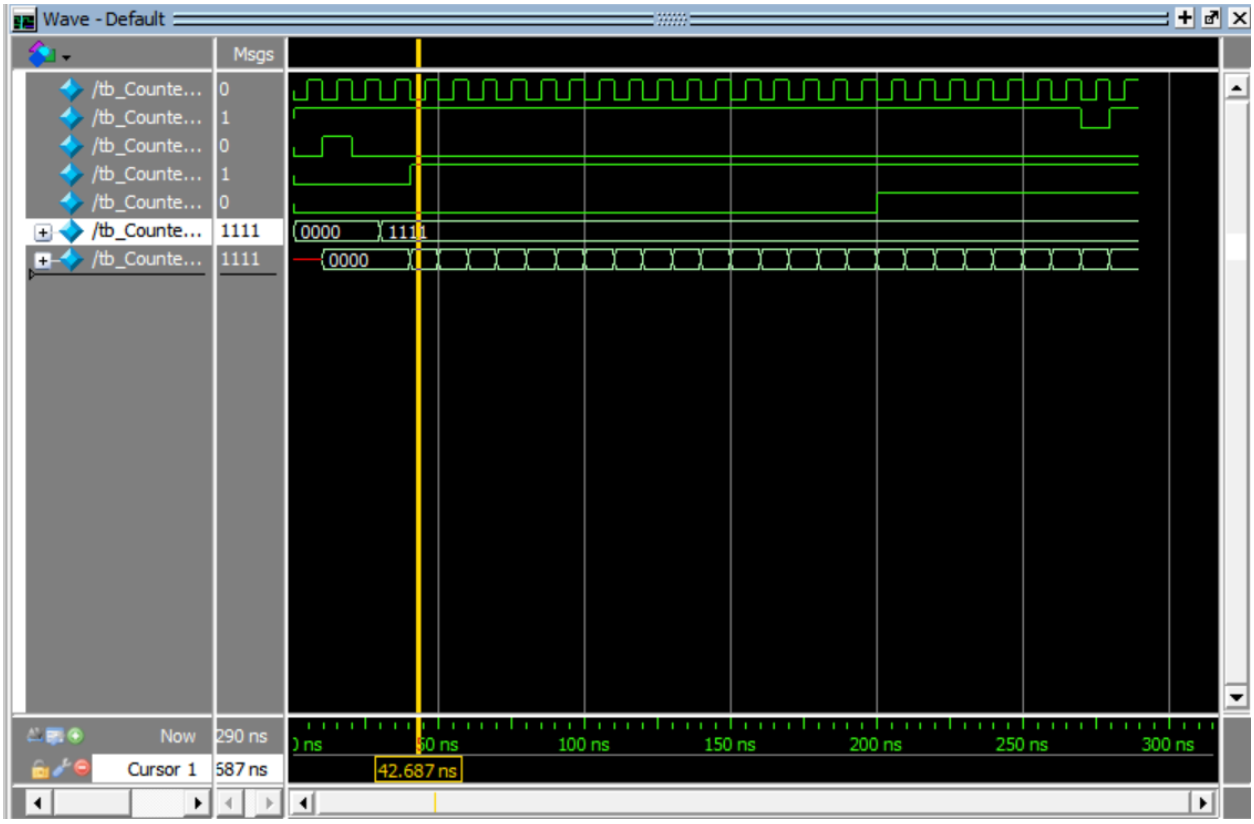

Figure 2.6 – *tb_ Counter_4bit Test Bench Monitor Output (Annotations to **Figure 2.5**)*

```
# Test Bench | Counter_4bit...
# Initialize inputs...
# Test Bench | Counter_4bit...
# Time =  0 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 0 | Up = 0 | Count_in = 0000 | Count_out =  x
# Time =  5 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 0 | Up = 0 | Count_in = 0000 | Count_out =  x
# Loading Counter with 15d
# Time = 10 ns | Clk = 0 | nReset = 1 | Load = 1 | Count_en = 0 | Up = 0 | Count_in = 0000 | Count_out =  0
# Time = 15 ns | Clk = 1 | nReset = 1 | Load = 1 | Count_en = 0 | Up = 0 | Count_in = 0000 | Count_out =  0
# Time = 20 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 0 | Up = 0 | Count_in = 0000 | Count_out =  0
# Time = 25 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 0 | Up = 0 | Count_in = 0000 | Count_out =  0
# Time = 30 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 0 | Up = 0 | Count_in = 1111 | Count_out =  0
# Time = 35 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 0 | Up = 0 | Count_in = 1111 | Count_out =  0
# Enabling Counting
# Time = 40 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out = 15
# Time = 45 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out = 15
# Countdown:
# Time = 50 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out = 14
# Time = 55 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out = 14
# Time = 60 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out = 13
# Time = 65 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out = 13
# Time = 70 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out = 12
# Time = 75 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out = 12
# Time = 80 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out = 11
# Time = 85 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out = 11
# Time = 90 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out = 10
# Time = 95 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out = 10
# Time = 100 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  9
# Time = 105 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  9
# Time = 110 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  8
```

```
# Time = 115 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  8
# Time = 120 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  7
# Time = 125 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  7
# Time = 130 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  6
# Time = 135 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  6
# Time = 140 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  5
# Time = 145 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  5
# Time = 150 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  4
# Time = 155 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  4
# Time = 160 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  3
# Time = 165 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  3
# Time = 170 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  2
# Time = 175 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  2
# Time = 180 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  1
# Time = 185 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  1
# Time = 190 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  0
# Time = 195 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 0 | Count_in = 1111 | Count_out =  0
# Time = 200 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  1
# Time = 205 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  1
# Time = 210 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  2
# Time = 215 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  2
# Time = 220 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  3
# Time = 225 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  3
# Time = 230 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  4
# Time = 235 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  4
# Time = 240 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  5
# Time = 245 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  5
# Time = 250 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  6
# Time = 255 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  6
# Time = 260 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  7
# Time = 265 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  7
# Resetting
```

```
# Time = 270 ns | Clk = 0 | nReset = 0 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  0
# Time = 275 ns | Clk = 1 | nReset = 0 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  0
# Time = 280 ns | Clk = 0 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  1
# Time = 285 ns | Clk = 1 | nReset = 1 | Load = 0 | Count_en = 1 | Up = 1 | Count_in = 1111 | Count_out =  1
# ** Note: $stop    : C:/Users/joshr/OneDrive/Desktop/University of San Carlos/USC Year 3 - Sem 1/HDL/Laboratories/LE6/Verilog/Counter_4bit/tb_Counter_4bi
t.v(74)
#    Time: 290 ns  Iteration: 0  Instance: /tb_Counter_4bit
# Break in Module tb_Counter_4bit at C:/Users/joshr/OneDrive/Desktop/University of San Carlos/USC Year 3 - Sem 1/HDL/Laboratories/LE6/Verilog/Counter_4bit
/tb_Counter_4bit.v line 74

VSIM 2>
```

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.