ECEN 248 - Lab Report

Lab Number: 8

Lab Title: Introduction to Logic Simulation and Verilog

Student's Name: Paola Avila

Student's UIN: 731007033

Date:4/4/2024

Objective

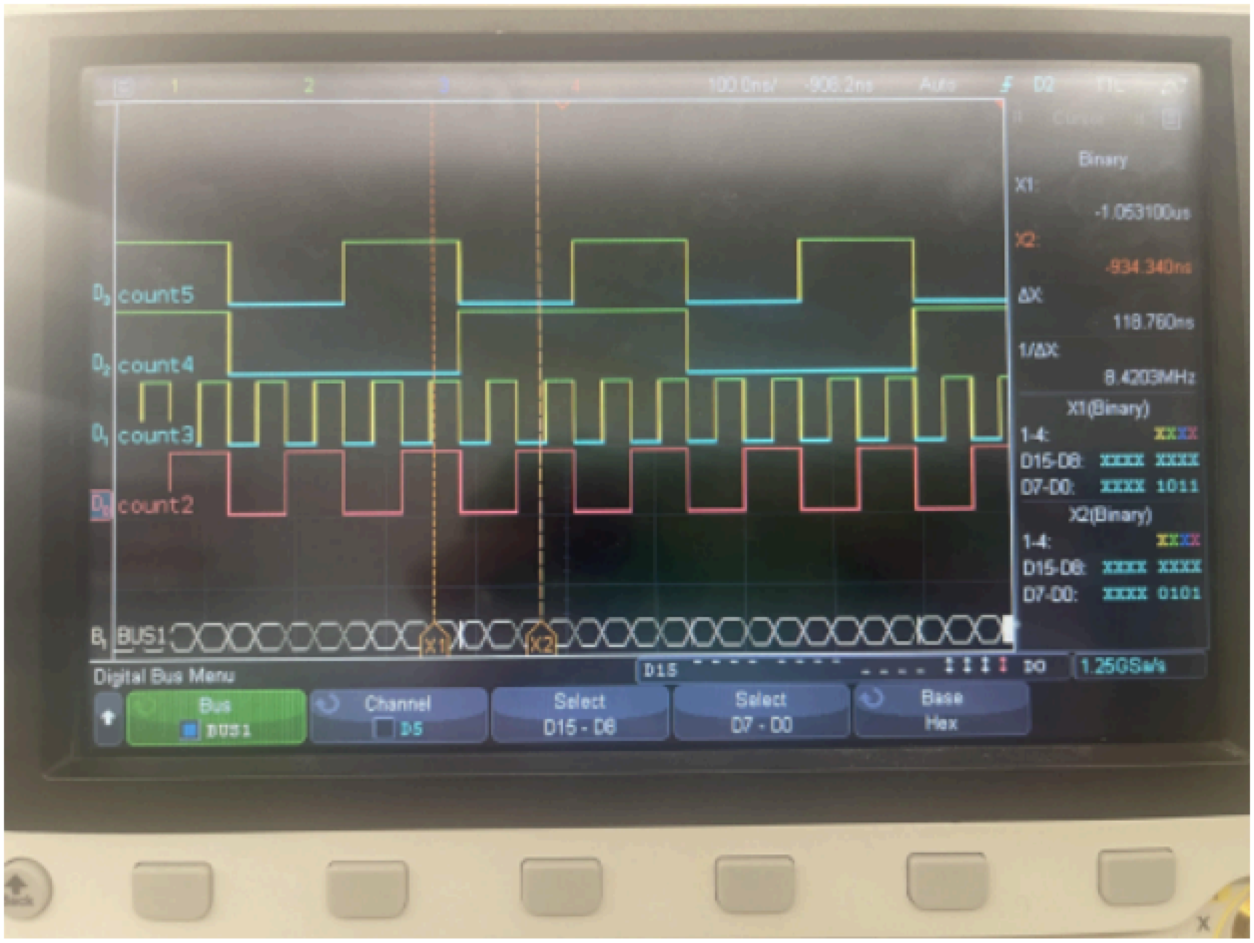
The objective of this lab was to develop a deeper understanding of sequential and synchronous circuits by designing and implementing a counter. Additionally, this lab provided an opportunity to gain hands-on experience with an oscilloscope for the first time.

Design

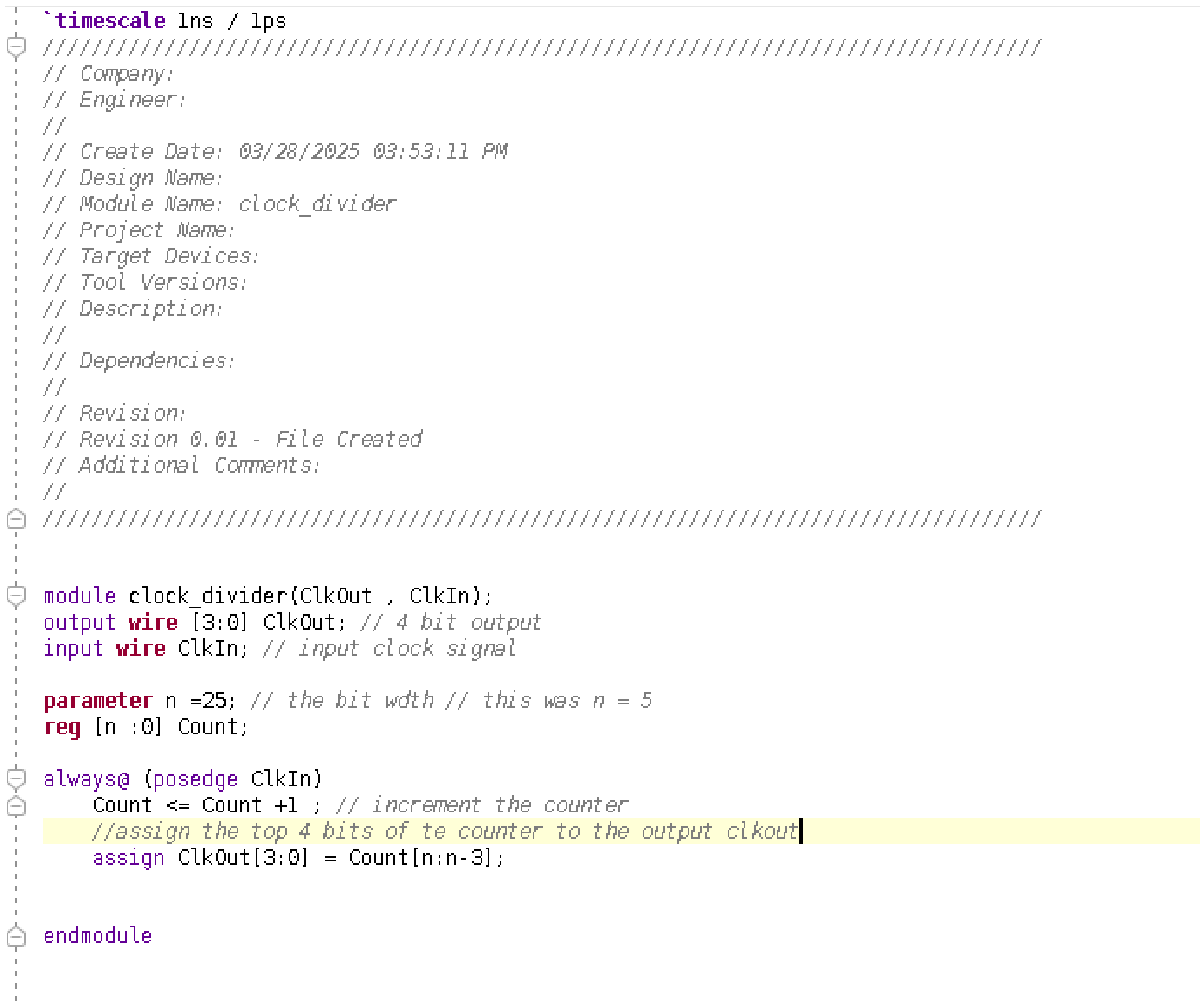
We began by designing a clock divider to generate a slower clock signal from a high-frequency source. Using an oscilloscope, we measured and verified the frequency reduction on the Zybo Z7-10 board. Next, we constructed a counter by integrating half adders, an up-counter, and the previously developed clock divider. The complete top-level design was programmed onto the Zybo Z7-10 using a custom XDC file. On the Zybo board, we displayed the output of our 7-bit counter, along with the carry bit, across four LEDs, and controlled its operation using two switches and two push buttons. Finally, we investigated the effects of button bounce by analyzing the outputs of both the noDebounce and withDebounce implementations. Through waveform analysis, we observed the impact of signal delay on the circuit's output behavior.

Results :

Experiment Part 1:

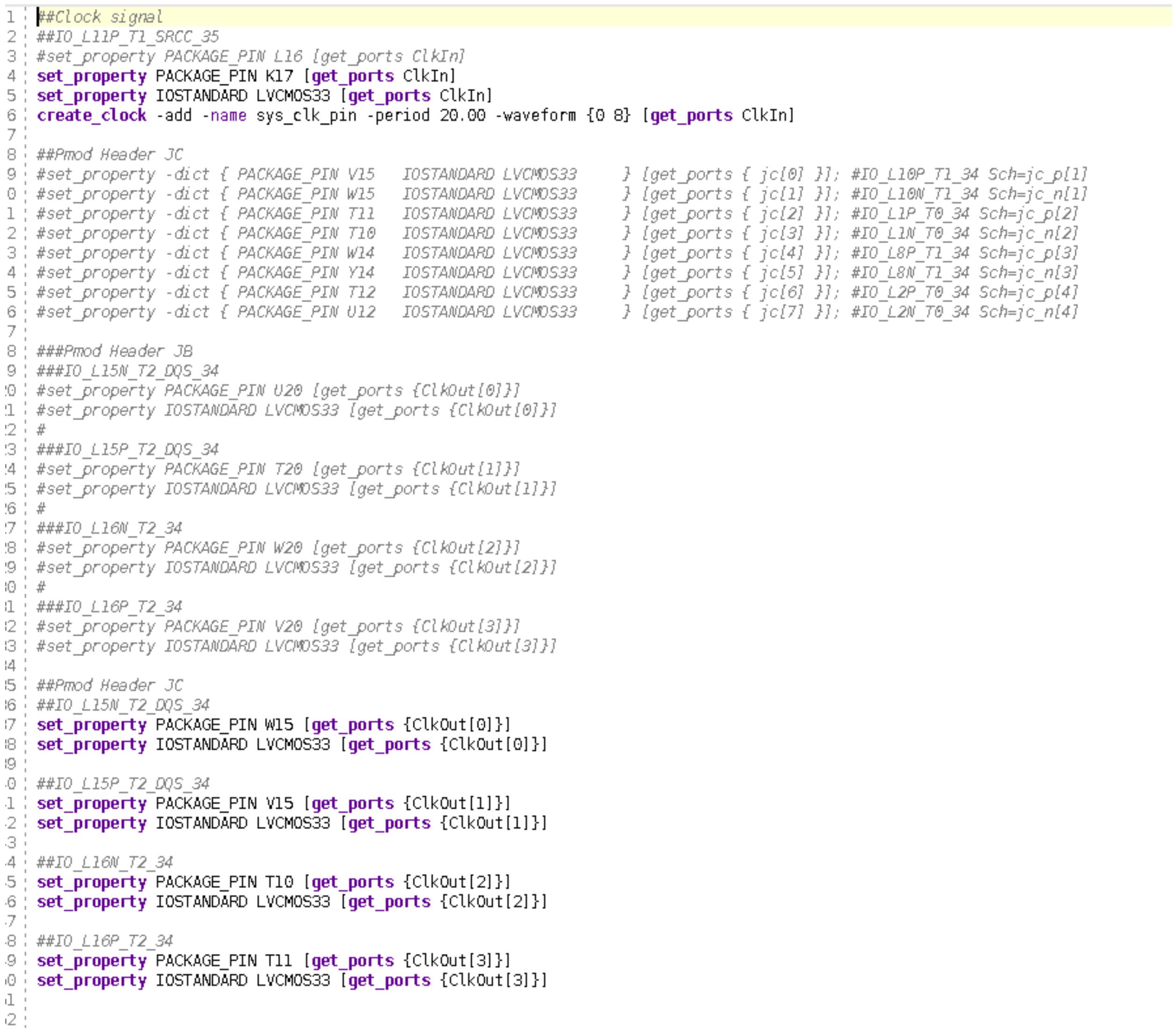


Clock divider . v



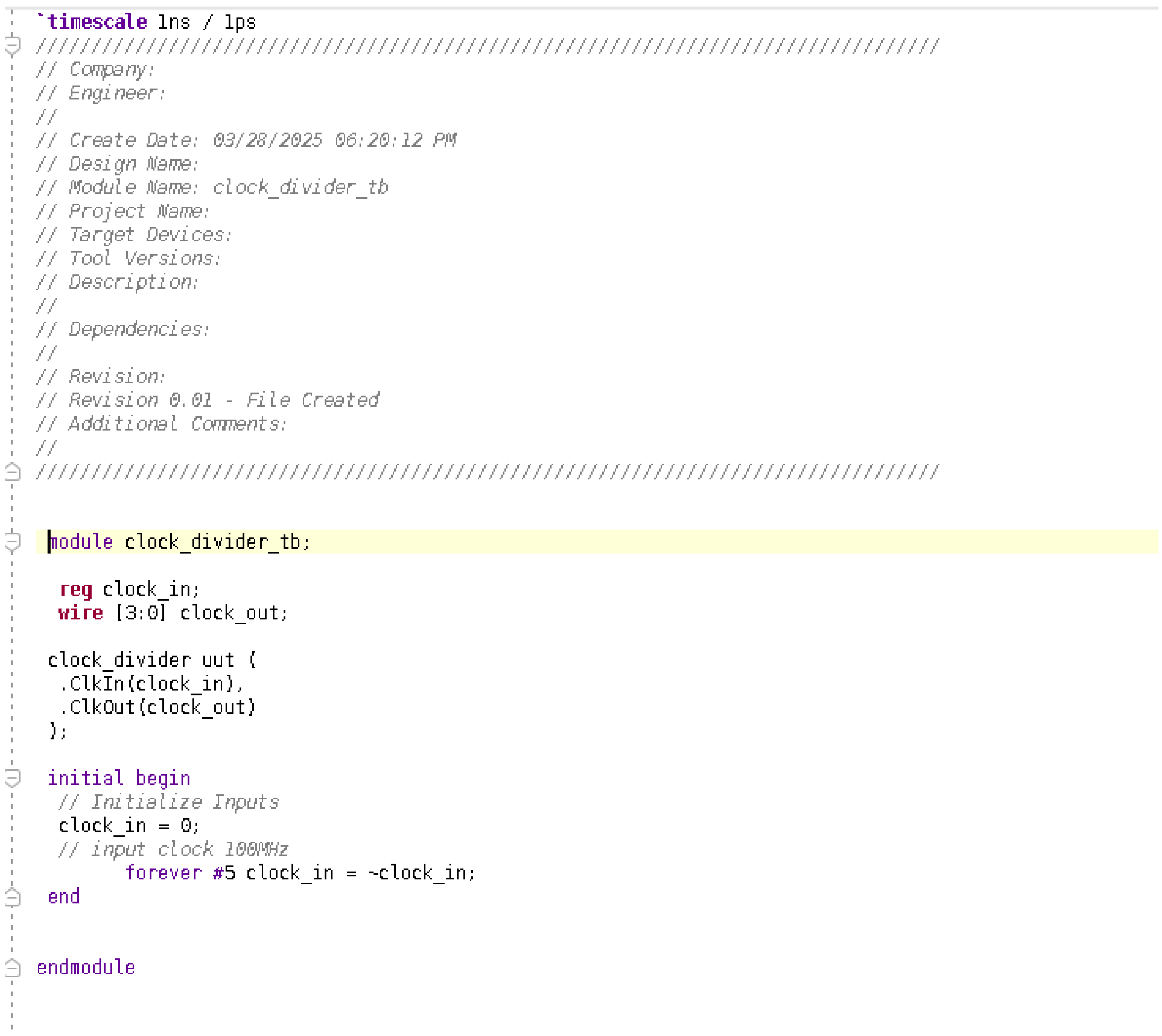
```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/28/2025 03:53:11 PM
// Design Name:
// Module Name: clock_divider
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module clock_divider(ClkOut , ClkIn);
output wire [3:0] ClkOut; // 4 bit output
input wire ClkIn; // input clock signal

parameter n =25; // the bit wdth // this was n = 5
reg [n :0] Count;

always@ (posedge ClkIn)
    Count <= Count +1 ; // increment the counter
    //assign the top 4 bits of te counter to the output clkout
    assign ClkOut[3:0] = Count[n:n-3];


endmodule
```

Clock_divider .xdc



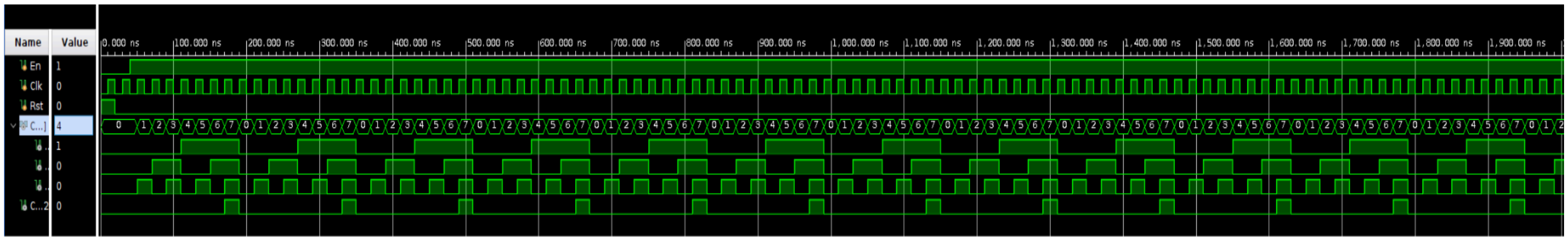
```
##Clock signal
##IO_L11P_T1_SRCC_35
#set_property PACKAGE_PIN L16 [get_ports ClkIn]
set_property PACKAGE_PIN K17 [get_ports ClkIn]
set_property IOSTANDARD LVCMOS33 [get_ports ClkIn]
create_clock -add -name sys_clk_pin -period 20.00 -waveform {0 8} [get_ports ClkIn]

##Pmod Header JC
#set_property -dict { PACKAGE_PIN V15   IOSTANDARD LVCMOS33     } [get_ports { jc[0] }]; #IO_L10P_T1_34 Sch=jc_p[1]
#set_property -dict { PACKAGE_PIN W15   IOSTANDARD LVCMOS33     } [get_ports { jc[1] }]; #IO_L10N_T1_34 Sch=jc_n[1]
#set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33     } [get_ports { jc[2] }]; #IO_L1P_T0_34 Sch=jc_p[2]
#set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33     } [get_ports { jc[3] }]; #IO_L1N_T0_34 Sch=jc_n[2]
#set_property -dict { PACKAGE_PIN W14   IOSTANDARD LVCMOS33     } [get_ports { jc[4] }]; #IO_L8P_T1_34 Sch=jc_p[3]
#set_property -dict { PACKAGE_PIN Y14   IOSTANDARD LVCMOS33     } [get_ports { jc[5] }]; #IO_L8N_T1_34 Sch=jc_n[3]
#set_property -dict { PACKAGE_PIN T12   IOSTANDARD LVCMOS33     } [get_ports { jc[6] }]; #IO_L2P_T0_34 Sch=jc_p[4]
#set_property -dict { PACKAGE_PIN U12   IOSTANDARD LVCMOS33     } [get_ports { jc[7] }]; #IO_L2N_T0_34 Sch=jc_n[4]

###Pmod Header JB
###IO_L15N_T2_DQS_34
#set_property PACKAGE_PIN U20 [get_ports {ClkOut[0]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {ClkOut[0]}]
#
###IO_L15P_T2_DQS_34
#set_property PACKAGE_PIN T20 [get_ports {ClkOut[1]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {ClkOut[1]}]
#
###IO_L16N_T2_34
#set_property PACKAGE_PIN W20 [get_ports {ClkOut[2]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {ClkOut[2]}]
#
###IO_L16P_T2_34
#set_property PACKAGE_PIN V20 [get_ports {ClkOut[3]}]
#set_property IOSTANDARD LVCMOS33 [get_ports {ClkOut[3]}]

##Pmod Header JC
##IO_L15N_T2_DQS_34
set_property PACKAGE_PIN W15 [get_ports {ClkOut[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ClkOut[0]}]

##IO_L15P_T2_DQS_34
set_property PACKAGE_PIN V15 [get_ports {ClkOut[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ClkOut[1]}]

##IO_L16N_T2_34
set_property PACKAGE_PIN T10 [get_ports {ClkOut[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ClkOut[2]}]

##IO_L16P_T2_34
set_property PACKAGE_PIN T11 [get_ports {ClkOut[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {ClkOut[3]}]
```

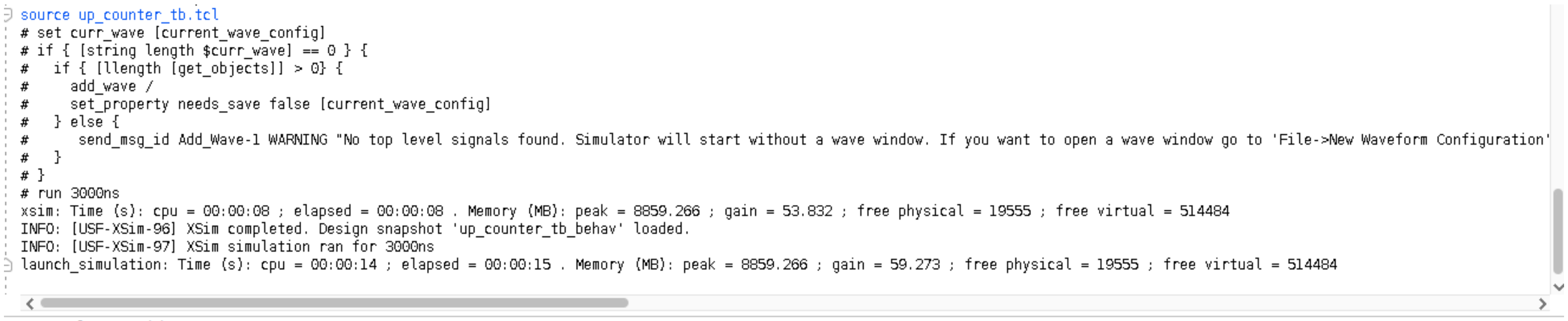
Clock_divider_tb :



```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/28/2025 06:20:12 PM
// Design Name:
// Module Name: clock_divider_tb
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


 module clock_divider_tb;

  reg clock_in;
  wire [3:0] clock_out;

 clock_divider uut (
  .ClkIn(clock_in),
  .ClkOut(clock_out)
 );

 initial begin
  // Initialize Inputs
  clock_in = 0;
  // input clock 100MHz
        forever #5 clock_in = ~clock_in;
 end


endmodule
```

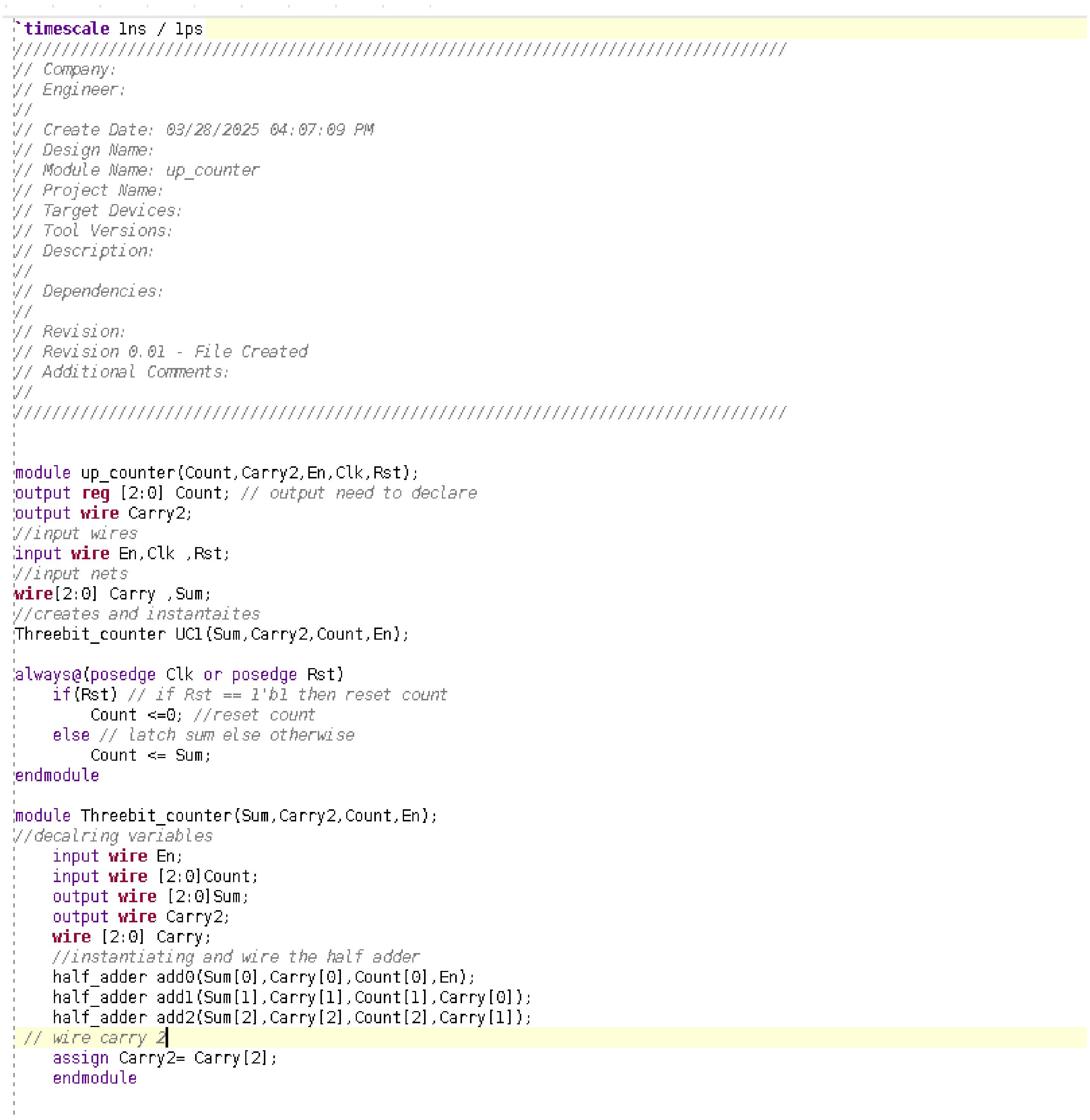
Experiment Part 2





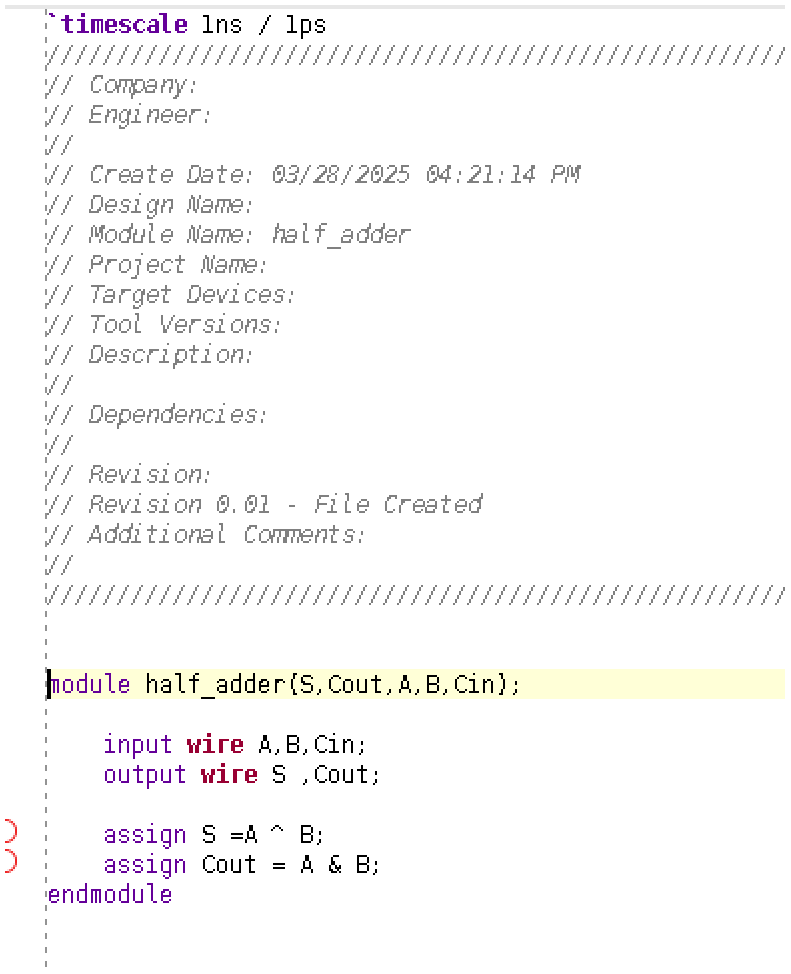
```
source up_counter_tb.tcl
# set curr_wave [current_wave_config]
# if { [string length $curr_wave] == 0 } {
#   if { [llength [get_objects]] > 0} {
#     add_wave /
#     set_property needs_save false [current_wave_config]
#   } else {
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave window go to 'File->New Waveform Configuration'
#   }
# }
# run 3000ns
xsim: Time (s): cpu = 00:00:08 ; elapsed = 00:00:08 . Memory (MB): peak = 8859.266 ; gain = 53.832 ; free physical = 19555 ; free virtual = 514484
INFO: [USF-XSim-96] XSim completed. Design snapshot 'up_counter_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 3000ns
launch_simulation: Time (s): cpu = 00:00:14 ; elapsed = 00:00:15 . Memory (MB): peak = 8859.266 ; gain = 59.273 ; free physical = 19555 ; free virtual = 514484
```

Up_counter.V



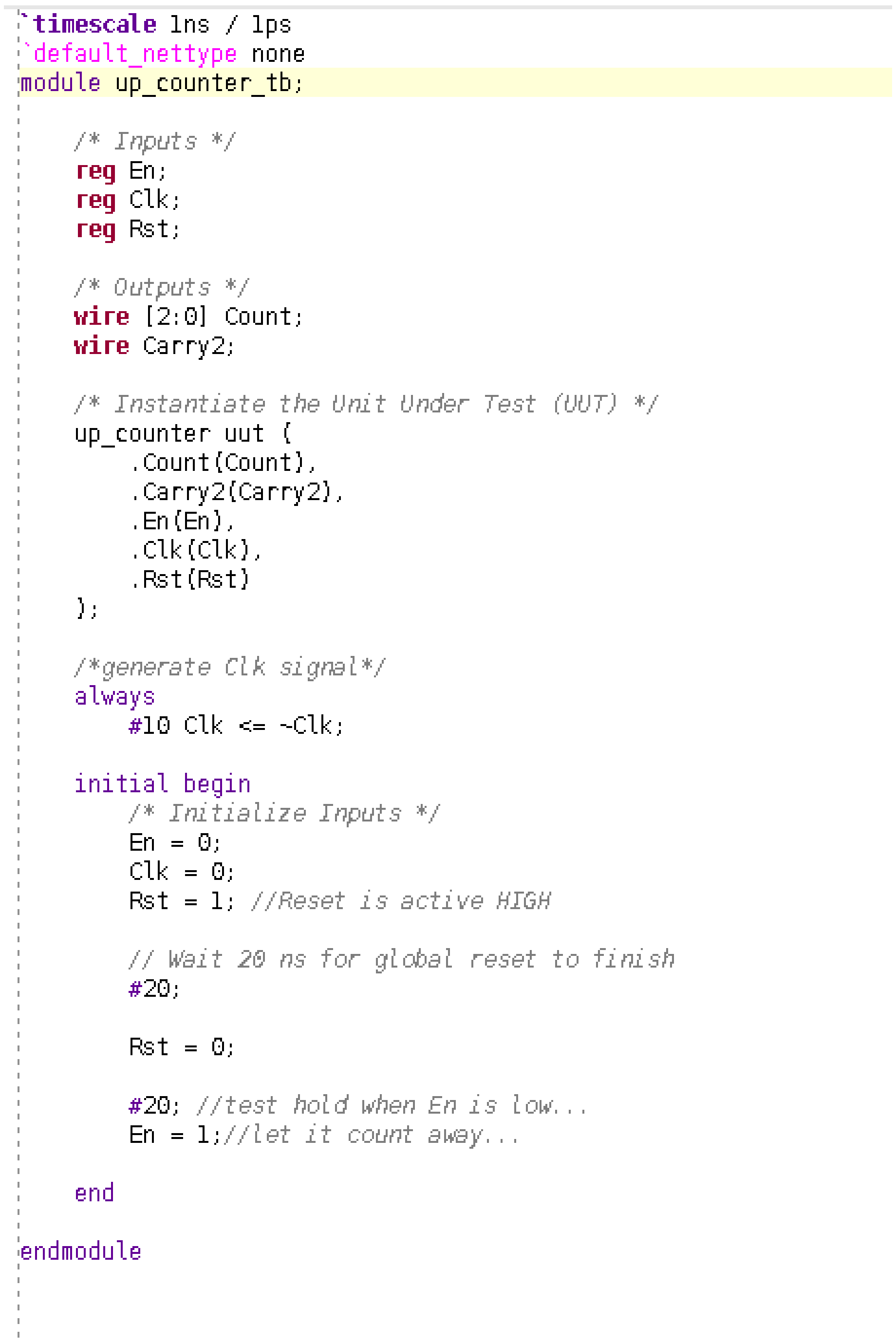
```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/28/2025 04:07:09 PM
// Design Name:
// Module Name: up_counter
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module up_counter(Count,Carry2,En,Clk,Rst);
output reg [2:0] Count; // output need to declare
output wire Carry2;
//input wires
input wire En,Clk ,Rst;
//input nets
wire[2:0] Carry ,Sum;
//creates and instantaites
Threebit_counter UC1(Sum,Carry2,Count,En);

always@(posedge Clk or posedge Rst)
    if(Rst) // if Rst == 1'b1 then reset count
        Count <=0; //reset count
    else // latch sum else otherwise
        Count <= Sum;
endmodule

module Threebit_counter(Sum,Carry2,Count,En);
//decalring variables
    input wire En;
    input wire [2:0]Count;
    output wire [2:0]Sum;
    output wire Carry2;
    wire [2:0] Carry;
    //instantiating and wire the half adder
    half_adder add0(Sum[0],Carry[0],Count[0],En);
    half_adder add1(Sum[1],Carry[1],Count[1],Carry[0]);
    half_adder add2(Sum[2],Carry[2],Count[2],Carry[1]);
 // wire carry 2
    assign Carry2= Carry[2];
    endmodule
```

Half_adder



```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/28/2025 04:21:14 PM
// Design Name:
// Module Name: half_adder
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module half_adder(S,Cout,A,B,Cin);

    input wire A,B,Cin;
    output wire S ,Cout;

    assign S =A ^ B;
    assign Cout = A & B;
endmodule
```

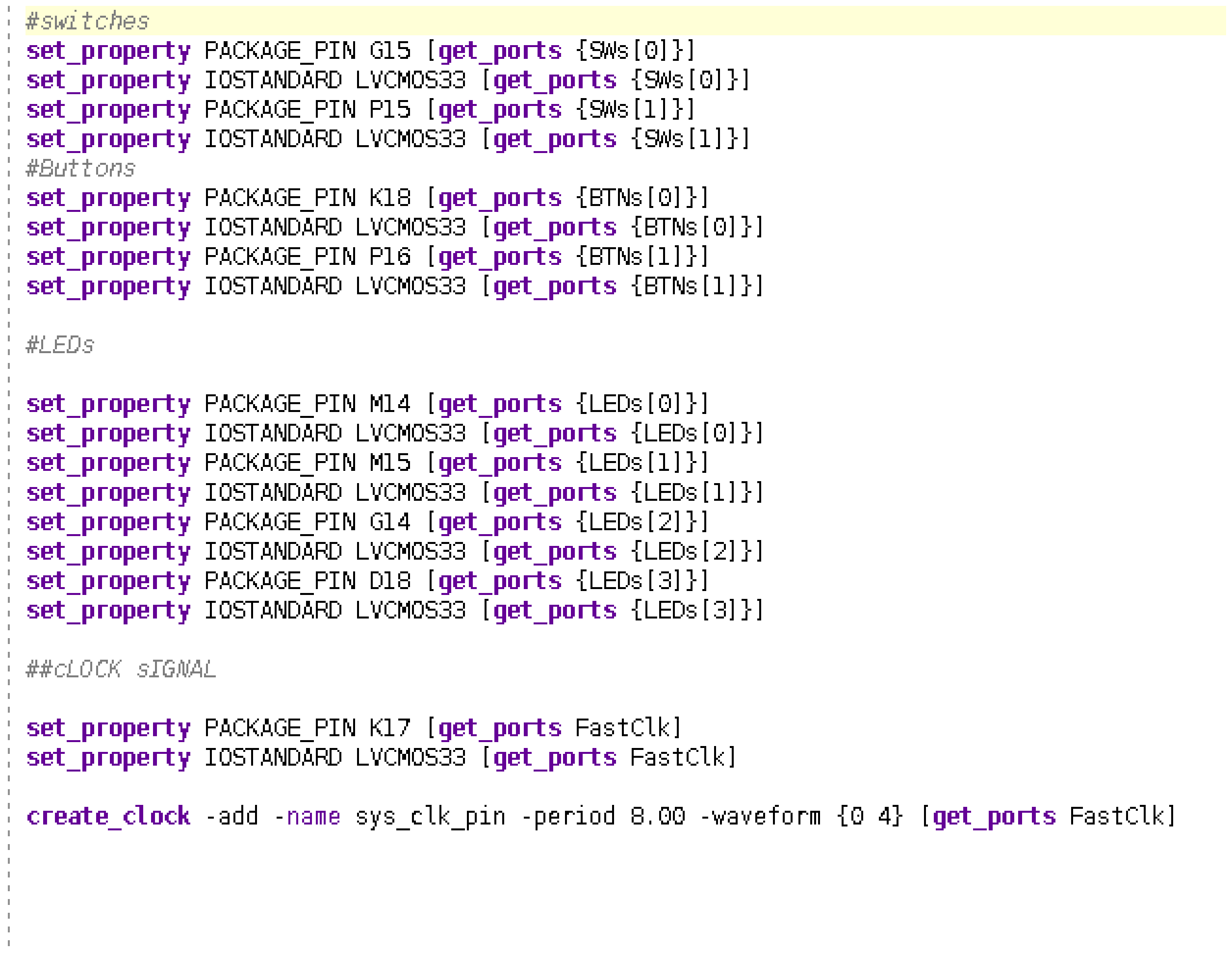
Up_counter_tb



```
`timescale 1ns / 1ps
`default_nettype none
module up_counter_tb;

    /* Inputs */
    reg En;
    reg Clk;
    reg Rst;

    /* Outputs */
    wire [2:0] Count;
    wire Carry2;

    /* Instantiate the Unit Under Test (UUT) */
    up_counter uut (
        .Count(Count),
        .Carry2(Carry2),
        .En(En),
        .Clk(Clk),
        .Rst(Rst)
    );

    /*generate Clk signal*/
    always
        #10 Clk <= ~Clk;

    initial begin
        /* Initialize Inputs */
        En = 0;
        Clk = 0;
        Rst = 1; //Reset is active HIGH

        // Wait 20 ns for global reset to finish
        #20;

        Rst = 0;

        #20; //test hold when En is low...
        En = 1;//let it count away...

    end

endmodule
```

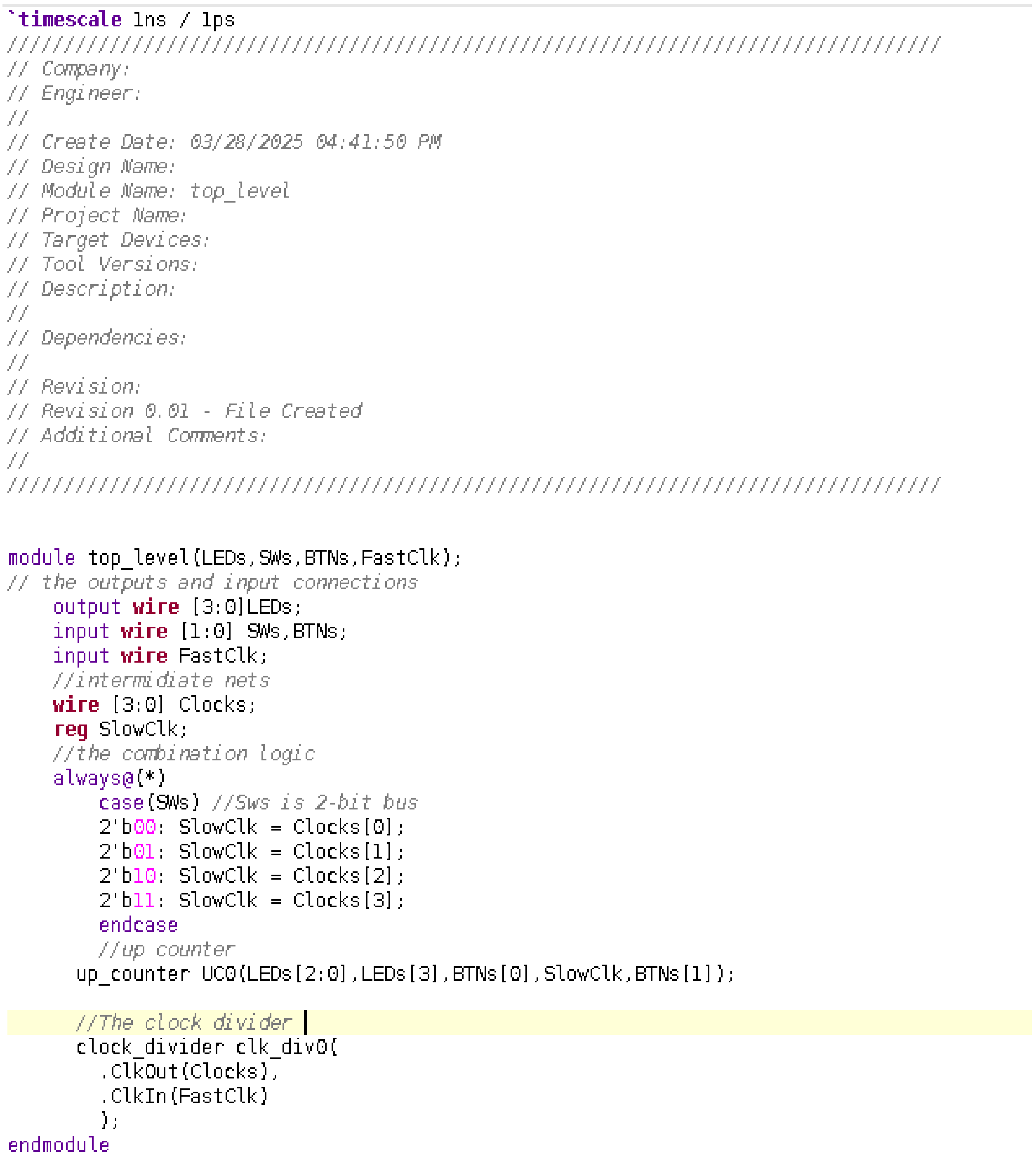
Experiment Part 2 Top Level

Top_level XDC :



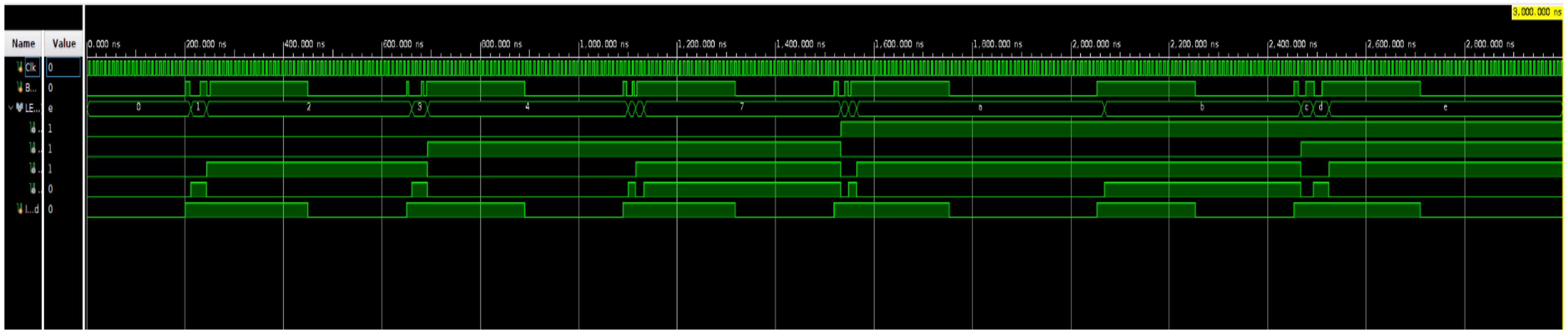
```
#switches
set_property PACKAGE_PIN G15 [get_ports {SWs[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SWs[0]}]
set_property PACKAGE_PIN P15 [get_ports {SWs[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {SWs[1]}]
#Buttons
set_property PACKAGE_PIN K18 [get_ports {BTNs[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {BTNs[0]}]
set_property PACKAGE_PIN P16 [get_ports {BTNs[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {BTNs[1]}]

#LEDs

set_property PACKAGE_PIN M14 [get_ports {LEDs[0]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LEDs[0]}]
set_property PACKAGE_PIN M15 [get_ports {LEDs[1]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LEDs[1]}]
set_property PACKAGE_PIN G14 [get_ports {LEDs[2]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LEDs[2]}]
set_property PACKAGE_PIN D18 [get_ports {LEDs[3]}]
set_property IOSTANDARD LVCMOS33 [get_ports {LEDs[3]}]

##cLOCK sIGNAL

set_property PACKAGE_PIN K17 [get_ports FastClk]
set_property IOSTANDARD LVCMOS33 [get_ports FastClk]

create_clock -add -name sys_clk_pin -period 8.00 -waveform {0 4} [get_ports FastClk]
```

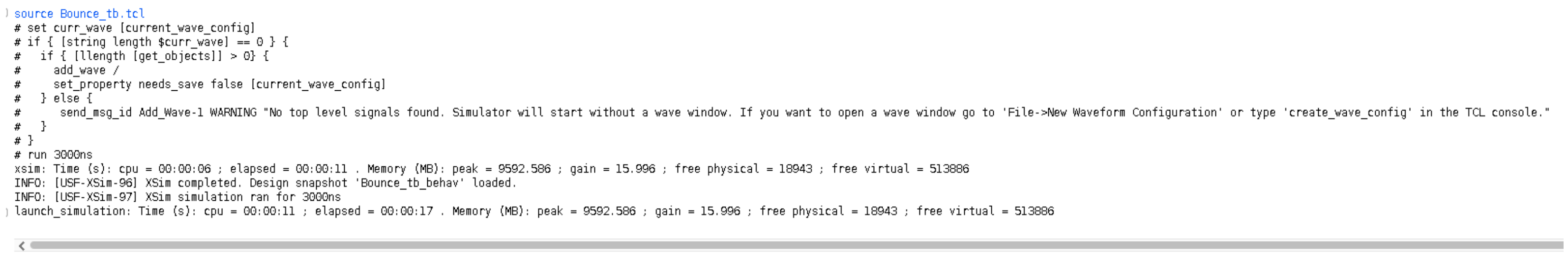
Top level .V



```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Company:
// Engineer:
//
// Create Date: 03/28/2025 04:41:50 PM
// Design Name:
// Module Name: top_level
// Project Name:
// Target Devices:
// Tool Versions:
// Description:
//
// Dependencies:
//
// Revision:
// Revision 0.01 - File Created
// Additional Comments:
//
//////////////////////////////////////////////////////////////////////////////////


module top_level(LEDs,SWs,BTNs,FastClk);
// the outputs and input connections
    output wire [3:0]LEDs;
    input wire [1:0] SWs,BTNs;
    input wire FastClk;
    //intermidiate nets
    wire [3:0] Clocks;
    reg SlowClk;
    //the combination logic
    always@(*)
        case(SWs) //Sws is 2-bit bus
        2'b00: SlowClk = Clocks[0];
        2'b01: SlowClk = Clocks[1];
        2'b10: SlowClk = Clocks[2];
        2'b11: SlowClk = Clocks[3];
        endcase
        //up counter
      up_counter UC0(LEDs[2:0],LEDs[3],BTNs[0],SlowClk,BTNs[1]);

      //The clock divider
      clock_divider clk_div0(
        .ClkOut(Clocks),
        .ClkIn(FastClk)
        );
endmodule
```

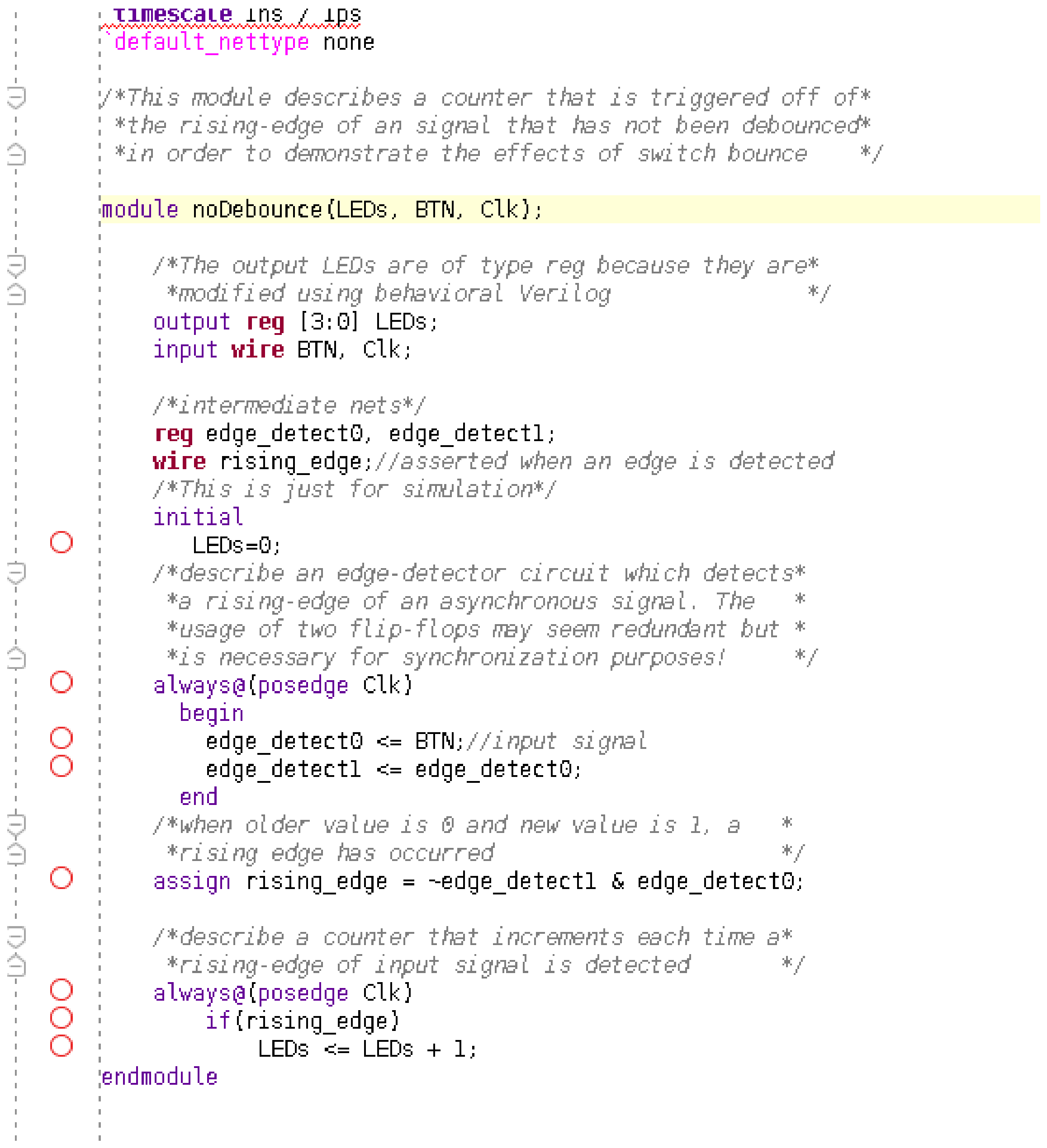
Experiment Part 3 noDebounce





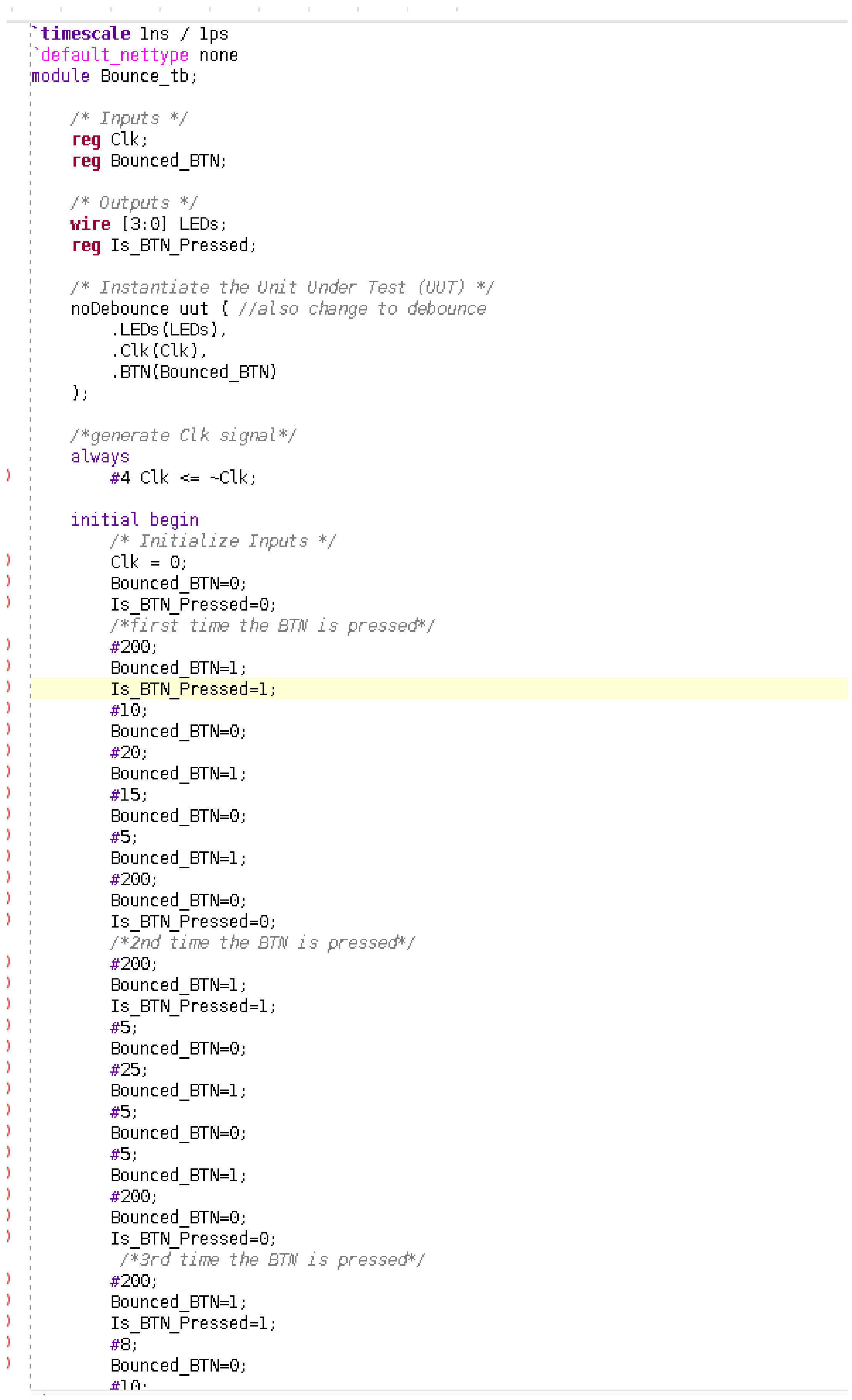
```
source Bounce_tb.tcl
# set curr_wave [current_wave_config]
# if { [string length $curr_wave] == 0 } {
#   if { [llength [get_objects]] > 0} {
#     add_wave /
#     set_property needs_save false [current_wave_config]
#   } else {
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave window go to 'File->New Waveform Configuration' or type 'create_wave_config' in the TCL console."
#   }
# }
# run 3000ns
xsim: Time (s): cpu = 00:00:06 ; elapsed = 00:00:11 . Memory (MB): peak = 9592.586 ; gain = 15.996 ; free physical = 18943 ; free virtual = 513886
INFO: [USF-XSim-96] XSim completed. Design snapshot 'Bounce_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 3000ns
launch_simulation: Time (s): cpu = 00:00:11 ; elapsed = 00:00:17 . Memory (MB): peak = 9592.586 ; gain = 15.996 ; free physical = 18943 ; free virtual = 513886
```

NoDebounce



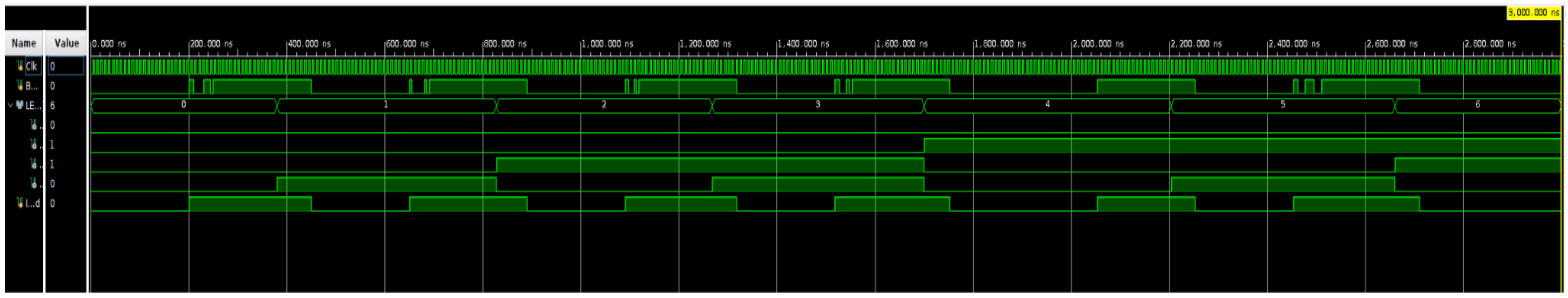
```
`timescale 1ns / 1ps
`default_nettype none

/*This module describes a counter that is triggered off of*
 *the rising-edge of an signal that has not been debounced*
 *in order to demonstrate the effects of switch bounce    */

module noDebounce(LEDs, BTN, Clk);

    /*The output LEDs are of type reg because they are*
     *modified using behavioral Verilog               */
    output reg [3:0] LEDs;
    input wire BTN, Clk;

    /*intermediate nets*/
    reg edge_detect0, edge_detect1;
    wire rising_edge;//asserted when an edge is detected
    /*This is just for simulation*/
    initial
       LEDs=0;
    /*describe an edge-detector circuit which detects*
     *a rising-edge of an asynchronous signal. The   *
     *usage of two flip-flops may seem redundant but *
     *is necessary for synchronization purposes!     */
    always@(posedge Clk)
      begin
        edge_detect0 <= BTN;//input signal
        edge_detect1 <= edge_detect0;
      end
    /*when older value is 0 and new value is 1, a    *
     *rising edge has occurred                       */
    assign rising_edge = ~edge_detect1 & edge_detect0;

    /*describe a counter that increments each time a*
     *rising-edge of input signal is detected       */
    always@(posedge Clk)
        if(rising_edge)
            LEDs <= LEDs + 1;
endmodule
```

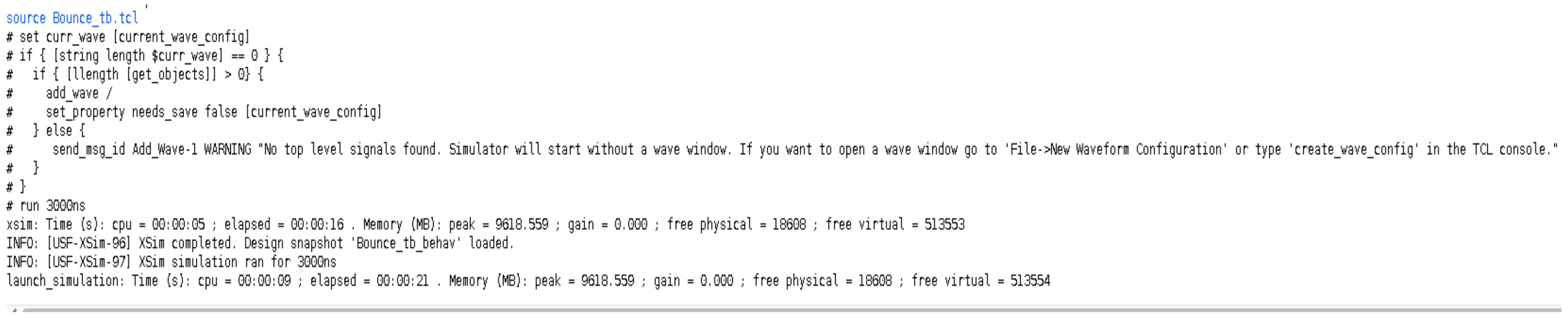
Bounce_Tb



```
`timescale 1ns / 1ps
`default_nettype none
module Bounce_tb;

    /* Inputs */
    reg Clk;
    reg Bounced_BTN;

    /* Outputs */
    wire [3:0] LEDs;
    reg Is_BTN_Pressed;

    /* Instantiate the Unit Under Test (UUT) */
    noDebounce uut ( //also change to debounce
        .LEDs(LEDs),
        .Clk(Clk),
        .BTN(Bounced_BTN)
    );

    /*generate Clk signal*/
    always
        #4 Clk <= ~Clk;

    initial begin
        /* Initialize Inputs */
        Clk = 0;
        Bounced_BTN=0;
        Is_BTN_Pressed=0;
        /*first time the BTN is pressed*/
        #200;
        Bounced_BTN=1;
        Is_BTN_Pressed=1;
        #10;
        Bounced_BTN=0;
        #20;
        Bounced_BTN=1;
        #15;
        Bounced_BTN=0;
        #5;
        Bounced_BTN=1;
        #200;
        Bounced_BTN=0;
        Is_BTN_Pressed=0;
        /*2nd time the BTN is pressed*/
        #200;
        Bounced_BTN=1;
        Is_BTN_Pressed=1;
        #5;
        Bounced_BTN=0;
        #25;
        Bounced_BTN=1;
        #5;
        Bounced_BTN=0;
        #5;
        Bounced_BTN=1;
        #200;
        Bounced_BTN=0;
        Is_BTN_Pressed=0;
         /*3rd time the BTN is pressed*/
        #200;
        Bounced_BTN=1;
        Is_BTN_Pressed=1;
        #8;
        Bounced_BTN=0;
        #10;
```

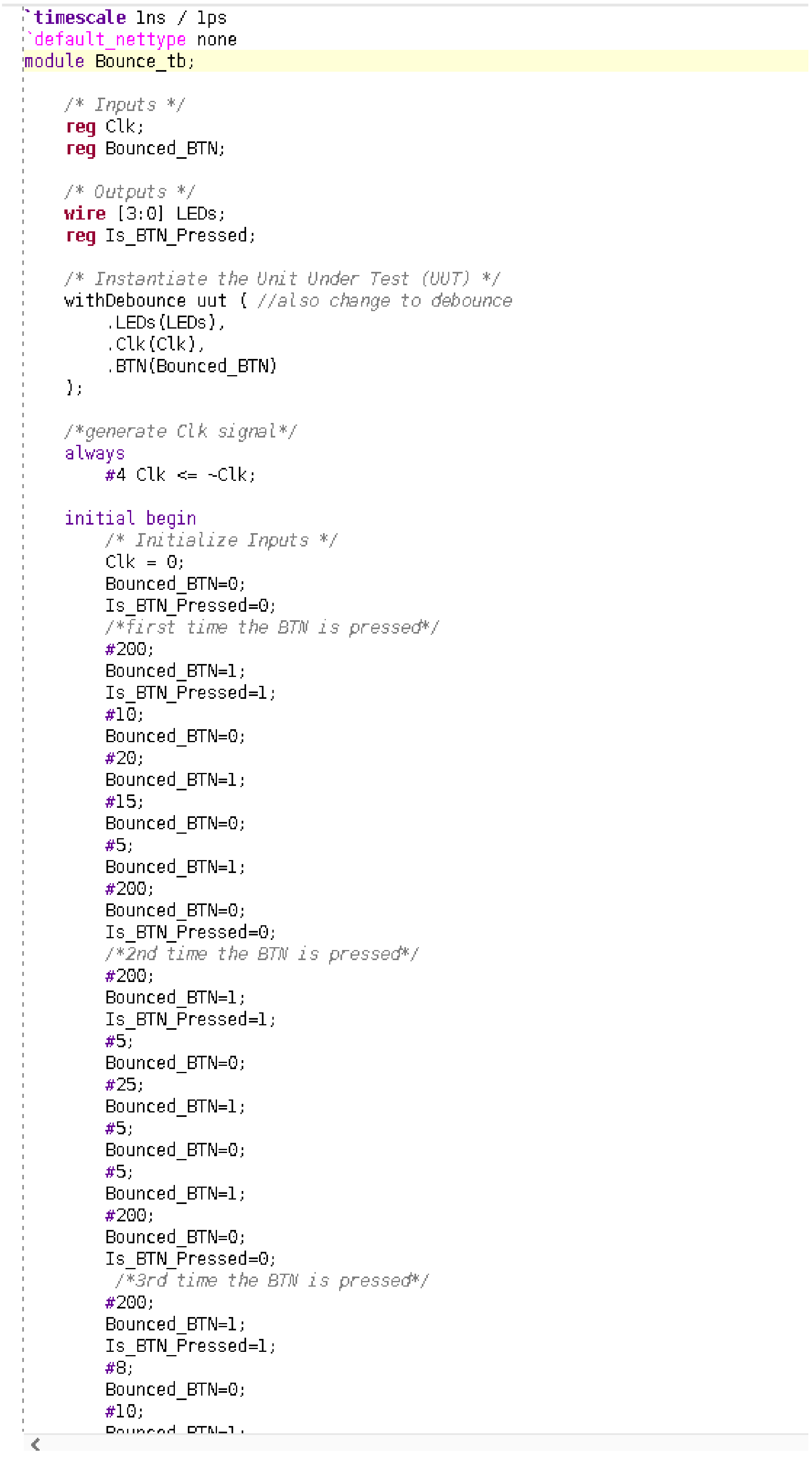
Experiment Part 3 withDebounce





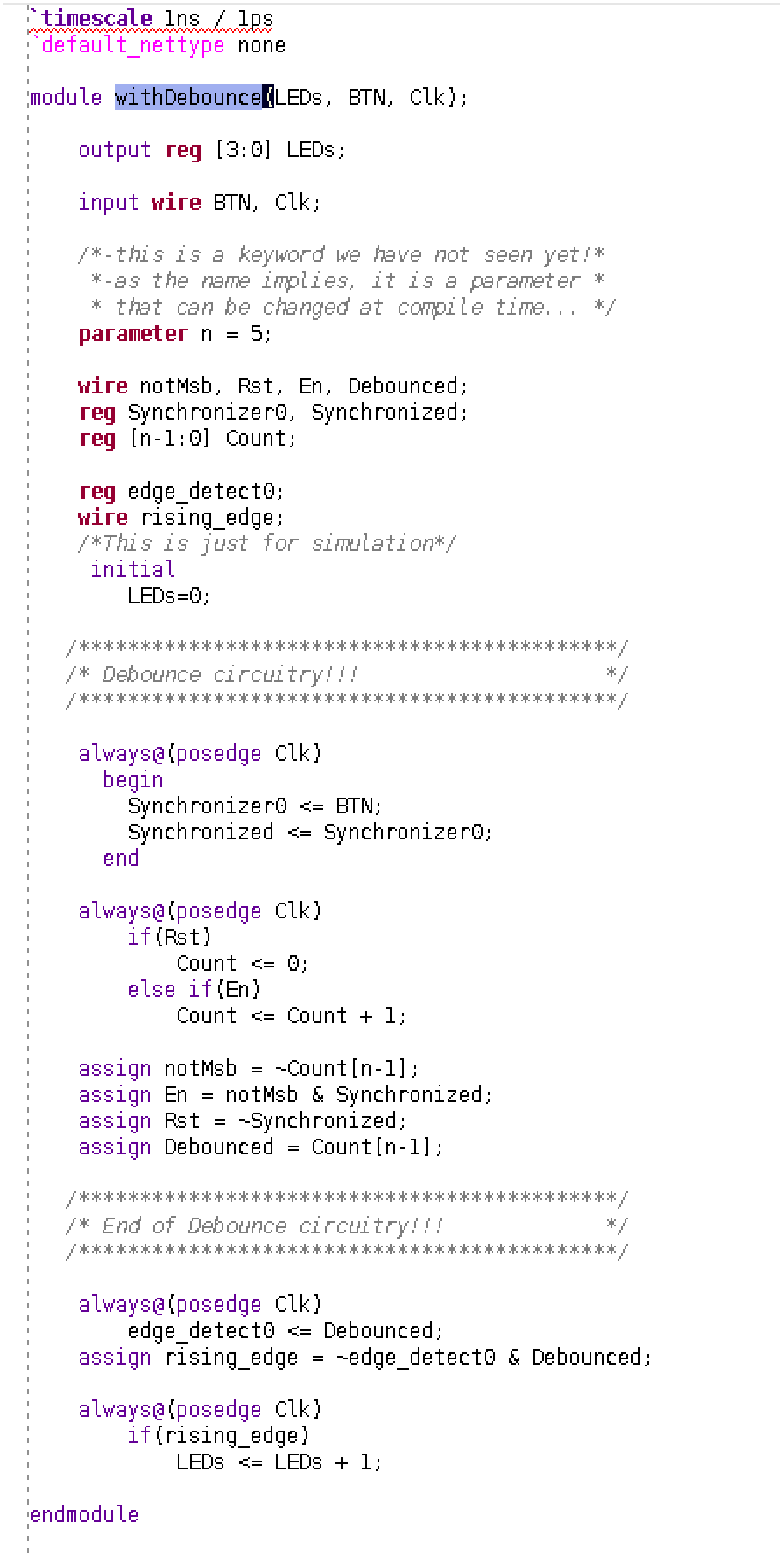
```
source Bounce_tb.tcl
# set curr_wave [current_wave_config]
# if { [string length $curr_wave] == 0 } {
#   if { [llength [get_objects]] > 0} {
#     add_wave /
#     set_property needs_save false [current_wave_config]
#   } else {
#      send_msg_id Add_Wave-1 WARNING "No top level signals found. Simulator will start without a wave window. If you want to open a wave window go to 'File->New Waveform Configuration' or type 'create_wave_config' in the TCL console."
#   }
# }
# run 3000ns
xsim: Time (s): cpu = 00:00:05 ; elapsed = 00:00:16 . Memory (MB): peak = 9618.559 ; gain = 0.000 ; free physical = 18608 ; free virtual = 513553
INFO: [USF-XSim-96] XSim completed. Design snapshot 'Bounce_tb_behav' loaded.
INFO: [USF-XSim-97] XSim simulation ran for 3000ns
launch_simulation: Time (s): cpu = 00:00:09 ; elapsed = 00:00:21 . Memory (MB): peak = 9618.559 ; gain = 0.000 ; free physical = 18608 ; free virtual = 513554
```

Bounce_tb



```
`timescale 1ns / 1ps
`default_nettype none
module Bounce_tb;

    /* Inputs */
    reg Clk;
    reg Bounced_BTN;

    /* Outputs */
    wire [3:0] LEDs;
    reg Is_BTN_Pressed;

    /* Instantiate the Unit Under Test (UUT) */
    withDebounce uut ( //also change to debounce
        .LEDs(LEDs),
        .Clk(Clk),
        .BTN(Bounced_BTN)
    );

    /*generate Clk signal*/
    always
        #4 Clk <= ~Clk;

    initial begin
        /* Initialize Inputs */
        Clk = 0;
        Bounced_BTN=0;
        Is_BTN_Pressed=0;
        /*first time the BTN is pressed*/
        #200;
        Bounced_BTN=1;
        Is_BTN_Pressed=1;
        #10;
        Bounced_BTN=0;
        #20;
        Bounced_BTN=1;
        #15;
        Bounced_BTN=0;
        #5;
        Bounced_BTN=1;
        #200;
        Bounced_BTN=0;
        Is_BTN_Pressed=0;
        /*2nd time the BTN is pressed*/
        #200;
        Bounced_BTN=1;
        Is_BTN_Pressed=1;
        #5;
        Bounced_BTN=0;
        #25;
        Bounced_BTN=1;
        #5;
        Bounced_BTN=0;
        #5;
        Bounced_BTN=1;
        #200;
        Bounced_BTN=0;
        Is_BTN_Pressed=0;
         /*3rd time the BTN is pressed*/
        #200;
        Bounced_BTN=1;
        Is_BTN_Pressed=1;
        #8;
        Bounced_BTN=0;
        #10;
        Bounced_BTN=1;
```

WithDebounce



```
`timescale 1ns / 1ps
`default_nettype none

module withDebounce(LEDs, BTN, Clk);

    output reg [3:0] LEDs;

    input wire BTN, Clk;

    /*-this is a keyword we have not seen yet!*
     *-as the name implies, it is a parameter *
     * that can be changed at compile time... */
    parameter n = 5;

    wire notMsb, Rst, En, Debounced;
    reg Synchronizer0, Synchronized;
    reg [n-1:0] Count;

    reg edge_detect0;
    wire rising_edge;
    /*This is just for simulation*/
     initial
        LEDs=0;

   /********************************************/
   /* Debounce circuitry!!!                    */
   /********************************************/

    always@(posedge Clk)
      begin
        Synchronizer0 <= BTN;
        Synchronized <= Synchronizer0;
      end

    always@(posedge Clk)
        if(Rst)
            Count <= 0;
        else if(En)
            Count <= Count + 1;

    assign notMsb = ~Count[n-1];
    assign En = notMsb & Synchronized;
    assign Rst = ~Synchronized;
    assign Debounced = Count[n-1];

   /********************************************/
   /* End of Debounce circuitry!!!             */
   /********************************************/

    always@(posedge Clk)
        edge_detect0 <= Debounced;
    assign rising_edge = ~edge_detect0 & Debounced;

    always@(posedge Clk)
        if(rising_edge)
            LEDs <= LEDs + 1;

endmodule
```

Conclusion:

Building on my previous experience with Verilog and learning new skills like operating an oscilloscope, I was able to successfully complete this lab. This process improved my ability to use an oscilloscope to evaluate design modules, such as the Verilog counter on the FPGA board. I also gained hands-on experience designing and implementing a debounce circuit to reduce electrical noise in the system. Additionally, I developed practical knowledge in mapping package pin numbers on the Zybo board to create an XDC file for the top-level design. A key takeaway from this lab was understanding the importance and functionality of a debounce circuit.

1. Measure the periods of COUNT2, COUNT3, COUNT4 and COUNT5

Count 2: 62ns

Count 3: 36 ns

Count 4: 256 ns

Count 5: 128 ns

2 . You should see that the test bench produces a Clk signal. What is the frequency of that signal?

F= 1/Tc so therefore freq= 50Mhz

3. You should also see that the test bench holds the counter in reset for a specific interval of time. How long is that interval?

The interval was 20 ns

4. After reset is de-asserted, the test bench holds the enable LOW for some amount of time before allowing the counter to run. How long is this time period?

The period was 20 ns

5. Finally, you should notice that the counter will roll over after reaching a maximum value. What is this maximum count value and what signal in the waveform could we use to know exactly when the counter is going to roll over?

The maximum count found is 111 and then rolls over. This can be due to the fact that the counter is 3 bitts

6. If we use a 125MHz clock to drive our frequency divider, what rate will the most significant bit of the divider oscillate at?

The Frequency (125 Mhz) / 2^26 = 1.862

7. How do the switches affect the LED Display? What does button 1 do? BTN1 serves as the reset button.

When Both switches are off (00), the output speed is too fast to observe. With SW0 on (01), the output is faster compared to SW1 on (10). The slowest output occurs when both SW0 and SW1 are on (11).

8.Does the counter in withDebounce.v work as expected

Yes it does the output changes when the button is pressed

10. Explain the operation of the circuit described in withDebounce.v?

The withDebounce circuit features a synchronizer that resets the counter when the signal is low and allows it to run when the signal is high.

Student Feedback

1.What did you like most about the lab assignment and why? What did you like least about it and why?

I think overall I don't have a part where I disliked the lab. I just wished there would be more details when it came to the xdc files and how to properly write one. Overall I really did enjoy the lab and working with verilog.

2. Were there any sections of the lab manual that were unclear? If so, what was unclear? Do you have Any suggestions for improving the clarity?

No overall everything was clear and straightforward but I would suggest adding more detail towards certacodes like the xdc files.

3. What suggestions do you have to improve the overall lab assignment

Overall again i fairly thought that this lab was straightforward i just think that some details should be added towards the part of the xdc files