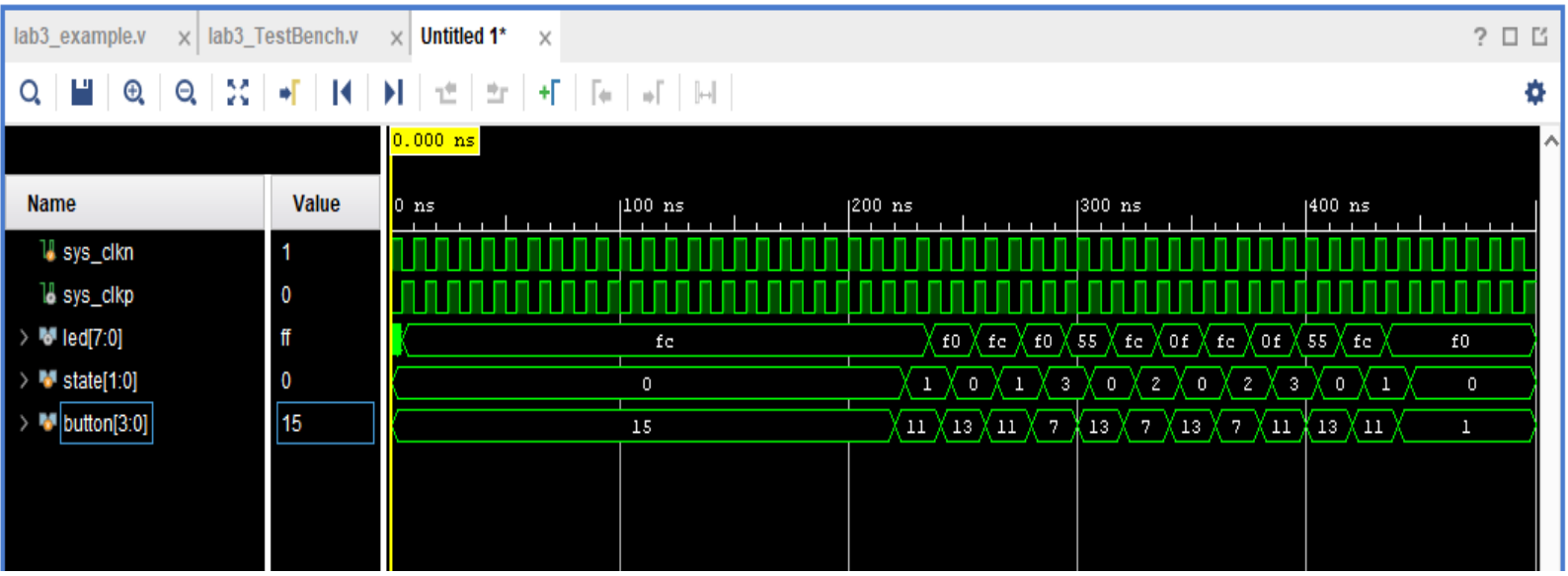
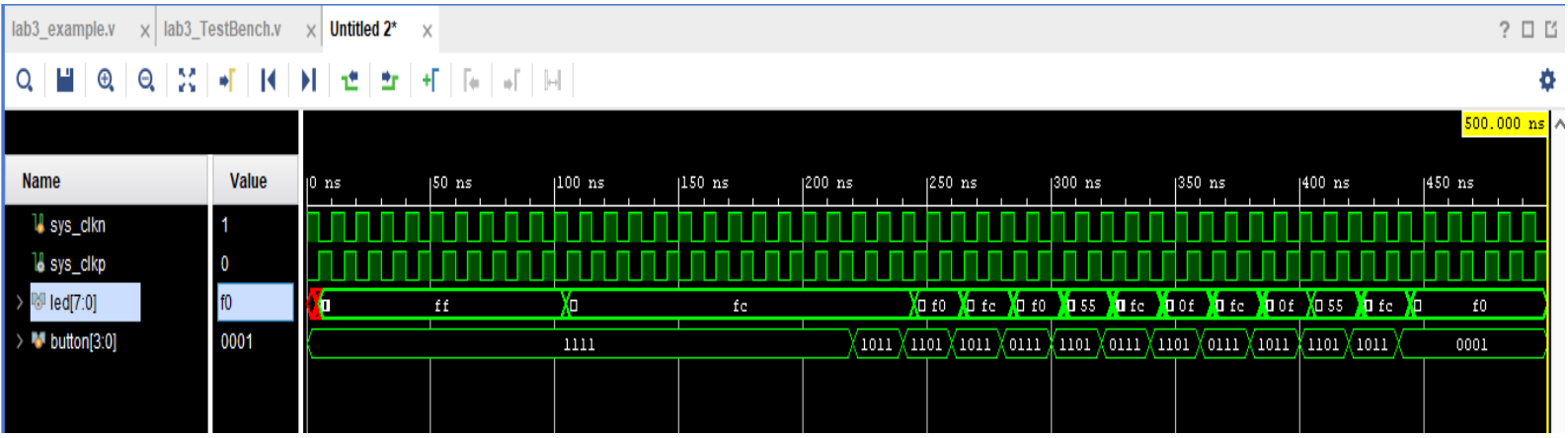
Name: Junzhe Wu Net id: Junzhew3

1. The Behavior Simulation results of example code



2. The Post-Implementation Timing Simulation results of example code



Three differences:

(1) There are some erroneous states that the LEDs transition through at beginning in the Post-Implementation Timing Simulation.

(2) The value of led variable is **ff** and then turn to **fc** at beginning in the Post-Implementation Timing Simulation. But the value of led is always **fc** at beginning in Behavior Simulation.

(3) The time of state change is different. When **button** variable change from 1111 to 1011, the led **variable** needs more time to change from fc to f0 in the Post-Implementation Timing Simulation.

Reasons:

Behavioral simulation allows you to verify syntax and functionality without timing information. So it does not contain specifics on how the design will be implemented. But Post-Implementation Timing Simulation take these behavioral designs into consideration and infer the actual gate structures and connections to be used, generating a netlist description and more detailed verification data.

In reality, when the power is on, the board needs a really short time to react and get ready to work. That is why there are some erroneous states that the LEDs transition through and the value of led

variable is **ff** at beginning in the Post-Implementation Timing Simulation. In this time, the value of led variable is **ff** which means that all led lights are off.

Finally, since Post-Implementation Timing Simulation take behavioral designs into consideration and infer the actual gate structures and connections to be used, the variable inside it needs more time to make the state change.

3. The time of state change and erroneous states in the Post-Implementation Timing Simulation

   The time of state change is 24.861ns.

   Reason of these erroneous states:

   In reality, when the power is on, the board needs a really short time to react and get ready to work. That is why there are some erroneous states that the LEDs transition through.

4. The time of state change in the Behavior Simulation

   The time of state change is 15ns.

   Reason of different time of state change in two kinds of simulations:

   Since Post-Implementation Timing Simulation take behavioral designs into consideration and infer the actual gate structures and connections to be used, the variable inside it needs more time to make the state change.

5. Verilog code:

```
`timescale 1ns / 1ps
module lab3_example(
    input [3:0] button,
    output [7:0] led,
    input sys_clkn,
    input sys_clkp
    );

    reg [2:0] state = 0;
    reg [7:0] led_register = 0;
    reg [3:0] button_reg;
    reg [28:0] counter=0;
    reg[1:0] h=0;
    reg[1:0] preG=0;
```

```
wire clk;
IBUFGDS osc_clk(
    .O(clk),
    .I(sys_clkp),
    .IB(sys_clkn)
);

assign led = ~led_register; //map led wire to led_register
localparam STATE_G1R2R3     = 3'd0;
localparam STATE_Y1R2R3     = 3'd1;
localparam STATE_R1G2R3     = 3'd2;
localparam STATE_R1Y2R3     = 3'd3;
localparam STATE_R1R2G3     = 3'd4;

always @(posedge clk)
begin
    button_reg = ~button;
    begin
        case (state)
            STATE_G1R2R3 : begin
                counter<=counter+1;
                if (counter == 10000) begin
                    counter<=0;
                    state<= STATE_Y1R2R3;
                end
                led_register <= 8'b10000101;
                if (button_reg == 4'b1000)  h<=1;
            end

            STATE_Y1R2R3 : begin
                counter<=counter+1;
                if (counter == 5000) begin
                    counter<=0;
                    if (h ==1 ) begin
                        state <= STATE_R1R2G3;
                        preG=0;
                    end
                    else state<= STATE_R1G2R3;
                end
                led_register <= 8'b01000101;
                if (button_reg == 4'b1000)  h<=1;
            end
```

```
            STATE_R1G2R3 : begin
                counter<=counter+1;
                if (counter == 10000) begin
                    counter<=0;
                    state<= STATE_R1Y2R3;
                end
                led_register <= 8'b00110001;
                if (button_reg == 4'b1000)  h<=1;
            end

            STATE_R1Y2R3 : begin
                counter<=counter+1;
                if (counter == 5000) begin
                    counter<=0;
                    if (h==1) begin
                        state <= STATE_R1R2G3;
                        preG=1;
                    end
                    else state<= STATE_G1R2R3;
                end
                led_register <= 8'b00101001;
                if (button_reg == 4'b1000)  h<=1;
            end

            STATE_R1R2G3 : begin
                counter<=counter+1;
                if (counter == 10000) begin
                    counter<=0;
                    h=0;
                    if (preG==0)
                        state<= STATE_R1G2R3;
                    else
                        state <= STATE_G1R2R3;
                end
                led_register <= 8'b00100110;
            end

            default: state <= STATE_G1R2R3;

        endcase
    end
  end
endmodule
```

6. test bench code:

```
`timescale 1ns / 1ps

module lab3_TestBench();
    //Declare wires and registers that will interface with the module
under test
    //Registers are initilized to known states. Wires cannot be
initilized.
    reg sys_clkn=1;
    wire sys_clkp;
    wire [7:0] led;
    reg [3:0] button;

    //Invoke the module that we like to test
    lab3_example ModuleUnderTest
(.button(button),.led(led),.sys_clkn(sys_clkn),.sys_clkp(sys_clkp));

    // Generate a clock signal. The clock will change its state every
5ns.
    //Remember that the test module takes sys_clkp and sys_clkn as input
clock signals.
    //From these two signals a clock signal, clk, is derived.
    //The LVDS clock signal, sys_clkn, is always in the opposite state
than sys_clkp.
    assign sys_clkp = ~sys_clkn;
    always begin
        #5 sys_clkn = ~sys_clkn;
    end

    initial begin
            #0 button <= 4'b1111;
            #25000 button <= 4'b0111;
            #50 button <= 4'b1111;

    end

endmodule
```