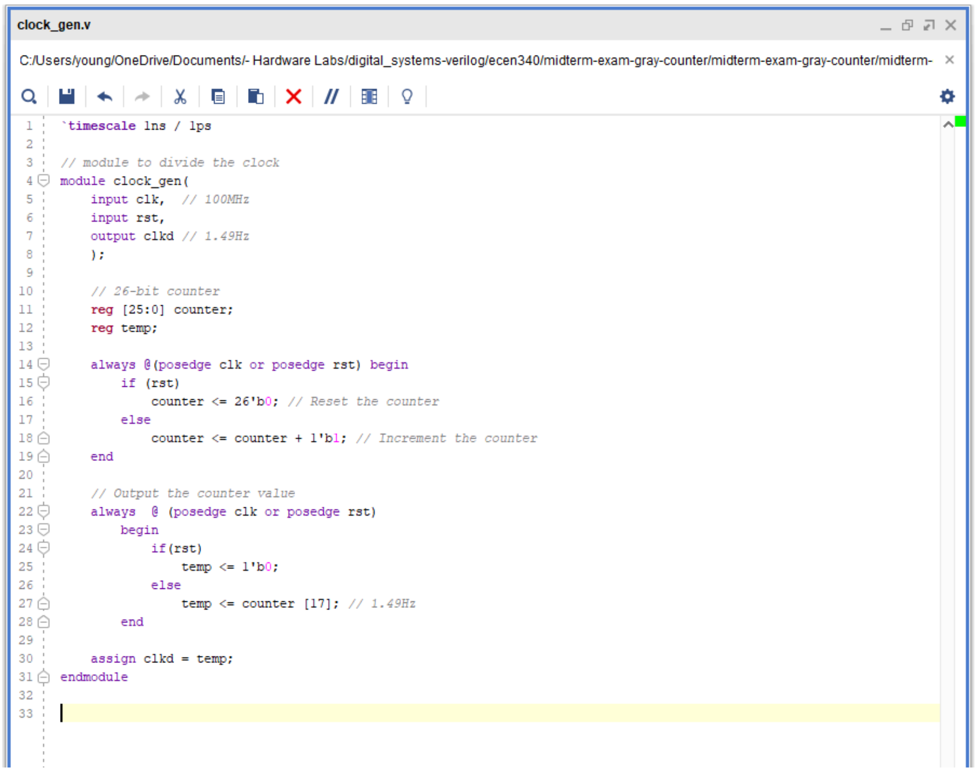
Brodric Young

ECEN 340

Midterm Lab

Module Code:

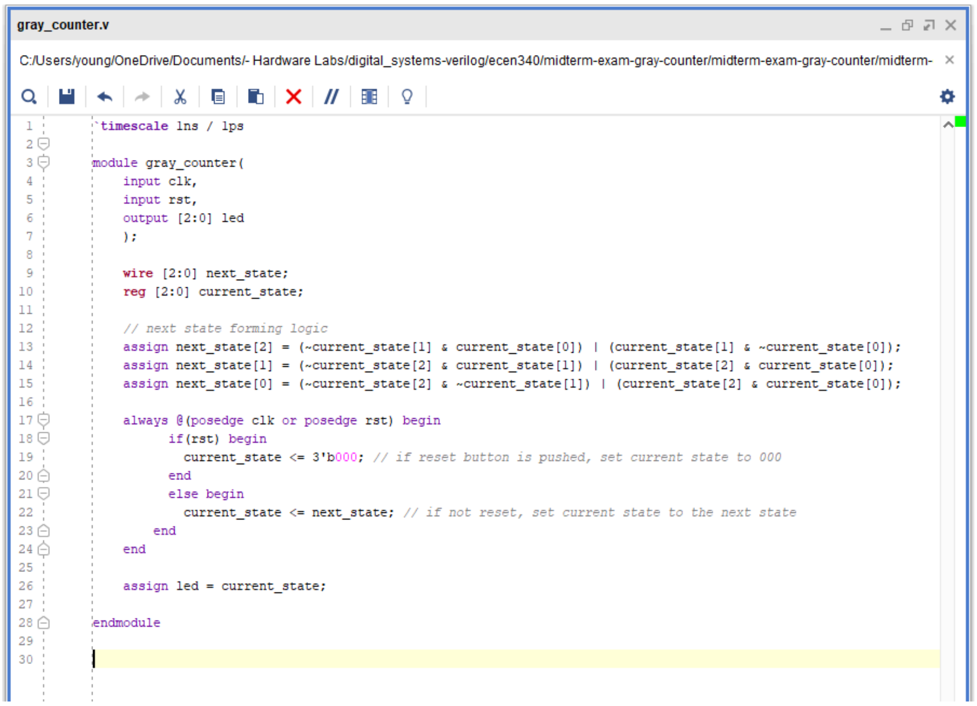


clock_gen.v

C:/Users/young/OneDrive/Documents/- Hardware Labs/digital_systems-verilog/ecen340/midterm-exam-gray-counter/midterm-exam-gray-counter/midterm-

```verilog
`timescale 1ns / 1ps

// module to divide the clock
module clock_gen(
    input clk,  // 100MHz
    input rst,
    output clkd // 1.49Hz
    );

    // 26-bit counter
    reg [25:0] counter;
    reg temp;

    always @(posedge clk or posedge rst) begin
        if (rst)
            counter <= 26'b0; // Reset the counter
        else
            counter <= counter + 1'b1; // Increment the counter
    end

    // Output the counter value
    always  @ (posedge clk or posedge rst)
        begin
            if(rst)
                temp <= 1'b0;
            else
                temp <= counter [17]; // 1.49Hz
        end

    assign clkd = temp;
endmodule

```

Figure 1: clock_gen module

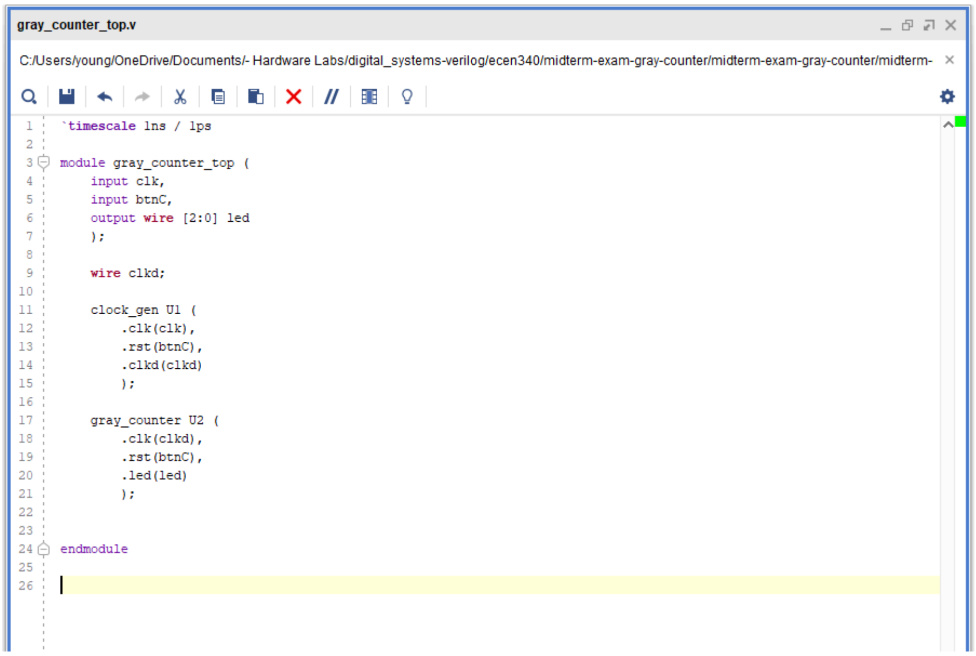


gray_counter.v

C:/Users/young/OneDrive/Documents/- Hardware Labs/digital_systems-verilog/ecen340/midterm-exam-gray-counter/midterm-exam-gray-counter/midterm-

```verilog
`timescale 1ns / 1ps

module gray_counter(
    input clk,
    input rst,
    output [2:0] led
    );

    wire [2:0] next_state;
    reg [2:0] current_state;

    // next state forming logic
    assign next_state[2] = (~current_state[1] & current_state[0]) | (current_state[1] & ~current_state[0]);
    assign next_state[1] = (~current_state[2] & current_state[1]) | (current_state[2] & current_state[0]);
    assign next_state[0] = (~current_state[2] & ~current_state[1]) | (current_state[2] & current_state[0]);

    always @(posedge clk or posedge rst) begin
          if(rst) begin
            current_state <= 3'b000; // if reset button is pushed, set current state to 000
          end
          else begin
            current_state <= next_state; // if not reset, set current state to the next state
        end
    end

    assign led = current_state;

endmodule

```

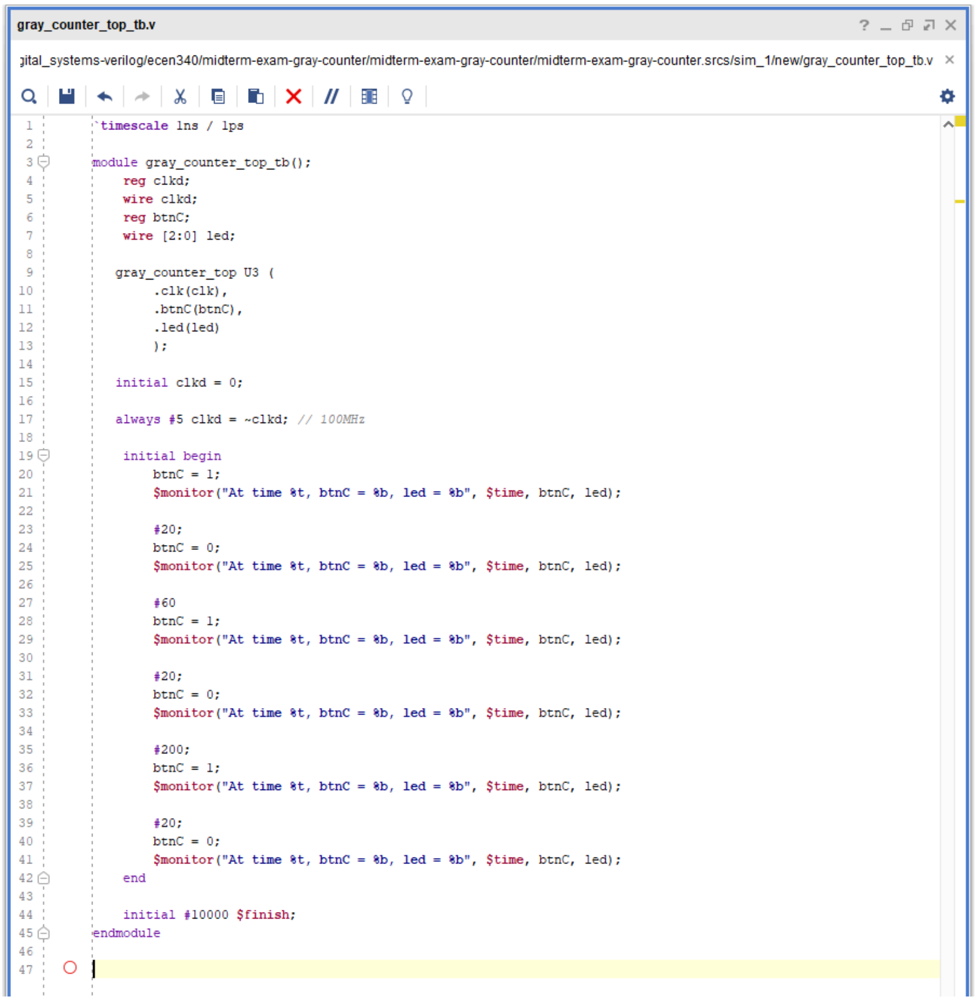
Figure 2: gray_counter module


gray_counter_top.v

C:/Users/young/OneDrive/Documents/- Hardware Labs/digital_systems-verilog/ecen340/midterm-exam-gray-counter/midterm-exam-gray-counter/midterm-

```
`timescale 1ns / 1ps

module gray_counter_top (
    input clk,
    input btnC,
    output wire [2:0] led
    );

    wire clkd;

    clock_gen U1 (
        .clk(clk),
        .rst(btnC),
        .clkd(clkd)
        );

    gray_counter U2 (
        .clk(clkd),
        .rst(btnC),
        .led(led)
        );


endmodule

```

Figure 3: gray_counter_top module



gray_counter_top_tb.v

jital_systems-verilog/ecen340/midterm-exam-gray-counter/midterm-exam-gray-counter/midterm-exam-gray-counter.srcs/sim_1/new/gray_counter_top_tb.v

```
`timescale 1ns / 1ps

module gray_counter_top_tb();
    reg clkd;
    wire clkd;
    reg btnC;
    wire [2:0] led;

   gray_counter_top U3 (
        .clk(clk),
        .btnC(btnC),
        .led(led)
        );

   initial clkd = 0;

   always #5 clkd = ~clkd; // 100MHz

    initial begin
        btnC = 1;
        $monitor("At time %t, btnC = %b, led = %b", $time, btnC, led);

        #20;
        btnC = 0;
        $monitor("At time %t, btnC = %b, led = %b", $time, btnC, led);

        #60
        btnC = 1;
        $monitor("At time %t, btnC = %b, led = %b", $time, btnC, led);

        #20;
        btnC = 0;
        $monitor("At time %t, btnC = %b, led = %b", $time, btnC, led);

        #200;
        btnC = 1;
        $monitor("At time %t, btnC = %b, led = %b", $time, btnC, led);

        #20;
        btnC = 0;
        $monitor("At time %t, btnC = %b, led = %b", $time, btnC, led);
    end

    initial #10000 $finish;
endmodule

```

Figure 4: test bench module

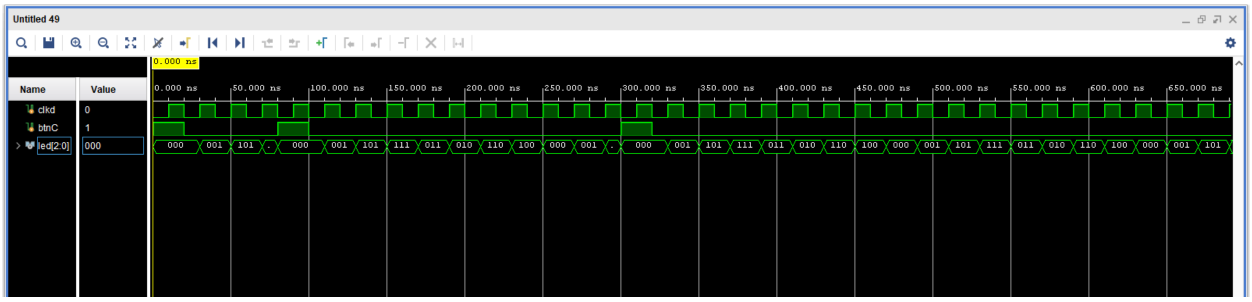


Figure 5: test bench simulation

<u>Functionality:</u>

In the end the gray counter functioned 100% correctly. It took some time and effort to fix all the bugs in my code but it eventually functioned properly. I tested it in the simulation and also verified the functionality on the Basys3 board to make sure it was correct.

<u>Simpler/alternative Method:</u>

Another method of implementing this gray counter could have been by using a case statement for the next state forming logic. This behavioral style would be simpler than the data flow way of using the ands, nots, and ors it took to implement the next state forming logic from boolean equations.