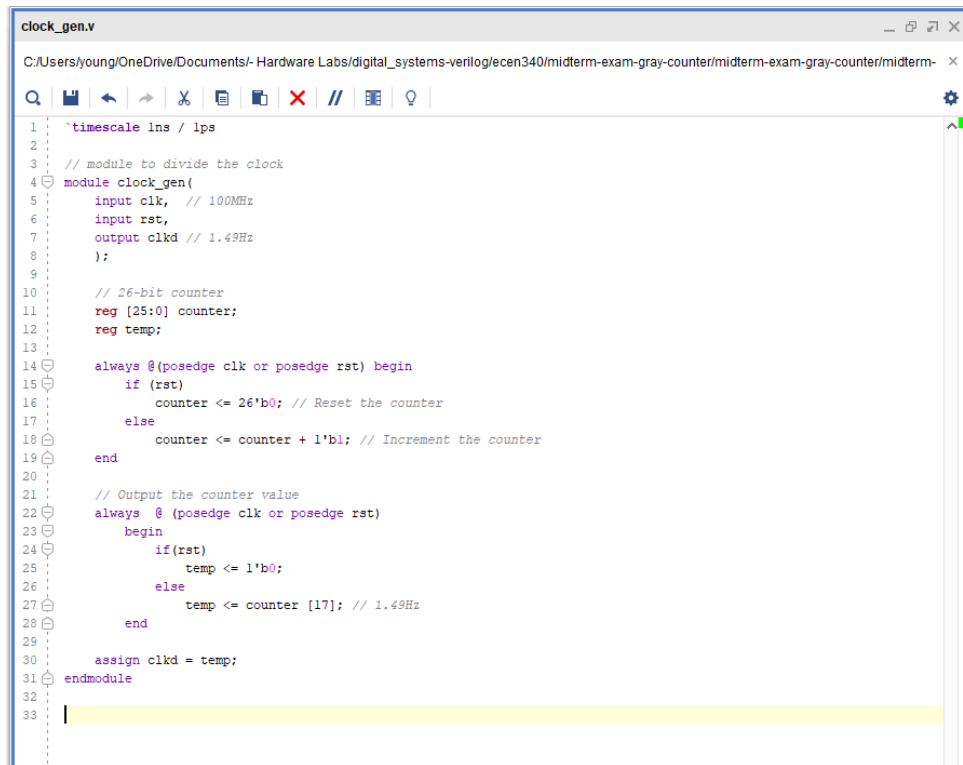


Brodric Young

ECEN 340

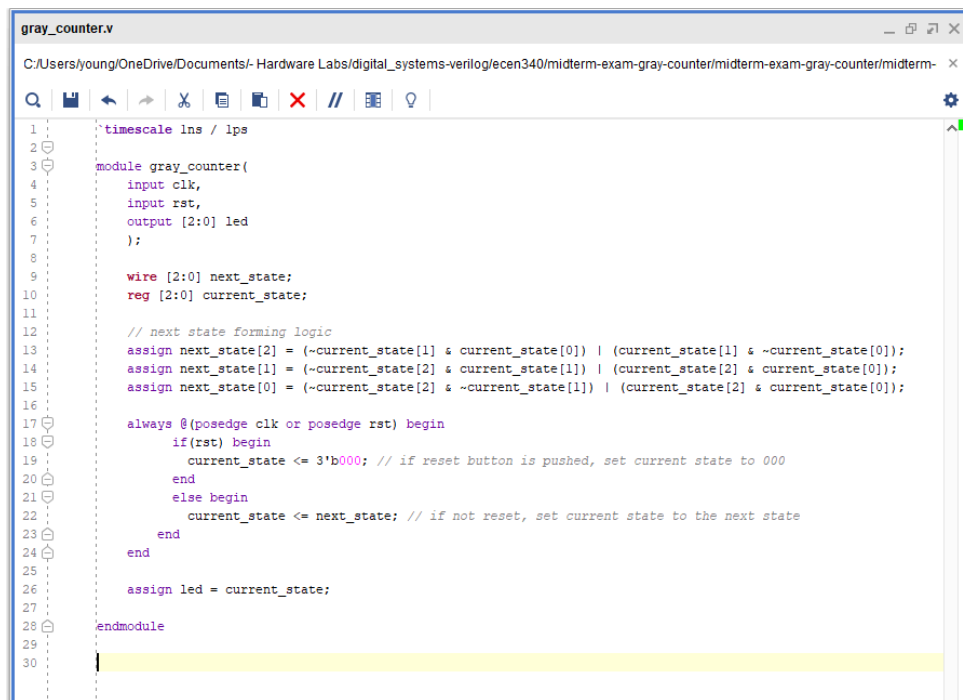
Midterm Lab

Module Code:



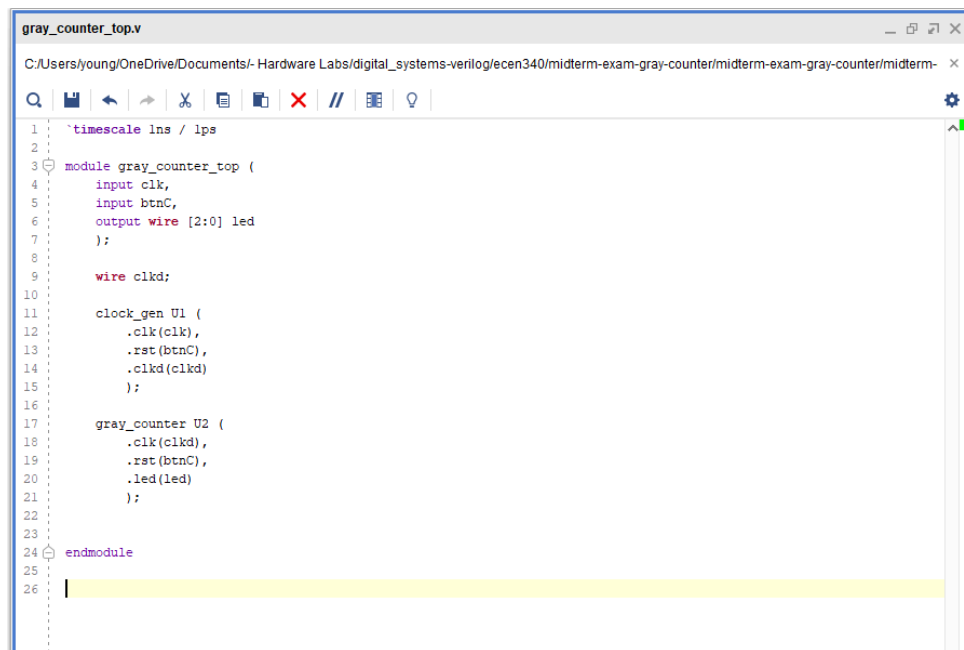
```
1  `timescale 1ns / 1ps
2
3  // module to divide the clock
4  module clock_gen(
5      input clk, // 100MHz
6      input rst,
7      output clkd // 1.49Hz
8  );
9
10     // 26-bit counter
11     reg [25:0] counter;
12     reg temp;
13
14     always @(posedge clk or posedge rst) begin
15         if (rst)
16             counter <= 26'b0; // Reset the counter
17         else
18             counter <= counter + 1'b1; // Increment the counter
19     end
20
21     // Output the counter value
22     always @(posedge clk or posedge rst)
23     begin
24         if (rst)
25             temp <= 1'b0;
26         else
27             temp <= counter [17]; // 1.49Hz
28     end
29
30     assign clkd = temp;
31 endmodule
```

Figure 1: clock_gen module



```
1  `timescale 1ns / 1ps
2
3  module gray_counter(
4      input clk,
5      input rst,
6      output [2:0] led
7  );
8
9      wire [2:0] next_state;
10     reg [2:0] current_state;
11
12     // next state forming logic
13     assign next_state[2] = (~current_state[1] & current_state[0]) | (current_state[1] & ~current_state[0]);
14     assign next_state[1] = (~current_state[2] & current_state[1]) | (current_state[2] & current_state[0]);
15     assign next_state[0] = (~current_state[2] & ~current_state[1]) | (current_state[2] & current_state[0]);
16
17     always @(posedge clk or posedge rst) begin
18         if (rst) begin
19             current_state <= 3'b000; // if reset button is pushed, set current state to 000
20         end
21         else begin
22             current_state <= next_state; // if not reset, set current state to the next state
23         end
24     end
25
26     assign led = current_state;
27
28 endmodule
```

Figure 2: gray_counter module



The image shows a screenshot of a Verilog code editor window titled "gray_counter_top.v". The window's address bar shows the file path: "C:/Users/young/OneDrive/Documents/- Hardware Labs/digital_systems-verilog/ecen340/midterm-exam-gray-counter/midterm-exam-gray-counter/midterm-". The editor contains the following Verilog code:

```
1  `timescale 1ns / 1ps
2
3  module gray_counter_top (
4      input clk,
5      input btnC,
6      output wire [2:0] led
7  );
8
9      wire clkd;
10
11      clock_gen U1 (
12          .clk(clk),
13          .rst(btnC),
14          .clkd(clkd)
15      );
16
17      gray_counter U2 (
18          .clk(clkd),
19          .rst(btnC),
20          .led(led)
21      );
22
23
24  endmodule
25
26
```

The code defines a module named "gray_counter_top" with three inputs: "clk", "btnC", and "led" (which is a 3-bit bus, indicated by "[2:0]"). The module contains two sub-modules: "clock_gen U1" and "gray_counter U2". "clock_gen U1" takes "clk" and "btnC" as inputs and outputs "clkd". "gray_counter U2" takes "clkd" and "btnC" as inputs and outputs "led". The code is formatted with line numbers on the left and a search bar on the right.

Figure 3: gray_counter_top module

```

1 timescale 1ns / 1ps
2
3 module gray_counter_top_tb();
4     reg clkd;
5     wire clkd;
6     reg btnC;
7     wire [2:0] led;
8
9     gray_counter_top U3 (
10         .clk(clkd),
11         .btnC(btnC),
12         .led(led)
13     );
14
15     initial clkd = 0;
16
17     always #5 clkd = ~clkd; // 100MHz
18
19     initial begin
20         btnC = 1;
21         $monitor("At time %t, btnC = %b, led = %b", $time, btnC, led);
22
23         #20;
24         btnC = 0;
25         $monitor("At time %t, btnC = %b, led = %b", $time, btnC, led);
26
27         #60;
28         btnC = 1;
29         $monitor("At time %t, btnC = %b, led = %b", $time, btnC, led);
30
31         #20;
32         btnC = 0;
33         $monitor("At time %t, btnC = %b, led = %b", $time, btnC, led);
34
35         #200;
36         btnC = 1;
37         $monitor("At time %t, btnC = %b, led = %b", $time, btnC, led);
38
39         #20;
40         btnC = 0;
41         $monitor("At time %t, btnC = %b, led = %b", $time, btnC, led);
42     end
43
44     initial #10000 $finish;
45 endmodule
46
47

```

Figure 4: test bench module

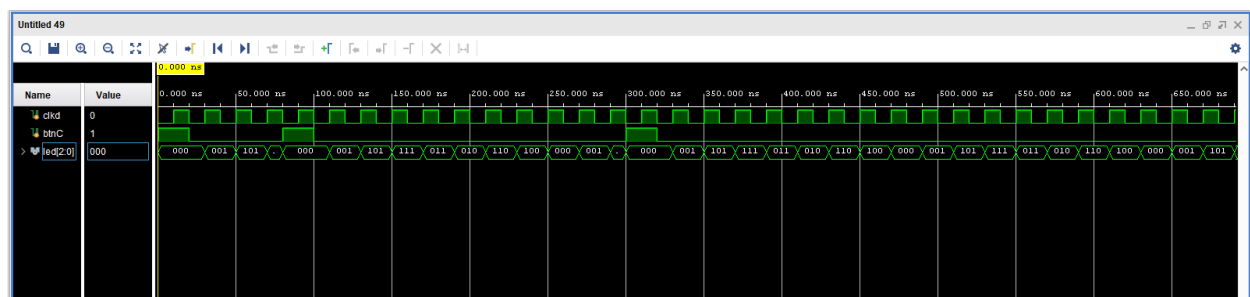


Figure 5: test bench simulation

Functionality:

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.

Simpler/alternative Method:

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.