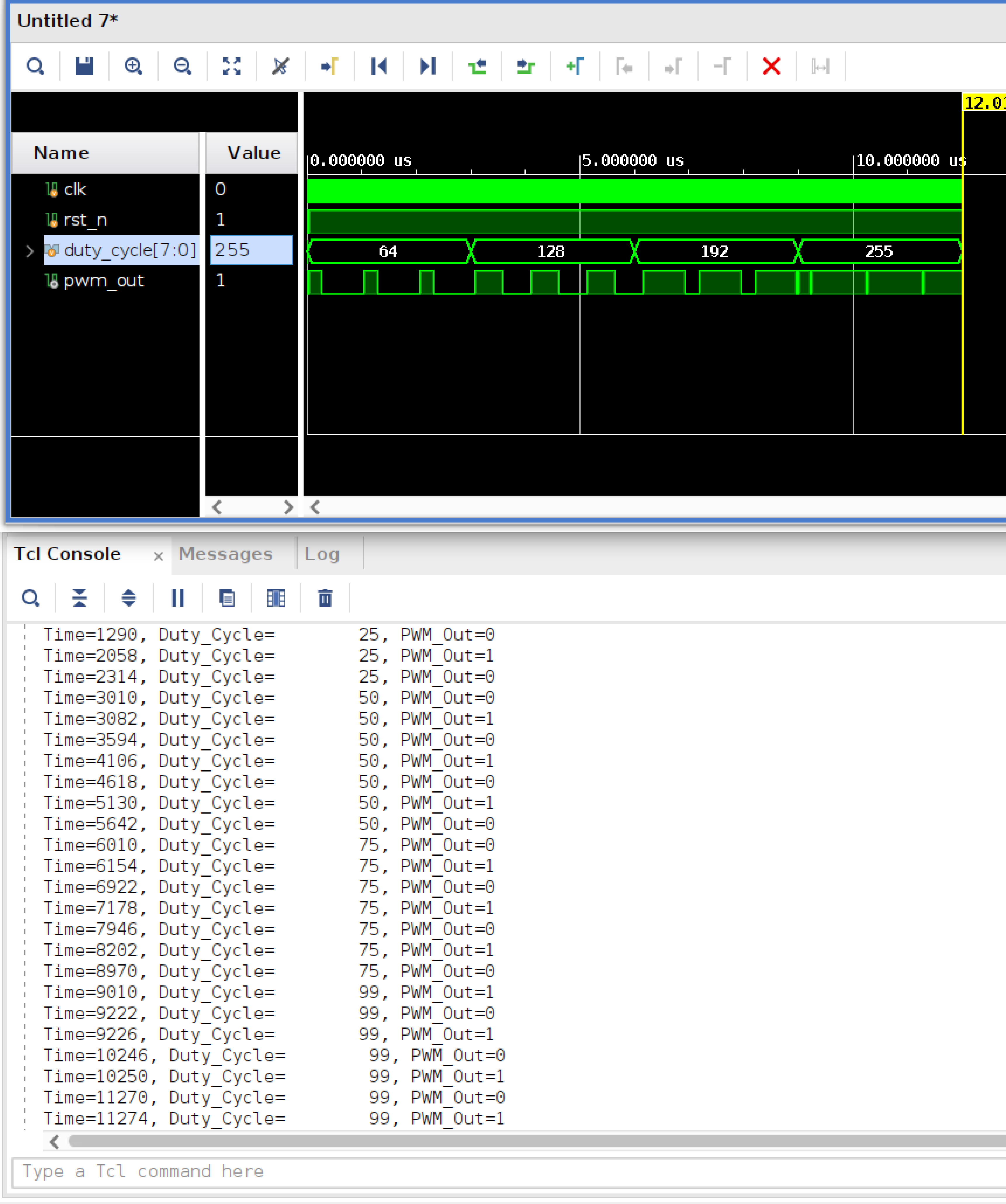
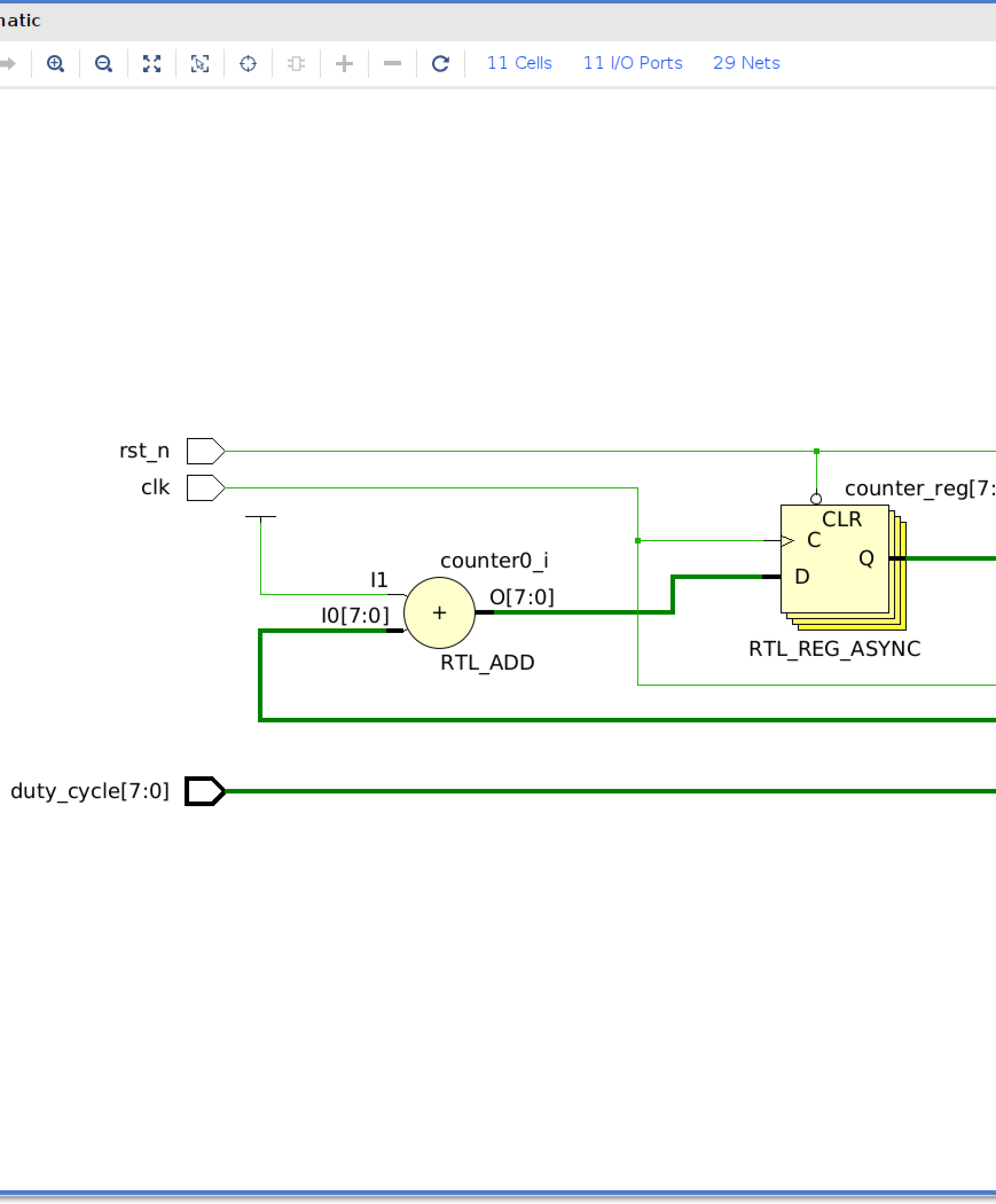
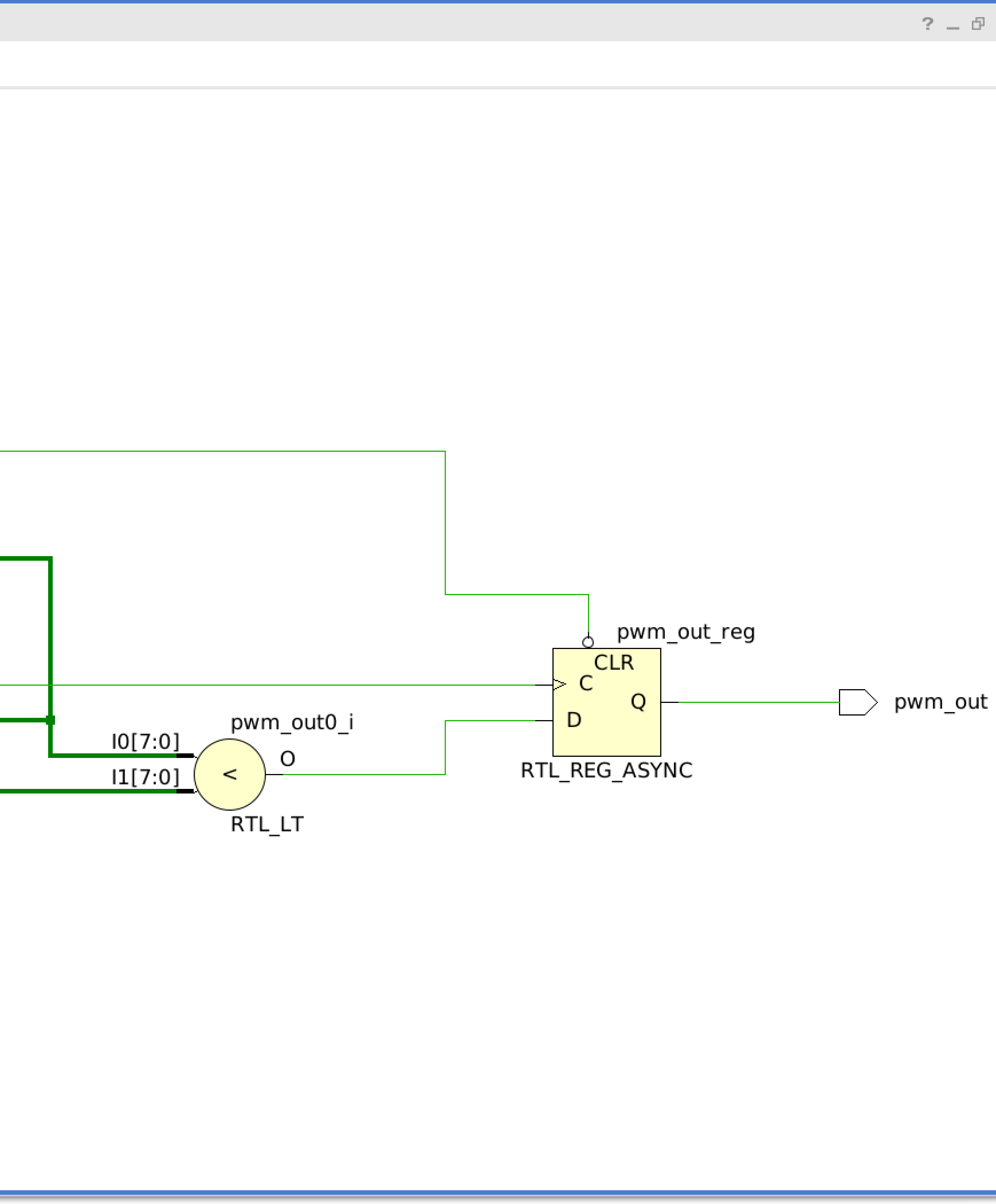
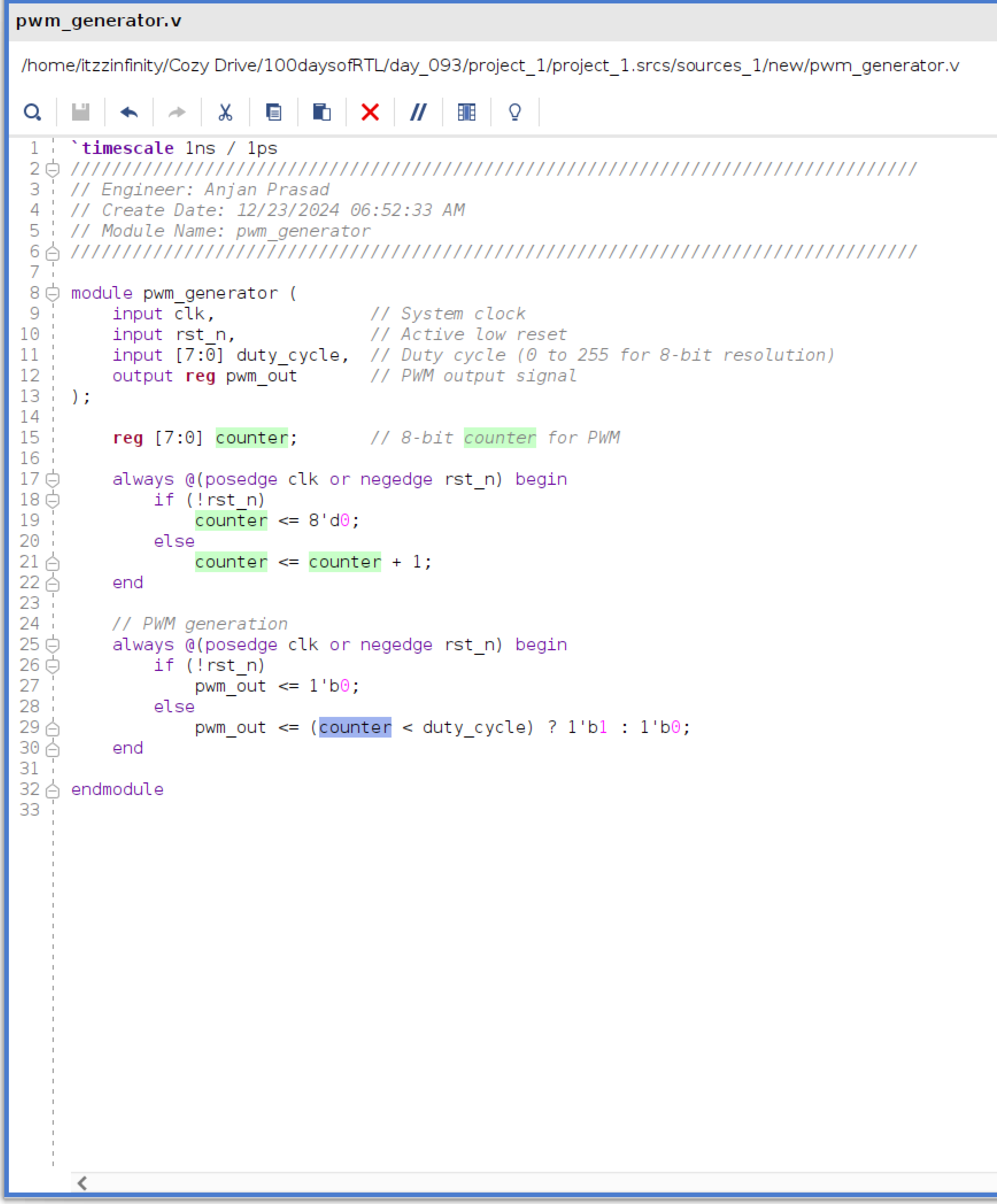


```
Time=1290, Duty_Cycle=        25, PWM_Out=0
Time=2058, Duty_Cycle=        25, PWM_Out=1
Time=2314, Duty_Cycle=        25, PWM_Out=0
Time=3010, Duty_Cycle=        50, PWM_Out=0
Time=3082, Duty_Cycle=        50, PWM_Out=1
Time=3594, Duty_Cycle=        50, PWM_Out=0
Time=4106, Duty_Cycle=        50, PWM_Out=1
Time=4618, Duty_Cycle=        50, PWM_Out=0
Time=5130, Duty_Cycle=        50, PWM_Out=1
Time=5642, Duty_Cycle=        50, PWM_Out=0
Time=6010, Duty_Cycle=        75, PWM_Out=0
Time=6154, Duty_Cycle=        75, PWM_Out=1
Time=6922, Duty_Cycle=        75, PWM_Out=0
Time=7178, Duty_Cycle=        75, PWM_Out=1
Time=7946, Duty_Cycle=        75, PWM_Out=0
Time=8202, Duty_Cycle=        75, PWM_Out=1
Time=8970, Duty_Cycle=        75, PWM_Out=0
Time=9010, Duty_Cycle=        99, PWM_Out=1
Time=9222, Duty_Cycle=        99, PWM_Out=0
Time=9226, Duty_Cycle=        99, PWM_Out=1
Time=10246, Duty_Cycle=        99, PWM_Out=0
Time=10250, Duty_Cycle=        99, PWM_Out=1
Time=11270, Duty_Cycle=        99, PWM_Out=0
Time=11274, Duty_Cycle=        99, PWM_Out=1
```

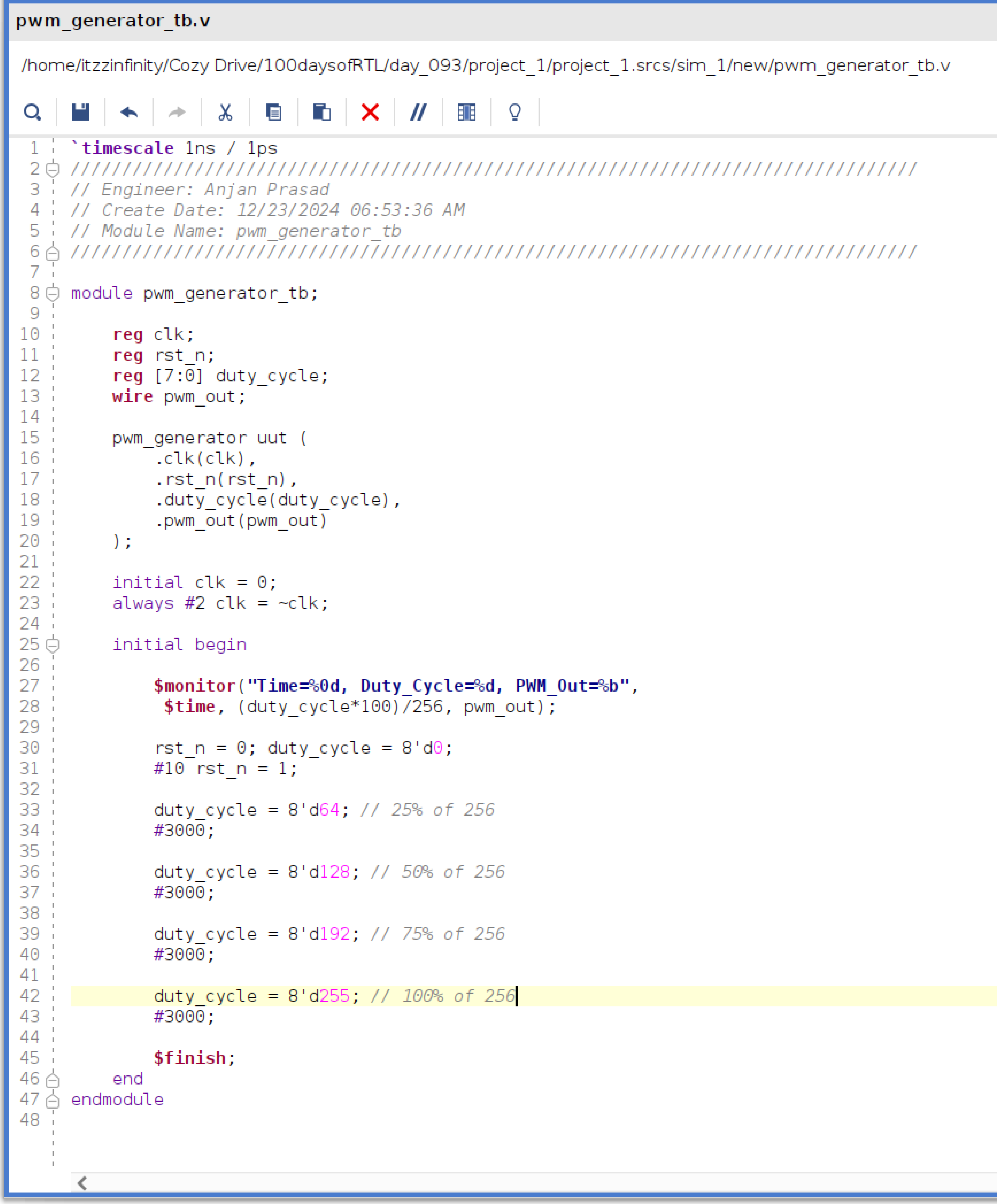


natic
11 Cells
11 I/O Ports
29 Nets
rst_n
clk
duty_cycle[7:0]
counter0_i
I1
I0[7:0]
O[7:0]
+
RTL_ADD
counter_reg[7:
CLR
C
Q
D
RTL_REG_ASYNC


pwm_out_reg
CLR
C
D
Q
RTL_REG_ASYNC
pwm_out
pwm_out0_i
I0[7:0]
I1[7:0]
<
O
RTL_LT


pwm_generator.v
/home/itzzinfinity/Cozy Drive/100daysofRTL/day_093/project_1/project_1.srcs/sources_1/new/pwm_generator.v

```verilog
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Engineer: Anjan Prasad
// Create Date: 12/23/2024 06:52:33 AM
// Module Name: pwm_generator
//////////////////////////////////////////////////////////////////////////////////

module pwm_generator (
    input clk,              // System clock
    input rst_n,            // Active low reset
    input [7:0] duty_cycle, // Duty cycle (0 to 255 for 8-bit resolution)
    output reg pwm_out      // PWM output signal
);

    reg [7:0] counter;      // 8-bit counter for PWM

    always @(posedge clk or negedge rst_n) begin
        if (!rst_n)
            counter <= 8'd0;
        else
            counter <= counter + 1;
    end

    // PWM generation
    always @(posedge clk or negedge rst_n) begin
        if (!rst_n)
            pwm_out <= 1'b0;
        else
            pwm_out <= (counter < duty_cycle) ? 1'b1 : 1'b0;
    end

endmodule
```


pwm_generator_tb.v
/home/itzzinfinity/Cozy Drive/100daysofRTL/day_093/project_1/project_1.srcs/sim_1/new/pwm_generator_tb.v

```
`timescale 1ns / 1ps
//////////////////////////////////////////////////////////////////////////////////
// Engineer: Anjan Prasad
// Create Date: 12/23/2024 06:53:36 AM
// Module Name: pwm_generator_tb
//////////////////////////////////////////////////////////////////////////////////

module pwm_generator_tb;

    reg clk;
    reg rst_n;
    reg [7:0] duty_cycle;
    wire pwm_out;

    pwm_generator uut (
        .clk(clk),
        .rst_n(rst_n),
        .duty_cycle(duty_cycle),
        .pwm_out(pwm_out)
    );

    initial clk = 0;
    always #2 clk = ~clk;

    initial begin

        $monitor("Time=%0d, Duty_Cycle=%d, PWM_Out=%b",
         $time, (duty_cycle*100)/256, pwm_out);

        rst_n = 0; duty_cycle = 8'd0;
        #10 rst_n = 1;

        duty_cycle = 8'd64; // 25% of 256
        #3000;

        duty_cycle = 8'd128; // 50% of 256
        #3000;

        duty_cycle = 8'd192; // 75% of 256
        #3000;

        duty_cycle = 8'd255; // 100% of 256
        #3000;

        $finish;
    end
endmodule

```