Design of basic PWM block using Verilog

Sandy Naguib – August 2025

Contents

1. Introduction

PWM stands for Pulse Width Modulation. It is a technique used to control the amount of power given to a digital circuit. This is done by encoding analog signal with digital pulses. It changes the pulse signal's width in electrical systems to regulate the average power supplied to a load. Among the applications that PWM can be useful is controlling brightness of a LED, speed of a motor or regulating the output of audio amplifiers.

The average voltage supplied to the digital circuit is determined by the duty cycle of the PWM. Duty cycle is the ratio between time when signal is on to the whole period of the PWM.

$$Duty\ Cycle\ (\%) = \left(\frac{Time\ high}{Total\ period}\right) \times 100$$

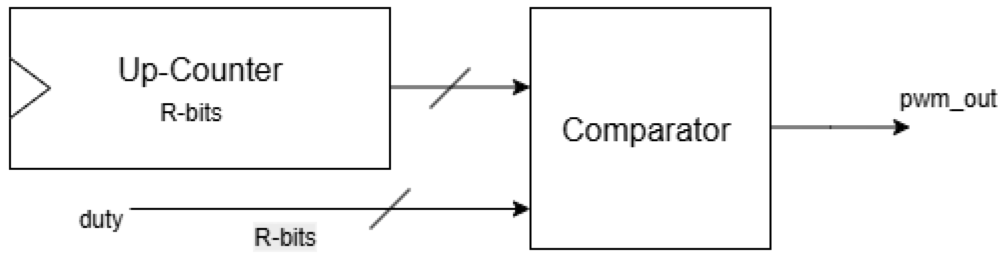
The average voltage seen by the analog device is:

$$V_{avg} = D \times V_{high}$$

2. Design Overview

2.1 Basic PWM.

Starting with the basic blocks of the system, it can be seen as follows:

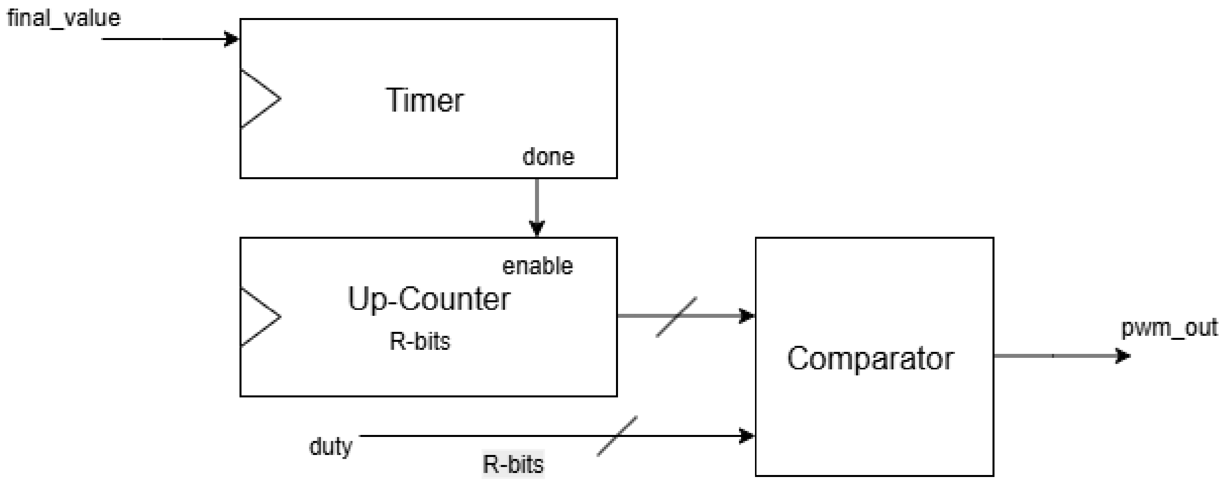


This simple system can work as a PWM, but it has some problems. It takes duty cycle as an input and have a counter that can counts to 2^R , the output pwm_out is high if the counter is less than the duty ($duty > counter\ out$) . Counter defines how long the PWM signal is.

The number of bits (R) represents the resolution of the PWM. It means how finely you can control the duty cycle and then you can control the average voltage seen by the circuit. It is the smallest change in duty cycle that your system can produce. As R increases, you have better control over the voltage.

Problem with this design: The period of the PWM signal is calculated by: $2^R \times T_{clk}$, where the counter is using the same clock of the system. However, this gives us restriction when we want to control the period of the output PWM signal. To change it you need to change either the system clock, which is not practical, or the number of bits, which is also not the best solution. So, we need another block that can give a degree of freedom to control the period, so a **timer** is needed.

2.2 Enhanced PWM design

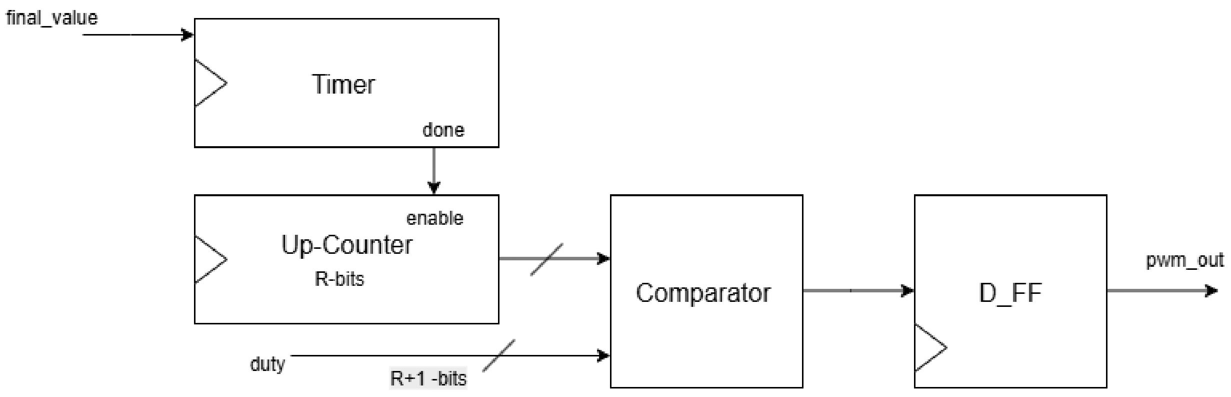


In this design, the timer generates an enable signal once it reaches the **final_value** so the counter counts up. This makes the period of the PWM signal now is given as follows:

$$T_{pwm} = 2^R \times (Timer\ final\ value + 1) \times T_{clk}$$

By changing the timer final value, we can control the period and the frequency of the PWM. We added 1 to the final value because it started counting from 0.

2.3 Final improved design

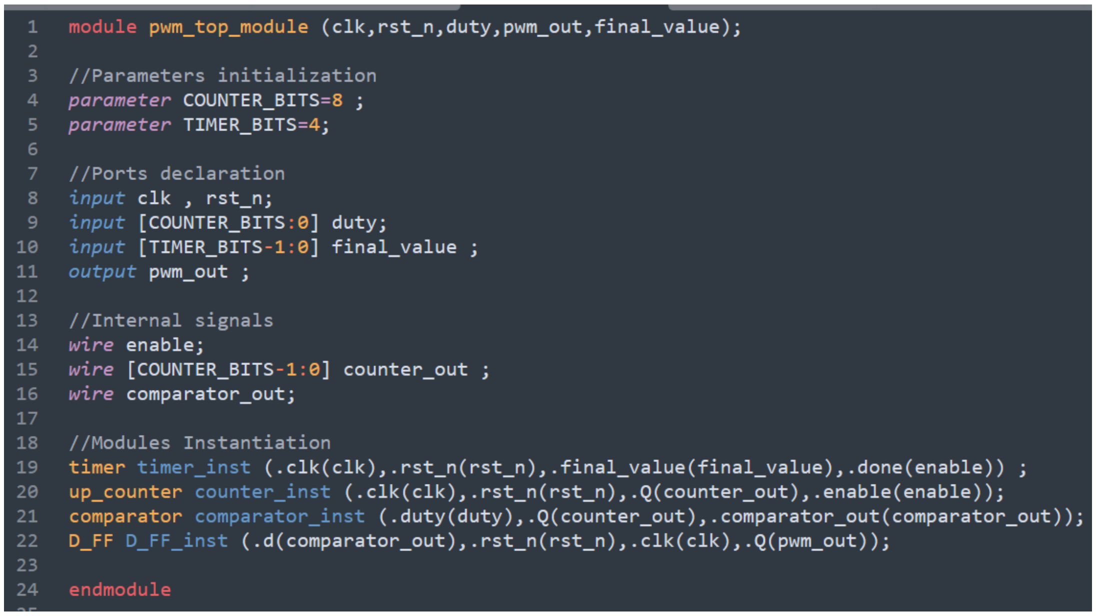


<u>Modifications:</u>

- D flip flop is added because the comparator is a combinational circuit, and we want our system to be synchronized to the system clock and output goes only at the clock edge and it also helps in avoiding some glitches.
- The number of duty bits increased to R+1, because at 100% duty cycle some glitches occur if it is only R bits. Assume R = 8 , to achieve duty 100% it will compare 255 < 255 which will make pwm_out signal low for a while. When increasing duty to extra one bit , 100% duty cycle will be 256 while the counter maximum value is 255 so it will stay high all over the cycle.

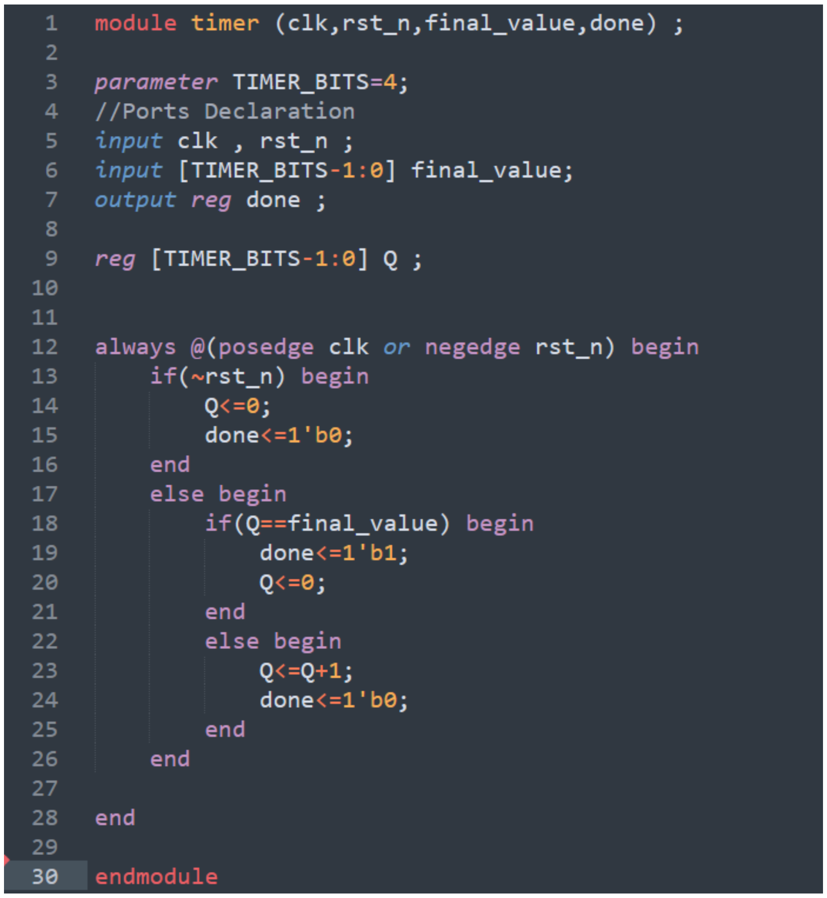
3 . RTL Code

3.1 PWM Top module



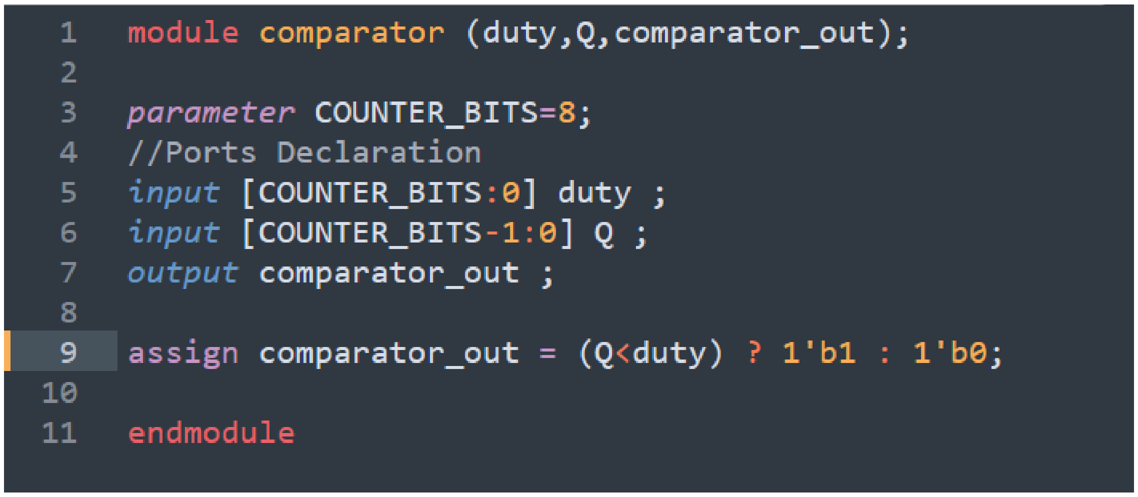
```verilog
module pwm_top_module (clk,rst_n,duty,pwm_out,final_value);

//Parameters initialization
parameter COUNTER_BITS=8 ;
parameter TIMER_BITS=4;

//Ports declaration
input clk , rst_n;
input [COUNTER_BITS:0] duty;
input [TIMER_BITS-1:0] final_value ;
output pwm_out ;

//Internal signals
wire enable;
wire [COUNTER_BITS-1:0] counter_out ;
wire comparator_out;

//Modules Instantiation
timer timer_inst (.clk(clk),.rst_n(rst_n),.final_value(final_value),.done(enable)) ;
up_counter counter_inst (.clk(clk),.rst_n(rst_n),.Q(counter_out),.enable(enable));
comparator comparator_inst (.duty(duty),.Q(counter_out),.comparator_out(comparator_out));
D_FF D_FF_inst (.d(comparator_out),.rst_n(rst_n),.clk(clk),.Q(pwm_out));

endmodule
```

3.2 Timer module



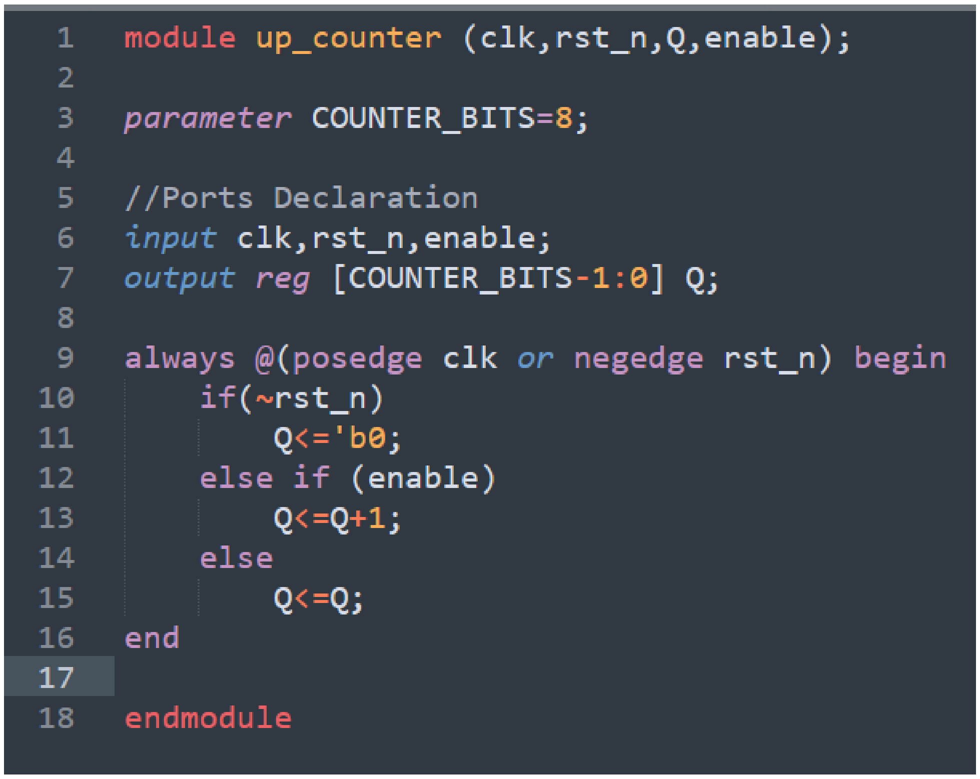
```verilog
module timer (clk,rst_n,final_value,done) ;

parameter TIMER_BITS=4;
//Ports Declaration
input clk , rst_n ;
input [TIMER_BITS-1:0] final_value;
output reg done ;

reg [TIMER_BITS-1:0] Q ;


always @(posedge clk or negedge rst_n) begin
    if(~rst_n) begin
        Q<=0;
        done<=1'b0;
    end
    else begin
        if(Q==final_value) begin
            done<=1'b1;
            Q<=0;
        end
        else begin
            Q<=Q+1;
            done<=1'b0;
        end
    end

end

endmodule
```

3.3 Comparator module



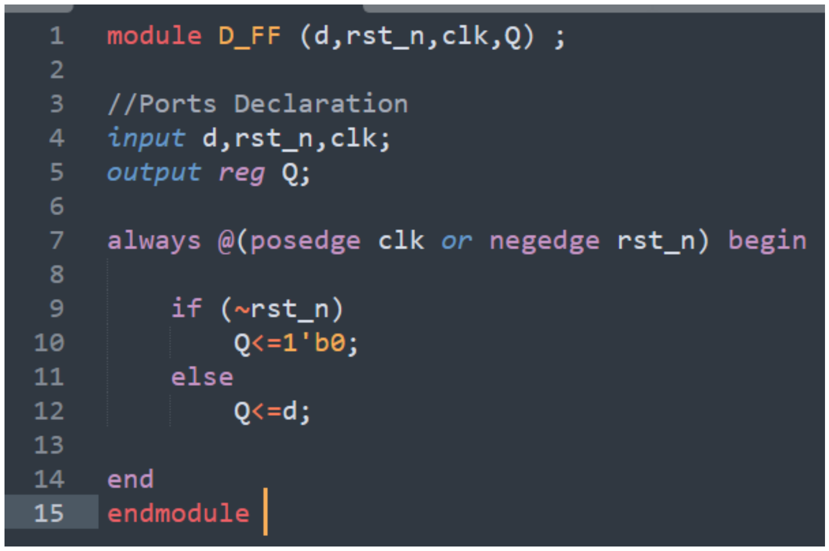
```verilog
module comparator (duty,Q,comparator_out);

parameter COUNTER_BITS=8;
//Ports Declaration
input [COUNTER_BITS:0] duty ;
input [COUNTER_BITS-1:0] Q ;
output comparator_out ;

assign comparator_out = (Q<duty) ? 1'b1 : 1'b0;

endmodule
```

3.4 Counter module



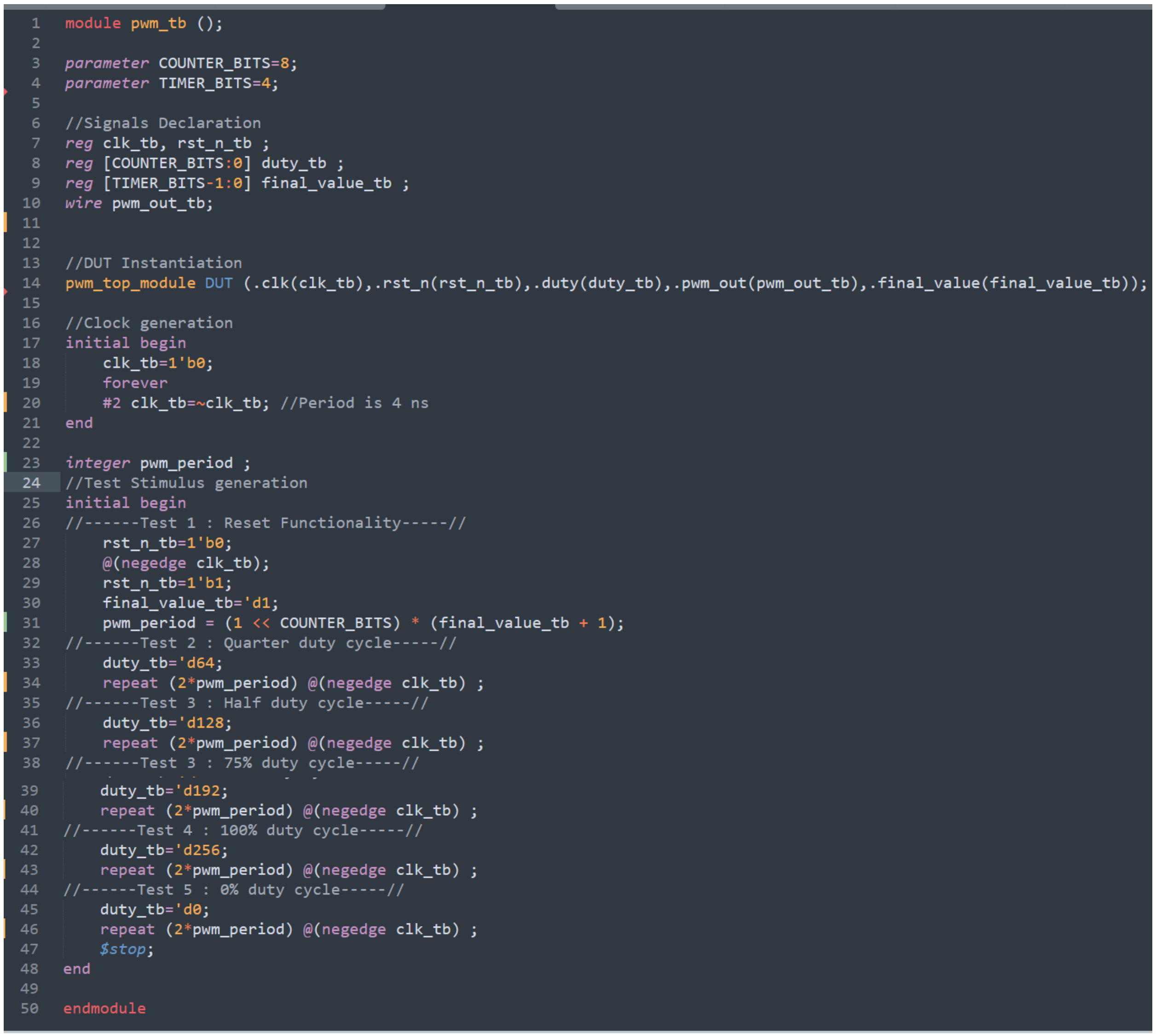
```verilog
module up_counter (clk,rst_n,Q,enable);

parameter COUNTER_BITS=8;

//Ports Declaration
input clk,rst_n,enable;
output reg [COUNTER_BITS-1:0] Q;

always @(posedge clk or negedge rst_n) begin
    if(~rst_n)
        Q<='b0;
    else if (enable)
        Q<=Q+1;
    else
        Q<=Q;
end

endmodule
```

3.5 D - Flip flop



```
module D_FF (d,rst_n,clk,Q) ;

//Ports Declaration
input d,rst_n,clk;
output reg Q;

always @(posedge clk or negedge rst_n) begin

    if (~rst_n)
        Q<=1'b0;
    else
        Q<=d;

end
endmodule
```

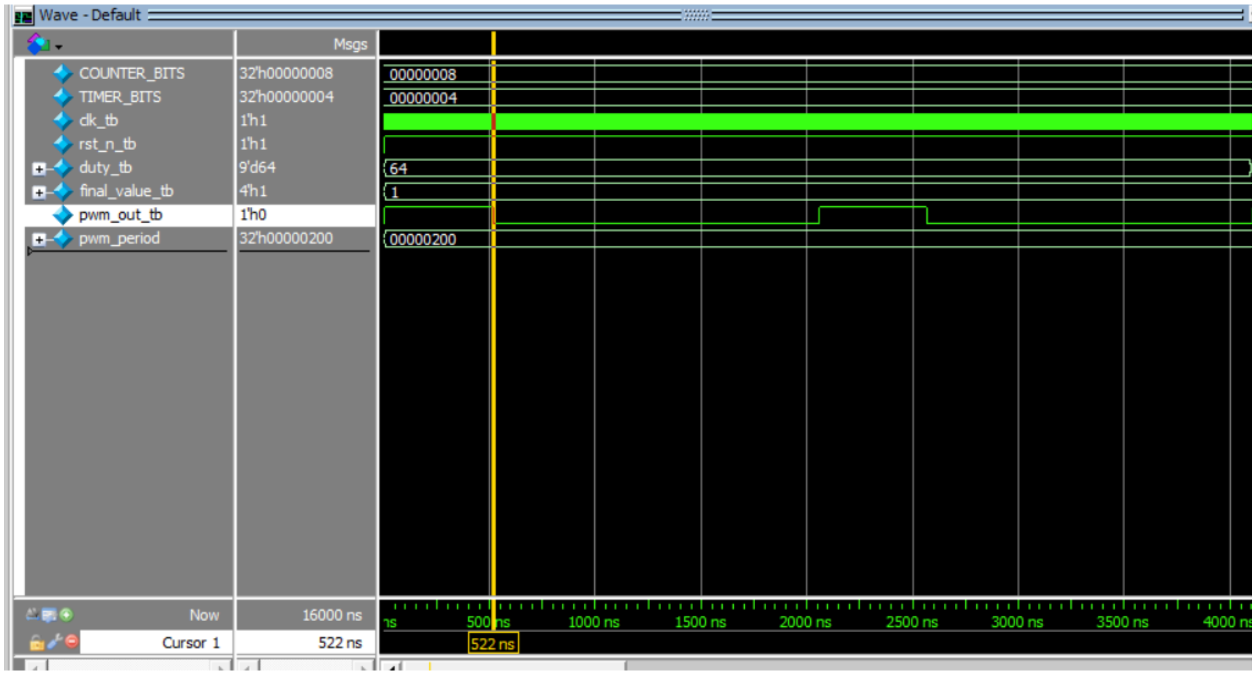
4. Testbench Code



```verilog
module pwm_tb ();

parameter COUNTER_BITS=8;
parameter TIMER_BITS=4;

//Signals Declaration
reg clk_tb, rst_n_tb ;
reg [COUNTER_BITS:0] duty_tb ;
reg [TIMER_BITS-1:0] final_value_tb ;
wire pwm_out_tb;


//DUT Instantiation
pwm_top_module DUT (.clk(clk_tb),.rst_n(rst_n_tb),.duty(duty_tb),.pwm_out(pwm_out_tb),.final_value(final_value_tb));

//Clock generation
initial begin
    clk_tb=1'b0;
    forever
    #2 clk_tb=~clk_tb; //Period is 4 ns
end

integer pwm_period ;
//Test Stimulus generation
initial begin
//------Test 1 : Reset Functionality-----//
    rst_n_tb=1'b0;
    @(negedge clk_tb);
    rst_n_tb=1'b1;
    final_value_tb='d1;
    pwm_period = (1 << COUNTER_BITS) * (final_value_tb + 1);
//------Test 2 : Quarter duty cycle-----//
    duty_tb='d64;
    repeat (2*pwm_period) @(negedge clk_tb) ;
//------Test 3 : Half duty cycle-----//
    duty_tb='d128;
    repeat (2*pwm_period) @(negedge clk_tb) ;
//------Test 3 : 75% duty cycle-----//
    duty_tb='d192;
    repeat (2*pwm_period) @(negedge clk_tb) ;
//------Test 4 : 100% duty cycle-----//
    duty_tb='d256;
    repeat (2*pwm_period) @(negedge clk_tb) ;
//------Test 5 : 0% duty cycle-----//
    duty_tb='d0;
    repeat (2*pwm_period) @(negedge clk_tb) ;
    $stop;
end

endmodule
```

5. Simulation Waveforms

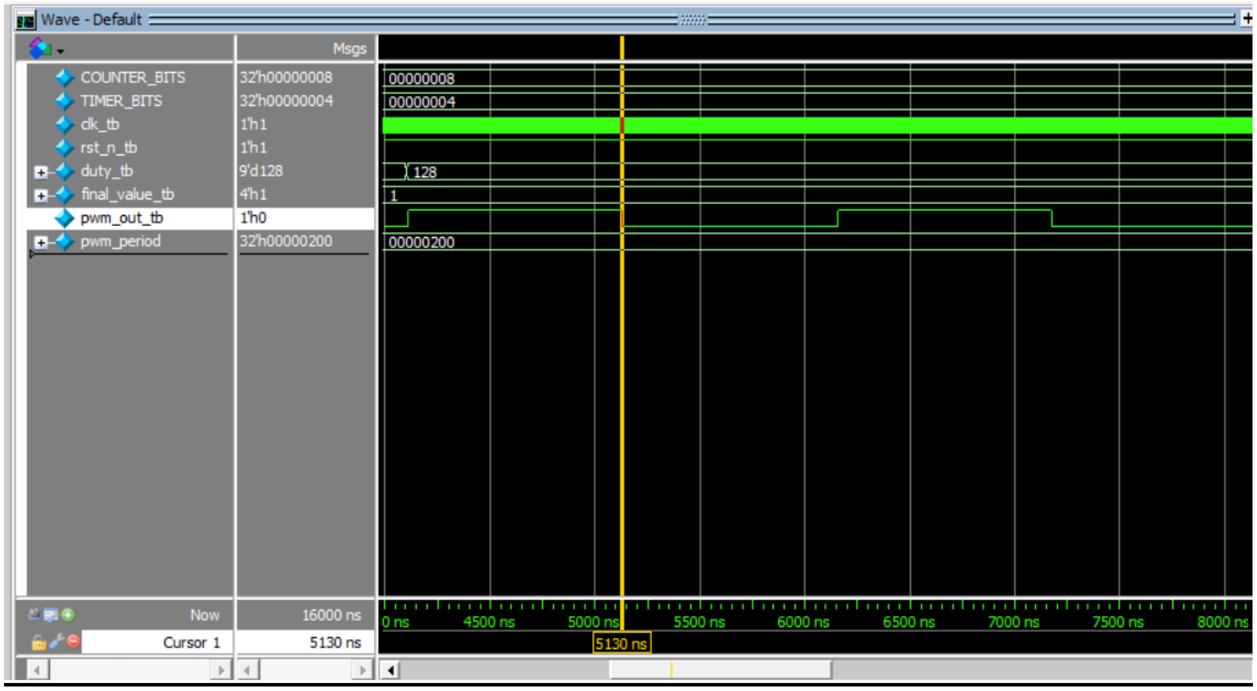
5.1: 25% duty cycle



Wave - Default

	Msgs
COUNTER_BITS	32'h00000008
TIMER_BITS	32'h00000004
clk_tb	1'h1
rst_n_tb	1'h1
duty_tb	9'd64
final_value_tb	4'h1
pwm_out_tb	1'h0
pwm_period	32'h00000200
Now	16000 ns
Cursor 1	522 ns

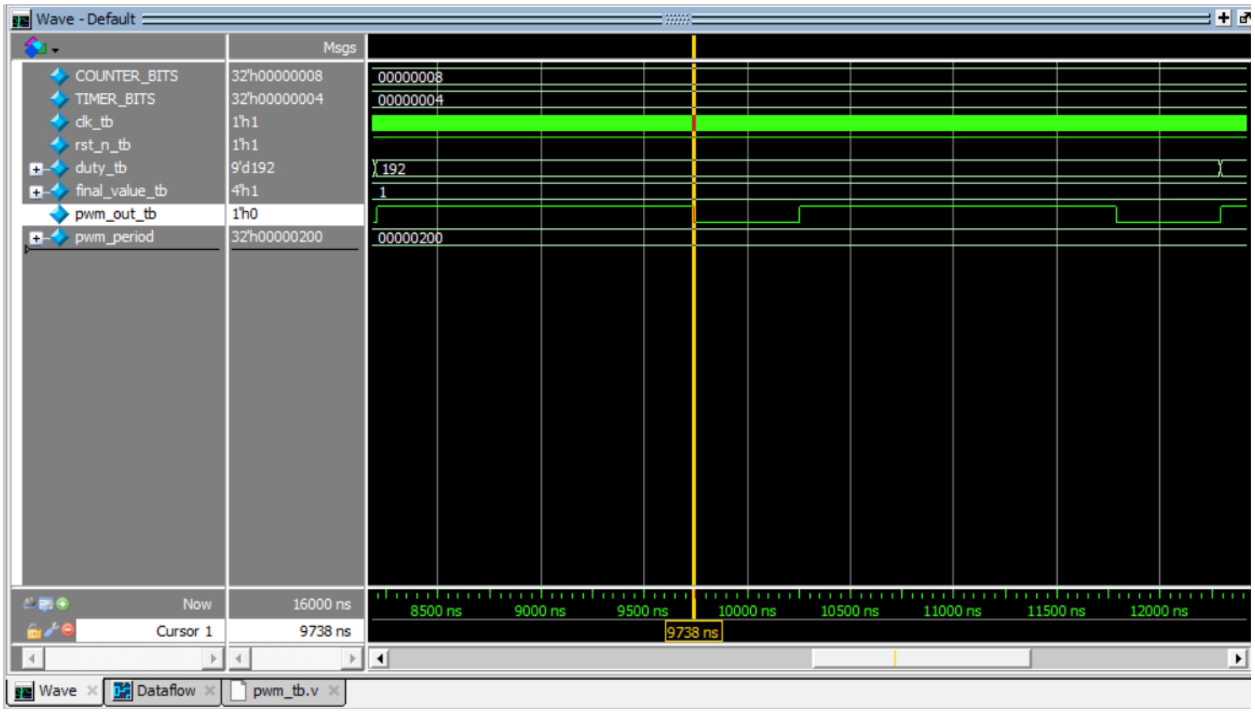
5.2: 50% duty cycle



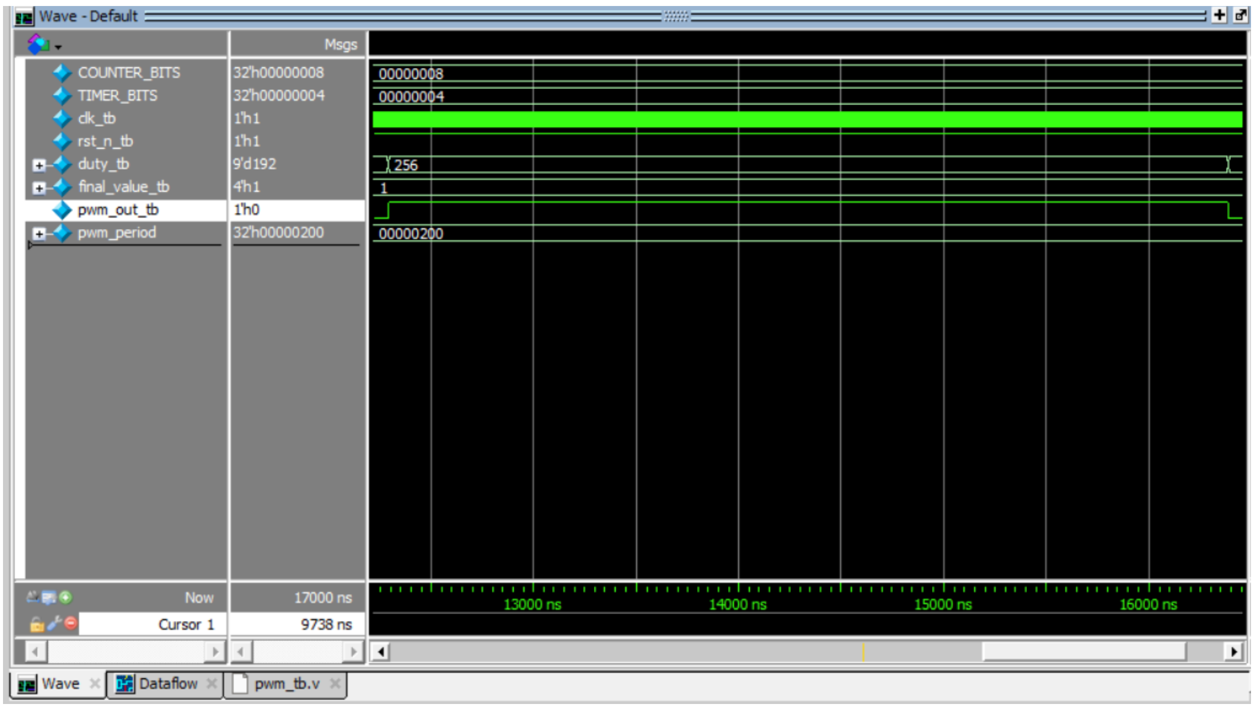
Wave - Default

	Msgs
COUNTER_BITS	32'h00000008
TIMER_BITS	32'h00000004
clk_tb	1'h1
rst_n_tb	1'h1
duty_tb	9'd128
final_value_tb	4'h1
pwm_out_tb	1'h0
pwm_period	32'h00000200
Now	16000 ns
Cursor 1	5130 ns

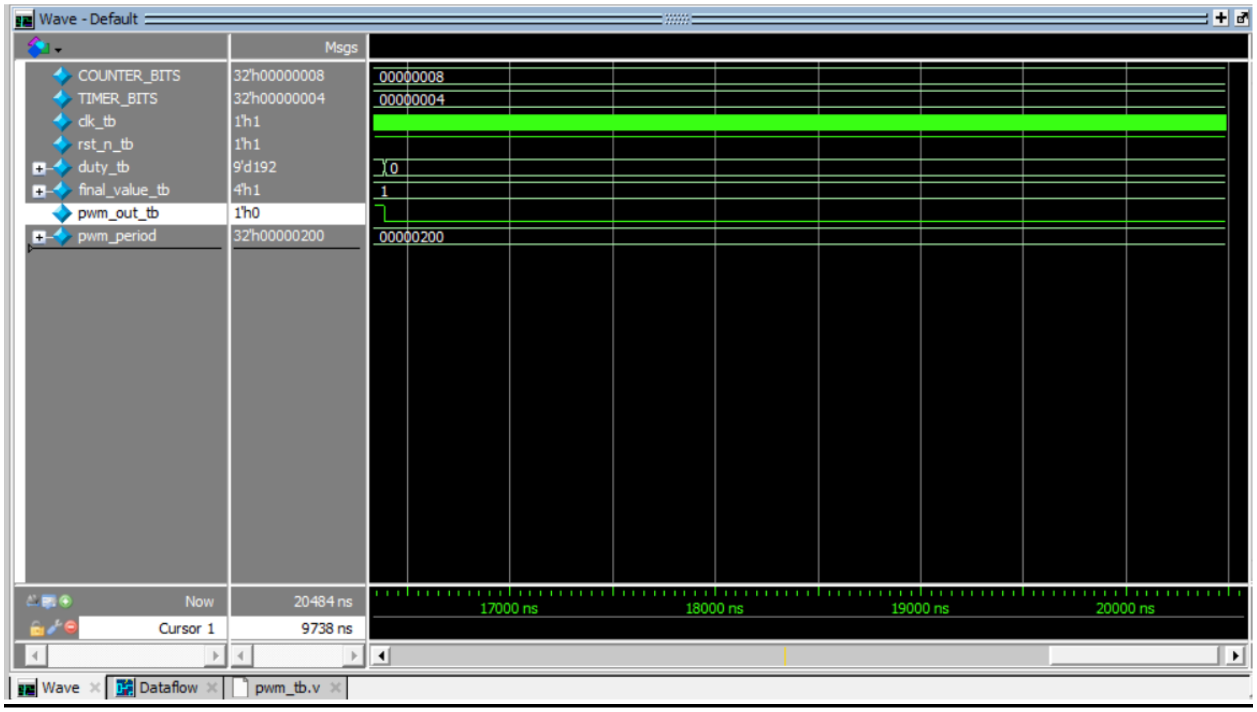
5.3 75% duty cycle



5.4 100% duty cycle

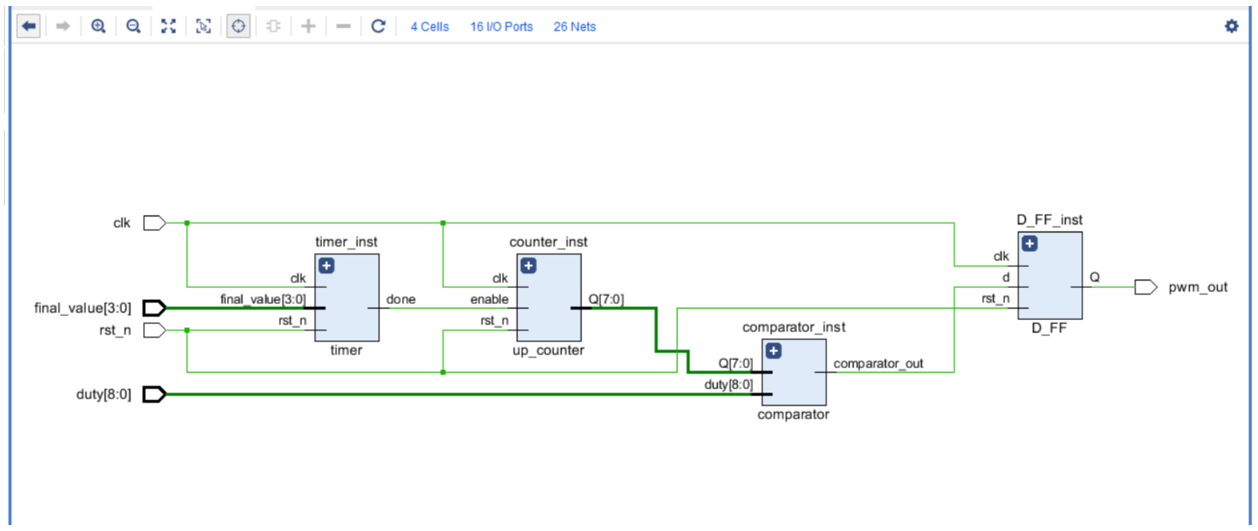


5.5 0% duty cycle

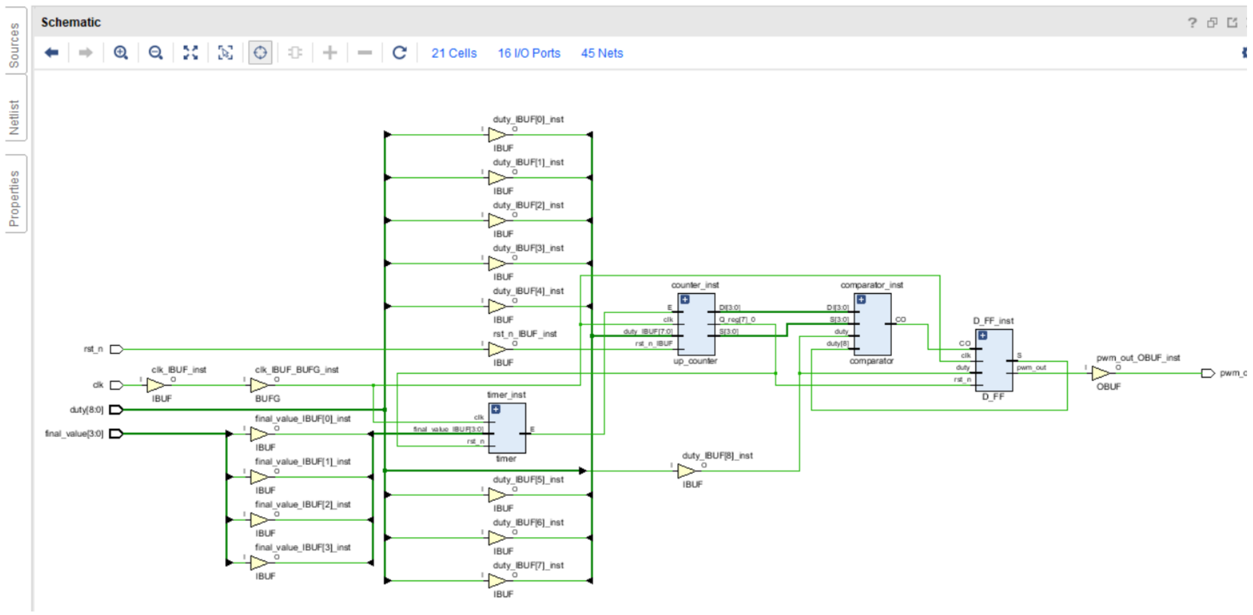


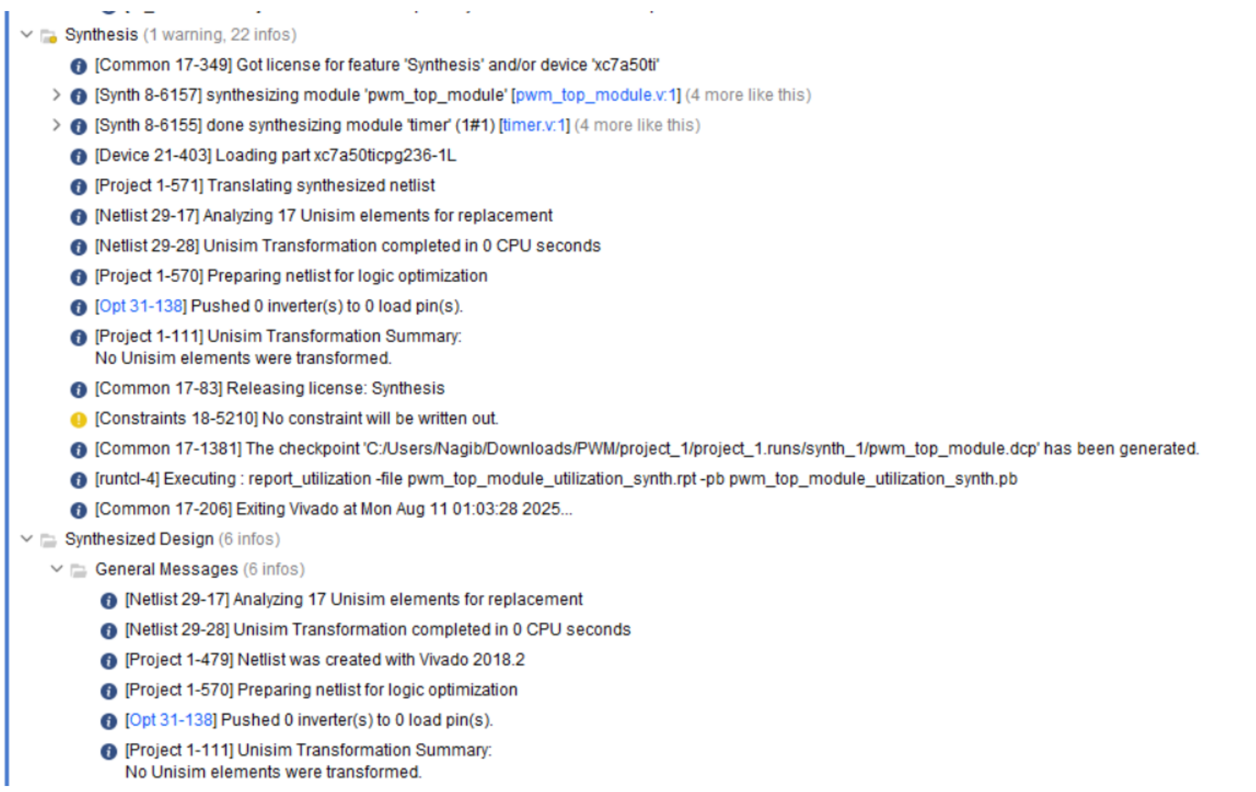
	Msgs	
COUNTER_BITS	32'h00000008	00000008
TIMER_BITS	32'h00000004	00000004
clk_tb	1'h1	
rst_n_tb	1'h1	
duty_tb	9'd192	0
final_value_tb	4'h1	1
pwm_out_tb	1'h0	
pwm_period	32'h00000200	00000200
Now	20484 ns	17000 ns 18000 ns 19000 ns 20000 ns
Cursor 1	9738 ns	

6. **Elaborated Design using Vivado**

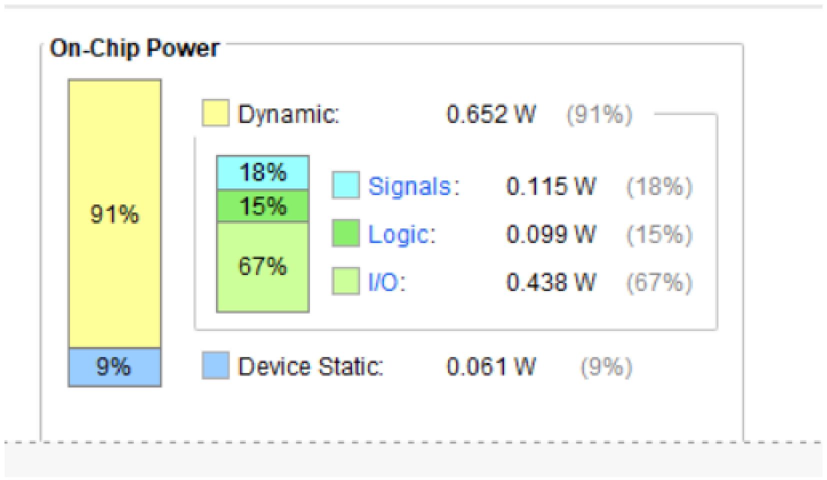


7. **Synthesis**



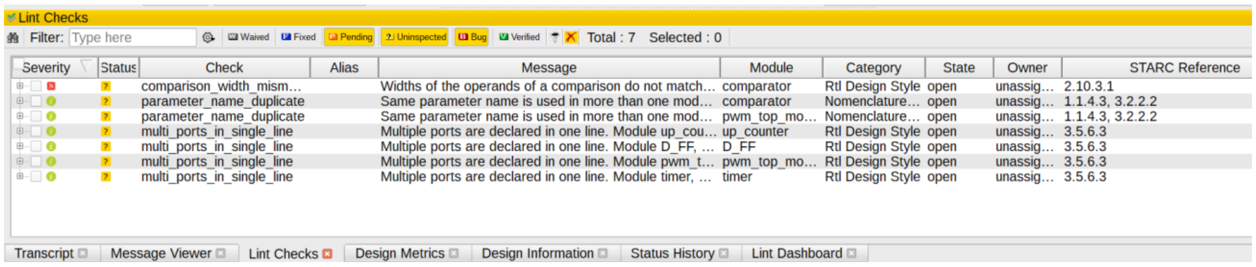


```
Synthesis (1 warning, 22 infos)
    [Common 17-349] Got license for feature 'Synthesis' and/or device 'xc7a50ti'
  > [Synth 8-6157] synthesizing module 'pwm_top_module' [pwm_top_module.v:1] (4 more like this)
  > [Synth 8-6155] done synthesizing module 'timer' (1#1) [timer.v:1] (4 more like this)
    [Device 21-403] Loading part xc7a50ticpg236-1L
    [Project 1-571] Translating synthesized netlist
    [Netlist 29-17] Analyzing 17 Unisim elements for replacement
    [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
    [Project 1-570] Preparing netlist for logic optimization
    [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
    [Project 1-111] Unisim Transformation Summary:
    No Unisim elements were transformed.
    [Common 17-83] Releasing license: Synthesis
    [Constraints 18-5210] No constraint will be written out.
    [Common 17-1381] The checkpoint 'C:/Users/Nagib/Downloads/PWM/project_1/project_1.runs/synth_1/pwm_top_module.dcp' has been generated.
    [runtcl-4] Executing : report_utilization -file pwm_top_module_utilization_synth.rpt -pb pwm_top_module_utilization_synth.pb
    [Common 17-206] Exiting Vivado at Mon Aug 11 01:03:28 2025...
Synthesized Design (6 infos)
  General Messages (6 infos)
      [Netlist 29-17] Analyzing 17 Unisim elements for replacement
      [Netlist 29-28] Unisim Transformation completed in 0 CPU seconds
      [Project 1-479] Netlist was created with Vivado 2018.2
      [Project 1-570] Preparing netlist for logic optimization
      [Opt 31-138] Pushed 0 inverter(s) to 0 load pin(s).
      [Project 1-111] Unisim Transformation Summary:
      No Unisim elements were transformed.
```



Dynamic:	0.652 W	(91%)
Signals:	0.115 W	(18%)
Logic:	0.099 W	(15%)
I/O:	0.438 W	(67%)
Device Static:	0.061 W	(9%)

8. Linting

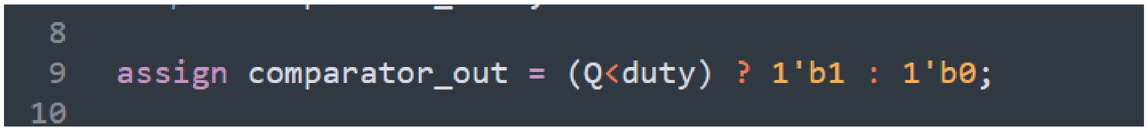


Lint Checks

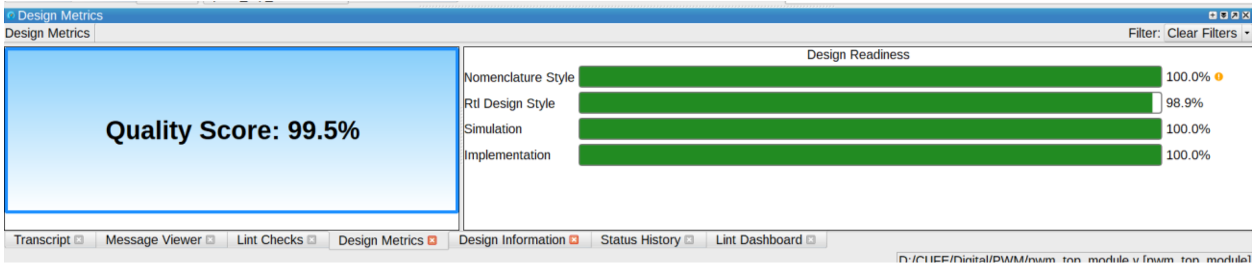
Filter: Type here | Waived | Fixed | Pending | Uninspected | Bug | Verified | Total : 7 Selected : 0

Severity	Status	Check	Alias	Message	Module	Category	State	Owner	STARC Reference
🟥	?	comparison_width_mism...		Widths of the operands of a comparison do not match...	comparator	Rtl Design Style	open	unassig...	2.10.3.1
●	?	parameter_name_duplicate		Same parameter name is used in more than one mod...	comparator	Nomenclature...	open	unassig...	1.1.4.3, 3.2.2.2
●	?	parameter_name_duplicate		Same parameter name is used in more than one mod...	pwm_top_mo...	Nomenclature...	open	unassig...	1.1.4.3, 3.2.2.2
●	?	multi_ports_in_single_line		Multiple ports are declared in one line. Module up_cou...	up_counter	Rtl Design Style	open	unassig...	3.5.6.3
●	?	multi_ports_in_single_line		Multiple ports are declared in one line. Module D_FF, ...	D_FF	Rtl Design Style	open	unassig...	3.5.6.3
●	?	multi_ports_in_single_line		Multiple ports are declared in one line. Module pwm_t...	pwm_top_mo...	Rtl Design Style	open	unassig...	3.5.6.3
●	?	multi_ports_in_single_line		Multiple ports are declared in one line. Module timer, ...	timer	Rtl Design Style	open	unassig...	3.5.6.3

The error shown by the linting tool is because of the line below in the comparator code. But it is okay, **and it can be waived** because it is intended that duty variable must be one bit more than the counter to avoid glitches in 100% duty cycle.



```
assign comparator_out = (Q<duty) ? 1'b1 : 1'b0;
```



Design Metrics

Quality Score: 99.5%

Design Readiness

Nomenclature Style	100.0%
Rtl Design Style	98.9%
Simulation	100.0%
Implementation	100.0%