MILITARY INSTITUTE OF SCIENCE AND TECHNOLOGY

COURSE CODE: EECE-318

COURSE TITLE: VLSI 1

Submitted To:

Sunjida Sultana — Asst. Prof, EECE Dept

K. M. Daiyan — Lecturer, EECE Dept

Submitted By:

Group-3

SEC-B

Project Name: LED Brightness controller based on PWM using verilog

Name:	**Student ID:**
Nafiz Bin Hasanat	202116070
Nasif Ahmed Rafe	202116072
Samiha Hasan	202116085

Index

Introduction:

a. PULSE WIDTH MODULATION (PWM):

PWM stands for Pulse Width Modulation. It is a method of controlling the average power or amplitude delivered by an electrical signal. The signal thus produced will have a train of pulses and these pulses will be in form of a square wave. That is, at any given instance of time the wave will either be high or will be low. It is achieved by changing the width of the signal & keeping the time period or frequency same. It is a powerful technique for controlling analog circuits with digital outputs. For a PWM signal we need to look at two important parameters associated with it one is the PWM duty cycle and the other is PWM frequency.

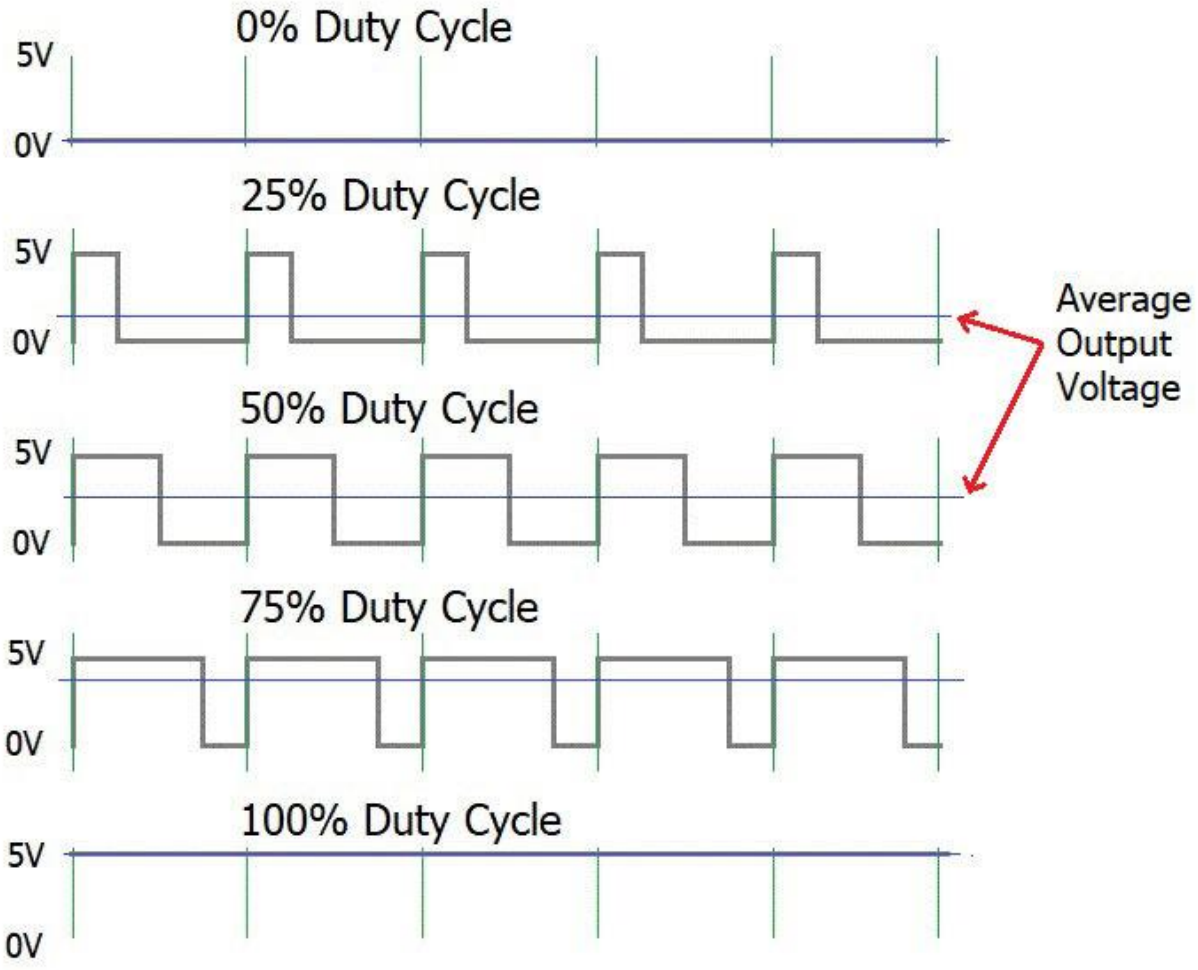


Figure: pulse width modulation

b. Duty Cycle of PWM:

PWM signal stays on for a particular time and then stays off for the rest of the period. What makes this PWM signal special and more useful is that we can set for how long it should stay on by controlling the duty cycle of the PWM signal.

The percentage of time in which the PWM signal remains HIGH (on time) is called as duty cycle. If the signal is always ON it is in 100% duty cycle and if it is always off it is 0% duty cycle. The formulae to calculate the duty cycle is shown below.
Duty Cycle =Turn ON time/ (Turn ON time + Turn OFF time)
By controlling the Duty cycle from 0% to 100% we can control the "on time" of PWM signal and thus the width of signal.

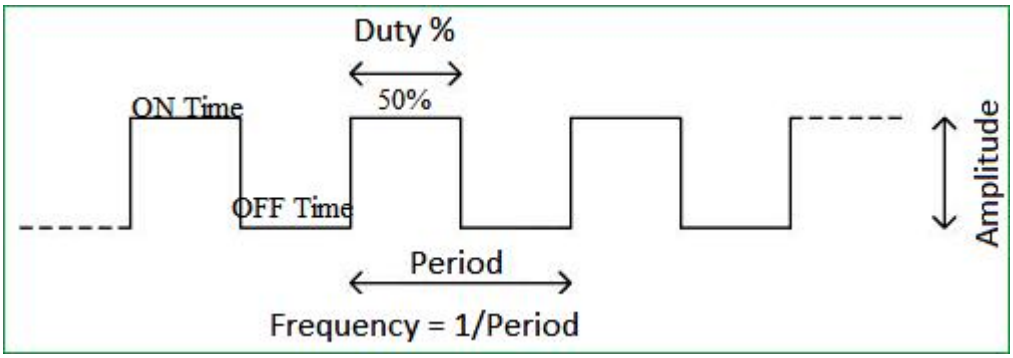


Figure: duty cycle of PWM

c. <u>Frequency of PWM:</u>

The frequency of a PWM signal determines how fast a PWM completes one period. One Period is the complete ON and OFF time of a PWM signal.
Frequency = 1/Time Period
Time Period = On time + Off time
How fast the PWM signal should turn on and turn off is decided by the frequency of the PWM signal and in that speed how long the PWM signal should remain turned on is decided by the duty cycle of the PWM signal.

d. <u>Advantages of PWM:</u>

Pulse Width Modulation (PWM) utilizes digital signals for controlling power applications and offers a relatively straightforward conversion back to analog with minimal hardware requirements. In contrast to analog systems like linear power supplies, which generate substantial heat due to acting as variable resistors carrying significant current, digital systems typically produce less heat. The primary source of heat in a switching device occurs during transitions, which are executed swiftly when the device is in an intermediate state between on and off. This is crucial because power is determined by the formula $P = E \times I$.
When either voltage or current approaches zero, power also approaches zero. PWM uses this principle effectively. It can exhibit characteristics of an analog control system, as the digital signal can freely transition. Moreover, PWM does not necessarily require data capture, although exceptions exist with more advanced controllers.

e. <u>Application of PWM:</u>

PWM works by pulsating DC current, and varying the amount of time that each pulse stays 'on' to control the amount of current that flows to a device such as an LED.

The longer each pulse is on, the brighter the LED will be. Due to the fact that the interval between pulses is so brief, the LED doesn't actually turn off. In other words, the LED's power source switches on and off so fast (thousands of times per second) that the LED actually stays on without flickering. With an RGB (red green blue) LED, you can control how much of each of the three colors you want in the mix of color by dimming them with various amounts. If all three are on in equal amounts, the result will be white light of varying brightness. Blue equally mixed with green will get teal.

Methodology:

To generate a PWM the following method was maintained:

1. Take a duty cycle of R bits.
2. Design an UP counter that counts from 0 to 2^R-1. Here R is the number of bits.
3. Design a Comparator that compares between the value of duty cycle and UP counter.
4. The output of the comparator will be HIGH as long as the UP counter's value is less than the duty cycle.
5. The output of the comparator will be LOW when the UP counter's value is greater than the duty cycle.
6. This is how the PWM will be generated.

The following procedure was maintained in the Verilog coding:

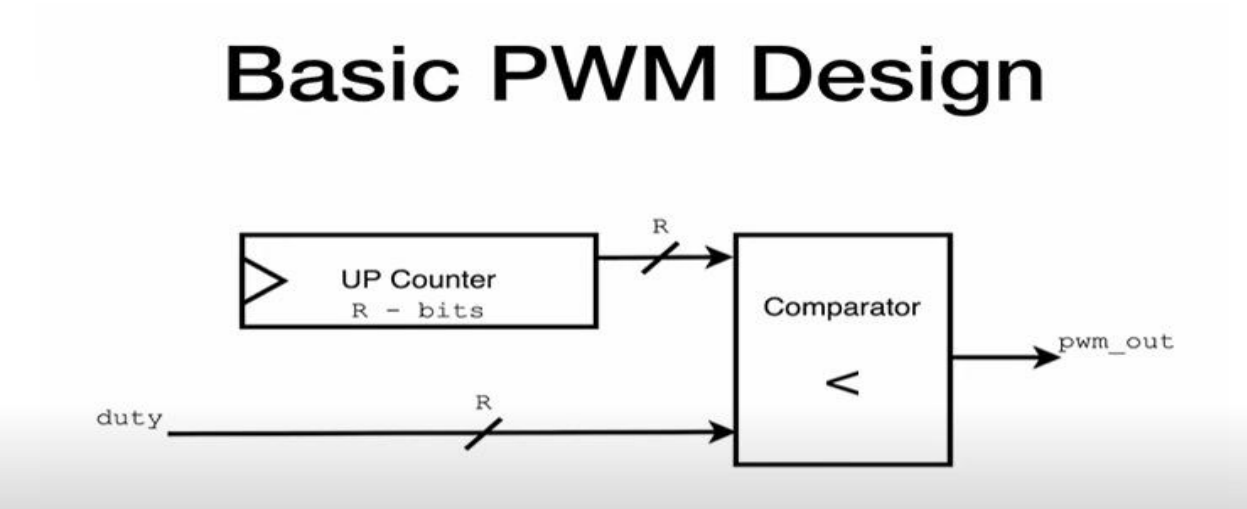
In design code,

1. A module named pwm_generator was taken that had 3 inputs (Duty cycle, Clock and reset) and 1 output (pwm_out).
2. A parameter R=8 was declared.
3. Two registers (Q_reg and Q_next) with a width of R bits was declared.
4. A synchronous reset was set for the counter. On the negative edge of the reset signal, if reset is active (low), Q_reg is set to 0. Otherwise, it retains its current value,Q_next.
5. The Counter logic was declared. On every change in any signal within the sensitivity list (*), Q_next is updated by incrementing Q_reg by 1.
6. The 'assign' statement was used to continuously assign the value of (Q_next < duty) to the pwm_out output. The expression (Q_next < duty) checks if the counter value is less than the duty cycle, producing a PWM signal.

In testbench code,

1. The simulation timescale was set to 1 nanosecond for simulation time and 1 picosecond for time precision.
2. clk, reset, and duty was declared as registers (reg).pwm_out was declared as a wire (wire).
3. A clock signal (clk) was generated with a period of 10 time units (1ns) using the forever loop. The clock toggles between 0 and 1 every 5 time units.
4. The reset signal was asserted and deasserted.
5. An initial duty cycle was set and the duty cycle was set to change at specified intervals.
6. First the duty cycle was 0.25 then 0.5, 0.75 and 0.9 respectively.

Block diagram:



Design code:

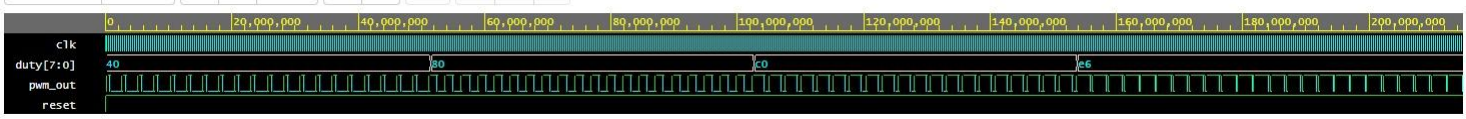
```
module pwm_generator(duty,clk,reset,pwm_out);
  parameter R=8;
  input clk,reset;
  input [R -1:0]duty;
  output pwm_out;
  reg [R-1:0]Q_reg,Q_next;
  always @(posedge clk, negedge reset)
    begin
      if (~reset)
        Q_reg<= 'b0;
      else
        Q_reg<=Q_next;
    end
  always@(*)
    begin
      Q_next=Q_reg+1;
    end
  assign pwm_out=(Q_next<duty);
endmodule
```

Testbench code:

```
`timescale 1ns/1ps
module pwm_generator_tb;
 localparam R=8;
  reg clk;
  reg reset;
  reg [7:0] duty;
  wire pwm_out;

  // Instantiate the PWM generator module
  pwm_generator uut (
    .clk(clk),
```

```
    .reset(reset),
    .duty(duty),
    .pwm_out(pwm_out)
  );

  // Clock generation
  initial begin
    clk = 0;
    forever #5 clk = ~clk;
  end

  // Stimulus
  initial begin
    reset = 0; // Assert reset
    duty = 8'b00000000; // Set initial duty cycle

    #10 reset = 1; // Deassert reset
    repeat(2*2**R)@(negedge clk)
    #100 duty = 0.25*(2**R); // Change duty cycle
    repeat(2*2**R)@(negedge clk)
      #100 duty = 0.5*(2**R);
     repeat(2*2**R)@(negedge clk)
      #100 duty = 0.75*(2**R);
     repeat(2*2**R)@(negedge clk)
       #100 duty = 0.9*(2**R);
    #10000 $finish; // Finish simulation after some time
  end

  initial begin
 // Dump waves
    $dumpfile("dump.vcd");
    $dumpvars(1);
  end

endmodule
```

Outputs:



Here is the output waveform of the code,

There are 4 waves (clk, reset, duty and pwm_out). First the duty cycle was 0.25(40H) then 0.5(80H), 0.75(C0H) and 0.9(E6H) respectively shown in the output.

Conclusion:

This design for the LED brightness controller utilizing PWM (Pulse Width Modulation) demonstrates a comprehensive description for controlling the brightness of an LED. The design effectively leverages digital signals to manage power applications, offering a flexible and efficient means of adjusting LED intensity. The use of PWM allows for a smooth transition between different brightness levels, capitalizing on the digital nature of the control system.