PWM GENERATOR

JATOTH VISHWAJITH RATHOD(EE16BTECH11014)
MALKAJGIRI BHAVYA SRI(EE16BTECH11020)

April 30, 2019

1 Information

This document is on the project Pulse width modulation. AIM:TO PREPARE A pulse width modulator .

2 Software Required

1. Verilog Language
2. iverilog software

3 Hardware Required

1. bread board,ico board
2. aurdino uno
3. raspberrypi
4. led lights
5. a motor with fan
6. multi strand wires
7. resistors
8. potentiometers,2 push button

4 Working

In pwm generator clock one of the seven segment displays is given power and the digit corresponding to that seven segment display is displayed. The digits that are displayed on the seven segment display is operated by using Boolean functions.The code is dumped aurdino(atmega) which operates the clock accordingly.

The time in the digital clock is changed by using two potentiometers and a push button. The potentiometers are connected to the analog pin of ATmega. At first, the push button is high and potentiometer is used in changing the hour display of the clock.Then the potentiometer is used in changing minutes display and it sets the time of the clock.

5 procedure

- Testing the LEDs and motor with fan.
- Writing code for pulse width modulator in verilog.
- writing the testbench code and pcf file for the input output pins.
- Implementing the increasing and decrease duty cycle through push button.
- varying the duty cycle from 0 to 100 percent.
- Extending the modulation to both leds and motor.
- observing the result in gtk wave.
- final observation in hardware part visually.

6 Report

Here we discussed regarding generation of PWM signals with varying duty cycle using verilog code and tested on FPGA. A FPGA with ico board is used as hardware and. The comparator is necessary to compare between the data available in register and counter to generate suitable PWM signals. The generated PWM signals have a fixed frequency depended on the frequency of square wave, and a variable duty cycle that changes from 0 to 100 percent. But the frequency can be changed on FPGA board based on our requirement, without changing anything in program. These signals can be used to drive a BLDC motor.

7 PWM CODE

```
// fpga4student.com: FPGA Projects, Verilog projects, VHDL projects
// Verilog project: Verilog code for PWM Generator with variable Duty Cycle
// Two debounced buttons are used to control the duty cycle (step size: 10%)
module PWM_Generator_Verilog
 (
 clk, // 100MHz clock input
 increase_duty, // input to increase 10% duty cycle
 decrease_duty, // input to decrease 10% duty cycle
 PWM_OUT // 10MHz PWM output signal
    );
 input clk;
 input increase_duty;
 input decrease_duty;
 output PWM_OUT;
 wire slow_clk_enable; // slow clock enable signal for debouncing FFs
 reg[27:0] counter_debounce=0;//counter for creating slow clock enable signals
 wire tmp1,tmp2,duty_inc;// temp flip-f sig debouncing the increasing button
 wire tmp3,tmp4,duty_dec;// temp flip-f sig debouncing the decreasing button
 reg[3:0] counter_PWM=0;// counter for creating 10Mhz PWM signal
 reg[3:0] DUTY_CYCLE=5; // initial duty cycle is 50%
  // Debouncing 2 buttons for inc/dec duty cycle
  // Firstly generate slow clock enable for debouncing flip-flop (4Hz)
 always @(posedge clk)
 begin
   counter_debounce <= counter_debounce + 1;
   //if(counter_debounce>=25000000) then
   // for running on FPGA -- comment when running simulation
   if(counter_debounce>=1)
   // for running simulation -- comment when running on FPGA
    counter_debounce <= 0;
 end
 // assign slow_clk_enable = counter_debounce == 25000000 ?1:0;
 // for running on FPGA -- comment when running simulation
 assign slow_clk_enable = counter_debounce == 1 ?1:0;
 // for running simulation -- comment when running on FPGA
 // debouncing FFs for increasing button
 DFF_PWM PWM_DFF1(clk,slow_clk_enable,increase_duty,tmp1);
 DFF_PWM PWM_DFF2(clk,slow_clk_enable,tmp1, tmp2);
 assign duty_inc =  tmp1 & (~ tmp2) & slow_clk_enable;
 // debouncing FFs for decreasing button
 DFF_PWM PWM_DFF3(clk,slow_clk_enable,decrease_duty, tmp3);
 DFF_PWM PWM_DFF4(clk,slow_clk_enable,tmp3, tmp4);
 assign duty_dec =  tmp3 & (~ tmp4) & slow_clk_enable;
 // vary the duty cycle using the debounced buttons above
 always @(posedge clk)
 begin
   if(duty_inc==1 && DUTY_CYCLE <= 9)
    DUTY_CYCLE <= DUTY_CYCLE + 1;// increase duty cycle by 10%
   else if(duty_dec==1 && DUTY_CYCLE>=1)
    DUTY_CYCLE <= DUTY_CYCLE - 1;//decrease duty cycle by 10%
 end
// Create 10MHz PWM signal with variable duty cycle controlled by 2 buttons
```

```
 always @(posedge clk)
 begin
   counter_PWM <= counter_PWM + 1;
   if(counter_PWM>=9)
    counter_PWM <= 0;
 end
 assign PWM_OUT = counter_PWM < DUTY_CYCLE ? 1:0;
endmodule
// Debouncing DFFs for push buttons on FPGA
module DFF_PWM(clk,en,D,Q);
input clk,en,D;
output reg Q;
always @(posedge clk)
begin
 if(en==1) // slow clock enable signal
  Q <= D;
end
endmodule

timescale 1ns / 1ps
// fpga4student.com: FPGA Projects, Verilog projects, VHDL projects
//  Verilog testbench code for PWM Generator with variable duty cycle
module tb_PWM_Generator_Verilog;
 // Inputs
 reg clk;
 reg increase_duty;
 reg decrease_duty;
 // Outputs
 wire PWM_OUT;
 // Instantiate the PWM Generator with variable duty cycle in Verilog
 PWM_Generator_Verilog PWM_Generator_Unit(
  .clk(clk),
  .increase_duty(increase_duty),
  .decrease_duty(decrease_duty),
  .PWM_OUT(PWM_OUT)
 );
 // Create 100Mhz clock
 initial begin
 clk = 0;
 forever #5 clk = ~clk;
 end
 initial begin
  increase_duty = 0;
  decrease_duty = 0;
  #100;
    increase_duty = 1;
  #100;// increase duty cycle by 10%
    increase_duty = 0;
  #100;
    increase_duty = 1;
  #100;// increase duty cycle by 10%
    increase_duty = 0;
  #100;
    increase_duty = 1;
```

```
  #100;// increase duty cycle by 10%
    increase_duty = 0;
  #100;
    decrease_duty = 1;
  #100;//decrease duty cycle by 10%
    decrease_duty = 0;
  #100;
    decrease_duty = 1;
  #100;//decrease duty cycle by 10%
    decrease_duty = 0;
  #100;
    decrease_duty = 1;
  #100;//decrease duty cycle by 10%
    decrease_duty = 0;
 end
 initial begin
        $dumpfile("pwm.vcd");
        $dumpvars;
    end
endmodule

pwm.pcf files
set_io clk R9
set_io increase_duty  B7
set_io decrease_duty  B6
set_io PWM_OUT A5
```