VIET NAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

ASSIGNMENT :PULSE WIDTH MODULATION

Course: *Logic Design with HDL (CO1026)* Lecturer: *Tran Hoang Quoc Bao*

Class: *L01*-Group: *03*

Submission date: *May 6, 2022*

Group's members:

- *Nguyen Thanh Dat - 2012938*
- *Lam Minh Vinh - 2120082*
- *Vo Hoai Nam - 2013834*
- *Nguyen Duc Tuan - 2014949*

Contents

1 Introduction

1.1 What is PWM generator?

- Pulse-width modulation (PWM), or pulse-duration modulation (PDM), is a method of reducing the average power delivered by an electrical signal, by effectively chopping it up into discrete parts.
- The average value of voltage (and current) fed to the load is controlled by turning the switch between supply and load on and off at a fast rate. The longer the switch is on compared to the off periods, the higher the total power supplied to the load.
- **Applications**: Pulse-width modulation (PWM) is a powerful technique for controlling analog circuits with a microcontroller's digital outputs. PWM is used in many applications, ranging from communications to power control and conversion.

1.2 Duty cycle:

- The duty cycle of the PWM signal refers to the ratio of the time that the signal is in a high(on) state over the total time it takes to complete one cycle. It is commonly expressed as a percentage or a ratio.
- A 50% duty cycle means that the high state takes half of the time and the low state takes the other half of the time, this is the same as an ideal square wave.
- If this ratio is greater than 50%, the logic high signal takes a longer time than logic low, vice versa. Thus, a 100% duty cycle means the signal is always on(full-scale), and the 0% duty cycle means the signal is always off(grounding)
- By controlling the Duty cycle from 0% to 100% we can control the "on time" of PWM signal and thus the width of signal. Since we can modulate the width of the pulse, it got its iconic name "Pulse width Modulation".

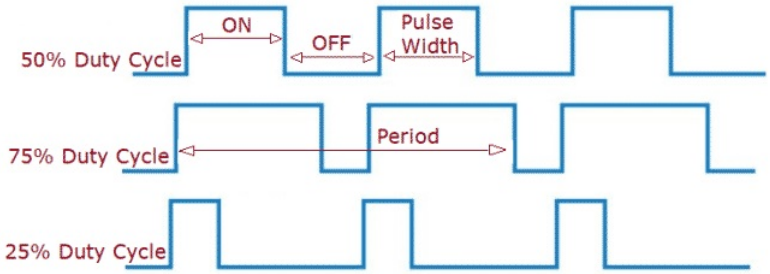


Figure 1: Duty cycle

2 Design

2.1 Tools and simulator to execute this assignment:

- Hardware specification language : Verilog HDL.
- Simulator execution : Xilinx Vivado.
- Tools kits development board:FPGA Arty Z7-20.

2.2 Functionality

By these theory we mentioned above, there are several modules that we implemented throughout this assignment.To be more specific,we emphasize the primary function of this implementation is duty-cycle-modifying on the modules no matter what value we wish.
Basically,there will be 2 primary function that is increasing and decreasing the duty cycle.
With these function above,the system will display in FPGA Arty - z7 kit:

- 2 buttons: modify the duty cycle (increase or decrease duty cycle by 5%)
- 1 RGB LED: for additional feature
- 1 LED: With regards to the signal on FPGA, the brighter this LED become, the higher duty cycle value is.

2.3 Block Diagram:

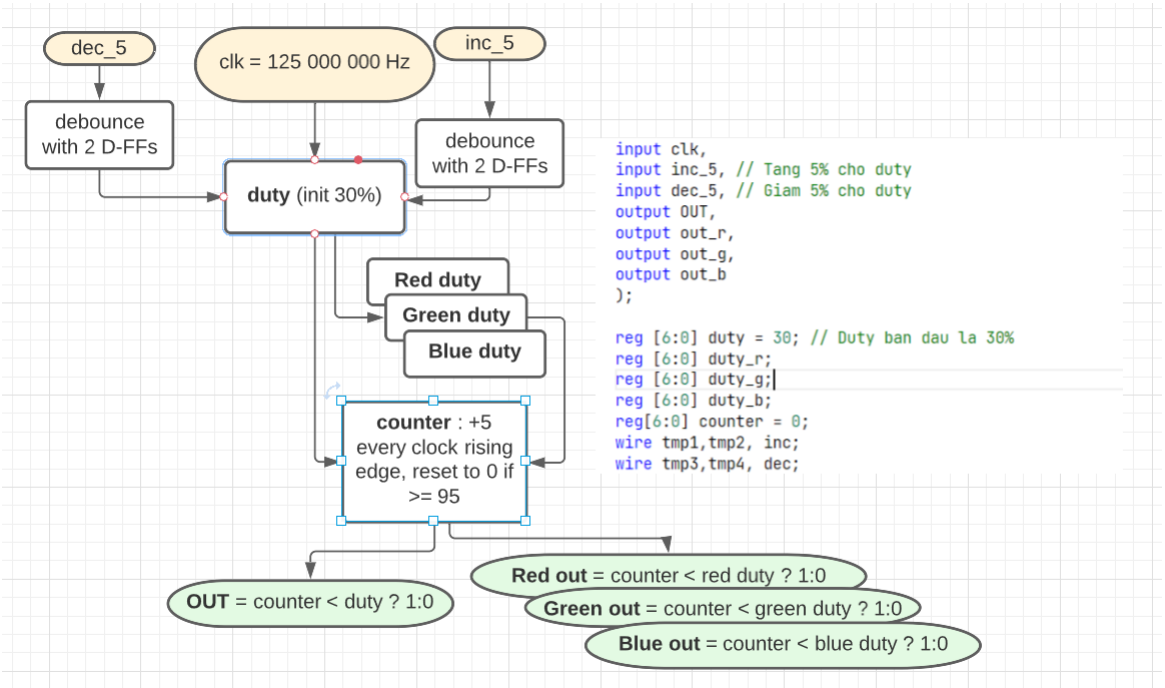


Figure 2: Main design

3 Implementation

Source code: provided in PWM.v and PWM_tb.v . This is a more detailed explanation of it:

3.1 Debounce for each input button

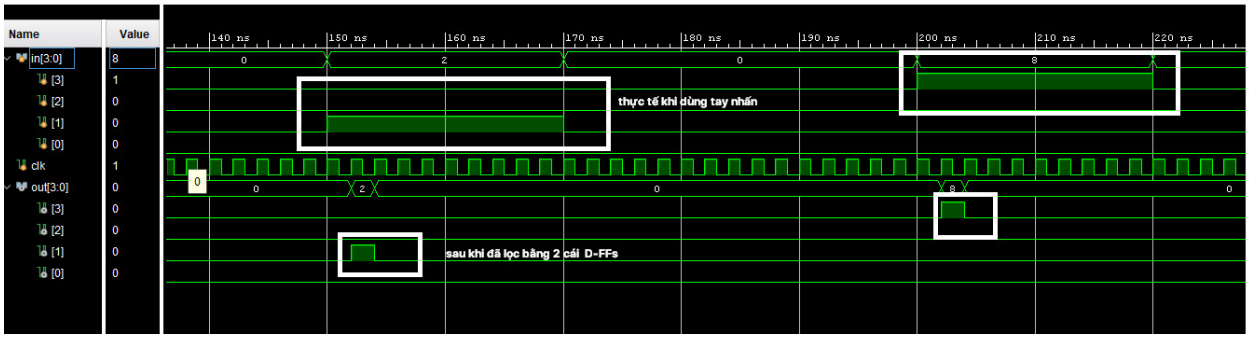
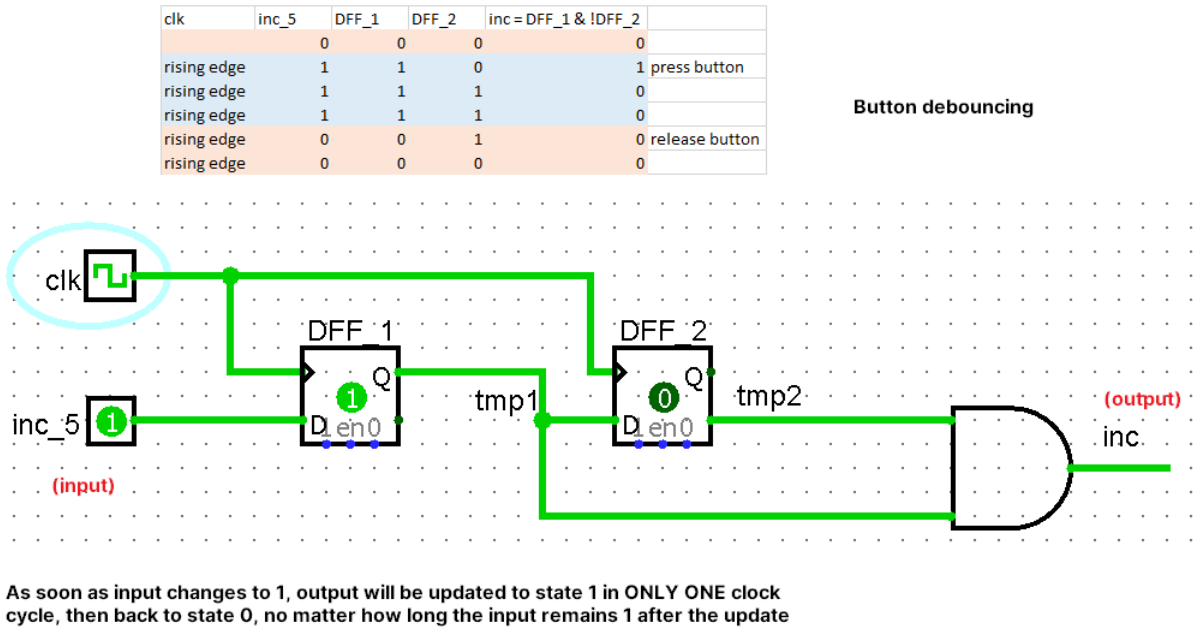


Figure 3: Before and after debouncing

- Because at every rising edge of the clock signal, if the button (for increase and decrease duty cycle) is pressed, the module would registers the input signal and updates the duty cycle accordingly with a rate of 125 000 000 times per second! (Arty Z7 clock frequency). Our goal is to have the module recognizes input only once every time a button is pressed, no matter how long the user holds it, therefore, logic-high input time has to be as short as one clock cycle, this technique is called "debouncing". In reality, the time we press the button is certainly longer than one clock cycle, we designed the following schematic for this problem:



clk	inc_5	DFF_1	DFF_2	inc = DFF_1 & !DFF_2	
	0	0	0	0	
rising edge	1	1	0	1	press button
rising edge	1	1	1	0	
rising edge	1	1	1	0	
rising edge	0	0	1	0	release button
rising edge	0	0	0	0	

Figure 4: Debouncing for button inc_5

- Using 2 D-Flip flops for each button, the input-HIGH signal will be recognized at the nearest rising edge of the clock, then be ignored from the next rising edge.
- We wrote a separate module for the flip flop in the design code.

3.2 Main module

- As in the main design block diagram, our PWM module in Verilog has the following ports:
  - Input:
    - clk: clock signal
    - inc_5: increase duty cycle by 5%
    - dec_5: decrease duty cycle by 5%
  - Output:
    - OUT: output signal with the desired duty cycle
    - out_r (additional): output signal for the color red on RGB LED
    - out_g (additional): output signal for the color green on RGB LED
    - out_b (additional): output signal for the color blue on RGB LED
- Beside, we also have some wires, and one counter register for counting clock's rising edge.
- How the module operate - at every clock rising edge:
  - Count from 0% to 100% (5% increment) then reset to 0 and count again.
  - If count value is less than duty value, output set to high, otherwise set to low.
  - Debounce inc_5 and dec_5 buttons, and update duty value.

3.3 Additional feature: Multi-color display on RGB LED

- Associate each OUT signal duty cycle with an unique color on the light spectrum (20 states), each color has a different brightness value for R-G-B channel, map that brightness value (0 to 255) to duty value (0 to 100) for out_r, out_g, out_b.
- Generate PWM output signal for each color channel the same way as main OUT signal above.

duty	0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80	85	90	95	100
out_r duty	0	100	100	100	100	75	40	0	0	0	0	0	0	0	40	75	100	100	100	100	100
out_g duty	0	0	40	75	100	100	100	100	100	100	100	75	40	0	0	0	0	0	0	0	100
out_b duty	0	0	0	0	0	0	0	0	40	75	100	100	100	100	100	100	100	75	40	0	100
R	0	225	225	225	225	169	90	0	0	0	0	0	0	0	90	169	225	225	225	225	225
G	0	0	90	169	225	225	225	225	225	225	225	169	90	0	0	0	0	0	0	0	225
B	0	0	0	0	0	0	0	0	90	169	225	225	225	225	225	225	225	169	90	0	225
(color)																					

Figure 5: Additional feature: color data

Note: The additional feature was demonstrated directly in the date of presentation with PFGA tool kit Arty Z7.
The result in FPGA approximately satisfied our expectation.

4 Results

This is the waveform generated after we have run simulation from our testbench

- *OUT* signal: 30% initially, then inc_5 pressed 4 times to make it to 50%, pressing it another 8 times raised the duty to 90%.

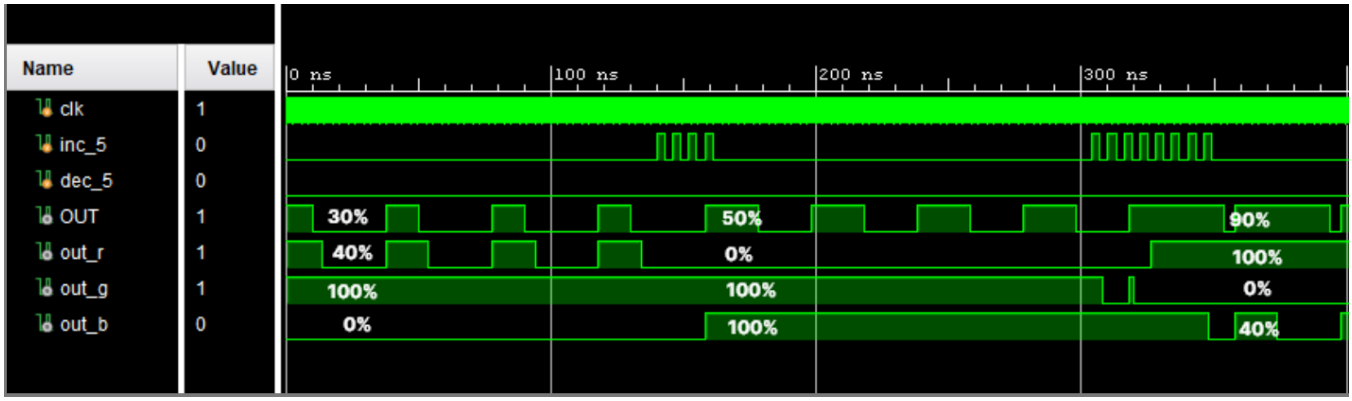


Figure 6: Increasing waveform

- For dec_5, press 5 times (90% to 65%), then press 9 times (65% to 20%)

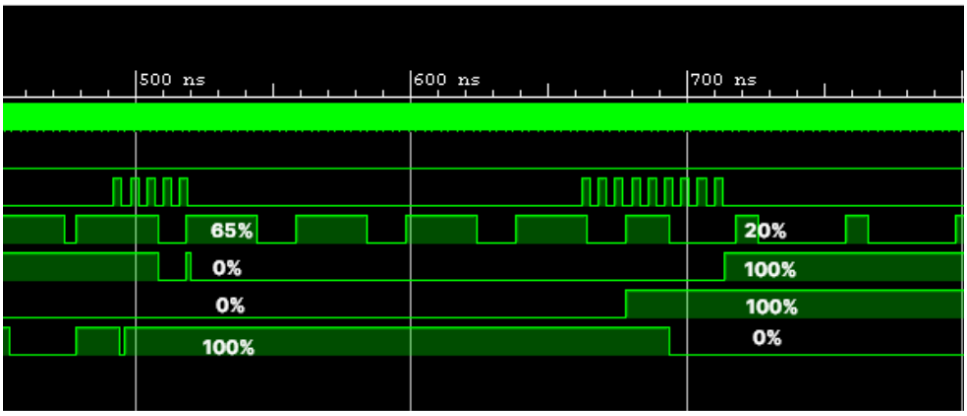


Figure 7: Decreasing waveform

- out_r, out_g, out_b: referencing to additional feature design data, this worked properly. We have fully waveform.

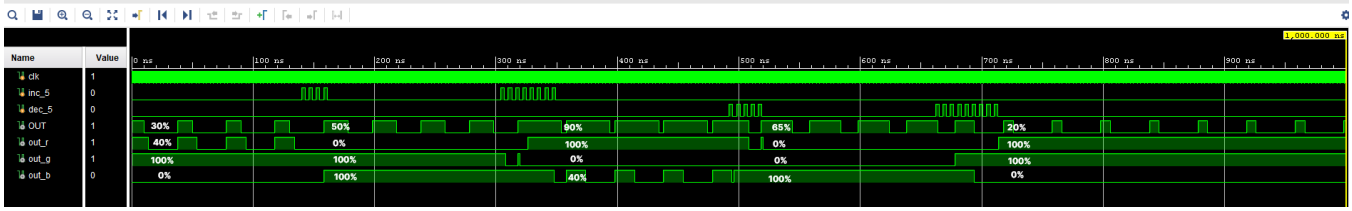


Figure 8: Waveform

5 Conclusion

- Thanks to this assignment, we have been motivated to review some primary techniques and elements in the logic design such as button debouncing, flip-flop, memory and connections, behavioral style in VerilogHDL. Besides, nearly all exercises in Lab 3 have played a key role in the current models in this assignment such as posegde -negedge detection generator and clock divider.
- The result of the implementation considerably satisfies our expectation.20 levels of brightness were demonstrated on a single LED, corresponding to 2 buttons that control the increment or decrement of the duty cycle by any value we wish.
- However, the RGB LED on the Arty Z7 board didn't show up all 20 colors correctly as in the additional feature design, it is obvious that the issue was not in our estimation with executing the colored design steps. A little more careful adjustment in duty cycle parameters for each color channel would fix it, this is going to be a significant task for our next update in the future and other applications.
- With regards to the process of fulfilling this final project, due to limited time and lack of experience, it is no doubt that mistake existence is inevitable. Therefore, your comment and consideration to improve this project absolutely are our expectation.

6 References:

- Digital Design With an Introduction to the Verilog HDL
- Fundamentals of digital logic with verilog design
- https://www.fpga4student.com/2017/08/verilog-code-for-pwm-generator.html
- Lecture slide compiled by M.Tran Thanh Binh.