# ECED2200 – Lab #6

#### STUDENT NAME(s):

STUDENT NUMBER(s)<u>: B00</u>

## **Pre-Lab Information**

It is recommended that you read this entire lab ahead of time. Doing so will save you considerable time during the lab.

There is also several videos showing the use of the software tools – you will save considerable time by watching those now. See <u>http://www.colinoflynn.com/teaching</u> and look at the 'lab 6' information.

# **Overall Objective**

In this lab you will learn about an application of counters: making a Pulse Width Modulator (PWM). This technique is used to adjust brightness of displays (e.g.: LCD backlight), motor speeds, and more.

## Background

Pulse Width Modulation is used to adjust the amount of time a signal is 'high'. We call this the 'duty cycle' of the pulse. The following figure shows a duty cycle of 50%, 75%, and 25% respectively. Note how for 75% duty cycle the signal is high 75% of the time:



If we repeat this signal fast enough, it can be 'averaged' into some analog value. For example if we connect such a PWM signal to an LED, and the pulse themselves come fast enough, our eye will just see varying values of an analog signal.



This work by <u>Colin O'Flynn</u> is licensed under a <u>Creative Commons Attribution-ShareAlike</u> <u>3.0 Unported License</u>. We will be constructing a 4-bit PWM signal. We can construct a PWM by the following method:



The 4-bit counter continuously counts up. When the value of the constant is greater than the current count value, it means the output of the comparator will be 'low'. The following shows the count sequence, constant, and output for a single round:

4-bit Counter	<b>Constant Value</b>	<b>Comparator Output</b>
0	4	0
1	4	0
2	4	0
3	4	0
4	4	0
5	4	1
6	4	1
7	4	1
8	4	1
9	4	1
10	4	1
11	4	1
12	4	1
13	4	1
14	4	1
15	4	1

You can see how adjusting the constant value will adjust the length of time the value is high for. You will be implementing two versions of the PWM counter – one you control this output high time, and one it is automatically adjusted.

## Deliverables

You will work in groups of 2 for this lab. Each group must deliver a lab report; the format for it is described separately.



This work by <u>Colin O'Flynn</u> is licensed under a <u>Creative Commons Attribution-ShareAlike</u> <u>3.0 Unported License</u>.

# **Required Materials**

- Computer with Xilinx ISE 13.2 Webpack installed.
  - All computers in the lab have this installed.
  - This is free software so you can install on your own computer if you wish, you can download it from <a href="http://www.xilinx.com/support/download/index.htm">http://www.xilinx.com/support/download/index.htm</a> select '13.2' on the side. The file is very large so you may wish to download at school, and you are required to register to license it.
- Example project file DigitalTrainer\_Simple.ZIP
  - This file contains an environment which is already setup for your lab.
- Digital Trainer Board

## Procedure

#### <u>Part 1-A</u>

- 1. Download DigitalTrainer\_Simple.zip from one of these locations:
  - a. <u>www.colinoflynn.com/teaching</u> under the ECED2200 lab #6.
- 2. Unzip this somewhere.
- 3. Open the DigitalTrainer\_Simple.xise
- 4. Open io\_connections.sch
- 5. Draw the following circuit. The CB4CE part is found under 'counters', and the 'COMPM4' part is found under 'Comparators'. **Be sure to place a CB4CE and NOT a CD4CE**.







This work by <u>Colin O'Flynn</u> is licensed under a <u>Creative Commons Attribution-ShareAlike</u> <u>3.0 Unported License</u>. 6. As before, save and close the schematic, then double-click the 'Implement Design' process:



- 7. You should get green Check-marks beside 'Generate Programming File':
  - Implement Design
    Synthesize XST
    Translate
    Fit
    Generate Programming File
    Configure Target Device
- 8. Plug in your Digital Explorer board.
- 9. Run 'program.bat' as before, and again you should see an indication it is successful:



10. Adjust the switch includes, and note what is happening to the LED brightness. Complete observations part 1-A.

#### <u>PART 1-B</u>

11. Modify the circuit by placing another CB4CE part down, and also using both the 'Greater Than' (GT) and 'Less Than' (LT) outputs:





Note how the 25KHz clock is driving the PWM counter still, but another counter clocked at 1 Hz will be driving the PWM comparison value. In addition two PWM outputs are generated.

- 12. Repeat steps 6-9.
- 13. Complete Part 1-B of the observations.

#### **Observations**

#### Part 1-A: Manual PWM

**Q1)** Fill in the following table. Note that the brightness of the LED is subjective, thus I don't expect everyone to have the same answers for the 50% brightness level.

Setting of Input Switches				LED Brightness (Approximately)
SW1	SW2	SW3	SW4	
				0%
				50%
				100%



**Q2)** What is the output period and frequency of this PWM circuit? Note the period of a signal is defined as the time between each rising edge, which can be seen below for a single period. This does not depend on the 'off' or 'on' time (e.g.: duty cycle).

The input frequency to the 4-bit counter is 25 KHz, which means the input period is  $1/25000 = 40 \ \mu$ S. Thus the 4-bit counter will be incrementing every  $40 \ \mu$ S – based on the number of states in this 4-bit counter, you should be able to deduce the final output period/frequency.

Alternatively, you may use the oscilloscope in the lab to simply measure the LED1 frequency.



**Q3)** Include a screen-shot or printout of your schematic.

#### Part 1-B: Auto PWM

Q1) What is the function of this circuit? How do the two banks of LED brightness vary?

**Q2)** Include a screen-shot or printout of your schematic.

#### Conclusion

In this lab you have learned about how PWM operate, and tested the implementation.

