

EE 4743/6743  
COMPUTER AIDED DESIGN OF DIGITAL SYSTEMS  
LAB9: PULSE WIDTH MODULATION

---

Learning Objectives: This experiment introduces pulse width modulation (PWM) for analog control

- Digital to Analog Conversion – For external interfacing
- PWM – For minimal analog conversion

This lab assumes that you have purchased a Basys development board. You will need to fill out the lab DATA SHEET located at the end of this lab assignment during the performance of the lab. There is NO PRELAB for this assignment.

---

DIGITAL / ANALOG CONVERSION

---

A digital circuit sometimes needs to interface to the outside world. Usually the outside world is analog. So we must be able to convert from one to the other. In a digital circuit, the values of 1 and 0 represent two different voltages. We can say there is a halfway voltage where if our input is above the halfway voltage, is it a 1. If it is below the halfway voltage, then it is a 0. If we want to convert from an analog to a digital circuit, then there are more divisions than just a single halfway voltage. When the input voltage is within a certain range, then it is assigned a digital value of multiple bits. See Figure 1 below for an example of how this conversion is done.

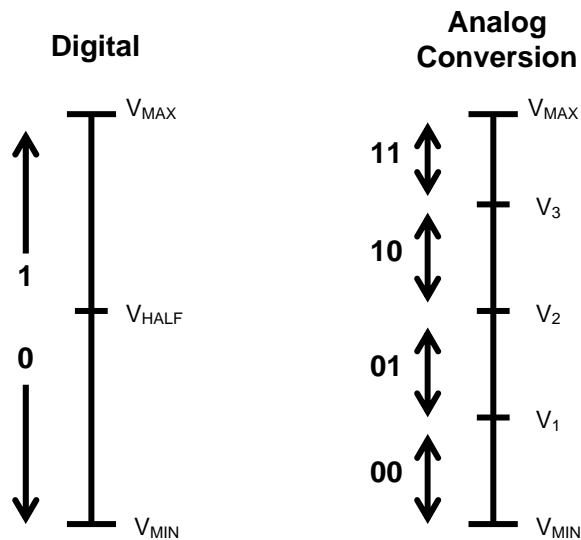


FIGURE 1: DIGITAL VS. ANALOG VOLTAGE REPRESENTATIONS

To convert a digital signal requires simple circuitry, but to convert an analog signal requires a more sophisticated circuit. However, sometimes using smart digital design techniques, this can be performed using purely digital circuitry.

---

PULSE WIDTH MODULATION

---

The output of a digital signal is either a 1 or a 0 – represented as a high and low voltage. If we have a digital clock, it constantly changes from a 1 to a 0 and back to a 1 after a set amount of time. The time from one rising edge to the next is called the clock period. Within each clock period, some of the time the voltage will be high, sometimes it will be low. The ratio of time that it is high vs. low is called the pulse width. For example, a clock which is high half the period and low half the period has a pulse width of 0.5 or 50%.

If we placed a DC multimeter on this clock output, it would read a voltage that is not at the high or low voltage. This is because it is trying to read the *DC voltage*. Since our signal is constantly changing voltages, the DC multimeter will find what the average voltage is. The average will be based on the high and low voltages, but also on the pulse width. For example, if our clock has a pulse width of 50%, then the DC multimeter will read a voltage exactly halfway between the high and low voltages. If the pulse width is 75%, then our clock is high 75% of the clock period. This will produce a DC voltage which is more toward the high voltage.

By varying the pulse width of our digital signal, also known as pulse width modulation (PWM), we can produce an analog DC voltage. However, the frequency of the signal must be high enough to produce this averaging effect. This technique can be applied to circuits that require a DC voltage such as lights or DC motors. A DC motor spins at a rate proportional to the DC voltage applied to it. So using PWM, we can digitally vary the speed of rotation without using a dedicated digital-to-analog converter. This technique can also be applied to LED light like on the Basys circuit board.

---

## IMPLEMENTATION

---

The objective of this lab is to implement an LED fader. This will make the LEDs on the Basys board have a variable brightness.

### GRADE C: IMPLEMENT A PWM SIGNAL GENERATOR

---

For this part of the lab, you must implement a pulse-width modulated signal. The modulating frequency for the signal will be 3.125MHz (period of 320ns), and can have a variable pulse width in steps of 20ns.

In other words, the system clock will be 50MHz, and the modulating frequency will be 16X that. To build this circuit you will need to use a 16-bit rotating shift register. The PWM signal will be one bit of the parallel output. As we clock the register at 50MHz, we should see a 3.125MHz output.

We will want to manually adjust the PWM signal, so set the upper 8-bits of the parallel input to all 0's, and connect the lower 8-bits of the parallel input to the eight switches on the Pegasus board. Then connect the load input to button(1). Connect the clear input to button(0). Select one bit of the parallel output and connect it to one of the LEDs. Compile and download your design.

Set all switches into the upper position and press button(1). The LED should turn on. Set switch(7) to the lower position and press button(1). There should be a slightly change in brightness, but perhaps not enough to notice depending on the ambient lighting in the room. Set switch(6) to the lower position and press button(1). Continue this with each switch and see the change in brightness. Demonstrate this to your TA.

*Note: This circuit will not produce the maximum brightness possible since it can only generate a 50% PWM signal.*

### GRADE B: LOAD THE ON-BOARD FLASH MEMORY WITH THE DESIGN

---

The next part of the lab will be to program the on-board Flash chip with the code from the previous part. One of the processes under "Generate Programming File" is "Generate PROM, ACE, or JTAG File". Execute that process. Create a Xilinx Serial PROM file in the MCS format. The specific flash chip on the Basys board is the **xcf01s** chip. Once you have specific the flash chip, you must add the bit file to the datastream. Once the program says it has successfully formatted the file, exit the program.

Execute the process "Configure Device". There are two devices that appear in the datastream – the FPGA and the Flash chip. This time we will assign the MCS file to the flash chip. Then program the flash chip.

You must reconfigure the board so that it will download the bit file from the flash chip automatically. There are three jumpers located on the board labeled M2, M1, and M0. Connect jumpers across all three of these. This puts the FPGA into master mode. Press the PROG button to reset the device, and it will download the code automatically. Show this operation to your TA. Remember to remove all three jumpers before you try and download from the computer again. It is not necessary, but it reduces the chances of mistakes.

#### GRADE A: CREATE A FADER USING THE PWM CIRCUIT

---

Your next task is to create a circuit which will automatically ramp the brightness of the LEDs up and down through the entire range of brightness. There are sixteen brightness settings using the previous circuit, so create a circuit which will display each of them from the lowest PWM signal (0%) to the highest (100%) and then back down to the lowest within three seconds.

The easiest method would be to use a 32-bit rotating shift register and load all 0's into the lower 16 bits and all 1's into the upper 16 bits. Then every 0.17 seconds, shift all values in the circuit. Then load the lower 16 values into the PWM circuit. In the first cycle the lower bits will contain only a single 1 and fifteen 0's. The next cycle will contain two 1's and fourteen 0's. This will continue until it has sixteen 1's loaded in the lower bits. Then the opposite will begin: first a single 0 and fifteen 1's, etc...

You will need to build a clockgen circuit similar to the ones previously used in other labs. You will need to divide the 50MHz clock by  $2^{21}$ . This will provide a clock that operates at about 24Hz.

You will need to build a 32-bit rotate shift register from two 16-bit shift registers. Construct it so that the lower 16 bits will be loaded with all 0's and the upper 16 bits with all 1's. Be sure to connect the load input to button(0) to start/restart the circuit.

Connect the parallel input of the PWM circuit to the lower 16 bits of the above circuit. The load input to the PWM must only be triggered once every time the 24Hz clock goes high. See the previous lab on how to construct a circuit which will generate a single pulse based on an input which stays high for multiple clock cycles.

Connect the PWM output to **all** LEDs and compile/download your design. The eight LEDs should ramp up and down through the entire range of brightness within three seconds. Show your TA its operation.

**LAB DATA PAGE**

NAME: \_\_\_\_\_

**GRADE C: IMPLEMENT A PWM SIGNAL GENERATOR**

- |  |                |
|--|----------------|
| 1. Program compiles  | (Yes/No) _____ |
| 2. The LED turns on when all switches are high                 | (Yes/No) _____ |
| 3. The LED brightness decreases as switches are changed to low | (Yes/No) _____ |
| 4. Button(0) turns off the LED                                 | (Yes/No) _____ |

Total hours reported (from work log) for this lab portion: \_\_\_\_\_

TA CHECKOFF SIGNATURE: \_\_\_\_\_ (must be legible!)

**GRADE B: LOAD THE ON-BOARD FLASH MEMORY WITH THE DESIGN**

- |   |                |
|---|----------------|
| 1. Program compiles                                   | (Yes/No) _____ |
| 2. Same operation occurs as above circuit             | (Yes/No) _____ |
| 3. The circuit works after the PROG button is pressed | (Yes/No) _____ |

Total hours reported (from work log) for this lab portion: \_\_\_\_\_

TA CHECKOFF SIGNATURE: \_\_\_\_\_ (must be legible!)

**GRADE A: USE TWO PRNG MODULES TO ENCRYPT AND DECRYPT A SERIAL STREAM OF DATA**

- |   |                |
|---|----------------|
| 1. Program compiles   | (Yes/No) _____ |
| 2. The brightness of the LEDs progress through the entire range | (Yes/No) _____ |
| 3. The fade up and down takes about three seconds               | (Yes/No) _____ |
| 4. All eight LEDs are fading up and down                        | (Yes/No) _____ |

Total hours reported (from work log) for this lab portion: \_\_\_\_\_

TA CHECKOFF SIGNATURE: \_\_\_\_\_ (must be legible!)

**Student Evaluation:**

What did you like most about this lab?

What would you change about this lab to make it better (not necessarily easier)?

