University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Sciences

EECS 150 R. Fearing and T. Cheng
Fall 1999

Checkpoint 2
Pulse Width Modulation

1 Objectives

In this checkpoint, you will:

1. get familiar with the protocol to use the LED board,
2. implement a PWM block,
3. and design the playback feature of the project (Mode 1).

2.1 Prelab

All the schematics needed for this lab must be entered in Xilinx Foundation Schematic Editor before coming to lab. Test your PWM and Mode 1 macros in the Logic Simulator before coming to lab. Please create a test script to show your TA that your design works.

2.2 Overview

The goal of this checkpoint is to design the single frame display feature of the project (from the serial port). The algorithm for this is shown below.

```
turn on UART and wait for UART to finish writing one video frame into the SRAM
turn off UART

SEND:
    for ROW = 0 to 9
        for TIME_SLICE = 0 to 31
            send 4-bit ROW to LED Array
            for COL = 0 to 13
                INTENSITY = DATA(ROW, COL)
                VALUE = INTENSITY > TIME_SLICE
                send VALUE to LED Array
            end
            DISPLAYENABLE = 1
        end
    end
    goto SEND
```

The final objective of this lab is a working Mode 1 FSM. **You may choose to follow any design methodology you want and you will not be deducted any points for not following the steps in this checkpoint.** However, you will not receive as much partial credit if you design does not work and you cannot demonstrate partial functionality. Since modular design will save you a lot of time debugging, this checkpoint is designed to lead you through several stages that will culminate in a working Mode 1 block. Also, the intermediate steps will net some partial credit if you can't get your design working.

In this lab, you should design a single FSM that has many functions. You should start by designing the easy functions and add more functionality step-by-step. The functions are listed below. Function A and B are debugging utilities and are not a part of the final project. Since Function C implements Mode 1 of the project, you can modify and use this piece of Checkpoint 2 for your final project.

You should add a TA file, U:\WVLIB\cs150\lab9\lab9pinout.sch. This file provides a mechanism to switch between the 3 functions, a ROM for Function B, and all the pin I/O you need (other than for debugging purposes) for this checkpoint. SW5-7 & SW5-8 control the data source which drives the LED block.

Function	Description	Usefulness	SW5[8:7]
A	Display DIP switches on LEDs	LED board works	1x
B	Display ROM contents on LEDs	PWM works	00
C	integrate UART, display data from the serial port	Mode 1 of the project works	01

3 Simulation

It is very important to create a test script to run simulations. This will save you a lot of time and frustration debugging a problem in your logic. These scripts can be used over and over to verify that you haven't "broken" a previously working design while optimizing old features or adding new ones.

To Do: As part of prelab, you must test your PWM block and your Mode 1 FSM in the Logic Simulator. You need to create a test script, for your PWM block, that will generate pulses of 0%, 50% and 100% duty cycle. So, if you test your PWM block with intensity values of 0, 15, and 31 out of 31 then you should expect the output of the PWM block to be a pulse that is high for 0, 15, and 31 time slices. Testing in simulation has many advantages, so we have created a block that you should use to simulate the functionality of the LED board without actually debugging in hardware. Add the component, LED_BLOCK, from U:\WVLIB\cs150\lab9\LIB\lab9, and connect it to the outputs that are sent to the actual LED board. You need to create another test script that will demonstrate your Mode 1 FSM loading the correct values into the LED board.

4 LED Board

For this part of the lab, you should design and simulate an interface to the LED board. This needs to be done as part of the prelab.

The LED Board schematic is provided on the last page of the project specifications. The board is organized into a 10x14 pixel matrix of LEDs. The board is composed of a shift register, a decoder, 140 LEDs, and drivers for each column and row of the LED matrix. The basic operation of the board is that the 18-bit shift register selects data to display and the display drivers create the potential difference needed to light up the LEDs selected. So, to light the board, your design must serially shift the row address and column bits into the shift register, and then enable the driver chips to actually light up the LEDs. The first 4 bits in the shift register specify which row to display, 0-9 inclusive. The next 14 bits in the shift register specify which columns to display, from the leftmost column to the right. Notice on the schematic, that the 4 row bits are

decoded, mapping those 4 bits to select a single row and the 14 column bits are not decoded, so each column bit maps to a single column. Some examples are shown below:

Bit Pattern in SR – Q[17:0]	Row selected – Q[17:14]	Columns selected – Q[13:0]
0100 10101010101010	4	0,2,4,6,8,10,12
1100 11111111111111	none	0 - 13
0001 00000001111111	1	7,8,9,10,11,12,13

The board has 3 inputs, which are a serial data line (DATAOUT), a glitch-free clock line (CLKOUT), and a display enable line (DISPE.H). DISPE.H switches on the driver chips to display the data you have loaded into the shift register. It is important to turn off the clock to the board when you want to display the shift register data. This is to ensure that the data being displayed is not shifting. **Make sure that your clock output is glitch-free since the shift register on the board does not have a clock enable input.** Glitches on CLKOUT *will* cause data to be shifted erroneously. DO NOT put any logic gates on the clock you send to the board, as this will ensure that you will have glitches. Instead, you can generate a glitch-free clock by using an OFD_1 (negative edge triggered output flip-flop) and OPAD combination.

The target refresh frequency for the LED board is 60 Hz. Since there are 10 rows and 32 time slices per row, our row refresh frequency is 19.2 kHz. So the period of the DISPE.H signal should be ~50 us.

To Do: The LED board has 5 inputs to connect: DISPE.H, CLR, CLKOUT, DATAOUT, Vdd and Vcc. NOTE: **Some of these pins are connected to NUMLED2, so you should not use NUMLED2 for debugging purposes in the future.**

Pin Name	Xilinx Board Pin #
DISPE.H	44
DATAOUT	40
CLKOUT	35

For Function A of the PWM block, your design should send data to the LED board and display the image sent. The column data should be taken from SW5-1 through SW5-7 (switch bank 5). There are 7 switches and 14 LED columns, so map each switch to two columns. You should use a loadable shift register to pipeline your output data (DATAOUT) as that makes it easier to implement functions B and C. If you choose to implement DATAOUT with logic gates rather than a parallel-to-serial shift register, then you will need to make more modifications to make your design work for Function B and C. It's also convenient to use a counter for your state machine since that eliminates NSD and it makes OD easy.

Your FSM should send out 18 bits of data serially, the first 4 bits are the row address, most significant bit first, and the next 14 bits are the column data, from left-most column to right. Your FSM also needs to send out a glitch-free clock signal for the LED board to shift in this data. To minimize setup and hold time violations on the LED board's flip-flops, the CLKOUT and DATAOUT should not change on the same clock edge. After you shift out 18 bits, assert DISPE.H and keep it asserted for ~50us (i.e. design a clock divider and a mechanism to stop your FSM and display the shift register data for 50us). The 4 row address bits should come from a 4-bit counter that you increment and reset appropriately. The LED board has 10 rows, so the row address bits need to count from 0-9. The column bits come from SW5, so just load and enable a 14-bit shift register appropriately.

5 Pulse Width Modulation

For the next part of the lab, you need to design a datapath which implements pulse width modulation. PWM is the algorithm you will use to display grey-scale images on LEDs. Basically, intensity values are mapped to the duration of time LEDs are lit. So brighter pixels correspond to LEDs that are on for a long time and dark pixels correspond to LEDs that are on for a short time. To do this, divide the refresh rate into 32 discrete time slices and apply the following algorithm.

```
if (intensity > time)
        then LED = 1
else LED = 0
```

To Do: The pieces you need to add to your design (if they weren't there) are a column address counter, a comparator, a time slice counter, a 10x14 8-bit wide ROM, and control logic. The column address counter is to address the ROM and it wasn't necessary before since SW5 was the column data. The comparator and time slice counter implement the PWM algorithm displayed above. The ROM simulates a frame of data written in the SRAM chip. The ROM will be provided as part of the TA schematic, U:\WVLIB\cs150\lab9\lab9pinout.sch. You can set the INIT values, of the ROM, to any pattern, but keep in mind that the data bits will be accessed from LSB first, assuming you use an up-counter to address it. Remember to re-run your simulation to see if you have "broken" function A, and of course verify in hardware that the simulation results are correct.

6 Mode 1 FSM

The Mode 1 FSM should read a single frame from the UART, and continuously transmit that frame to the LED board. You need to add the UART from checkpoint 1. The Mode 1 FSM should be a simple design, which receives and writes a single frame of data into the SRAM. Then it should continuously display that frame on the LED board. The only difference between Function B and C is integrating the UART to write the contents of the SRAM. To integrate the UART, you should design a frame reader block that will follow the serial protocol and write data bytes to the SRAM. You should implement the small frame protocol in the project specifications. See Figure 2 on page7 of the project specifications. This frame reader block should be implemented and tested in the Logic Simulator before you attempt to integrate it into the Mode 1 schematic. So, the mode 1 FSM should turn on the frame reader for one frame and then turn the frame reader off and keep the PWM block on until the global reset is asserted.

Name: _____ Name: _____

Lab Section (check one)
M: ❒ AM ❒ PM T: ❒ AM ❒ PM W: ❒ AM ❒ PM Th: ❒ PM

7.1 Prelab Checkoff

1. Schematics for PWM and Mode 1 data and control entered, script files for testing **TA:** _____ (25%)

Reminder: You need to have complete state diagrams for controllers ready for your design review meeting in assigned lab slot this week.

7.2 Checkoff

2. Display image received through serial port (Mode 1) **TA:** _____ (75%)

   Partial credit for partial functionality

8. Turned in on time **TA:** _____ (full credit)

9. Turned in up to one week late **TA:** _____ (half credit)