A Theatre Lighting Board

Maura Cordial and Irene Zhang

Introductory Digital Systems Laboratory

November 3, 2006

Abstract

The goal of this project is to design and implement a lighting board controller. The lighting board will be used to control eight dimmers, each of which can power up to two theater lights. Each dimmer is assigned to a channel that can be programmed at different intensities(0%–100%) levels for each of the 128 cues. Each cue has an up time and down time, up to 60 seconds, indicating how long it takes to bring the channels up to their respective intensities for the cue and back down at the end. There are also some optional parameters for each cue: wait, follow and link. Wait indicates a period of time to wait before bringing up a cue after the go signal. Follow indicates a time period to keep the cue up before continuing to the next cue. Link indicates the next cue to go into. Link can be either sequential or used to create loops. The user will program cues using a standard computer keyboard and screen. The screen will display detailed information for a single cue and basic information for every cue. The board will operate in two modes: blind and live. In live mode, changes to a cue will affect actual channel intensities in real time. In blind mode, the changes will not be visually seen until that cue is in live mode.

Contents

1 Introduction **2**

2 Design **2**
2.1 Module Overview . . . 2
2.2 Keyboard Input (Maura) . . . 3
2.3 Screen Output (Maura) . . . 4
2.4 Cue Memory (Irene) . . . 5
2.5 Processor (Irene) . . . 5
2.5.1 Registers . . . 5
2.5.2 Instruction Memory . . . 6
2.6 DMX Output (Maura) . . . 6

3 Testing **6**
3.1 Processor . . . 6
3.2 Memories . . . 6
3.3 Key Board . . . 6
3.4 Screen Output . . . 7
3.5 DMX Output . . . 7

4 Appendix **7**
4.1 Theatre Lighting Term Definitions . . . 7

List of Figures

1 Module Diagram (black:Modules, cyan:RAM) . . . 2
2 Basic Cue Data . . . 5
3 Extended Cue Data . . . 5
4 Channel Data . . . 5

1 Introduction

This document details the proposed design of a lighting board console. Lighting boards are used mainly in theatre and entertainment settings to program the lighting for an event. A lighting board allows the user to manipulate channel intensities to produce different looks and effects with the lighting instruments. These looks can then be made into a cue and saved in the lighting board's memory. The lighting design for a theatre production is composed of multiple preprogrammed cues that are run during the show from the board's memory. During a show, the lights transition between cue to cue either manually or automatically. Most cues are run manually by pressing the go button, which tells the lighting board to go to the next cue in the sequence. If the cue transition happens automatically, there is no need to press the go button to transition into the next cue.

We will be implementing a lighting board similar to ones used in theatres around the world. The board will be able to control eight channels on two dimmer boxes. These channels will control the lights that will be plugged into the dimmers. The dimmers will be interfaced to the labkit through the use of DMX. DMX the industry standard for communication between dimmers and the lighting board. Our lighting board will be able to record up to 128 cues. Each cue has special parameters such as an up/down time, follow, link, and wait. Our lighting board will use a computer screen and keyboard for programming cues. The keyboard will have special key bindings that will control key programming and functional operations on the channels. The special key binding will allow us to implement all of the functionalities of a real lighting board without the specialized interface on real lighting boards.

2 Design

This section discusses the proposed design of the lighting controller.

2.1 Module Overview

We are using a shared memory, processor-based implementation for the lighting board controller. Most of the functionalities will be handled as instructions in the processor. The modules for the lighting board controller fall into four categories: user interface, processor,cue memory and DMX. The user interface consists of everything the user will interact with such as the modules that interact with the keyboard and screen. The processor module will be responsible for the bulk of the functionalities of the controller, such as loading and saving cues. The cue memory stores the information for each cue. The DMX module will interface with the dimmer boxes to control the theatre lights.

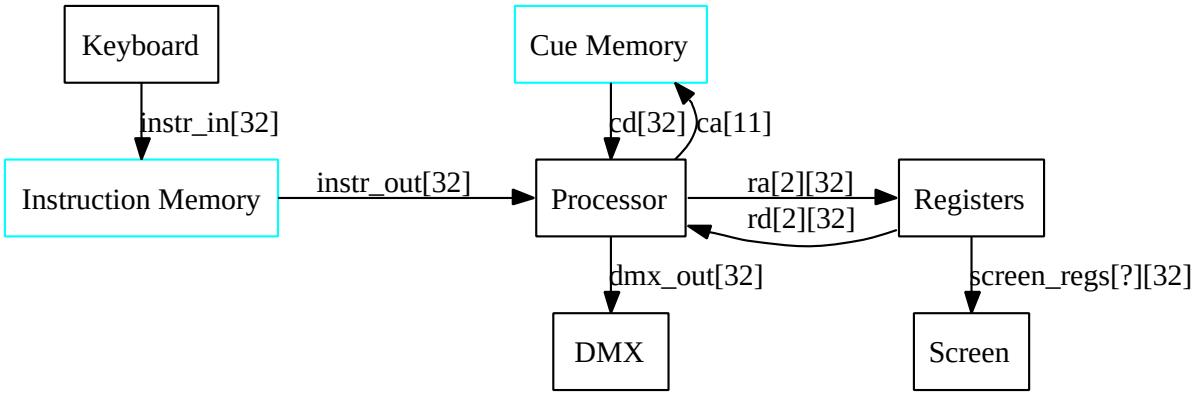

Figure 1: Module Diagram (black:Modules, cyan:RAM)

User Interface The screen module is responsible for drawing the screen interface for editing cues. Some parts of the screen will be constant such as channel labels, while other parts will have to be changed according to data changes such as channel intensities. The keyboard module will be the input side of the user interface. The keyboard module interprets key presses and sends the appropriate instructions to the processor. For example, if the user presses record, the keyboard module will issue a store instruction for each of the channel registers.

Processor The processor will be used to implement most of the functionalities of the controller. The processor will be a stripped down version of a generic processor. The unnecessary functionalities will be taken out. This simplifies the design and allows the processor to run faster. The instruction set will also be a reduced to only the instruction we will be using. Special instructions will be added for some of the functionality such as DMX commands.

Cue Memory The Cue memory will be where the cues for the show are stored. All the information for a single cue will be stored in four 32-bit numbers. The first two numbers hold information about the cue and the second two store channel intensities for the cue.

DMX Interface The DMX interface sends signals to the dimmer in DMX to manipulate the theatre lights. The dimmer takes a binary sequence that encodes the light command. The DMX interface will accept cues and convert them to the format needed to interface with the dimmer

2.2 Keyboard Input (Maura)

Input: ps/2 port
Output: instruction[32]

The keyboard module will be the interfacing unit between the processor and the user input. The keyboard will be connected to the labkit through the PS/2 port. We will create a module that will allow the keyboard to communicate with the labkit. Once the keyboard can communicate with the labkit, the input from the keyboard will be converted into the processor instructions or macros and sent to the instruction buffer. The processor will then execute the instructions in the instruction buffer and any needed macros.

The keyboard will need specialized key bindings to replicate buttons on a real lighting board. The special keys that will be needed:

- Channel – selects a channel from 1 to 8 that you want to edit
- At – allows you to assign an intensity level to a channel
- Go – tells the processor that it is time to move to the next cue
- Enter – this allows you to confirm intermediate steps while programming a cue
- Numbers: 0 through 9 – will be used for channel, cue, and intensity selection
- Arrow Keys – allows you to navigate the screen
- Link – allows you to play back cues out of sequence by linking them together
- Follow – will allow you to playback a series of cues automatically
- Wait – is the delay between when the Go button is pressed and when the actual fading of the cue begins
- Record - will save all the attributes of the cues into ram
- Black Out – will capture all lights and set their intensity immediately to zero (optional)

- Time – allows the user to select the up and down
- Blind – changes to blind mode
- Live – changes to live mode

2.3 Screen Output (Maura)

Input: cue registers

The screen constructed from multiple sprites. These sprites will display relevant information to the user, such as the previous, current, and future cue, channels, intensity, and up/down time. Each of the dynamic sprites will obtain its data from specific reserved registers.

The screen will display all the channels for the current cue, all relevant information for the current cue, as well as the up/down time of the previous and following cues. Centered at the top will be a label displaying the current mode. The channels will be displayed beneath in two rows, with four channels per row. The bottom left corner of the screen will display the current cue and detailed information about the cue. A cue list will be presented in the bottom right section of the screen. This box will contain the basic information about the previous, current, and following cue.

Static Sprites There are certain areas of the screen that will never change, so we have created a section of Static Sprites. These sprites will serve as headings or labels. One set of static sprites will display the channel number labels. Another set of sprites will display the headings for each cue. The headings for the relevant information for each cue are cue number, up time, down time, wait, follow, and link. The cue list box headings will be up time, down time, and cue number and the labels previous, current, and next.

Cue Sprites The Cue Sprites will read the cue data from the registers and will display the appropriate information in the various portions on the screen. The basic cue information will be stored in a 32-bit value containing the cue number, up time, down time, wait, and flags to check to see if the lighting board needs to grab the follow and link information. If the follow and link data is needed, this information is extracted from the next register.

The current cue number, up time, and down time will also be display in the cue list box on the right side of the screen. To complete the cue list, the module will have to keep track of the current cue that is stored in a register, and then ask for the cue data for the previous and next cue and store this information in the appropriate subsections of the list.

Channel Sprites The Channel Sprites will contain the intensity displays for each channel. This data is stored in two 32-bit value channel data structures. The data structure is split into 4 sections, so that each data structure holds the intensity information for four channels. The first data structure will contain information for the top row (the first four channels), thereby leaving the second data structure to store the information for the last four channels.

Cursor Sprite The cursor sprite will be a layering sprite that hides the value in the register with a new temporary value until the new value is stored into the register. The new value is stored into the register by either the push of enter or record button (dependent upon the type of information). This will allow the user to visually see what they are changing and to make sure that they remember to record the information if they so desire.

2.4 Cue Memory (Irene)

Input: address[11],we[1]
Output: dataout[32]

The Cue memory will be a 32x512 RAM. Each cue takes up 4 addresses. The first address will be basic data about the cue such as up and down time. The second line holds extended data such as follow and link. The basic data will contain both a follow and link flag indicating that the cue is using the follow and link values. Thus, data at the second address may not have to be loaded if the flags are checked first.

uptime(8)	downtime(8)	wait(8)	followflag(1)	linkflag(1)	unused(6)

Figure 2: Basic Cue Data

follow(8)	link(7)	unused(17)

Figure 3: Extended Cue Data

The last 2 lines hold the channel intensities. The channel intensities range from 0-100, so only 7 bits are needed for each channel. The 4 channels are evenly spaced over 32-bits.

Channel 1(7)	unused(1)	Channel 2(7)	unused(1)	Channel 3(7)	unused(1)	Channel 4(7)	unused(1)
Channel 5(7)	unused(1)	Channel 6(7)	unused(1)	Channel 7(7)	unused(1)	Channel 8(7)	unused(1)

Figure 4: Channel Data

USB Memory If there is additional time, we would like to be able to save cues on a USB. This would ensure that cues would not have to be reprogrammed every time the FPGAs are reprogrammed.

2.5 Processor (Irene)

Input: instruction[32]
Output: cue address[11], cue data[32], dmx

The processor will be a reduced version of the unpipelined 6.004 Beta. The multiplier will be removed to increase speed and branch instructions will not be implemented. There will be a jump instruction for implementing macros. Many of the controller's functionalities are a small set of repetitive instructions, such as load cue and record cue. These will be encoded into the instruction memory as macros. The processor will still have an ALU, a control logic module and a program counter. These will be modified to fit the needs of the lighting controller.

2.5.1 Registers

Input: ra[5],rb[5],rc[5],ra2sel[5],wsel,wd,werf[1]
Output: rd1[32],rd2[32],channels[2][32]

There will be 32 32-bit registers for the processor to use. Some registers are reserved for specific purposes, while others will serve as general purpose memory for the processor. The special registers include:

- R0 - constant 0
- R1 - current cue address
- R2 - current cue channels 1-4

- R3 - current cue channels 5-8
- R30 - return address

2.5.2 Instruction Memory

Input: PC[8]
Output: Instruction[32]

The instruction memory will be a 32x256 RAM. Some of the memory will be shared with the keyboard module so the keyboard can call instructions to respond to user commands. This section will act like an FIFO instruction buffer. Another section of the memory will be preprogrammed with macros for sets of frequently used instructions, which also can be called by the keyboard module. The processor will include a program counter to keep track of the current instruction address.

2.6 DMX Output (Maura)

Input: Address
Output: DMX 512

The lighting board will communicate with the dimmers through the use of DMX. The DMX protocol will transmit 512-bits of information into the dimmer to tell the dimmer what intensity the dimmer should set the light. The DMX converter module will have a 32-bit input from the processor and will output DMX 512 to the dimmer.

Since our labkit does not have a DMX connection (5-pin input), we will have to create a converter that will allow the labkit to communicate with the dimmer through DMX, this will be the DMX converter module. There are many computer converters available that already does this, though they can be a bit pricey. We are waiting for the DMX standard to arrive to get more details about how to actual implement the interface between the dimmer and the labkit. This might be an item that we buy (the prices range from $30-50). In either case, we will have to buy some parts depending on how we actually implement the interface, but we are waiting to consult the DMX standard before finalizing a design.

3 Testing

3.1 Processor

Module testing for the processor will probably involve writing a test rig that runs every instruction in the instruction set. For each instruction, the control logic needs to be checked to ensure that the correct data is being routed to each part of the processor. To test the processor on the labkit, the instruction memory can be filled with test instructions and the output can be fed to the logic analyzer. Once the module is integrated with the rest of the controller we can use keyboard to issue commands and check that the dimmers and the screen are responding correctly.

3.2 Memories

The various memories can be test to make sure they are reading and writing correctly. No additional tests will be needed.

3.3 Key Board

The keyboard will be tested in a few sections. We have to test to make sure that the module is interfaced with the labkit properly. For testing purposes, if the keyboard and the labkit are properly connected, an LED will turn on when a key is hit. This will be tested by making sure that we can read data from the keyboard into the labkit. Within the functionality of the keyboard, the first test will to make sure that the

module is registering that a key has been entered. Once a key stroke has been registered, the next step is to check the functionality of that key binding assignment and that the proper instruction is issued. Once the two line up, the keyboard will function properly.

3.4 Screen Output

Each dynamic sprite will be tested in a very similar fashion, but the static sprite is different. It is fairly obvious if the sprites are working or not, since they are visual. I would go about testing the static sprites by making sure that they are given the appropriate default data. I would then load the load the visual and start debugging from what was wrong visually.

The dynamic sprites need a bit more testing implemented. In each case, I will have to make sure that the appropriate data is being passed and then displayed, since they collect and then display information from the cue memory and registers. I can go about testing to make sure that the right data is being passed by using the hex display. I can break down the information into segments and see where the data passing falls apart.

3.5 DMX Output

We are not sure how to go about testing the DMX Output in smaller blocks since we have not finished this design. The best way to test the DMX module will probably be to look at the lights to check functionalities.

4 Appendix

4.1 Theatre Lighting Term Definitions

- blind mode – when the light board is in this mode, you can edit future cues without altering the current light intensities (i.e. the current cue)
- channel – a label for a dimmer so that the light intensity can be set; in our design the dimmer and the channel are synonymous
- cue – specific combination of light intensities for one stage scene
- dimmer – the power source for stage lights that controls the intensity of up to two lights; controlled by DMX
- dimmer box – the physical box for powering theatre lights; A dimmer box contains 4 dimmers
- down time – the amount of time it takes to fade a cue down
- follow – a way to have the next cue follow the present cue after a certain period of time
- go – signal to the lighting board to transition to the next cue
- intensity – a percentage of the maximum light brightness
- link – a way to link cues non–sequentially
- live mode – when the lighting board is in this mode, any changes made to the current cue will affect the current light intensities
- up time – the amount of time it takes to fade a cue up to the final level
- wait – the amount of time the lighting board waits to load the cue after the go signal.