Lab 3
Asynchronous Memory and Decoder

Due: Before your lab section April 24 or April 25

Introduction

In this lab you will accomplish several goals:

- Implement a decoder based on a memory map
- Interface to asynchronous memory
- Create a system to process read and write commands
- ~~Document your system design using a schematic diagram~~

All lab assignments must be done **individually**.

Requirements

This lab has you combine your boards and implement the required functionality to produce a cohesive system. Your system will perform the following operations:

1. read a command and address from switches connected to the CPLD
2. read a data field from switches connected to the CPLD, if needed
3. perform the indicated operation
4. display a summary to the LCD

Your system operates by reading an address and operation from the switches on the CPLD board when the user presses the User Button on the ARM board. The switches indicate if the operation is a read or a write and the address for the intended operation. If the operation is a write, then a second read of the switches provides the data to write to the indicated device. Again, the switches are read when the user presses the User Button. Given the required inputs, the ARM board issues the command to the CPLD board, which performs the operation. The ARM board then begins again by waiting for the user to press the User Button and start a new transaction.

The user may perform a read or write to the following CPLD peripherals: the bar LEDs, the CPLD board LEDs, the switches, and each of the memory chips. Each of the CPLD peripherals must use shared data and address buses, so you will need to create a decoder to determine which devices are active for each address. An address map and operation details are provided below.

CPLD Board

Your CPLD must interface to the ARM board over the SPI bus and act as a bus master to the shared buses within the CPLD. This includes creating a module for each device on the internal CPLD buses and an address decoder. The CPLD system is shown in Figure 2.

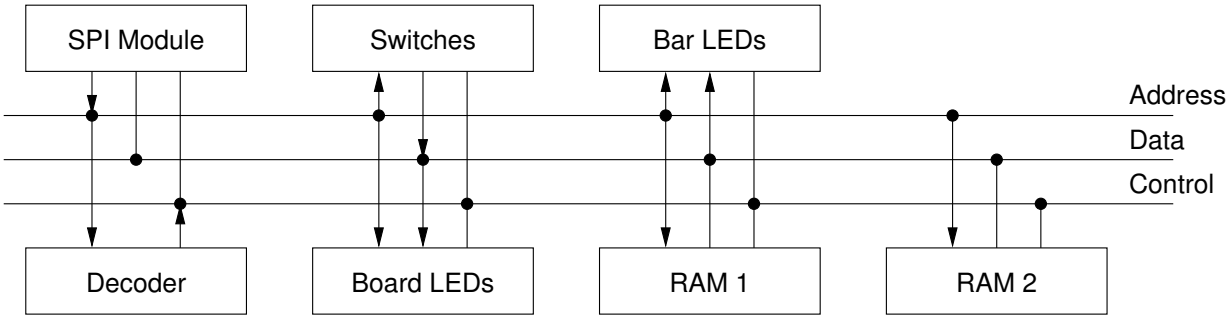

Figure 1: CPLD System Diagram

Your decoder takes the address as input and produces chip select (or chip enable) lines for each peripheral. The memory map for your system is as follows:

Address Range	Peripheral
0x74	Switches
0x6C	Bar LEDs
0x50–0x5F	RAM 2
0x2F	CPLD Board LEDs
0x00–0x0F	RAM 1

Some details about each peripheral:

RAM 1/2 The memory chips you have within your kit. Only use the first 16 entries of each memory chip.

Board LEDs The LEDs built into the CPLD board. Writes change which LEDs are lit and reads return the currently lit pattern.

Bar LEDs The separate bar LEDs that you wired for Lab 2. Writes change which LEDs are lit and reads return the currently lit pattern.

Switches The input switches you wired for Lab 2. Writes are ignored and reads return the state of the switches.

ARM Board

The ARM board initializes all the operations and orchestrates the use of peripherals. The program running in the ARM board repeatedly cycles through asking for operations and then performing those operations based on button presses. Your program performs the operations outlined at the start of the lab. Between each step, the program pauses and waits for the user to press the User Button. At the end of each operation, your program should display a summary of what happened on the LCD and pause for approximately one second.

The first step is to retrieve the switch values to determine the operation and address. The most significant switch indicates a read (switch off) or a write (switch on) and the remaining switches provide a 7-bit address.

If the operation is a write, the program next performs another read of the switches to get the data to write. To provide the user time to change the switches, the program waits until the user presses the User Button to read the data. If the operation is a read, this step is skipped.

Finally, the program performs the operation and displays a summary on the LCD. For example, if the operation is a read to address 0x2F that returns 0xC5, the LCD would indicate `2F R C5` and if the operation is to write the value 0x51 to address 0x6C, the LCD would indicate `6C W 51`.

Schematic Diagram

Schematic Diagram dropped from Lab 3.

~~To document your system, create a schematic diagram of your system. Your schematic should indicate all the major components and the pins selected. The documentation created for Lab 2 will help you when creating the schematic diagram.~~

Demonstrations

Each week you will have small demonstrations leading to a full demonstration of your system during the last week of Lab 3. The demonstrations and deadlines are as follows:

April 3/4 Wire wrap at least one memory chip to the CPLD and implement the LED and switch blocks.
April 10/11 Wire wrap both memory chips and implement memory reads.
April 17/18 Implement memory writes.
April 24/25 Full system demonstration.

Results

Your work for this lab will be evaluated based on:

- ~~Schematic diagram of your system~~
- Source code (C, assembly, and Verilog) that you submit through Learn
- Demonstration of your working system to your lab TA on the due date

Grading

Your lab will be graded based on:

- ~~20 points: Schematic Diagram~~
- 60 points: Demonstrations
- 20 points: Code quality and neatness

Hints

- Design and test your system in steps.
- Test for a button press and then wait for the button to be released before continuing. This prevents a single button press from being detected as multiple presses.
- Example module definitions are listed below, but these are not the required module definitions. You may also want to use the chip oscillator.

```
module led_ctl(input read_n, write_n, reset_n, ce_n, inout [7:0] data,
    output reg [7:0] leds);
module switch_ctl(input read_n, ce_n, output reg [7:0] data, input [7:0] switches);
module spi_ctl(input nss, mosi, sck, output miso, input reset_n,
    output reg [6:0] address_bus, inout [7:0] data_bus, output reg read_n,
    output reg write_n);
module mem_ctl(input read_n, write_n, ce_n, output ceh_n, ce2, we_n, oe_n);
module decoder(input [6:0] address, output reg bar_led_ce_n, board_led_ce_n,
    switch_ce_n, ram1_ce_n, ram2_ce_n);
module top_module(input sck, nss, mosi, reset_n, output miso,
    output [16:0] mem_address, inout [7:0] mem_data, output mem1_ce2, mem1_ce_n,
    mem1_read_n, mem1_write_n, mem2_ce2, mem2_ce_n, mem2_read_n, mem2_write_n,
    output[7:0] board_leds, bar_leds, input [7:0] switches);
```

- A system schematic is below.

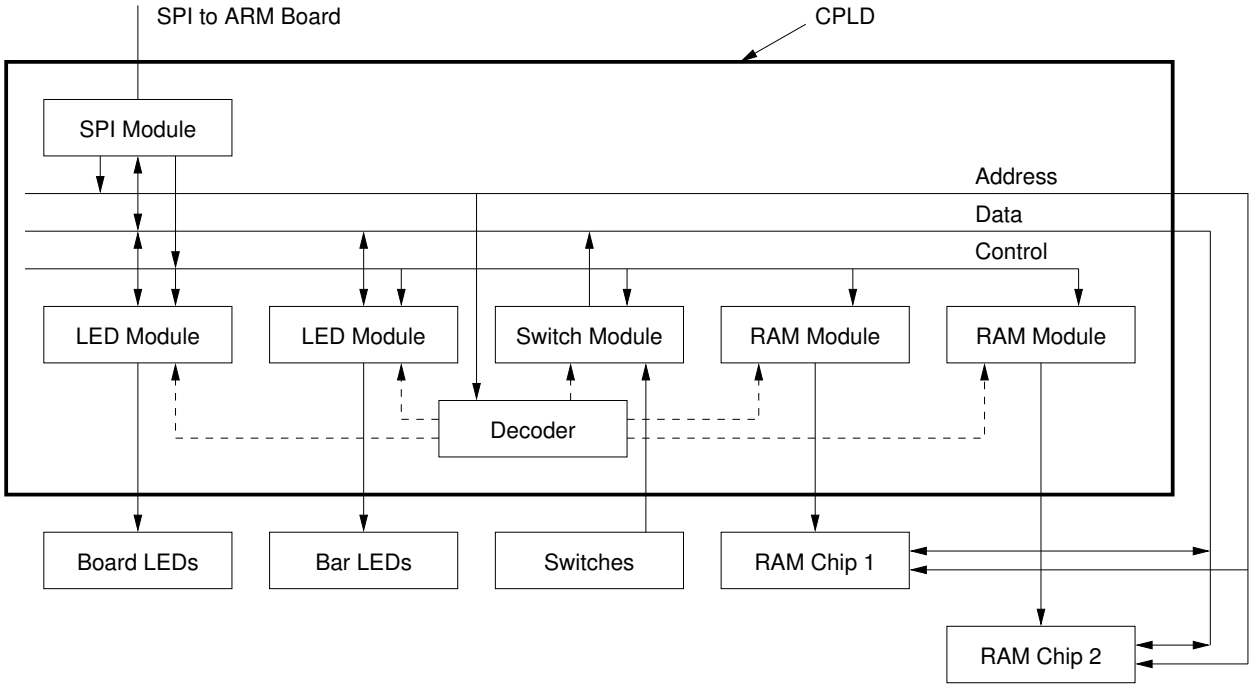

Figure 2: CPLD Schematic