Lab 2
Serial Bus and Basic I/O

Due: Before your lab section March 27 or March 28

Introduction

In this lab you will accomplish several goals:

- Interface your boards using a serial bus
- Explore the use of wire wrapping for system development
- Implement a serial bus interface in Verilog
- Read and display processor flags (condition codes)
- Create basic I/O using switches and LEDs

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 two four-bit signed numbers from switches connected to the CPLD
2. transfer the data to the ARM board across an SPI bus
3. perform computation based on the user button (add if not pressed, subtract if pressed)
4. display the numerical result and process flags on the LCD
5. return the result to the CPLD
6. display the result (flags and value) on LEDs connected to the CPLD

Example inputs and results are below.

	CPLD Board		ARM Board	
	Switches	**LEDs**	**Button**	**LCD**
Example 1	1010 0011	■□■■□■	Not Pressed	`N1V0-3`
Example 2	1100 0110	□■□■■□	Pressed	`N0V1+6`

CPLD Board

The CPLD is responsible for interfacing to the switches and LEDs along with communicating with the ARM board over the SPI bus. The CPLD board requirements are:

- Read the eight switches as two four-bit signed binary numbers. When performing subtraction, subtract the right number from the left number.
- Transfer the two values over the SPI bus in a single transaction.
- Accept the data returned from the ARM board in a single transaction.
- Display the result and flags on the external LEDs. You may separate the flags and result for readability. Do not use the LEDs on the CPLD board.
- The reset button on the ARM board should also reset the CPLD board.

ARM Board

The ARM board calculates the numerical result of an operation selected by the user and returns the result to the CPLD board. Additionally, the ARM board displays the results on the LCD. The ARM board requirements are:

- Accept the operands from the CPLD board over the SPI bus in a single transaction.
- Add the operands if the User Button on the ARM board is not pressed and subtract the operands (left number on the switches minus right number on the switches) if the User Button is pressed.
- Determine if the operation resulted in Overflow (V) and if the result was negative (N) by looking at the Process Status Register (PSR). You must get the flag values from the PSR.
- Performing the computation and checking the PSR must be done in assembly.
- Display the flags and result on the LCD. The LCD should clearly indicate the flag values and display the numerical result in decimal. The previous examples provide a suggested format.
- Return the four-bit result (even if incorrect) and the flags to the CPLD board over the SPI bus in a single transaction.

Documentation

This lab requires you to select a significant portion of the implementation details. Therefore, you need to describe your design choices and document your implementation in a Lab Report in sufficient detail that a colleague could recreate the functionality of your system from identical parts. Your documentation should not be an instruction manual on how to recreate your system, but a description of the system details. For example, you want to describe what the LEDs display, but not how you wire wrapped the board.

Some questions you should answer:

- What information does each LED display?
- What pins did you use on each board? What signal is used by each of those pins?
- What is the format of the data as it travels across the SPI bus?
- What is the format of your LCD screen and what information does it display?

Beyond the questions above, include any other information you feel is necessary. You do not need schematics (that's for the next lab). Submit your report as a PDF with your source files.

Arguments and Return Values for Assembly Procedures

Lab 1 demonstrated how to call an assembly procedure from C code. However, the assembly procedure took no arguments and didn't return any values. Processor registers are used in Embedded Workbench to pass arguments to and return values from a procedure. For example, a procedure that has two integer operands and an integer return value will have the operands in `R0` and `R1` at the start of the procedure and the value in `R0` will be used as the return value. Below is some example code.

A function declared in C as:

```
extern int asm_function(int a, int b);
```

would have an assembly implementation as:

```
asm_function:
    ; At this point a is in R0
    ; and b is in R1

    ; ... Functionality of the procedure

    ; Place the return value in R0

    BX LR
```

Results

Your work for this lab will be evaluated based on:

- Source code (C, assembly, and Verilog) that you submit through Learn
- A Lab Report documenting your design 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: Lab Report
- 60 points: Demonstration
  - 20 points: I/O with switches and LEDs
  - 30 points: Data transfer across SPI bus
  - 10 points: Display on LCD
- 20 points: Code quality and neatness

Hints

- Design and test your system in steps. For example: test your CPLD I/O by driving the LEDs with a pattern determined by the input switches; test your SPI code by temporarily wiring the MOSI pin to the MISO pin; and test your system integration by transferring a value read from the switches to the ARM board and displaying the value on the LCD.
- The SPI source file in the Standard Peripheral Library has instructions on how to use the SPI interface. Additionally, you can view the library help file to view the API documentation.
- The NSS pin on the ARM microcontroller is unavailable for use in this lab, so you will have to select a "software NSS" during SPI configuration.
- Since the system only includes one master and one slave on the SPI bus, the NSS is is not strictly required. However, including an NSS pin can simplify your implementation. Use a GPIO pin as the NSS. Document this in your lab report, if applicable.