Lab 2 (EECE 344)

Jeremiah Mahler

April 3, 2012

Contents

1 Introduction

The purpose of the "device" described in this document is mainly to orient a user with some of the basic facilities of a STM32L Discovery[7] (ARM) board and a Lattice CPLD[3] [1]. It is somewhat of a Rube Goldberg machine for electrical engineers in that it does neat things but it does not do much that is useful.

The goal of this document is to describe the "device" in enough detail such that a person could re-create the system if given an identical parts. This document is not an instruction manual and therefore does not describe each step that would be required. It also does not describe tasks such as how to solder or wire wrap.

The general operation of this device is fairly simple. The ARM board is clocked and contains a cpu so it can do the processing tasks. The CPLD does not use a clock so it can only do logic or sequential tasks. These two devices communicate using SPI with the ARM as the master and the CPLD as the slave.

The CPLD acts as an interface. A bank of switches are used to enter 8-bits of data. This data is sent to the ARM when requested. And any data it receives (during the full duplex SPI transaction) is displayed on to an external bank of LEDs.

The ARM board does more useful computations. The 8-bit data it receives is treated as two 4-bit signed numbers in 2's compliment form. These two numbers can be added or subtracted from each other. A user input button provides the switch, when released they are added, when depressed they are subtracted. Whether this operation resulted in an overflow and whether or not the result is signed must also be accounted for. Then this result is sent to two different places.

The first place the result is sent is back to the CPLD to be displayed on the external LEDs. The second place is on to LCD on board the ARM.

The specific details of this general plan will be described in the remainder of this document.

2 Pin Assignments

One of the critical design specifications of this device are the pin assignments. Table 1 lists the connections that were used. Keep in mind that these assignments are specific to this design and their choice was somewhat arbitrary. Other pins, such as those for the SPI on the ARM (PA5, PA12, PA11), cannot be changed due conflicts with other resources on the board.

Deciphering these pin assignments can be somewhat confusing. On the ARM board there are headers on either side with each pin clearly labeled. On the CPLD the pins correspond to headers (J9, J7, etc) and pins within those headers[3, Pg. 11-14]. The header and pin numbers are printed at the end of each header. The "Mach XO Ball" is used to identify the location on the chip itself. This is used for assigning pins when configuring the CPLD using Diamond[5].

Configuring the CPLD pins using Diamond[5] can be confusing with the bewildering array of options and settings. Luckily the requirements for this device are fairly simple. All pins should be configured for low voltage 3.3 volt CMOS. Output pins can be configured for 8 mA. And no pull up or pull down options are required.

For the pins used with for the SPI that connect between the ARM and the CPLD there are no special requirements. They are both designed to work with 3.3 volt CMOS voltage levels.

For the ground connections each board provides several connection points. In general any connection to ground should work and it is not required to connect to a specific ground pin. For example the ARM board has four pins labeled GND. Any of these should work equally well. It is also a good idea to connect the grounds together so that they all reference the same common ground.

[1] For the remainder of this document the term "CPLD" will be used to refer to the CPLD board and "ARM" will be used to refer to the ARM board

SPI						
Verilog		**ARM**	**CPLD**			
name	description	pin	function	Mach XO Ball	Header	Pin
SCLK	SPI clock	PA5	PT9B	D7	J9	11
SS_L	SPI slave select	PB5	PR4C	F13	J7	1
MOSI	SPI master out slave in	PA12	PR4D	F12	J7	3
MISO	SPI master in slave out	PA11	PR5C	B16	J7	5
	ground	GND			J8	5
switches						
Verilog		**input switches**	**CPLD**			
name	description	pin	function	Mach XO Ball	Header	Pin
in_sw[0]	input switch 8	8	PT2C	B2	J5	1
in_sw[1]	input switch 7	7	PT9A	D8	J5	2
in_sw[2]	input switch 6	6	PT2D	B3	J5	3
in_sw[3]	input switch 5	5	PT9C	E8	J5	4
in_sw[4]	input switch 4	4	PT3A	A2	J5	5
in_sw[5]	input switch 3	3	PT9D	E9	J5	6
in_sw[6]	input switch 2	2	PT3B	A3	J5	7
in_sw[7]	input switch 1	1	PT10A	A10	J5	8
	power, Vdd, pull up				J9	1
	ground				J6	2
LEDs						
Verilog		**output LEDs**	**CPLD**			
name	description	pin	function	Mach XO Ball	Header	Pin
led_ext[0]	output led 8	8	PT15D	B5	J5	23
led_ext[1]	output led 7	7	PT12B	A12	J5	24
led_ext[2]	output led 6	6	PT6E	E7	J5	25
led_ext[3]	output led 5	5	PT12C	B11	J5	26
led_ext[4]	output led 4	4	PT6F	E6	J5	27
led_ext[5]	output led 3	3	PT12D	B12	J5	28
led_ext[6]	output led 2	2	PT16C	A5	J5	29
led_ext[7]	output led 1	1	PT13C	C11	J5	30
	ground				J6	16
reset						
Verilog		**ARM**	**CPLD**			
name	description	pin	function	Mach XO Ball	Header	Pin
rst_l	active low reset	NRST	PL7B	G2	J3	27

Table 1: Definition of the pin assignments between the ARM board, the CPLD, and other devices. Notice that the switch and LEDs are reversed. This was done so that the orientation from LSB to MSB is from right to left.

option	value
MSB	first
CPOL	0
CPHA	0
SS_L	slave select, active low

Table 2: SPI configuration options

The input switches to the CPLD can be interfaced by connecting one end to ground and the other end to the pin along with a pull up resistor to Vdd. A resistor value between 1k and 10k should be acceptable. And the pull up voltage for Vdd can be sourced from a pin on the board (Table 1).

The output LEDs are connected to the CPLD using a series resistor. Vdd would connect to the resistor which connects to the led (forward biased) which connects to the pin. The value of the resistor should limit the current to approximately 10 mA. A value of 300 Ω is a typical value.

To reset the boards their reset pins must be configured. The ARM board provides a `NRST` pin which is at Vcc when enabled and goes low when the reset button is pushed[7, Pg. 17, 20]. This can then be connected to the CPLD to cause it to reset using `GSRN`[2, Pg. 13, 46, 50, 53; 3, Pg. 8]. Table 1 lists the exact pins that were used.

3 Serial Peripheral Interface (SPI)

In order for the ARM and CPLD to communicate using SPI[1, Pg. 278; 6, Pg. 665] they must both have the same options set. Table 2 lists the options that were used.

The MSB option specifies that the most significant bit is sent out first. CPOL specifies the polarity as zero and CPHA specifies the phase as zero (first clock edge). SS_L specifies that slave is selected when this line is low.

The pins used for the SPI on the ARM board were chosen because they did not conflict with any other on board devices. This resulted in pins PA5, PA12 and PA11 with SPI1 for SCK, MOSI, and MISO respectively[7, Pg. 24]

Since this device does not require high speed it was chosen to limit the speeds in an effort to improve reliability. During testing the value of the SPI clock was found to be approximately 60 kHz. Since the CPLD is driven by this clock and does not require a specific frequency it may work with other values. Certainly values within the same order of magnitude are likely to work.

The ARM, as the SPI master, controls when an SPI transaction occurs. The simplest way to control this, as was done in this design, is to use polling. This method is inefficient since data is repeatedly being sent whether it needs to or not but it has the side effect of helping to mitigate errors. Since each new transaction resets the registers to the beginning, any erroneous data will be quickly overwritten with new data. This will work as long as the number of transactions without errors far exceed the number of transactions with errors.

4 CPLD input/output formats

As was discussed previously, the CPLD has switch inputs and LED outputs. The orientation and format of this bits is the subject of this section.

The bank of switches are typically numbered. In this case they were numbered one through eight. The common orientation is from right to left starting with the LSB. This orientation was applied here but the result is largest switch number corresponds to the LSB (reversed).

The position of the output LEDs mimic thos of the input switches with the LSB on the right most side.

5 Calculations

The bit positions described here are referenced using the most significant bit (MSB) and least significant bit (LSB). The orientation from left to right does not necessarily correspond to the orientation of the switches and LEDs. Refer to Section 2 for a description of the orientation.

The are several steps that must be performed in order for the ARM to perform the necessary calculations. Each will be discussed below.

First the data must be received. The data should arrive over the SPI with the MSB first. And this needs to be split in to two 4-bit segments as is shown below.

```
MSB         LSB
8 7 6 5 4 3 2 1     8-bit data
   /        \
4 3 2 1   4 3 2 1   4-bit segments
   B         A      numbers A and B
```

But the ARM does not have a 4-bit data type. In order to solve this problem bit masks can be used to translate the data in to the required size. For example if the size necessary is 32-bits. This data is first set to zero, then logically OR'ed with the incoming data to "transfer" the bits.

Next it must be chosen which operation to perform, add or subtract. To decide which the on board user push button[7, Pg. 17] is used. When it is released addition is performed, when it is pressed, subtraction is performed. When subtraction is performed the second number (LSB section) is subtracted from the first (MSB section).

Another design requirement is that the add and subtract operations set the overflow and carry bits. To accomplish this using ARM specific assembly instructions must be used [4, Pg. 54] and a status register must be examined[4, Pg. 40].

Since the 4-bits had to be placed in to a larger data type they should be shifted to towards the MSB. This will ensure that the operation will set the overflow and sign bit appropriately.

Once the addition or subtraction has been performed and the results of status register are known this data must be manipulated in to two different formats, one for the LCD, and the other for the LEDs.

The format for the LED bar with 10 positions is shown below.

```
     LEDS
MSB         LSB
8 7 6 5 4 3 2 1 X X
------------------
    N V 4 3 2 1
```

Where `N` is 1 if it is negative and 0 if positive. `V` is 1 if overflow occurred and 0 otherwise. And positions 4 through 1 represent the unsigned 4-bit result with the MSB at position 4.

Some examples of the format of the LCD are shown below.

```
N1V0-3
N0V1+6
N0V0 0
```

Notice that `N` and `V` precede the bit which indicates the negative flag or overflow respectively. The remaining space is used to display the result in decimal including the sign. A negative zero is disallowed.

Table 3 list several test cases showing the expected outputs on the LEDs and the LCD when given certain switch inputs and whether or not the user button is pressed. This list is far from exhaustive. Given that the input switches have 8-bits and for each of these bits there are two possibilities (button pressed/released) this results in 512 possible combinations ($2 \cdot 2^8$).

switches		**LEDs**		**LCD**	
A	B	button=pressed	button=released	button=pressed	button=released
0000	0000	00 0000	00 0000	N0V0 0	N0V0 0
0000	0001	10 1111	00 0001	N1V0-1	N0V0+1
1010	0011	01 0111	10 1101	N0V1+7	N1V0-3
1100	0100	00 0000	10 1000	N1V0-8	N0V0 0
1100	0110	01 0110	00 0010	N0V1+6	N0V0+2
1111	1000	00 0111	01 0111	N0V0+7	N0V1+7

Table 3: Calculation test values showing switch inputs, LED, and LCD outputs. The orientation used here has the MSB on the left and the LSB on the right. For the subtraction operation the results equals A - B. When the button is released addition is performed, when it is pressed subtraction is performed.

6 References

[1] F.M. Cady. Microcontrollers and microcomputers: principles of software and hardware engineering. Oxford University Press, 2009.

[2] Lattice Semiconductor Corportation. Machxo family data sheet, 2010. DS1002 Version 02.9, July 2010.

[3] Lattice Semiconductor Corportation. Machxo2280 break board evaluation kit users guide, 2011. March 2011, Revision: EB66_01.0.

[4] S.B. Furber. ARM system-on-chip architecture. Addison-Wesley, 2000.

[5] Lattice Semiconductor. Lattice diamond design software. `http://www.latticesemi.com/products/designsoftware/diamond/index.cfm?source=topnav`, 2012. [Online; accessed 23-March-2012].

[6] ST. Rm0038 reference manual - stm32l151xx, stm32l152xx and stm32l162xx advanced arm-based 32-bit mcus, 2012.

[7] ST. Um1079 user manual - stm32l-discovery, 2012. Doc ID 018789 Rev 2.