Lab 3 (EECE 344)

Jeremiah Mahler

March 24, 2012

Contents

1 Introduction

The device described here shows how to construct a bus using a Lattice MachXO[3] board which connects to leds, switches and multiple ram memory chips. And how to control reads/writes to this bus over SPI by an ARM STM32L Discovery[6] board.

2 Pin Assignments, Schematics

The pin assignments define all the interconnecting wires between the ARM, CPLD and other components.

To locate a pin on the CPLD requires two designations[3, Pg. 11-14]. The first is the header which has names such as J9, J7, etc. And the second is the number of the pin. The board will have pin numbers at the begining and end of a header to denote the orientation.

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.

When specifiying the pin constraints in Diamond[4] the header and pin number are not available. Instead the "Mach XO Ball" must be specified. And this value is included in the following pin assignments.

The pin assignments are given in Tables 1, 2, 4 as well as the schematic in Figure 1.

The pins were configured with standard options: low voltage 3.3 volt CMOS with no pull up or pull down. Output pins to drive the leds were configured for 8 mA drive current.

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 4).

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[6, 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 4 lists the exact pins that were used.

RAM			CPLD			
pin	label	description	function	Mach XO Ball	Header	Pin
12	A0	mem_address	PL17D	L4	J4	36
11	A1	mem_address	PL12D	L2	J4	35
10	A2	mem_address	PL17C	L5	J4	34
9	A3	mem_address	PL12C	K2	J4	33
8	A4	mem_address	PL15C	M2	J4	26
7	A5	mem_address	PL10C	G1	J4	21
6	A6	mem_address	PL10D	H1	J4	23
5	A7	mem_address	PL8D	H3	J4	19
27	A8	mem_address	PL15D	N2	J4	28
26	A9	mem_address	PL11C	J3	J4	29
23	A10	mem_address	PL19A	N4	J4	38
25	A11	mem_address	PL11D	K3	J4	31
4	A12	mem_address	PL8C	G3	J4	17
28	A13	mem_address	PL16D	R2	J4	32
3	A14	mem_address	PL7C	E1	J4	13
31	A15	mem_address	PL7D	F1	J4	15
2	A16	mem_address	PL6D	D1	J4	11
13	DQ0	mem_data	PL7A_LV_T	F2	J3	25
14	DQ1	mem_data	PL17A_LV_T	K5	J3	32
15	DQ2	mem_data	PL18A_LV_T	M5	J3	38
17	DQ3	mem_data	PL9A_LV_T	H4	J3	37
18	DQ4	mem_data	PL8A_LV_T	G4	J3	31
19	DQ5	mem_data	PL16A_LV_T	J4	J3	26
20	DQ6	mem_data	PL15A_LV_T	L3	J3	20
21	DQ7	mem_data	PL5A_LV_T	B1	J3	19
32	Vcc	suppy voltage			J9	5
16	Vss	ground			J3	36

Table 1: Pin assignments between RAM chip and the CPLD which are common to both chips. See Table 2 for those pins which are unique for each chip.

RAM #1				CPLD			
pin	label	variable	description	function	Mach XO Ball	Header	Pin
30	CE2	mem1_ce2	chip enable	PL14C	N1	J4	37
22	CE#	mem1_ceh_n	chip enable	PL19B	N3	J4	40
29	WE#	mem1_we_n	write enable	PL14D	P1	J4	39
24	OE#	mem1_oe_n	output enable	PL16C	R1	J4	30

RAM #2				CPLD			
pin	label	variable	description	function	Mach XO Ball	Header	Pin
30	CE2	mem2_ce2	chip enable	PL8B	G5	J3	33
22	CE#	mem2_ceh_n	chip enable	PL11B	J2	J3	4
29	WE#	mem2_we_n	write enable	PL19B	H5	J3	39
24	OE#	mem2_oe_n	output enable	PL18B	M4	J3	40

Table 2: Pin assignments between RAM chip and the CPLD which are unique to each RAM chip.

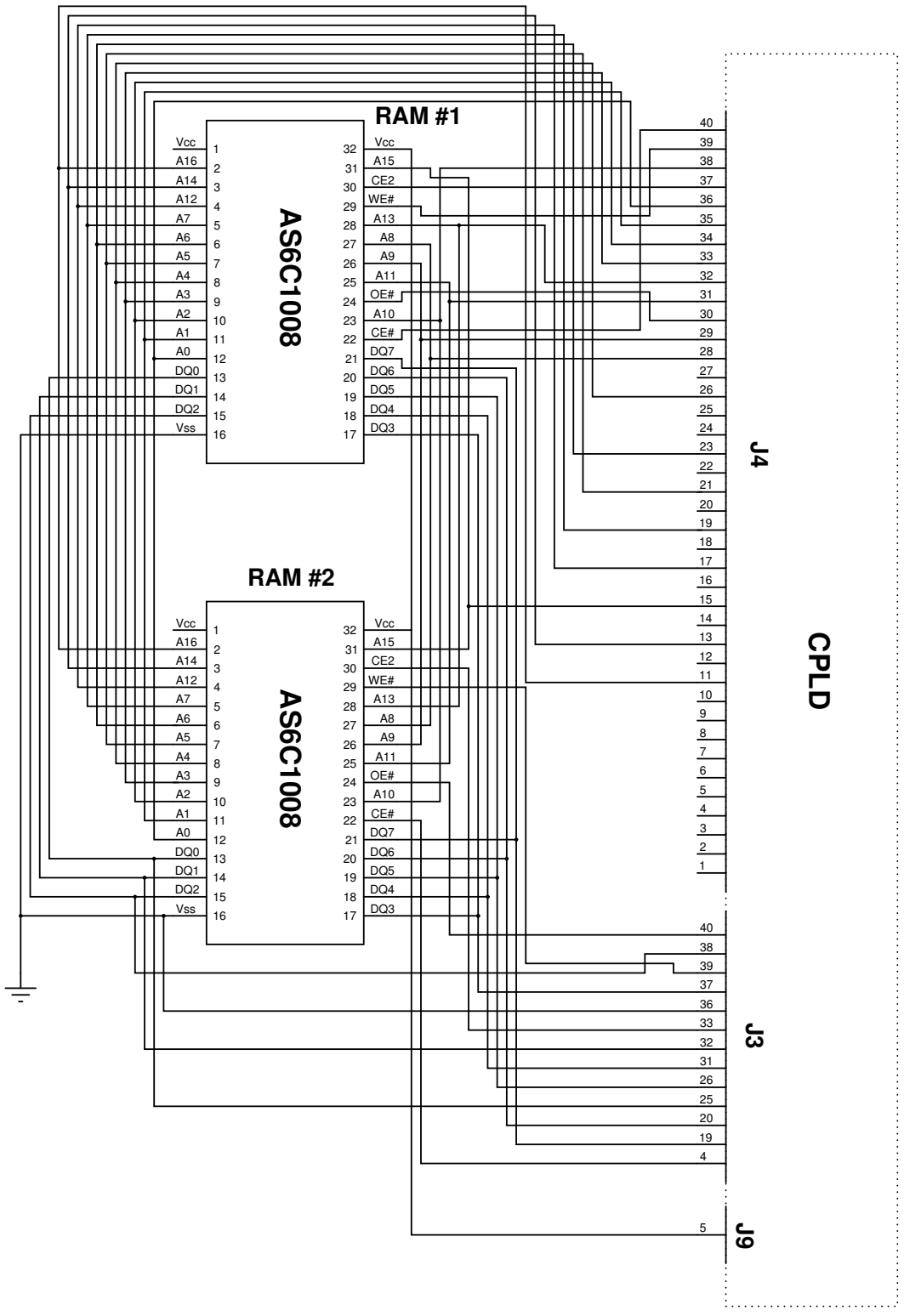


Figure 1: Schematic diagram of wires between both RAM chips and the CPLD.

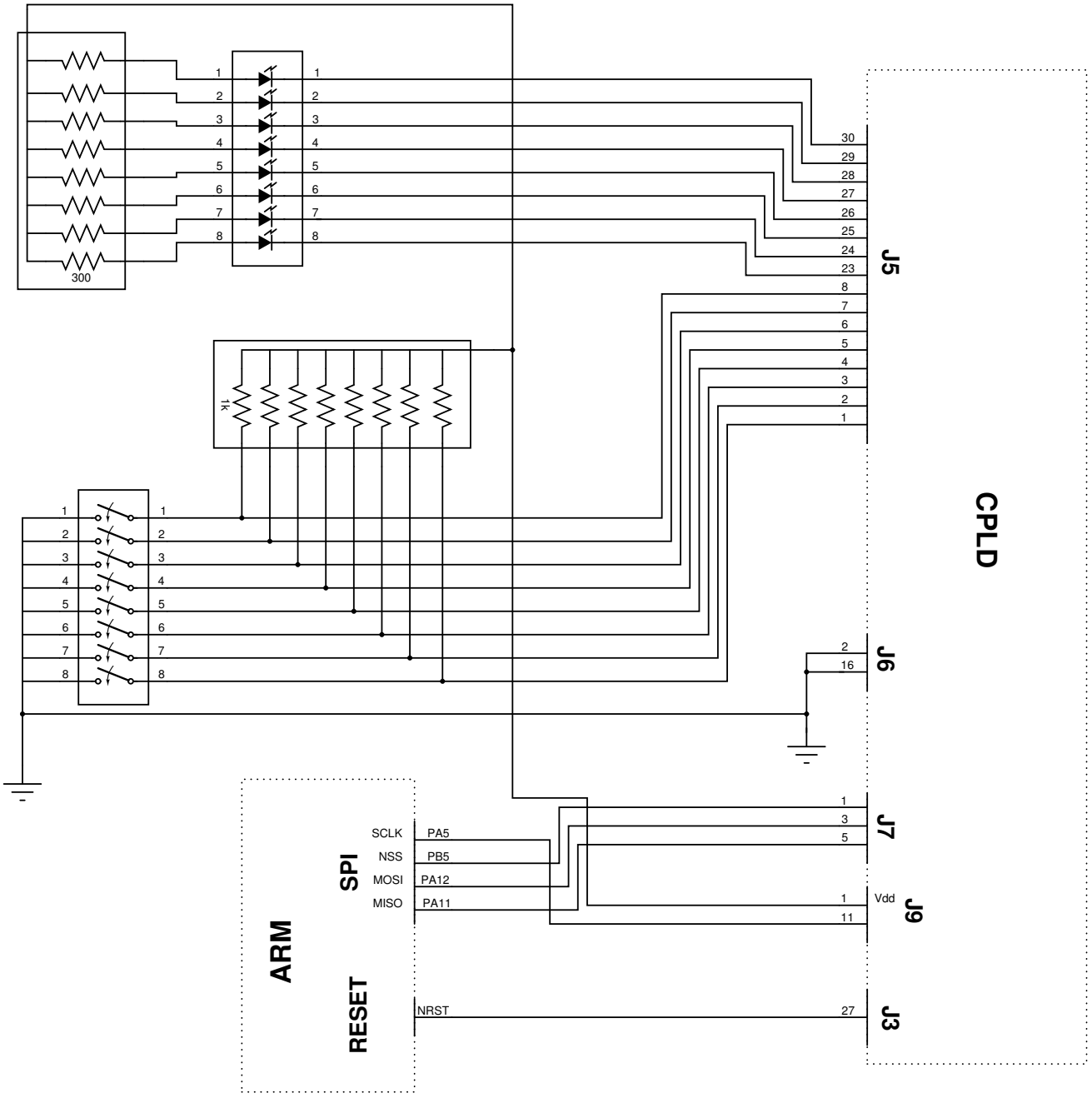


Figure 2: Schematic diagram of wires between the ARM and the CPLD.

Memory Map		
address (hex)	name	variable
0x74	switches	switch_ce_n
0x6C	bar leds	bar_led_ce_n
0x50 - 0x5F	RAM #2	mem2_ce_n
0x2F	board leds	board_led_ce_n
0x00 - 0x0F	RAM #1	mem1_ce_n

Table 3: Device memory map.

SPI						
Verilog		**ARM**	**CPLD**			
name	description	pin	function	Mach XO Ball	Header	Pin
SCLK	SPI clock	PA5	PT9B	D7	J9	11
NSS	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
switches						
Verilog		**input switches**	**CPLD**			
name	description	pin	function	Mach XO Ball	Header	Pin
switches[0]	input switch 8	8	PT2C	B2	J5	1
switches[1]	input switch 7	7	PT9A	D8	J5	2
switches[2]	input switch 6	6	PT2D	B3	J5	3
switches[3]	input switch 5	5	PT9C	E8	J5	4
switches[4]	input switch 4	4	PT3A	A2	J5	5
switches[5]	input switch 3	3	PT9D	E9	J5	6
switches[6]	input switch 2	2	PT3B	A3	J5	7
switches[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
bar_leds[0]	output led 8	8	PT15D	B5	J5	23
bar_leds[1]	output led 7	7	PT12B	A12	J5	24
bar_leds[2]	output led 6	6	PT6E	E7	J5	25
bar_leds[3]	output led 5	5	PT12C	B11	J5	26
bar_leds[4]	output led 4	4	PT6F	E6	J5	27
bar_leds[5]	output led 3	3	PT12D	B12	J5	28
bar_leds[6]	output led 2	2	PT16C	A5	J5	29
bar_leds[7]	output led 1	1	PT13C	C11	J5	30
	power, Vdd, pull up				J9	1
	ground				J6	16
reset						
Verilog		**ARM**	**CPLD**			
name	description	pin	function	Mach XO Ball	Header	Pin
reset_n	active low reset	NRST	PL7B	G2	J3	27

Table 4: 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.

8	7 1
rw bit	address
data	

Table 5: Format of two byte SPI transactions.

option	value
MSB	first
CPOL	0
CPHA	0
NSS	slave select

Table 6: SPI configuration options

3 SPI protocol

Communication with the "bus" on the CPLD is accomplished using SPI [1, Pg. 278; 5, Pg. 665]. For the bus there are two required operations: read and write. And in order to perform these operations it requires a minimum of two transactions over the SPI. A read, for example, would send the address in the first transaction. And in the second transaction the data read would be returned. A write is similar except that for the second transaction the data to be written is sent. The format of the transaction is defined in Table 5.

This implementation has the SPI settings hard coded as shown in Table 6. Both the ARM and the CPLD must use these same settings in order to work properly with each other.

A complication of this two byte protocol pertains to how to determine which byte is the first and which is the second. The method used here is to modify the SPI protocol so that the NSS signal is held active across two bytes instead of just one. This solution requires no additional hardware and is straight forward to implement in the CPLD.

Because the SPI is full duplex process a byte is sent and received on every transaction. Because of this some values are ignored in certain situations. For example, during a read operation, the address is sent first, then a second byte must be sent in order to get the received value. The second value sent is ignored and could be any value.

For a detailed timing diagram of the read and write operations refer to Figures 3 and 4.

When values are read or written to the bus they must be held for some amount of time. In this implementation this is accomplished by controlling the pulse width from the end of the 16'th `sck` edge and the positive edge of `nss` (going disabled). Since the clock speed of the SPI is on the order of kHz and the clock inside the CPLD is on the order of MHz there should be plenty of time for any memory timings or other operations to be done.

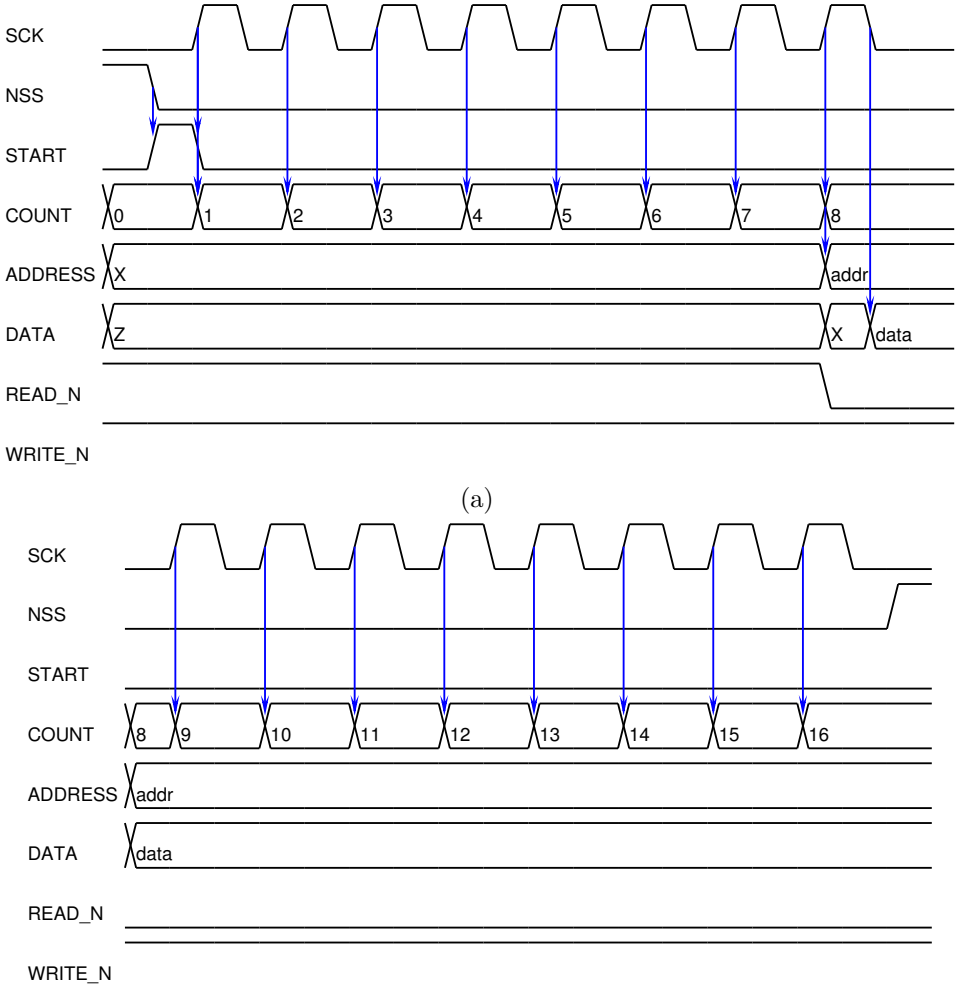
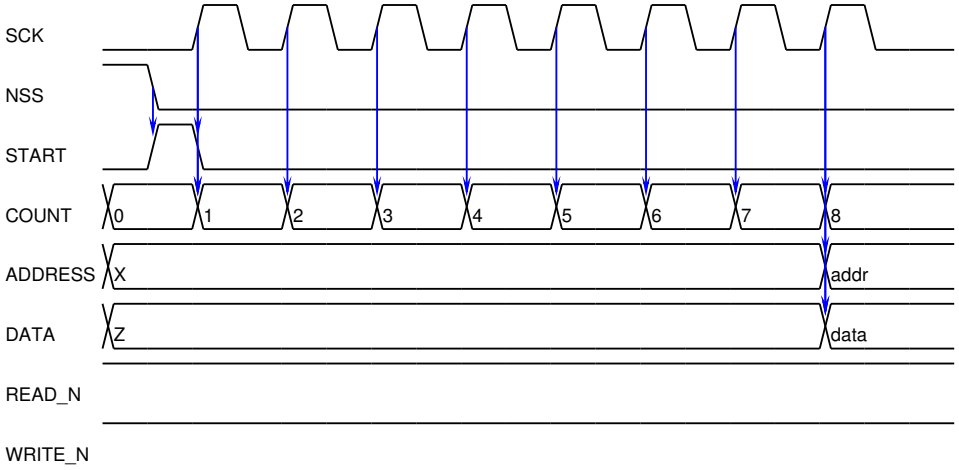
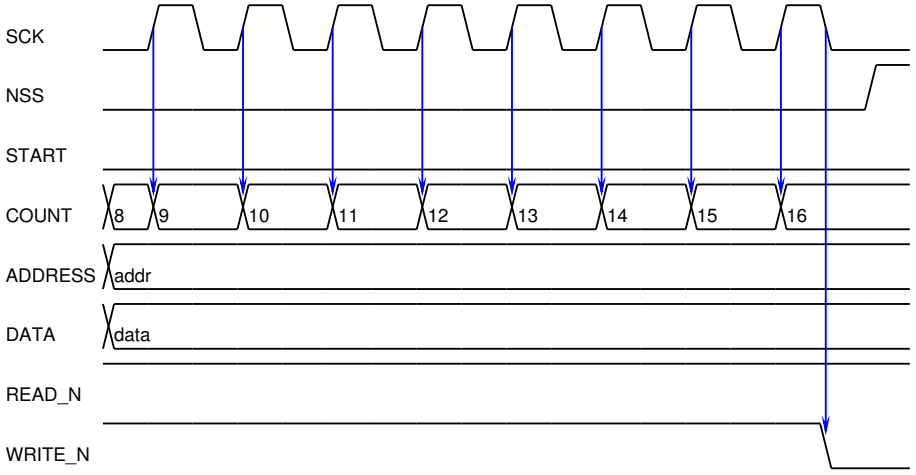


Figure 3: Timing diagram of SPI read cycle. Part (a) is the first 8-bits and part (b) is the second 8-bits continuing from (a). Notice in part (a) where the ADDRESS is valid and READ_N becomes active (low). The data is not immediately read but instead it is delayed until the negative edge of the SCK. This is necessary for device such as RAM chips which may require a short period of time to produce valid data.



(a)



(b)

Figure 4: Timing diagram of SPI write cycle. Part (a) is the first 8-bits and part (b) is the second continuing from (a). A write is not initiated until the end of the second byte because it has to read this byte entirely before it can be written.

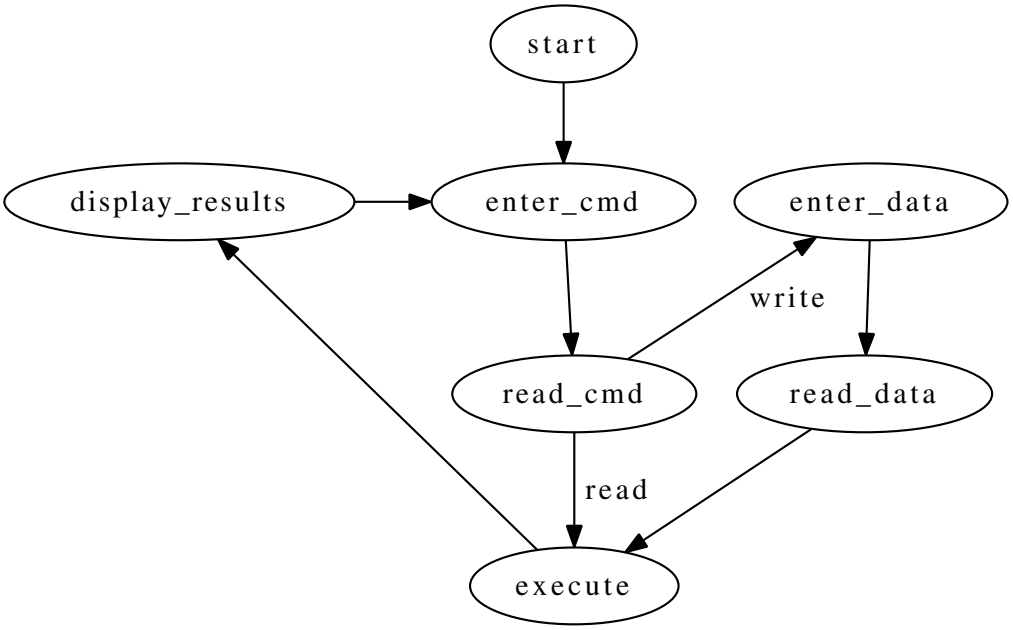


Figure 5: State diagram of ARM board operation. The states 'display_results', 'enter_cmd' and 'enter_data' wait for the user set the input switches as necessary and then press the user button. 'read_cmd' reads the input switches to get the command (address and rw bit). If the command is a write 'read_data' reads the switches again to get the data. And 'execute' performs the read/write command.

4 User Interface

In order for the user to be able to perform read and write operations to the bus devices an interface needs to be defined. The only means of input is using an eight position DIP switch. Outputs are provided by an LCD, a bar of eight LEDS, and a group of eight LEDs on board the CPLD.

The bar of LEDs and the group of LEDs on the CPLD both operate similarly. They act as an eight bit register which can be read from and written to. Refer to Table 3 for their specific addresses.

The LCD is the primary means of user feedback. It instructs the user when to enter a command or data and it displays the results. Figure 5 gives an overview of the states of the system.

As an example of reading the switches. Notice that the only difference is the inclusion of the rw bit as part of the data value whereas the address is only 7-bits and excludes this part.

```
CMD
74 R F4
```

Refer to the memory map (Table 3) for a complete list of the addresses that can be read from and written to.

5 Conclusion

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 breakout board evaluation kit users guide, 2011. March 2011, Revision: EB66_01.0.

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

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

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