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.

IMPORTANT - As of this writing this project has not been completed. The information contained in this document has not been verified against a working device.

The primary obstruction to the completion of this project was due to the difficulty in constructing a synthesizeable solution for the SPI protocol. Verilog provides a rich language[7] but only a small subset of its language is synthesizeable using Diamond[4].

2 Pin Assignments

One of the critical design specifications for this device are the pin assignments. Table 2 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.

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[4].

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 2).

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 2 lists the exact pins that were used.

3 SPI protocol

The SPI protocol uses typical SPI transitions [1, Pg. 278; 5, Pg. 665] and the configuration options used are given in Table 3.

Normally the SPI protocol is enabled using the NSS pin at the start of a transaction and disabled at the end of 8-bits. In this case it was necessary to transfer two bytes. And it was also necessary to detect the start/end of the transaction. To accomplish this the SPI protocol was modified so that NSS remains enabled for two bytes (16 bits). This solutions required no additional pins.

To communicate with the bus requires two bytes of data with 8-bits each. The first byte sent from the master to the slave contains the read/write bit and the address. The first return byte is ignored.

For the second byte it depends on whether it is a read or a write operation. For a read any data can be sent and the value read will be returned. This in analogous to a printer "form feed". For a write the data to be written must be sent as the second byte. The value returned should be ignored.

RAM			CPLD			
pin	label	description	function	Mach XO Ball	Header	Pin
12	A0	address	PL17D	L4	J4	36
11	A1	address	PL12D	L2	J4	35
10	A2	address	PL17C	L5	J4	34
9	A3	address	PL12C	K2	J4	33
8	A4	address	PL15C	M2	J4	26
7	A5	address	PL10C	G1	J4	21
6	A6	address	PL10D	H1	J4	23
5	A7	address	PL8D	H3	J4	19
27	A8	address	PL15D	N2	J4	28
26	A9	address	PL11C	J3	J4	29
23	A10	address	PL19A	N4	J4	38
25	A11	address	PL11D	K3	J4	31
4	A12	address	PL8C	G3	J4	17
28	A13	address	PL16D	R2	J4	32
3	A14	address	PL7C	E1	J4	13
31	A15	address	PL7D	F1	J4	15
2	A16	address	PL6D	D1	J4	11
30	CE2	chip enable	PL14C	N1	J4	37
22	CE#	chip enable	PL19B	N3	J4	40
29	WE#	write enable	PL14D	P1	J4	39
24	OE#	output enable	PL16C	R1	J4	30
13	DQ0	data	PL7A_LV_T	F2	J3	25
14	DQ1	data	PL17A_LV_T	K5	J3	32
15	DQ2	data	PL18A_LV_T	M5	J3	38
17	DQ3	data	PL9A_LV_T	H4	J3	37
18	DQ4	data	PL8A_LV_T	G4	J3	31
19	DQ5	data	PL16A_LV_T	J4	J3	26
20	DQ6	data	PL15A_LV_T	L3	J3	20
21	DQ7	data	PL5A_LV_T	B1	J3	19
32	Vcc	suppy voltage			J9	5
16	Vss	ground			J3	36

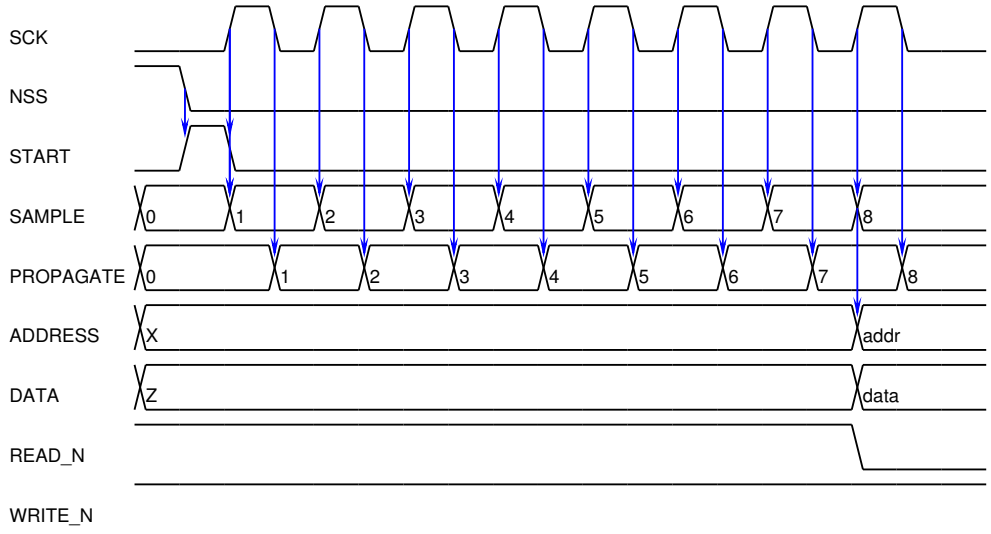
Table 1: Pin assignments between RAM chip #1 and CPLD.

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

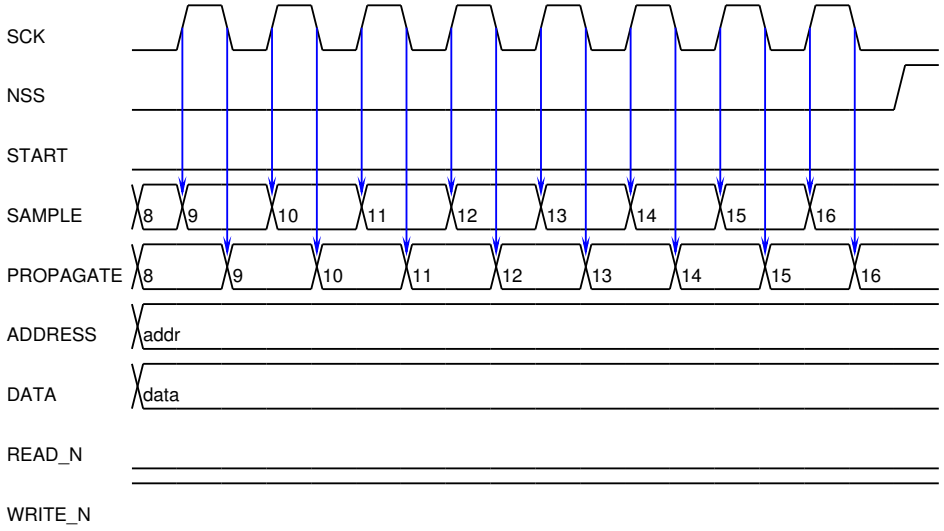
Table 2: Definition of the pin assignments between the ARM board, the CPLD, and other devices. Notice that the switch and LEDs are reversed. This was done so that the orientation from LSB to MSB is from right to left.

option	value
MSB	first
CPOL	0
CPHA	0
NSS	slave select on low

Table 3: SPI configuration options

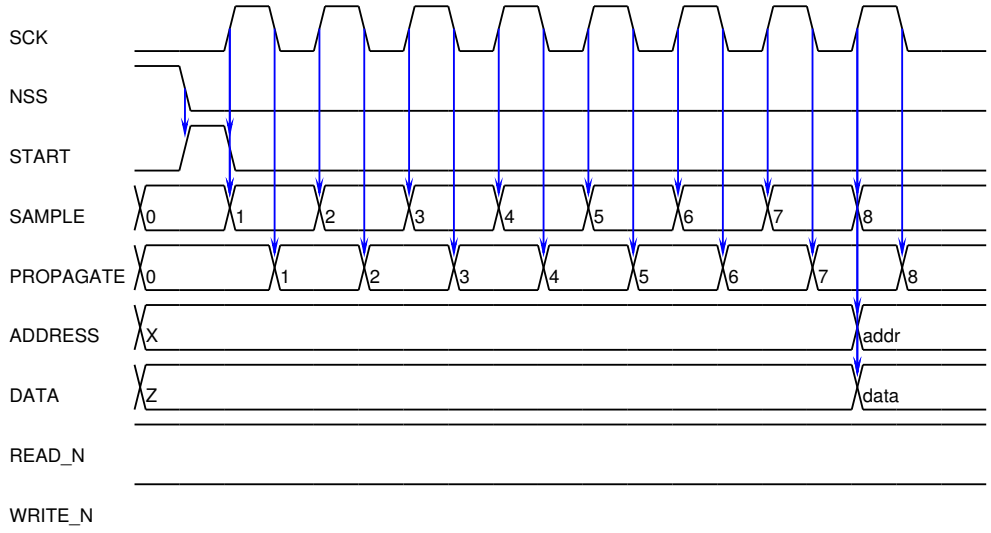


(a)

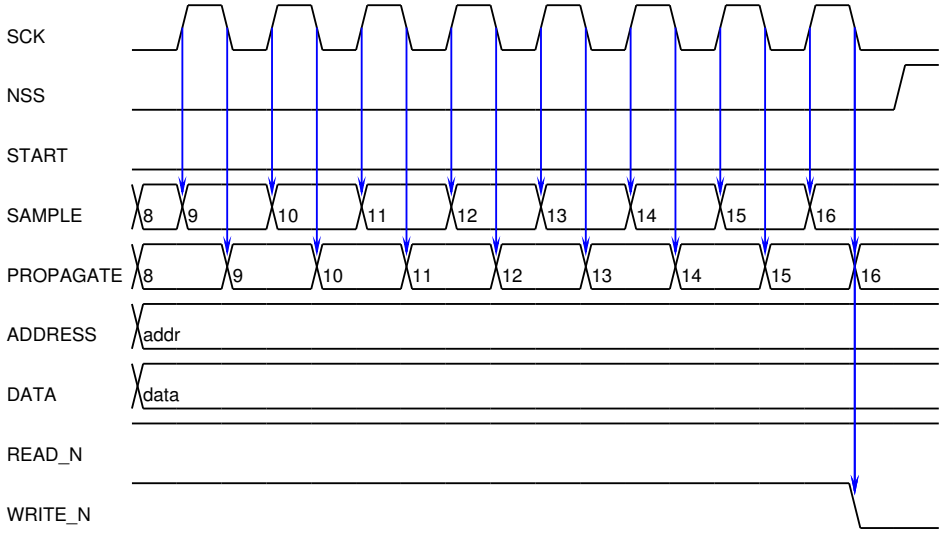


(b)

Figure 1: Timing diagram of a SPI read cycle. Part (a) is the first 8-bits and part (b) is the second continuing from (a). The transactions at the start of b aren't really true, the data is constant from a to b. In part (a) the data is not valid until a cycle after the address is set. This is due to the time it takes the decoder to enable the chip and get valid data on the data bus. The data must be valid before the start of the second byte when it is loaded in to the shift register to be sent back on the SPI.



(a)



(b)

Figure 2: Timing diagram of a 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.

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.

4 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] 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.

[7] D.E. Thomas and P.R. Moorby. The Verilog Hardware Description Language. Number v. 2 in The Verilog Hardware Description Language. Kluwer Academic Publishers, 2002.