

SPI Interface

Version 2: corrected timing diagram.

Introduction

The **Serial Peripheral Interface** (SPI) is a synchronous serial interface often used to communicate between a microcontroller (acting as the “master”) and a peripheral IC (the “slave”). The interface uses a clock (SCLK), a slave-select (SS) and serial input and/or output signals (MISO and MOSI).

In this lab you will design and implement an SPI master interface to transmit and receive four BCD digits (16 bits).

You will use the Analog Discovery 2 (AD2) oscilloscope, logic analyzer and SPI protocol analyzer functions to display the waveforms and data. The transmitted and received data will be displayed on the four-digit 7-segment display used in previous labs.

A suggested implementation is given that consists of two state machines: a controller and a datapath. The controller is based on an binary counter and the datapath on a shift register.

Specifications

Your design should have a 50 MHz clock input, outputs to drive a 4-digit 7-segment multiplexed LED display as in previous labs, an SPI master interface and two pushbutton inputs.

Pressing the **reset** button loads the data shift register with a secret number that is different for each student. You can retrieve this value from the Grades section of the course web site. Sharing this value will result in disciplinary action.

When the **transmit** button is pushed, **ss_n** is asserted and the 16 bits in the data register are transmitted serially, most-significant-bit first, over **mosi** at 1 MHz or less and 16 bits are received over **miso**. **mosi** should change on the falling edge of **sclk** and the value of **miso** should be captured on the rising edge of **sclk** (this is the **CPOL=0, CPHA=0** format).

The contents of the data shift register – the secret number when **reset** is pressed and the received 16-

bit value after that — should be displayed on the four-digit LED display.

Your design should have a test output, **mosi_n**, that is the inverted MOSI output. This signal will be connected to the MISO input to demonstrate the receive function.

Suggested Solution

Interface

You can download a Quartus project archive from the course web site that includes a **lab7.sv** Verilog file with the following module declaration:

```
module lab7
( input logic clock,           // 50 MHz clock
  input logic reset_in, transmit_in, // reset, send
  ↪ buttons
  output logic [3:0] en,       // digit enables
  output logic a, b, c, d, e, f, g, dp, // segments
  output logic ss_n, sclk, mosi, // SPI master
  output logic mosi_n,        // MISO test
  ↪ output
  input logic miso_in         // MISO input
) ;
```

Controls

The two pushbutton inputs should be debounced.

The **transmit** input should be registered so you can detect a rising edge by comparing the flip-flop input and output. For example:

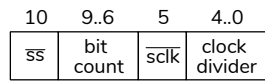
```
debounce db0 ( transmit_in, clock, transmit_next );
always_ff @(posedge clock) transmit = transmit_next ;

...
assign ... = !transmit && transmit_next ? ... : ... ;
```

Controller

A counter can be used as the controller state machine because this SPI interface always goes through the same sequence of states.

The solution described here uses an 11-bit binary counter that combines four counters: the most significant bit is a sign bit, the next 4-bits count the data bits, one bit serves as the serial clock, and the least significant 5 bits divide the clock rate by 32:



On a rising edge of `transmit` the counter is loaded with $16 \times 64 - 1$ (`11'b011_1111_1111`). The counter value is decremented once per clock period until it “wraps around” to `-1`.

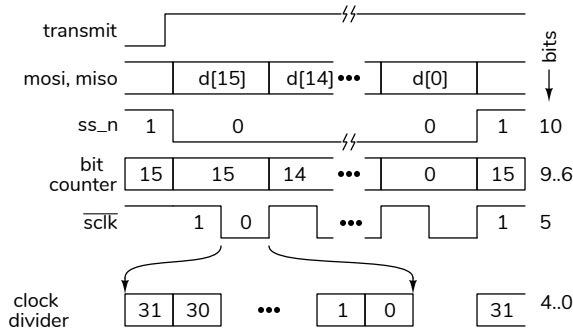
Bit 10 is connected to `ss_n`. It loaded with 0 and when the counter “wraps around” to `-1` it becomes 1.

Bits 9 to 6 is the number of bit being sent. It is loaded with 15 (`4'b1111`) and decrements down to 0.

Bit 5 is inverted and connected to `sclk`. It is loaded with 1 and decremented (inverted) at a rate of $50/32 \approx 1.56$ MHz, creating a clock with a frequency of about 781 kHz.

Bits 4 to 0 These “clock divider” bits are loaded with 31 (`5'b11111`) and are decremented at the clock rate (50 MHz).

The following timing diagram shows the relationship between the clock divider bits, `sclk`, the bit counter bits and `ss_n`.



Datapath

The `miso` input is registered (loaded into a flip-flop) on the rising edge of `sclk`:

```
logic miso, miso_next ; // registered MISO input
assign miso_next = !sclk && sclk_next ? miso_in : miso ;
always @(posedge clock) miso = miso_next ;
```

This allows `miso_in` to be captured on the rising edge of `sclk` before being shifted into the `data` register on the falling edge of `sclk`.

The 16-bit shift register (labelled `d` in the timing diagram) is loaded with the student-specific secret number when `reset` is asserted. On the falling edge of `sclk` the shift register is shifted left by one bit and the registered value of `miso` (see above) is shifted into the least-significant bit.

Hints

The pushbutton inputs should be debounced. You can use the `debounce` module made available in the previous lab.

To detect rising or falling edges, compare the current and `_next` values of a register. For example, if `sclk` is a register’s output and `sclk_next` is its input, then `sclk && !sclk_next` is true only during the clock period before falling edge of `sclk`.

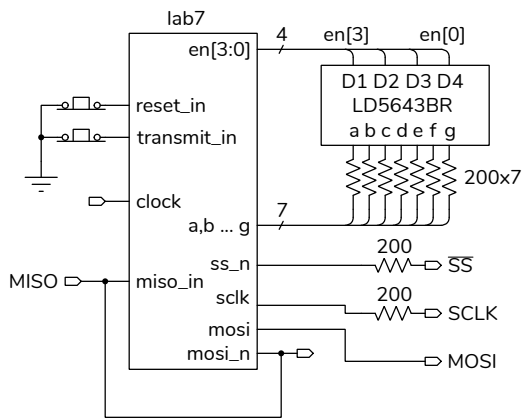
Remember that the coding guidelines only allow one signal to be used as a clock. You cannot use any of these other signals in the sensitivity lists of `always` statements (for example, you cannot use `@posedge(sclk)`).

Components

You will need the same components as in the previous labs that used the 7-segment LED display, two pushbutton switches and the AD2.

CPLD I/O

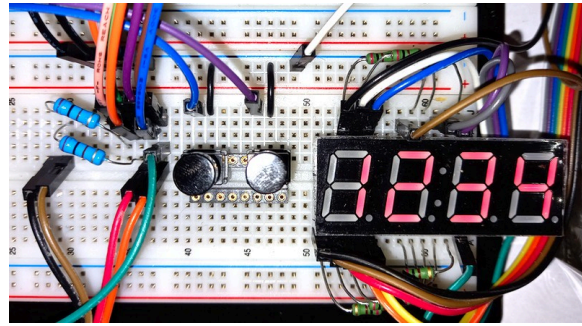
The connections to the CPLD are shown in the following schematic:



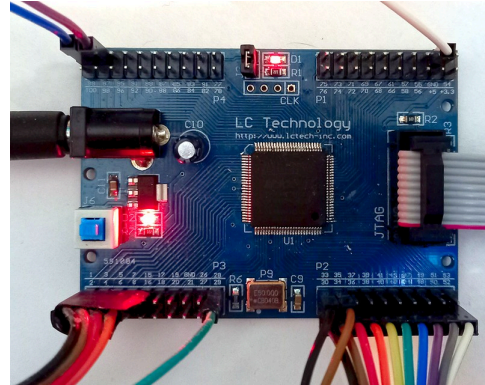
The pin assignments for the LED display are the same as in the previous lab. Use internal pull-up resistors on the `reset` and `transmit` pushbutton inputs. The four SPI interface pins are connected to the breadboard so that they can be monitored with the AD2 analog (scope) or digital inputs. The pin assignments are shown below:

To	Assignment Name	Value
out a	Location	PIN_33
out b	Location	PIN_44
out c	Location	PIN_38
in clock	Location	PIN_12
out d	Location	PIN_34
out dp	Location	PIN_36
out e	Location	PIN_30
out en[0]	Location	PIN_42
out en[1]	Location	PIN_48
out en[2]	Location	PIN_50
out en[3]	Location	PIN_35
out f	Location	PIN_52
out g	Location	PIN_40
in miso_in	Location	PIN_7
out mosi	Location	PIN_5
out mosi_n	Location	PIN_28
in reset_in	Location	PIN_99
in reset_in	Weak Pull-Up Resistor	On
out sclk	Location	PIN_3
out ss_n	Location	PIN_1
in transmit_in	Location	PIN_97
in transmit_in	Weak Pull-Up Resistor	On

The wiring to the 7-segment LED, the pushbuttons, the SPI interface signals and the AD2 test leads is shown below:



and the connections to the CPLD pin headers are shown below:

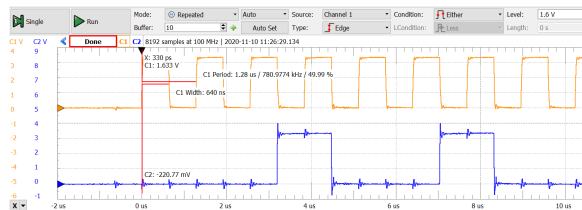


The series resistors on `sclk` and `ss_n` terminate these signals and reduce ringing that can cause false triggering.

Use of AD2

You can use the AD2 scope, logic analyzer and protocol windows for troubleshooting and to verify the operation of your design.

The two scope channels can be connected to the pushbutton inputs or the SPI signals to verify the voltage levels and check for signal integrity issues such as noise, glitches or ringing. You can trigger on the falling edge of `ss_n` to capture one transfer. The following example shows the `sclk` on channel 1 and `mosi` on channel 2 along with a measurement of the `sclk` period:



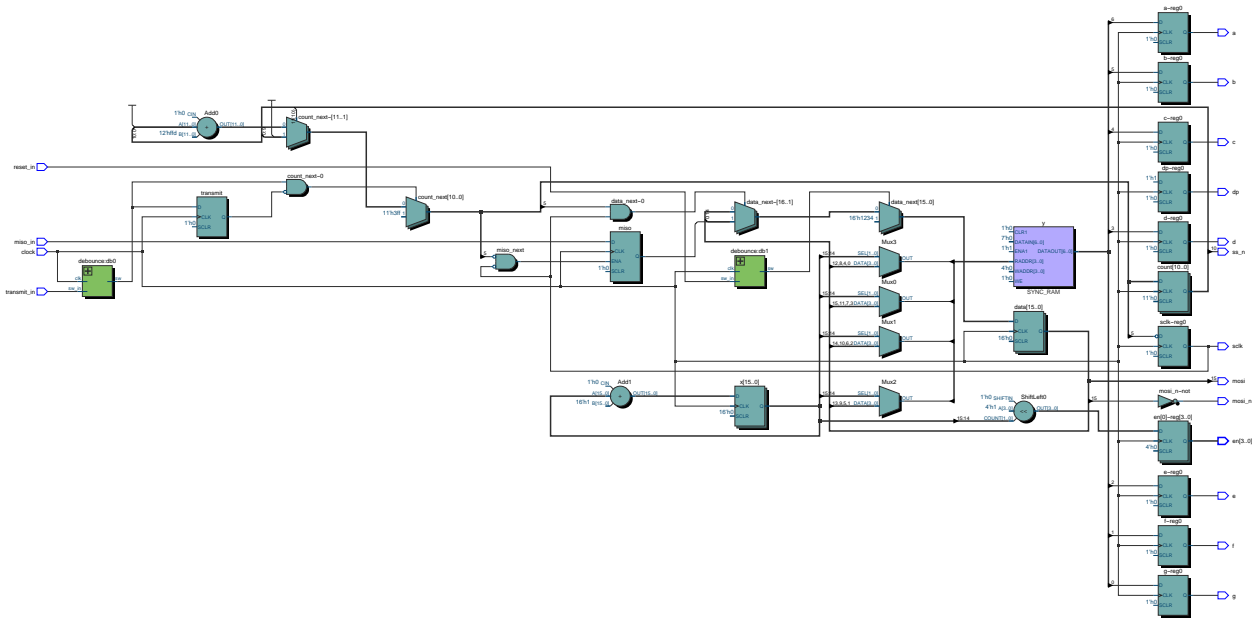
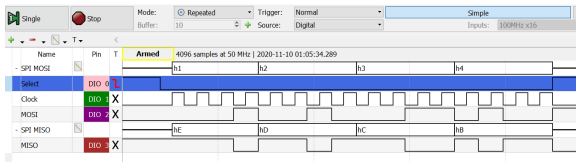
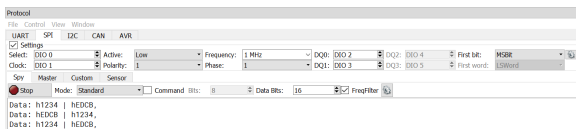


Figure 1: Example RTL Schematic for Lab 7.

The logic analyzer can be used to display digital signals. It can display bus values (not used here) and multi-bit serial signals. The example below uses a configuration called “SPI MOSI/MISO” which shows the transmitted and received serial data. The trigger (T column) has been set to the falling edge of the Select signal and the MOSI bit width has been set to 4 bits:



The protocol analyzer decodes more complex protocols such as those including device addresses and variable-length fields (neither used here). The following screen captures show the SPI decoding when a 16-bit value is transmitted and its complement is received three times:



You can troubleshoot your design by defining extra CPLD pins as outputs and connecting these to digital inputs of the AD2. This allows the logic analyzer

to monitor other signals in your design. For example, you can output the controller counter value or the datapath shift register to see if they’re behaving as you expect.

Note that you need to connect the two AD2 ground pins (the two black wires) and the two ’scope inverting inputs (blue and orange with white stripes) to the ground bus on your breadboard.

The default connections for the logic analyzer SPI MOSI/MISO and the SPI protocol analyzer are:

CPLD Pin	AD2 Signal	Lead Color	SPI Signal
1	DI00	pink	SS_N
3	DI01	green	SCLK
5	DI02	violet	MOSI
7	DI03	brown	MISO

Note: There seems to be a **conflict** between the AD2 and the USB-Blaster drivers. You may need to disconnect the AD2 each time you program the CPLD.

Submission

To get credit for completing this lab, submit the following to the Assignment folder for this lab on the

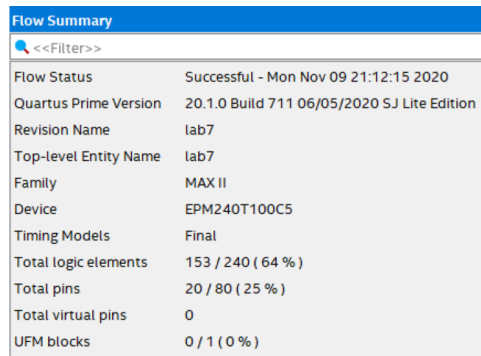
course website:

1. A PDF document containing:

- Your name, BCIT ID, course number and lab number.
- A listing of your System Verilog file. You need not include the `debounce.sv` file. You must follow the coding guidelines given on the “Course Information” section of the course website. Note that these may have changed.

The listing should be included as text rather than an image.

- a screen capture of your compilation report (Window > Compilation Report) similar to: this:



The image shows a screenshot of the 'Flow Summary' window in Quartus. It displays the following information:

Flow Summary	
Flow Status	Successful - Mon Nov 09 21:12:15 2020
Quartus Prime Version	20.1.0 Build 711 06/05/2020 SJ Lite Edition
Revision Name	lab7
Top-level Entity Name	lab7
Family	MAX II
Device	EPM240T100C5
Timing Models	Final
Total logic elements	153 / 240 (64 %)
Total pins	20 / 80 (25 %)
Total virtual pins	0
UFM blocks	0 / 1 (0 %)

- screen captures of the AD2 'scope, logic analyzer and protocol analyzer similar to those shown above and demonstrating the operation of your interface.

2. The PDF file containing the schematic created by Tools > Netlist Viewers > RTL Viewer, and then File > Export... . The file should look similar to Figure 1.

3. A video of your breadboard showing:

- your secret number being loaded when the `reset` button is pushed
- the complemented value being displayed when the `transmit` button is pushed
- the original value being displayed when the `transmit` button is pushed again