ECED2200 – Lab #5

STUDENT NAME(s): _____ .

STUDENT NUMBER(s): B00 _____ .

Pre-Lab Information

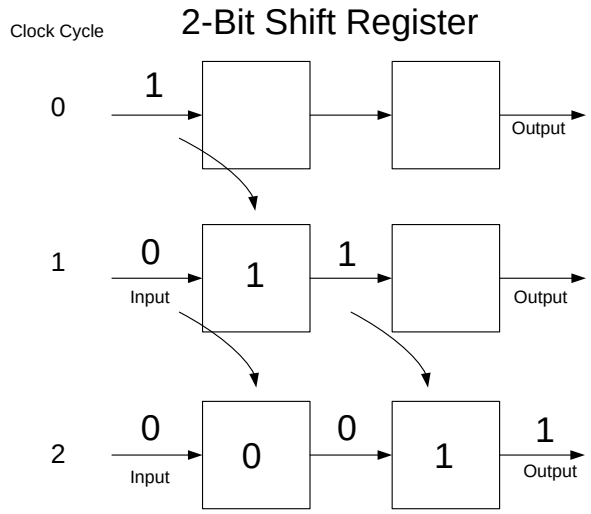
It is recommended that you read this entire lab ahead of time. Doing so will save you considerable time during the lab.

There is also several videos showing the use of the software tools – you will save considerable time by watching those now. See www.colinoflynn.com/teaching and look at the 'lab 4' information.

Overall Objective

In the first part of this lab you will learn about changing input/output modes of pin drivers, to learn about serial to parallel shift registers, and parallel to serial shift registers.

A Shift Register contains multiple 'bits', the value of such a bit being shifted between each internal register on the clock edge. Consider the following simple 2-bit Shift Register:



As we increase the 'Clock Cycle', the value at the input gets 'shifted through' the register. This is the basic idea of a shift register.

There is some additional things you might see too – for example we can actually pre-load the shift registers in parallel. Then over the next clock cycles we'll see the outputs of the shift register one at a time.

In real life this is very useful, since we may need to send a lot of data over a single wire for example. We can use shift registers to accomplish this, since we can output many bits a single bit at a time.

Deliverables

You will work in groups of 2 for this lab. Each group must deliver a lab report; the format for it is described separately.

Part #1: Transmitting Data Serially

Objective

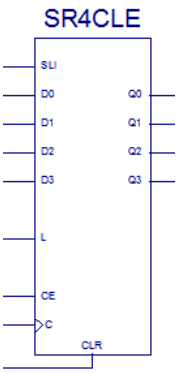
- Learn about Serial-to-Parallel and Parallel-to-Serial Block
- Learn about changing output/input pins around

Required Materials

- Computer with Xilinx ISE 13.2 Webpack installed.
    - All computers in the lab have this installed.
    - This is free software so you can install on your own computer if you wish, you can download it from http://www.xilinx.com/support/download/index.htm - select '13.2' on the side. The file is very large so you may wish to download at school, and you are required to register to license it.
- Example project file DigitalTrainer_Simple.ZIP
    - This file contains an environment which is already setup for your lab.
- Digital Trainer Board

Background

We will be using a 4-bit shift register component. This shift register is shown below:



The inputs are as follows:

SLI	Shift-Left serial Input
D0..D3	Parallel Data Input
L	Load: Loads shift register on next clock pulse
CE	Clock Enable (active high)
C	Clock
CLR	Asynchronous Clear

To use this as a serial-to-parallel shift register:

SLI	Serial data input
D0..D3	Disconnected
L	Low
CE	Clock Enable (will tie high in this lab)
C	Clock
CLR	Asynchronous Clear
Q0..Q3	Parallel Output

Or to use as a parallel-to-serial shift register:

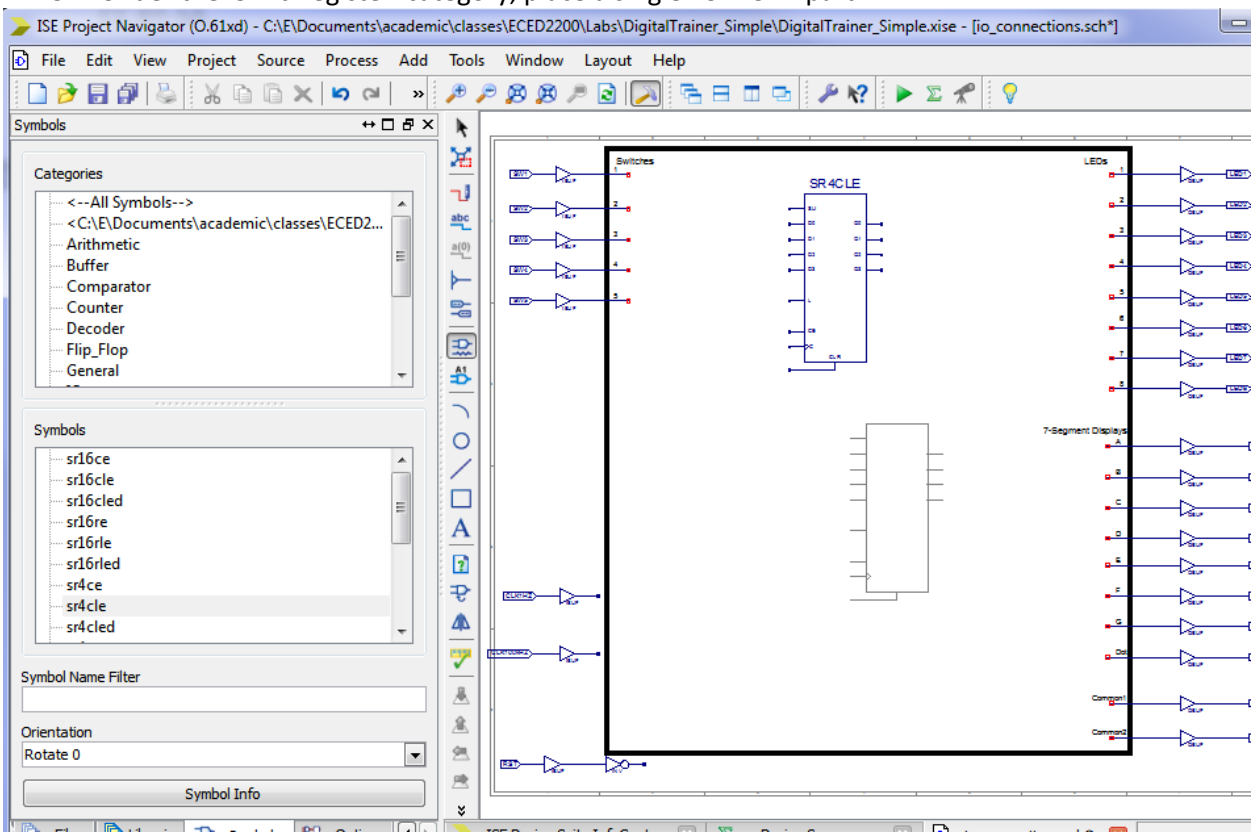
SLI	Disconnected
D0..D3	Parallel Data In
L	Pulse high for one clock edge when parallel data valid
CE	Clock Enable (will tie high in this lab)
C	Clock
CLR	Asynchronous Clear
Q0..Q2	Disconnected
Q3	Serial Data Out

Procedure

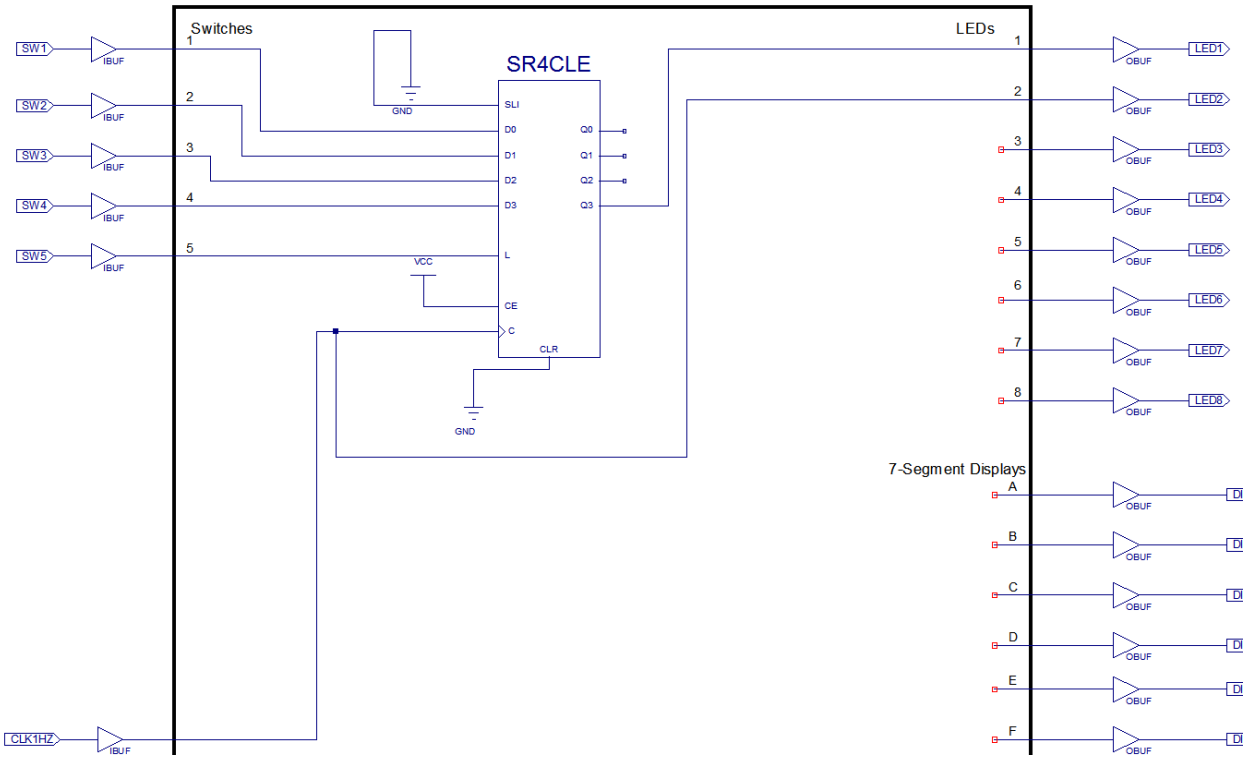
Part 1-A

1. Download DigitalTrainer_Simple.zip from one of these locations:
    a. http://www.assembla.com/wiki/show/bora/Base_Reference_File
    b. www.colinoflynn.com/teaching under the ECED2200 lab #4.

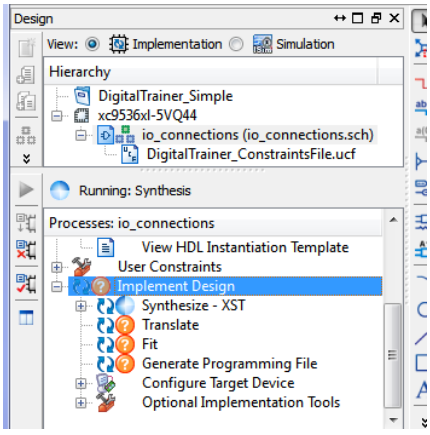
2. Unzip this somewhere.
3. Open the DigitalTrainer_Simple.xise
4. Open io_connections.sch
5. Under the 'Shift Register' category, place a single: 'SR4CLE' part:



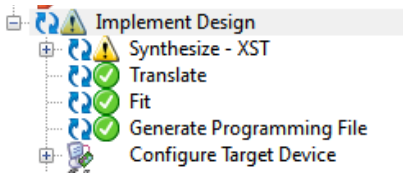
6. Wire up SW1…SW4 to inputs D0…D3. Connect SLI to ‘GND’, CE to ‘VCC’, and C to ‘CLK1HZ’. Also connect CLR to ‘GND’. Finally as outputs LED1 is connected to Q3, and the CLK is outputted onLED2. Your schematic should now look something like this:



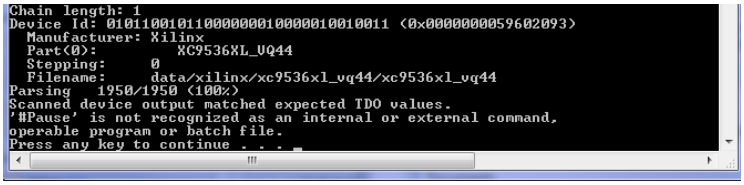
7. As before, save and close the schematic, then double-click the ‘Implement Design’ process:



8. You should get green Check-marks beside ‘Generate Programming File’:



9. Plug in your Digital Explorer board.
10. Run 'program.bat' as before, and again you should see an indication it is successful:

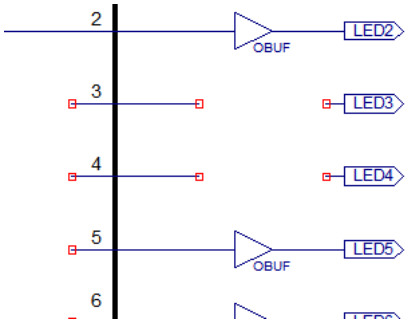


11. **Complete the 'Observations' section for the Parallel-to-Serial Converter**

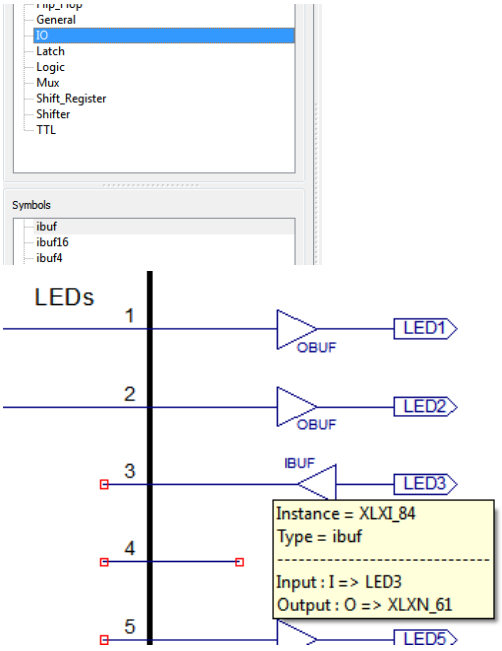
Part 1-B

NOTE: The diagrams here show you completing part 1-A, then part 1-B. You may complete part 1-B without doing part 1-A. For part 1-C you can use two separate digital trainer boards wired together instead of one board with both parts.

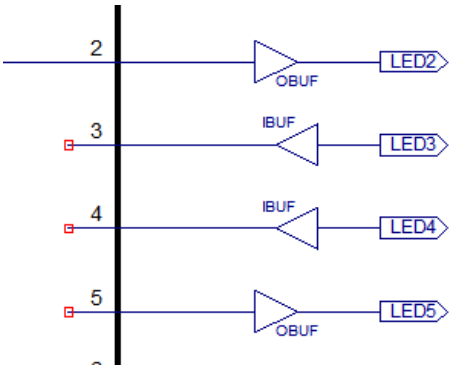
12. We will modify two of the LEDs to use those pins as INPUTS. Let's see how to modify the I/O direction. First select the 'OBUF' part beside LED3 and hit delete. Do the same thing with LED4.



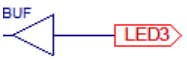
13. From the IO category place an 'ibuf' component. You will need to rotate the component with Ctrl-R to point INWARDS. Place this component as shown below:

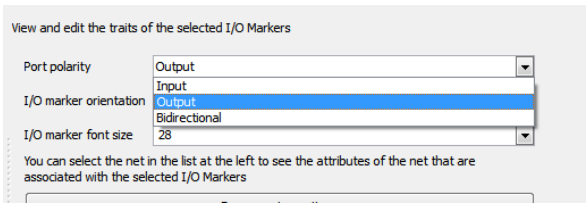


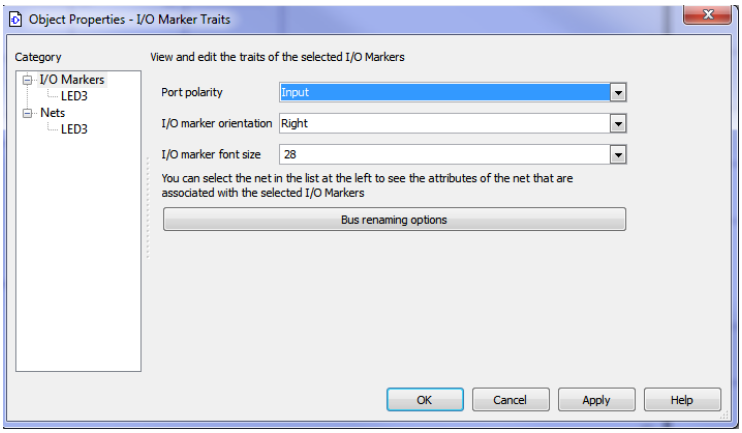
14. Repeat this for the second instance:



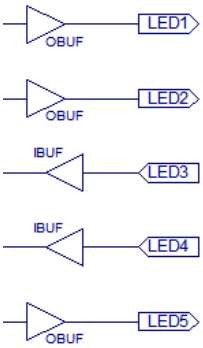
15. Now double-click on the ‘LED3’ port, and change the polarity to ‘Input’, and hit OK. Repeat with LED4.



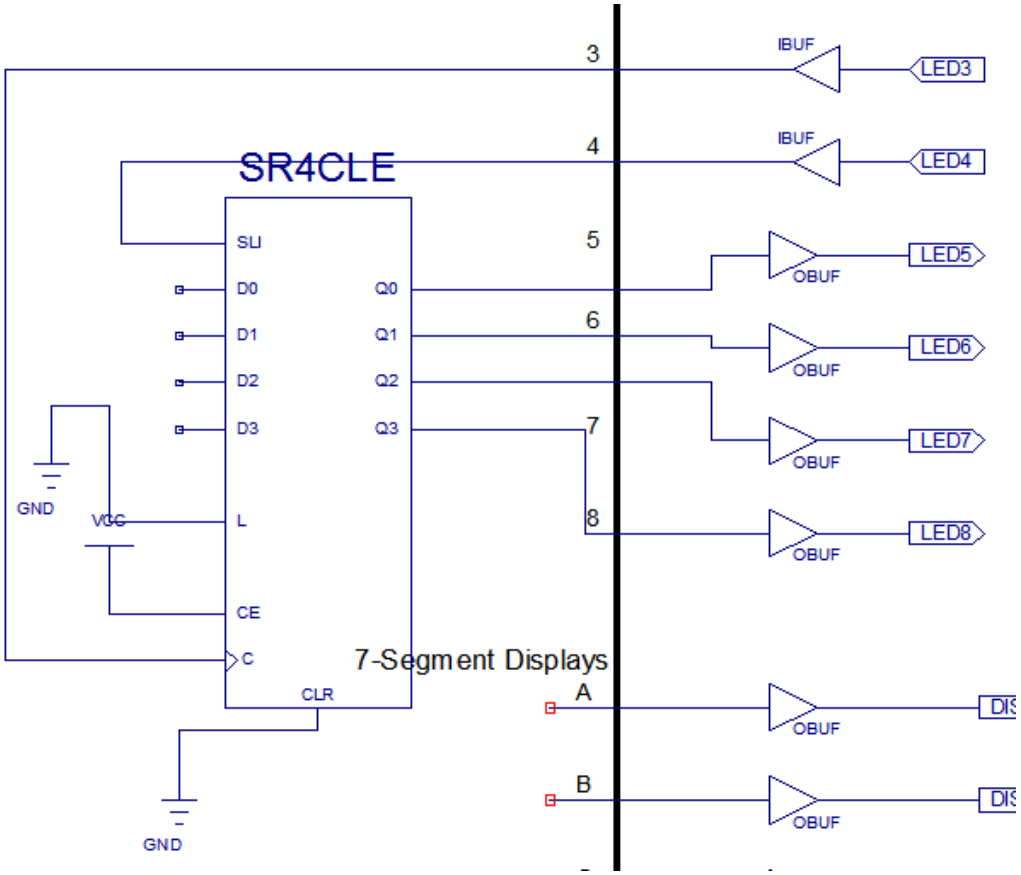




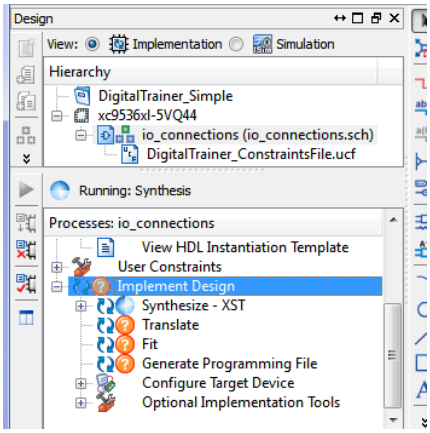
16. Your schematic should now look like this:



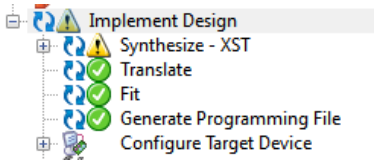
17. Place another SR4CLE part from the ‘Shift Register’ category
18. Wire up SLI to LED4, C to LED3, L to GND, CE to VCC, Q0…Q3 to LED5..LED8, and CLR to GND. The example schematic below shows this drawn where we also have the SR4CLE from part 1-A. Again you may ignore that if you have split the work in half, and one person is doing part 1-A and one person part 1-B.



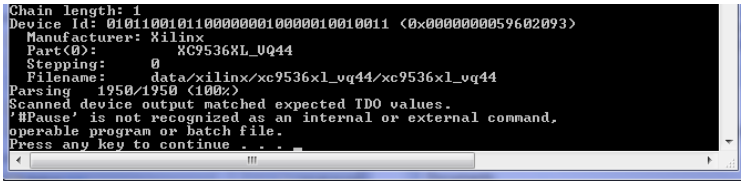
19. As before, save and close the schematic, then double-click the ‘Implement Design’ process:



20. If you get an error, check your schematic. In particular did you forget to switch one of the outputs to an input (LED3/LED4)? Look at the shape of the port, it should be pointing in.
21. You should get green Check-marks beside 'Generate Programming File':



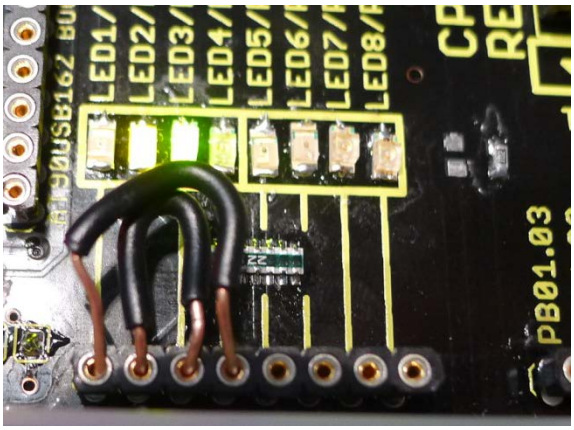
22. Plug in your Digital Explorer board.
23. Run 'program.bat' as before, and again you should see an indication it is successful:



24. **Complete the 'Observations' section for the Serial-to-Parallel Converter**

Part 1-C

25. Complete the schematic which has BOTH the serial-to-parallel (part 1B) and parallel-to-serial (part 1A) on it. Note both parts use different LEDs/Switches, so there is no 'conflict' of pins. Simply create a schematic that includes BOTH part 1A & part 1B by first drawing the connections in 1A, then the connections in 1B. As an alternative you may implement them on separate Digital Trainer boards.
26. Using wires connect the parallel-to-serial converter to the serial-to-parallel converter. If you implemented them on the same board wire LED2 to LED3 and LED1 to LED4 as shown below.



27. If you implemented each side on *different* boards: wire LED1 of the board containing Part1-A schematic to LED4 of the board containing Part1-B schematic. Wire LED2 of the board containing Part1-A schematic to LED3 of the board containing Part1-B schematic.
28. Complete the Observations for **Part 1-C**.

Observations

Part 1-A: Parallel to Serial Converter

Q1) The parallel to serial converter is running with a 1Hz clock. The shift register works as follows:

1. Set SW5 to 0 (right)
2. Set SW1/SW2/SW3/SW4 as desired (e.g. as shown in the table).
3. Set SW5 to 1 (left). When SW5 is set to '1', the data is continually being loaded into the shift register. It's only after we set SW5 to 0 (discussed next) that the shift register is allowed to operate, and shifts output the input we loaded.
4. Wait at least 1 pulses of LED2, then set SW5 to 0 (right) when LED2 is ON. The timing will require some practice.
5. LED1 has the serial output, and LED2 is the 'clock'. By counting pulses of LED2 you can see what the current bit being shifted out is. The location of clock pulse 1 is tricky – it's the initial state of LED1 when SW5 is still set to '1' typically. Thus the first pulse after setting SW5 to 0 will normally be clock pulse #2. You can record clock pulse #1 by recoding the state of LED1 after you put SW5 to 1. Once you put SW5 to 0 start counting clock pulse 2..3..4.. etc.

Using the above procedure, load the following numbers into the shift register. Write the state of LED1 on each clock pulse. The first row is filled in for you, use it to validate you're reading the clock pulses correctly.

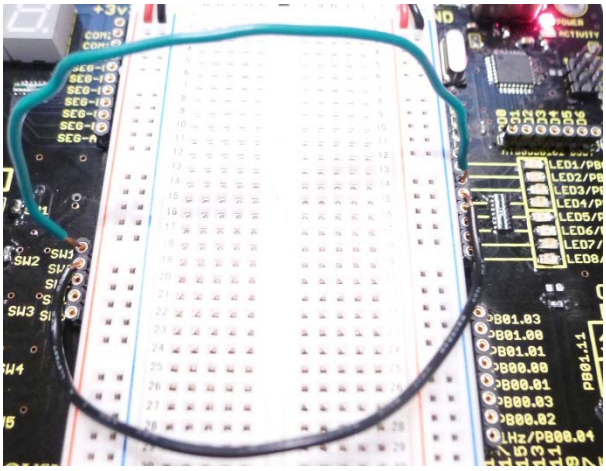
Setting of Input Switches				State of LED1 on Clock Pulse #				
SW1	SW2	SW3	SW4	1	2	3	4	5
0	0	0	1	1	0	0	0	0
1	0	0	0					
1	1	1	0					
1	0	1	1					
1	0	1	0					
1	1	1	1					

Q2) Assume I input a number where SW1 = MSB, SW4 = LSB (e.g.: in the table above you are inputting 1,8,14,11,10). Which bit is being shifted out first of this shift register you have implemented: MSB or LSB?

Q3) After you set SW5 to 0, does it matter if you change the state of SW1-SW4? As an example load the 1-1-1-1 state. After you set SW5 low, set SW1-SW4 (or just one of them if it's easier) to 0. Does it change the output?

Part 1-B: Serial to Parallel Converter

Using wires, connect SW1 to LED3 and SW2 to LED4 as shown below:



SW1 is now the CLOCK input, and SW2 is now the DATA input. Note: if you have done the wires correctly, LED3 should output the same as the SW1 (clock) position, and LED4 should output the same as the SW2 (data) position.

Q1) Complete the following truth table by moving down the table, and filling in the state of Q0/Q1/../Q3. Note that due to switch bounce your results may not be completely predictable.

SW1 (clk)	SW2 (data)	LED5 (Q0)	LED6 (Q1)	LED7 (Q2)	LED8 (Q3)
0	0				
0	1				
1	1				
0	1				
1	1				
0	1				
1	0				
0	0				
1	0				
0	0				
0	0				
0	1				
1	1				
0	0				
1	0				
0	0				
1	0				
0	0				

Part 1-C: Connected

The serial-to-parallel and parallel-to-serial converters are now connected together. As in part 1-A you load data with this procedure:

1. Set SW5 to 0 (right)
2. Set SW1/SW2/SW3/SW4 as desired
3. Set SW5 to 1 (left), wait until LED5 illuminates, then set SW5 to 0. You must do this quickly or you will end up with too many 1's shifted in!
4. LED1 has the serial output, and LED2 is the 'clock'. By counting pulses of LED2 you can see what the current bit being shifted out is. Note that normally clock pulse '1' is while load (SW5) is still HIGH. Thus after setting SW5 to 0, the next pulse of LED2 is clock pulse #2.
   (Note LED2 and LED3 should both be blinking at 1 Hz)
5. Data will also be shifted into LED5…LED8.

Q1) Load the following number, then record the state of LED5/LED6/LED7/LED8 after each clock pulse. You may need to try several times to catch all states if you miss a clock pulse. Based on your knowledge of a shift register, you should have some idea where the 'correct' output should be. As a hint: you are connecting a parallel-to-serial converter to a serial-to-parallel converter. The objective is to send the states of SW1,SW2,SW3,SW4 to the LEDs, but we are using only two wires instead of four!

Number: SW1 = 0, SW2 = 0, SW3 = 0, SW4 = 1

Clock Pulse #	LED5	LED6	LED7	LED8
1				
2				
3				
4				
5				
6				
7				
8				

Number: SW1 = 1, SW2 = 1, SW3 = 0, SW4 = 1

Clock Pulse #	LED5	LED6	LED7	LED8
1				
2				
3				
4				
5				
6				
7				
8				

Q2) At what clock output is the data 'valid', that is to say LED5…LED8 represents the state of SW1..SW4.

Q3) What problems does this design have, how might you improve them?

Conclusion

This lab has introduced many facets of serial communication. You have implemented several synchronous serial data transfer blocks.