## 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 in the procedure.

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)

1. 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?