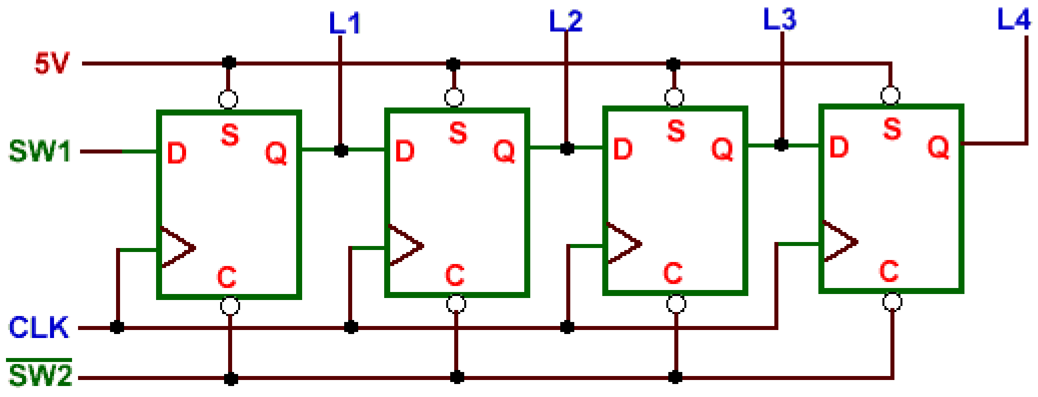
Name:
Muhammad Bilal Elahi
ID:
15434
Department:
Computer science
Submitted to:
Sir Muhammad Amin

Shift Registers

Introduction

A shift register is an n-bit register with provision for shifting its stored data by one position at each clock pulse. The logical configuration of a shift register consists of a chain of flip-flops connected in cascade, with the output of one flip-flop connected to the input of the next flip-flop. All flip-flops receive a common clock pulse which causes the shift from one stage to the next. Fig. 1 shows a simple shift register configuration. The new bit to be shifted into one end must be specified, and the bit shifted off the other end is lost unless it is saved externally. Although Fig. 1 shows a right-shift register, the same register can obviously be used for left shifts simply by reversing the sense of the bits. Most shift registers have provision for shifting only in one direction, but some have a control input that allows either left or right shifting to be specified at each clock.



One way to load *n* bits of data into the flip-flop chain is to load the data one bit each clock cycle using the serial input. Some shift registers also have parallel inputs that can be used to load all *n* bits in one clock cycle. The output of a shift register can be observed one bit at a time at the serial output, but some shift registers also have parallel outputs for observing all *n* bits at once.

Shift registers are classified according to three basic considerations: their method of data handling (serial-in serial-out, serial-in parallel-out, and parallel-in serial-out), their direction of data movement (shift right, shift left, and bidirectional), and their bit length. One of the important applications of shift register circuits is in serial computation. Compared to parallel computation, where all bits in a word are processed at the same cycle, serial computation process words in one bit per cycle. Therefore, serial computation is slower, but it has the advantage of requiring less hardware and wiring. A serial adder will be built in this experiment as an example.

2. Objectives

Students are expected to understand various data handling methods in shift registers and their usage.

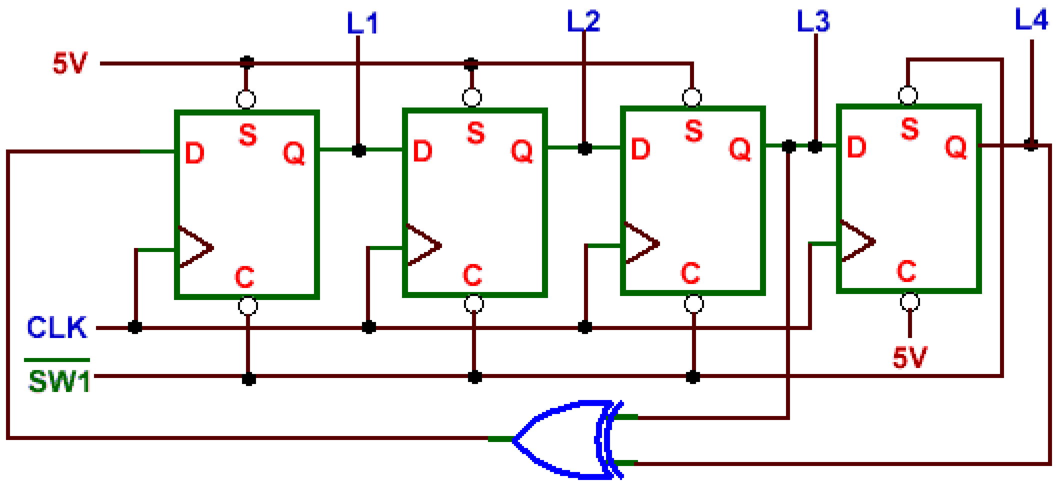
3. Experiment

3.1. 4-bit Shift Register

Use two 7474 dual flip-flops to connect a serial-in, parallel-out shift register as shown in Fig. 1. Connect L1–L4 to four LEDs (with current-limiting resistors), SW1 and $\boxed{\text{SW2}}$ to switches, and CLK to a pulser. Initially, set $\boxed{\text{SW2}}$ to logic 1. Switch $\boxed{\text{SW2}}$ being at logic 1 clears all flip-flops. Now set SW1 to logic 1 and $\boxed{\text{SW2}}$ to logic 0. Push the pulser button several times to allow more logic 1 to be shifted into the shift register. Change SW1 to logic 0 and repeat the experiment again.

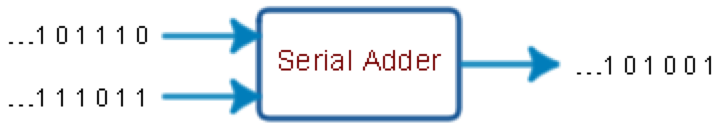
3.2. Pseudo-random Sequence Generator

Now use the above circuit to build a pseudo-random binary sequence generator as shown in Fig. 2. This binary sequence generator will display a random output (repeats every 2^n–1 bits, where n is the number of flip-flops used in the shift register). The IC 7486 provides the exclusive-OR needed in the circuit. To start the sequence generator, set the initial state of the shift register to 0001 by setting the switch $\boxed{\text{SW1}}$ to logic 1. Then change $\boxed{\text{SW1}}$ to logic 0 as this will release the control input. Now apply the clock and record the output in a table. Does the output show randomness? Does the output repeat after 15 pulses?



Serial Adder

A serial adder adds bits adds a pair of binary numbers serially with a simple full adder. The carry out of the full adder is transferred into a D flip-flop and the output of this carry flip-flop is then used as the input carry for the next pair of significant bits. Fig. 3 illustrates an example of serial addition.



To perform the addition, the following steps are used.

1. Shift the first 8-bit augend into A (remember to shift in LSB first) and addend into B. This is done by performing the following steps:

    i. Clear registers A and B, and the D flip-flop. The registers and flip-flop have a *clear* input that can be used to accomplish this task.

    ii. Shift the augend through the serial input. For 8-bit registers, one bit will be shifted into register B for each of eight clock cycles, least significant bit first. (Since registers A and B initially contain all zeroes, the output of the full adder will be 0 for both the sum and carry outputs during these clock cycles. You can either load these zeroes into register A or just leave A unchanged. In either case, after eight clock cycles, Register B will contain one operand and register A will contain all zeroes.

iii. Shift the addend through the serial input. This is actually two tasks performed at once. While the addend is being shifted into register B, the augend is being shifted out of B and into register A. To understand how this occurs, recall that register A contains all zeroes. Whatever bit is being shifted out of B is being added to the zero in the least significant bit of A. The sum will just be the least significant bit of B, which is shifted into the most significant position of register A. After eight clock cycles, register A contains the augend and register B contains the addend.

2. Clear the D flip-flop and run the adder for eight cycles to obtain the sum.

Serial Input	Register A	Register B	D FF	Comments
X	00000000	00000000	0	Clear A, B, D
1	00000000	10000000	0	
0	00000000	01000000	0	
1	00000000	10100000	0	
1	00000000	11010000	0	
1	00000000	11101000	0	
0	00000000	01110100	0	
1	00000000	10111010	0	
0	00000000	01011101	0	Augend in B; 0 in A
0	10000000	00101110	0	0+1=1

Conclusion:

Shift register is verified.