Unit 6 Shift Registers and Applications

Structure:

6.1 Introduction

In the previous unit, we studied about the definition of sequential circuit, latches and flip flops and various types of flip-flops. We also studied about real world applications of sequential circuits. Historically a shift register was built as early as in 1940's. It was a code-breaking machine, having five-stages and was built using vacuum tubes. Shift registers are typically used as converters between serial and parallel devices. Most of the data processing logic works on set of bits together. An example can be the CPU of a computer that works on data bits stored in the registers. Hence mapping serial stream of data into parallel data bytes is really useful. At the same time it is easier to implement serial data transmission logic. We can think of shift registers as delay elements. We can implement stack of a CPU in hardware using bi-directional shift registers. Unlike mono-shots (mono-stable multi-vibrators), the accuracy of timing offered by shift registers is independent of its component values. Hence pulse extenders can be realized using them. However the timing properties of shift register are defined and limited by the clock signal supplied to it.

Architectures of old computers involved shift registers in the CPU for holding and moving the data into the ALU (Arithmetic Logic Unit). The two operands of addition were first shifted into the shift registers and then shifted out into

ALU for performing addition. The answer was again moved back to one of the shift register, generally known as the Accumulator.

Most of the CPU's have machine level instructions that can move the bits stored in a shift register to the left or right in a group. These instructions are 'shift left' and 'shift right'. Higher level languages also provide constructs that make use of these instructions. As we know from binary arithmetic, left shifting of data multiplies it by two and right shift of it divides the same by two.

Historical computers used very large sized shift registers (serial-in – serial-out type), having sizes of few thousands of bits. These worked like delay line memory elements. In this unit we study about the definition of shift registers, its types and their operation. We also study some shift register ICs.

Objectives:

By the end of Unit 6, the learners are able to:

- define shift registers.
- list and explain different types of Shift Registers
- discuss on IC 74LS395

6.2 Definition of Shift Registers

Shift Register is a set of binary storage elements, typically flip-flops combined and linked together to facilitate the movement of the data bits stored, from one to another and in and out of it, whenever desired by activating control signals.

Inputs to the shift registers can be serial or parallel. Similarly outputs of the shift registers can be serial or parallel. Thus we can have four types of shift registers based on the serial or parallel nature of inputs and outputs. They are listed below.

- Serial Input Serial Output (SISO)
- Serial Input Parallel Output (SIPO)
- Parallel Input Serial Output (PISO)
- Parallel Input Parallel Output (PIPO)

There are bi-directional shift registers that allow shifting of data bits in both directions, i.e. from left to right and vice versa. If we connect inputs and

outputs of a serial-in serial-out shift register, we get so called **circular shift register**.

6.3 Types of Shift Registers

Now let us study the different types of shift registers.

6.3.1 Serial-In Serial-Out (SISO)

Operationally this is one of the simple types. As the name suggests, data bits are stored in serially and in the same way these get out of the shift register serially. This concept is shown in figure 6.1.

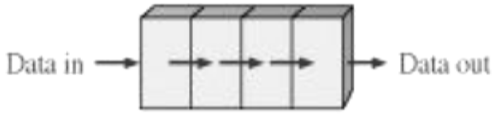

Figure 6.1: Serial-In Serial-Out (SISO)

The string of bits that we want to shift in are given to the input pin named 'Data In'. Each bit presented at 'Data In' is shifted to its right one flip-flop at a time, every time 'Data Shift' signal is enabled. First time, the bit on 'Data In' line is moved into the first 'flip-flop's output. The data bit on the rightmost flip-flop gets shifted out through the output line of the shift register 'Data Out'. The bit that goes out through 'Data Out' is lost. The figure 6.2 shows the 4-bit Serial-In Serial-Out (SISO) Shift Register.

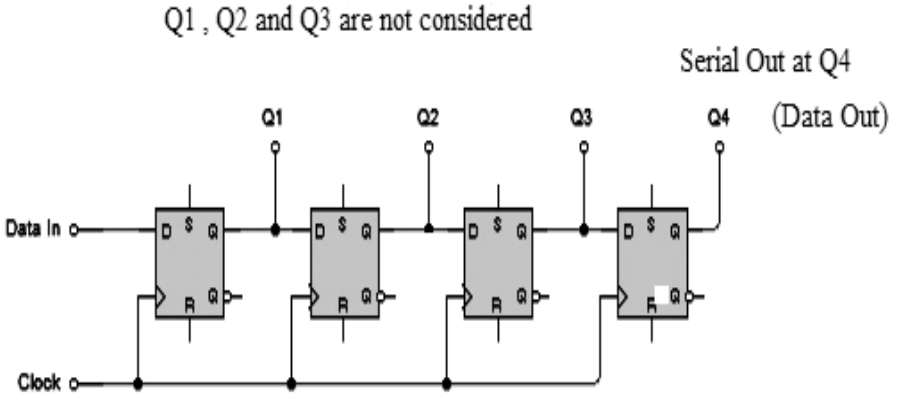

Figure 6.2: 4-bit Serial-In Serial-Out (SISO) Shift Register

Let us see more clearly how this works. Each row of four bits in the table 6.1 can be imagined to be the outputs of the four flip-flops that make up our SISO (serial-in serial-out) shift register.

Table 6.1: Truth table of SISO shift register

Clock Pulse	Data In	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0	0
1	**1 (LSB)**	1	0	0	0
2	**0**	0	1	0	0
3	**1**	1	0	1	0
4	**1 (MSB)**	1	1	0	**1**
5	0	0	1	1	**0**
6	0	0	0	1	**1**
7	0	0	0	0	**1**
8	0	0	0	0	0

Let the register is reset so that all the flip-flop outputs to 0. We now feed a stream of bits (LSB) 1, 0, 1 ,1 (MSB) at 'Data In' line and at the same time pulse the 'Data Shift' signal for every data bit. We see that data is right shifted for every pulse and after four pulses the register is filled with 1101 from left to right. If we clock further, the data can be read out from 'Data Out' with simultaneous right shift of data bits as shown in the table 6.1. From the clock pulse 5^{th} onwards, we assumed that the *data In* takes the 0 as its input data.

If we collect all the data bits at 'Data Out' line after the register is filled with the serial data input, we get 10110000 as the bit stream. So what we get at the output is what we had put in through the input. However the output is delayed by four cycles of clocking or pulsing the 'Data Shift' line. The data movement though SISO shift register resembles a queue implemented in hardware. To start over again, the register flip-flops can be set to 0 by asserting the 'Reset' input. Since the data bits that are shifted out of 'Data Out' are lost, we call this *destructive readout*.

Non-destructive readout

It is possible to ensure reading the data out from a SISO shift register is non-destructive. All we have to do is to connect the 'Data Out' line to

'Data In' line under the control of an additional input line. Let us call this 'Read/Write (R/W)' line. When this input is set to 1(or Write), it behaves like a normal SISO register with destructive readout. When R/W is set to 0 (Read), 'Data Out' gets logically connected to 'Data In' and the data bits that get shifted out upon pulsing the 'Data Shift' line become the inputs and get shifted in again. Thus the data is retained in the shift register.

6.3.2 Serial-In Parallel-Out (SIPO)

This is almost similar to SISO shift register discussed in the section 6.3.1 except that the data are readout in parallel at the same time. this means, We can input the data bits into this shift register serially via 'Data In' input line and data can be read out in parallel from data out lines. The concept of SIPO is shown in figure 6.3.

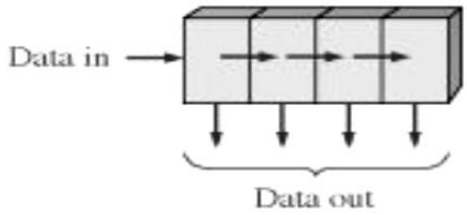

Figure 6.3: Serial-In Parallel-Out (SIPO)

Note that we may readout the data in two ways.

In the first method, data is readout in parallel from the flip-flop outputs Q1 through Q4. Hence the name 'Parallel Out'. Alternatively we can do destructive readout serially via 'Data Out' or Q4 line and replace it with new data if presented at 'Data In'. The figure 6.4 shows the 4-bit Serial-In Parallel-Out (SIPO) Shift Register and table 6.2 shows its truth table.

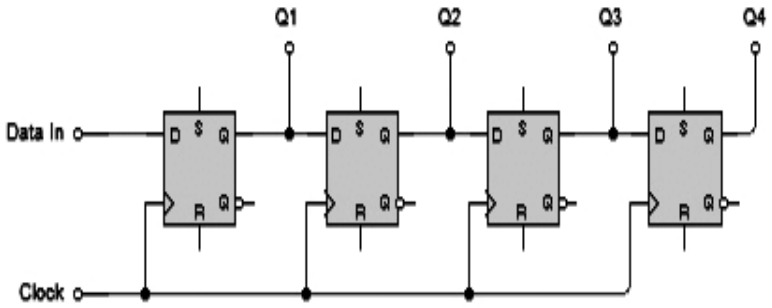

Figure 6.4: 4-bit Serial-In Parallel-Out (SIPO) Shift Register

Table 6.2: Truth table of SIPO shift register

Clock Pulse	Data In	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0	0
1	**1 (LSB)**	1	0	0	0
2	**0**	0	1	0	0
3	**1**	1	0	1	0
4	**1 (MSB)**	**1**	**1**	**0**	**1**

6.3.3 Parallel-In Serial-Out (PISO)

In Parallel-In, Serial-Out (PISO) shift register, the data input is given in parallel to the input line of each of the flip-flops and outputs are readout serially from single output line (Data Out) as shown in figure 6.5.

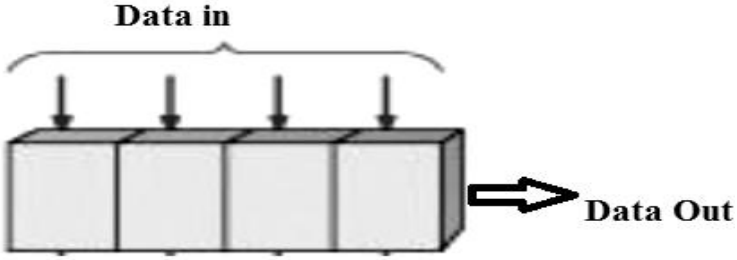

Figure 6.5: Parallel-In Serial-Out (PISO)

The figure 6.6 shows the 4-bit Parallel-In Serial-Out (PISO) Shift Register and table 6.3 shows its truth table. In this type, the data input can be given in parallel to the input line of each of the flip-flops, D1 through D4. To switch between data-input and data-shift mode, we use a signal 'Write/Shift' (W/S).

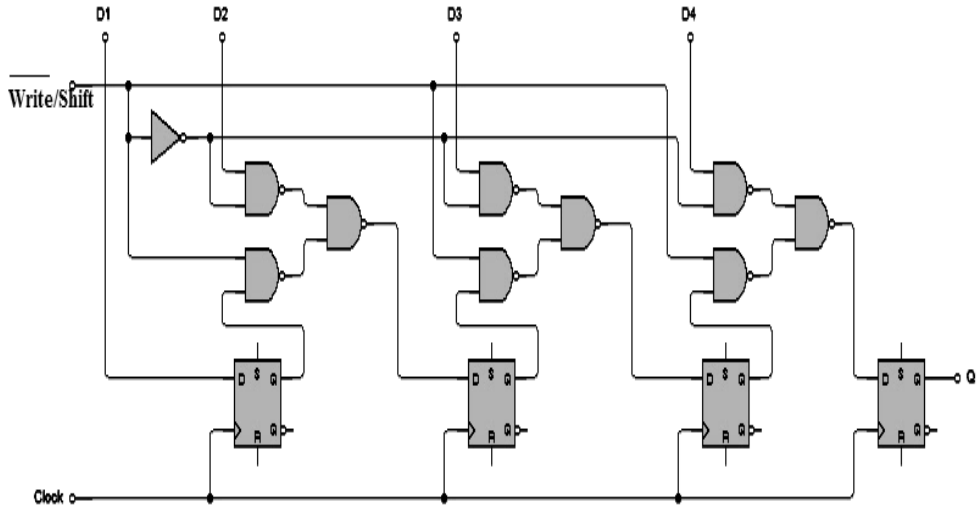

Figure 6.6: 4-Bit PISO Shift Register

When we set W/S input to 0, the data bits on D1-D4 lines get loaded to the outputs of the respective flip-flops, Q1-Q4. For shifting the data out, we set W/S to 1. Now the shift register works much like a SISO register and D1 acts like 'Data In' line. For the first few cycles, the number of which is equal to number of flip-flops in the register, the data bits readout of Q4 represent the parallel data that was read in through D1-D4 inputs.

Table 6.3: Truth table of PISO shift register

Clock Pulse	Parallel Data $D_1D_2D_3D_4$	Q_1	Q_2	Q_3	Q_4
0	0 0 0 0	0	0	0	0
1	**1 0 1 1**	**1**	**0**	**1**	**1**
2	X X X X	x	1	0	**1**
3	X X X X	x	x	1	**0**
4	X X X X	x	x	x	**1**

X= 0 or 1

6.3.4 Parallel-In Parallel-Out (PIPO)

The PIPO register is mainly used to shift a given set of bits and present it to the next stage as illustrated below. Input for shifting can be loaded into the register in parallel and the shifted output can be read out of the register in

parallel as well. Hence the name PIPO shift register. The concept of PIPO is shown in figure 6.7.

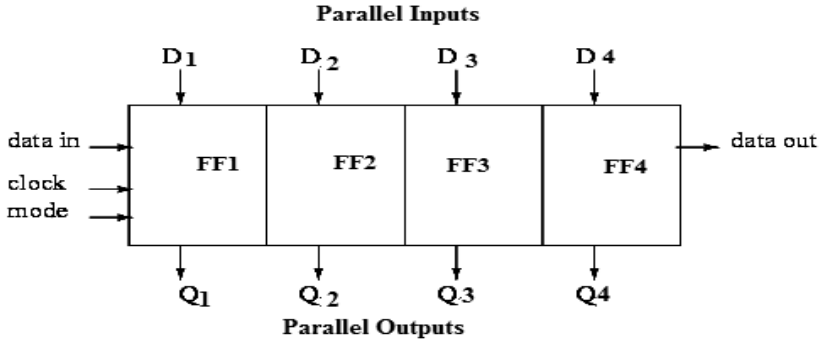

Figure 6.7: PIPO Shift Register

In the above schematic diagram of PIPO shift register, we load the data at D_1-D3 inputs. The 'mode' input is used to swap between 'parallel load' and 'shift' operation. Additional control pins may be provided for changing the direction of the data shift. Once the 'mode' input is set to 'shift' the loaded data can be shifted upon supplying the clock signal. The shifted data can be obtained at the outputs Q_1-Q3. The 'data in' and 'data out' lines can be used to enable serial input and serial output operation. They can also be used for cascading additional stages. Shift register shown in figure 6.7 can also be used to input/output data serially. So we can call this a universal shift register.

The figure 6.8 shows the 4-bit Parallel-In Parallel-Out (PIPO) Shift Register and table 6.4 shows its truth table.

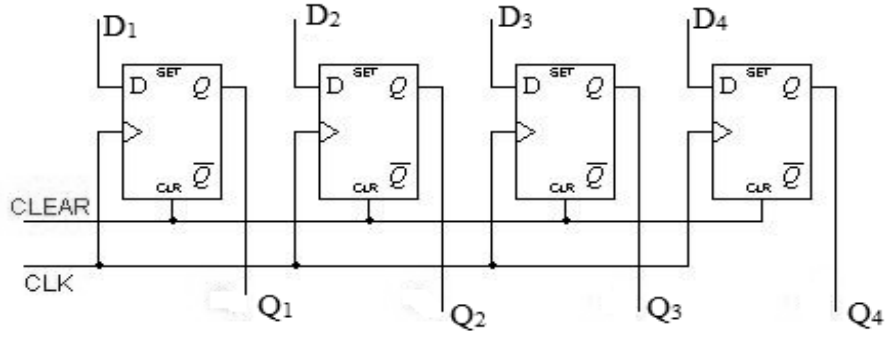

Figure 6.8: 4-bit Parallel-In Parallel-Out (PIPO) Shift Register

Table 6.4: Truth table of PIPO shift register

Clock Pulse	Parallel Input Data $D_1D_2D_3D_4$	Q_1	Q_2	Q_3	Q_4
1	**1 0 1 1**	**1**	**0**	**1**	**1**

6.4 Case Study IC's

The following schematic diagram shown in figure 6.9 depicts the hardware details of a PIPO shift register. The register shifts the data towards right. The buffers shown are necessary for real physical IC that is used with other components.

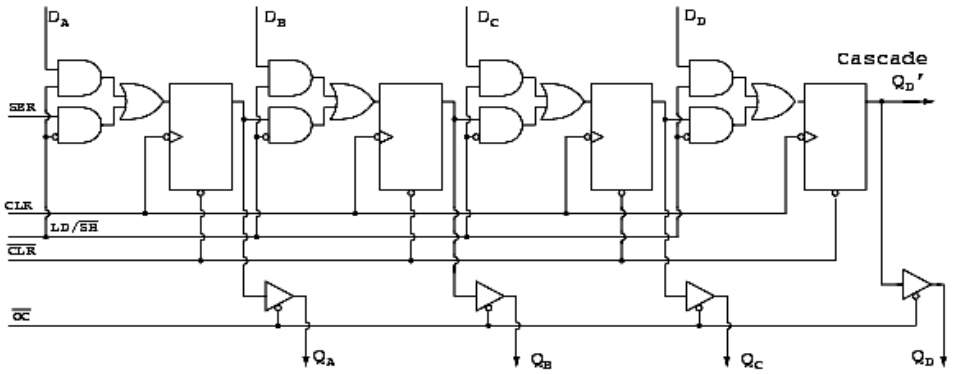

Figure 6.9: 74LS395 Parallel-In Parallel-Out Shift Register with Tri-state output

The 74LS395 IC resembles very much like our theoretical PIPO shift register that shifts data right. The schematic diagram shown above is simplified version of that found in the data sheet. You may refer to the detailed information.

LD/SH' controls the AND-OR multiplexer at the data input to the FF's. If LD/SH'=1, the upper four AND gates are enabled allowing application of parallel inputs D_A D_B D_C D_D to the four FF data inputs. Note the inverter bubble at the clock input of the four FFs. This indicates that the 74LS395 clocks data on the negative going clock, which is the high to low transition. The four bits of data will be clocked in parallel from D_A D_B D_C D_D to Q_A Q_B Q_C Q_D at the next negative going clock. In this "real part", OC' must be low if the data needs to be available at the actual output pins, as opposed to only on the internal FFs.

The previously loaded data may be shifted right by one bit position if LD/SH'=0 for the succeeding negative going clock edges. Four clocks would shift the data entirely out of our 4-bit shift register. The data would be lost unless our device was cascaded from QD' to SER of another device. The figure 6.10 shows the loading and shifting of parallel data.

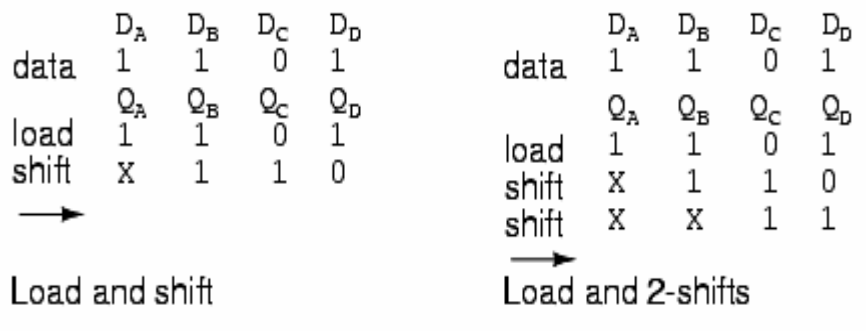

Figure 6.10: Parallel-in/ Parallel-out shift register

Above, a data pattern is presented to inputs D_A D_B D_C D_D. The pattern is loaded to Q_A Q_B Q_C Q_D. Then it is shifted one bit to the right. The incoming data is indicated by X, meaning that we do no know what it is. If the input (SER) were grounded, for example, we would know what data (0) was shifted in. Also shown, is right shifting by two positions, requiring two clocks.

The figure 6.11 serves as a reference for the hardware involved in right shifting of data.

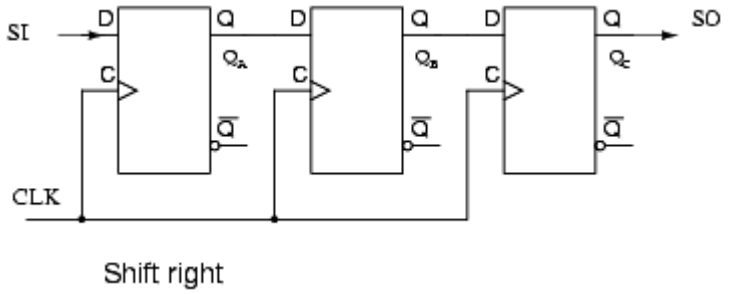

Figure 6.11: Right shifting

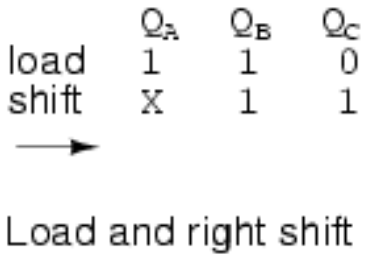

Figure 6.12: Right shifting of data.

Right shifting of data is shown in figure 6.12 and is provided for reference to the previous right shifter shown in figure 6.11.

The figure 6.13 serves as a reference for the hardware involved in left shifting of data.

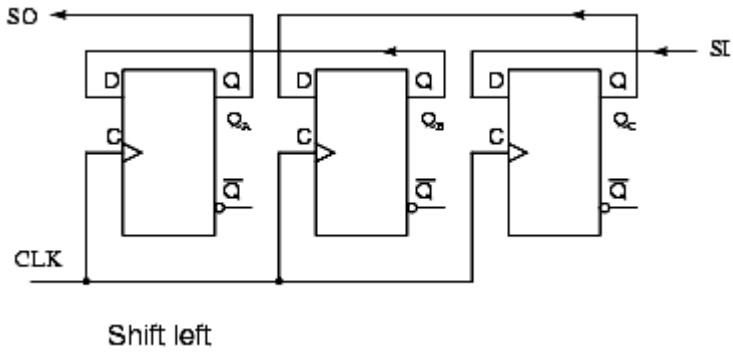

Figure 6.13: Left shifting of data.

If we need to shift left, then the FFs need to be rewired. Compare to the previous right shifter, SI and SO have been reversed. SI shifts to QC. QC shifts to QB. QB shifts to QA. QA leaves on the SO connection, where it could cascade to another shifter SI. This left shift is backwards from the right shift sequence as shown in figure 6.14.

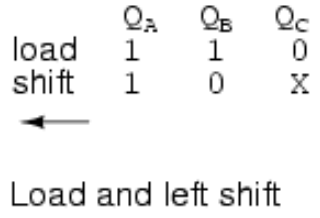

Figure 6.14: Left shift operation

Above we shift the same data pattern left by one bit. The issue with the above arrangement is that, it is not much in demand in the market. Hence it is hardly manufactured. We can reverse the shifting direction by appropriately wiring a physical shift register IC externally. Thus the direction of shift loses significance in a way. However better arrangement would be to bring the desired direction of shift under the control of an input signal as shown in figure 6.15.

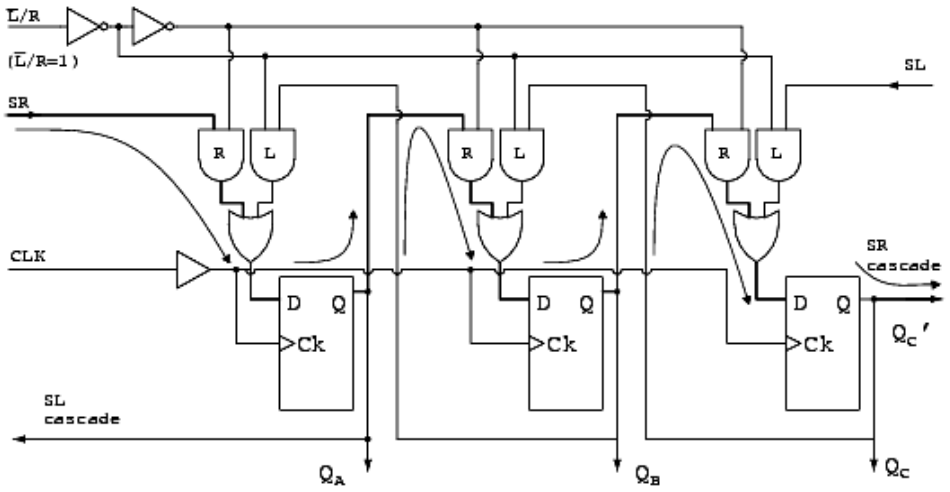

Figure 6.15: Left shift /right register, right action

What we have above is a hypothetical shift register capable of shifting either direction under the control of L'/R. It is setup with L'/R=1 to shift the normal direction, right. L'/R=1 enables the multiplexer AND gates labeled R. This allows data to follow the path illustrated by the arrows, when a clock is applied.

Data shifts in at SR, to Q_A, to Q_B, to Q_C, where it leaves at SR cascade. This pin could drive SR of another device to the right.

What if we change L'/R to L'/R=0?

With L'/R=0, the multiplexer AND gates labeled L are enabled, yielding a path, shown by the arrows in figure 6.16.

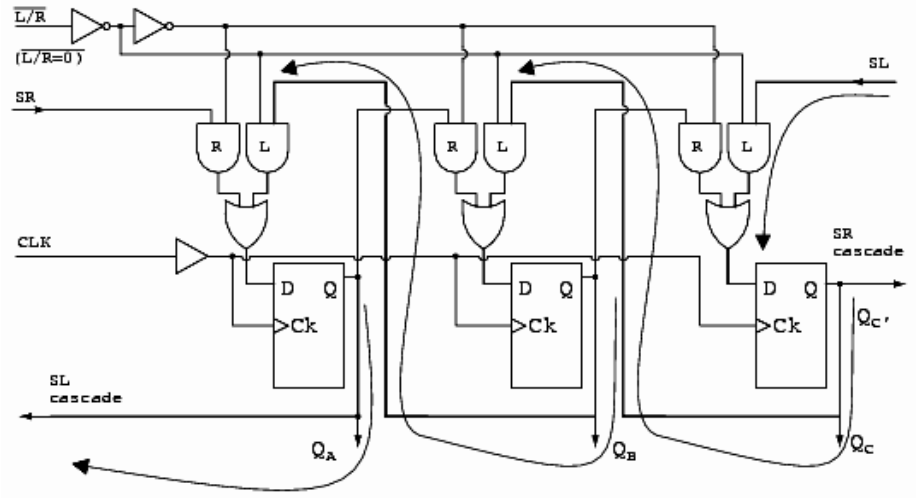

Figure 6.16: Left shift /right register, left action

Data shifts in at SL, to Q_C, to Q_B, to Q_A, where it leaves at SL cascade. This pin could drive SL of another device to the left.

The prime virtue of the above two figures 6.15 and 6.16 illustrating the "shift left/ right register" is simplicity. The operation of the left right control L'/R=0 is easy to follow. A commercial part needs the parallel data loading implied by the section title. This appears in the figure 6.17 shown below.

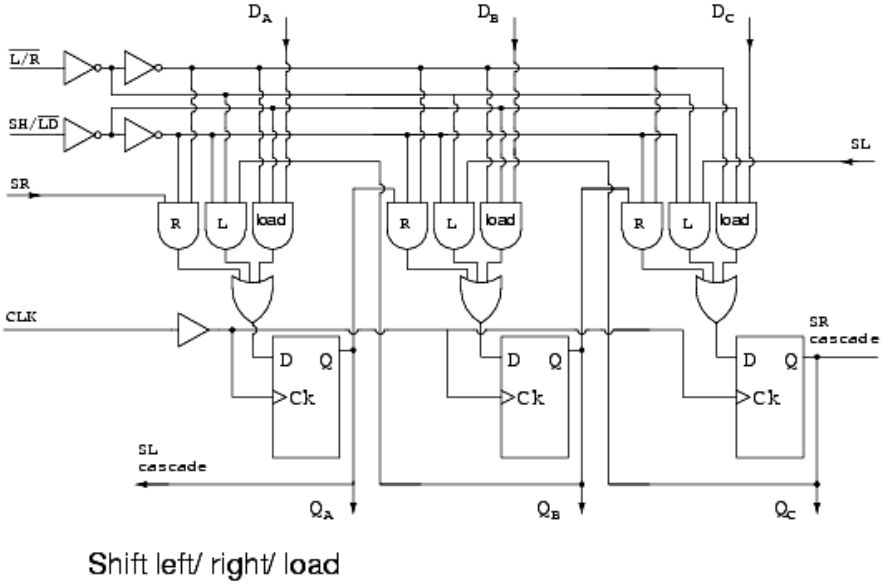

Figure 6.17: Shift left/right/ the parallel data loading

Now that we can shift both left and right via L'/R, let us add SH/LD', shift/ load, and the AND gates labeled "load" to provide for parallel loading of data from inputs D_A D_B D_C. When SH/LD'=0, AND gates R and L are disabled, AND gates "load" is enabled to pass data D_A D_B D_C to the FF data inputs. The next clock CLK will clock the data to Q_A Q_B Q_C. As long as the same data is present it will be re-loaded on succeeding clocks. However, data present for only one clock will be lost from the outputs, when it is no longer present on the data inputs. One solution is to load the data on one clock, and then proceed to shift on the next four clocks. This problem is remedied in the 74ALS299 by the addition of another AND gate to the multiplexer.

If SH/LD' is changed to SH/LD'=1, the AND gates labeled "load" are disabled, allowing the left/ right control L'/R to set the direction of shift on the L or R AND gates. Shifting is as in the previous figures.

The only thing needed to produce a viable integrated device is to add the fourth AND gate to the multiplexer as alluded for the 74ALS299.

6.5 Summary

Let us recapitulate the important concepts discussed in this unit.

- Shift Register is a set of binary storage elements, typically flip-flops combined and linked together to facilitate the movement of the data bits stored, from one to another and in and out of it, whenever desired by activating control signals.
- Four types of shift registers are: Serial Input Serial Output (SISO), Serial Input Parallel Output (SIPO), Parallel Input Serial Output (PISO) and Parallel Input Parallel Output (PIPO).
- In SIPO, we can input the data bits into this shift register serially via ‘Data In’ input line and data can be read out in parallel from data out lines.
- In Parallel-In, Serial-Out (PISO) shift register, the data input is given in parallel to the input line of each of the flip-flops and outputs are readout serially from single output line.
- In Parallel-In, Parallel-Out (PIPO) shift register, the data input is given in parallel to the input line of each of the flip-flops and outputs are readout in parallel from all the output lines.

Self Assessment Questions

1. The storage elements in shift registers are _____.
2. There are _____ types of shift registers.
3. When 'Data Out' is fed back to 'Data In' of a SISO shift register, It becomes _____shift register.
4. In _____ shift register, we can input the data bits into this shift register serially via 'Data In' input line and data can be read out in parallel from data out lines.
5. In_____shift register, input for shifting can be loaded into the register in parallel and the shifted output can be read out of the register in parallel as well.
6. _____ IC is Parallel-In Parallel-Out Shift Register with Tri-state output.

6.6 Terminal Questions

1. What is a shift register? Mention the types of shift register.
2. Draw and explain 4-bit serial-in-parallel-out shift register.
3. Draw and explain the operation of parallel-in-parallel-out shift register.
4. Discuss on 74LS395 IC

6.7 Answers

Self-Assessment Questions

1. Flip-Flops
2. Four
3. Circular.
4. SIPO (Serial-In Parallel-out)
5. PIPO (Parallel-In Parallel-out)
6. 74LS395

Terminal Questions

1. Refer Section 6.3
2. Refer Section 6.3
3. Refer Section 6.3
4. Refer Section 6.4