REGISTERS

We have studied combinational functional blocks, and then we examined sequential circuits. Now, we bring the two ideas together and present sequential functional blocks, generally referred to as *registers* and *counters*. The sequential circuits that were analyzed or designed did not have any particular structure, and the number of flip-flops was small. In contrast, the circuits we consider here have more structure, with multiple stages or cells that are identical or close to identical, making expansion very simple. *Registers* are particularly useful for storing information during the processing of data, and *counters* assist in sequencing the processing.

REGISTERS AND LOAD ENABLE

A register includes a set of flip-flops. Since each flip-flop is capable of storing one bit of information, an *n*-bit register, composed of *n* flip-flops, is capable of storing *n* bits of binary information. By the broadest definition, a *register* consists of a set of flip-flops, together with gates that implement their state transitions. This broad definition includes the various sequential circuits considered earlier. More commonly, the term *register* is applied to a set of flip-flops, possibly with added combinational gates, that perform data-processing tasks. The flip-flops hold data, and the gates determine the new or transformed data to be transferred into the flip-flops.

A *counter* is a register that goes through a predetermined sequence of states upon the application of clock pulses. The gates in the counter are connected in a way that produces the prescribed sequence of binary states. Although counters are a special type of registers, it is common to differentiate them from registers.

Registers and counters are sequential functional blocks that are used extensively in the design of digital systems in general and in digital computers in particular. Registers are useful for storing and manipulating information; counters are employed in circuits that sequence and control operations in a digital system.

The simplest register is one that consists of only flip-flops without external gates. Figure 1(a) shows such a register constructed from four *D*-type flip-flops. The common *Clock* input triggers all flip-flops on the rising edge of each pulse, and the binary information available at the four *D* inputs is transferred into the 4-bit register. The four *Q* outputs can be sampled to obtain the binary information stored in the register. The ***Clear*** input goes to the ***R*** inputs of all four flip-flops and is used to clear the register to all 0s prior to its clocked operation. This input is labeled ***Clear*** rather than *Clear*, since a 0 must be applied to reset the flip-flops asynchronously. Activation of the asynchronous ***R*** inputs to flip-flops during normal clocked operation can lead to circuit designs that are highly delay dependent and that can, therefore, malfunction. Thus, we maintain ***Clear*** at logic 1 during normal clocked operation, allowing it to be logic 0 only when a system reset is desired. We note that the ability to clear a register to all 0s is optional; whether a clear operation is provided depends upon the use of the register in the system.

The transfer of new information into a register is referred to as *loading* the register. If all the bits of the register are loaded simultaneously with a common clock pulse, we say that the loading is done in parallel. A positive clock transition applied to the *Clock* input of the register of Figure 1(a) loads all four *D* inputs into the flip-flops in parallel.

Figure 1(b) shows a symbol for the register in Figure 1(a). This symbol permits the use of the register in a design hierarchy. It has all inputs to the logic circuit on its left and all outputs from the circuit

on the right. The inputs include the clock input with the dynamic indicator to represent positive-edge triggering of the flip-flops. We note that the name *Clear* appears inside the symbol, with a bubble in the signal line on the outside of the symbol. This notation indicates that application of a logic 0 to the signal line activates the clear operation on the flip-flops in the register. If the signal line were labeled outside the symbol, the label would be ***Clear***.

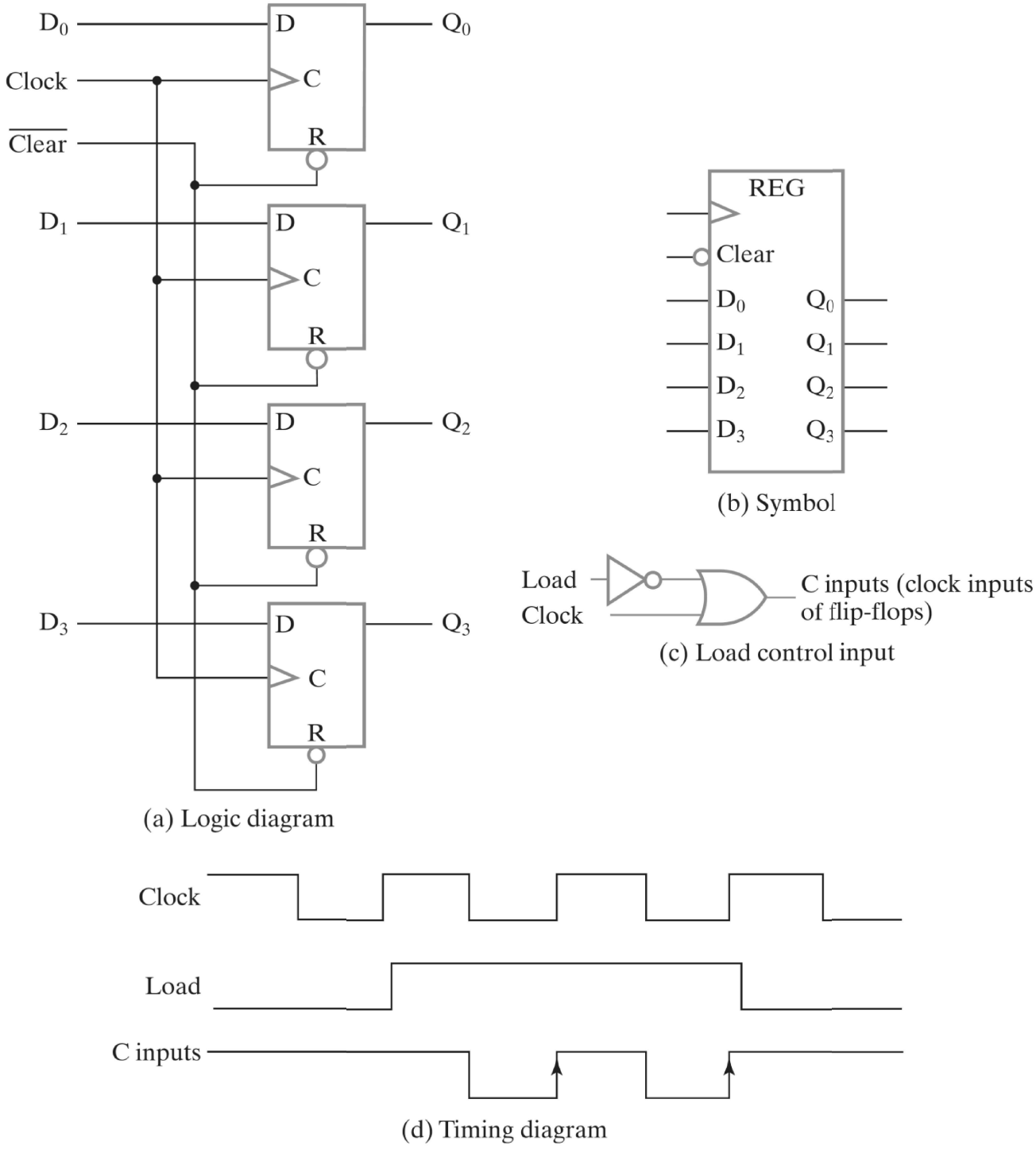


Figure 1: A 4-bit Register

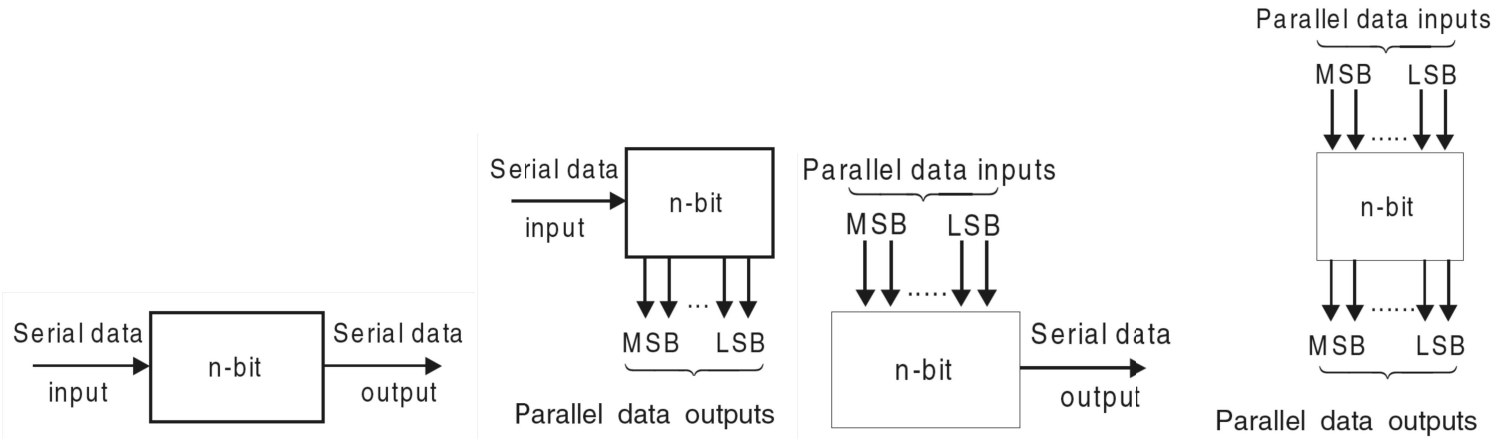
SHIFT REGISTER

A register capable of shifting its binary contents either to the left or to the right is called a *shift register*. The shift register permits the stored data to move from a particular location to some other location within the register. Registers can be designed using discrete flip-flops (S-R, J-K, and D-type).

The data in a shift register can be shifted in two possible ways: (*a*) serial shifting and (*b*) parallel shifting. The serial shifting method shifts one bit at a time for each clock pulse in a serial manner, beginning with either LSB or MSB. On the other hand, in parallel shifting operation, all the data (input or output) gets shifted simultaneously during a single clock pulse. Hence, we may say that parallel shifting operation is much faster than serial shifting operation.

There are two ways to shift data into a register (serial or parallel) and similarly two ways to shift the data out of the register. This leads to the construction of four basic types of registers as shown in Figures 2(*a*) to 2(*d*). All of the four configurations are commercially available as TTL MSI/LSI circuits. They are:

1. Serial in/Serial out (SISO)
2. Serial in/Parallel out (SIPO)
3. Parallel in/Serial out (PISO)
4. Parallel in/Parallel out (PIPO).



(a) Serial in/Serial out (b) Serial in/Parallel out (c) Parallel in/Serial out (d) Parallel in/Parallel out

Figure 2: Four types of shift registers

SERIAL-IN–-SERIAL-OUT SHIFT REGISTER

From the name itself it is obvious that this type of register accepts data serially, *i.e.*, one bit at a time at the single input line. The output is also obtained on a single output line in a serial fashion. The data within the register may be shifted from left to right using *shift-left* register, or may be shifted from right to left using *shift-right* register.

SHIFT-RIGHT REGISTER

A shift-right register can be constructed with either J-K or D flip-flops as shown in Figure 3. A J-K flip-flop–based shift register requires connection of both J and K inputs. Input data are connected to the J and K inputs of the left most (lowest order) flip-flop. To input a 0, one should apply a 0 at the J input, *i.e.*, J = 0 and K = 1 and vice versa. With the application of a clock pulse the data will be shifted by one bit to the right.

In the shift register using D flip-flop, D input of the left most flip-flop is used as a serial input line. To input 0, one should apply 0 at the D input and vice versa.

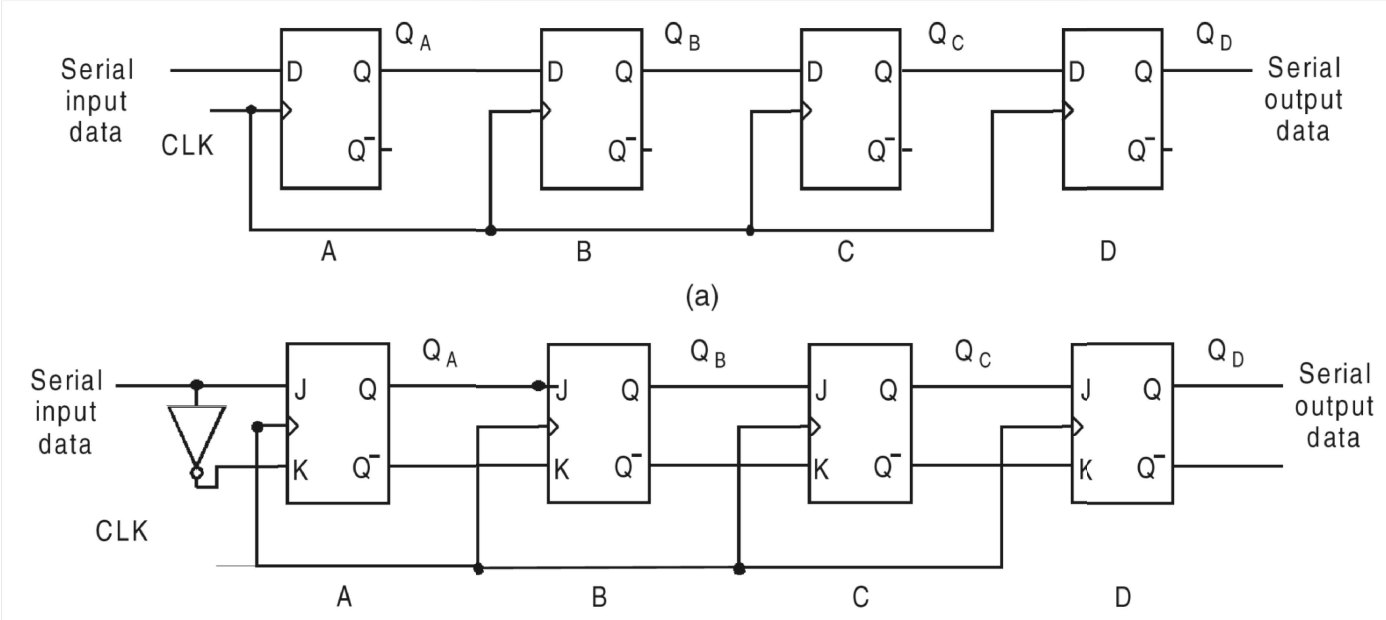


Figure 3: Shift-right register (a) using D flip-flops, (b) using J-K flip-flops

The clock pulse is applied to all the flip-flops simultaneously. When the clock pulse is applied, each flip-flop is either set or reset according to the data available at that point of time at the respective inputs of the individual flip-flops. Hence the input data bit at the serial input line is entered into flip-flop A by the first clock pulse. At the same time, the data of stage A is shifted into stage B and so on to the following stages. For each clock pulse, data stored in the register is shifted to the right by one stage. New data is entered into stage A, whereas the data present in stage D are shifted out (to the right).

Timing pulse	Q_A	Q_B	Q_C	Q_D	*Serial output at* Q_D
Initial value	0	0	0	0	0
After 1st clock pulse	1	0	0	0	0
After 2nd clock pulse	1	1	0	0	0
After 3rd clock pulse	0	1	1	0	0
After 4th clock pulse	1	0	1	1	1

Table 1: Operation of the Shift-right Register

For example, consider that all the stages are reset and a logical input 1011 is applied at the serial input line connected to stage A. The data after four clock pulses is shown in Table 1. Let us now illustrate the entry of the 4-bit number 1011 into the register, beginning with the right-most bit. A 1 is applied at the serial input line, making D = 1.

As the first clock pulse is applied, flip-flop A is SET, thus storing the 1. Next, a 1 is applied to the serial input, making D = 1 for flip-flop A and D = 1 for flip-flop B also, because the input of flip-flop B is connected to the Q_A output.

When the second clock pulse occurs, the 1 on the data input is "shifted" to the flip-flop A and the 1 in the flip-flop A is "shifted" to flip-flop B. The 0 in the binary number is now applied at the serial input line, and the third clock pulse is now applied. This 0 is entered in flip-flop A and the 1 stored in flip-flop A is now "shifted" to flip-flop B and the 1 stored in flip-flop B is now "shifted" to flip-flop C. The last bit in the binary number that is the 1 is now applied at the serial input line and the fourth clock pulse is now applied. This 1 now enters the flip-flop A and the 0 stored in flip-flop A is now "shifted" to flip-flop B and the 1 stored in flip-flop B is now "shifted" to flip-flop C and the 1 stored in flip-flop C is now "shifted" to flip-flop D. Thus the entry of the 4-bit binary number in the shift-right register is now complete.

From the third column of Table 1 we can get the serial output of the data that is being entered in the register. We find that after the first, second, and the third clock pulses the output at the serial output line i.e., Q_D is 0. After the fourth clock pulse the output at the serial output line is 1. If we want to get the total data that we have entered in the register in a serial manner from Q_D, then we have to apply another three clock pulses. After the fifth clock pulse we will gate another 1 at Q_D. After the sixth clock pulse the output at Q_D will be 0 and after the seventh clock pulse the output at Q_D will be 1. In this process of the fifth, sixth, and the seventh clock pulses if no data is being supplied at the serial input line then the A, B, and C flip-flops will again be RESET with output 0.

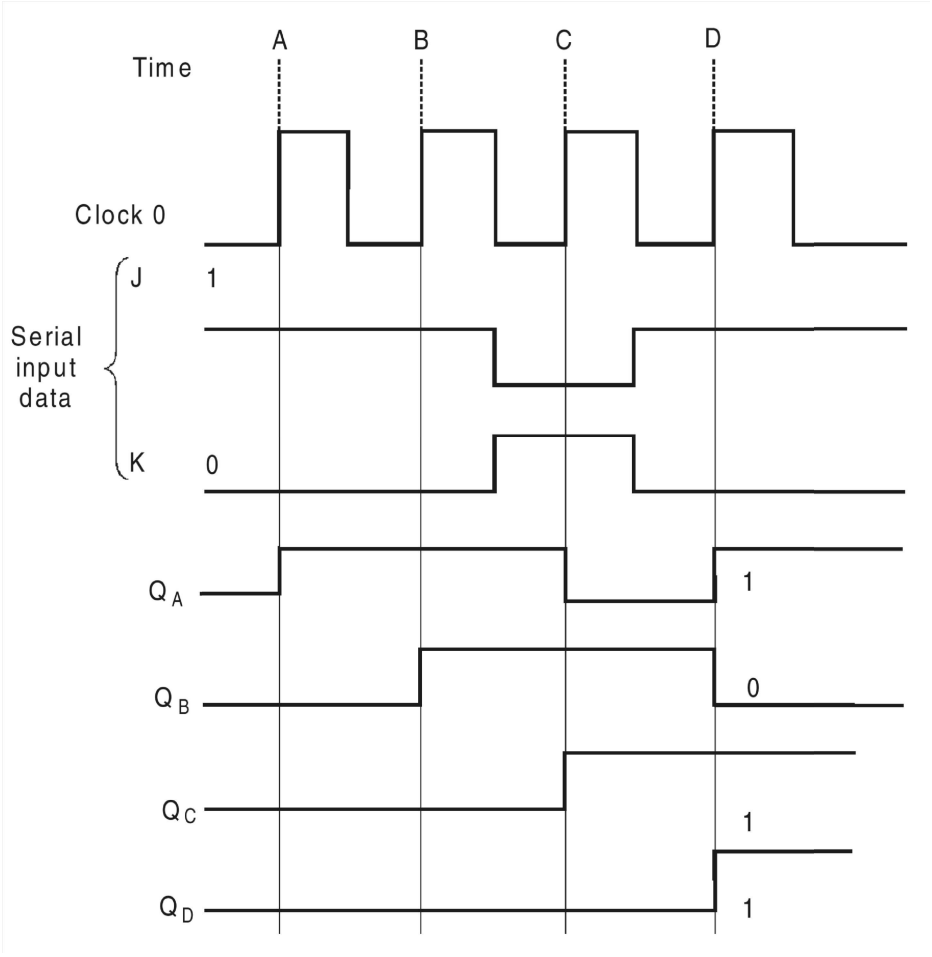


Figure 4: Waveforms of 4-bit serial input shift-right register

The waveforms shown in Figure 4 illustrate the entry of a 4-bit number 1011. For a J-K flip-flop, the data bit to be shifted into the flip-flop must be present at the J and K inputs when the clock transitions from low to high occur. Since the data bit is either 1 or 0, there can be two different cases:

1. To shift a 1 into the flip-flop, J = 1 and K = 0,

2. To shift a 0 into the flip-flop, J = 0 and K = 1.

At time A: All the flip-flops are reset. At the serial data input line a 1 is given and with the first clock pulse this 1 is shifted at Q_A making $Q_A = 1$. At the same time the 0 in Q_A is shifted to Q_B, and the 0 in Q_B is shifted to Q_C and the 0 in Q_C is shifted to Q_D. Hence the flip-flop outputs just after time A are $Q_AQ_BQ_CQ_D = 1000$.

At time B: The flip-flop A contains 1, and all other flip-flop contains 0. Now, again, 1 is given at the serial data input line. With the second clock pulse this 1 is shifted to Q_A. The 1 in Q_A is shifted to Q_B and the 0 in Q_B is shifted to Q_C and the 0 in Q_C is shifted to Q_D. Hence the flip-flop outputs just after time B are $Q_AQ_BQ_CQ_D = 1100$.

At time C: The flip-flop A and flip-flop B contain 1, and all other flip-flops contain 0. Now a 0 is given at the serial data input line. With the third clock pulse this 0 is shifted to Q_A. The 1 in Q_A is shifted to Q_B and the 1 in Q_B is shifted to Q_C and the 0 in Q_C is shifted to Q_D. Hence the flip-flop outputs just after time C are $Q_AQ_BQ_CQ_D = 0110$.

At time D: The flip-flop B and flip-flop C contain 1, and all other flip-flops contain 0. Now another 1 is given at the serial data input line. With the fourth clock pulse this 1 is shifted to Q_A. The 0 in Q_A is shifted to Q_B and the 1 in Q_B is shifted to Q_C and the 1 in Q_C is shifted to Q_D. Hence the flip-flop outputs just after time C are $Q_AQ_BQ_CQ_D = 1011$.

To summarize, we have shifted 4 data bits in a serial manner into four flip-flops. These 4 data bits could represent a 4-bit binary number 1011, assuming that we began shifting with the LSB first. Notice that the LSB is in D and the MSB is in A. These four flip-flops could be defined as a 4-bit shift register.

SHIFT-LEFT REGISTER

A shift-left register can also be constructed with either J-K or D flip-flops as shown in Figure 5. Let us now illustrate the entry of the 4-bit number 1110 into the register, beginning with the right-most bit. A 0 is applied at the serial input line, making D = 0. As the first clock pulse is applied, flip-flop A is RESET, thus storing the 0. Next a 1 is applied to the serial input, making D = 1 for flip-flop A and D = 0 for flip-flop B, because the input of flip-flop B is connected to the Q_A output.

When the second clock pulse occurs, the 1 on the data input is “shifted” to the flip-flop A and the 0 in the flip-flop A is “shifted” to flip-flop B. The 1 in the binary number is now applied at the serial input line, and the third clock pulse is now applied. This 1 is entered in flip-flop A and the 1 stored in flip-flop A is now “shifted” to flip-flop B and the 0 stored in flip-flop B is now “shifted” to flip-flop C. The last bit in the binary number that is the 1 is now applied at the serial input line and the fourth clock pulse is now applied. This 1 now enters the flip-flop A and the 1 stored in flip-flop A is now “shifted” to flip-flop B and the 1 stored in flip-flop B is now “shifted” to flip-flop C and the 0 stored in flip-flop C is now “shifted” to flip-flop D. Thus the entry of the 4-bit binary number in the shift-right register is now complete.

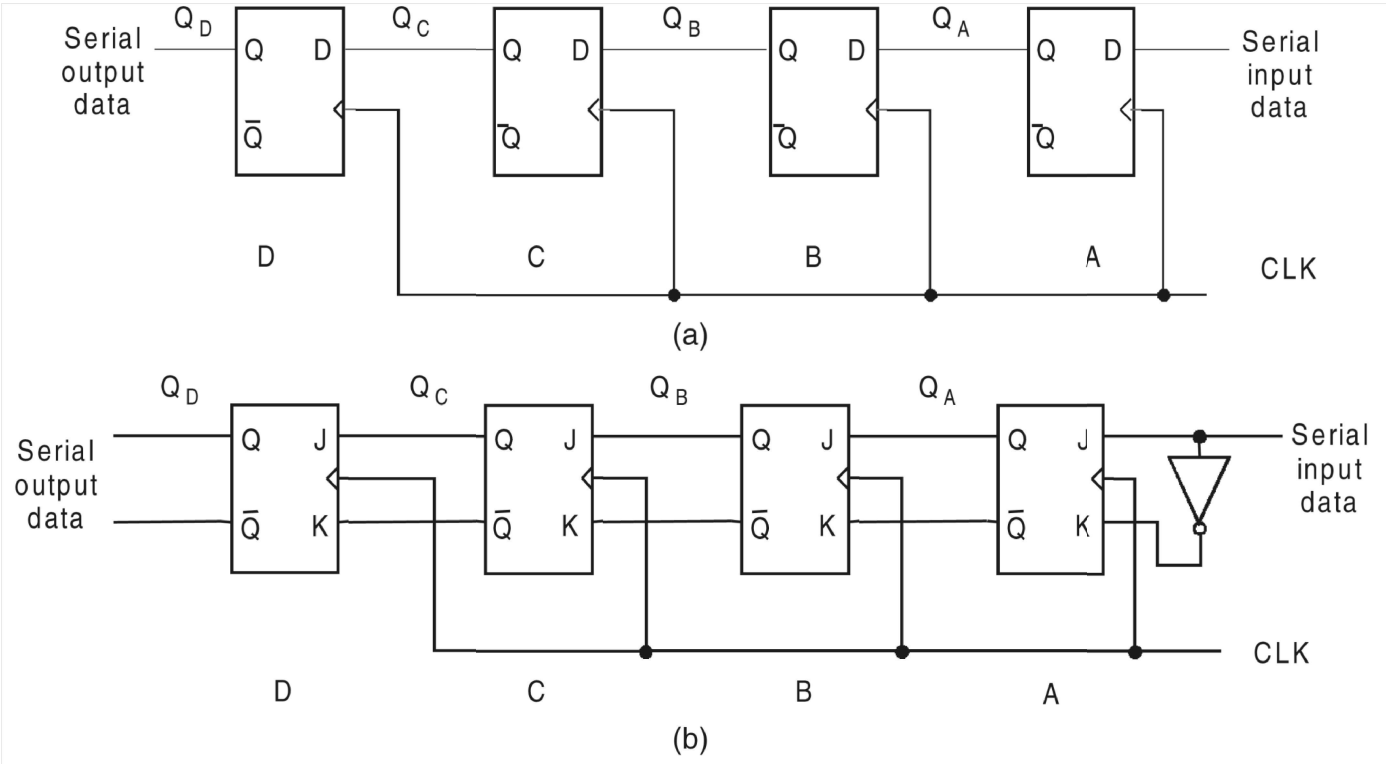


Figure 5: Shift-left register (*a*) using D flip-flops, (*b*) using J-K flip-flops

Timing pulse	Q_D	Q_C	Q_B	Q_A	*Serial output at* Q_D
Initial value	0	0	0	0	0
After 1st clock pulse	0	0	0	0	0
After 2nd clock pulse	0	0	0	1	0
After 3rd clock pulse	0	0	1	1	0
After 4th clock pulse	0	1	1	1	0

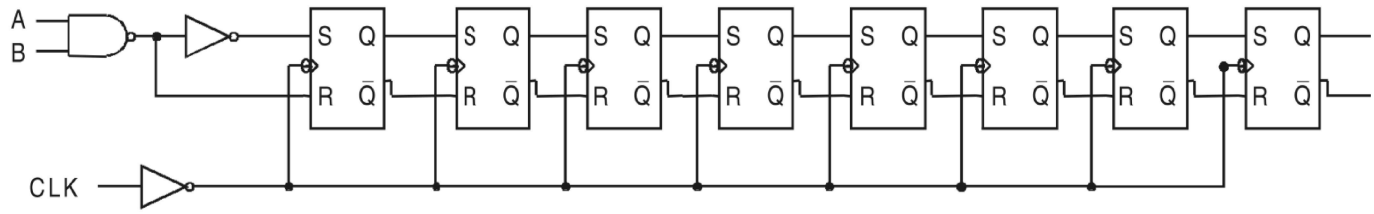
Table 2: Operation of the Shift-left Register

8-BIT SERIAL-IN–SERIAL-OUT SHIFT REGISTER

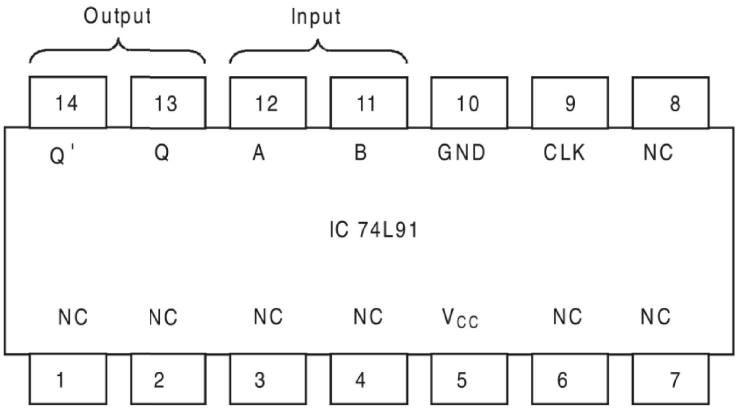
The pinout and logic diagram of an 8-bit serial-in–serial-out shift register, IC 74L91, is shown in Figure 6. This is an 8-bit TTL MSI chip. There are eight S-R flip-flops connected to provide a serial input as well as a serial output. The clock input at each flip-flop is negative edge-triggered. However, the applied clock signal is passed through an inverter. Hence the data will be shifted on the positive edges of the input clock pulses.

An inverter is connected in between R and S on the fist flip-flop. This means that this circuit functions as a D-type flip-flop. So the input to the register is a single liner on which the data can be shifted into the register appears serially. The data input is applied at either A (pin 12) or B (pin 11). The data level at A (or B) is complemented by the NAND gate and then applied to the R input of the first flip-flop. The same data level is complemented by the NAND gate and then again complemented by the inverter before it appears at the S input. So, a 0 at input A will *reset* the first flip-flop (in other words this 0 is shifted into the first flip-flop) on a positive clock transition.

The NAND gate with A and B inputs provide a gating function for the input data stream if required, if gating is not required, connect pins 11 and 12 together and apply the input data stream to this connection.



(a) Logic Diagram



(b) Pinout diagram of IC 74L91

Figure 6: 8-bit shift register—IC 74L91

SERIAL-IN–PARALLEL-OUT REGISTER

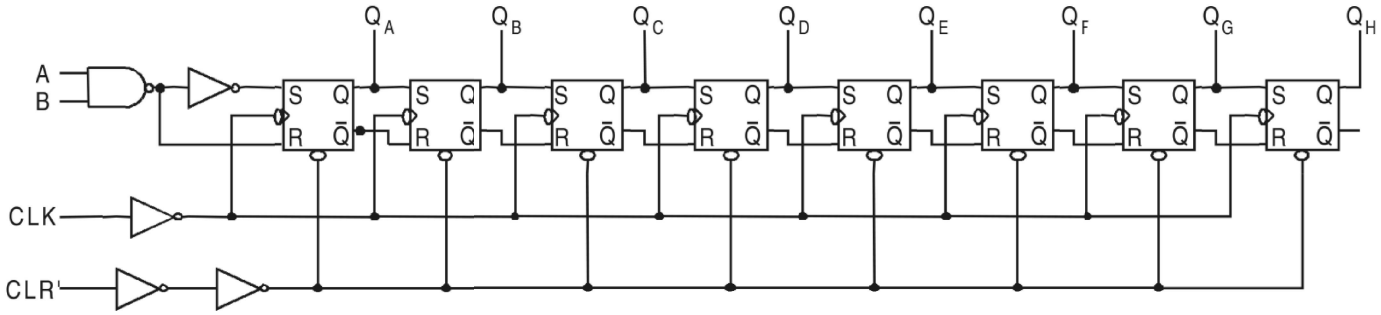
In this type of register, the data is shifted in serially, but shifted out in parallel. To obtain the output data in parallel, it is required that all the output bits are available at the same time. This can be accomplished by connecting the output of each flip-flop to an output pin. Once the data is stored in the flip-flop the bits are available simultaneously. The basic configuration of a serial-in–parallel-out shift register is shown in Figure 2(b).

8-BIT SERIAL-IN–PARALLEL-OUT SHIFT REGISTER

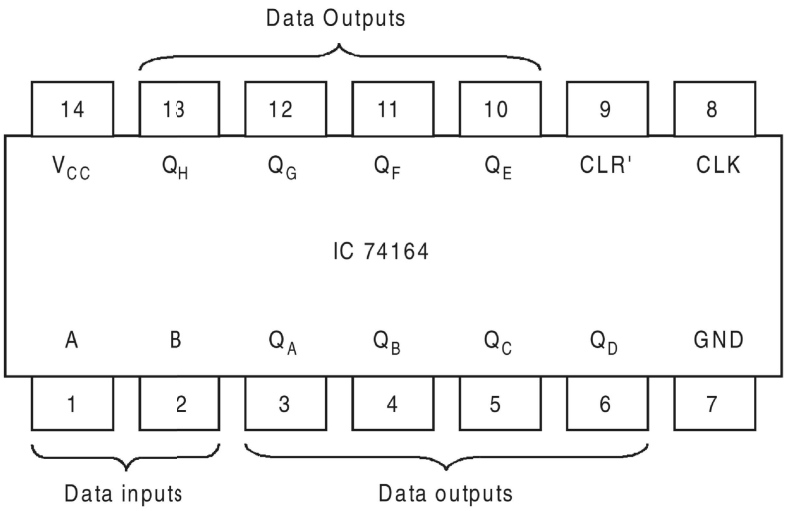
The pinout and logic diagram of an 8-bit serial-in–parallel-out shift register, IC 74164, is shown in Figure 7. There are eight S-R flip-flops, which are all sensitive to negative clock transitions. The logic diagram in Figure 7 is almost the same as shown in Figure 6 with only two exceptions: (1) each flip-flop has an asynchronous CLEAR input; and (2) the true side of each flip-flop is available as an output—thus all 8 bits of any number stored in the register are available simultaneously as an output (this is a parallel data output).

Hence, a low level at the CLR input to the chip (pin 9) is applied through an amplifier and will reset every flip-flop. As long as the CLR input to the chip is LOW, the flip-flop outputs will all remain low. It means that, in effect, the register will contain all zeros.

Shifting of data into the register in a serial fashion is exactly the same as the IC 74L91. Data at the serial input may be changed while the clock is either low or high, but the usual hold and setup times must be observed. The data sheet for this device gives hold time as 0.0 ns and setup time as 30 ns.



(a) Logic diagram



(b) Pinout diagram of IC 74164

Figure 7: 8-bit shift register—IC 74164

Now let us analyze the gated serial inputs A and B. Suppose that the serial data is connected to B; then A can be used as a control line. Here's how it works:

A is held high: The NAND gate is enabled and the serial input data passes through the NAND gate inverted. The input data is shifted serially into the register.

A is held low: The NAND gate output is forced high, the input data steam is inhibited, and the next clock pulse will shift a 0 into the first flip-flop. Each succeeding positive clock pulse will shift another 0 into the register. After eight clock pulses, the register will be full of zeros.

Example 1: *How long will it take to shift an 8-bit number into a shift register if the clock is set at 1 MHz?*

Solution: A minimum of eight clock Pulses will be required since the data is entered serially. One clock pulse period is 1000 ns, so it will require 8000 ns minimum.

PARALLEL-IN–SERIAL-OUT REGISTER

In the preceding two cases the data was shifted into the registers in a serial manner. We now can develop an idea for the parallel entry of data into the register. Here the data bits are entered into the flip-flops simultaneously, rather than a bit-by-bit basis.

A 4-bit parallel-in–serial-out register is illustrated in Figure 8. A, B, C, and D are the four parallel data input lines and *SHIFT* / ***LOAD*** (*SH* / ***LD***) is a control input that allows the four bits of data at A, B, C,

and D inputs to enter into the register in parallel or shift the data in serial. When *SHIFT* / ***LOAD*** is HIGH, AND gates G_1, G_3, and G_5 are enabled, allowing the data bits to shift right from one stage to the next.

When *SHIFT* / ***LOAD*** is LOW, AND gates G_2, G_4, and G_6 are enabled, allowing the data bits at the parallel inputs. When a clock pulse is applied, the flip-flops with D = 1 will be set and the flip-flops with D = 0 will be reset, thereby storing all the four bits simultaneously. The OR gates allow either the normal shifting operation or the parallel data-entry operation, depending on which of the AND gates are enabled by the level on the *SHIFT* / ***LOAD*** input.

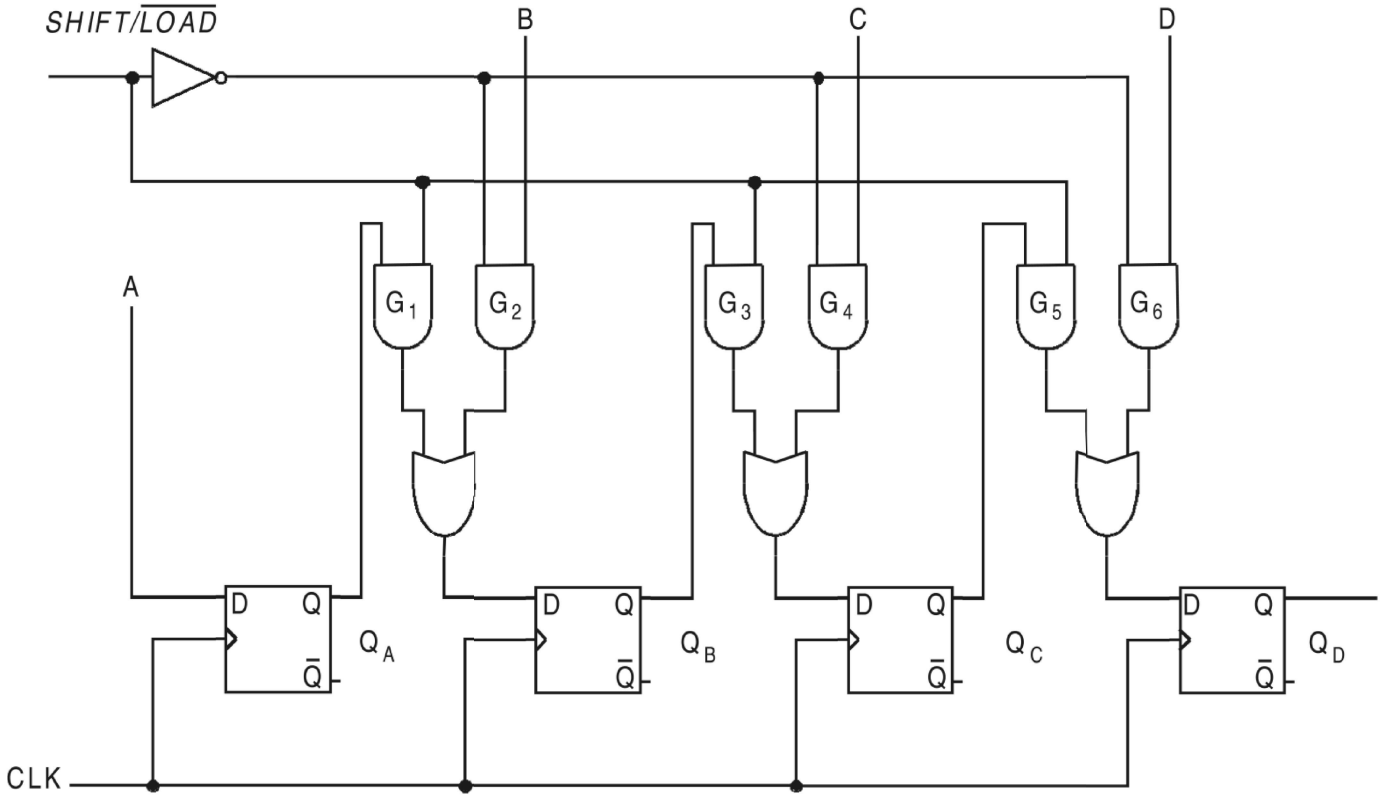
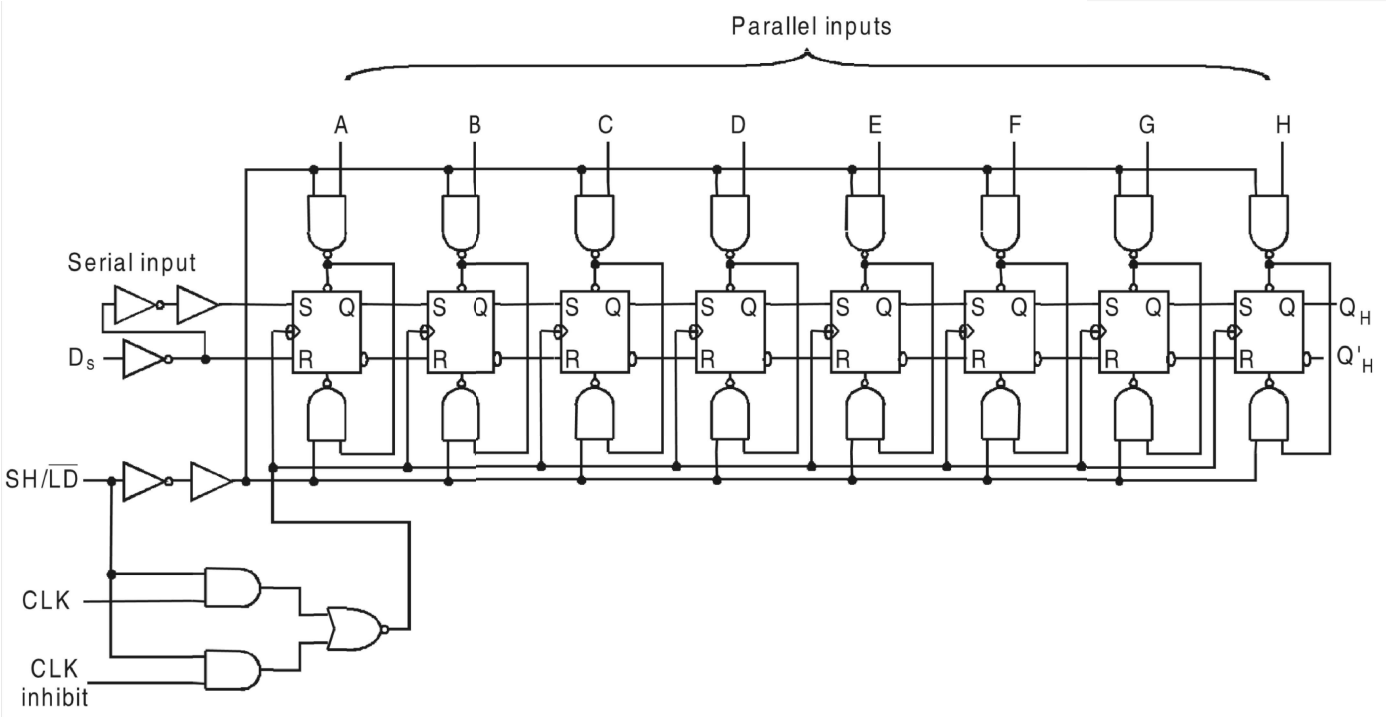


Figure 8: A 4-bit parallel-in–serial-out shift register

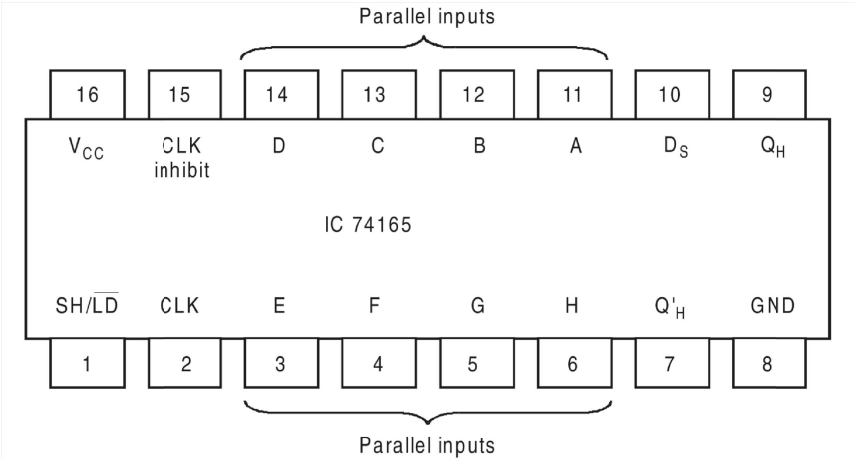
8-BIT PARALLEL-IN–SERIAL-OUT SHIFT REGISTER

The pinout and logic diagram of an 8-bit serial/parallel-in and serial-out shift register (IC 74165) is shown in Figure 9. The data can be loaded into the register in parallel and shifted out serially at QH using either of two clocks (CLK or CLK inhibit). It also contains a serial input, DS through which the data can be serially shifted in.

When the input *SHIFT* / ***LOAD*** (*SH* / ***LD***) is LOW, it enables all the NAND gates for parallel loading. When an input data bit is a 0, the flip-flop is asynchronously RESET by a LOW output of the lower NAND gate. Similarly, when the input data bit is a 1, the flip-flop is asynchronously SET by a LOW output of the upper NAND gate. The clock is inhibited during parallel loading operation. A HIGH on the *SHIFT* / ***LOAD*** input enables the clock causing the data in the register to shift right. With the low to high transitions of either clock, the serial input data (D_S) are shifted into the 8-bit register.



(a) Logic diagram



(b) Pinout diagram of IC 74165

Figure 9: 8-bit serial/parallel-in and serial-out shift register—IC 74165

PARALLEL-IN–PARALLEL-OUT REGISTER

There is a fourth type of register which is designed such that data can be shifted into or out of the register in parallel. Also, in this type of register there is no interconnection between the flip-flops since no serial shifting is required. Hence, the moment the parallel entry of the data is accomplished the data will be available at the parallel outputs of the register.

Here the parallel inputs to be applied at A, B, C, and D inputs are directly connected to the D inputs of the respective flip-flops. On applying the clock transitions, these inputs are entered into the register and are immediately available at the outputs Q_1, Q_2, Q_3, and Q_4.

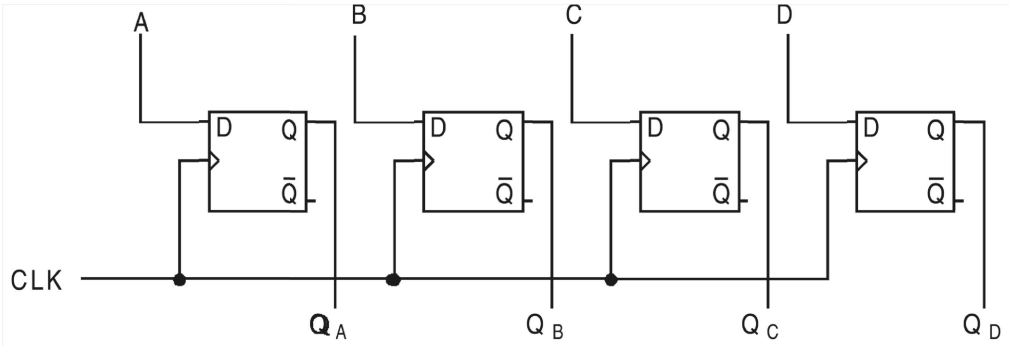


Figure 10: A 4-bit parallel-in–parallel-out shift register

8-BIT SERIAL/PARALLEL-IN AND SERIAL/PARALLEL-OUT SHIFT REGISTER

The pinout diagram of an 8-bit parallel-in and parallel-out shift register, IC 74198, is shown in Figure 11. IC 74198 is a 24-pin package where 16 pins are needed just for the input and output data lines. This chip can even shift data through the register in either direction, *i.e.*, shift-right and shift-left.

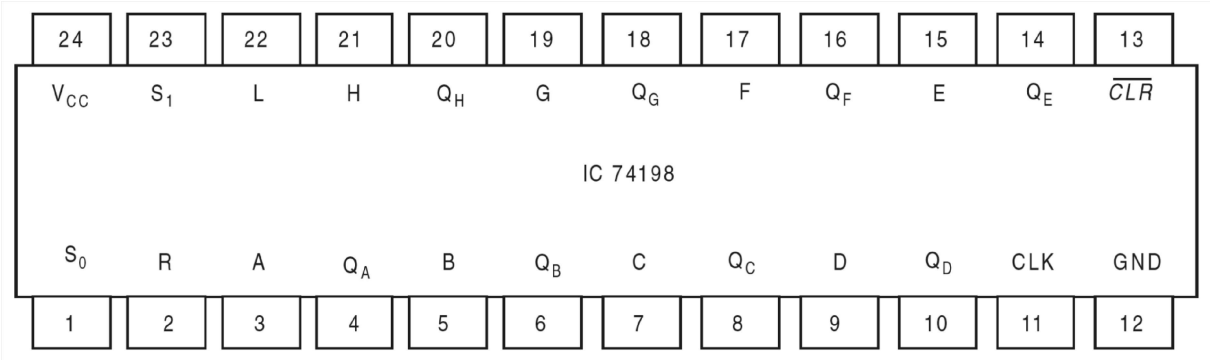


Figure 11: Pinout diagram of IC 74198, 8-bit parallel-in–parallel-out shift register.

L, i.e., pin 22 in Figure 11, represents the shift-left serial input and R (pin 2) represents the shift-right serial input. An 8-bit register can be created by either connecting two 4-bit registers in series or by manufacturing the two 4-bit registers on a single chip and placing the chip in a 24-pin package such as IC 74198.

There are a number of 4-bit parallel-input–parallel-output shift registers available since they can be conveniently packaged in a 16-pin dual-inline package. IC 74195 is a 4-bit TTL MSI having both serial/parallel input and serial/parallel output capability. The pinout diagram of IC 74195 is shown in Figure 12. Since this IC has a serial input, it can also be used for serial-in–serial-out and serial-in–parallel-out operation. This IC can be used for parallel-in–serial-out operation by using Q_D as the output.

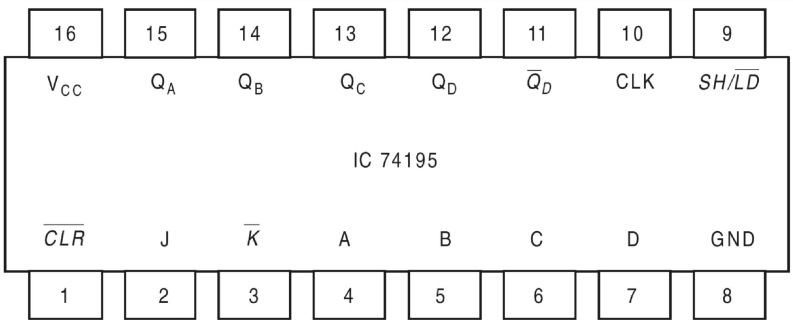


Figure 12: Pinout diagram of IC 74195

UNIVERSAL REGISTER

A register that is capable of transferring data in only one direction is called a '***unidirectional shift register***,' whereas the register that is capable of transferring data in both left and right direction is called a '***bidirectional shift register***.' Now if the register has both the shift-right and shift-left capabilities, along with the necessary input and output terminals for parallel transfer, then it is called a shift register with parallel load or '***universal shift register***.'

The most general shift register has all the capabilities listed below. Others may have only some of these functions, with at least one shift operation.

1. A shift-right control to enable the shift-right operation and the serial input and output lines associated with the shift-right.
2. A shift-left control to enable the shift-left operation and the serial input and output lines associated with the shift-left.
3. A parallel-load control to enable a parallel transfer and the n input lines associated with the parallel transfer.
4. *n* parallel output lines.
5. A clear control to clear the register to 0.
6. A CLK input for clock pulses to synchronize all operations.
7. A control state that leaves the information in the register unchanged even though clock pulses are continuously applied.

The diagram of a shift-register with all the capabilities listed above is shown in Figure 13. This is similar to IC type 74194. Though it consists of four D flip-flops, S-R flip-flops can also be used with an inverter inserted between the S and R terminals. The four multiplexers drawn are also part of the register. The four multiplexers have two common selection lines S_1 and S_0. When $S_1S_0 = 00$, the input 0 is selected for each of the multiplexers. Similarly, when $S_1S_0 = 01$, the input 1, when $S_1S_0 = 10$, the input 2 and for S_1 $S_0 = 11$, the input 3, is selected for each of the multiplexers.

The S_1 and S_0 inputs control the mode of operation of the register as specified in the entries of functions in Table 3. When $S_1S_0 = 00$, the present value of the register is applied to the D inputs of the flip-flops. Hence this condition forms a path from the output of each flip-flop into the input of the same flip-flop. The next clock pulse transition transfers into each flip-flop the binary value held previously, and no change of state occurs. When $S_1S_0 = 01$, terminals 1 of each of the multiplexer inputs have a path to the D inputs of each of the flip-flops. This causes a shift-right operation, with the serial input transferred into flip-flop A_4. Similarly, with $S_1S_0 = 10$, a shift-left operation results, with the other serial input going into flip-flop A_1. Finally, when $S_1S_0 = 11$, the binary information on the parallel input lines is transferred into the register simultaneously during the next clock pulse.

A universal register is a general-purpose register capable of performing three operations: shift-right, shift-left, and parallel load. Not all shift registers available in MSI circuits have all these capabilities. The particular application dictates the choice of one MSI circuit over another. The pinout diagram of a 4-bit bidirectional shift register with parallel load IC 74194 is shown in Figure 14.

The parallel loading of data is accomplished with a positive transition of the clock and by applying the four bits of data to the parallel inputs and a HIGH to the S_1 and S_0 inputs. Similarly, shift-right is accomplished synchronously with the positive edge of the clock when S_0 is HIGH and S_1 is LOW. In this mode the serial data is entered at the shift-right serial input. In the same manner, when S_0 is LOW and S_1 is HIGH, data bits shift left synchronously with the clock pulse and new data is entered at the shift-left serial input.

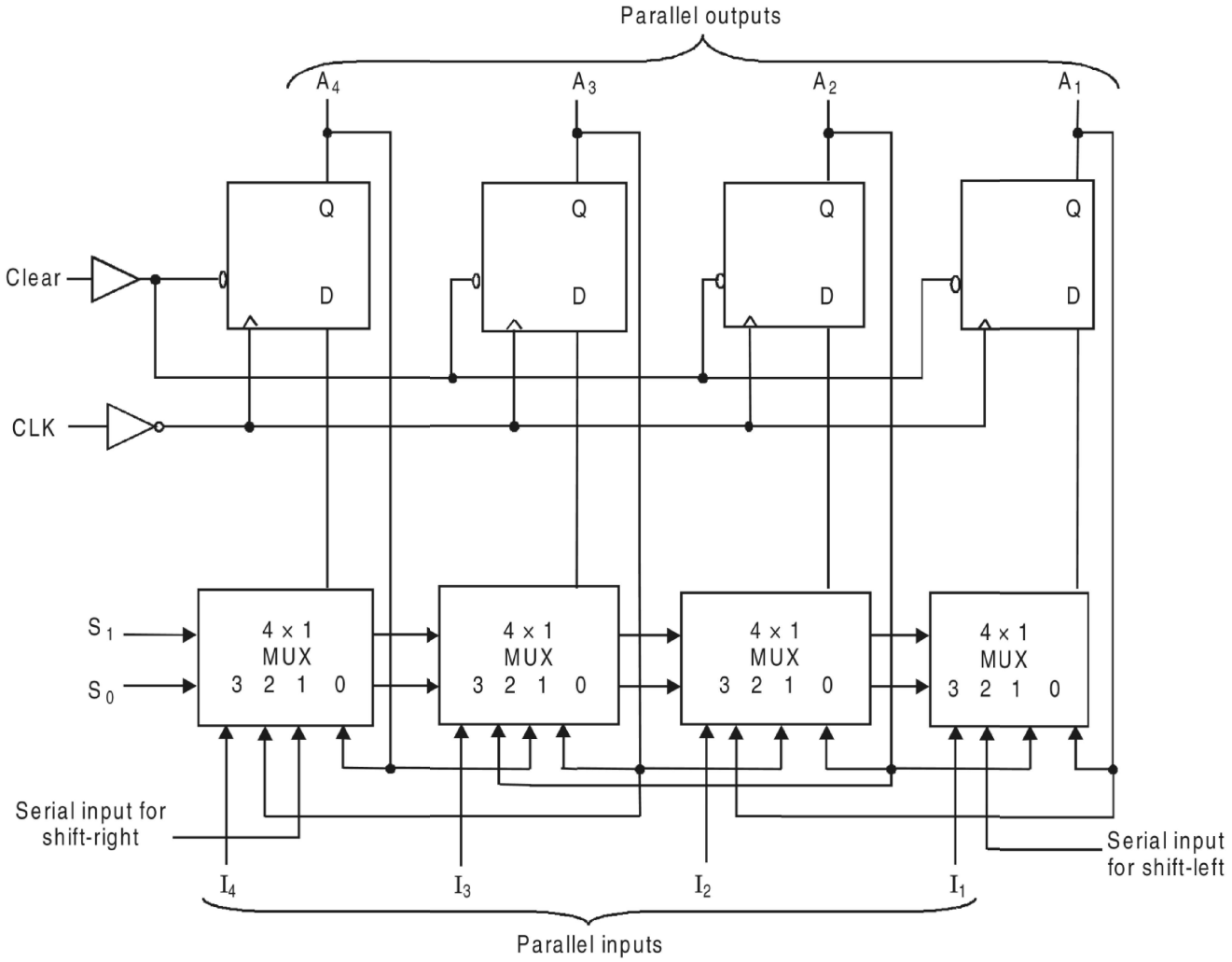


Figure 13: 4-bit universal shift register

Mode control		*Register operation*
S_1	S_0	
0	0	No change
0	1	Shift-right
1	0	Shift-left
1	1	Parallel load

Table 3: Function table for the universal register

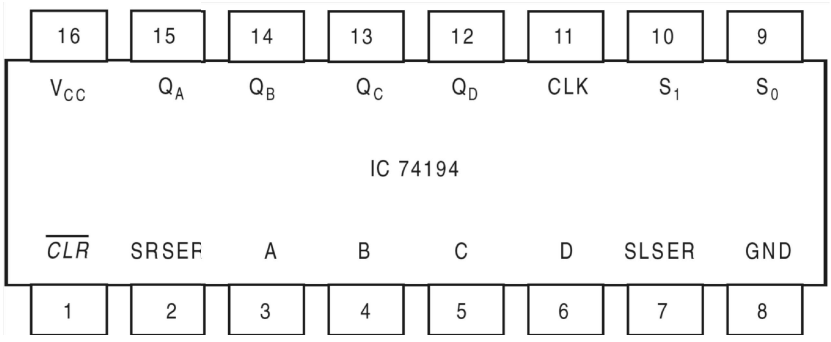


Figure 14: Pinout diagram of IC 74194