8–1 Shift Register Operations

Shift registers consist of arrangements of flip-flops and are important in applications involving the storage and transfer of data in a digital system. A register has no specified sequence of states, except in certain very specialized applications. A register, in general, is used solely for storing and shifting data (1s and 0s) entered into it from an external source and typically possesses no characteristic internal sequence of states.

After completing this section, you should be able to

- Explain how a flip-flop stores a data bit
- Define the storage capacity of a shift register
- Describe the shift capability of a register

A register can consist of one or more flip-flops used to store and shift data.

A **register** is a digital circuit with two basic functions: data storage and data movement. The storage capability of a register makes it an important type of memory device. Figure 8–1 illustrates the concept of storing a 1 or a 0 in a D flip-flop. A 1 is applied to the data input as shown, and a clock pulse is applied that stores the 1 by *setting* the flip-flop. When the 1 on the input is removed, the flip-flop remains in the SET state, thereby storing the 1. A similar procedure applies to the storage of a 0 by *resetting* the flip-flop, as also illustrated in Figure 8–1.

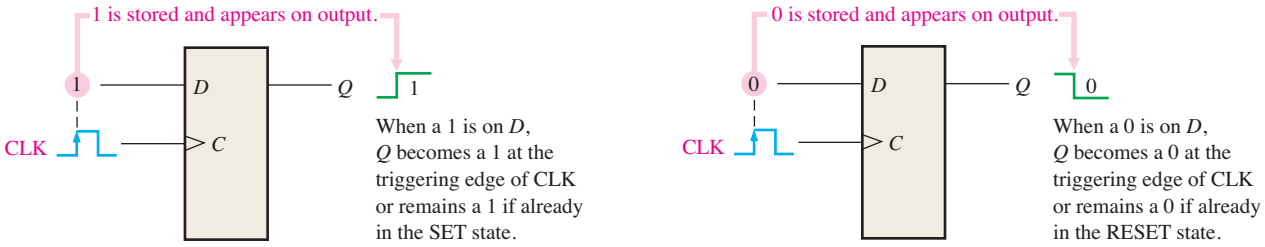


FIGURE 8–1 The flip-flop as a storage element.

The *storage capacity* of a register is the total number of bits (1s and 0s) of digital data it can retain. Each **stage** (flip-flop) in a shift register represents one bit of storage capacity; therefore, the number of stages in a register determines its storage capacity.

The *shift capability* of a register permits the movement of data from stage to stage within the register or into or out of the register upon application of clock pulses. Figure 8–2

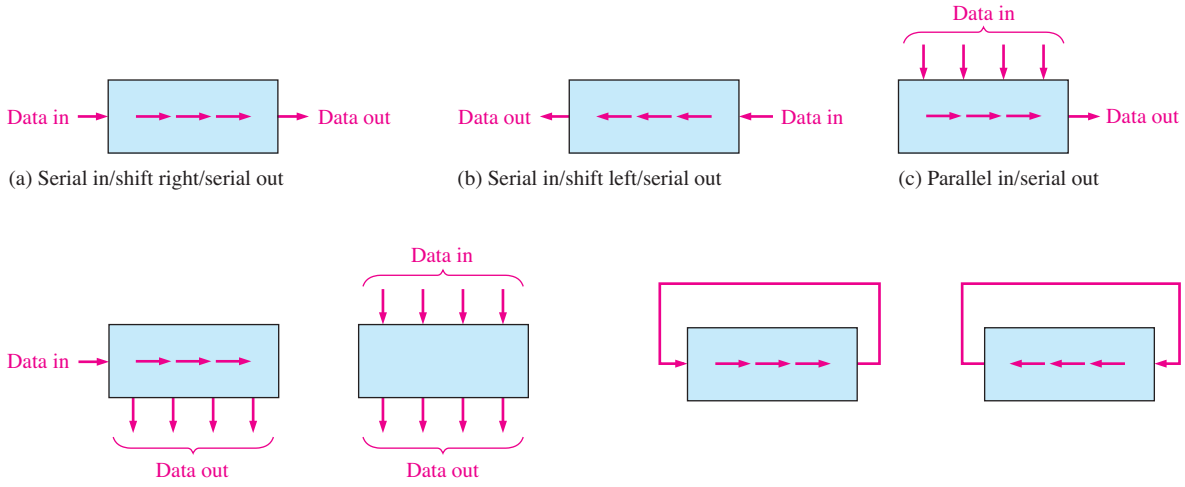


FIGURE 8–2 Basic data movement in shift registers. (Four bits are used for illustration. The bits move in the direction of the arrows.)

illustrates the types of data movement in shift registers. The block represents any arbitrary 4-bit register, and the arrows indicate the direction of data movement.

SECTION 8–1 CHECKUP

Answers are at the end of the chapter.

1. What determines the storage capacity of a shift register?
2. What two principal functions are performed by a shift register?

8–2 Types of Shift Register Data I/Os

In this section, four types of shift registers based on data input and output (inputs/outputs) are discussed: serial in/serial out, serial in/parallel out, parallel in/serial out, and parallel in/parallel out.

After completing this section, you should be able to

- Describe the operation of four types of shift registers
- Explain how data bits are entered into a shift register
- Describe how data bits are shifted through a register
- Explain how data bits are taken out of a shift register
- Develop and analyze timing diagrams for shift registers

Serial In/Serial Out Shift Registers

The serial in/serial out shift register accepts data serially—that is, one bit at a time on a single line. It produces the stored information on its output also in serial form. Let's first look at the serial entry of data into a typical shift register. Figure 8–3 shows a 4-bit device implemented with D flip-flops. With four stages, this register can store up to four bits of data.

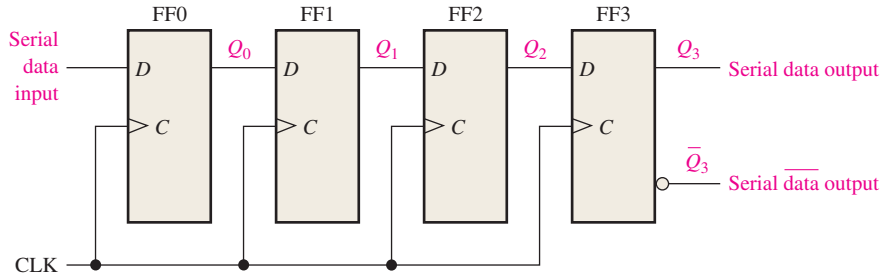


FIGURE 8–3 Serial in/serial out shift register.

InfoNote

Frequently, it is necessary to *clear* an internal register in a processor. For example, a register may be cleared prior to an arithmetic or other operation. One way that registers in a processor are cleared is using software to subtract the contents of the register from itself. The result, of course, will always be zero. For example, a processor instruction that performs this operation is SUB AL,AL. with this instruction, the register named AL is cleared.

Table 8–1 shows the entry of the four bits 1010 into the register in Figure 8–3, beginning with the least significant bit. The register is initially clear. The 0 is put onto the data input line, making $D = 0$ for FF0. When the first clock pulse is applied, FF0 is reset, thus storing the 0.

TABLE 8–1

Shifting a 4-bit code into the shift register in Figure 8–3. Data bits are indicated by a beige screen.

CLK	FF0 (Q_0)	FF1 (Q_1)	FF2 (Q_2)	FF3 (Q_3)
Initial	0	0	0	0
1	**0**	0	0	0
2	**1**	**0**	0	0
3	**0**	**1**	**0**	0
4	**1**	**0**	**1**	**0**

Next the second bit, which is a 1, is applied to the data input, making $D = 1$ for FF0 and $D = 0$ for FF1 because the D input of FF1 is connected to the Q_0 output. When the second clock pulse occurs, the 1 on the data input is shifted into FF0, causing FF0 to set; and the 0 that was in FF0 is shifted into FF1.

The third bit, a 0, is now put onto the data-input line, and a clock pulse is applied. The 0 is entered into FF0, the 1 stored in FF0 is shifted into FF1, and the 0 stored in FF1 is shifted into FF2.

For serial data, one bit at a time is transferred.

The last bit, a 1, is now applied to the data input, and a clock pulse is applied. This time the 1 is entered into FF0, the 0 stored in FF0 is shifted into FF1, the 1 stored in FF1 is shifted into FF2, and the 0 stored in FF2 is shifted into FF3. This completes the serial entry of the four bits into the shift register, where they can be stored for any length of time as long as the flip-flops have dc power.

If you want to get the data out of the register, the bits must be shifted out serially to the Q_3 output, as Table 8–2 illustrates. After CLK4 in the data-entry operation just described, the LSB, 0, appears on the Q_3 output. When clock pulse CLK5 is applied, the second bit appears on the Q_3 output. Clock pulse CLK6 shifts the third bit to the output, and CLK7 shifts the fourth bit to the output. While the original four bits are being shifted out, more bits can be shifted in. All zeros are shown being shifted in, after CLK8.

TABLE 8–2

Shifting a 4-bit code out of the shift register in Figure 8–3. Data bits are indicated by a beige screen.

CLK	FF0 (Q_0)	FF1 (Q_1)	FF2 (Q_2)	FF3 (Q_3)
Initial	**1**	**0**	**1**	**0**
5	0	**1**	**0**	**1**
6	0	0	**1**	**0**
7	0	0	0	**1**
8	0	0	0	0

EXAMPLE 8–1

Show the states of the 5-bit register in Figure 8–4(a) for the specified data input and clock waveforms. Assume that the register is initially cleared (all 0s).

Solution

The first data bit (1) is entered into the register on the first clock pulse and then shifted from left to right as the remaining bits are entered and shifted. The register contains $Q_4Q_3Q_2Q_1Q_0 = 11010$ after five clock pulses. See Figure 8–4(b).

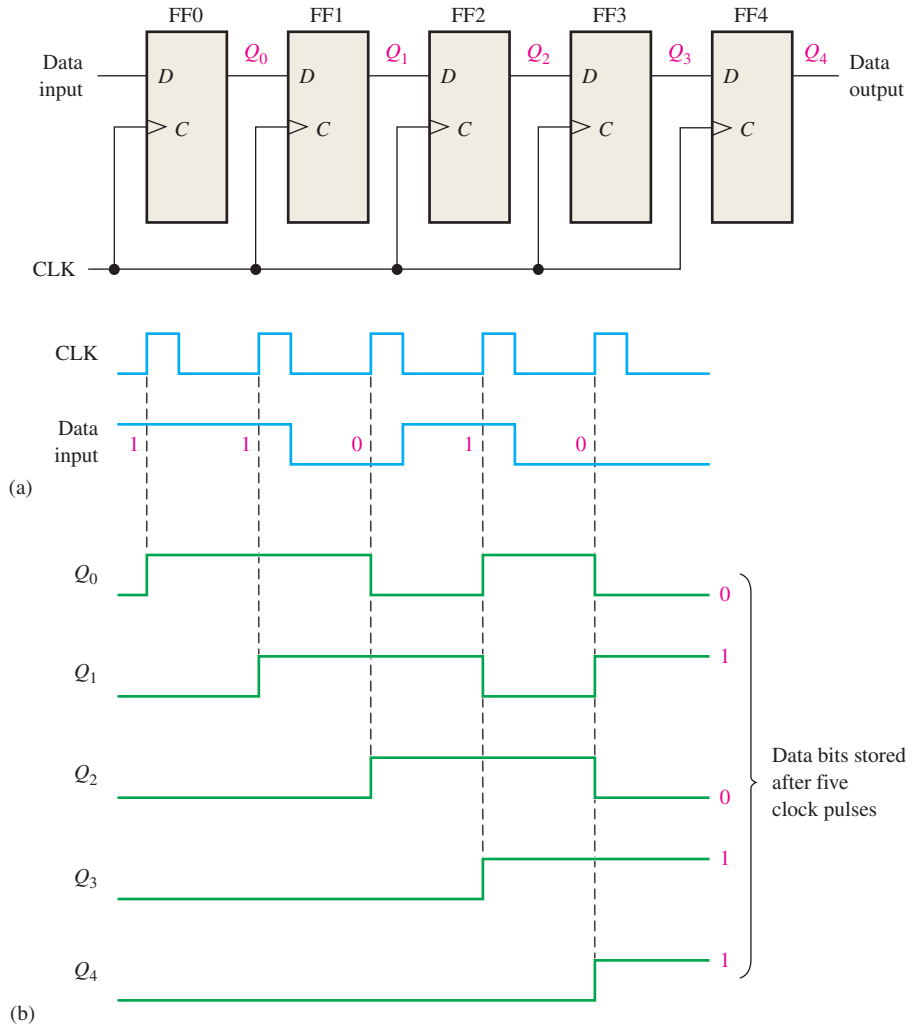


FIGURE 8–4 Open file F08-04 to verify operation. *A Multisim tutorial is available on the website.*

MultiSim

Related Problem*

Show the states of the register if the data input is inverted. The register is initially cleared.

*Answers are at the end of the chapter.

A traditional logic block symbol for an 8-bit serial in/serial out shift register is shown in Figure 8–5. The "SRG 8" designation indicates a shift register (SRG) with an 8-bit capacity.

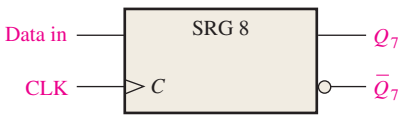


FIGURE 8–5 Logic symbol for an 8-bit serial in/serial out shift register.

Serial In/Parallel Out Shift Registers

Data bits are entered serially (least-significant bit first) into a serial in/parallel out shift register in the same manner as in serial in/serial out registers. The difference is the way in which the data bits are taken out of the register; in the parallel output register, the output of each stage is available. Once the data are stored, each bit appears on its respective output line, and all bits are available simultaneously, rather than on a bit-by-bit basis as with the serial output. Figure 8–6 shows a 4-bit serial in/parallel out shift register and its logic block symbol.

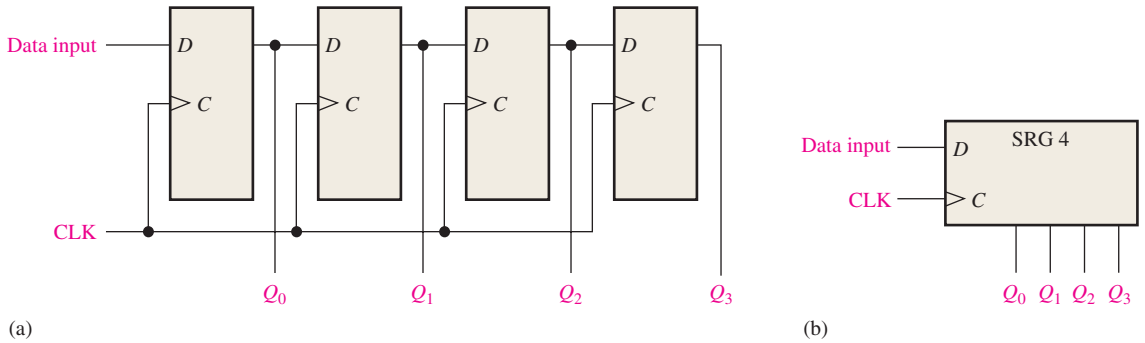


FIGURE 8–6 A serial in/parallel out shift register.

EXAMPLE 8–2

Show the states of the 4-bit register (SRG 4) for the data input and clock waveforms in Figure 8–7(a). The register initially contains all 1s.

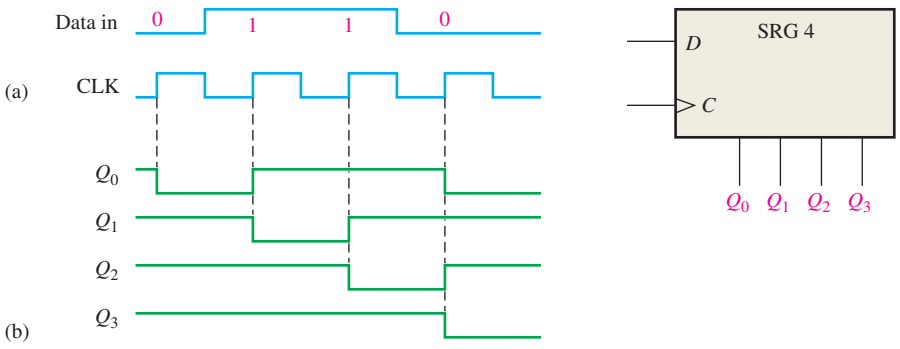


FIGURE 8–7

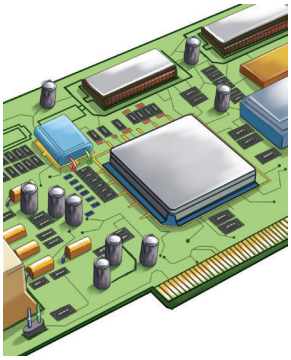
Solution

The register contains 0110 after four clock pulses. See Figure 8–7(b).

Related Problem

If the data input remains 0 after the fourth clock pulse, what is the state of the register after three additional clock pulses?

IMPLEMENTATION: 8-BIT SERIAL IN/PARALLEL OUT SHIFT REGISTER



Fixed-Function Device The 74HC164 is an example of a fixed-function IC shift register having serial in/parallel out operation. The logic block symbol is shown in Figure 8–8. This device has two gated serial inputs, *A* and *B*, and an asynchronous clear ($\overline{CLR}$) input that is active-LOW. The parallel outputs are Q_0 through Q_7.

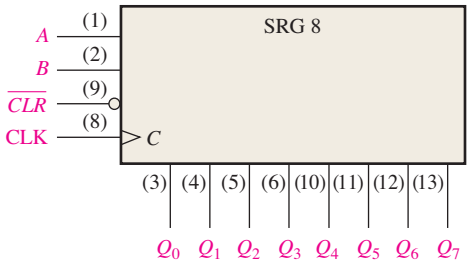


FIGURE 8–8 The 74HC164 8-bit serial in/parallel out shift register.

A sample timing diagram for the 74HC164 is shown in Figure 8–9. Notice that the serial input data on input *A* are shifted into and through the register after input *B* goes HIGH.

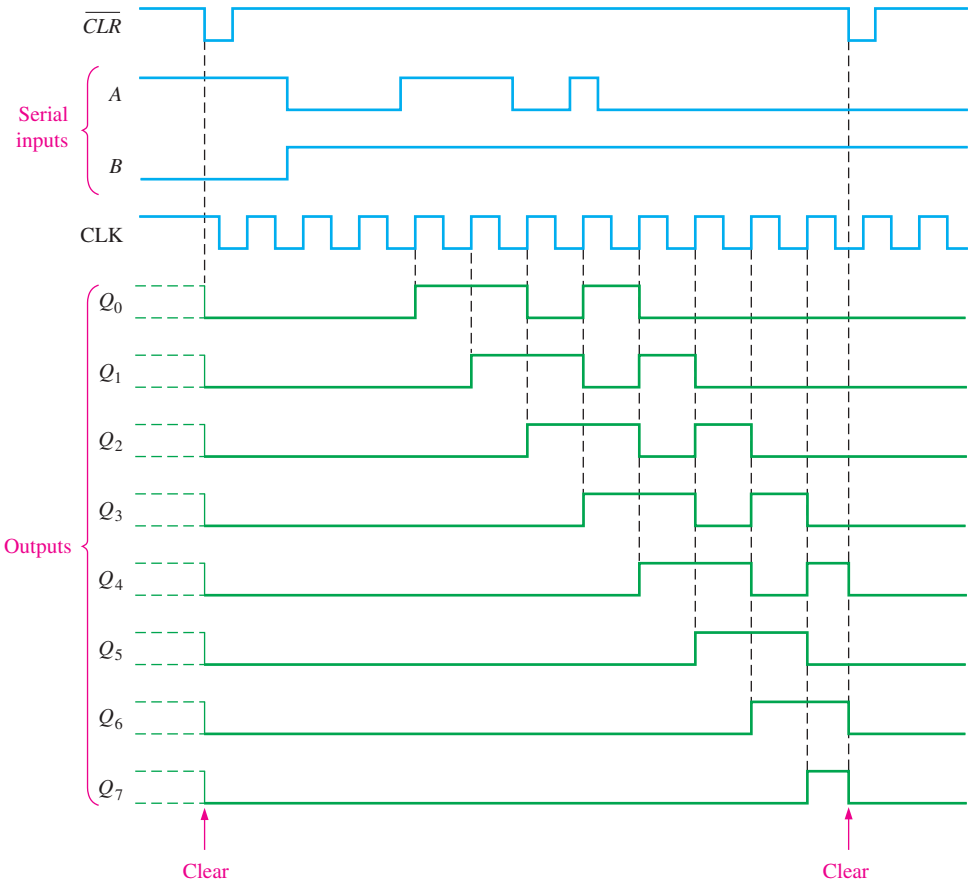


FIGURE 8–9 Sample timing diagram for a 74HC164 shift register.

Programmable Logic Device (PLD) The 8-bit serial in/parallel out shift register can be described using VHDL and implemented as hardware in a PLD. The program code is as follows. (Blue comments are not part of the program.)

```
library ieee;
use ieee.std_logic_1164.all;
entity SerInParOutShift is
   port (D0, Clock, Clr: in std_logic; Q0, Q1, Q2, Q3,
   Q4, Q5, Q6, Q7: inout std_logic);
end entity SerInParOutShift;
architecture LogicOperation of SerInParOutShift is
component dffl is
   port (D, Clock: in std_logic; Q: inout std_logic);
end component dff1;
begin
      FF0: dff1 port map(D=>D0 and Clr, Clock=>Clock, Q=>Q0);
      FF1: dff1 port map(D=>Q0 and Clr, Clock=>Clock, Q=>Q1);
      FF2: dff1 port map(D=>Q1 and Clr, Clock=>Clock, Q=>Q2);
      FF3: dff1 port map(D=>Q2 and Clr, Clock=>Clock, Q=>Q3);
      FF4: dff1 port map(D=>Q3 and Clr, Clock=>Clock, Q=>Q4);
      FF5: dff1 port map(D=>Q4 and Clr, Clock=>Clock, Q=>Q5);
      FF6: dff1 port map(D=>Q5 and Clr, Clock=>Clock, Q=>Q6);
      FF7: dff1 port map(D=>Q6 and Clr, Clock=>Clock, Q=>Q7);
   end architecture LogicOperation;
```

D0: Data input
Clock: System clock
Clr: Clear
Q0–Q7: Register outputs

D flip-flop with preset and clear inputs was described in Chapter 7 and is used as a component.

Instantiations describe how the flip-flops are connected to form the register.

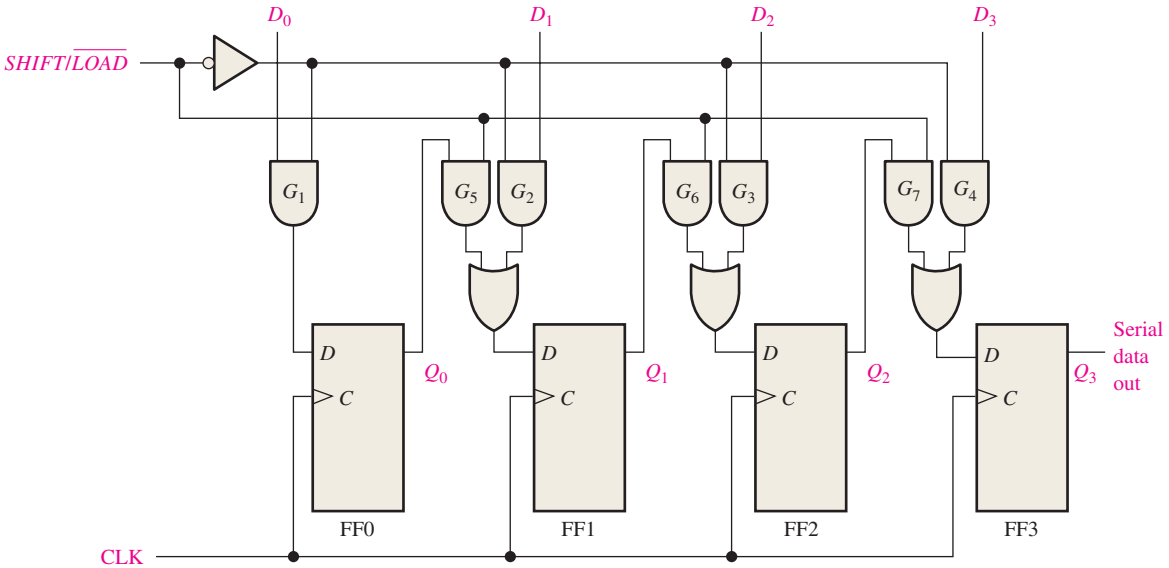
Parallel In/Serial Out Shift Registers

For a register with parallel data inputs, the bits are entered simultaneously into their respective stages on parallel lines rather than on a bit-by-bit basis on one line as with serial data inputs. The serial output is the same as in serial in/serial out shift registers, once the data are completely stored in the register.

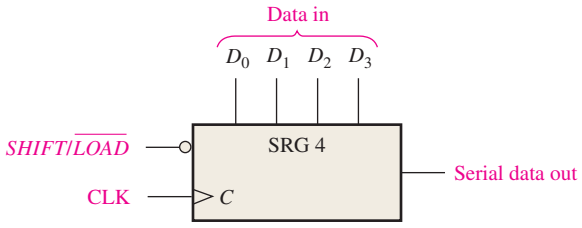
For parallel data, multiple bits are transferred at one time.

Figure 8–10 illustrates a 4-bit parallel in/serial out shift register and a typical logic symbol. There are four data-input lines, D_0, D_1, D_2, and D_3, and a $SHIFT/\overline{LOAD}$ input, which allows four bits of data to **load** in parallel into the register. When $SHIFT/\overline{LOAD}$ is LOW, gates G_1 through G_4 are enabled, allowing each data bit to be applied to the D input of its respective flip-flop. When a clock pulse is applied, the flip-flops with $D = 1$ will set and those with $D = 0$ will reset, thereby storing all four bits simultaneously.

When $SHIFT/\overline{LOAD}$ is HIGH, gates G_1 through G_4 are disabled and gates G_5 through G_7 are enabled, allowing the data bits to shift right from one stage to the next. The OR gates allow either the normal shifting operation or the parallel data-entry operation, depending on which AND gates are enabled by the level on the $SHIFT/\overline{LOAD}$ input. Notice that FF0 has a single AND to disable the parallel input, D_0. It does not require an AND/OR arrangement because there is no serial data in.



(a) Logic diagram



(b) Logic symbol

FIGURE 8–10 A 4-bit parallel in/serial out shift register. Open file F08-10 to verify operation.

MultiSim

EXAMPLE 8–3

Show the data-output waveform for a 4-bit register with the parallel input data and the clock and *SHIFT*/$\overline{LOAD}$ waveforms given in Figure 8–11(a). Refer to Figure 8–10(a) for the logic diagram.

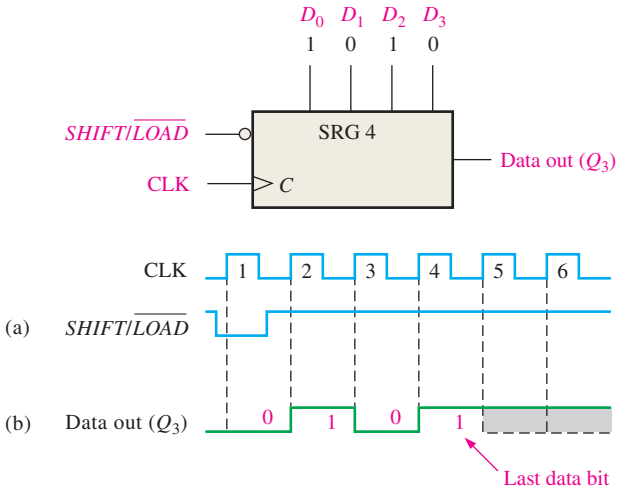


FIGURE 8–11

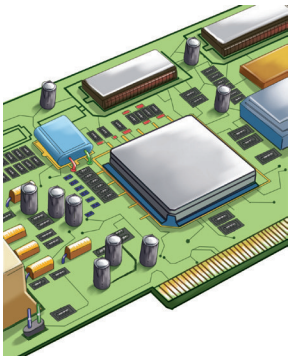
Solution

On clock pulse 1, the parallel data ($D_0D_1D_2D_3 = 1010$) are loaded into the register, making Q_3 a 0. On clock pulse 2 the 1 from Q_2 is shifted onto Q_3; on clock pulse 3 the 0 is shifted onto Q_3; on clock pulse 4 the last data bit (1) is shifted onto Q_3; and on clock pulse 5, all data bits have been shifted out, and only 1s remain in the register (assuming the D_0 input remains a 1). See Figure 8–11(b).

Related Problem

Show the data-output waveform for the clock and *SHIFT*/$\overline{LOAD}$ inputs shown in Figure 8–11(a) if the parallel data are $D_0D_1D_2D_3 = 0101$.

IMPLEMENTATION: 8-BIT PARALLEL LOAD SHIFT REGISTER



Fixed-Function Device The 74HC165 is an example of a fixed-function IC shift register that has a parallel in/serial out operation (it can also be operated as serial in/serial out). Figure 8–12 shows a typical logic block symbol. A LOW on the *SHIFT*/$\overline{LOAD}$ input (*SH*/$\overline{LD}$) enables asynchronous parallel loading. Data can be entered serially on the *SER* input. Also, the clock can be inhibited anytime with a HIGH on the *CLK INH* input. The serial data outputs of the register are Q_7 and its complement $\overline{Q}_7$. This implementation is different from the synchronous method of parallel loading previously discussed, demonstrating that there are usually several ways to accomplish the same function.

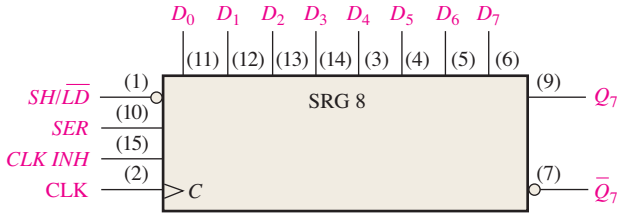


FIGURE 8–12 The 74HC165 8-bit parallel load shift register.

Figure 8–13 is a timing diagram showing an example of the operation of a 74HC165 shift register.

Programmable Logic Device (PLD) The 8-bit parallel load shift register is a parallel in/serial out device and can be implemented in a PLD with the following VHDL code:

```
library ieee:
use ieee.std_logic_1164.all;

entity ParSerShift is
    port (D0, D1, D2, D3, D4, D5, D6, D7, SHLD, Clock:
          in std_logic; Q, QNot: inout std_logic);
end entity ParSerShift;

architecture LogicOperation of ParSerShift is
    signal S1, S2, S3, S4, S5, S6, S7,
           Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7: std_logic;
function ShiftLoad (A,B,C: in std_logic)return std_logic is
begin
    return ((A and B) or (not B and C));
end function ShiftLoad;
```

D0-D7: Parallel input
SHLD: Shift Load input
Clock: System clock
Q: Serial output
QNot: Inverted serial output

S1-S7: Shift load signals from function ShiftLoad
Q0-Q7: Intermediate variables for flip-flop stages

Function ShiftLoad provides the AND-OR function shown in Figure 8–10 to allow the parallel load of data or data shift from one flip-flop stage to the next.

```
component dff1 is
port (D, Clock: in std_logic;
      Q: inout std_logic);
end component dff1;
```

D flip-flop component used as storage for shift register

```
begin
   SL1:S1 <=ShiftLoad(Q0, SHLD, D1);
   SL2:S2 <=ShiftLoad(Q1, SHLD, D2);
   SL3:S3 <=ShiftLoad(Q2, SHLD, D3);
   SL4:S4 <=ShiftLoad(Q3, SHLD, D4);
   SL5:S5 <=ShiftLoad(Q4, SHLD, D5);
   SL6:S6 <=ShiftLoad(Q5, SHLD, D6);
   SL7:S7 <=ShiftLoad(Q6, SHLD, D7);
   FF0: dff1 port map(D=>D0 and not SHLD, Clock=>Clock, Q=>Q0);
   FF1: dff1 port map(D=>S1, Clock=>Clock, Q=>Q1);
   FF2: dff1 port map(D=>S2, Clock=>Clock, Q=>Q2);
   FF3: dff1 port map(D=>S3, Clock=>Clock, Q=>Q3);
   FF4: dff1 port map(D=>S4, Clock=>Clock, Q=>Q4);
   FF5: dff1 port map(D=>S5, Clock=>Clock, Q=>Q5);
   FF6: dff1 port map(D=>S6, Clock=>Clock, Q=>Q6);
   FF7: dff1 port map(D=>S7, Clock=>Clock, Q=>Q);
   QNot <=not Q;
end architecture LogicOperation;
```

ShiftLoad instances SL1–SL7 allow eight bits of data to load into flip-flop stages FF0–FF7 or to shift through the register providing the parallel load serial out function.

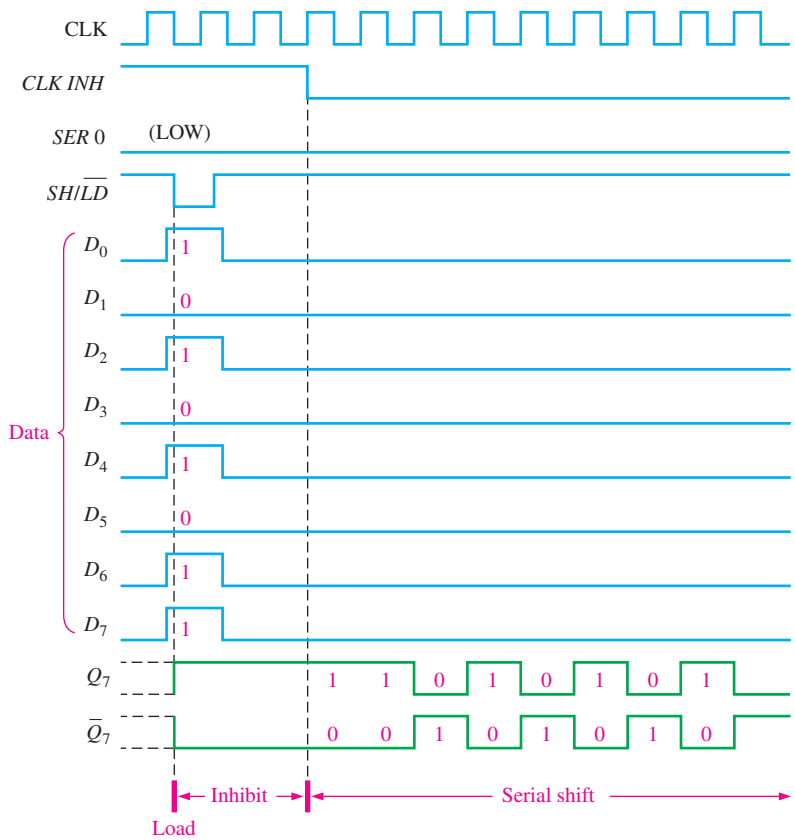


FIGURE 8–13 Sample timing diagram for a 74HC165 shift register.

Parallel In/Parallel Out Shift Registers

Parallel entry and parallel output of data have been discussed. The parallel in/parallel out register employs both methods. Immediately following the simultaneous entry of all data bits, the bits appear on the parallel outputs. Figure 8–14 shows a parallel in/parallel out shift register.

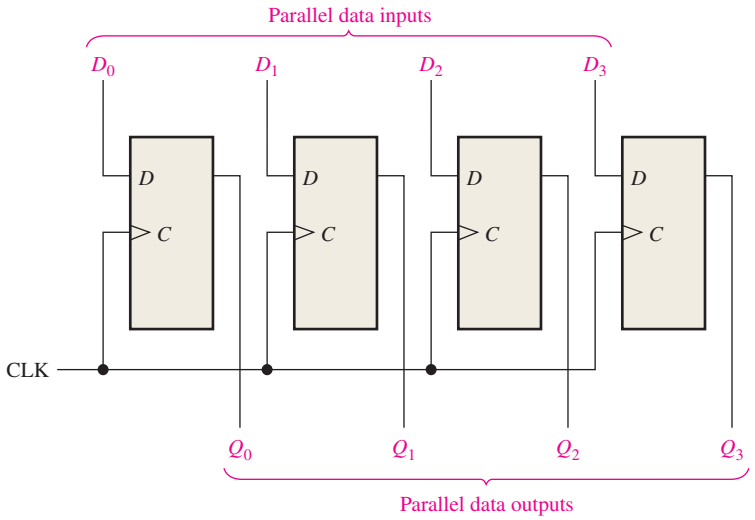
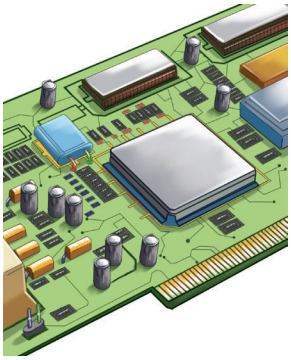


FIGURE 8–14 A parallel in/parallel out register.

IMPLEMENTATION: 4-BIT PARALLEL-ACCESS SHIFT REGISTER



Fixed-Function Device The 74HC195 can be used for parallel in/parallel out operation. Because it also has a serial input, it can be used for serial in/serial out and serial in/parallel out operations. It can be used for parallel in/serial out operation by using Q_3 as the output. A typical logic block symbol is shown in Figure 8–15.

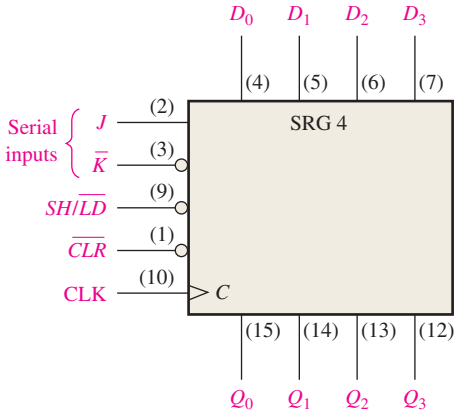


FIGURE 8–15 The 74HC195 4-bit parallel access shift register.

When the $SHIFT/\overline{LOAD}$ input ($SH/\overline{LD}$) is LOW, the data on the parallel inputs are entered synchronously on the positive transition of the clock. When ($SH/\overline{LD}$) is HIGH, stored data will shift right (Q_0 to Q_3) synchronously with the clock. Inputs J and $\overline{K}$ are the serial data inputs to the first stage of the register (Q_0); Q_3 can be used for serial output data. The active-LOW clear input is asynchronous.