Digital System Design

Electrical Department-Fourth Stage

Lecture Three

By
Assistant Lecturer: Adnan Ali Abdullah

2025-2024

Shift Registers

Lecture Outline

- Shift Register Operations
- Types of Shift Register Data I/Os
- Bidirectional Shift Registers
- Shift Register Counters
- Shift Register Applications

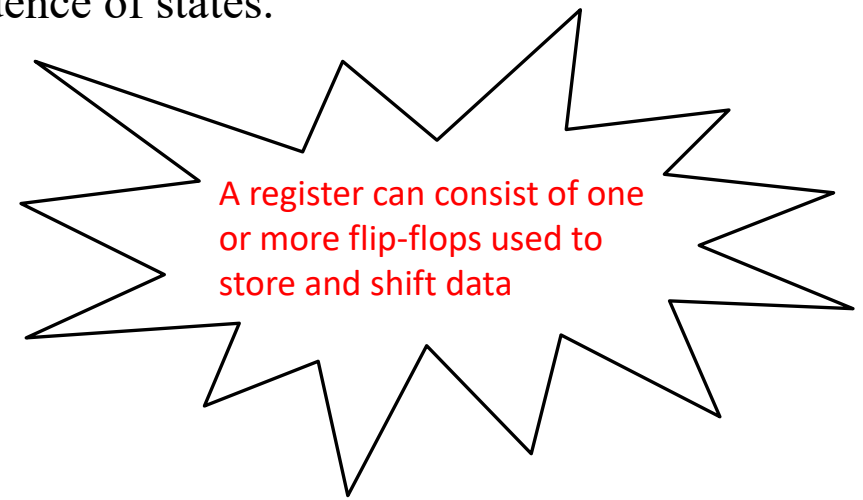
Introduction

Shift registers are a type of sequential logic circuit used primarily for the storage of digital data and typically do not possess a characteristic internal sequence of states.

❑ 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.

- 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 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**.

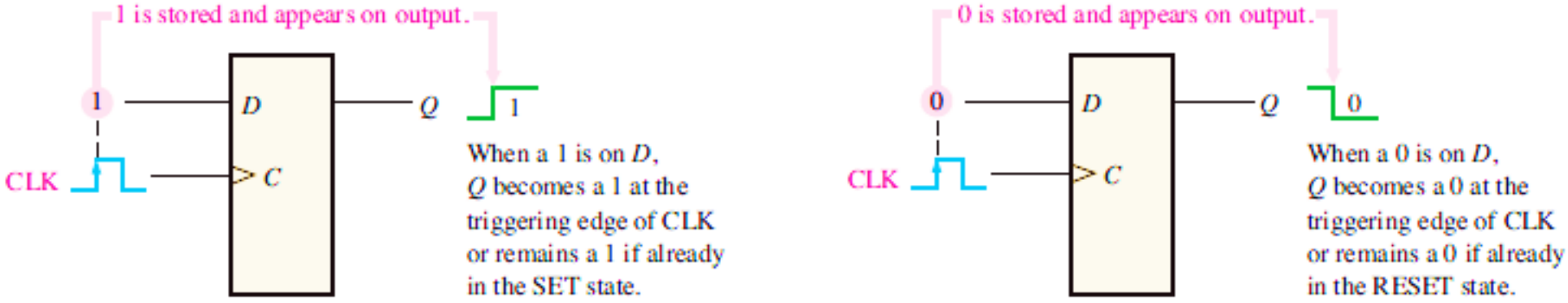


FIGURE 1 The flip-flop as a storage element.

- 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 2 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.

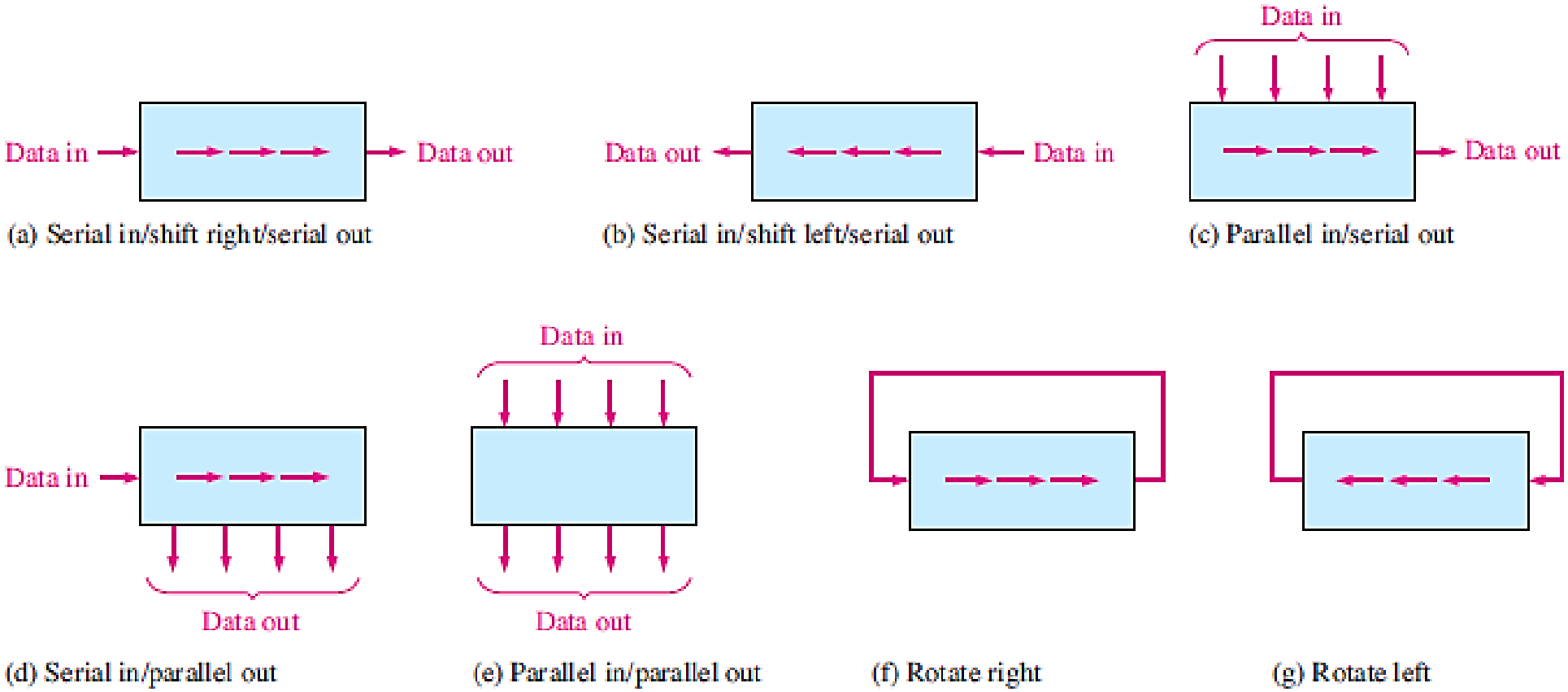


FIGURE 2 Basic data movement in shift registers. (Four bits are used for illustration. The bits move in the direction of the arrows.

❑ 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.

1. 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.
- Figure 3 shows a 4-bit device implemented with D flip-flops. With four stages, this register can store up to four bits of data.

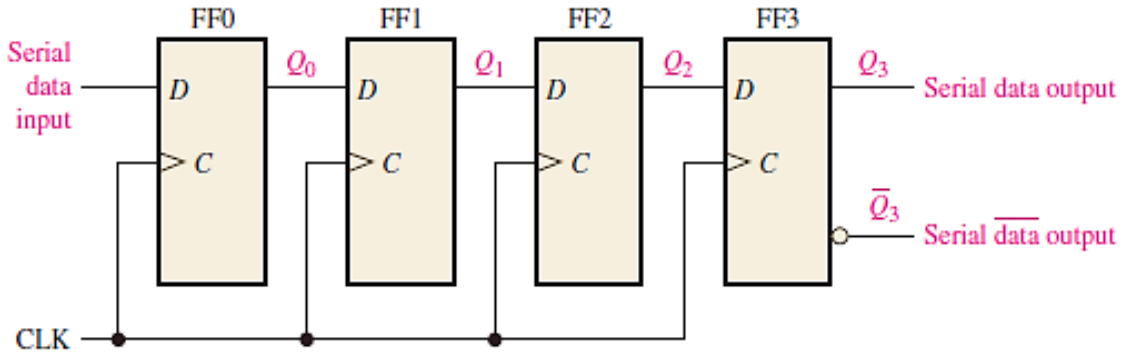


FIGURE 3 Serial in/serial out shift register.

- Table 1 shows the entry of the four bits 1010 into the register in Figure 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.

- 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 Q0 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.

TABLE 1

Shifting a 4-bit code into the shift register in Figure 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**

- 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.

- 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 Q3 output, as Table 8–2 illustrates. After CLK4 in the data-entry operation just described, the LSB, 0, appears on the Q3 output. When clock pulse CLK5 is applied, the second bit appears on the Q3 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

Show the states of the 5-bit register in Figure 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
$Q_4 Q_3 Q_2 Q_1 Q_0 = 11010$ after five clock pulses. See Figure 4(b).

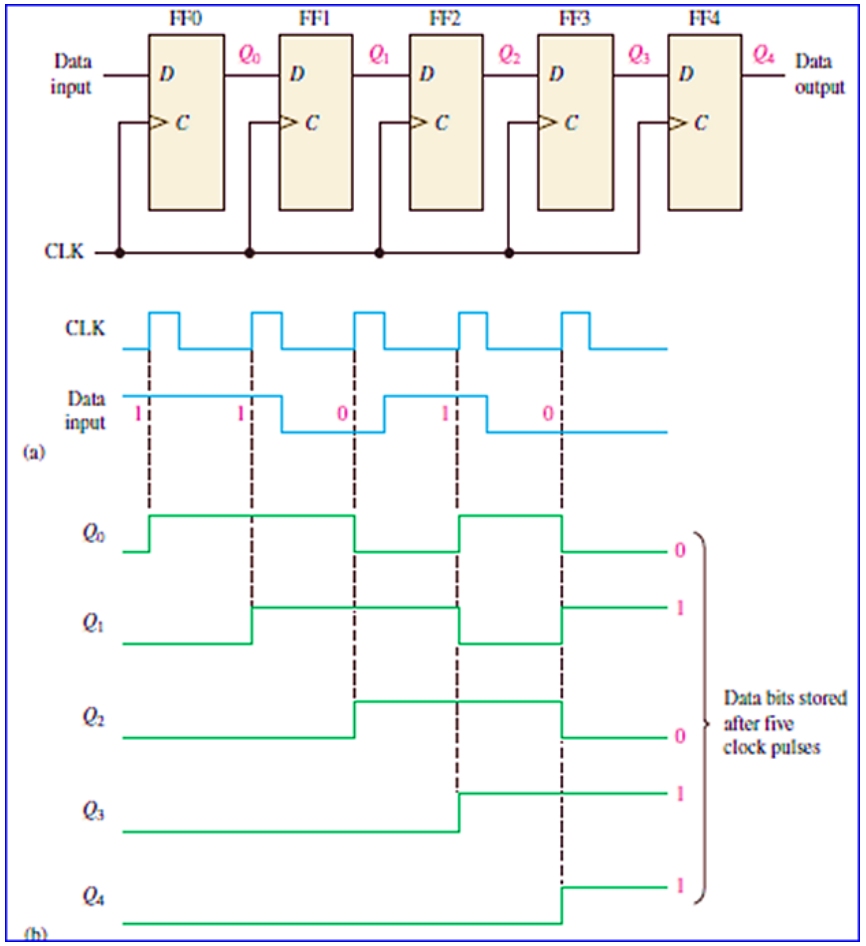


FIGURE 4

Related Problem*

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

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

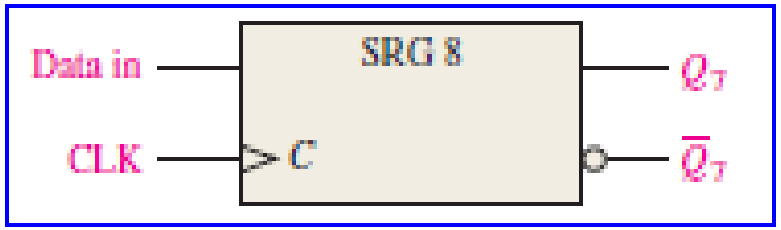


FIGURE 5 Logic symbol for an 8-bit serial in/serial out shift register.

2. 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 6 shows a 4-bit serial in/parallel out shift register and its logic block symbol.

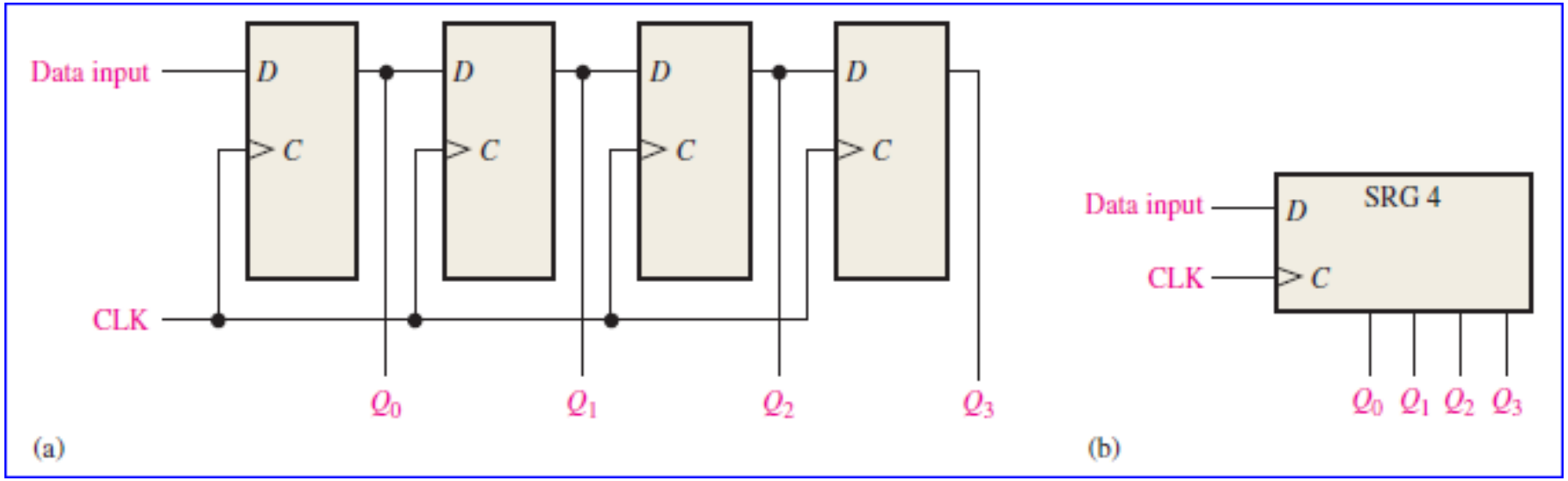


FIGURE 6 A serial in/parallel out shift register.

Example

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

Solution

The register contains 0110 after four clock pulses. See Figure 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?

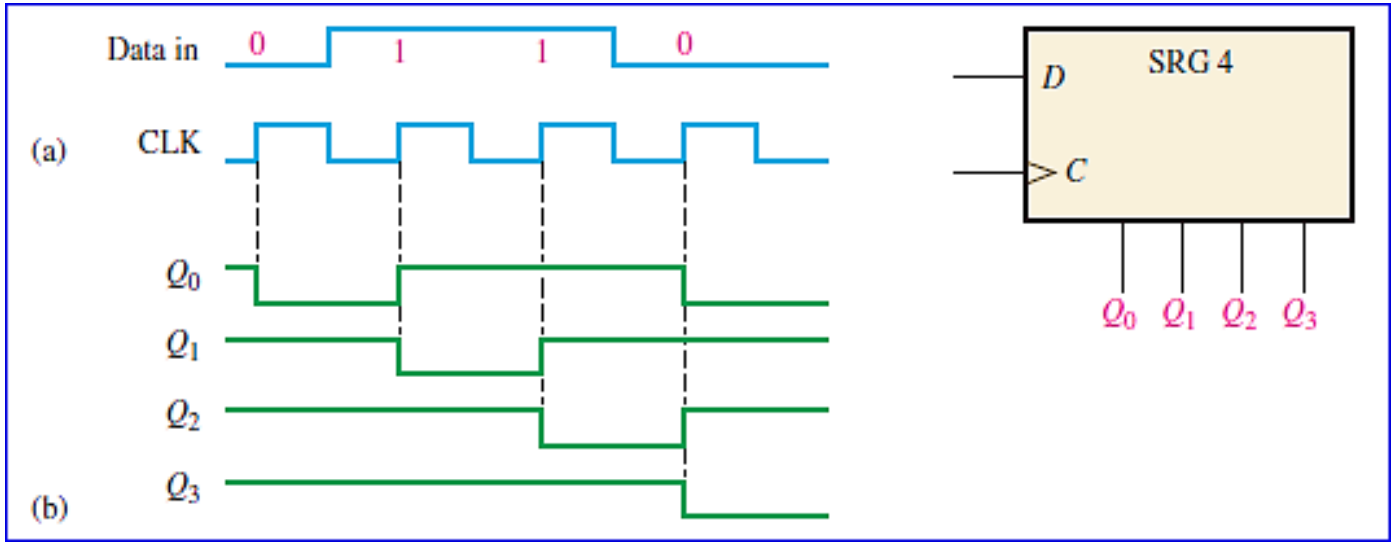


FIGURE 7

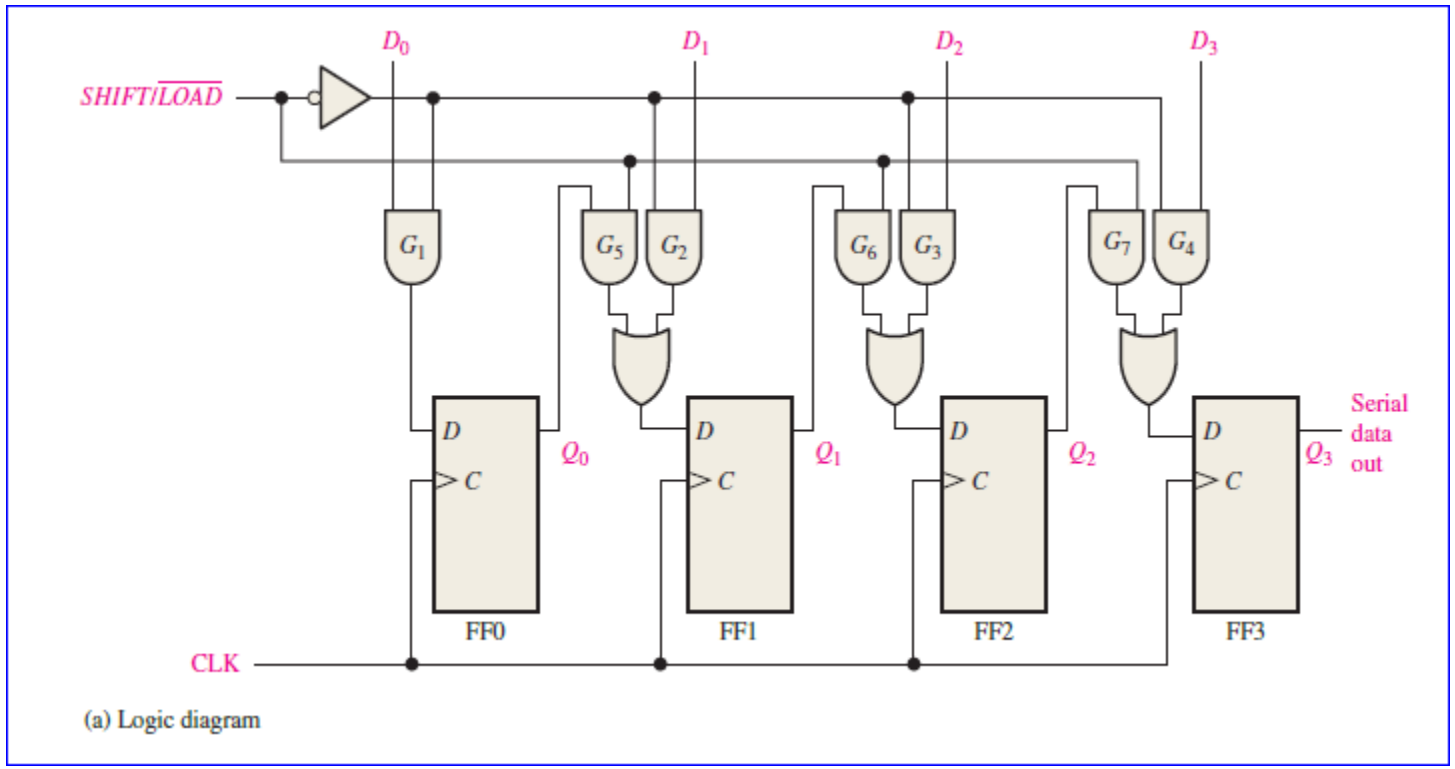
3. Parallel In/Serial Out Shift Registers

- 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

- Figure 10 illustrates a 4-bit parallel in/serial out shift register and a typical logic symbol. There are four data-input lines, D0, D1, D2, and D3, and a SHIFT/$\overline{\text{LOAD}}$ input, which allows four bits of data to load in parallel into the register. When SHIFT/$\overline{\text{LOAD}}$ is LOW, gates G1 through G4 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.

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

- When SHIFT/$\overline{\text{LOAD}}$ is HIGH, gates G1 through G4 are disabled and gates G5 through G7 are enabled, allowing the data bits to shift right from one stage to the next. The OR gates mallow either the normal shifting operation or the parallel data-entry operation, depending on which AND gates are enabled by the level on the SHIFT/$\overline{\text{LOAD}}$ input. Notice that FF0 has a single AND to disable the parallel input, D0. It does not require an AND/OR arrangement because there is no serial data in.



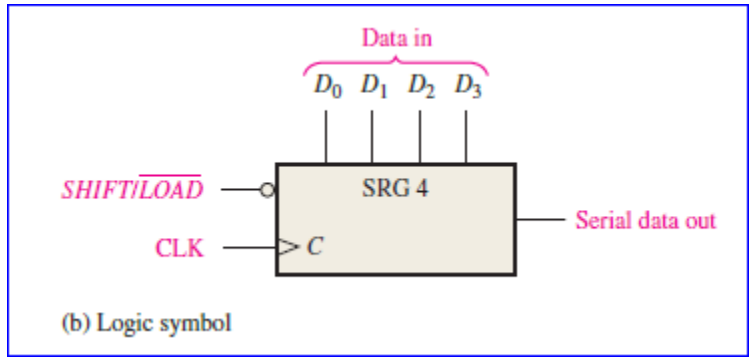


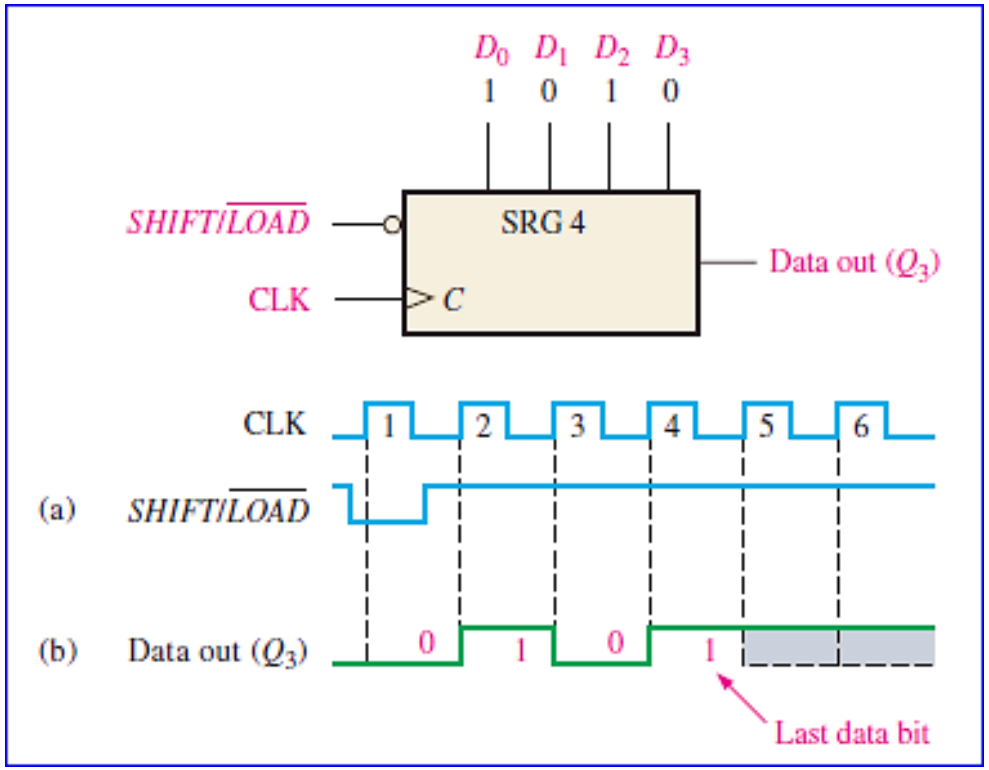
FIGURE 10 A 4-bit parallel in/serial out shift register

Example

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

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 11(b).



Related Problem

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

❑ Bidirectional Shift Registers

- A **bidirectional** shift register is one in which the data can be shifted either left or right. It can be implemented by using gating logic that enables the transfer of a data bit from one stage to the next stage to the right or to the left, depending on the level of a control line.

A 4-bit bidirectional shift register is shown in Figure 17. A HIGH on the RIGHT/$\overline{\text{LEFT}}$ control input allows data bits inside the register to be shifted to the right, and a LOW enables data bits inside the register to be shifted to the left.

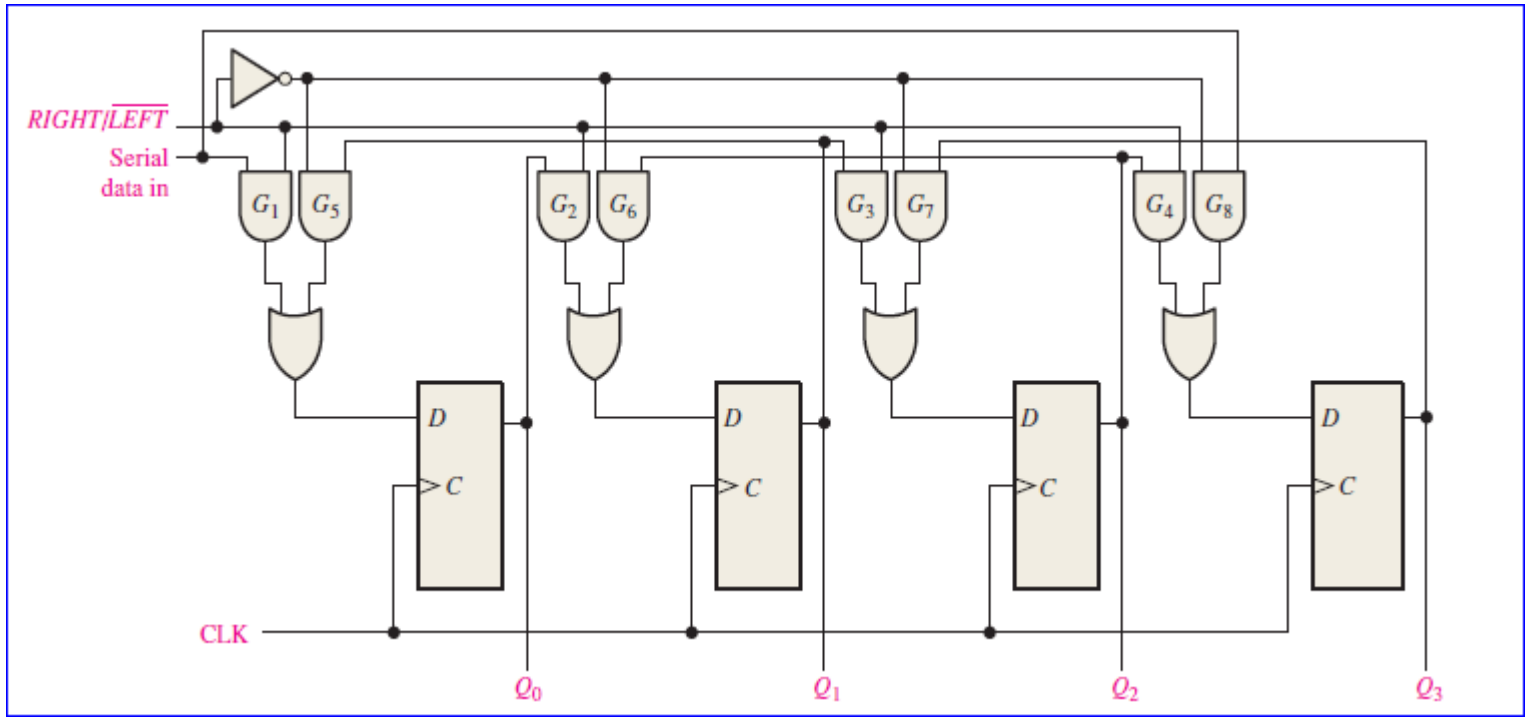


FIGURE 17 Four-bit bidirectional shift register

- An examination of the gating logic will make the operation apparent. When the RIGHT/$\overline{\text{LEFT}}$ control input is HIGH, gates G_1 through G_4 are enabled, and the state of the Q output of each flip-flop is passed through to the D input of the *following* flip-flop. When a clock pulse occurs, the data bits are shifted one place to the *right.* When the RIGHT/$\overline{\text{LEFT}}$ control input is LOW, gates G_5 through G_8 are enabled, and the Q output of each flip-flop is passed through to the D input of the *preceding* flip-flop. When a clock pulse occurs, the data bits are then shifted one place to the *left.*

Example

Determine the state of the shift register of Figure 17 after each clock pulse for the given RIGHT/$\overline{\text{LEFT}}$ control input waveform in Figure 18(a). Assume that $Q0 = 1$, $Q1 = 1$, $Q2 = 0$, and $Q3 = 1$ and that the serial data-input line is LOW.

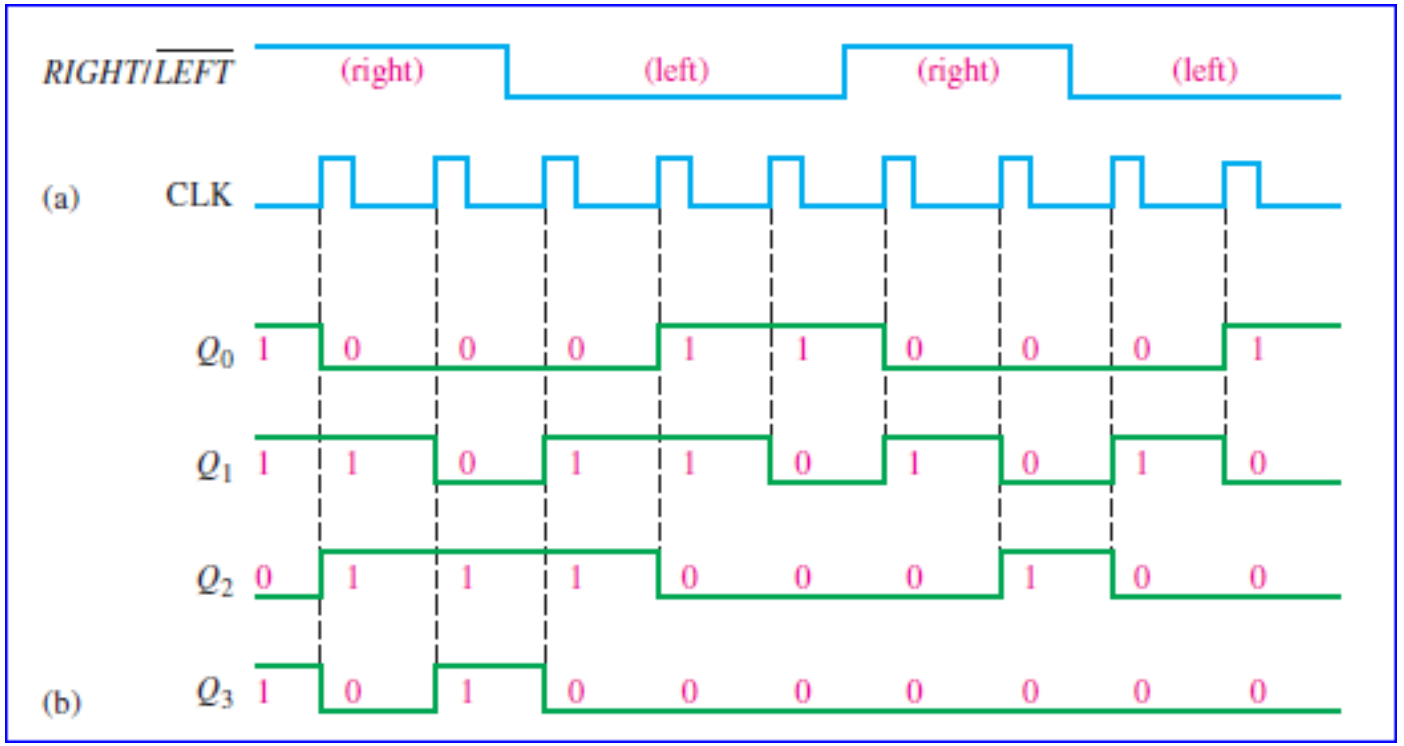


FIGURE 18

Solution

See Figure 18(b).

❑ Shift Register Counters

- A shift register counter is basically a shift register with the serial output connected back to the serial input to produce special sequences. These devices are often classified as counters because they exhibit a specified sequence of states. Two of the most common types of shift register counters, the Johnson counter and the ring counter, are introduced in this section

The Johnson Counter

- In a **Johnson counter** the complement of the output of the last flip-flop is connected back to the *D* input of the first flip-flop (it can be implemented with other types of flip-flops as well). If the counter starts at 0, this feedback arrangement produces a characteristic sequence of states, as shown in Table 3 for a 4-bit device and in Table 4 for a 5-bit device.

  - Notice that the 4-bit sequence has a total of eight states, or bit patterns, and that the 5-bit sequence has a total of ten states. In general, a Johnson counter will produce a modulus of 2n, where n is the number of stages in the counte

- The implementations of the 4-stage and 5-stage johnson counters are shown in Figure 21. The implementation of a Johnson counter is very straightforward and is the same regardless of the number of stages. The *Q* output of each stage is connected to the *D* input of the next

TABLE 3

Four-bit Johnson sequence.

Clock Pulse	Q_0	Q_1	Q_2	Q_3
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1

TABLE 4

Five-bit Johnson sequence.

Clock Pulse	Q_0	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0	0
1	1	0	0	0	0
2	1	1	0	0	0
3	1	1	1	0	0
4	1	1	1	1	0
5	1	1	1	1	1
6	0	1	1	1	1
7	0	0	1	1	1
8	0	0	0	1	1
9	0	0	0	0	1

- the next stage (assuming that *D* flip-flops are used). The single exception is that the *Q* output of the last stage is connected back to the *D* input of the first stage. As the sequences in Table 3 and 8–4 show, if the counter starts at 0, it will "fill up" with 1s from left to right, and then it will "fill up" with 0s again.

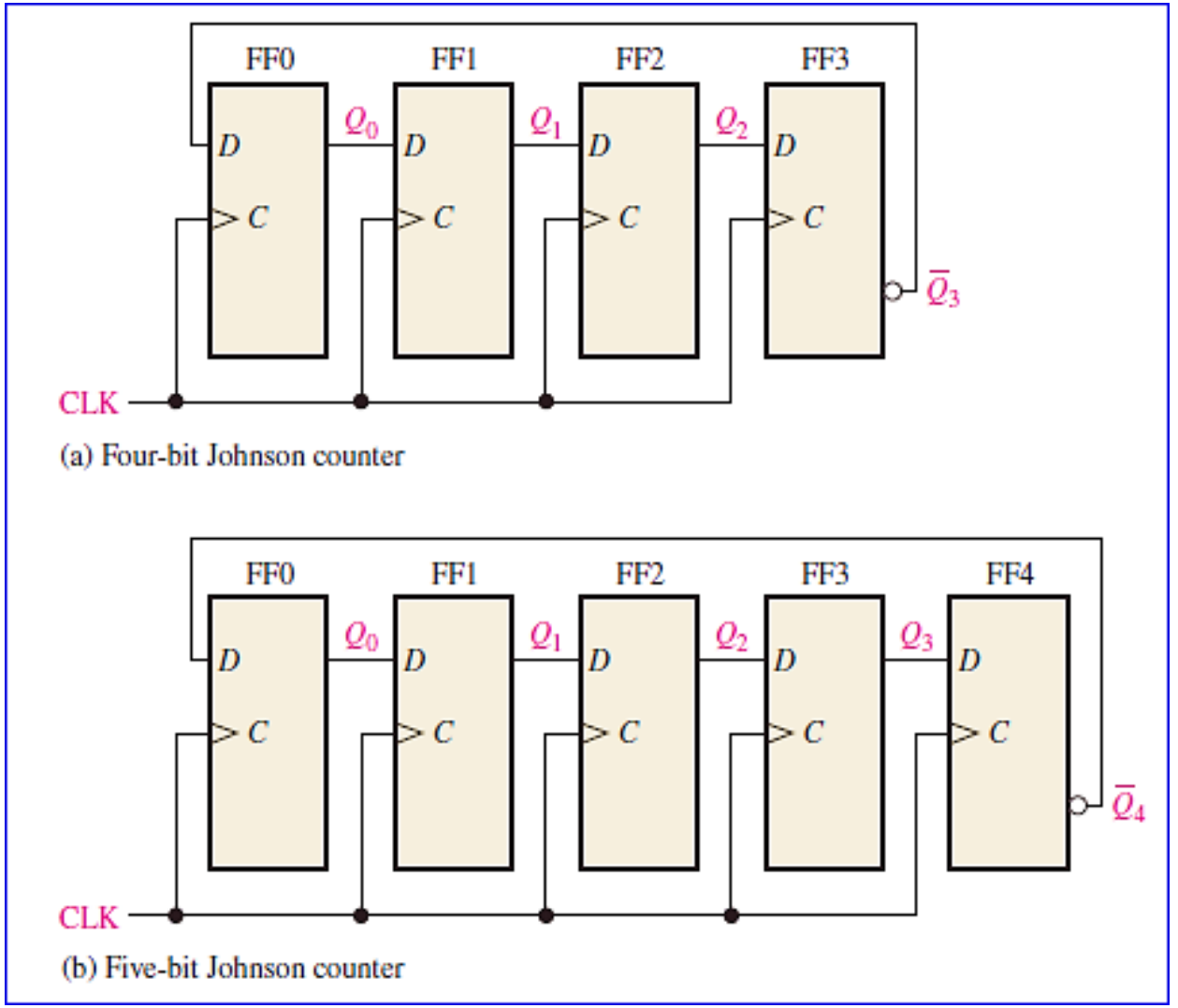


FIGURE 21 Four-bit and 5-bit Johnson counters.

- Diagrams of the timing operations of the 4-bit and 5-bit counters are shown in Figures 22 and 23, respectively.

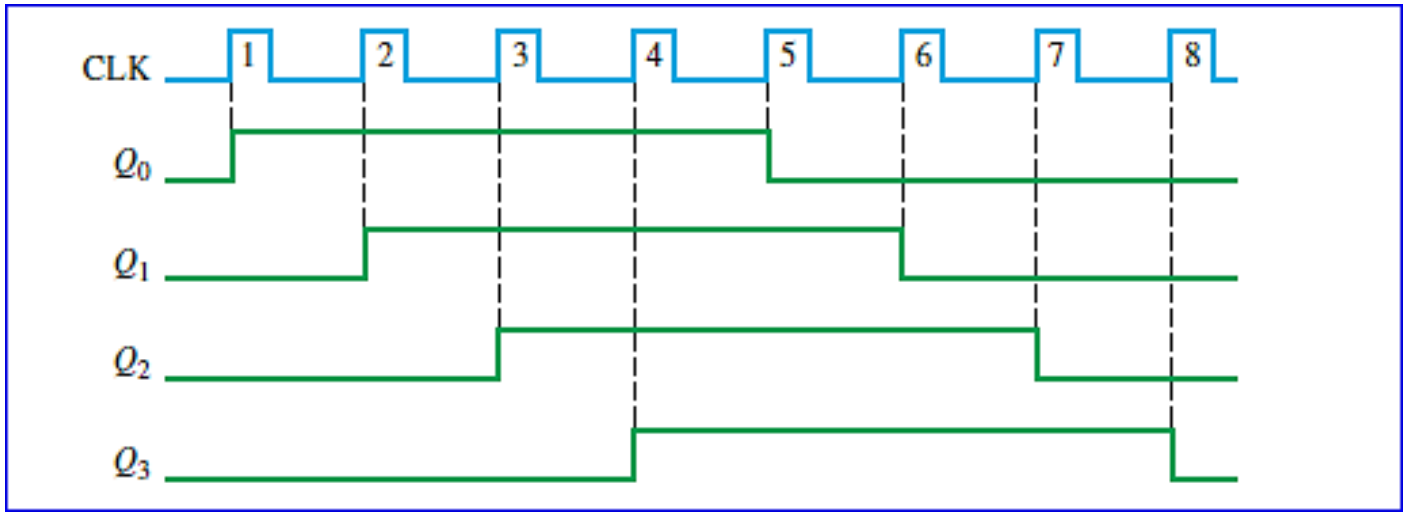


FIGURE 22 Timing sequence for a 4-bit Johnson counter.

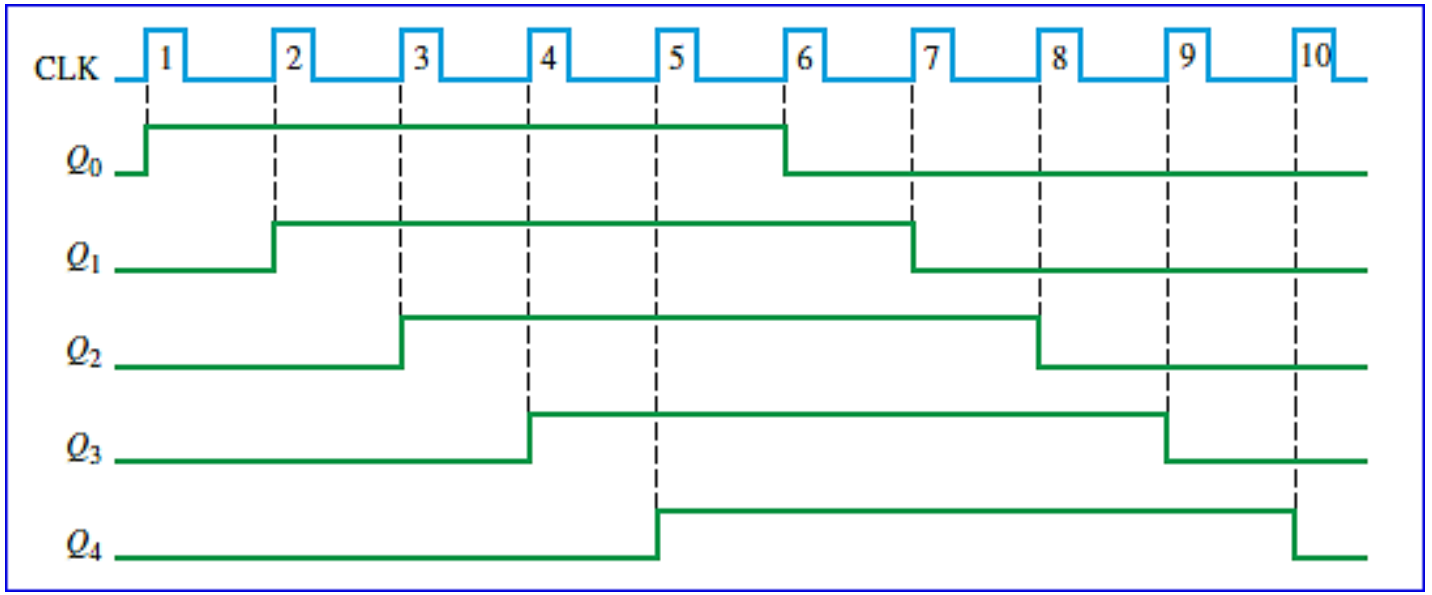


FIGURE 23 Timing sequence for a 5-bit Johnson counter

❑ The Ring Counter

- A **ring counter** utilizes one flip-flop for each state in its sequence. It has the advantage that decoding gates are not required. In the case of a 10-bit ring counter, there is a unique output for each decimal digit.

- A logic diagram for a 10-bit ring counter is shown in Figure 24. The sequence for this ring counter is given in Table 5. Initially, a 1 is preset into the first flip-flop, and the rest of the flip-flops are cleared. Notice that the interstage connections are the same as those for a Johnson counter, except that Q rather than Q is fed back from the last stage.

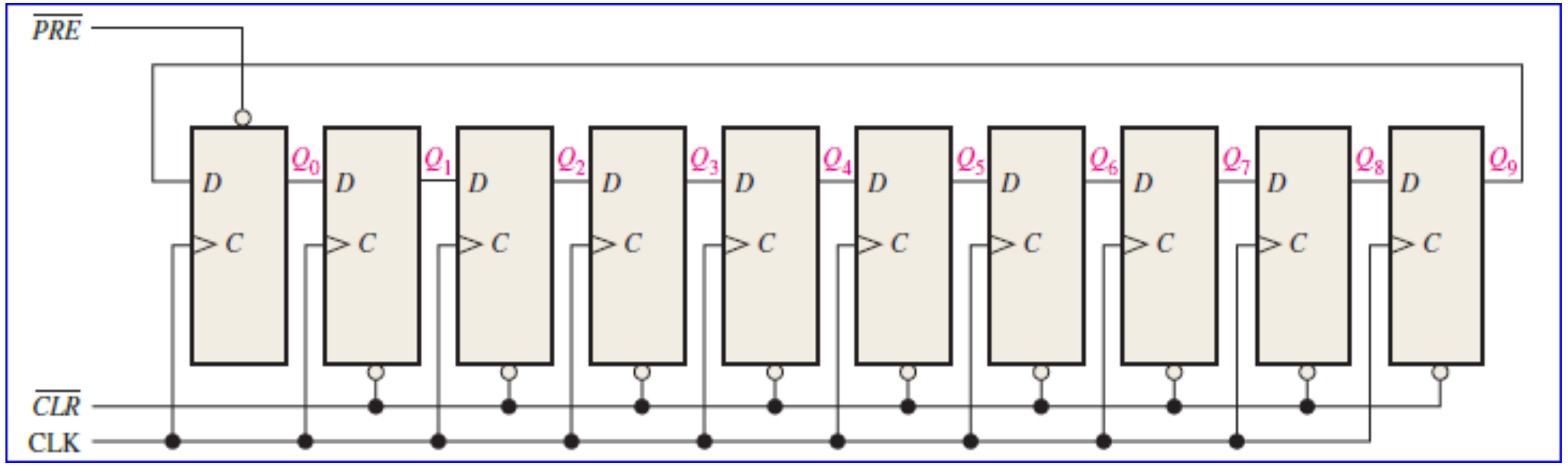


FIGURE 24 A 10-bit ring counter.

TABLE 5-5

Ten-bit ring counter sequence.

Clock Pulse	Q_0	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9
0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	0	0	0	0	0
3	0	0	0	1	0	0	0	0	0	0
4	0	0	0	0	1	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0
6	0	0	0	0	0	0	1	0	0	0
7	0	0	0	0	0	0	0	1	0	0
8	0	0	0	0	0	0	0	0	1	0
9	0	0	0	0	0	0	0	0	0	1

The ten outputs of the counter indicate directly the decimal count of the clock pulse. For instance, a 1 on $Q0$ represents a zero, a 1 on $Q1$ represents a one, a 1 on $Q2$ represents a two, a 1 on $Q3$ represents a three, and so on. You should verify for yourself that the 1 is always retained in the counter and simply shifted "around the ring," advancing one stage for each clock pulse. Modified sequences can be achieved by having more than a single 1 in the counter, as
illustrated in next Example.

Example

If a 10-bit ring counter similar to Figure 24 has the initial state 1010000000, determine the waveform for each of the Q outputs.

Solution

See Figure 25.

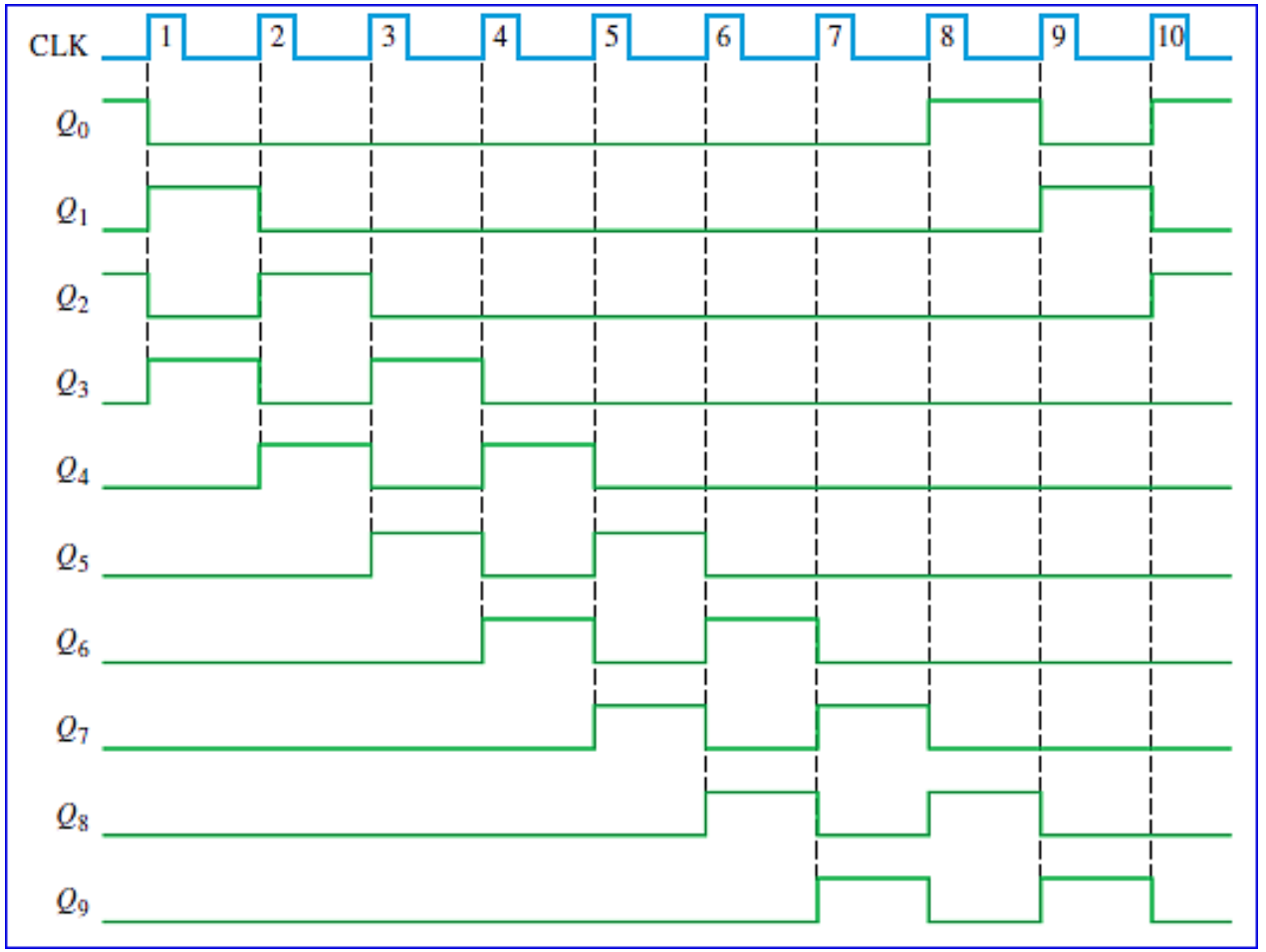


Figure 25

Related Problem

If a 10-bit ring counter has an initial state 0101001111, determine the waveform for each Q output.

❑ Shift Register Applications

❑ Time Delay

- A serial in/serial out shift register can be used to provide a time delay from input to output that is a function of both the number of stages (n) in the register and the clock frequency.

- When a data pulse is applied to the serial input as shown in Figure 26 it enters the first stage on the triggering edge of the clock pulse. It is then shifted from stage to stage on each successive clock pulse until i appears on the serial output n clock periods later.

- This time delay operation is illustrated in Figure 26, in which an 8-bit serial in/serial out shift register is used with a clock frequency of 1 MHz to achieve a time delay (td) of 8 μs (8 $\times$1 μs). This time can be adjusted up or down by changing the clock frequency. The time delay can also be increased by cascading shift registers and decreased by taking the outputs from successively lower stages in the register if the outputs are available, as illustrated in next Example.

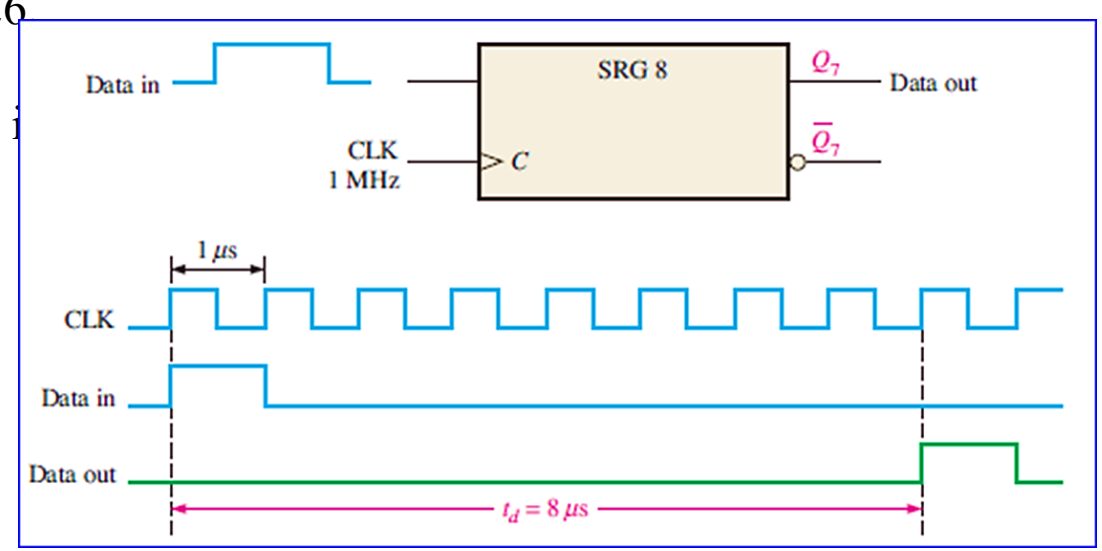


FIGURE 26 The shift register as a time-delay device.

Example

Determine the amount of time delay between the serial input and each output in Figure 27. Show a timing diagram to illustrate.

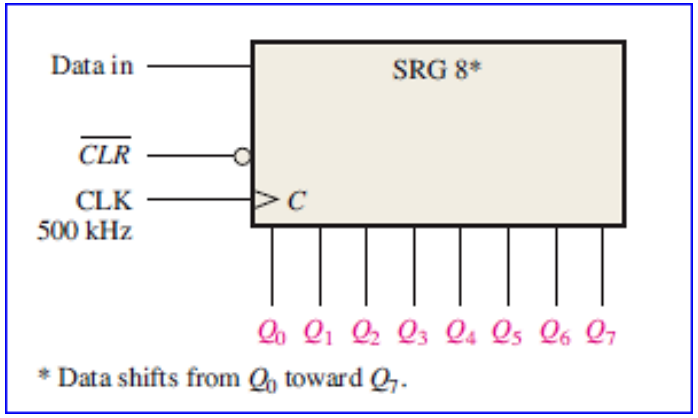


FIGURE 27

Solution

The clock period is 2 μs. Thus, the time delay can be increased or decreased in 2 μs increments from a minimum of 2 μs to a maximum of 16 πs, as illustrated in Figure 28.

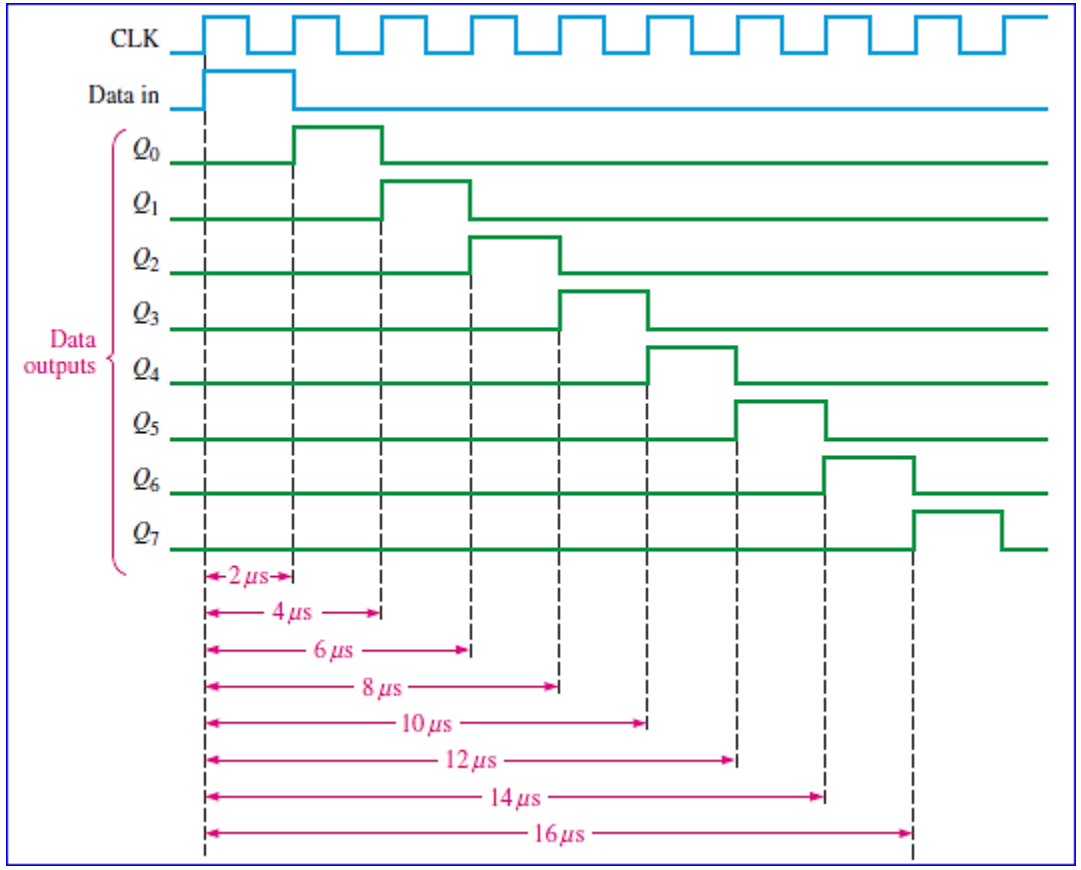


FIGURE 28 Timing diagram showing time delays for the register in Figure 27

❑ Serial-to-Parallel Data Converter

- Serial data transmission from one digital system to another is commonly used to reduce the number of wires in the transmission line. For example, eight bits can be sent serially over one wire, but it takes eight wires to send the same data in parallel.

- Serial data transmission is widely used by peripherals to pass data back and forth to a computer. For example, USB (universal serial bus) is used to connect keyboards printers, scanners, and more to the computer. All computers process data in parallel form, thus the requirement for serial-to-parallel conversion. A simplified serial-to-parallel data converter, in which two types of shift registers are used, is shown in Figure 31.

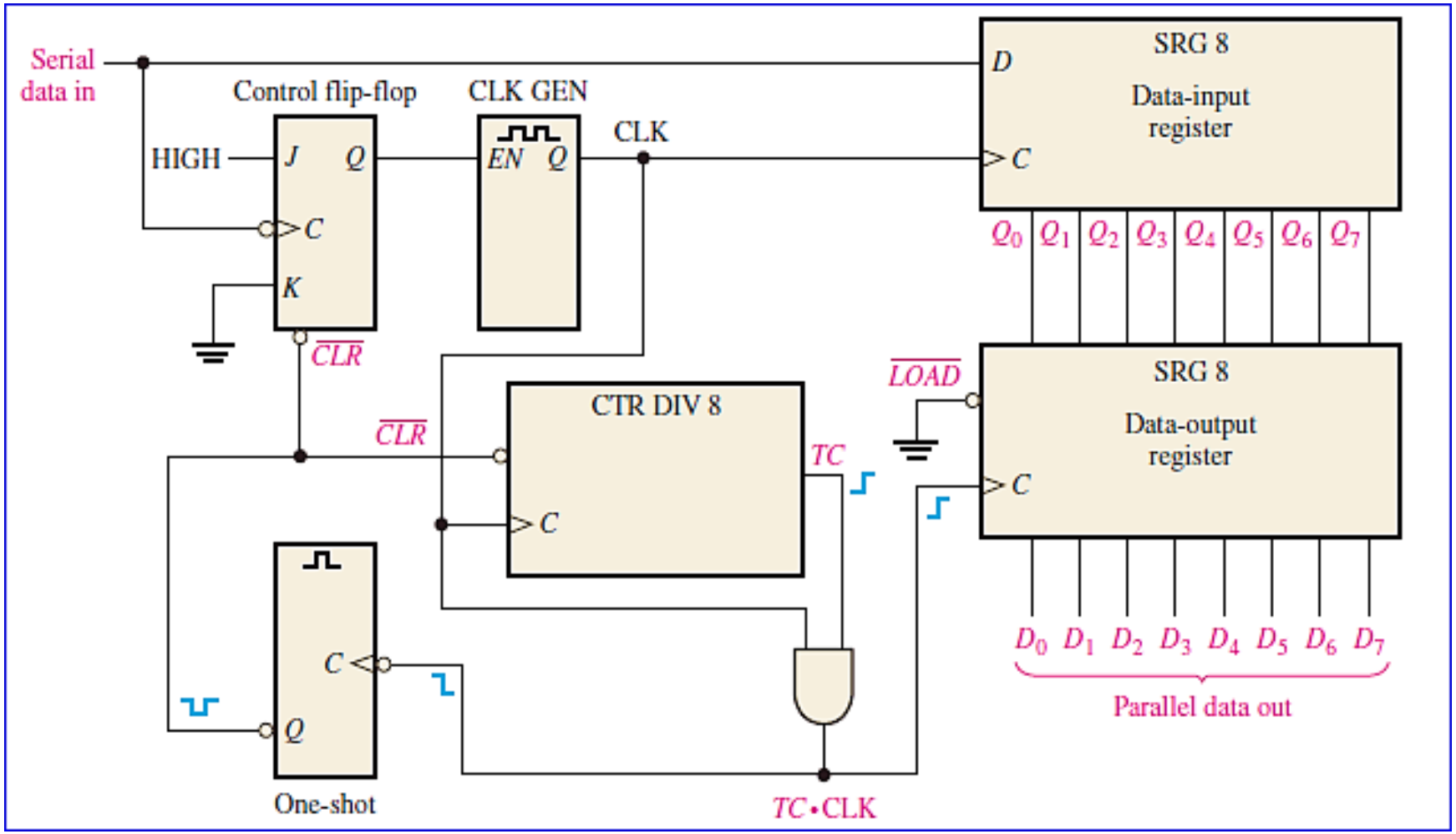


FIGURE 31 Simplified logic diagram of a serial-to-parallel converter.

- To illustrate the operation of this serial-to-parallel converter, the serial data format shown in Figure 32 is used. It consists of eleven bits. The first bit (start bit) is always 0 and always begins with a HIGH-to-LOW transition. The next eight bits (D_7 through D_0) are the data bits (one of the bits can be parity), and the last one or two bits (stop bits) are always 1s. When no data are being sent, there is a continuous HIGH on the serial data line.

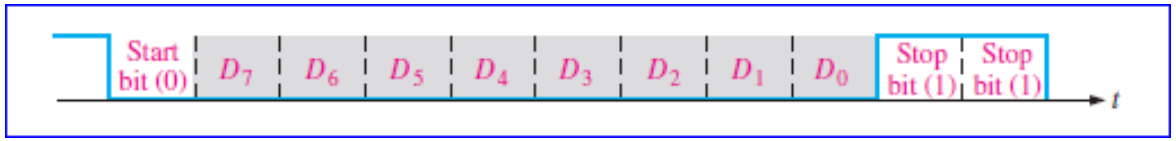


FIGURE 32 Serial data format.

- The HIGH-to-LOW transition of the start bit sets the control flip-flop, which enables the clock generator. After a fixed delay time, the clock generator begins producing a pulse waveform, which is applied to the data-input register and to the divide-by-8 counter. The clock has a frequency precisely equal to that of the incoming serial data, and the first clock pulse after the start bit occurs during the first data bit.

- The timing diagram in Figure 33 illustrates the following basic operation: The eight data bits (D_7 through D_0) are serially shifted into the data-input register. Shortly after th

- eighth clock pulse, the terminal count (TC) goes from LOW to HIGH, indicating the counter is at the last state. This rising edge is AND ed with the clock pulse, which is still HIGH, producing a rising edge at TC # CLK. This parallel loads the eight data bits from the data input shift register to the data-output register. A short time later, the clock pulse goes LOW and this HIGH-to-LOW transition triggers the one-shot, which produces a short-duration pulse to clear the counter and reset the control flip-flop and thus disable the clock generator. The system is now ready for the next group of eleven bits, and it waits for the next HIGH-to-LOW transition at the beginning of the start bit. By reversing the process just stated, parallel-to-serial data conversion can be accomplished. Since the serial data format must be produced, start and stop bits must be added to the sequence.

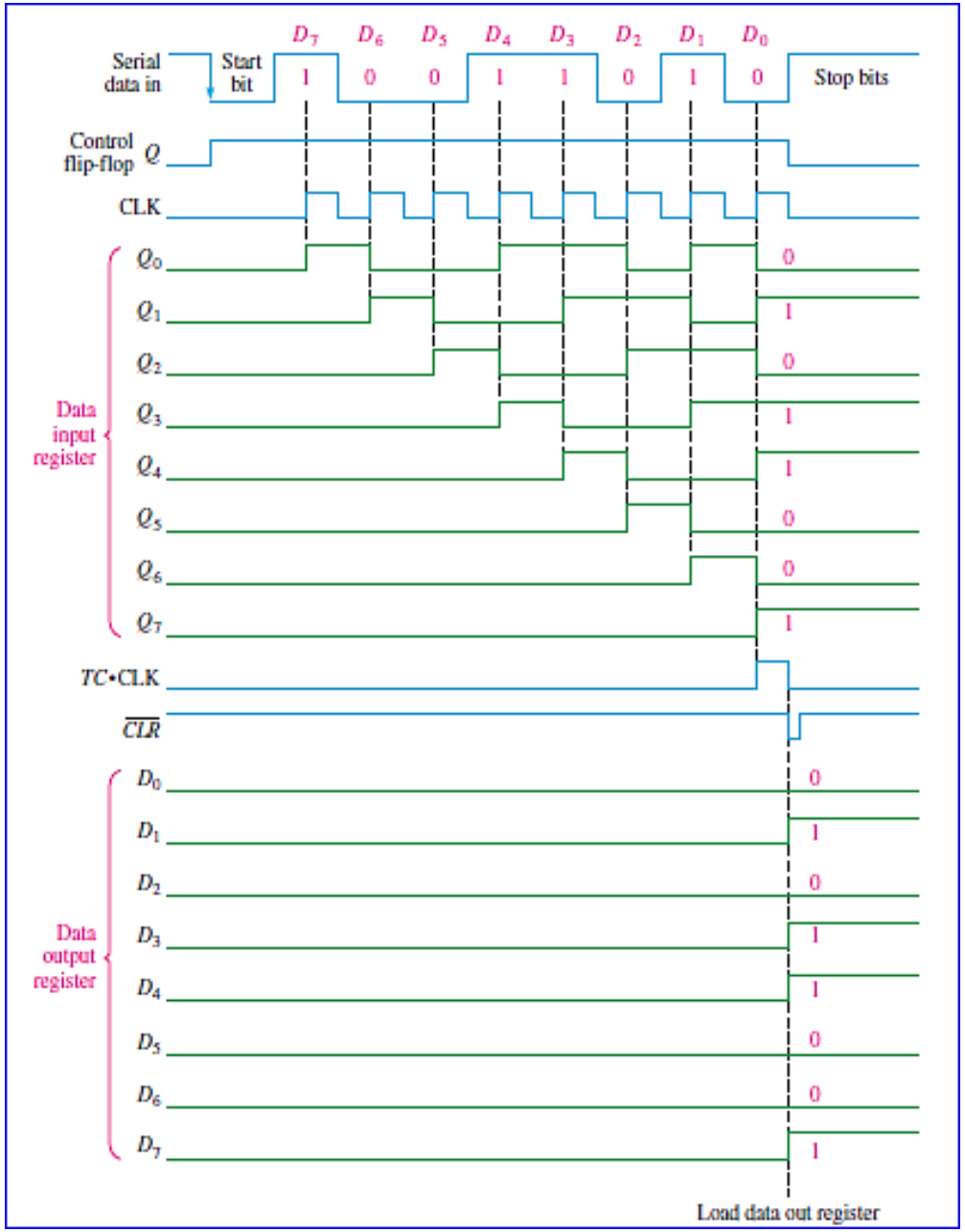


FIGURE 33 Timing diagram illustrating the operation of the serial-to-parallel data converter in Figure 31.