Shift Registers:

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** 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

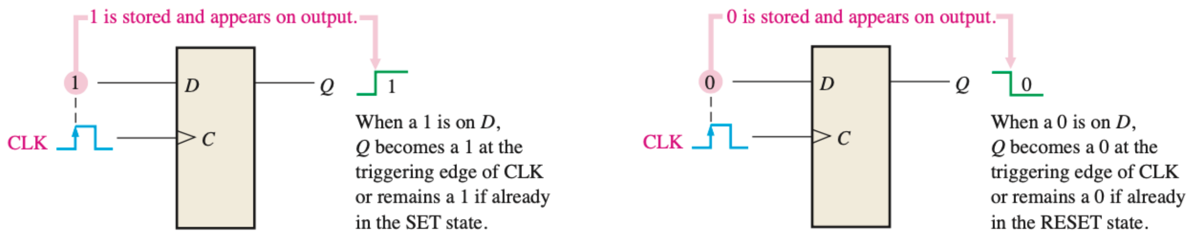

1 is stored and appears on output.
1
CLK
D
C
Q
1
When a 1 is on D, Q becomes a 1 at the triggering edge of CLK or remains a 1 if already in the SET state.
0 is stored and appears on output.
0
CLK
D
C
Q
0
When a 0 is on D, Q becomes a 0 at the triggering edge of CLK or remains a 0 if already in the RESET state.

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.

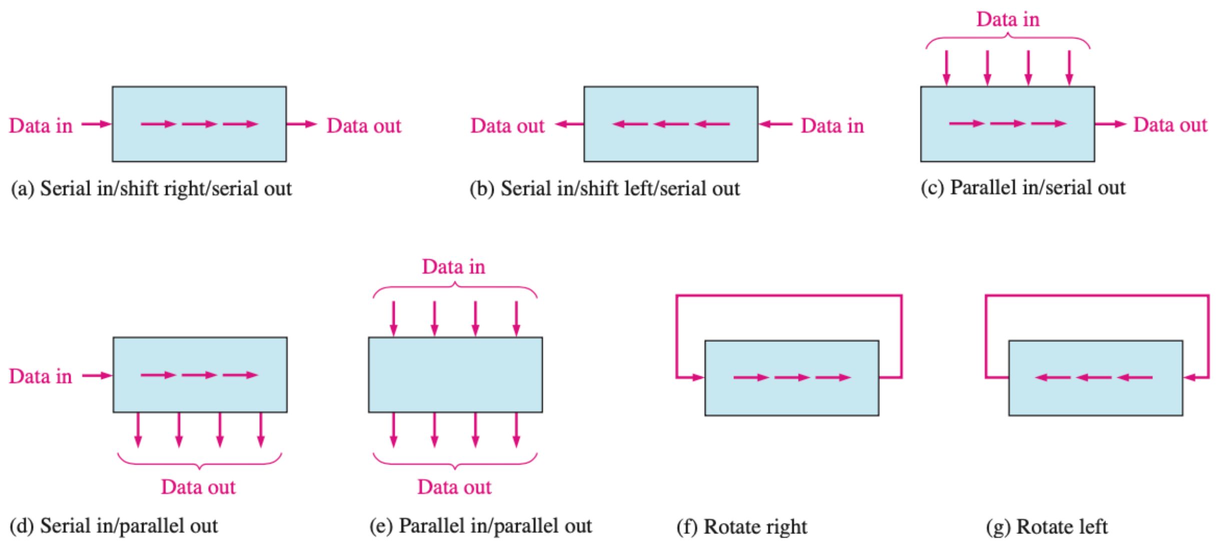

Data in
Data out
(a) Serial in/shift right/serial out
Data out
Data in
(b) Serial in/shift left/serial out
Data in
Data out
(c) Parallel in/serial out
Data in
Data out
(d) Serial in/parallel out
Data in
Data out
(e) Parallel in/parallel out
(f) Rotate right
(g) Rotate left

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

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.

Types of Shift Registers Data I/Os

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. Let's first look at the serial entry of data into a typical shift register. 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.

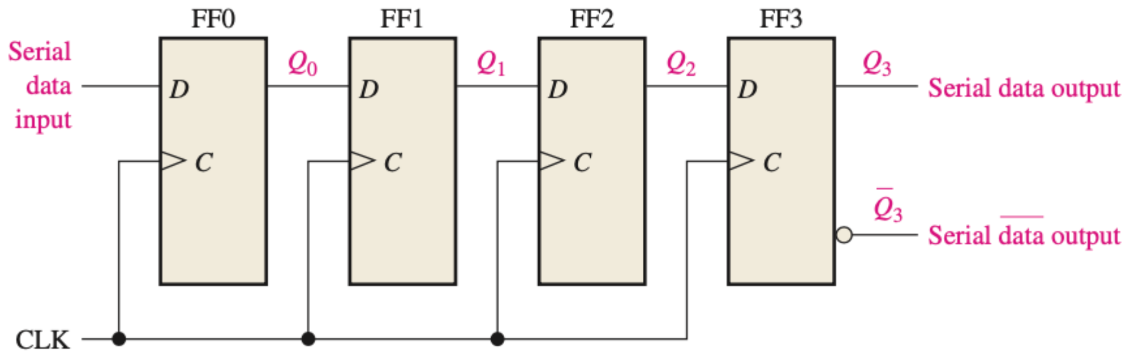


Figure 3 serial in/serial out shift register

2- 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.

Table 1 Shifting 4 bits into the shift register in figure 3

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.

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 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 2 Shifting 4 bits out of the shift register in figure 3

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 1: Show the states of the 5-bit register in Figure 4 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 $Q4Q3Q2Q1Q0 = 11010$ after five clock pulses. See Figure 4(b).

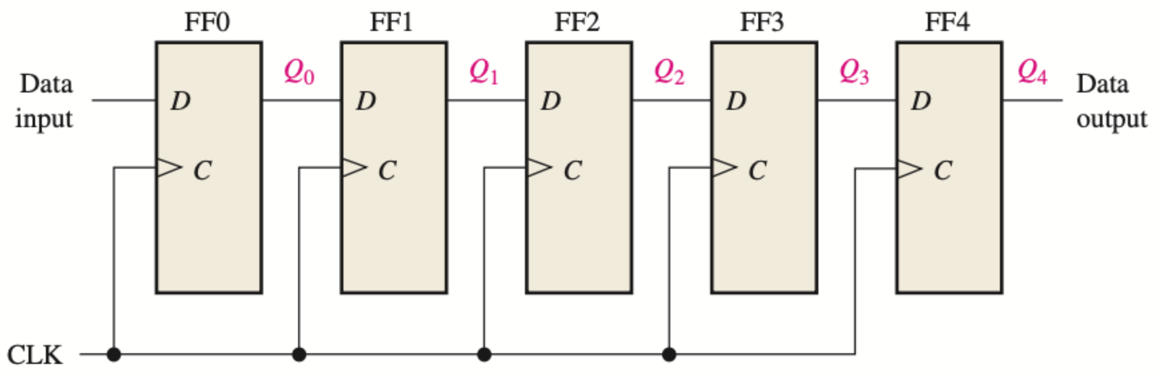


Figure 4 SISO shift register

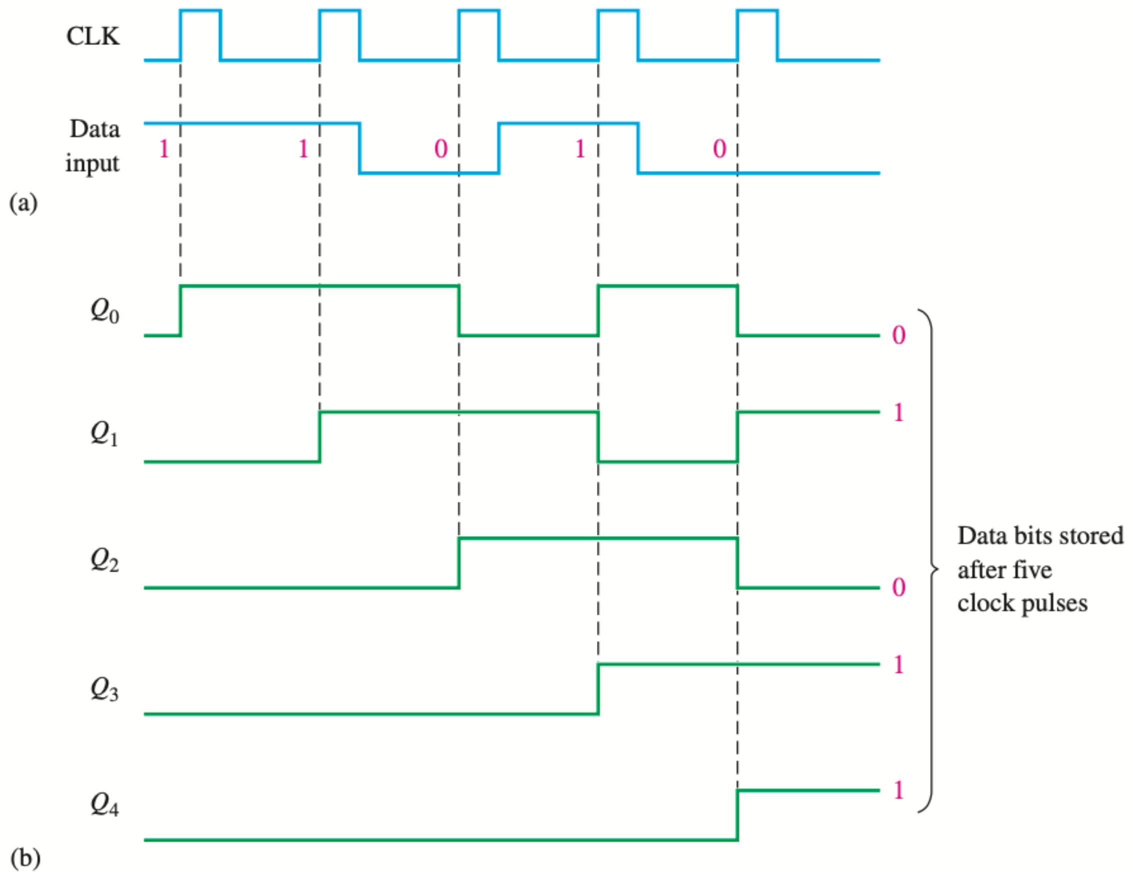


Figure 5 waveforms for SISO

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.

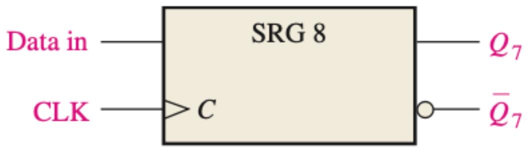


Figure 6 Logic symbol for an 8-bit serial in/serial out shift register.

3- 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 out- put 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.

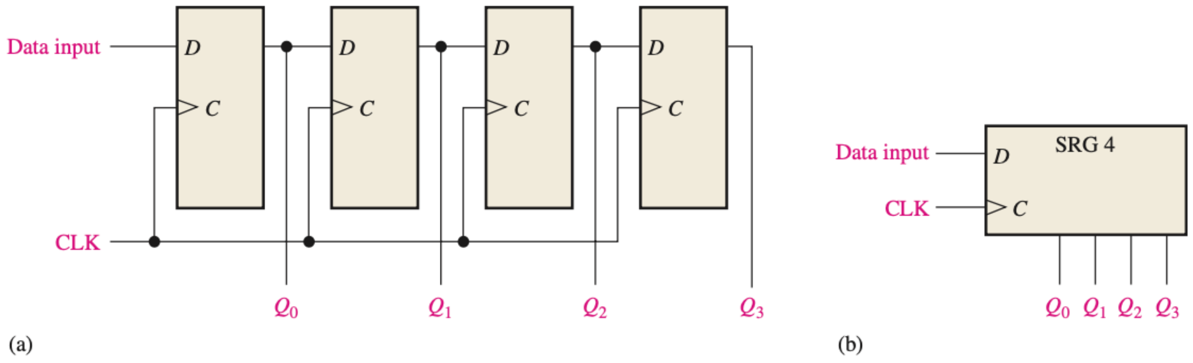
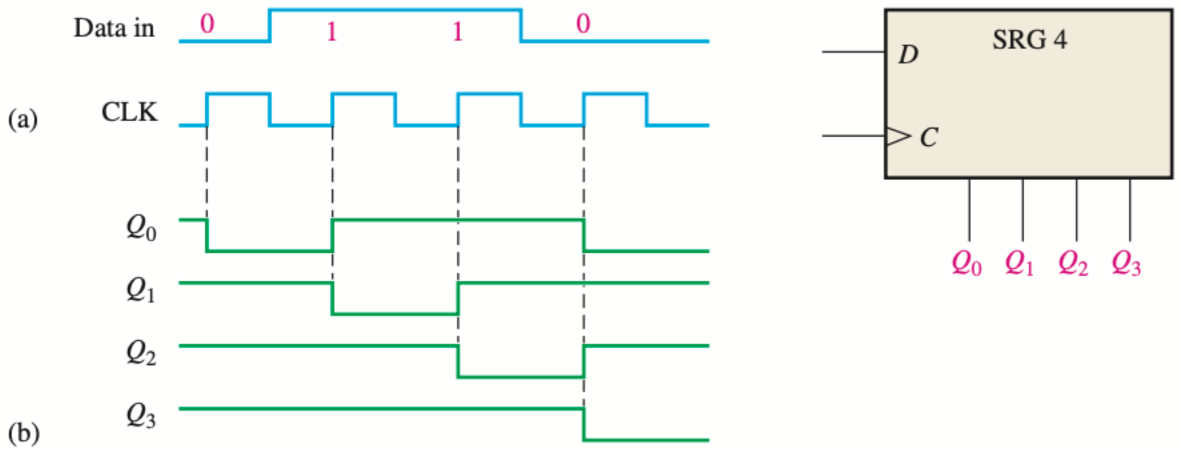


Figure 7 A serial in/parallel out shift register.

EXAMPLE 2:- 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

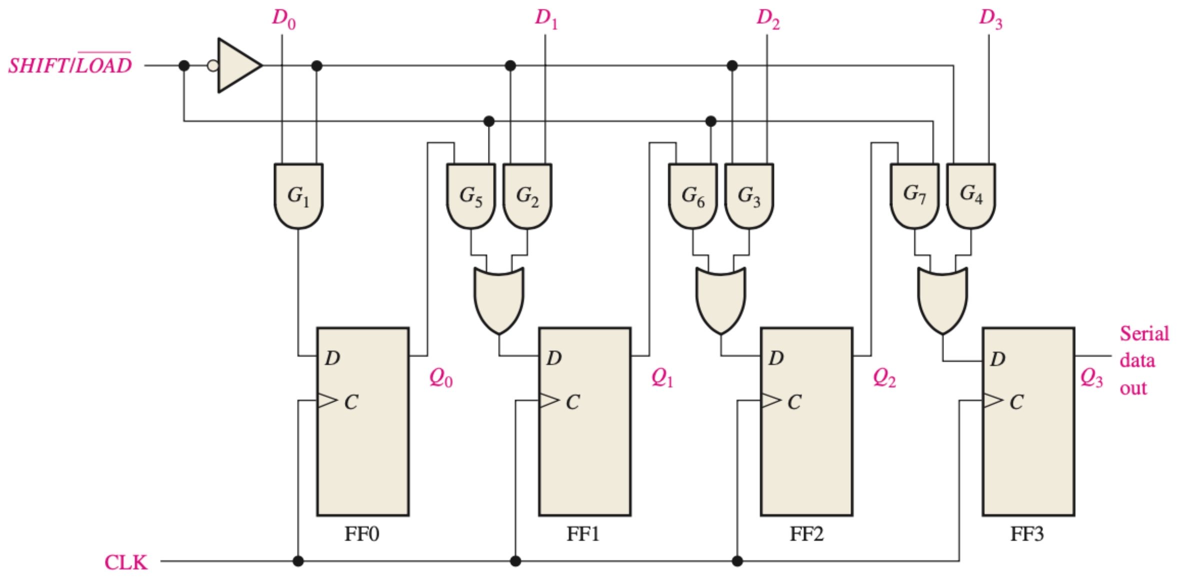
The register contains 0110 after four clock pulses.

4- Parallel In/Serial Out Shift Registers

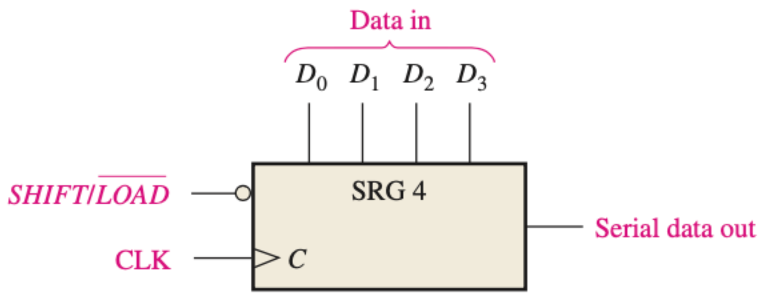
For a register with parallel data inputs, the bits are entered simultaneously into their respective stages on parallel lines rather than on a bit-by-bit basis on one line as with serial data inputs. The serial output is the same as in serial in/serial out shift registers, once the data are completely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

Figure 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*/*LOAD* input, which allows four bits of data to **load** in parallel into the register. When *SHIFT*/*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*/*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 A 4-bit parallel in/serial out shift register.

Example 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 9 (a). Refer to Figure 8 (a) for the logic diagram.

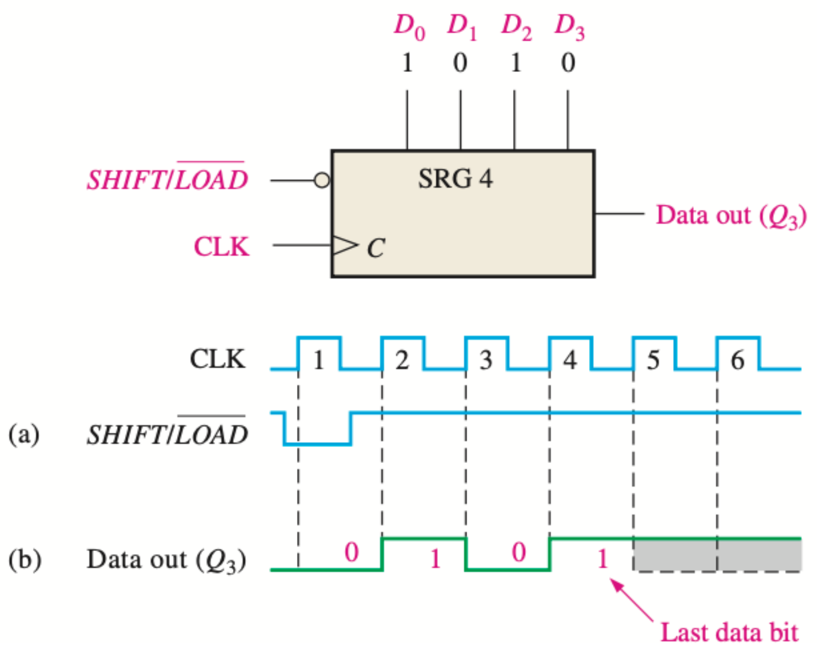


Figure 9 waveform for a 4-bit register PISO

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).

5- 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 10 shows a parallel in/parallel out shift register.

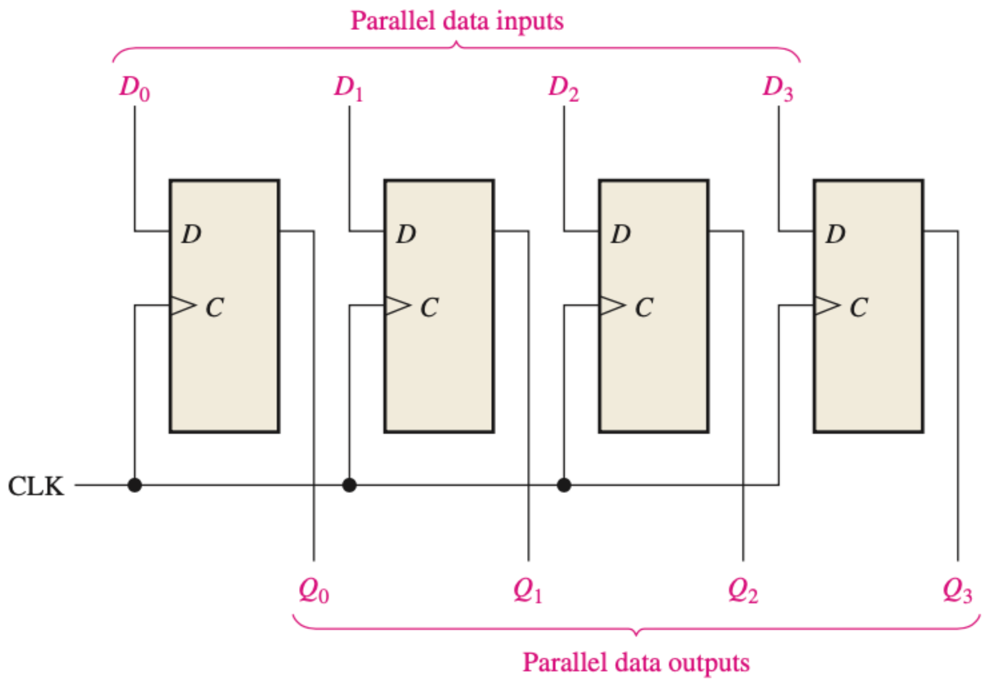


Figure 10 A parallel in/parallel out register.

6- 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.

After completing this section, you should be able to

- Explain the operation of a bidirectional shift register
- Discuss the 74HC194 4-bit bidirectional universal shift register
- Develop and analyse timing diagrams for bidirectional shift registers

A 4-bit bidirectional shift register is shown in Figure 11. A HIGH on the *RIGHT*/*LEFT* control input allows data bits inside the register to be shifted to the right, and a LOW

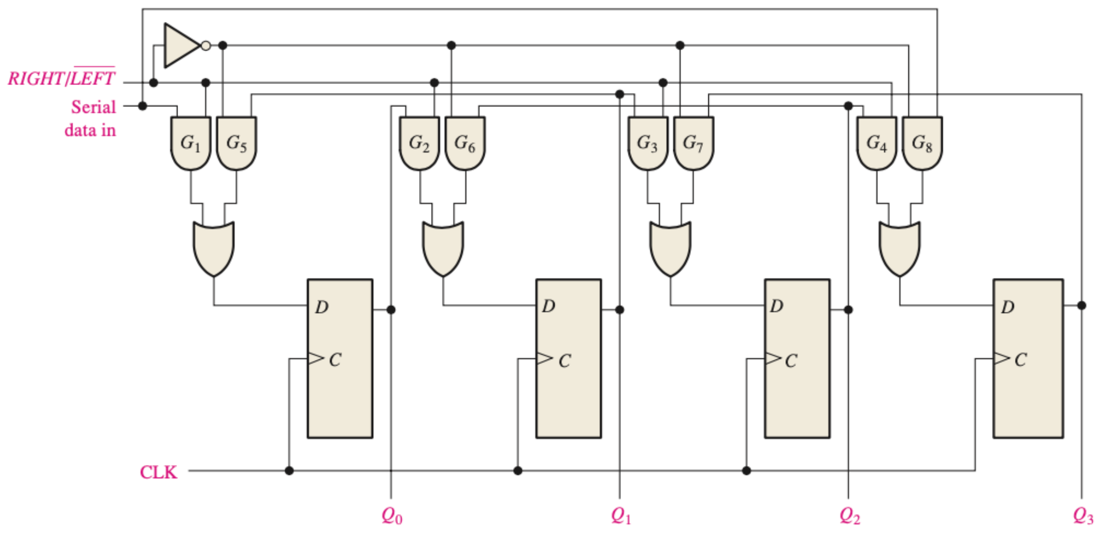


Figure 11 Four-bit bidirectional shift register

enables data bits inside the register to be shifted to the left. An examination of the gating logic will make the operation apparent. When the *RIGHT*/*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*/*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*.

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 shifts register counters, the Johnson counter and the ring counter, are introduced in this section.

After completing this section, you should be able to

- Discuss how a shift register counter differs from a basic shift register
- Explain the operation of a Johnson counter
- Specify a Johnson sequence for any number of bits
- Explain the operation of a ring counter and determine the sequence of any specific ring counter

1- 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 2*n,* where *n* is the number of stages in the counter.

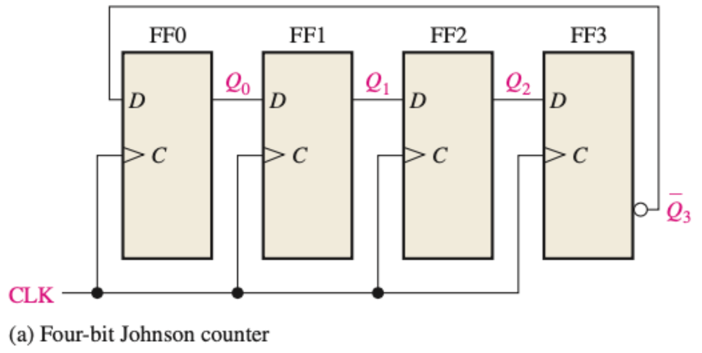
Table 3 for a 4-bit Jonson Counter

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 for a 5-bit Jonson Counter

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 implementations of the 4-stage and 5-stage Johnson counters are shown in Figure 12. 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



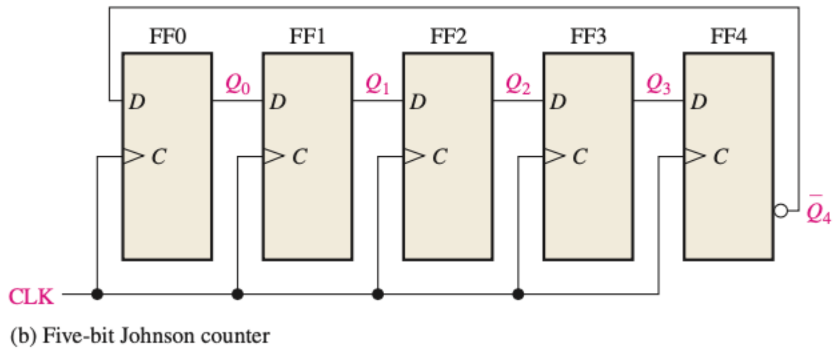


Figure 12 Four-bit and 5-bit Johnson counters.

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 8–3 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.

Diagrams of the timing operations of the 4-bit and 5-bit counters are shown in Figures 13 and 14, respectively.

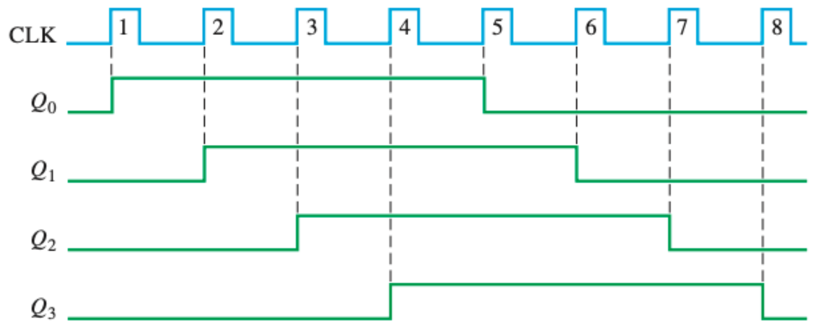


Figure 13 Timing sequence for a 4-bit Johnson counter.

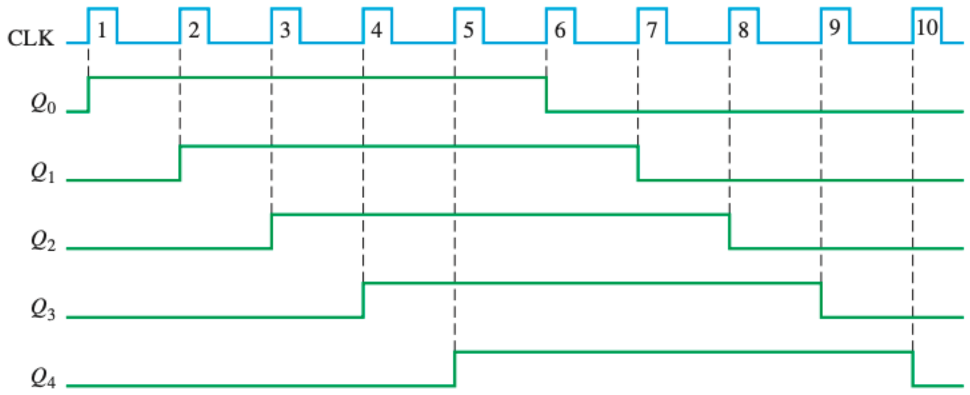


Figure 14 Timing sequence for a 5-bit Johnson counter.

2- 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 15. The sequence for this ring counter is given in Table 8–5. 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

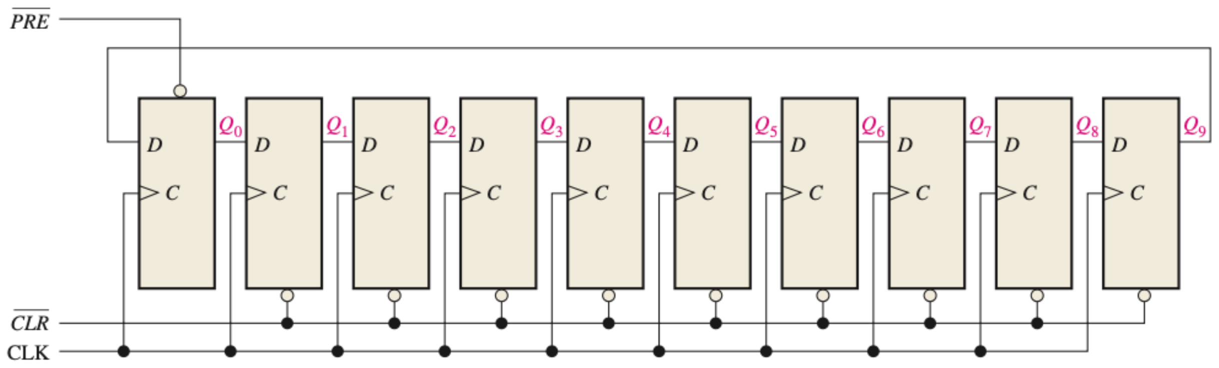


Figure 15 A 10-bit ring counter.

Table 5 Truth Table for a 10-bit ring counter

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

Example: If a 10-bit ring counter similar to Figure 14 has the initial state 1010000000, determine the waveform for each of the Q outputs.

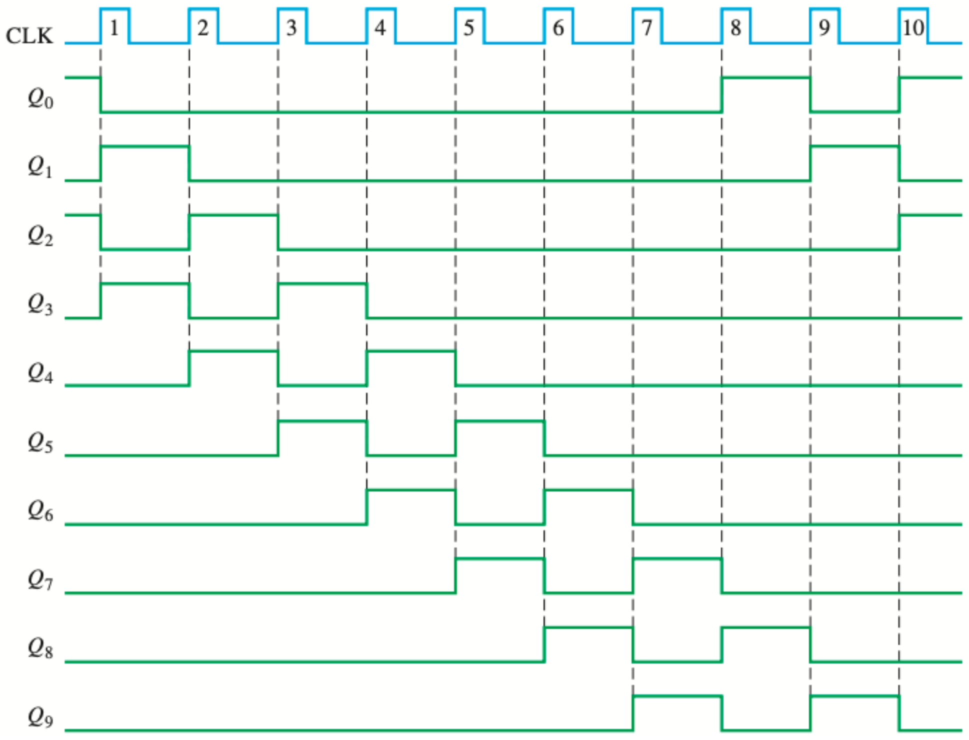


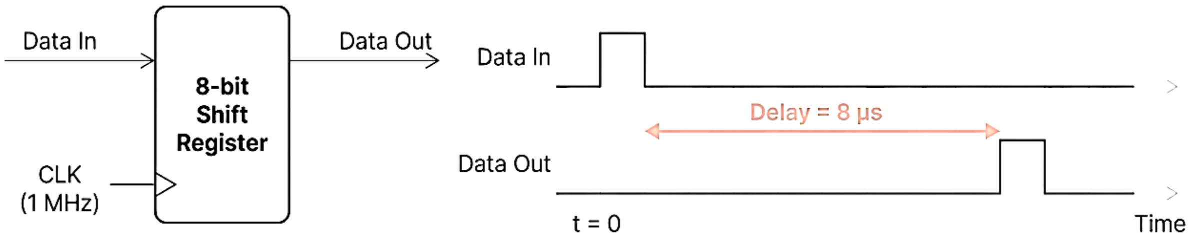
Figure 16 waveform for 10-bit ring counter

Shift Register Application:

1- Application: Creating Precise Time Delays

A serial in/serial out shift register can be used to delay a signal. A data pulse applied to the input appears at the output n clock periods later, where n is the number of stages.

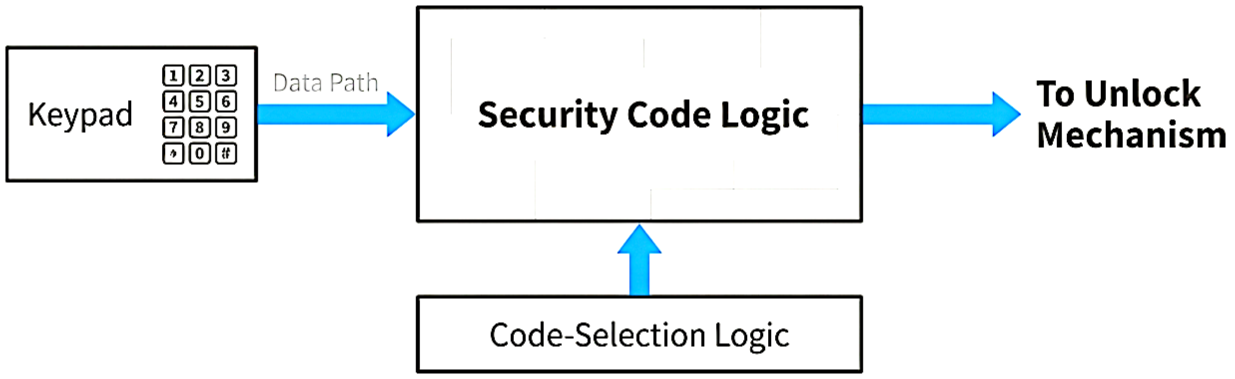
$$\text{Time Delay } (t_d) = \text{Number of Stages } (n) \times \text{Clock Period } (T_{CLK})$$



With an 8-stage register and a 1 MHz clock (1 μs period), the output is delayed by 8 × 1 μs = 8 μs.

2- Applied Logic: Assembling a Security System

We will now build a security system that provides coded access to a secured area. A4-digit code is stored, and a user must enter the correct code on a keypad to gain access.



This system uses multiple shift registers for storing the entered code, sequencing through the stored code digits, and tracking the state of the entry process. It is a perfect demonstration of registers working in concert.

Inside the Security Logic:

1. Initialisation: Pressing '#' loads a preset pattern (00010000) into Register C.

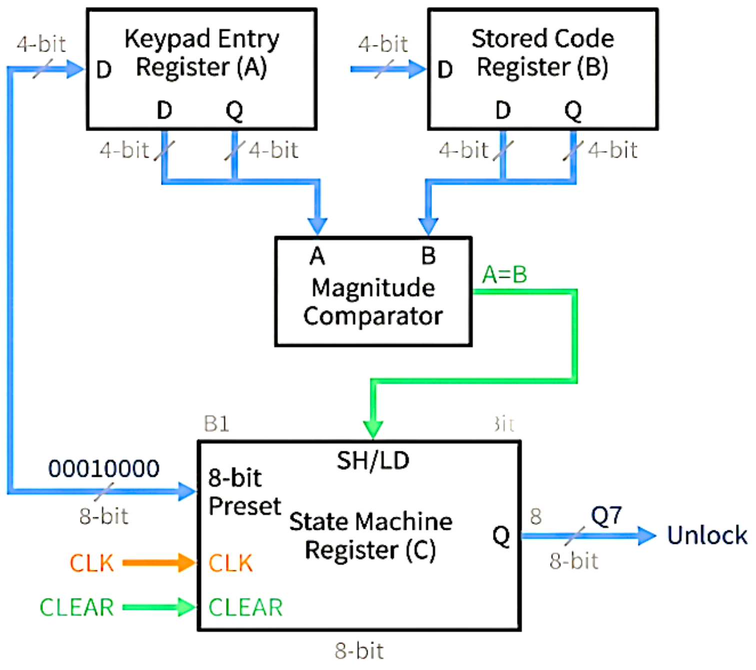
2. Digit Entry: Each keypad press loads the entered BCD digit into Register A and the corresponding stored digit into Register B.

3. Comparison: The Comparator checks if Register A equals Register B.

4. Correct Digit: If they match, the comparator output goes HIGH, putting Register Cin SHIFT mode. The '1' in Register C shifts one position to the right.

5. Incorrect Digit: If they don't match, the comparator output goes LOW, putting Register Cin LOAD mode, re-initialising it to '00010000'.

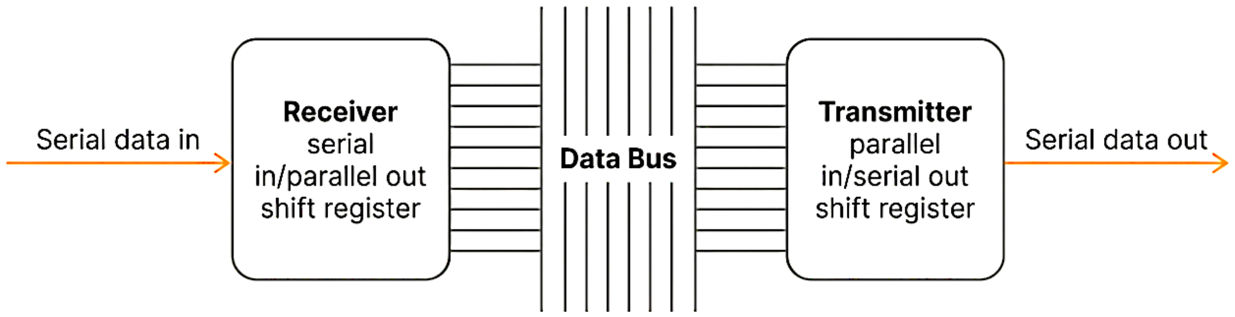
6. Success: After the fourth correct digit, the '1' reaches the output of Register C, activating the unlock mechanism.



3- Application: Interfacing with the Serial World

Computers process data in parallel, but it's often transmitted serially (e.g., via USB) to reduce wiring. ASerial-to-Parallel Converter is essential for this interface. At its core is a SIPO shift register that captures the incoming serial bits.

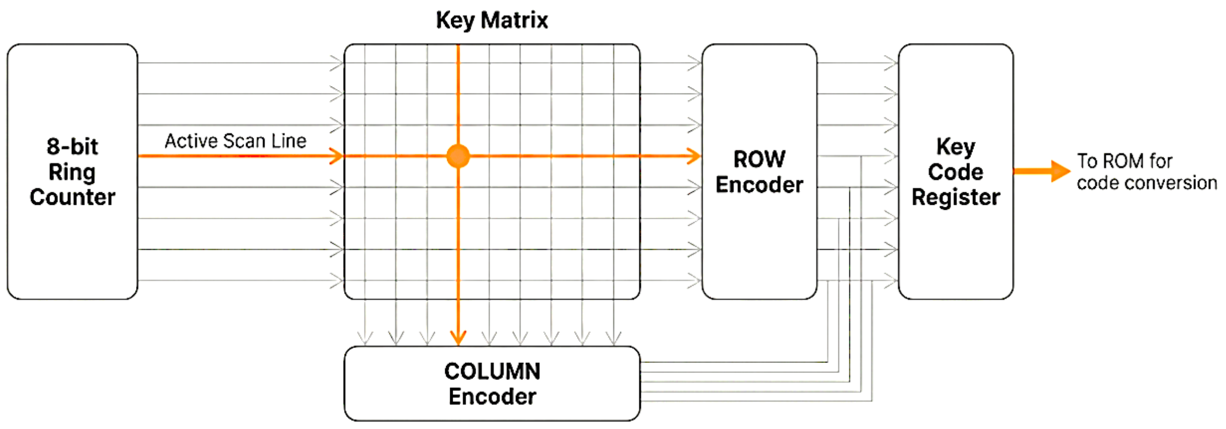
A Universal Asynchronous Receiver/Transmitter (UART) is a dedicated IC that performs both serial-to-parallel and parallel-to-serial conversions.


Serial data in
Receiver
serial in/parallel out shift register
Data Bus
Transmitter
parallel in/serial out shift register
Serial data out

4- Application: The Keyboard Encoder

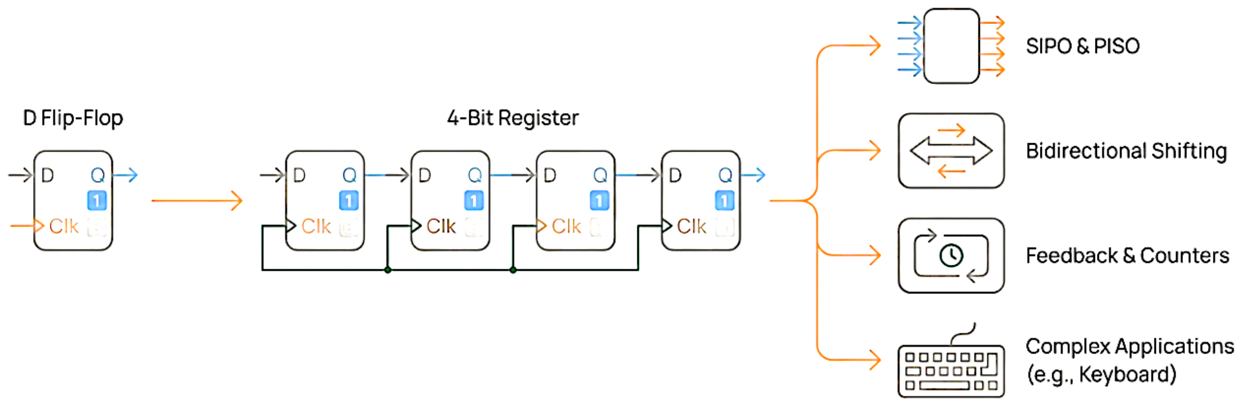
Shift registers are key components in keyboard encoders. Aring counter is used to sequentially "scan" the rows of the key matrix by applying a LOW signal to one row at a time. When a key is

pressed, it connects a row to a column. Encoders convert the active row and column into a unique binary code, which is then stored ni a parallel-in/parallel-out register (the "key code register").



From a Single Flip-Flop to a Complete System

The shift register is a testament to emergent capability in digital logic. A simple chain of 1-bit memory stages, when configured in different ways, becomes an indispensable tool for data manipulation and system control.



Homework

1. How many states are there in an 8-bit Johnson counter sequence?
2. Write the sequence of states for a 3-bit Johnson counter starting with 000.
3. How many clock pulses are required to enter a byte of data serially into an 8-bit shift register?
4. Explain the function of the *SHIFT*/*LOAD* input.