SHIFT REGISTERs

Digital Logic Design (CC131)

BSCS 2nd Semester Spring-2026

By

Syed Zulfiqar Ali Shah,

Associate Professor (CS) / HoD

Govt. College of Management Sciences, Kohat

www.linkedin.com/in/syedzulfiqaralishah

REGISTER

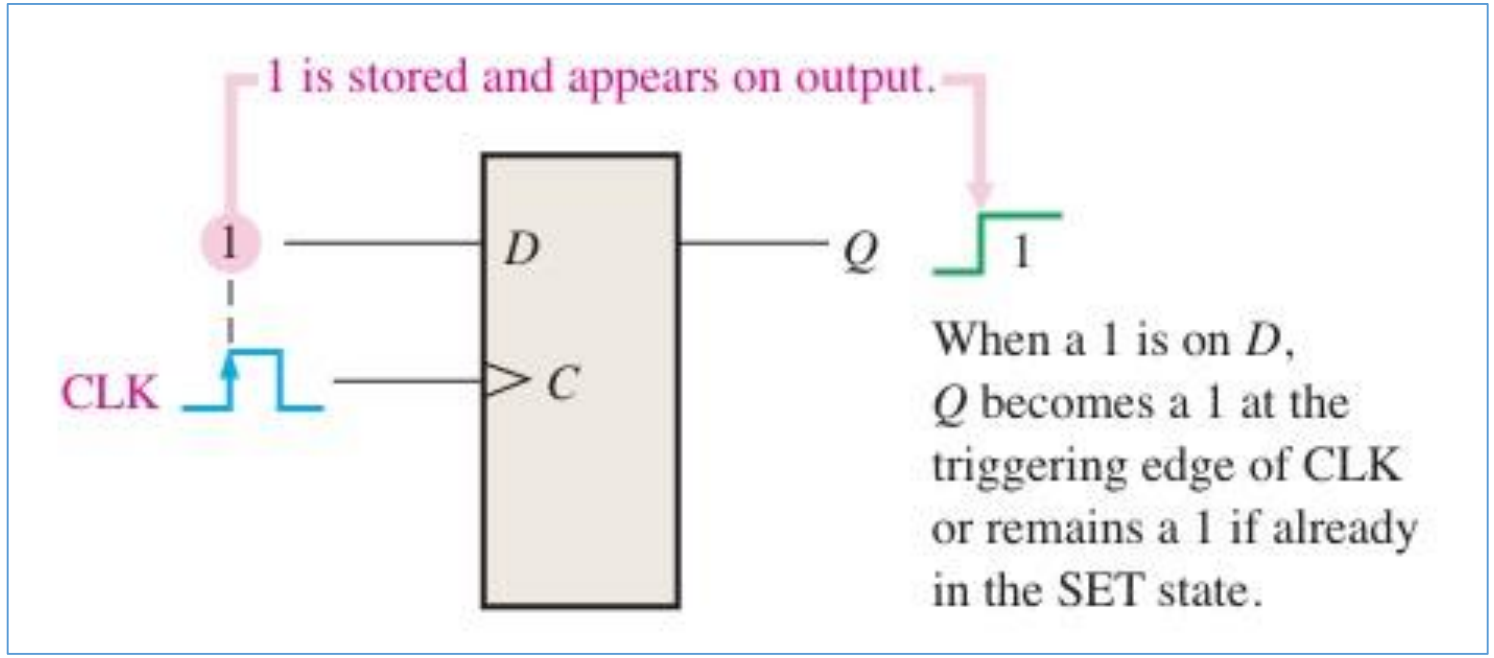
- A Register and a Shift Register are both sequential logic circuits made from flip-flops, but they serve different purposes.

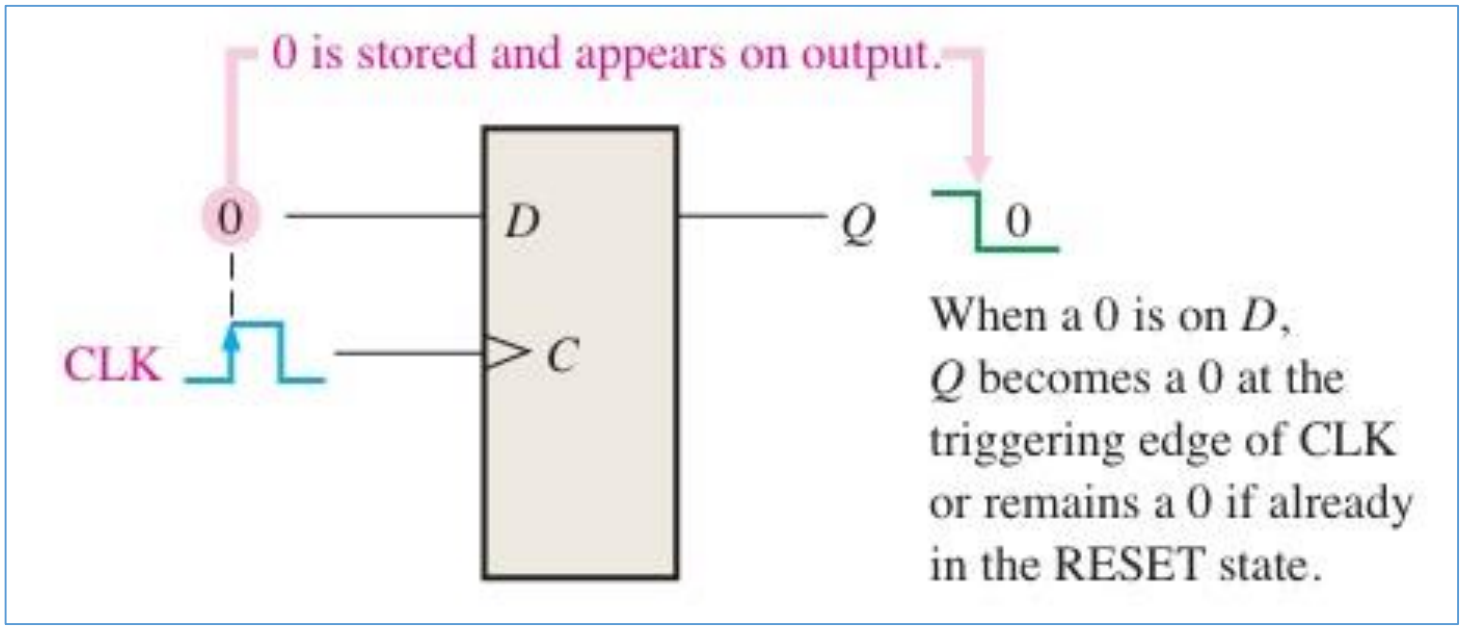
- A register is a group of flip-flops used to store binary information.
  - Each flip-flop stores **1 bit**.
  - An n-bit register contains n flip-flops.
  - All flip-flops usually share the same clock.

For example:

- 4 flip-flops → 4-bit register
- Can store values like:

1010, 1111, 0001


1 is stored and appears on output.
1
D
Q
CLK
C
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
D
Q
CLK
C
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.

REGISTER

Main Purpose

- Temporary data storage
- Holding binary information in digital systems

Example

A 4-bit register may store:

$$Q_3\,Q_2\,Q_1\,Q_0 \;=\; 1101$$

and keep this value unchanged until a new value is loaded.

SHIFT REGISTER

- A **shift register** is a special type of register in which the stored data can be **shifted** from one flip-flop to another on each clock pulse.
- The bits move:
    - left or
    - right
    - one position at a time.

SHIFT REGISTER

Example of Right Shift

Initial contents:	1011
After one clock pulse:	0101
After another:	0010

Bits are “shifting” through the register.

SHIFT REGISTER

Why is it called a Shift Register?

Because besides storing data, it can also:

- move data serially,
- transfer bits,
- convert serial ↔ parallel data.

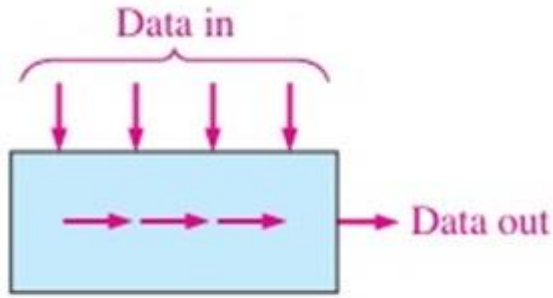
TYPES OF SHIFT REGISTERs

1. **SISO** – Serial In Serial Out
2. **SIPO** – Serial In Parallel Out
3. **PISO** – Parallel In Serial Out
4. **PIPO** – Parallel In Parallel Out

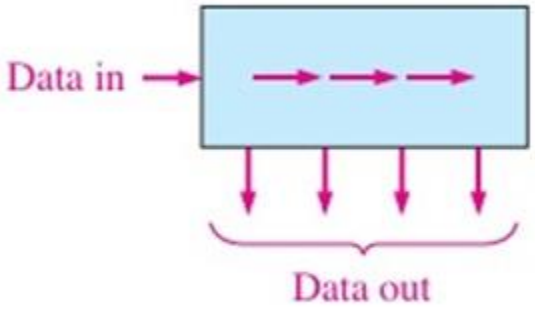
These describe how data enters and leaves the register.

(a) Serial in/shift right/serial out

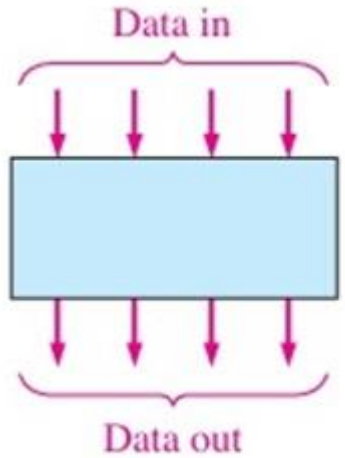
(b) Serial in/shift left/serial out



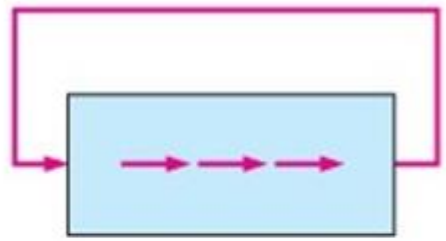
(c) Parallel in/serial out



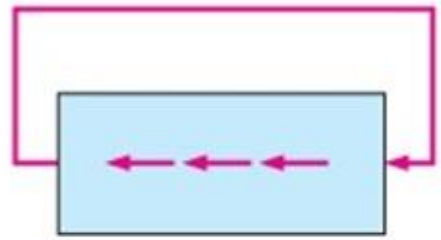
(d) Serial in/parallel out



(e) Parallel in/parallel out



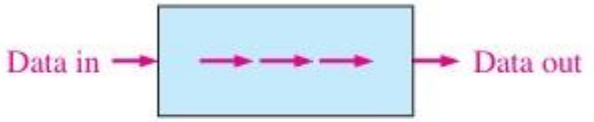
(f) Rotate right



(g) Rotate left

1. Serial In Serial Out

- The SISO shift register accepts data serially.

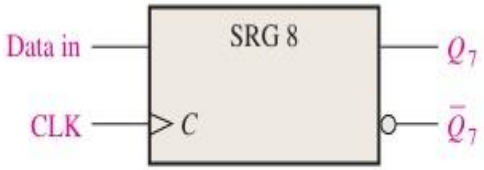


(a) Serial in/shift right/serial out

- It has a Single line and only one bit enters at a time.

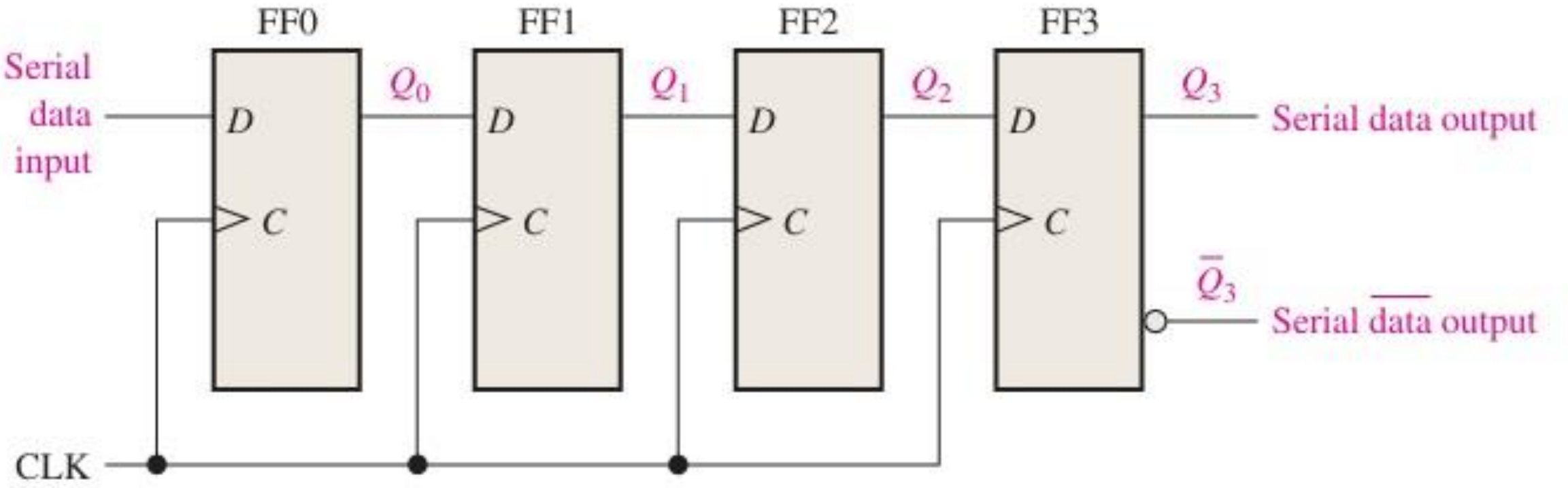
- Stored data is produced on Output in serial form, means only one bit gets out at a time.

(b) Serial in/shift left/serial out



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

1. Serial In Serial Out

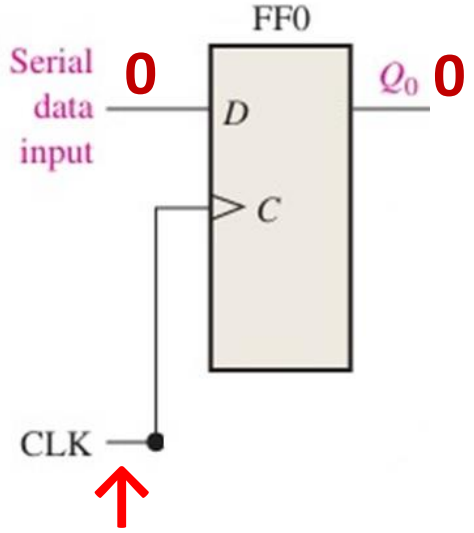


4-bit Serial in / Serial out shift register

Serial Data Input/Entry

1 0 1 0

- Data = 1 0 1 0
- Data entry starts from the LSB
- The register is initially clear



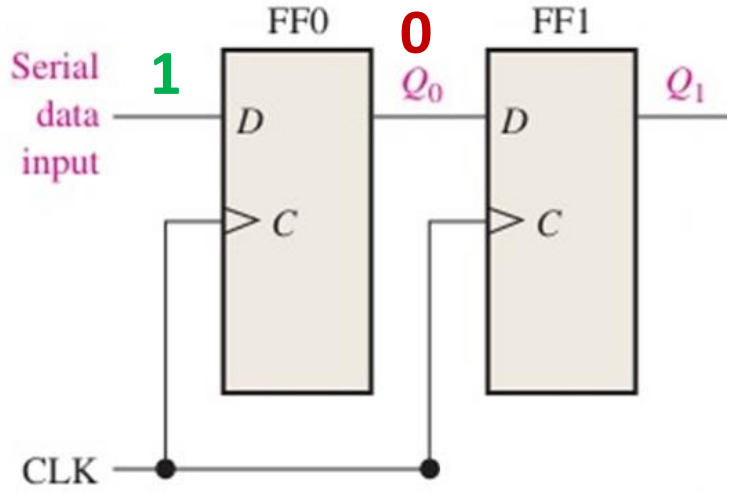
1. First the LSB 0 is put onto the data input line, making D = 0 for FF0.

2. Then the first clock pulse is applied, FF0 is RESET, storing the 0.

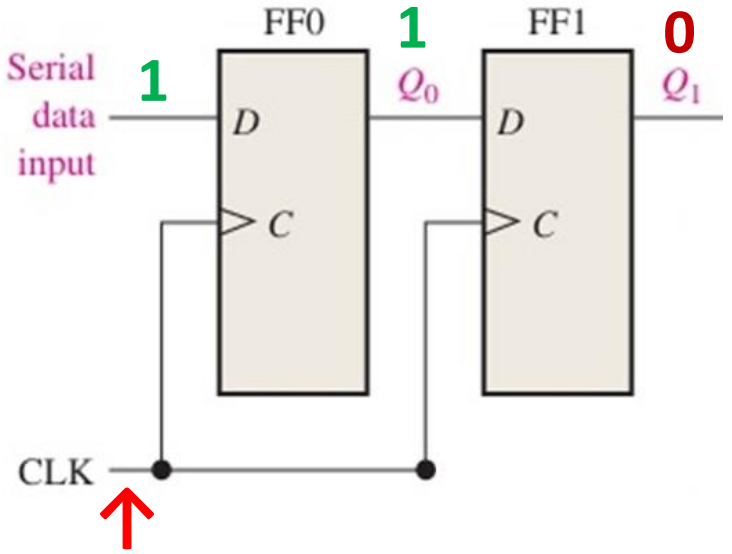
Serial Data Input/Entry

1 0 1

3. Next the second bit 1 is applied to the data input line, making D=1 for FF0 and D=0 for FF1

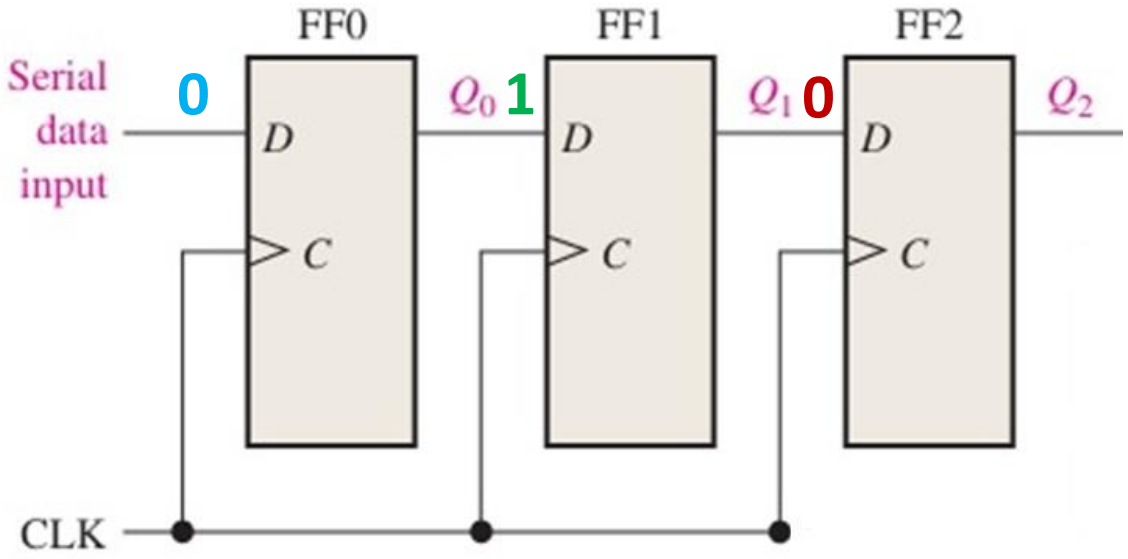


4. Second clock pulse applies, the 1 on data input D is shifted into FF0, FF0 is SET.
0 that was on FF0 is shifted into FF1

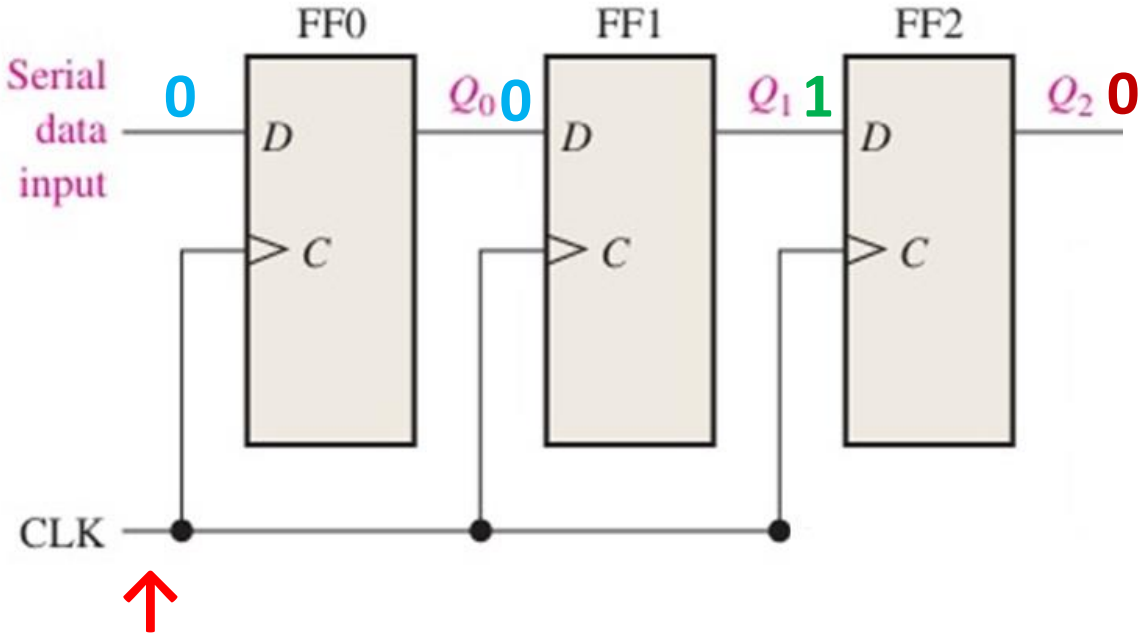


Serial Data Input/Entry 1 0

5. Next the third bit, a 0 is applied to the data input line,
making D=0 for FF0 and
D=1 for FF1 and
D=0 for FF2

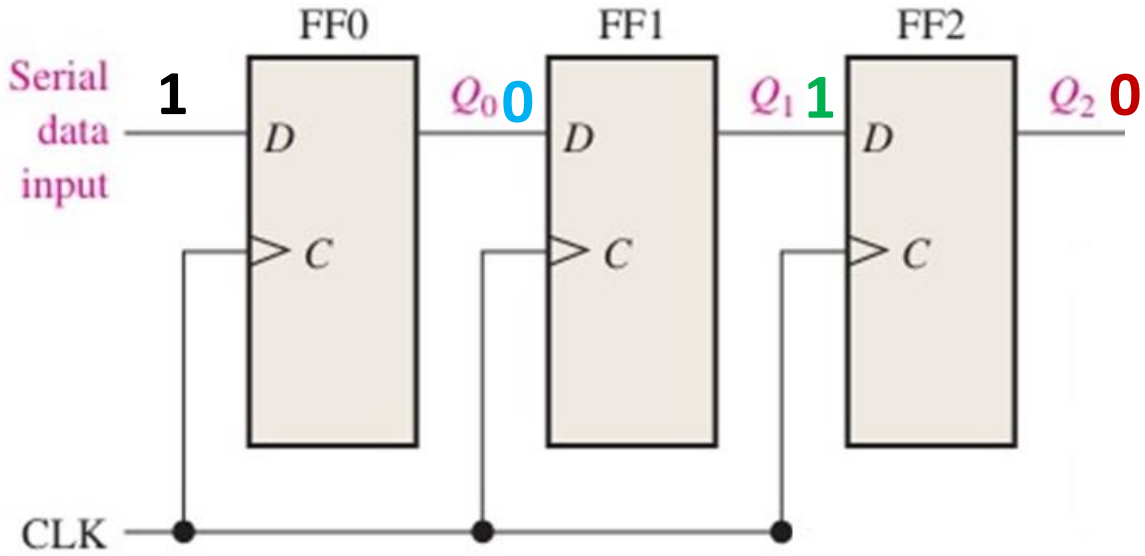


6. Third clock pulse applies which makes the
FF2 (Q_2) = 0,
FF1 (Q_1) = 1, and
FF0 (Q_0) = 0

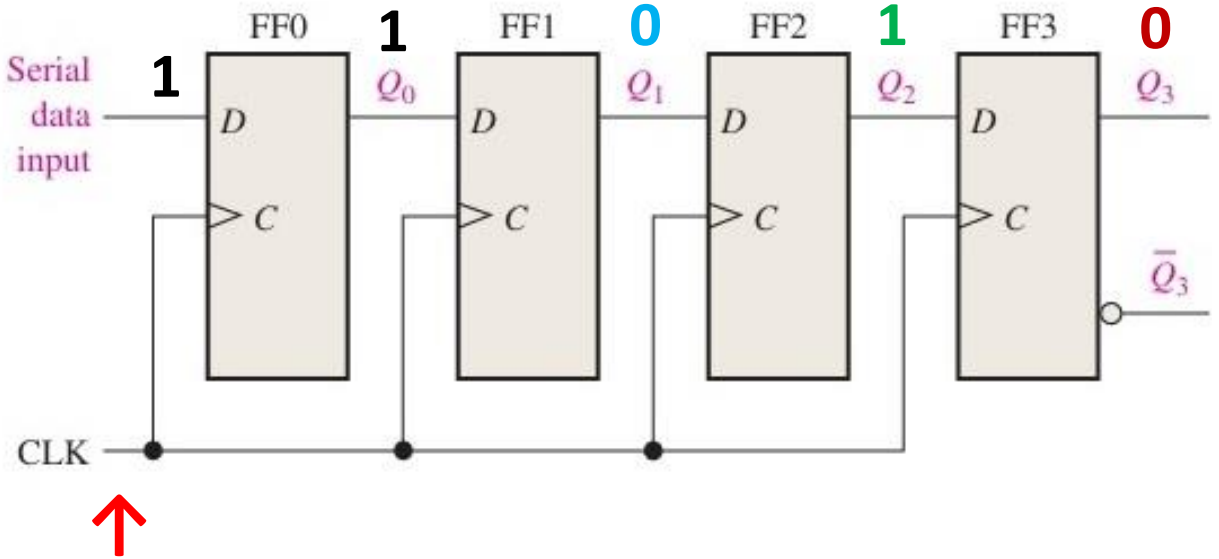


Serial Data Input/Entry

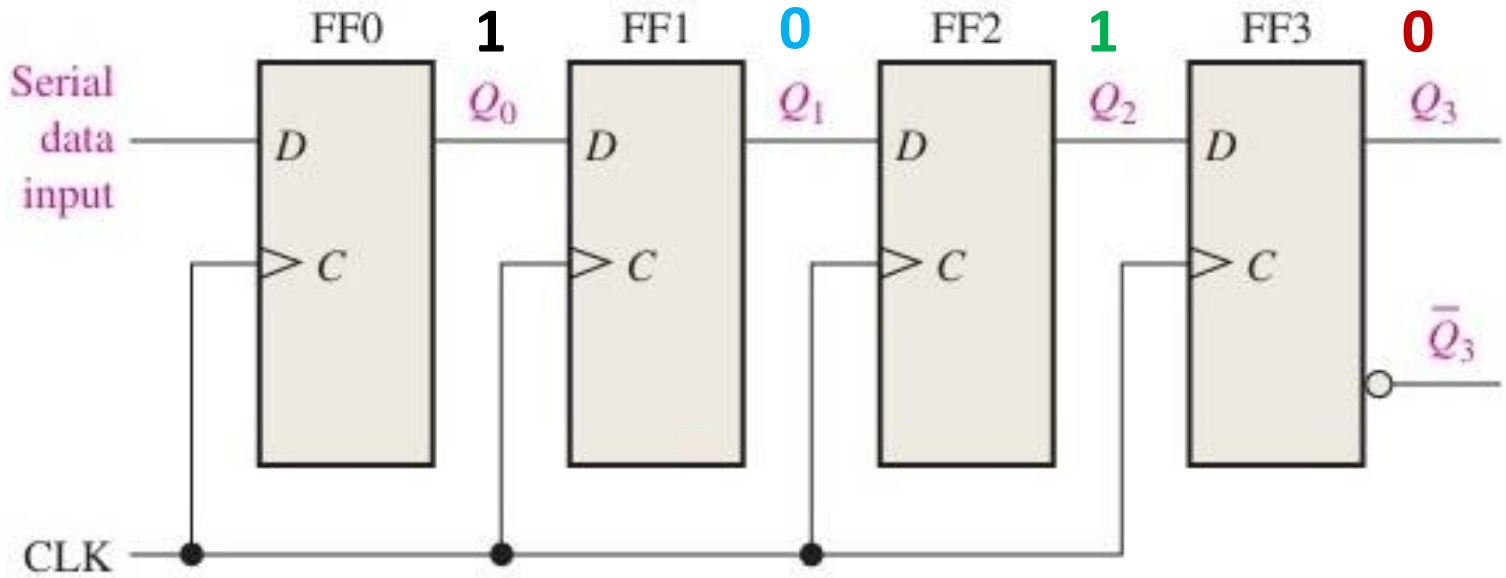
7. Next the Fourth bit, a 1 is applied to the data input line, making D=1 for FF0 ,
   D=0 for FF1 ,
   D=1 for FF2 and
   D = 0 for FF3



8. Fourth clock pulse applies which makes the
   FF3 (Q_3) = 0,
   FF2 (Q_2) = 1,
   FF1 (Q_1) = 0, and
   FF0 (Q_0) = 1



Serial Data Input/Entry



- All 4 bits are stored in the register.
- These bits will remain there until the flop-flop is receiving power.
- And whatever input is applied on D, it has no effect on the output if clock edges are not occurring.

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**

Note:

If the flip-flops were ideal and had Zero delay, then signal would have raced through all the four stages instantly during a single clock pulse and they would have all ended up with the same value.

But in real world, the shift register works correctly because of physical properties called “Hold time” and “Propagation Delay”.

Hold Time:

The input data on D must remain steady for a short period after the clock edge to ensure it is captured.

Propagation Delay:

The output of first flip-flop (FF0) must take longer to change than the Hold Time of FF1.

For a shift register to function correctly, two things must happen in a specific order during the nanoseconds that a clock pulse rises:

1. **Sampling:**
   Every flip-flop “looks” at the data present at its D input the exact moment the clock edge rises.

2. **Updating:**
   After a tiny delay (propagation delay), the flip-flop updates its Q output to match what it just saw.

The Sequence of Events

- Because of propagation delay, the change at the output of FF0 doesn't happen instantly.
- By the time FF0 changes its output Q_0 , the clock edge has already passed for FF1.
- **At T=0 (Clock Edge):**

  FF1 samples the "old" value that was sitting at Q_0.

- **At T=5 nanoseconds (after the edge):**

  FF0 finally updates its output Q_0 to the new bit.

- **Result:**

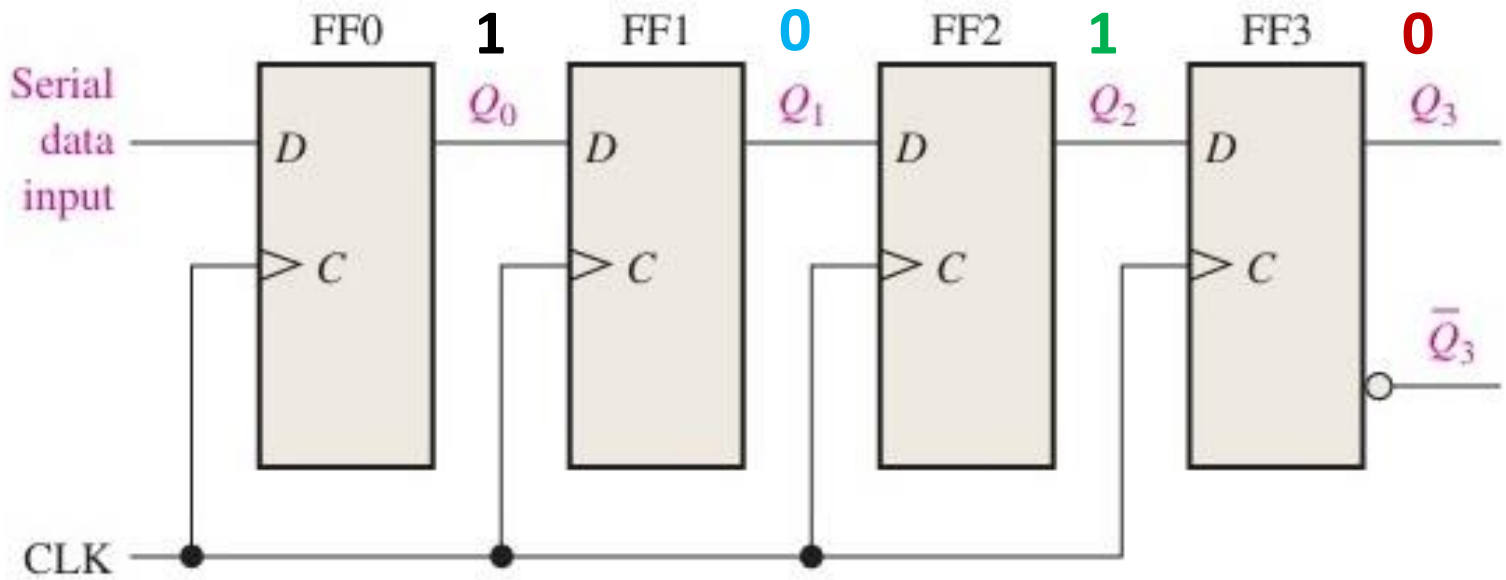
  FF1 has already captured the old data and "closed its door" before the new data from FF0 arrived.

To sum up;

The *data signal* is always “one step behind” the clock signal.

Because data signal has to physically travel through the internal transistors of the flip-flop.

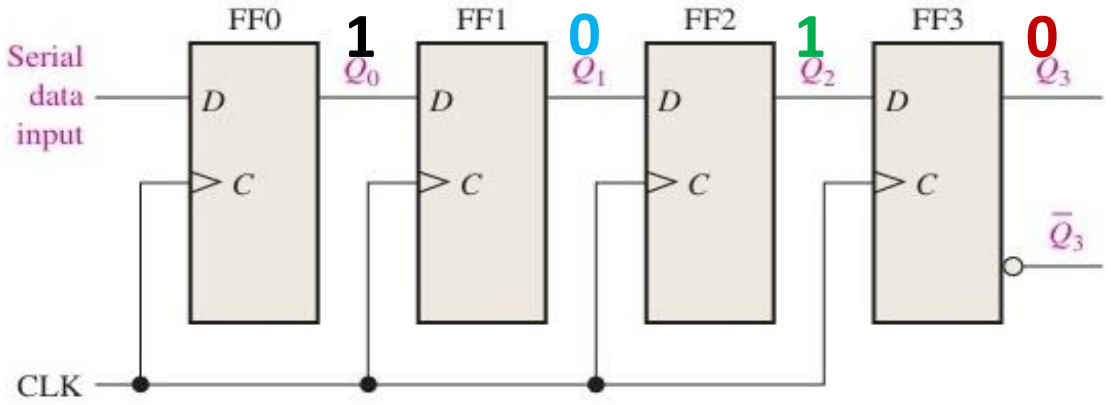
Serial Data Output



- In order to get the data out of the register, the bits are shifted out serially to the output Q_3.

- Currently (after clock pulse-4), the LSB 0 is appearing on output Q_3.

Serial Data Output



- When clock pulse-5 is applied, the second bit 1 appears on output Q_3.
- When clock pulse-6 is applied, third bit 0 appears on output Q_3.
- When clock pulse-7 is applied, last bit 1 appears on output Q_3.
- When clock pulse-8 is applied, the bit on the output Q_3 shifts out.

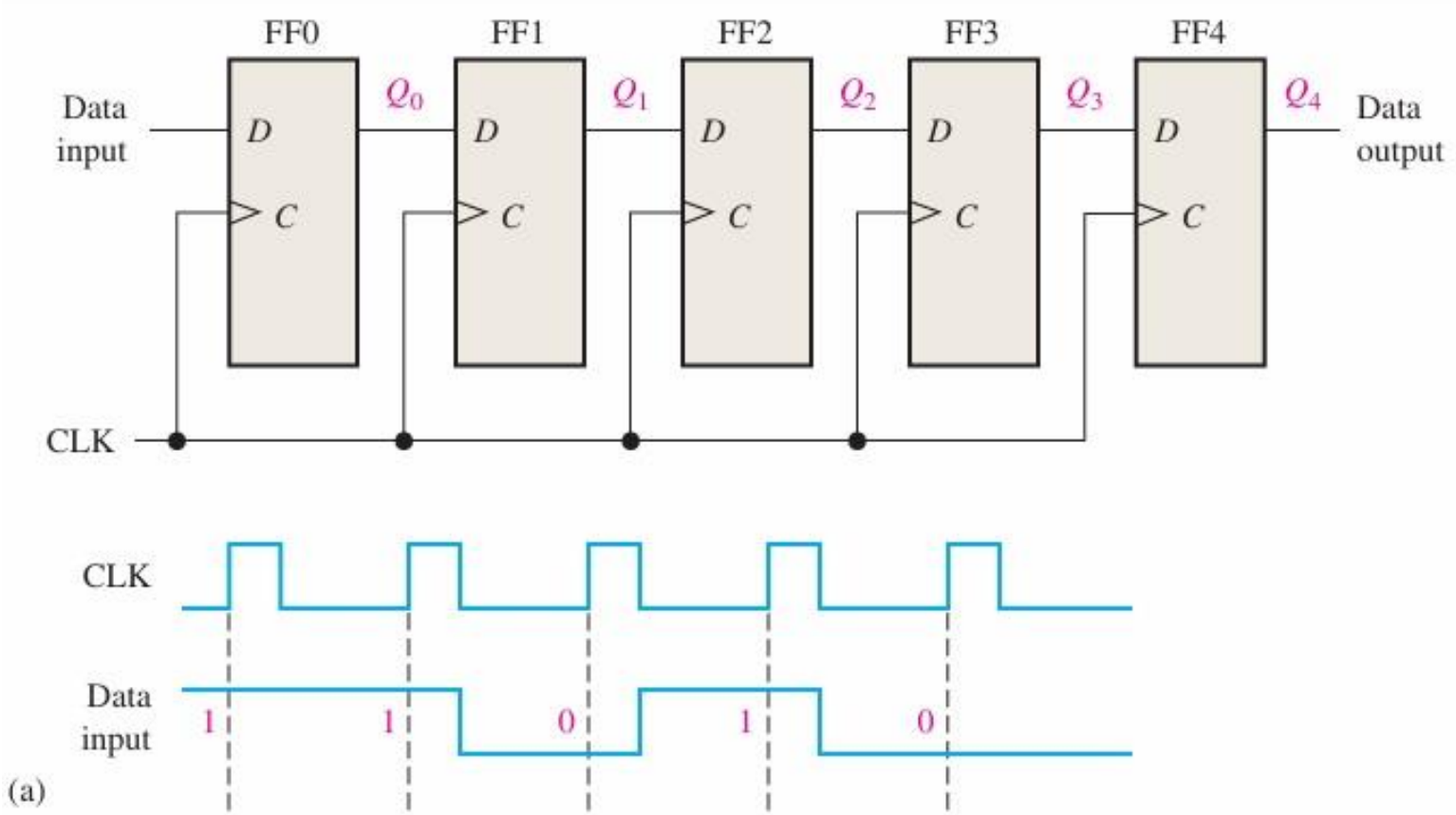
TABLE 8–2

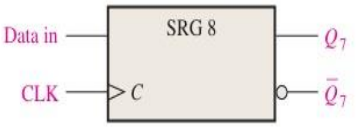
Shifting a 4-bit code out of the shift register in Figure 8–3. Data bits are indicated by a beige screen.

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 8–4(a) for the specified data input and clock waveforms. Assume that the register is initially cleared (all 0s).

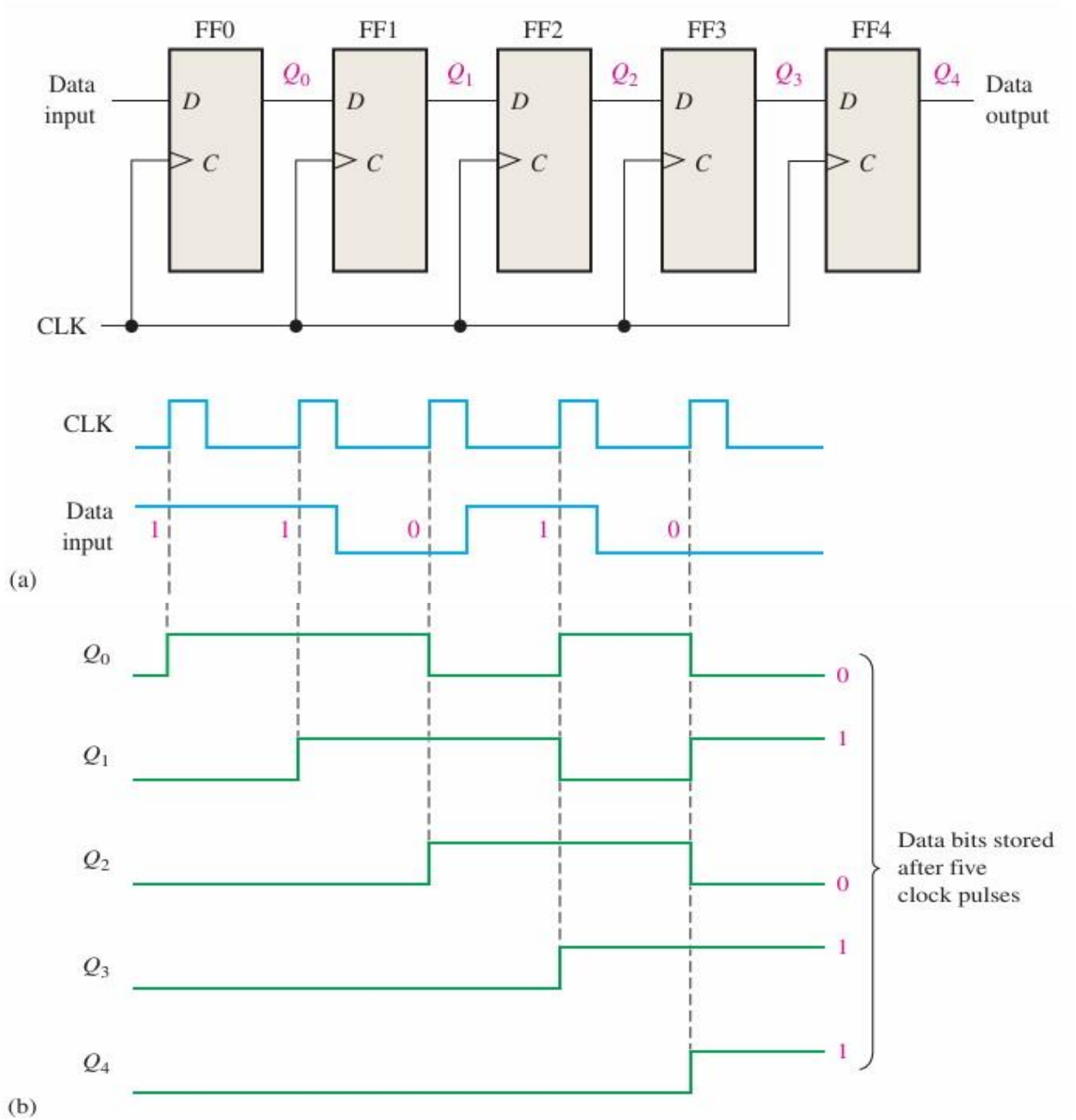




Logic symbol for an 8-bit serial in/serial out shift register.

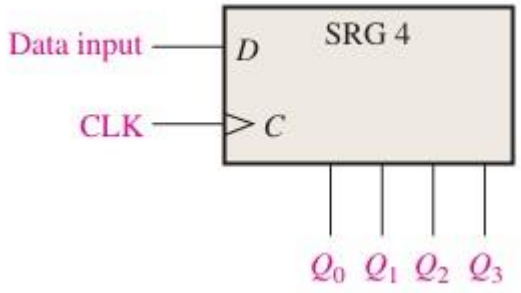
1. Serial In Serial Out

Solution

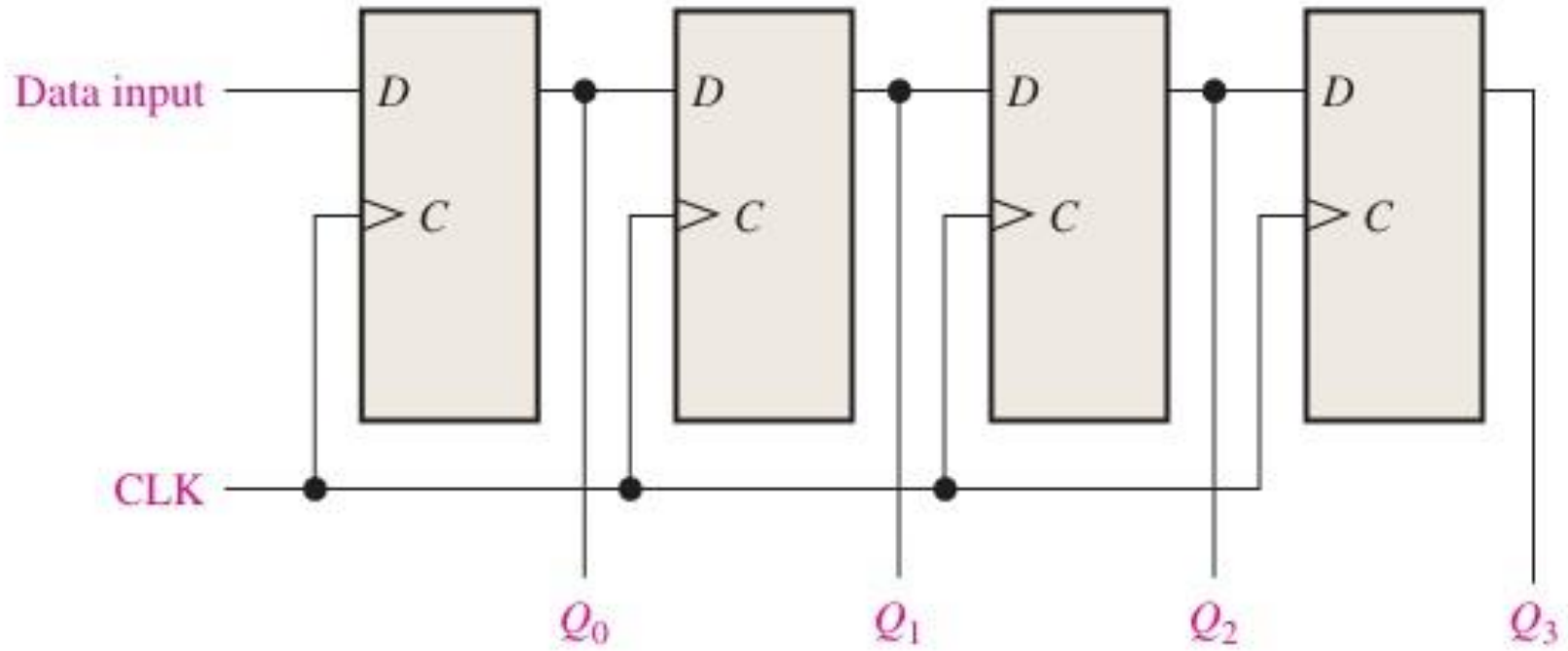


2. Serial In Parallel Out

- This shift register has a single data input line but multiple data output lines.
- Data entry in this type of shift register is similar to that of SISO shift register.
- At any instant of time, the data of each flip-flop (stage) is available simultaneously.



2. Serial In Parallel Out

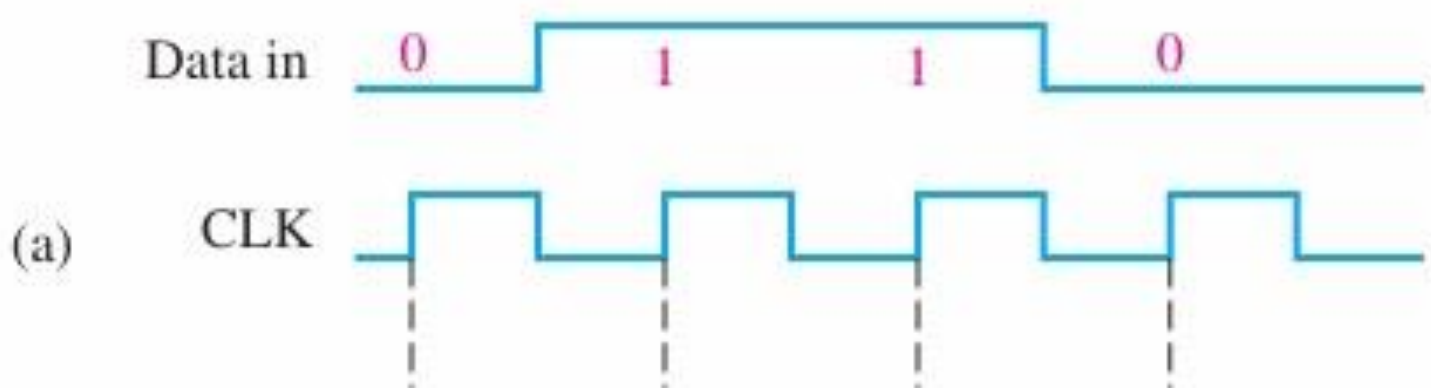


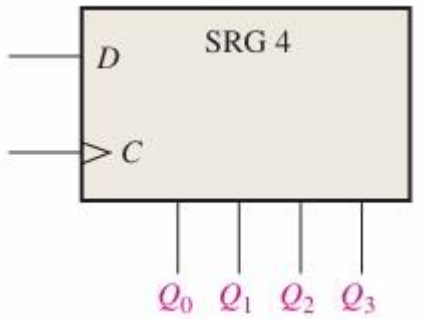
Logic Diagram

2. Serial In Parallel Out

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.

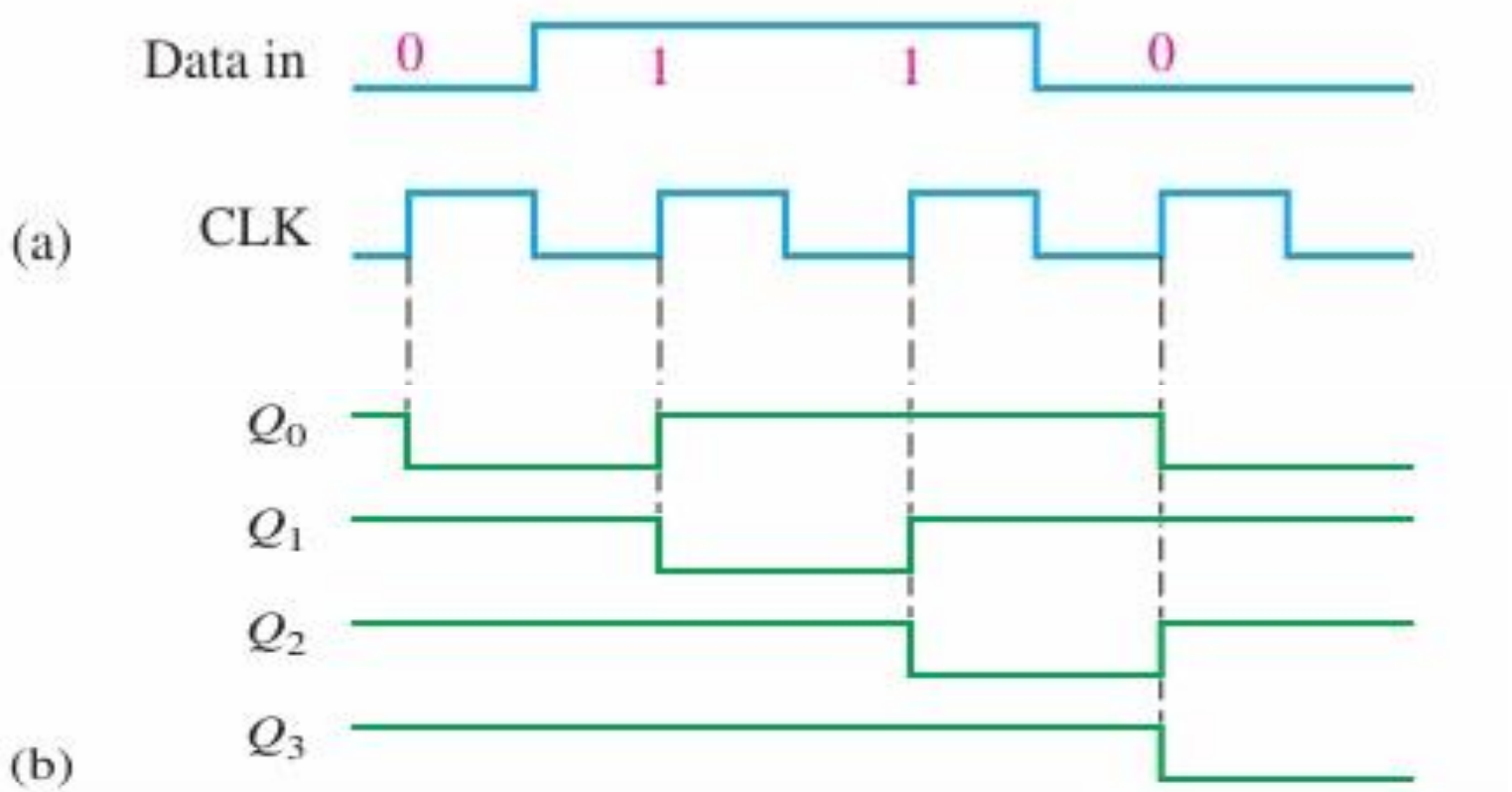




2. Serial In Parallel Out

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.



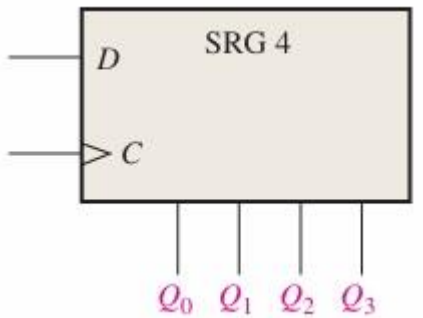
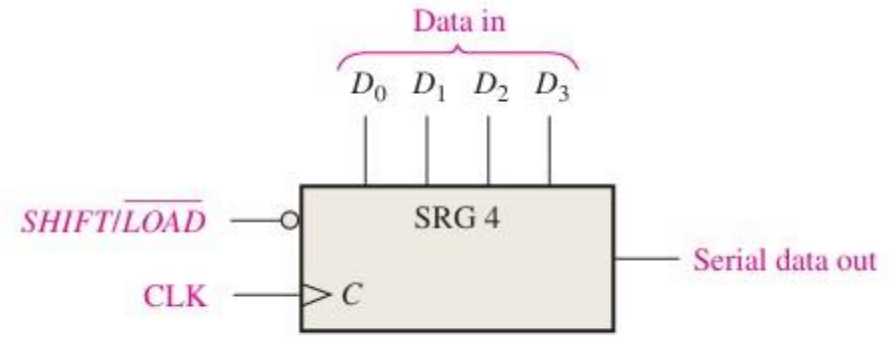


FIGURE 8–7

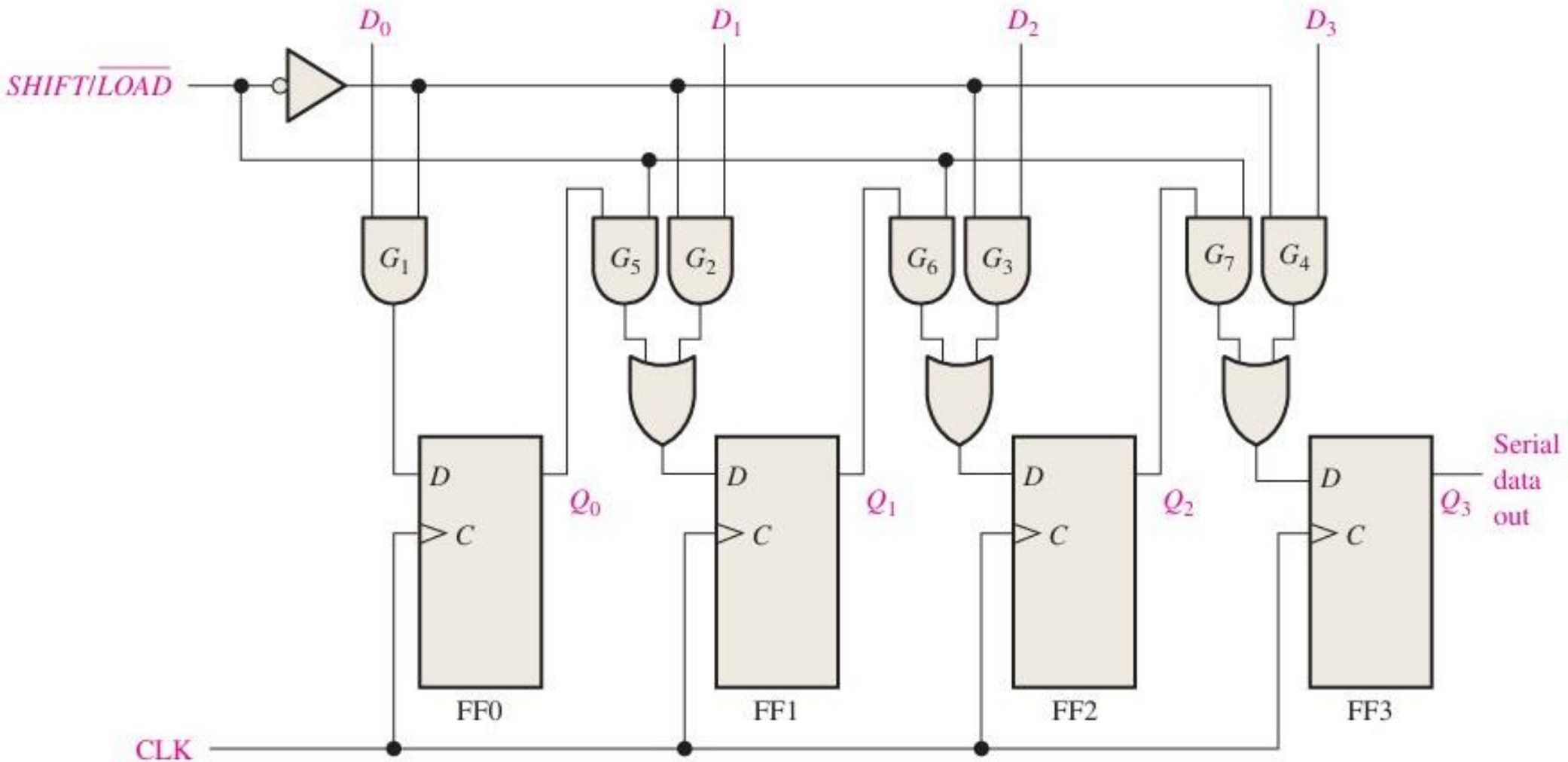
3. Parallel In Serial Out

- This shift register has multiple data input lines but only one output line.
- So data output is similar to SISO shift register.



(b) Logic symbol

3. Parallel In Serial Out



(a) Logic diagram

3. Parallel In Serial Out

- There are Four data input lines D_0 , D_1 , D_2 , D_3 and a SHIFT / $\overline{\text{LOAD}}$ input.
- The SHIFT/$\overline{\text{LOAD}}$ input allows the 4 bits of data to load in parallel into the register.
- When SHIFT/ $\overline{\text{LOAD}}$ is LOW,

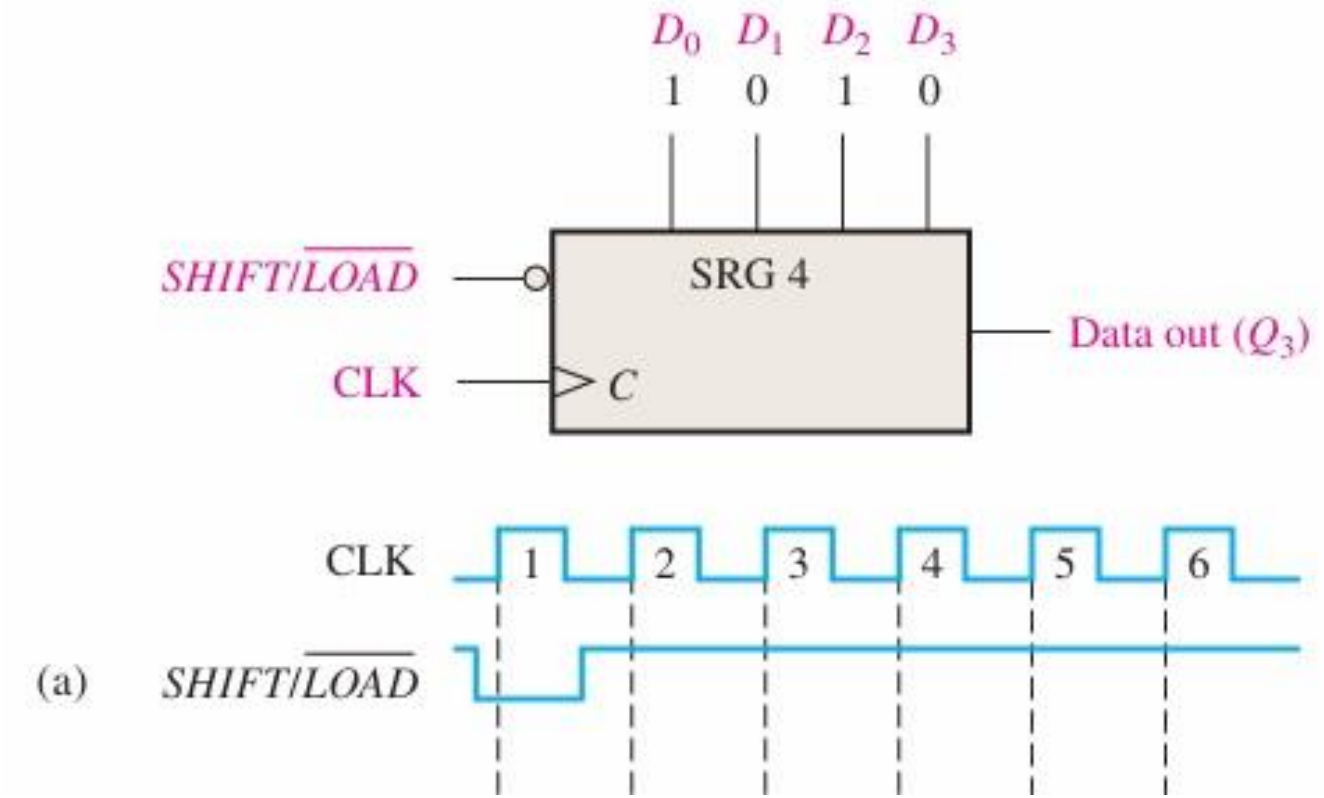
  gates G_1 through G_4 are enabled. This allows each bit to be applied to the data input of its respective flip-flop.
- When clock pulse is applied, the flip-flops are SET or RESET.
- When SHIFT/$\overline{\text{LOAD}}$is HIGH,

  G_1 through G_4 are disabled and G_5 through G_7 are enabled. This allows 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 SHIFT/ $\overline{\text{LOAD}}$ input.

3. Parallel In Serial Out

EXAMPLE 8–3

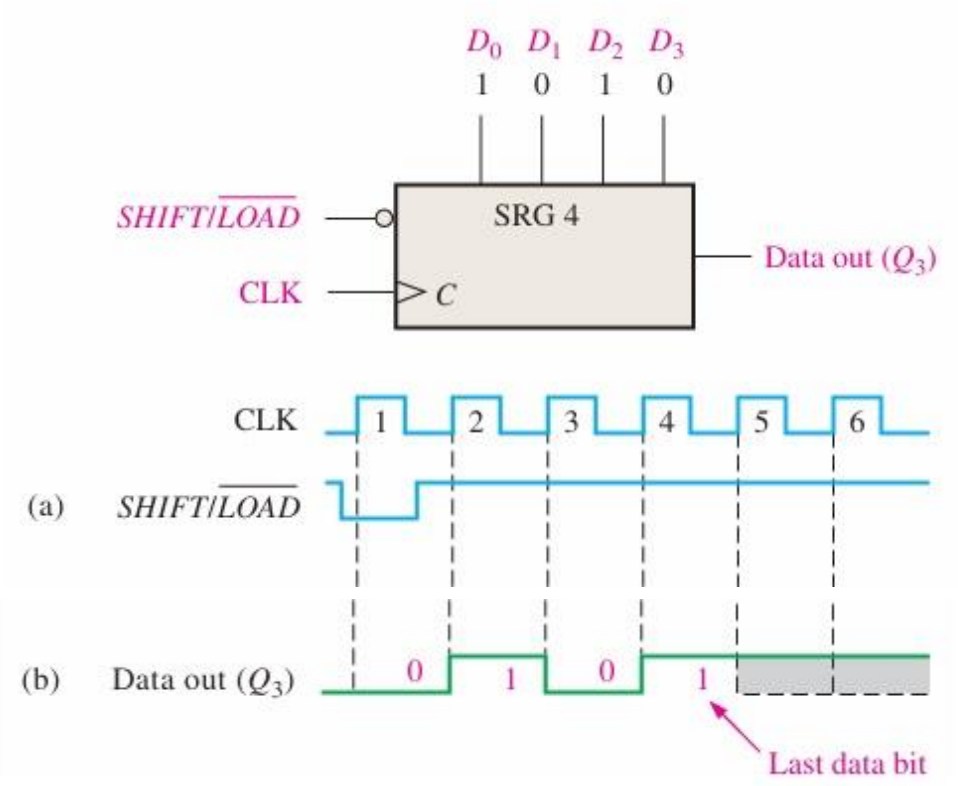
Show the data-output waveform for a 4-bit register with the parallel input data and the clock and *SHIFT*/$\overline{LOAD}$ waveforms given in Figure 8–11(a). Refer to Figure 8–10(a) for the logic diagram.



3. Parallel In Serial Out

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.

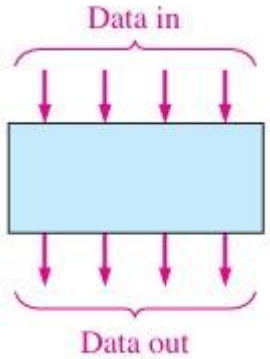


Solution

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).

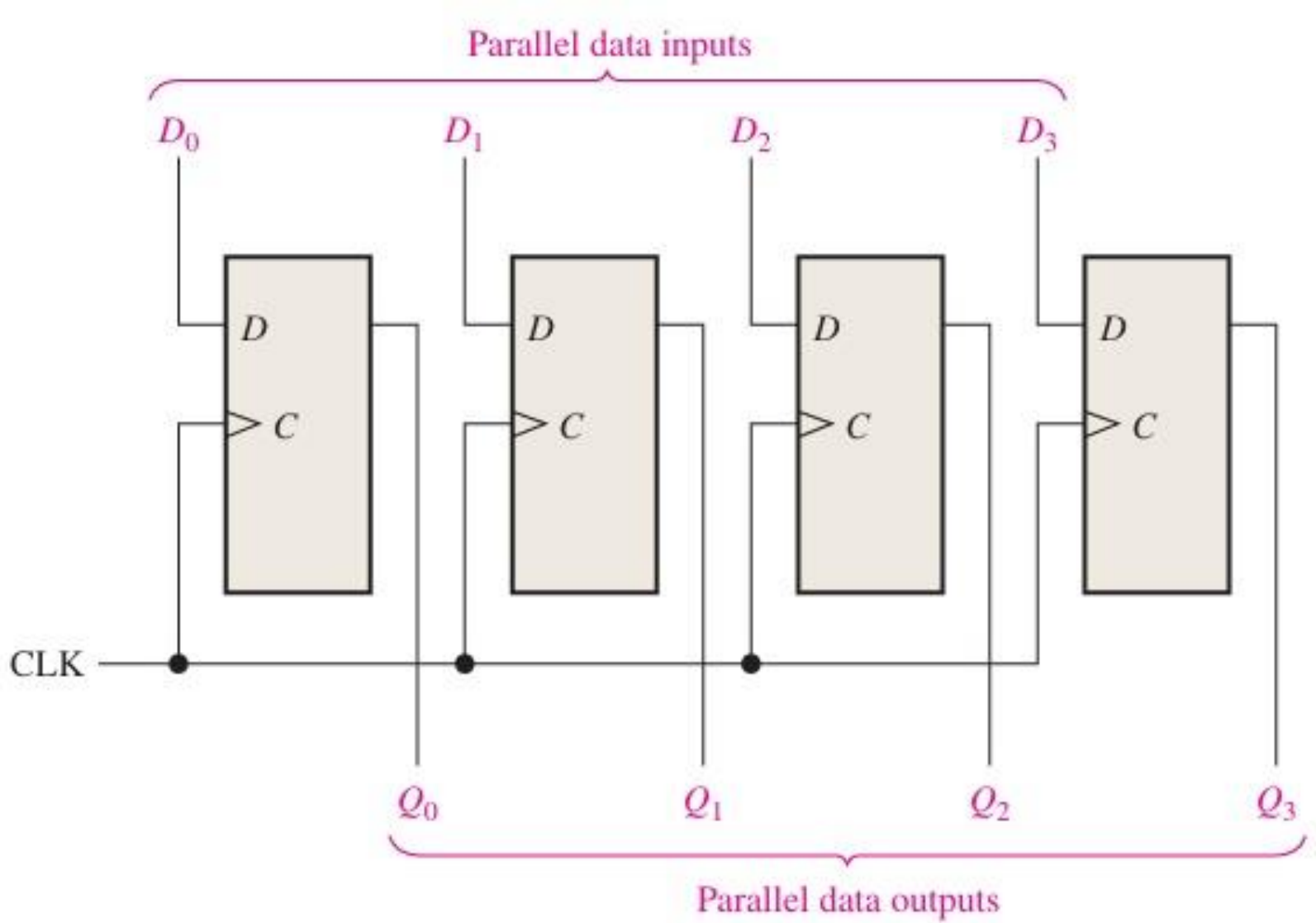
4. Parallel In Parallel Out

- In this register multiple data input lines as well as multiple data output lines are available.
- Data bits from all inputs enter their respective flip-flops (stages) simultaneously.
- Once the data is stored, each bit appears on its respective output line simultaneously.



(e) Parallel in/parallel out

4. Parallel In Parallel Out



Logic Diagram

Thank You