Chapter 10  Shift Registers

10.1 Introduction

A Register is a digital circuit that both stores data and moves data. The term shift register is used to highlight that the register also moves data. Since a shift register stores data it implemented using flip-flops. The flip-flop operation required is that of a D-type flip-flop. Remember that the truth table for a D-type flip-flop is as follows:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Outputs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$D$</td>
<td>$C$</td>
<td>$Q$</td>
</tr>
<tr>
<td>$D$</td>
<td>$\uparrow$</td>
<td>$D$</td>
</tr>
</tbody>
</table>

In other words, when a clock edge is detected (in this case rising clock edge) the output $Q$ becomes equal to the input $D$. The D-type flip-flop can be implemented from a JK flip-flop as follows:

We can say that when a clock edge is detected the D-type flip-flop stores the value of $D$. For each bit that we want to store we require a flip-flop. Therefore if we require to store 4 bits then 4 flip-flops are needed. Each flip-flop stores one bit, that is each stage (the number of stages is often used to refer to the number of flip-flops a register contains and hence the number of bits it can store) of the register stores one bit.

There are a variety of implementations of shift registers. Simplified diagrams are shown below naming the various types possible (4-bit shift registers are shown).
1. Serial-in Parallel-out
Data in → Data out

2. Parallel-in Parallel-out
Data in → Data out

3. Parallel-in Serial-out
Data in → Data out

4. Serial-in Parallel-out
Data in → Data out

1(c) Rotate Left

1(d) Rotate Right
Note that the variations that exist for Serial-in Serial-out shift registers also exist for the other types of shift register. They are not shown here for simplicity.

The remaining sections of this chapter look at the implementation of each of the four types of shift register.

10.2 Serial-In Serial-Out Shift Registers

A Serial-in Serial-out shift register can be implemented using D-type flip-flops joined together, the output of one flip-flop used as the input to the next flip-flop. The circuit for a 4-bit Serial-in Serial-out shift register is shown below.

The operation of the serial-in Serial-out shift register can be easily explained. Consider the circuit shown. On each clock edge (rising in this case) we can say the following about the outputs of each stage of the register (i.e. each D-type flip-flop in the register):

\[
\begin{align*}
\text{Data out} & = Q_3 = D_3 = Q_2; \\
Q_2 & = D_2 = Q_1; \\
Q_1 & = D_1 = Q_0; \\
Q_0 & = D_0 = \text{Data in};
\end{align*}
\]

On each rising clock edge

Which simplifies to:

\[
\begin{align*}
\text{Data out} & = Q_2; \\
Q_3 & = Q_1; \\
Q_1 & = Q_0; \\
Q_0 & = \text{Data in};
\end{align*}
\]

On each rising clock edge

Hence we can use these equations to step through the register's operation.

Consider that all the flip-flops outputs are 0. If Data in is LOW, i.e. 0, and a rising clock edge occurs then the equations listed can be applied to determine the new outputs of the flip-flops, i.e.
Data out = Q_2 = 0;
Q_2 = Q_1 = 0;
Q_1 = Q_0 = 0;
Q_0 = Data in = 0;

<table>
<thead>
<tr>
<th>Q_0</th>
<th>Q_1</th>
<th>Q_2</th>
<th>Q_3 = Data out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

There is no change. If Data in is HIGH, i.e. 1, and a rising clock edge occurs then once again the equations listed can be applied to determine the new outputs of the flip-flops, i.e.

Data out = Q_2 = 0;
Q_2 = Q_1 = 0;
Q_1 = Q_0 = 0;
Q_0 = Data in = 1;

<table>
<thead>
<tr>
<th>Q_0</th>
<th>Q_1</th>
<th>Q_2</th>
<th>Q_3 = Data out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

A 1 is now present at the output of the first flip-flop. If Data in remains HIGH, i.e. 1, and another rising clock edge occurs then the flip-flop outputs become:

Data out = Q_2 = 0;
Q_2 = Q_1 = 0;
Q_1 = Q_0 = 1;
Q_0 = Data in = 1;

<table>
<thead>
<tr>
<th>Q_0</th>
<th>Q_1</th>
<th>Q_2</th>
<th>Q_3 = Data out</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Data in is made LOW, i.e. 0, and another rising clock edge occurs. The flip-flop outputs become:

Data out = Q_2 = 0;
Q_2 = Q_1 = 1;
Q_1 = Q_0 = 1;
Q_0 = Data in = 0;

<table>
<thead>
<tr>
<th>Q_0</th>
<th>Q_1</th>
<th>Q_2</th>
<th>Q_3 = Data out</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Data in is made HIGH, i.e. 1, and another rising clock edge occurs. The flip-flop outputs become:

Data out = Q_2 = 1;
Q_2 = Q_1 = 1;
Q_1 = Q_0 = 0;
Q_0 = Data in = 1;
We can see that the bit sequence $1101$ has been serially shifted into the register. As rising clock edges continue to occur the sequence will be serially shifted out of the shift register, i.e.

**Data in** is made LOW, i.e. 0, and another rising clock edge occurs. The flip-flop outputs become:

$\text{Data out} = Q_2 = 1$;
$Q_2 = Q_1 = 0$;
$Q_1 = Q_0 = 1$;
$Q_0 = \text{Data in} = 0$;

<table>
<thead>
<tr>
<th>$Q_0$</th>
<th>$Q_1$</th>
<th>$Q_2$</th>
<th>$Q_3 = \text{Data out}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Data in** remains LOW, i.e. 0, and another rising clock edge occurs. The flip-flop outputs become:

$\text{Data out} = Q_2 = 0$;
$Q_2 = Q_1 = 1$;
$Q_1 = Q_0 = 0$;
$Q_0 = \text{Data in} = 0$;

<table>
<thead>
<tr>
<th>$Q_0$</th>
<th>$Q_1$</th>
<th>$Q_2$</th>
<th>$Q_3 = \text{Data out}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

**Data in** remains LOW, i.e. 0, and another rising clock edge occurs. The flip-flop outputs become:

$\text{Data out} = Q_2 = 1$;
$Q_2 = Q_1 = 0$;
$Q_1 = Q_0 = 0$;
$Q_0 = \text{Data in} = 0$;

<table>
<thead>
<tr>
<th>$Q_0$</th>
<th>$Q_1$</th>
<th>$Q_2$</th>
<th>$Q_3 = \text{Data out}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Data in** is made LOW, i.e. 0, and another rising clock edge occurs. The flip-flop outputs become:

$\text{Data out} = Q_2 = 0$;
$Q_2 = Q_1 = 0$;
$Q_1 = Q_0 = 0$;
$Q_0 = \text{Data in} = 0$;

<table>
<thead>
<tr>
<th>$Q_0$</th>
<th>$Q_1$</th>
<th>$Q_2$</th>
<th>$Q_3 = \text{Data out}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
At this point the shift register has returned to its original state of 0s at the outputs of the flip-flops. It can be seen therefore that with the circuit given a 4-bit number can be shifted serially into the register, where it is stored, before being shifted serially out of the shift register.

If it is desired that the data be rotated in the shift register the circuit would need to be modified as shown:

If $\text{ROTATE}$ is HIGH then the effect is to pass Data in through the AND gate to the OR gate and block the Data out from the OR gate. This is the case because we know that (from Chapter 3) an AND gate will only give an output of 1 when both inputs are 1. When $\text{ROTATE}$ is LOW then the effect is to block Data in and pass Data out to the input of the first flip-flop. Then, each time a rising clock edge is detected, the data will be rotated through the shift register.

### 10.3 Parallel-In Parallel-Out Shift Registers

A Parallel-in Parallel-out shift register can be implemented as follows.
The operation of the Parallel-in Parallel-out shift register is straightforward. When a rising clock edge is detected the outputs become equal to the inputs, i.e.

\[
\begin{align*}
    Q_3 &= D_3 \\
    Q_2 &= D_2 \\
    Q_1 &= D_1 \\
    Q_0 &= D_0 \\
\end{align*}
\]

**10.4 Serial-In Parallel-Out Shift Registers**

A Serial-in Parallel-out shift register can be implemented using a circuit similar to a Serial-in Serial-out shift register, i.e. the D-type flip-flops are joined together and the output of one flip-flop used as the input to the next flip-flop. The circuit is:

The operation of the Serial-in Parallel-out shift register is the same as that of the Serial-in Serial-out register except that the Data out is \( n \)-bits for an \( n \)-bit register, in this case \( n = 4 \). Hence the data out is \( Q_0, Q_1, Q_2 \) and \( Q_3 \). In the Serial-in Serial-out register Data out is \( Q_3 \) only.

**10.5 Parallel-In Serial-Out Shift Registers**

The Parallel-in Serial-out shift register is the most complex of the shift registers because it logic to determine whether you are loading data to the register or you are shifting data currently in the register. The circuit for a Parallel-in Serial-out register is shown on the following page. The circuit should be interpreted as follows. Each input to each flip-flop can either be data in or data shifted from the previous flip-flop. Which is required depends on the state of \( SHIFT / LOAD \). If it is HIGH, i.e. we want the register to shift data (not read in new data) then the data in inputs are disabled because they are ANDed with 0. The shift data (the output of the previous flip-flop) is
passed through to the input of the next flip-flop because it is ANDed with 1. When \( \text{SHIFT} / \overline{\text{LOAD}} \) is LOW then we want to load new data to the register. In this case the data in is passed through to the input of each flip-flop and the shift data is blocked. The circuit is shown below:

Notice that the behaviour of the \( \text{SHIFT} / \overline{\text{LOAD}} \) circuitry is the same as that of the \( \overline{\text{ROTATE}} \) circuitry in Section 10.2.