Clocked flip-flops are also provided with asynchronous or direct Set and Reset inputs, $S_d$ and $R_d$ (also called Preset and Clear, respectively) which override all other inputs (J, K, CLK).

The $S_d$ and $R_d$ inputs may be active low; in that case, they are denoted by $S_d'$ and $R_d'$.

The asynchronous inputs are convenient for “starting up” a circuit in a known state.

---

### JK flip-flop: asynchronous inputs

<table>
<thead>
<tr>
<th>$S_d$</th>
<th>$R_d$</th>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>$Q_{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>$X$</td>
<td>$X$</td>
<td>$X$</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>$X$</td>
<td>$X$</td>
<td>$X$</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>$X$</td>
<td>$X$</td>
<td>$X$</td>
<td>invalid</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>$\uparrow$</td>
<td>0</td>
<td>0</td>
<td>$Q_n$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>$\uparrow$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>$\uparrow$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>$\uparrow$</td>
<td>1</td>
<td>1</td>
<td>$\overline{Q}_n$</td>
</tr>
</tbody>
</table>

normal operation
Clocked flip-flops are also provided with *asynchronous* or *direct* Set and Reset inputs, $S_d$ and $R_d$, (also called Preset and Clear, respectively) which override all other inputs ($J, K, CLK$).
Clocked flip-flops are also provided with *asynchronous* or *direct* Set and Reset inputs, \( S_d \) and \( R_d \), (also called Preset and Clear, respectively) which override all other inputs (J, K, CLK).

* The \( S_d \) and \( R_d \) inputs may be active low; in that case, they are denoted by \( \overline{S_d} \) and \( \overline{R_d} \).
Clocked flip-flops are also provided with asynchronous or direct Set and Reset inputs, $S_d$ and $R_d$, (also called Preset and Clear, respectively) which override all other inputs (J, K, CLK).

The $S_d$ and $R_d$ inputs may be active low; in that case, they are denoted by $\overline{S_d}$ and $\overline{R_d}$.

The asynchronous inputs are convenient for “starting up” a circuit in a known state.
The D flip-flop can be used to delay the Data (D) signal by one clock period. With \( J = D \), \( K = D \), we have either \( J = 0 \), \( K = 1 \) or \( J = 1 \), \( K = 0 \); the next \( Q \) is 0 in the first case, 1 in the second case. Instead of a JK flip-flop, an RS flip-flop can also be used to make a D flip-flop, with \( S = D \), \( R = D \).
The D flip-flop can be used to delay the Data (D) signal by one clock period.

<table>
<thead>
<tr>
<th>CLK</th>
<th>D</th>
<th>Q_{n+1}</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

* Instead of a JK flip-flop, an RS flip-flop can also be used to make a D flip-flop, with \( S = D \), \( R = \overline{D} \).
The D flip-flop can be used to delay the Data (D) signal by one clock period.

With $J = D$, $K = \overline{D}$, we have either $J = 0$, $K = 1$ or $J = 1$, $K = 0$; the next $Q$ is 0 in the first case, 1 in the second case.
The D flip-flop can be used to delay the Data (D) signal by one clock period.

With $J = D$, $K = \overline{D}$, we have either $J = 0$, $K = 1$ or $J = 1$, $K = 0$; the next $Q$ is 0 in the first case, 1 in the second case.

Instead of a JK flip-flop, an RS flip-flop can also be used to make a D flip-flop, with $S = D$, $R = \overline{D}$. 
Let $Q_1 = Q_2 = Q_3 = Q_4 = 0$ initially.

**Shift register**
Let $Q_1 = Q_2 = Q_3 = Q_4 = 0$ initially.

Let $Q_1 = Q_2 = Q_3 = Q_4 = 0$ initially.

(SEQUEL file: ee101_shift_reg_1.sqproj)
Let $Q_1 = Q_2 = Q_3 = Q_4 = 0$ initially.
Let $Q_1 = Q_2 = Q_3 = Q_4 = 0$ initially.

![Shift register diagram](image-url)
Let $Q_1 = Q_2 = Q_3 = Q_4 = 0$ initially.
Let $Q_1 = Q_2 = Q_3 = Q_4 = 0$ initially.
Let $Q_1 = Q_2 = Q_3 = Q_4 = 0$ initially.

The data ($D$) keeps shifting right after each active clock edge.

M. B. Patil, IIT Bombay
Let \( Q_1 = Q_2 = Q_3 = Q_4 = 0 \) initially.

The data (D) keeps shifting right after each active clock edge.

(SEQUEL file: ee101_shift_reg_1.sqproj)
Let $Q_1 = Q_2 = Q_3 = Q_4 = 0$ initially.
Let $Q_1 = Q_2 = Q_3 = Q_4 = 0$ initially.

The data (D) keeps shifting right after each active clock edge.

(SEQUEL file: ee101_shift_reg_1.sqproj)
Let $Q_1 = Q_2 = Q_3 = Q_4 = 0$ initially.

The data (D) keeps shifting right after each active clock edge.

(SEQUEL file: ee101_shift_reg_1.sqproj)

M. B. Patil, IIT Bombay
Let \( Q_1 = Q_2 = Q_3 = Q_4 = 0 \) initially.

\[ \begin{array}{cccc}
D & Q_1 & Q_2 & Q_3 \\
\hline
D & Q & D & Q \\
\hline
D & Q & D & Q \\
\hline
D & Q & D & Q \\
\end{array} \]

CLK

\[ \begin{array}{cccc}
Q_4 & Q_3 & Q_2 & Q_1 \\
\hline
0 & 1 & 0 & 1 \\
1 & 1 & 0 & 0 \\
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 \\
\end{array} \]

(SEQUEL file: ee101_shift_reg_1.sqproj)
Let $Q_1 = Q_2 = Q_3 = Q_4 = 0$ initially.

* The data (D) keeps shifting right after each active clock edge.
Parallel transfer between shift registers

After the active clock edge, the contents of the A register ($A_3 A_2 A_1 A_0$) are copied to the B register.
Parallel transfer between shift registers

* After the active clock edge, the contents of the A register \((A_3A_2A_1A_0)\) are copied to the B register.
When the mode input (M) is 1, we have
\[ D_0 = D_R, \quad D_1 = Q_0, \quad D_2 = Q_1, \quad D_3 = Q_2. \]

When the mode input (M) is 0, we have
\[ D_0 = Q_1, \quad D_1 = Q_2, \quad D_2 = Q_3, \quad D_3 = D_L. \]

* M = 1 → shift right operation.
* M = 0 → shift left operation.
When the mode input (M) is 1, we have
\[ D_0 = D_R, \quad D_1 = Q_0, \quad D_2 = Q_1, \quad D_3 = Q_2. \]
* When the mode input (M) is 1, we have
  \[ D_0 = D_R, \quad D_1 = Q_0, \quad D_2 = Q_1, \quad D_3 = Q_2. \]

* When the mode input (M) is 0, we have
  \[ D_0 = Q_1, \quad D_1 = Q_2, \quad D_2 = Q_3, \quad D_3 = D_L. \]
When the mode input (M) is 1, we have
\[ D_0 = D_R, \quad D_1 = Q_0, \quad D_2 = Q_1, \quad D_3 = Q_2. \]

When the mode input (M) is 0, we have
\[ D_0 = Q_1, \quad D_1 = Q_2, \quad D_2 = Q_3, \quad D_3 = D_L. \]

* \( M = 1 \) → shift right operation.
* \( M = 0 \) → shift left operation.
Multiplication using shift and add

\[
\begin{array}{c}
1 & 0 & 1 & 1 \\
\times & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & Z \\
\hline
0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & Z & Z \\
\hline
1 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & Z & Z & Z \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}
\]

A_3 A_2 A_1 A_0 (decimal 11)
B_3 B_2 B_1 B_0 (decimal 13)
since B_0 = 1
since B_1 = 0
addition
since B_2 = 1
addition
since B_3 = 1
addition (decimal 143)

Note that Z = 0. We use Z to denote 0s which are independent of the numbers being multiplied.
Multiplication using shift and add

\[ \begin{array}{c}
1 & 0 & 1 & 1 \\
\times & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 1 \\
+ & 0 & 0 & 0 & 0 & Z \\
\hline
0 & 1 & 0 & 1 & 1 \\
+ & 1 & 0 & 1 & 1 & Z & Z \\
\hline
1 & 1 & 0 & 1 & 1 & 1 \\
+ & 1 & 0 & 1 & 1 & Z & Z & Z \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array} \]

\( A_3A_2A_1A_0 \) (decimal 11)
\( B_3B_2B_1B_0 \) (decimal 13)

since \( B_0 = 1 \)
since \( B_1 = 0 \)

addition
since \( B_2 = 1 \)

addition
since \( B_3 = 1 \)

addition (decimal 143)

Note that \( Z = 0 \). We use \( Z \) to denote 0s which are independent of the numbers being multiplied.
Multiplication using shift and add

\[
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
\times & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & Z \\
+ & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & Z & Z \\
+ & 1 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & Z & Z \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{array}
\]
Multiplication using shift and add

\[
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
\times & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 1 \\
+ & 0 & 0 & 0 & 0 & Z \\
\hline
1 & 0 & 1 & 1 & Z & Z \\
+ & 1 & 1 & 0 & 1 & 1 \\
\hline
1 & 1 & 0 & 1 & 1 & 1 \\
+ & 1 & 0 & 1 & 1 & Z & Z \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}
\]

\[A_3A_2A_1A_0 \text{ (decimal 11)}\]
\[B_3B_2B_1B_0 \text{ (decimal 13)}\]

\[
\begin{array}{cccc}
\text{Register 2} & \text{Register 1} \\
\hline
Z & Z & Z & Z \\
1 & 0 & 1 & 1 \\
\hline
1 & 0 & 1 & 1 & Z & Z & Z & Z \\
+ & & & & & & & \\
\hline
1 & 1 & 0 & 1 & 1 & 1 \\
+ & 1 & 0 & 1 & 1 & Z & Z \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}
\]

Note that \(Z = 0\). We use \(Z\) to denote 0s which are independent of the numbers being multiplied.
Multiplication using shift and add

\[
\begin{array}{c}
1 & 0 & 1 & 1 \\
\times & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 1 \\
& 0 & 0 & 0 & 0 & Z \\
& 0 & 1 & 0 & 1 & 1 \\
& 1 & 0 & 1 & 1 & Z & Z \\
\hline
1 & 1 & 0 & 1 & 1 & 1 \\
& 1 & 0 & 1 & 1 & Z & Z & Z & Z \\
\end{array}
\]

\[A_3A_2A_1A_0 \text{ (decimal 11)}\]
\[B_3B_2B_1B_0 \text{ (decimal 13)}\]
since $B_0 = 1$
since $B_1 = 0$
addition
since $B_2 = 1$
addition
since $B_3 = 1$
addition
(decimal 143)

Note that $Z = 0$. We use $Z$ to denote 0s which are independent of the numbers being multiplied.
Multiplication using shift and add

\[ \begin{array}{cccc}
A_3A_2A_1A_0 & (\text{decimal 11}) \\
B_3B_2B_1B_0 & (\text{decimal 13}) \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 0 & 1 & 1 \\
\times & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & Z \\
\hline
0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & Z & Z \\
\hline
1 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & Z & Z & Z \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{array} \]

since \( B_0 = 1 \)
since \( B_1 = 0 \)
addition
since \( B_2 = 1 \)
addition
since \( B_3 = 1 \)
addition
(denom 143)

Note that \( Z = 0 \). We use \( Z \) to denote 0s which are independent of the numbers being multiplied.
Multiplication using shift and add

\[
\begin{array}{c}
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
\times & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & Z \\
\hline
0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & Z & Z \\
\hline
1 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & Z & Z \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{array}
\end{array}
\]

A_3A_2A_1A_0 \text{ (decimal 11)} \times B_3B_2B_1B_0 \text{ (decimal 13)}

since B_0 = 1

addition

since B_1 = 0

addition

since B_2 = 1

addition

since B_3 = 1

\text{addition} \quad \text{(decimal 143)}

Note that Z = 0. We use Z to denote 0s which are independent of the numbers being multiplied.
Multiplication using shift and add

\[
\begin{array}{c}
1 & 0 & 1 & 1 \\
\times & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & Z \\
+ & 0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & Z & Z \\
+ & 1 & 1 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & Z & Z & Z \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\(A_3A_2A_1A_0 \) (decimal 11)
\(B_3B_2B_1B_0 \) (decimal 13)

Since \(B_0 = 1\)
Since \(B_1 = 0\)
Addition
Since \(B_2 = 1\)
Addition
Since \(B_3 = 1\)
Addition (decimal 143)

Note that \(Z = 0\). We use \(Z\) to denote 0s which are independent of the numbers being multiplied.
Multiplication using shift and add

\[
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
\times & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & Z \\
\hline
0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & Z & Z \\
\hline
1 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & Z & Z \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 \\
\end{array}
\]

\(A_3A_2A_1A_0\) (decimal 11)
\(B_3B_2B_1B_0\) (decimal 13)

since \(B_0 = 1\)
since \(B_1 = 0\)
addition
since \(B_2 = 1\)
addition
since \(B_3 = 1\)
addition (decimal 143)

Note that \(Z = 0\). We use \(Z\) to denote 0s which are independent of the numbers being multiplied.
Multiplication using shift and add

\[
\begin{array}{cccccccc}
1 & 0 & 1 & 1 \\
\times & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & Z \\
\hline
0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & Z & Z \\
\hline
1 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & Z & Z \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1
\end{array}
\]

\(A_3A_2A_1A_0\) (decimal 11)
\(B_3B_2B_1B_0\) (decimal 13)

since \(B_0 = 1\)
since \(B_1 = 0\)
addition
since \(B_2 = 1\)
addition
since \(B_3 = 1\)
addition
(decimal 143)

Note that \(Z = 0\). We use \(Z\) to denote 0s which are independent of the numbers being multiplied.
Multiplication using shift and add

\[
\begin{array}{c}
1 & 0 & 1 & 1 \\
\times & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & Z \\
\hline
0 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & Z Z \\
\hline
1 & 1 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & Z Z \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{array}
\]

A_3A_2A_1A_0 (decimal 11)
B_3B_2B_1B_0 (decimal 13)
since B_0 = 1
since B_1 = 0
addition
since B_2 = 1
addition
since B_3 = 1
addition (decimal 143)

Note that Z = 0. We use Z to denote 0s which are independent of the numbers being multiplied.

Register 2

\[
\begin{array}{c}
Z & Z & Z & Z \\
1 & 0 & 1 & 1 \\
Z & 1 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 \\
Z & 0 & 1 & 0 & 1 \\
1 & 1 & Z & Z \\
Z & 1 & 1 & 0 & 1 \\
1 & 1 & 1 & Z \\
\end{array}
\]

Register 1

\[
\begin{array}{c}
Z & Z & Z & Z \\
1 & 0 & 1 & 1 \\
1 & Z & Z & Z \\
0 & 0 & 0 & 0 \\
1 & Z & Z & Z \\
1 & Z & Z & Z \\
1 & 1 & Z & Z \\
1 & 1 & Z & Z \\
\end{array}
\]

initialize
load 1011 since B_0 = 1
add
shift
load 0000 since B_1 = 0
add
shift
load 1011 since B_2 = 1
add
shift
load 1011 since B_3 = 1
add
shift
Multiplication using shift and add

\[
\begin{array}{c}
1 \ 0 \ 1 \ 1 \\
\times \ 1 \ 1 \ 0 \ 1 \\
\hline
1 \ 0 \ 1 \ 1 \\
0 \ 0 \ 0 \ 0 \ Z \\
+ \\
0 \ 1 \ 0 \ 1 \ 1 \\
1 \ 0 \ 1 \ 1 \ Z \ Z \\
+ \\
1 \ 1 \ 0 \ 1 \ 1 \\
1 \ 0 \ 1 \ 1 \ Z \ Z \\
\hline
1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1
\end{array}
\]

A_3A_2A_1A_0 (decimal 11) 
B_3B_2B_1B_0 (decimal 13)

since B_0 = 1
since B_1 = 0

addition

since B_2 = 1

addition

since B_3 = 1

addition (decimal 143)

Note that Z = 0. We use Z to denote 0s which are independent of the numbers being multiplied.
### Multiplication using shift and add

<table>
<thead>
<tr>
<th>A3A2A1A0 (decimal 11)</th>
<th>B3B2B1B0 (decimal 13)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 1</td>
<td>1 1 0 1</td>
</tr>
<tr>
<td><strong>×</strong></td>
<td><strong>×</strong></td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>0 0 0 0 Z</td>
</tr>
<tr>
<td><strong>+</strong></td>
<td><strong>+</strong></td>
</tr>
<tr>
<td>0 1 0 1 1</td>
<td>1 0 1 1 Z Z Z</td>
</tr>
<tr>
<td><strong>+</strong></td>
<td><strong>+</strong></td>
</tr>
<tr>
<td>1 1 0 1 1 1</td>
<td>0 0 1 1 1 Z Z Z</td>
</tr>
<tr>
<td><strong>addition</strong></td>
<td><strong>addition</strong></td>
</tr>
<tr>
<td>since B0 = 1</td>
<td>since B0 = 1</td>
</tr>
<tr>
<td>since B1 = 0</td>
<td>since B1 = 0</td>
</tr>
<tr>
<td>addition</td>
<td>addition</td>
</tr>
<tr>
<td>since B2 = 1</td>
<td>since B2 = 1</td>
</tr>
<tr>
<td>addition</td>
<td>addition</td>
</tr>
<tr>
<td>since B3 = 1</td>
<td>since B3 = 1</td>
</tr>
<tr>
<td>addition</td>
<td>addition</td>
</tr>
<tr>
<td>(decimal 143)</td>
<td>(decimal 143)</td>
</tr>
</tbody>
</table>

Note that Z = 0. We use Z to denote 0s which are independent of the numbers being multiplied.
Multiplication using shift and add

\[
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
\times & 1 & 1 & 0 & 1 \\
\hline
1 & 0 & 1 & 1 \\
+ & 0 & 0 & 0 & 0 \ Z \\
\hline
0 & 1 & 0 & 1 & 1 \\
+ & 1 & 0 & 1 & 1 \ Z \ Z \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\end{array}
\]

\( A_3A_2A_1A_0 \) (decimal 11)
\( B_3B_2B_1B_0 \) (decimal 13)

since \( B_0 = 1 \)
since \( B_1 = 0 \)
addition
since \( B_2 = 1 \)
addition
since \( B_3 = 1 \)
addition (decimal 143)

Note that \( Z = 0 \). We use \( Z \) to denote 0s which are independent of the numbers being multiplied.
Parallel in-serial out data movement

- All flip-flops are cleared in the beginning (with \( \text{Load} = \text{Clear} = 1 \), \( \text{Sd} = 0 \)).
- When \( \text{Load} = 1 \), \( \text{Sd} = A_i, \text{Rd} = 0 \) → \( A_i \) gets loaded into the \( i \)th flip-flop. (We will assume that \( \text{CLK} \) has been made 0 in this initial phase.)
- Subsequently, with every clock pulse, the data shifts right and appears serially at the output \( Q_0 \).
Parallel in-serial out data movement

* All flip-flops are cleared in the beginning (with $R_d = \text{Clear} = 1$, $S_d = 0$).
Parallel in-serial out data movement

* All flip-flops are cleared in the beginning (with $R_d = \text{Clear} = 1$, $S_d = 0$).
* When $\text{Load} = 1$, $S_d = A_i$, $R_d = 0 \rightarrow A_i$ gets loaded into the $i^{th}$ flip-flop. (We will assume that CLK has been made 0 in this initial phase.)
Parallel in-serial out data movement

- All flip-flops are cleared in the beginning (with \( R_d = \text{Clear} = 1, \ S_d = 0 \)).
- When Load = 1, \( S_d = A_i, \ R_d = 0 \rightarrow A_i \) gets loaded into the \( i^{th} \) flip-flop. (We will assume that CLK has been made 0 in this initial phase.)
- Subsequently, with every clock pulse, the data shifts right and appears \textit{serially} at the output \( Q_0 \).
Parallel in-serial out data movement

- All flip-flops are cleared in the beginning (with \( R_d = \text{Clear} = 1, \ S_d = 0 \)).
- When \( \text{Load} = 1, \ S_d = A_i, \ R_d = 0 \) → \( A_i \) gets loaded into the \( i^{th} \) flip-flop. (We will assume that \( \text{CLK} \) has been made 0 in this initial phase.)
- Subsequently, with every clock pulse, the data shifts right and appears \textit{serially} at the output \( Q_0 \).
Parallel in-serial out data movement

* All flip-flops are cleared in the beginning (with $R_d = \text{Clear} = 1$, $S_d = 0$).
* When $\text{Load} = 1$, $S_d = A_i$, $R_d = 0 \rightarrow A_i$ gets loaded into the $i^{th}$ flip-flop. (We will assume that CLK has been made 0 in this initial phase.)
* Subsequently, with every clock pulse, the data shifts right and appears *serially* at the output $Q_0$. 

\[ Q_0 = A_0 \quad Q_0 = A_1 \quad Q_0 = A_2 \quad Q_0 = A_3 \]
Parallel in-serial out data movement

- All flip-flops are cleared in the beginning (with $R_d = \text{Clear} = 1$, $S_d = 0$).
- When Load = 1, $S_d = A_i$, $R_d = 0 \rightarrow A_i$ gets loaded into the $i^{th}$ flip-flop. (We will assume that CLK has been made 0 in this initial phase.)
- Subsequently, with every clock pulse, the data shifts right and appears serially at the output $Q_0$. 

M. B. Patil, IIT Bombay
Parallel in-serial out data movement

* All flip-flops are cleared in the beginning (with \( R_d = \text{Clear} = 1, S_d = 0 \)).
* When \( \text{Load} = 1 \), \( S_d = A_i \), \( R_d = 0 \) \( \rightarrow A_i \) gets loaded into the \( i^{th} \) flip-flop. (We will assume that \( \text{CLK} \) has been made 0 in this initial phase.)
* Subsequently, with every clock pulse, the data shifts right and appears \textit{serially} at the output \( Q_0 \).
Parallel in-serial out data movement

* All flip-flops are cleared in the beginning (with $R_d = \text{Clear} = 1$, $S_d = 0$).
* When Load = 1, $S_d = A_i$, $R_d = 0 \rightarrow A_i$ gets loaded into the $i^{th}$ flip-flop. (We will assume that CLK has been made 0 in this initial phase.)
* Subsequently, with every clock pulse, the data shifts right and appears serially at the output $Q_0$. 

\[ Q_0 = A_0 \quad Q_0 = A_1 \quad Q_0 = A_2 \quad Q_0 = A_3 \]
* All flip-flops are cleared in the beginning (with $R_d = \text{Clear} = 1$, $S_d = 0$).
* When Load = 1, $S_d = A_i$, $R_d = 0 \rightarrow A_i$ gets loaded into the $i^{th}$ flip-flop. (We will assume that CLK has been made 0 in this initial phase.)
* Subsequently, with every clock pulse, the data shifts right and appears *serially* at the output $Q_0$. 
A counter with $k$ states is called a modulo-$k$ (mod-$k$) counter.

A counter can be made with flip-flops, each flip-flop serving as a memory element with two states (0 or 1).

If there are $N$ flip-flops in a counter, there are $2^N$ possible states (since each flip-flop can have $Q = 0$ or $Q = 1$). It is possible to exclude some of these states. → $N$ flip-flops can be used to make a mod-$k$ counter with $k \leq 2^N$.

Typically, a reset facility is also provided, which can be used to force a certain state to initialize the counter.
A counter with $k$ states is called a modulo-$k$ (mod-$k$) counter.
A counter with \( k \) states is called a modulo-\( k \) (mod-\( k \)) counter.

A counter can be made with flip-flops, each flip-flop serving as a memory element with two states (0 or 1).
- A counter with $k$ states is called a modulo-$k$ ($\text{mod-}k$) counter.
- A counter can be made with flip-flops, each flip-flop serving as a memory element with two states (0 or 1).
- If there are $N$ flip-flops in a counter, there are $2^N$ possible states (since each flip-flop can have $Q = 0$ or $Q = 1$). It is possible to exclude some of these states. 
  → $N$ flip-flops can be used to make a mod-$k$ counter with $k \leq 2^N$. 

A counter with $k$ states is called a modulo-$k$ ($\text{mod-}k$) counter. A counter can be made with flip-flops, each flip-flop serving as a memory element with two states (0 or 1). If there are $N$ flip-flops in a counter, there are $2^N$ possible states (since each flip-flop can have $Q = 0$ or $Q = 1$). It is possible to exclude some of these states. → $N$ flip-flops can be used to make a mod-$k$ counter with $k \leq 2^N$. 

Typically, a reset facility is also provided, which can be used to force a certain state to initialize the counter.
* A counter with $k$ states is called a modulo-$k$ (mod-$k$) counter.
* A counter can be made with flip-flops, each flip-flop serving as a memory element with two states (0 or 1).
* If there are $N$ flip-flops in a counter, there are $2^N$ possible states (since each flip-flop can have $Q = 0$ or $Q = 1$). It is possible to exclude some of these states. → $N$ flip-flops can be used to make a mod-$k$ counter with $k \leq 2^N$.
* Typically, a reset facility is also provided, which can be used to force a certain state to initialize the counter.
The counter outputs (i.e., the flip-flop outputs, $Q_0, Q_1, \ldots, Q_{N-1}$) can be decoded using appropriate logic. In particular, it is possible to have a decoder output (say, $X$) which is 1 only for state $i$, and 0 otherwise. For $k$ clock pulses, we get a single pulse at $X$, i.e., the clock frequency has been divided by $k$. For this reason, a mod-$k$ counter is also called a divide-by-$k$ counter.
The counter outputs (i.e., the flip-flop outputs, \( Q_0, Q_1, \ldots, Q_{N-1} \)) can be decoded using appropriate logic.
Counters

* The counter outputs (i.e., the flip-flop outputs, $Q_0$, $Q_1$, $\cdots$, $Q_{N-1}$) can be decoded using appropriate logic.

* In particular, it is possible to have a decoder output (say, $X$) which is 1 only for state $i$, and 0 otherwise.

→ For $k$ clock pulses, we get a single pulse at $X$, i.e., the clock frequency has been divided by $k$. For this reason, a mod-$k$ counter is also called a divide-by-$k$ counter.
A binary ripple counter

\[ J\overline{K} \]

\[
\begin{array}{c}
\text{CLK} \\
\text{FF0} \\
J \quad Q \\
K \quad \overline{Q} \\
\text{FF1} \\
J \quad Q \\
K \quad \overline{Q} \\
\text{FF2} \\
J \quad Q \\
K \quad \overline{Q} \\
\end{array}
\]

\[
\begin{array}{c}
\text{CLK} \\
Q_0 \\
Q_1 \\
Q_2 \\
\end{array}
\]

For FF1 and FF2, \( Q_0 \) and \( Q_1 \), respectively, provide the clock.

Note that the direct inputs \( S_d \) and \( R_d \) (not shown) are assumed to be \( S_d = R_d = 0 \) for all flip-flops, allowing normal flip-flop operation.

M. B. Patil, IIT Bombay
A binary ripple counter

* $J = K = 1$ for all flip-flops. Let $Q_0 = Q_1 = Q_2 = 0$ initially.
A binary ripple counter

* $J = K = 1$ for all flip-flops. Let $Q_0 = Q_1 = Q_2 = 0$ initially.
* Since $J = K = 1$, each flip-flop will toggle when an active (in this case, negative) clock edge arrives.
* \( J = K = 1 \) for all flip-flops. Let \( Q_0 = Q_1 = Q_2 = 0 \) initially.
* Since \( J = K = 1 \), each flip-flop will toggle when an active (in this case, negative) clock edge arrives.
* For FF1 and FF2, \( Q_0 \) and \( Q_1 \), respectively, provide the clock.
* $J = K = 1$ for all flip-flops. Let $Q_0 = Q_1 = Q_2 = 0$ initially.

* Since $J = K = 1$, each flip-flop will toggle when an active (in this case, negative) clock edge arrives.

* For FF1 and FF2, $Q_0$ and $Q_1$, respectively, provide the clock.
A binary ripple counter

* $J = K = 1$ for all flip-flops. Let $Q_0 = Q_1 = Q_2 = 0$ initially.
* Since $J = K = 1$, each flip-flop will toggle when an active (in this case, negative) clock edge arrives.
* For FF1 and FF2, $Q_0$ and $Q_1$, respectively, provide the clock.
A binary ripple counter

* $J = K = 1$ for all flip-flops. Let $Q_0 = Q_1 = Q_2 = 0$ initially.
* Since $J = K = 1$, each flip-flop will toggle when an active (in this case, negative) clock edge arrives.
* For FF1 and FF2, $Q_0$ and $Q_1$, respectively, provide the clock.
* $J = K = 1$ for all flip-flops. Let $Q_0 = Q_1 = Q_2 = 0$ initially.
* Since $J = K = 1$, each flip-flop will toggle when an active (in this case, negative) clock edge arrives.
* For FF1 and FF2, $Q_0$ and $Q_1$, respectively, provide the clock.
A binary ripple counter

- \( J = K = 1 \) for all flip-flops. Let \( Q_0 = Q_1 = Q_2 = 0 \) initially.
- Since \( J = K = 1 \), each flip-flop will toggle when an active (in this case, negative) clock edge arrives.
- For FF1 and FF2, \( Q_0 \) and \( Q_1 \), respectively, provide the clock.
- Note that the direct inputs \( S_d \) and \( R_d \) (not shown) are assumed to be \( S_d = R_d = 0 \) for all flip-flops, allowing normal flip-flop operation.

M. B. Patil, IIT Bombay
A binary ripple counter

The counter has 8 states, \( Q_2 Q_1 Q_0 = 000, 001, 010, 011, 100, 101, 110, 111 \). It is a mod-8 counter. In particular, it is a binary, mod-8, up counter (since it counts up from 000 to 111).

If the clock frequency is \( f_c \), the frequency at the \( Q_0 \), \( Q_1 \), \( Q_2 \) outputs is \( f_c / 2, f_c / 4, f_c / 8 \), respectively. For this counter, therefore, div-by-2, div-by-4, div-by-8 outputs are already available, without requiring decoding logic.

This type of counter is called a "ripple" counter since the clock transitions ripple through the flip-flops.

M. B. Patil, IIT Bombay
A binary ripple counter

The counter has 8 states, \( Q_2Q_1Q_0 = 000, 001, 010, 011, 100, 101, 110, 111 \).
\( \rightarrow \) it is a mod-8 counter. In particular, it is a binary, \textit{mod-8, up} counter (since it counts \textit{up} from 000 to 111).
A binary ripple counter

* The counter has 8 states, $Q_2 Q_1 Q_0 = 000, 001, 010, 011, 100, 101, 110, 111$. → it is a mod-8 counter. In particular, it is a binary, mod-8, up counter (since it counts up from 000 to 111).
* If the clock frequency is $f_c$, the frequency at the $Q_0$, $Q_1$, $Q_2$ outputs is $f_c/2$, $f_c/4$, $f_c/8$, respectively. For this counter, therefore, div-by-2, div-by-4, div-by-8 outputs are already available, without requiring decoding logic.
* The counter has 8 states, $Q_2 Q_1 Q_0 = 000, 001, 010, 011, 100, 101, 110, 111$.  
→ it is a mod-8 counter. In particular, it is a binary, mod-8, up counter (since it counts up from 000 to 111).

* If the clock frequency is $f_c$, the frequency at the $Q_0, Q_1, Q_2$ outputs is $f_c/2$, $f_c/4$, $f_c/8$, respectively. For this counter, therefore, div-by-2, div-by-4, div-by-8 outputs are already available, without requiring decoding logic.

* This type of counter is called a “ripple” counter since the clock transitions ripple through the flip-flops.
If positive edge-triggered flip-flops are used, we get a binary down counter (counting down from 1111 to 0000).

M. B. Patil, IIT Bombay
If positive edge-triggered flip-flops are used, we get a binary *down* counter (counting down from 1111 to 0000).
Binay ripple counters

* Home work: Sketch the waveforms (CLK, Q₀, Q₁, Q₂), and tabulate the counter states in each case.
Up-down binary ripple counters

When Mode (M) = 1, the counter counts up; else, it counts down.

(SEQUEL file: ee101_counter3.sqproj)

M. B. Patil, IIT Bombay
When Mode \((M) = 1\), the counter counts up; else, it counts down.

(SEQUEL file: ee101_counter_3.sqproj)
Decade counter using direct inputs

When the counter reaches $Q_3 Q_2 Q_1 Q_0 = 1010$ (i.e., decimal 10), $Q_3 Q_1 = 1$, and the flip-flops are cleared to $Q_3 Q_2 Q_1 Q_0 = 0000$.

The counter counts from 0000 (decimal 0) to 1001 (decimal 9) → "decade counter."

(SEQUEL file: ee101_counter_5.sqproj)
Decade counter using direct inputs

* When the counter reaches $Q_3Q_2Q_1Q_0 = 1010$ (i.e., decimal 10), $Q_3Q_1 = 1$, and the flip-flops are cleared to $Q_3Q_2Q_1Q_0 = 0000$. 
Decade counter using direct inputs

When the counter reaches \( Q_3 Q_2 Q_1 Q_0 = 1010 \) (i.e., decimal 10), \( Q_3 Q_1 = 1 \), and the flip-flops are cleared to \( Q_3 Q_2 Q_1 Q_0 = 0000 \).

The counter counts from 0000 (decimal 0) to 1001 (decimal 9) → “decade counter.”

(SEQUEL file: ee101_counter_5.sqproj)
A synchronous counter

Since all flip-flops are driven by the same clock, the counter is called a "synchronous" counter.

J₀ = K₀ = 1,
J₁ = K₁ = Q₀,
J₂ = K₂ = Q₁ Q₀,
J₃ = K₃ = Q₂ Q₁ Q₀.

FF0 toggles at every active edge.
FF1 toggles if Q₀ = 1 (just before the active clock edge); else, it retains its previous state.
Similar comments apply to FF2 and FF3.

From the waveforms, we see that it is a binary up counter.

M. B. Patil, IIT Bombay
A synchronous counter

* Since all flip-flops are driven by the same clock, the counter is called a “synchronous” counter.
A synchronous counter

Since all flip-flops are driven by the same clock, the counter is called a “synchronous” counter.

* $J_0 = K_0 = 1$, $J_1 = K_1 = Q_0$, $J_2 = K_2 = Q_1 Q_0$, $J_3 = K_3 = Q_2 Q_1 Q_0$. 

M. B. Patil, IIT Bombay
A synchronous counter

* Since all flip-flops are driven by the same clock, the counter is called a “synchronous” counter.
* \( J_0 = K_0 = 1, J_1 = K_1 = Q_0, J_2 = K_2 = Q_1 Q_0, J_3 = K_3 = Q_2 Q_1 Q_0. \)
* FF0 toggles at every active edge. FF1 toggles if \( Q_0 = 1 \) (just before the active clock edge); else, it retains its previous state. Similar comments apply to FF2 and FF3.
A synchronous counter

* Since all flip-flops are driven by the same clock, the counter is called a “synchronous” counter.
* \( J_0 = K_0 = 1, \ J_1 = K_1 = Q_0, \ J_2 = K_2 = Q_1 Q_0, \ J_3 = K_3 = Q_2 Q_1 Q_0. \)
* FF0 toggles at every active edge. FF1 toggles if \( Q_0 = 1 \) (just before the active clock edge); else, it retains its previous state. Similar comments apply to FF2 and FF3.
A synchronous counter

* Since all flip-flops are driven by the same clock, the counter is called a “synchronous” counter.
* $J_0 = K_0 = 1$, $J_1 = K_1 = Q_0$, $J_2 = K_2 = Q_1Q_0$, $J_3 = K_3 = Q_2Q_1Q_0$.
* FF0 toggles at every active edge.
  FF1 toggles if $Q_0 = 1$ (just before the active clock edge); else, it retains its previous state.
  Similar comments apply to FF2 and FF3.
Since all flip-flops are driven by the same clock, the counter is called a “synchronous” counter.

- $J_0 = K_0 = 1$, $J_1 = K_1 = Q_0$, $J_2 = K_2 = Q_1 Q_0$, $J_3 = K_3 = Q_2 Q_1 Q_0$.  
- FF0 toggles at every active edge.  
  FF1 toggles if $Q_0 = 1$ (just before the active clock edge); else, it retains its previous state.  
  Similar comments apply to FF2 and FF3.
A synchronous counter

Since all flip-flops are driven by the same clock, the counter is called a “synchronous” counter.

* $J_0 = K_0 = 1$, $J_1 = K_1 = Q_0$, $J_2 = K_2 = Q_1 Q_0$, $J_3 = K_3 = Q_2 Q_1 Q_0$.

* FF0 toggles at every active edge.
  FF1 toggles if $Q_0 = 1$ (just before the active clock edge); else, it retains its previous state.
  Similar comments apply to FF2 and FF3.

M. B. Patil, IIT Bombay
A synchronous counter

* Since all flip-flops are driven by the same clock, the counter is called a “synchronous” counter.
* \( J_0 = K_0 = 1, \ J_1 = K_1 = Q_0, \ J_2 = K_2 = Q_1 Q_0, \ J_3 = K_3 = Q_2 Q_1 Q_0. \)
* FF0 toggles at every active edge.
  FF1 toggles if \( Q_0 = 1 \) (just before the active clock edge); else, it retains its previous state.
  Similar comments apply to FF2 and FF3.
* From the waveforms, we see that it is a binary up counter.
Consider the reverse problem: We are given \( Q_n \) and the next desired state (\( Q_{n+1} \)). What should \( J \) and \( K \) be in order to make that happen? 

\( Q_n = 0, Q_{n+1} = 0 \): We can either force \( Q_{n+1} = 0 \) with \( J = 0, K = 1 \), or let \( Q_{n+1} = Q_n = 0 \) by making \( J = 0, K = 0 \). 

\( J = 0, K = X \) (i.e., \( K \) can be 0 or 1).

Similarly, work out the other entries in the table.

The table for a negative edge-triggered flip-flop would be identical except for the active edge.
Design of synchronous counters

* Consider the reverse problem: We are given $Q_n$ and the next desired state ($Q_{n+1}$). What should $J$ and $K$ be in order to make that happen?
Consider the reverse problem: We are given $Q_n$ and the next desired state ($Q_{n+1}$). What should $J$ and $K$ be in order to make that happen?

* $Q_n = 0$, $Q_{n+1} = 0$: We can either force $Q_{n+1} = 0$ with $J = 0$, $K = 1$, or let $Q_{n+1} = Q_n = 0$ by making $J = 0$, $K = 0$. 
* Consider the reverse problem: We are given $Q_n$ and the next desired state ($Q_{n+1}$). What should $J$ and $K$ be in order to make that happen?

* $Q_n = 0$, $Q_{n+1} = 0$: We can either force $Q_{n+1} = 0$ with $J = 0$, $K = 1$, or let $Q_{n+1} = Q_n = 0$ by making $J = 0$, $K = 0$.
  $\rightarrow J = 0$, $K = X$ (i.e., $K$ can be 0 or 1).
Consider the reverse problem: We are given $Q_n$ and the next desired state ($Q_{n+1}$). What should $J$ and $K$ be in order to make that happen?

* $Q_n = 0, Q_{n+1} = 0$: We can either force $Q_{n+1} = 0$ with $J = 0, K = 1$, or let $Q_{n+1} = Q_n = 0$ by making $J = 0, K = 0$.
  → $J = 0, K = X$ (i.e., $K$ can be 0 or 1).
Consider the reverse problem: We are given $Q_n$ and the next desired state ($Q_{n+1}$). What should $J$ and $K$ be in order to make that happen?

* $Q_n = 0, Q_{n+1} = 0$: We can either force $Q_{n+1} = 0$ with $J = 0, K = 1$, or let $Q_{n+1} = Q_n = 0$ by making $J = 0, K = 0$.
  → $J = 0, K = X$ (i.e., $K$ can be 0 or 1).

* Similarly, work out the other entries in the table.
Consider the reverse problem: We are given $Q_n$ and the next desired state ($Q_{n+1}$). What should $J$ and $K$ be in order to make that happen?

* $Q_n = 0$, $Q_{n+1} = 0$: We can either force $Q_{n+1} = 0$ with $J = 0$, $K = 1$, or let $Q_{n+1} = Q_n = 0$ by making $J = 0$, $K = 0$.
  → $J = 0$, $K = X$ (i.e., $K$ can be 0 or 1).

* Similarly, work out the other entries in the table.
Consider the reverse problem: We are given \( Q_n \) and the next desired state (\( Q_{n+1} \)). What should \( J \) and \( K \) be in order to make that happen?

- \( Q_n = 0, Q_{n+1} = 0 \): We can either force \( Q_{n+1} = 0 \) with \( J = 0, K = 1 \), or let \( Q_{n+1} = Q_n = 0 \) by making \( J = 0, K = 0 \).
  \[ \rightarrow J = 0, K = X \text{ (i.e., } K \text{ can be 0 or 1).} \]

- Similarly, work out the other entries in the table.

- The table for a negative edge-triggered flip-flop would be identical except for the active edge.
Design a synchronous mod-5 counter with the given state transition table.

Design of synchronous counters
Design of synchronous counters

Design a synchronous mod-5 counter with the given state transition table.

Outline of method:
Design a synchronous mod-5 counter with the given state transition table.

Outline of method:

* State 1 → State 2 means
  - $Q_2$: 0 → 0,
  - $Q_1$: 0 → 0,
  - $Q_0$: 0 → 1.
Design a synchronous mod-5 counter with the given state transition table.

Outline of method:

* State 1 → State 2 means
  - $Q_2$: 0 → 0,
  - $Q_1$: 0 → 0,
  - $Q_0$: 0 → 1.

* Refer to the right table. For $Q_2$: 0 → 0, we must have $J_2 = 0$, $K_2 = X$, and so on.
Design a synchronous mod-5 counter with the given state transition table.

Outline of method:

* State 1 $\rightarrow$ State 2 means
  $Q_2$: 0 $\rightarrow$ 0,
  $Q_1$: 0 $\rightarrow$ 0,
  $Q_0$: 0 $\rightarrow$ 1.

* Refer to the right table. For $Q_2$: 0 $\rightarrow$ 0, we must have $J_2 = 0$, $K_2 = X$, and so on.

* When we cover all transitions in the left table, we have the truth tables for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, $K_2$ in terms of $Q_1$, $Q_2$, $Q_3$. 
Design a synchronous mod-5 counter with the given state transition table.

Outline of method:

* State 1 → State 2 means
  $Q_2$: 0 → 0,
  $Q_1$: 0 → 0,
  $Q_0$: 0 → 1.

* Refer to the right table. For $Q_2$: 0 → 0, we must have $J_2 = 0$, $K_2 = X$, and so on.

* When we cover all transitions in the left table, we have the truth tables for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, $K_2$ in terms of $Q_1$, $Q_2$, $Q_3$.

* The last step is to come up with suitable functions for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, $K_2$ in terms of $Q_1$, $Q_2$, $Q_3$. This can be done with K-maps. (If the number of flip-flops is more than 4, other techniques can be employed.)
We now have the truth tables for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, $K_2$ in terms of $Q_0$, $Q_1$, $Q_2$. The next step is to find logical functions for each of them. Note that we have not tabulated the $J$ and $K$ values for those combinations of $Q_0$, $Q_1$, $Q_2$ which do not occur in the state transition table (such as $Q_2 Q_1 Q_0 = 110$). We treat these as don’t care conditions (next slide).
We now have the truth tables for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, $K_2$ in terms of $Q_0$, $Q_1$, $Q_2$. The next step is to find logical functions for each of them.

Note that we have not tabulated the $J$ and $K$ values for those combinations of $Q_0$, $Q_1$, $Q_2$ which do not occur in the state transition table (such as $Q_2 Q_1 Q_0 = 110$). We treat these as don’t care conditions (next slide).

M. B. Patil, IIT Bombay
Design of synchronous counters

<table>
<thead>
<tr>
<th>state</th>
<th>Q_2</th>
<th>Q_1</th>
<th>Q_0</th>
<th>J_2</th>
<th>K_2</th>
<th>J_1</th>
<th>K_1</th>
<th>J_0</th>
<th>K_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLK</th>
<th>Q_n</th>
<th>Q_{n+1}</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>
Design of synchronous counters

We now have the truth tables for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, $K_2$ in terms of $Q_0$, $Q_1$, $Q_2$. The next step is to find logical functions for each of them.

Note that we have not tabulated the $J$ and $K$ values for those combinations of $Q_0$, $Q_1$, $Q_2$ which do not occur in the state transition table (such as $Q_2$ $Q_1$ $Q_0$ = 110). We treat these as don’t care conditions (next slide).

<table>
<thead>
<tr>
<th>state</th>
<th>$Q_2$</th>
<th>$Q_1$</th>
<th>$Q_0$</th>
<th>$J_2$</th>
<th>$K_2$</th>
<th>$J_1$</th>
<th>$K_1$</th>
<th>$J_0$</th>
<th>$K_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLK</th>
<th>$Q_n$</th>
<th>$Q_{n+1}$</th>
<th>$J$</th>
<th>$K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>
Design of synchronous counters

<table>
<thead>
<tr>
<th>state</th>
<th>Q₂</th>
<th>Q₁</th>
<th>Q₀</th>
<th>J₂</th>
<th>K₂</th>
<th>J₁</th>
<th>K₁</th>
<th>J₀</th>
<th>K₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We now have the truth tables for \(J_0, K_0, J_1, K_1, J_2, K_2\) in terms of \(Q_0, Q_1, Q_2\). The next step is to find logical functions for each of them. Note that we have not tabulated the \(J\) and \(K\) values for those combinations of \(Q_0, Q_1, Q_2\) which do not occur in the state transition table (such as \(Q_2 Q_1 Q_0 = 110\)). We treat these as don’t care conditions (next slide).
### Design of synchronous counters

#### Truth Tables

<table>
<thead>
<tr>
<th>State</th>
<th>$Q_2$</th>
<th>$Q_1$</th>
<th>$Q_0$</th>
<th>$J_2$</th>
<th>$K_2$</th>
<th>$J_1$</th>
<th>$K_1$</th>
<th>$J_0$</th>
<th>$K_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

#### Transition Table

<table>
<thead>
<tr>
<th>CLK</th>
<th>$Q_n$</th>
<th>$Q_{n+1}$</th>
<th>$J$</th>
<th>$K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>
Design of synchronous counters

We now have the truth tables for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, $K_2$ in terms of $Q_0$, $Q_1$, $Q_2$. The next step is to find logical functions for each of them. Note that we have not tabulated the $J$ and $K$ values for those combinations of $Q_0$, $Q_1$, $Q_2$ which do not occur in the state transition table (such as $Q_2 Q_1 Q_0 = 110$). We treat these as don't care conditions (next slide).
Design of synchronous counters

<table>
<thead>
<tr>
<th>state</th>
<th>Q_2</th>
<th>Q_1</th>
<th>Q_0</th>
<th>J_2</th>
<th>K_2</th>
<th>J_1</th>
<th>K_1</th>
<th>J_0</th>
<th>K_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLK</th>
<th>Q_n</th>
<th>Q_{n+1}</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>
Design of synchronous counters

We now have the truth tables for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, $K_2$ in terms of $Q_0$, $Q_1$, $Q_2$. The next step is to find logical functions for each of them.

Note that we have not tabulated the $J$ and $K$ values for those combinations of $Q_0$, $Q_1$, $Q_2$ which do not occur in the state transition table (such as $Q_2 Q_1 Q_0 = 110$). We treat these as don't care conditions (next slide).
We now have the truth tables for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, $K_2$ in terms of $Q_0$, $Q_1$, $Q_2$. The next step is to find logical functions for each of them.

Note that we have not tabulated the $J$ and $K$ values for those combinations of $Q_0$, $Q_1$, $Q_2$ which do not occur in the state transition table (such as $Q_2 Q_1 Q_0 = 110$). We treat these as don’t care conditions (next slide).
Design of synchronous counters

We now have the truth tables for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, $K_2$ in terms of $Q_0$, $Q_1$, $Q_2$. The next step is to find logical functions for each of them.

Note that we have not tabulated the $J$ and $K$ values for those combinations of $Q_0$, $Q_1$, $Q_2$ which do not occur in the state transition table (such as $Q_2 Q_1 Q_0 = 110$). We treat these as don't care conditions (next slide).

---

<table>
<thead>
<tr>
<th>state</th>
<th>$Q_2$</th>
<th>$Q_1$</th>
<th>$Q_0$</th>
<th>$J_2$</th>
<th>$K_2$</th>
<th>$J_1$</th>
<th>$K_1$</th>
<th>$J_0$</th>
<th>$K_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLK</th>
<th>$Q_n$</th>
<th>$Q_{n+1}$</th>
<th>$J$</th>
<th>$K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>
Design of synchronous counters

<table>
<thead>
<tr>
<th>state</th>
<th>Q2</th>
<th>Q1</th>
<th>Q0</th>
<th>J2</th>
<th>K2</th>
<th>J1</th>
<th>K1</th>
<th>J0</th>
<th>K0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We now have the truth tables for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, $K_2$ in terms of $Q_0$, $Q_1$, $Q_2$. The next step is to find logical functions for each of them. Note that we have not tabulated the $J$ and $K$ values for those combinations of $Q_0$, $Q_1$, $Q_2$ which do not occur in the state transition table (such as $Q_2Q_1Q_0 = 110$). We treat these as don’t care conditions (next slide).

<table>
<thead>
<tr>
<th>CLK</th>
<th>Qn</th>
<th>Qn+1</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>

M. B. Patil, IIT Bombay
Design of synchronous counters

<table>
<thead>
<tr>
<th>state</th>
<th>Q₂</th>
<th>Q₁</th>
<th>Q₀</th>
<th>J₂</th>
<th>K₂</th>
<th>J₁</th>
<th>K₁</th>
<th>J₀</th>
<th>K₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLK</th>
<th>Q₀</th>
<th>Q₁</th>
<th>Q₂</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Design of synchronous counters

<table>
<thead>
<tr>
<th>State</th>
<th>Q_2</th>
<th>Q_1</th>
<th>Q_0</th>
<th>J_2</th>
<th>K_2</th>
<th>J_1</th>
<th>K_1</th>
<th>J_0</th>
<th>K_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We now have the truth tables for J_0, K_0, J_1, K_1, J_2, K_2 in terms of Q_0, Q_1, Q_2. The next step is to find logical functions for each of them.

Note that we have not tabulated the J and K values for those combinations of Q_0, Q_1, Q_2 which do not occur in the state transition table (such as Q_2 Q_1 Q_0 = 110). We treat these as don’t care conditions (next slide).

<table>
<thead>
<tr>
<th>CLK</th>
<th>Q_n</th>
<th>Q_{n+1}</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>†</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>†</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>†</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>†</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>
Design of synchronous counters

<table>
<thead>
<tr>
<th>state</th>
<th>Q₂</th>
<th>Q₁</th>
<th>Q₀</th>
<th>J₂ K₂</th>
<th>J₁ K₁</th>
<th>J₀ K₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1 X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We now have the truth tables for \( J_0 \), \( K_0 \), \( J_1 \), \( K_1 \), \( J_2 \), \( K_2 \) in terms of \( Q_0 \), \( Q_1 \), \( Q_2 \). The next step is to find logical functions for each of them.

Note that we have not tabulated the \( J \) and \( K \) values for those combinations of \( Q_0 \), \( Q_1 \), \( Q_2 \) which do not occur in the state transition table (such as \( Q_2 Q_1 Q_0 = 110 \)). We treat these as don’t care conditions (next slide).
### Design of synchronous counters

<table>
<thead>
<tr>
<th>state</th>
<th>Q₂</th>
<th>Q₁</th>
<th>Q₀</th>
<th>J₂</th>
<th>K₂</th>
<th>J₁</th>
<th>K₁</th>
<th>J₀</th>
<th>K₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>X</td>
</tr>
</tbody>
</table>

We now have the truth tables for J₀, K₀, J₁, K₁, J₂, K₂ in terms of Q₀, Q₁, Q₂. The next step is to find logical functions for each of them. Note that we have not tabulated the J and K values for those combinations of Q₀, Q₁, Q₂ which do not occur in the state transition table (such as Q₂ Q₁ Q₀ = 110). We treat these as don’t care conditions (next slide).
We now have the truth tables for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, $K_2$ in terms of $Q_0$, $Q_1$, $Q_2$. The next step is to find logical functions for each of them.
### Design of synchronous counters

#### State Transition Table

<table>
<thead>
<tr>
<th>State</th>
<th>$Q_2$</th>
<th>$Q_1$</th>
<th>$Q_0$</th>
<th>$J_2$</th>
<th>$K_2$</th>
<th>$J_1$</th>
<th>$K_1$</th>
<th>$J_0$</th>
<th>$K_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>$X$</td>
<td>1</td>
<td>$X$</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>$X$</td>
<td>$X$</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$X$</td>
<td>$X$</td>
<td>0</td>
<td>1</td>
<td>$X$</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$X$</td>
<td>$X$</td>
<td>1</td>
<td>$X$</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$X$</td>
<td>$X$</td>
<td>1</td>
<td>$X$</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$X$</td>
<td>$X$</td>
<td>1</td>
<td>$X$</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Transition Table

<table>
<thead>
<tr>
<th>CLK</th>
<th>$Q_n$</th>
<th>$Q_{n+1}$</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>$X$</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>$X$</td>
<td>0</td>
</tr>
</tbody>
</table>
Design of synchronous counters

We now have the truth tables for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, $K_2$ in terms of $Q_0$, $Q_1$, $Q_2$. The next step is to find logical functions for each of them. Note that we have not tabulated the $J$ and $K$ values for those combinations of $Q_0$, $Q_1$, $Q_2$ which do not occur in the state transition table (such as $Q_2Q_1Q_0 = 110$). We treat these as don’t care conditions (next slide).

<table>
<thead>
<tr>
<th>state</th>
<th>$Q_2$</th>
<th>$Q_1$</th>
<th>$Q_0$</th>
<th>$J_2$</th>
<th>$K_2$</th>
<th>$J_1$</th>
<th>$K_1$</th>
<th>$J_0$</th>
<th>$K_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLK</th>
<th>$Q_n$</th>
<th>$Q_{n+1}$</th>
<th>$J$</th>
<th>$K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>
Design of synchronous counters

<table>
<thead>
<tr>
<th>state</th>
<th>Q₂</th>
<th>Q₁</th>
<th>Q₀</th>
<th>J₂</th>
<th>K₂</th>
<th>J₁</th>
<th>K₁</th>
<th>J₀</th>
<th>K₀</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

We now have the truth tables for J₀, K₀, J₁, K₁, J₂, K₂ in terms of Q₀, Q₁, Q₂. The next step is to find logical functions for each of them. Note that we have not tabulated the J and K values for those combinations of Q₀, Q₁, Q₂ which do not occur in the state transition table (such as Q₂Q₁Q₀ = 110). We treat these as don’t care conditions (next slide).

<table>
<thead>
<tr>
<th>CLK</th>
<th>Qₙ</th>
<th>Qₙ₊₁</th>
<th>J</th>
<th>K</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>
Design of synchronous counters

We now have the truth tables for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, $K_2$ in terms of $Q_0$, $Q_1$, $Q_2$. The next step is to find logical functions for each of them. Note that we have not tabulated the $J$ and $K$ values for those combinations of $Q_0$, $Q_1$, $Q_2$ which do not occur in the state transition table (such as $Q_2 Q_1 Q_0 = 110$). We treat these as don't care conditions (next slide).
Design of synchronous counters

We now have the truth tables for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, $K_2$ in terms of $Q_0$, $Q_1$, $Q_2$. The next step is to find logical functions for each of them.
We now have the truth tables for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, $K_2$ in terms of $Q_0$, $Q_1$, $Q_2$. The next step is to find logical functions for each of them.

* Note that we have not tabulated the $J$ and $K$ values for those combinations of $Q_0$, $Q_1$, $Q_2$ which do not occur in the state transition table (such as $Q_2Q_1Q_0 = 110$). We treat these as don’t care conditions (next slide).
Design of synchronous counters

We treat the unused states (Q₂Q₁Q₀ = 101, 110, 111) as (additional) don’t care conditions. Since these are different from the don’t care conditions arising from the state transition table, we mark them with a different colour.

We will assume that a suitable initialization facility is provided to ensure that the counter starts up in one of the five allowed states (say, Q₂Q₁Q₀ = 000).

From the K-maps, J₂ = Q₁Q₀, K₂ = 1, J₁ = Q₀, K₁ = Q₀, J₀ = Q₂, K₀ = 1.

M. B. Patil, IIT Bombay
We treat the unused states \((Q_2Q_1Q_0 = 101, 110, 111)\) as (additional) don’t care conditions. Since these are different from the don’t care conditions arising from the state transition table, we mark them with a different colour.
We treat the unused states \((Q_2Q_1Q_0 = 101, 110, 111)\) as (additional) don’t care conditions. Since these are different from the don’t care conditions arising from the state transition table, we mark them with a different colour.

We will assume that a suitable initialization facility is provided to ensure that the counter starts up in one of the five allowed states (say, \(Q_2Q_1Q_0 = 000\)).
Design of synchronous counters

<table>
<thead>
<tr>
<th>state</th>
<th>Q_2</th>
<th>Q_1</th>
<th>Q_0</th>
<th>J_2</th>
<th>K_2</th>
<th>J_1</th>
<th>K_1</th>
<th>J_0</th>
<th>K_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

* We treat the unused states \((Q_2 Q_1 Q_0 = 101, 110, 111)\) as (additional) don’t care conditions. Since these are different from the don’t care conditions arising from the state transition table, we mark them with a different colour.

* We will assume that a suitable initialization facility is provided to ensure that the counter starts up in one of the five allowed states (say, \(Q_2 Q_1 Q_0 = 000\)).

* From the K-maps, \(J_2 = Q_1 Q_0, K_2 = 1, J_1 = Q_0, K_1 = Q_0, J_0 = \overline{Q_2}, K_0 = 1\).
Design of synchronous counters: verification

*Note that the design is independent of whether positive or negative edge-triggered flip-flops are used.*

M. B. Patil, IIT Bombay
Design of synchronous counters: verification

\* \( J_2 = Q_1 Q_0, \ K_2 = 1, \ J_1 = Q_0, \ K_1 = Q_0, \ J_0 = \overline{Q}_2, \ K_0 = 1. \)
* $J_2 = Q_1 Q_0$, $K_2 = 1$, $J_1 = Q_0$, $K_1 = Q_0$, $J_0 = \overline{Q_2}$, $K_0 = 1$.

* Note that the design is independent of whether positive or negative edge-triggered flip-flops are used.
Consider two counters, Counter 1 (mod-\(k_1\)) and Counter 2 (mod-\(k_2\)). (Each of them can be ripple or synchronous type.)
Consider two counters, Counter 1 (mod-$k_1$) and Counter 2 (mod-$k_2$). (Each of them can be ripple or synchronous type.)

Since Counter 1 has $k_1$ states and Counter 2 has $k_2$ states, we can get a new counter with $k_1 k_2$ states if appropriate synchronisation is provided between the two clocks.
Consider two counters, Counter 1 (mod-$k_1$) and Counter 2 (mod-$k_2$). (Each of them can be ripple or synchronous type.)

Since Counter 1 has $k_1$ states and Counter 2 has $k_2$ states, we can get a new counter with $k_1k_2$ states if appropriate synchronisation is provided between the two clocks.

There are two ways of providing synchronisation:
Consider two counters, Counter 1 (mod-$k_1$) and Counter 2 (mod-$k_2$). (Each of them can be ripple or synchronous type.)

Since Counter 1 has $k_1$ states and Counter 2 has $k_2$ states, we can get a new counter with $k_1k_2$ states if appropriate synchronisation is provided between the two clocks.

There are two ways of providing synchronisation:
- derive Clock 2 from Clock 1 (using some decoding logic, if necessary)
Consider two counters, Counter 1 (mod-$k_1$) and Counter 2 (mod-$k_2$). (Each of them can be ripple or synchronous type.)

Since Counter 1 has $k_1$ states and Counter 2 has $k_2$ states, we can get a new counter with $k_1 k_2$ states if appropriate synchronisation is provided between the two clocks.

There are two ways of providing synchronisation:

- derive Clock 2 from Clock 1 (using some decoding logic, if necessary)
* Consider two counters, Counter 1 (mod-$k_1$) and Counter 2 (mod-$k_2$). (Each of them can be ripple or synchronous type.)
* Since Counter 1 has $k_1$ states and Counter 2 has $k_2$ states, we can get a new counter with $k_1 k_2$ states if appropriate synchronisation is provided between the two clocks.
* There are two ways of providing synchronisation:
  - derive Clock 2 from Clock 1 (using some decoding logic, if necessary)
  - drive the two counters with the same clock
Consider two counters, Counter 1 (mod-$k_1$) and Counter 2 (mod-$k_2$). (Each of them can be ripple or synchronous type.)

Since Counter 1 has $k_1$ states and Counter 2 has $k_2$ states, we can get a new counter with $k_1k_2$ states if appropriate synchronisation is provided between the two clocks.

There are two ways of providing synchronisation:
- derive Clock 2 from Clock 1 (using some decoding logic, if necessary)
- drive the two counters with the same clock
Let us combine the mod-2 and mod-5 counters to make a mod-10 counter.

We will follow two approaches (as described earlier):

A: The clock for the second (mod-5) counter is derived from the first (mod-2) counter.

B: A common clock is used to drive the mod-2 and mod-5 counters.

M. B. Patil, IIT Bombay
Let us combine the mod-2 and mod-5 counters to make a mod-10 counter.
Let us combine the mod-2 and mod-5 counters to make a mod-10 counter.

We will follow two approaches (as described earlier):

A: The clock for the second (mod-5) counter is derived from the first (mod-2) counter.

B: A common clock is used to drive the mod-2 and mod-5 counters.
Approach A

(SEQUEL file: ee101_counter_7.sqproj)

M. B. Patil, IIT Bombay
Approach B

(SEQUEL file: ee101_counter_8.sqproj)
* Show that, by connecting the $\overline{Q}$ output of the mod-2 counter (instead of the $Q$ output) to the clock input of the mod-5 counter in the ripple connection (“Approach A”) circuit, we get a decade counter, counting up from 0000 to 1001.
* Show that, by connecting the $\overline{Q}$ output of the mod-2 counter (instead of the $Q$ output) to the clock input of the mod-5 counter in the ripple connection (“Approach A”) circuit, we get a decade counter, counting up from 0000 to 1001.

* Derive appropriate decoding logic for each of the ten counters states (i.e., the output should be 1 for only that particular state and 0 otherwise).
* Show that, by connecting the $\overline{Q}$ output of the mod-2 counter (instead of the $Q$ output) to the clock input of the mod-5 counter in the ripple connection ("Approach A") circuit, we get a decade counter, counting up from 0000 to 1001.

* Derive appropriate decoding logic for each of the ten counters states (i.e., the output should be 1 for only that particular state and 0 otherwise).

* Derive appropriate decoding logic which will give a symmetrical square wave (i.e., a duty cycle of 50 %) with a frequency of $f_c/10$, where $f_c$ is the clock frequency.

M. B. Patil, IIT Bombay
Combination of counters

* Show that, by connecting the \( \overline{Q} \) output of the mod-2 counter (instead of the \( Q \) output) to the clock input of the mod-5 counter in the ripple connection ("Approach A") circuit, we get a decade counter, counting up from 0000 to 1001.

* Derive appropriate decoding logic for each of the ten counters states (i.e., the output should be 1 for only that particular state and 0 otherwise).

* Derive appropriate decoding logic which will give a symmetrical square wave (i.e., a duty cycle of 50 \%) with a frequency of \( f_c/10 \), where \( f_c \) is the clock frequency.

* Verify your design by simulation.