The digital circuits we have seen so far (gates, multiplexer, demultiplexer, encoders, decoders) are combinational in nature, i.e., the output(s) depends only on the present values of the inputs and not on their past values.

In sequential circuits, the “state” of the circuit is crucial in determining the output values. For a given input combination, a sequential circuit may produce different output values, depending on its previous state.

In other words, a sequential circuit has a memory (of its past state) whereas a combinational circuit has no memory.

Sequential circuits (together with combinational circuits) make it possible to build several useful applications, such as counters, registers, arithmetic/logic unit (ALU), all the way to microprocessors.
The digital circuits we have seen so far (gates, multiplexer, demultiplexer, encoders, decoders) are *combinatorial* in nature, i.e., the output(s) depends only on the *present* values of the inputs and *not* on their past values.
Sequential circuits

* The digital circuits we have seen so far (gates, multiplexer, demultiplexer, encoders, decoders) are combinatorial in nature, i.e., the output(s) depends only on the present values of the inputs and not on their past values.

* In sequential circuits, the “state” of the circuit is crucial in determining the output values. For a given input combination, a sequential circuit may produce different output values, depending on its previous state.
* The digital circuits we have seen so far (gates, multiplexer, demultiplexer, encoders, decoders) are *combinatorial* in nature, i.e., the output(s) depends only on the *present* values of the inputs and *not* on their past values.

* In *sequential* circuits, the “state” of the circuit is crucial in determining the output values. For a given input combination, a sequential circuit may produce different output values, depending on its previous state.

* In other words, a sequential circuit has a *memory* (of its past state) whereas a combinatorial circuit has no memory.
* The digital circuits we have seen so far (gates, multiplexer, demultiplexer, encoders, decoders) are *combinatorial* in nature, i.e., the output(s) depends only on the *present* values of the inputs and *not* on their past values.

* In *sequential* circuits, the “state” of the circuit is crucial in determining the output values. For a given input combination, a sequential circuit may produce different output values, depending on its previous state.

* In other words, a sequential circuit has a *memory* (of its past state) whereas a combinatorial circuit has no memory.

* Sequential circuits (together with combinatorial circuits) make it possible to build several useful applications, such as counters, registers, arithmetic/logic unit (ALU), all the way to microprocessors.
NAND latch (RS latch)

* $A, B$: inputs, $X_1, X_2$: outputs

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>$X_1$</th>
<th>$X_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*For $A = B = 0$, $X_1$ and $X_2$ are both 1. This combination of $A$ and $B$ is not allowed for reasons that will become clear later.*

M. B. Patil, IIT Bombay
NAND latch (RS latch)

* $A, B$: inputs, $X_1, X_2$: outputs
* Consider $A = 1, B = 0$.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X_1</th>
<th>X_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
- $A, B$: inputs, $X_1, X_2$: outputs
- Consider $A = 1, B = 0$. 
  $B = 0 \Rightarrow X_2 = 1$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>$X_1$</th>
<th>$X_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
* $A$, $B$: inputs, $X_1$, $X_2$: outputs

* Consider $A = 1$, $B = 0$.
  $B = 0 \Rightarrow X_2 = 1$
**NAND latch (RS latch)**

* $A, B$: inputs, $X_1, X_2$: outputs

* Consider $A = 1, B = 0$.

  $B = 0 \Rightarrow X_2 = 1 \Rightarrow X_1 = \overline{A}X_2 = \overline{1} \cdot 1 = 0$. 

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$X_1$</th>
<th>$X_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
NAND latch (RS latch)

* $A, B$: inputs, $X_1, X_2$: outputs

* Consider $A = 1, B = 0$.
  
  $B = 0 \Rightarrow X_2 = 1 \Rightarrow X_1 = \overline{A X_2} = \overline{1} \cdot 1 = 0$. 

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X_1</th>
<th>X_2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

M. B. Patil, IIT Bombay
NAND latch (RS latch)

A, B: inputs, X₁, X₂: outputs

Consider A = 1, B = 0.

\[ B = 0 \Rightarrow X_2 = 1 \Rightarrow X_1 = \overline{A \cdot X_2} = \overline{1} \cdot \overline{1} = 0. \]

Overall, we have X₁ = 0, X₂ = 1.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X₁</th>
<th>X₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
NAND latch (RS latch)

* $A, B$: inputs, $X_1, X_2$: outputs

* Consider $A = 1$, $B = 0$.
  
  $B = 0 \Rightarrow X_2 = 1 \Rightarrow X_1 = \overline{A} X_2 = \overline{1} \cdot 1 = 0$.

  Overall, we have $X_1 = 0$, $X_2 = 1$.

* Consider $A = 0$, $B = 1$. 

<table>
<thead>
<tr>
<th>$A$</th>
<th>$B$</th>
<th>$X_1$</th>
<th>$X_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

M. B. Patil, IIT Bombay
NAND latch (RS latch)

\[
\begin{array}{ccc}
\text{A} & 0 & 1 \\
\text{B} & 1 & 0 \\
\end{array}
\]

\[
\begin{array}{ccc}
\text{X}_1 & \text{X}_2 \\
0 & 1 \\
1 & 0 \\
1 & 1 \\
0 & 0 \\
\end{array}
\]

\begin{itemize}
  \item \textit{A, B: inputs, X}_1, \textit{X}_2: outputs
  \item Consider \(\text{A} = 1, \text{B} = 0\).
    \(\text{B} = 0 \Rightarrow \text{X}_2 = 1 \Rightarrow \text{X}_1 = \overline{\text{A} \text{X}_2} = 1 \cdot 1 = 0\).
    Overall, we have \(\text{X}_1 = 0, \text{X}_2 = 1\).
  \item Consider \(\text{A} = 0, \text{B} = 1\).
    \(\rightarrow \text{X}_1 = 1, \text{X}_2 = 0\).
\end{itemize}
A, B: inputs, X₁, X₂: outputs

Consider A = 1, B = 0.
B = 0 => X₂ = 1 => X₁ = \overline{A} \overline{X₂} = \overline{1} \cdot \overline{1} = 0.
Overall, we have X₁ = 0, X₂ = 1.

Consider A = 0, B = 1.
\rightarrow X₁ = 1, X₂ = 0.

Consider A = B = 1.
NAND latch (RS latch)

* $A, B$: inputs, $X_1, X_2$: outputs

* Consider $A = 1, B = 0$.
  
  $B = 0 \Rightarrow X_2 = 1 \Rightarrow X_1 = \overline{A X_2} = \overline{1} \cdot \overline{1} = 0$.  
  
  Overall, we have $X_1 = 0, X_2 = 1$.

* Consider $A = 0, B = 1$.
  
  $X_1 = 1, X_2 = 0$.

* Consider $A = B = 1$.
  
  $X_1 = \overline{A X_2} = \overline{X_2}, X_2 = \overline{B X_1} = \overline{X_1} \Rightarrow X_1 = X_2$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>$X_1$</th>
<th>$X_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A, B: inputs, X₁, X₂: outputs

Consider A = 1, B = 0.
B = 0 ⇒ X₂ = 1 ⇒ X₁ = \overline{A} \overline{X₂} = \overline{1} \overline{1} = 0.
Overall, we have X₁ = 0, X₂ = 1.

Consider A = 0, B = 1.
⇒ X₁ = 1, X₂ = 0.

Consider A = B = 1.
X₁ = \overline{A} \overline{X₂} = \overline{X₂}, X₂ = \overline{B} \overline{X₁} = \overline{X₁} ⇒ X₁ = X₂
NAND latch (RS latch)

* $A, B$: inputs, $X_1, X_2$: outputs

* Consider $A = 1, B = 0$.
  
  $B = 0 \Rightarrow X_2 = 1 \Rightarrow X_1 = \overline{A \overline{X_2}} = 1 \cdot 1 = 0$.

  Overall, we have $X_1 = 0, X_2 = 1$.

* Consider $A = 0, B = 1$.

  $\Rightarrow X_1 = 1, X_2 = 0$.

* Consider $A = B = 1$.

  $X_1 = \overline{A \overline{X_2}} = \overline{X_2}, \ X_2 = \overline{B \overline{X_1}} = \overline{X_1} \Rightarrow X_1 = X_2$

  If $X_1 = 1, X_2 = 0$ previously, the circuit continues to "hold" that state.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>$X_1$</th>
<th>$X_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**NAND latch (RS latch)**

\[
\begin{array}{c}
\text{A} \\
\text{B}
\end{array}
\begin{array}{c}
\text{X}_2 \\
\text{X}_1
\end{array}
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X₁</th>
<th>X₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* A, B: inputs, X₁, X₂: outputs

* Consider A = 1, B = 0.
  \[ B = 0 \Rightarrow X_2 = 1 \Rightarrow X_1 = \overline{A} \overline{X_2} = \overline{1} \cdot 1 = 0. \]
  Overall, we have \( X_1 = 0, X_2 = 1 \).

* Consider A = 0, B = 1.
  \[ \rightarrow X_1 = 1, X_2 = 0. \]

* Consider A = B = 1.
  \[ X_1 = \overline{A} \overline{X_2} = \overline{X_2}, \quad X_2 = \overline{B} X_1 = X_1 \Rightarrow \boxed{X_1 = X_2} \]

If \( X_1 = 1, X_2 = 0 \) previously, the circuit continues to “hold” that state. Similarly, if \( X_1 = 0, X_2 = 1 \) previously, the circuit continues to “hold” that state.
NAND latch (RS latch)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X₁</th>
<th>X₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* A, B: inputs, X₁, X₂: outputs

* Consider A = 1, B = 0.
  
  \[ B = 0 \Rightarrow X₂ = 1 \Rightarrow X₁ = \overline{A X₂} = \overline{1} \cdot 1 = 0. \]
  
  Overall, we have X₁ = 0, X₂ = 1.

* Consider A = 0, B = 1.
  
  \[ \Rightarrow X₁ = 1, X₂ = 0. \]

* Consider A = B = 1.
  
  \[ X₁ = \overline{A X₂} = X₂, \ X₂ = \overline{B X₁} = \overline{X₁} \Rightarrow X₁ = X₂ \]
  
  If X₁ = 1, X₂ = 0 previously, the circuit continues to “hold” that state. Similarly, if X₁ = 0, X₂ = 1 previously, the circuit continues to “hold” that state.

The circuit has “latched in” the previous state.
NAND latch (RS latch)

\[
\text{A, B: inputs, } X_1, X_2: \text{ outputs}
\]

* Consider \( A = 1, B = 0. \)
  \( B = 0 \Rightarrow X_2 = 1 \Rightarrow X_1 = \overline{A}X_2 = 1 \cdot 1 = 0. \)
  Overall, we have \( X_1 = 0, X_2 = 1. \)

* Consider \( A = 0, B = 1. \)
  \( \rightarrow X_1 = 1, X_2 = 0. \)

* Consider \( A = B = 1. \)
  \( X_1 = \overline{A}X_2 = \overline{X_2}, X_2 = \overline{B}X_1 = X_1 \Rightarrow \boxed{X_1 = X_2} \)
  If \( X_1 = 1, X_2 = 0 \) previously, the circuit continues to “hold” that state.
  Similarly, if \( X_1 = 0, X_2 = 1 \) previously, the circuit continues to “hold” that state.
  The circuit has “latched in” the previous state.

* For \( A = B = 0, X_1 \) and \( X_2 \) are both 1. This combination of \( A \) and \( B \) is \textit{not} allowed for reasons that will become clear later.
A, B: inputs, X₁, X₂: outputs

Consider \( A = 1, B = 0 \).
\[ B = 0 \Rightarrow X₂ = 1 \Rightarrow X₁ = \overline{A} \overline{X₂} = \overline{1} \cdot \overline{1} = 0. \]
Overall, we have \( X₁ = 0, X₂ = 1 \).

Consider \( A = 0, B = 1 \).
\[ \rightarrow X₁ = 1, X₂ = 0. \]

Consider \( A = B = 1 \).
\[ X₁ = \overline{A} \overline{X₂} = \overline{X₂}, \quad X₂ = \overline{B} \overline{X₁} = X₁ \Rightarrow X₁ = X₂ \]
If \( X₁ = 1, X₂ = 0 \) previously, the circuit continues to “hold” that state.
Similarly, if \( X₁ = 0, X₂ = 1 \) previously, the circuit continues to “hold” that state.
The circuit has “latched in” the previous state.

For \( A = B = 0 \), \( X₁ \) and \( X₂ \) are both 1. This combination of \( A \) and \( B \) is not allowed for reasons that will become clear later.
The combination $A = 1, B = 0$ serves to reset $X_1$ to 0 (irrespective of the previous state of the latch).

The combination $A = 0, B = 1$ serves to set $X_1$ to 1 (irrespective of the previous state of the latch).

In other words, $A = 1, B = 0 \rightarrow$ latch gets reset to 0. $A = 0, B = 1 \rightarrow$ latch gets set to 1.

The $A$ input is therefore called the RESET (R) input, and $B$ is called the SET (S) input of the latch.

$X_1$ is denoted by $Q$, and $X_2$ (which is $X_1$ in all cases except for $A = B = 0$) is denoted by $Q$. 

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>$X_1$</th>
<th>$X_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>
* The combination $A = 1, B = 0$ serves to reset $X_1$ to $0$ (irrespective of the previous state of the latch).

\[
\begin{array}{c|cc}
A & B & X_1 & X_2 \\
\hline
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 \\
1 & 1 & \text{previous} & \text{valid} \\
0 & 0 & \text{invalid} & \text{valid}
\end{array}
\]
NAND latch (RS latch)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>X₁</th>
<th>X₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

* The combination \( A = 1, B = 0 \) serves to reset \( X_1 \) to 0 (irrespective of the previous state of the latch).
* The combination \( A = 0, B = 1 \) serves to set \( X_1 \) to 1 (irrespective of the previous state of the latch).
The combination $A = 1, B = 0$ serves to **reset** $X_1$ to 0 (**irrespective of** the previous state of the latch).

The combination $A = 0, B = 1$ serves to **set** $X_1$ to 1 (**irrespective of** the previous state of the latch).

In other words,

- $A = 1, B = 0 \rightarrow$ latch gets reset to 0.
- $A = 0, B = 1 \rightarrow$ latch gets set to 1.
* The combination $A = 1, B = 0$ serves to reset $X_1$ to 0 (irrespective of the previous state of the latch).
* The combination $A = 0, B = 1$ serves to set $X_1$ to 1 (irrespective of the previous state of the latch).
* In other words, $A = 1, B = 0 \rightarrow$ latch gets reset to 0.
  $A = 0, B = 1 \rightarrow$ latch gets set to 1.
* The $A$ input is therefore called the RESET (R) input, and $B$ is called the SET (S) input of the latch.

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>$X_1$</th>
<th>$X_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>
The combination $A = 1, B = 0$ serves to reset $X_1$ to 0 (irrespective of the previous state of the latch).

The combination $A = 0, B = 1$ serves to set $X_1$ to 1 (irrespective of the previous state of the latch).

In other words,

$A = 1, B = 0 \rightarrow$ latch gets reset to 0.

$A = 0, B = 1 \rightarrow$ latch gets set to 1.

The $A$ input is therefore called the RESET (R) input, and $B$ is called the SET (S) input of the latch.

$X_1$ is denoted by $Q$, and $X_2$ (which is $\overline{X_1}$ in all cases except for $A = B = 0$) is denoted by $\overline{Q}$.
* The combination $A = 1$, $B = 0$ serves to reset $X_1$ to 0 (irrespective of the previous state of the latch).
* The combination $A = 0$, $B = 1$ serves to set $X_1$ to 1 (irrespective of the previous state of the latch).
* In other words,
  
  $A = 1$, $B = 0 \rightarrow$ latch gets reset to 0.
  $A = 0$, $B = 1 \rightarrow$ latch gets set to 1.

* The $A$ input is therefore called the RESET (R) input, and $B$ is called the SET (S) input of the latch.
* $X_1$ is denoted by $Q$, and $X_2$ (which is $\overline{X_1}$ in all cases except for $A = B = 0$) is denoted by $\overline{Q}$.
NAND latch (RS latch)

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>(\overline{Q})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

*Up to \(t = t_1\), \(R = 0\), \(S = 1\) → \(Q = 1\).
*At \(t = t_1\), \(R\) goes high → \(R = S = 1\), and the latch holds its previous state → no change at the output.
*At \(t = t_2\), \(S\) goes low → \(R = 1\), \(S = 0\) → \(Q = 0\).
*At \(t = t_3\), \(S\) goes high → \(R = S = 1\), and the latch holds its previous state → no change at the output.

M. B. Patil, IIT Bombay
### NAND latch (RS latch)

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>( \overline{Q} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td>( \overline{Q} )</td>
</tr>
</tbody>
</table>

* Up to \( t = t_1 \), \( R = 0, S = 1 \) → \( Q = 1 \).
NAND latch (RS latch)

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>( \overline{Q} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

* Up to \( t = t_1 \), \( R = 0, S = 1 \) \( \rightarrow \) \( Q = 1 \).
* At \( t = t_1 \), \( R \) goes high \( \rightarrow \) \( R = S = 1 \), and the latch holds its previous state \( \rightarrow \) no change at the output.
NAND latch (RS latch)

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>\overline{Q}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

* Up to \( t = t_1 \), \( R = 0, S = 1 \rightarrow Q = 1 \).

* At \( t = t_1 \), \( R \) goes high \( \rightarrow R = S = 1 \), and the latch holds its previous state \( \rightarrow \) no change at the output.

* At \( t = t_2 \), \( S \) goes low \( \rightarrow R = 1, S = 0 \rightarrow Q = 0 \).
NAND latch (RS latch)

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>( \overline{Q} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>( \text{previous} )</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

* Up to \( t = t_1 \), \( R = 0, S = 1 \) \( \rightarrow \) \( Q = 1 \).
* At \( t = t_1 \), \( R \) goes high \( \rightarrow \) \( R = S = 1 \), and the latch holds its previous state \( \rightarrow \) no change at the output.
* At \( t = t_2 \), \( S \) goes low \( \rightarrow \) \( R = 1, S = 0 \) \( \rightarrow \) \( Q = 0 \).
* At \( t = t_3 \), \( S \) goes high \( \rightarrow \) \( R = S = 1 \), and the latch holds its previous state \( \rightarrow \) no change at the output.
Why not allow $R = S = 0$?

- It makes $Q = \overline{Q} = 1$, i.e., $Q$ and $\overline{Q}$ are not inverse of each other any more.
- More importantly, when $R$ and $S$ both become 1 simultaneously (starting from $R = S = 0$), the final outputs $Q$ and $\overline{Q}$ cannot be uniquely determined. We could have $Q = 0$, $\overline{Q} = 1$ or $Q = 1$, $\overline{Q} = 0$, depending on the delays associated with the two NAND gates.
Why not allow $R = S = 0$?

- It makes $Q = \overline{Q} = 1$, i.e., $Q$ and $\overline{Q}$ are not inverse of each other any more.
- More importantly, when $R$ and $S$ both become 1 simultaneously (starting from $R = S = 0$), the final outputs $Q$ and $\overline{Q}$ cannot be uniquely determined. We could have $Q = 0$, $\overline{Q} = 1$ or $Q = 1$, $\overline{Q} = 0$, depending on the delays associated with the two NAND gates.
Why not allow \( R = S = 0 \)?

- It makes \( Q = \overline{Q} = 1 \), i.e., \( Q \) and \( \overline{Q} \) are not inverse of each other any more.
Why not allow $R = S = 0$?
- It makes $Q = \overline{Q} = 1$, i.e., $Q$ and $\overline{Q}$ are not inverse of each other any more.
- More importantly, when $R$ and $S$ both become 1 simultaneously (starting from $R = S = 0$), the final outputs $Q$ and $\overline{Q}$ cannot be uniquely determined. We could have $Q = 0$, $\overline{Q} = 1$ or $Q = 1$, $\overline{Q} = 0$, depending on the delays associated with the two NAND gates.
Why not allow $R = S = 0$?

- It makes $Q = \overline{Q} = 1$, i.e., $Q$ and $\overline{Q}$ are not inverse of each other any more.

- More importantly, when $R$ and $S$ both become 1 simultaneously (starting from $R = S = 0$), the final outputs $Q$ and $\overline{Q}$ cannot be uniquely determined. We could have $Q = 0$, $\overline{Q} = 1$ or $Q = 1$, $\overline{Q} = 0$, depending on the delays associated with the two NAND gates.
Why not allow \( R = S = 0 \)?

- It makes \( Q = \overline{Q} = 1 \), i.e., \( Q \) and \( \overline{Q} \) are not inverse of each other any more.

- More importantly, when \( R \) and \( S \) both become 1 simultaneously (starting from \( R = S = 0 \)), the final outputs \( Q \) and \( \overline{Q} \) cannot be uniquely determined. We could have \( Q = 0, \overline{Q} = 1 \) or \( Q = 1, \overline{Q} = 0 \), depending on the delays associated with the two NAND gates.
Why not allow $R = S = 0$?
- It makes $Q = \overline{Q} = 1$, i.e., $Q$ and $\overline{Q}$ are not inverse of each other any more.
- More importantly, when $R$ and $S$ both become 1 simultaneously (starting from $R = S = 0$), the final outputs $Q$ and $\overline{Q}$ cannot be uniquely determined. We could have $Q = 0$, $\overline{Q} = 1$ or $Q = 1$, $\overline{Q} = 0$, depending on the delays associated with the two NAND gates.
Why not allow \( R = S = 0 \)?

- It makes \( Q = \overline{Q} = 1 \), i.e., \( Q \) and \( \overline{Q} \) are not inverse of each other any more.
- More importantly, when \( R \) and \( S \) both become 1 simultaneously (starting from \( R = S = 0 \)), the final outputs \( Q \) and \( \overline{Q} \) cannot be uniquely determined. We could have \( Q = 0, \overline{Q} = 1 \) or \( Q = 1, \overline{Q} = 0 \), depending on the delays associated with the two NAND gates.

### Truth Table

<table>
<thead>
<tr>
<th>( R )</th>
<th>( S )</th>
<th>( Q )</th>
<th>( \overline{Q} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td>invalid</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Why not allow $R = S = 0$?
- It makes $Q = \overline{Q} = 1$, i.e., $Q$ and $\overline{Q}$ are not inverse of each other any more.
- More importantly, when $R$ and $S$ both become 1 simultaneously (starting from $R = S = 0$), the final outputs $Q$ and $\overline{Q}$ cannot be uniquely determined. We could have $Q = 0$, $\overline{Q} = 1$ or $Q = 1$, $\overline{Q} = 0$, depending on the delays associated with the two NAND gates.
The NOR latch is similar to the NAND latch:

- When $R = 1$, $S = 0$, the latch gets reset to $Q = 0$.
- When $R = 0$, $S = 1$, the latch gets set to $Q = 1$.
- For $R = S = 0$, the latch retains its previous state (i.e., the previous values of $Q$ and $\overline{Q}$).
- $R = S = 1$ is not allowed for reasons similar to those discussed in the context of the NAND latch.

\[
\begin{array}{|c|c|c|c|}
\hline
R & S & Q & \overline{Q} \\
\hline
1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 \\
0 & 0 & \text{previous} & \\
1 & 1 & \text{invalid} & \\
\hline
\end{array}
\]
The NOR latch is similar to the NAND latch:
When $R = 1$, $S = 0$, the latch gets reset to $Q = 0$.
When $R = 0$, $S = 1$, the latch gets set to $Q = 1$.

* The NOR latch is similar to the NAND latch:
When $R = 1$, $S = 0$, the latch gets reset to $Q = 0$.
When $R = 0$, $S = 1$, the latch gets set to $Q = 1$. 

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>$\overline{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>


M. B. Patil, IIT Bombay
* The NOR latch is similar to the NAND latch:
  When \( R = 1, \) \( S = 0 \), the latch gets reset to \( Q = 0 \).
  When \( R = 0, \) \( S = 1 \), the latch gets set to \( Q = 1 \).

* For \( R = S = 0 \), the latch retains its previous state (i.e., the previous values of \( Q \) and \( \overline{Q} \)).
The NOR latch is similar to the NAND latch:

When $R = 1$, $S = 0$, the latch gets reset to $Q = 0$.
When $R = 0$, $S = 1$, the latch gets set to $Q = 1$.

For $R = S = 0$, the latch retains its previous state (i.e., the previous values of $Q$ and $\overline{Q}$).

$R = S = 1$ is not allowed for reasons similar to those discussed in the context of the NAND latch.
Comparison of NAND and NOR latches

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>Q̅</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>Q̅</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>
**NAND latch: alternative node names**

Active low input nodes:

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>̅Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

![NAND latch diagram]
NAND latch: alternative node names

Active low input nodes:

<table>
<thead>
<tr>
<th>$\bar{R}$</th>
<th>$\bar{S}$</th>
<th>$Q$</th>
<th>$\bar{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

Active high input nodes:
Chatter (bouncing) due to a mechanical switch

When the switch is thrown from A to B, $V_o$ is expected to go from 0 V to $V_s$ (say, 5 V). However, mechanical switches suffer from "chatter" or "bouncing," i.e., the transition from A to B is not a single, clean one. As a result, $V_o$ oscillates between 0 V and 5 V before settling to its final value (5 V).

In some applications, this chatter can cause malfunction → need a way to remove the chatter.

M. B. Patil, IIT Bombay
When the switch is thrown from A to B, $V_o$ is expected to go from 0 V to $V_s$ (say, 5 V). However, mechanical switches suffer from “chatter” or “bouncing,” i.e., the transition from A to B is not a single, clean one. As a result, $V_o$ oscillates between 0 V and 5 V before settling to its final value (5 V).
Chatter (bouncing) due to a mechanical switch

When the switch is thrown from A to B, $V_o$ is expected to go from 0 V to $V_s$ (say, 5 V). However, mechanical switches suffer from "chatter" or "bouncing," i.e., the transition from A to B is not a single, clean one. As a result, $V_o$ oscillates between 0 V and 5 V before settling to its final value (5 V).

In some applications, this chatter can cause malfunction → need a way to remove the chatter.

M. B. Patil, IIT Bombay
* When the switch is thrown from A to B, $V_o$ is expected to go from 0 V to $V_s$ (say, 5 V).

* However, mechanical switches suffer from “chatter” or “bouncing,” i.e., the transition from A to B is not a single, clean one. As a result, $V_o$ oscillates between 0 V and 5 V before settling to its final value (5 V).
When the switch is thrown from A to B, $V_o$ is expected to go from 0 V to $V_s$ (say, 5 V).

However, mechanical switches suffer from “chatter” or “bouncing,” i.e., the transition from A to B is not a single, clean one. As a result, $V_o$ oscillates between 0 V and 5 V before settling to its final value (5 V).
* When the switch is thrown from A to B, $V_o$ is expected to go from 0 V to $V_s$ (say, 5 V).

* However, mechanical switches suffer from “chatter” or “bouncing,” i.e., the transition from A to B is not a single, clean one. As a result, $V_o$ oscillates between 0 V and 5 V before settling to its final value (5 V).

* In some applications, this chatter can cause malfunction → need a way to remove the chatter.
Because of the chatter, the $S$ and $R$ inputs may have multiple transitions when the switch is thrown from $A$ to $B$. However, for $S = R = 1$, the previous value of $Q$ is retained, causing a single transition in $Q$, as desired.
Because of the chatter, the $S$ and $R$ inputs may have multiple transitions when the switch is thrown from A to B.
* Because of the chatter, the $S$ and $R$ inputs may have multiple transitions when the switch is thrown from A to B.
* However, for $S = R = 1$, the previous value of $Q$ is retained, causing a single transition in $Q$, as desired.
* Complex digital circuits are generally designed for *synchronous* operation, i.e., transitions in the various signals are synchronised with the *clock*. 
* Complex digital circuits are generally designed for *synchronous* operation, i.e., transitions in the various signals are synchronised with the *clock*.

* Synchronous circuits are easier to design and troubleshoot because the voltages at the nodes (both output nodes and internal nodes) can change only at specific times.
* Complex digital circuits are generally designed for *synchronous* operation, i.e., transitions in the various signals are synchronised with the *clock*.

* Synchronous circuits are easier to design and troubleshoot because the voltages at the nodes (both output nodes and internal nodes) can change only at specific times.

* A clock is a periodic signal, with a positive-going transition and a negative-going transition.
* Complex digital circuits are generally designed for *synchronous* operation, i.e., transitions in the various signals are synchronised with the *clock*.

* Synchronous circuits are easier to design and troubleshoot because the voltages at the nodes (both output nodes and internal nodes) can change only at specific times.

* A clock is a periodic signal, with a positive-going transition and a negative-going transition.

* The clock frequency determines the overall speed of the circuit. For example, a processor that operates with a 1 GHz clock is 10 times faster than one that operates with a 100 MHz clock.
The “clock”

* Complex digital circuits are generally designed for *synchronous* operation, i.e., transitions in the various signals are synchronised with the *clock*.

* Synchronous circuits are easier to design and troubleshoot because the voltages at the nodes (both output nodes and internal nodes) can change only at specific times.

* A clock is a periodic signal, with a positive-going transition and a negative-going transition.

![Diagram of a clock signal with positive and negative edges]

* The clock frequency determines the overall speed of the circuit. For example, a processor that operates with a 1 GHz clock is 10 times faster than one that operates with a 100 MHz clock.

Intel 80286 (IBM PC-AT): 6 MHz
Modern CPU chips: 2 to 3 GHz.
Clocked RS latch

When clock is inactive (0), $A = B = 1$, and the latch holds the previous state.

When clock is active (1), $A = S$, $B = R$. Using the truth table for the NAND RS latch (right), we can construct the truth table for the clocked RS latch.

Note that the above table is sensitive to the level of the clock (i.e., whether CLK is 0 or 1).

M. B. Patil, IIT Bombay
When clock is inactive (0), $A = B = 1$, and the latch holds the previous state.

<table>
<thead>
<tr>
<th>CLK</th>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>$\overline{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

M. B. Patil, IIT Bombay
When clock is inactive (0), $A = B = 1$, and the latch holds the previous state.

When clock is active (1), $A = \overline{S}$, $B = \overline{R}$. Using the truth table for the NAND RS latch (right), we can construct the truth table for the clocked RS latch.
* When clock is inactive (0), \( A = B = 1 \), and the latch holds the previous state.

* When clock is active (1), \( A = \overline{S}, B = \overline{R} \). Using the truth table for the NAND RS latch (right), we can construct the truth table for the clocked RS latch.

* Note that the above table is sensitive to the level of the clock (i.e., whether CLK is 0 or 1).

\[
\begin{array}{c|c|c|c|c}
\text{CLK} & \text{R} & \text{S} & Q & \overline{Q} \\
0 & X & X & \text{previous} \\
1 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 1 & 0 \\
1 & 0 & 0 & \text{previous} \\
1 & 1 & 1 & \text{invalid} \\
\end{array}
\]
Clocked RS latch

CLK R S | Q  Q
---------|--------
0 X X   | previous
1 1 0   | 0 1
1 0 1   | 1 0
1 0 0   | previous
1 1 1   | invalid

M. B. Patil, IIT Bombay
* The clocked RS latch seen previously is *level-sensitive*, i.e., if the clock is active (CLK = 1), the flip-flop output is allowed to change, depending on the R and S inputs.
* The clocked RS latch seen previously is *level-sensitive*, i.e., if the clock is active (CLK = 1), the flip-flop output is allowed to change, depending on the R and S inputs.

* In an *edge-sensitive* flip-flop, the output can change only at the active clock edge (i.e., CLK transition from 0 to 1 or from 1 to 0).
Edge-triggered flip-flops

* The clocked RS latch seen previously is **level-sensitive**, i.e., if the clock is active (CLK = 1), the flip-flop output is allowed to change, depending on the R and S inputs.

* In an **edge-sensitive** flip-flop, the output can change only at the active clock **edge** (i.e., CLK transition from 0 to 1 or from 1 to 0).

* Edge-sensitive flip-flops are denoted by the following symbols:
JK flip-flop: introduction

Truth table for RS latch

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>( \overline{Q} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

When CLK = 0, we have \( R = S = 1 \), and the RS latch holds the previous \( Q \). In other words, nothing happens as long as CLK = 0.

When CLK = 1:

- \( J = K = 0 \) → \( R = S = 1 \), RS latch holds previous \( Q \), i.e., \( Q_{n+1} = Q_n \), where \( n \) denotes the \( n \)th clock pulse (This notation will become clear shortly).
- \( J = 0 \), \( K = 1 \) → \( R = 1 \), \( S = Q_n \).

Case (i): \( Q_n = 0 \) → \( S = 1 \) (i.e., \( R = S = 1 \)) → \( Q_{n+1} = Q_n = 0 \).

Case (ii): \( Q_n = 1 \) → \( S = 0 \) (i.e., \( R = 1 \), \( S = 0 \)) → \( Q_{n+1} = 0 \).

In either case, \( Q_{n+1} = 0 \).
A. When CLK = 0, we have \( R = S = 1 \), and the RS latch holds the previous \( Q \). In other words, nothing happens as long as CLK = 0.
When CLK = 0, we have R = S = 1, and the RS latch holds the previous Q. In other words, nothing happens as long as CLK = 0.
JK flip-flop: introduction

Truth table for RS latch

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>̅Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

Truth table for JK flip-flop

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>Q (Q_{n+1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>previous (Q_n)</td>
</tr>
</tbody>
</table>

* When CLK = 0, we have R = S = 1, and the RS latch holds the previous Q. In other words, nothing happens as long as CLK = 0.

* When CLK = 1:
* When \( \text{CLK} = 0 \), we have \( R = S = 1 \), and the RS latch holds the previous \( Q \). In other words, nothing happens as long as \( \text{CLK} = 0 \).

* When \( \text{CLK} = 1 \):
  - \( J = K = 0 \rightarrow R = S = 1 \), RS latch holds previous \( Q \), i.e., \( Q_{n+1} = Q_n \), where \( n \) denotes the \( n^{th} \) clock pulse (This notation will become clear shortly).
**Truth table for RS latch**

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>(\overline{Q})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

* When \(\text{CLK} = 0\), we have \(R = S = 1\), and the RS latch holds the previous \(Q\). In other words, nothing happens as long as \(\text{CLK} = 0\).

* When \(\text{CLK} = 1\):
  - \(J = K = 0\) \(\rightarrow\) \(R = S = 1\), RS latch holds previous \(Q\), i.e., \(Q_{n+1} = Q_n\), where \(n\) denotes the \(n^{th}\) clock pulse (This notation will become clear shortly).

**Truth table for JK flip-flop**

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>(Q (Q_{n+1}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>previous ((Q_n))</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>previous ((Q_n))</td>
</tr>
</tbody>
</table>

M. B. Patil, IIT Bombay
When \( \text{CLK} = 0 \), we have \( R = S = 1 \), and the RS latch holds the previous \( Q \). In other words, nothing happens as long as \( \text{CLK} = 0 \).

When \( \text{CLK} = 1 \):
- \( J = K = 0 \rightarrow R = S = 1 \), RS latch holds previous \( Q \), i.e., \( Q_{n+1} = Q_n \), where \( n \) denotes the \( n^{\text{th}} \) clock pulse (This notation will become clear shortly).
- \( J = 0, K = 1 \rightarrow R = 1, S = \overline{Q_n} \).
**JK flip-flop: introduction**

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>(\overline{Q})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

Truth table for RS latch

* When \(\text{CLK} = 0\), we have \(R = S = 1\), and the RS latch holds the previous \(Q\). In other words, nothing happens as long as \(\text{CLK} = 0\).

* When \(\text{CLK} = 1\):
  - \(J = K = 0 \rightarrow R = S = 1\), RS latch holds previous \(Q\), i.e., \(Q_{n+1} = Q_n\), where \(n\) denotes the \(n\)th clock pulse (This notation will become clear shortly).
  - \(J = 0, K = 1 \rightarrow R = 1, S = \overline{Q_n}\).
    Case (i): \(Q_n = 0 \rightarrow S = 1\) (i.e., \(R = S = 1\)) \(\rightarrow Q_{n+1} = Q_n = 0\).
JK flip-flop: introduction

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>āQ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

Truth table for RS latch

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>Q(Q_{n+1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>previous (Q_{n})</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>previous (Q_{n})</td>
</tr>
</tbody>
</table>

Truth table for JK flip-flop

* When CLK = 0, we have R = S = 1, and the RS latch holds the previous Q. In other words, nothing happens as long as CLK = 0.

* When CLK = 1:
  - J = K = 0 → R = S = 1, RS latch holds previous Q, i.e., Q_{n+1} = Q_{n}, where n denotes the n^{th} clock pulse (This notation will become clear shortly).
  - J = 0, K = 1 → R = 1, S = ĀQ_{n}.
    - Case (i): Q_{n} = 0 → S = 1 (i.e., R = S = 1) → Q_{n+1} = Q_{n} = 0.
    - Case (ii): Q_{n} = 1 → S = 0 (i.e., R = 1, S = 0) → Q_{n+1} = 0.
When CLK = 0, we have R = S = 1, and the RS latch holds the previous Q. In other words, nothing happens as long as CLK = 0.

When CLK = 1:
- J = K = 0 → R = S = 1, RS latch holds previous Q, i.e., \( Q_{n+1} = Q_n \), where \( n \) denotes the \( n^{th} \) clock pulse (This notation will become clear shortly).
- J = 0, K = 1 → R = 1, S = \( \bar{Q}_n \).
  - Case (i): \( Q_n = 0 \) → S = 1 (i.e., R = S = 1) → \( Q_{n+1} = Q_n = 0 \).
  - Case (ii): \( Q_n = 1 \) → S = 0 (i.e., R = 1, S = 0) → \( Q_{n+1} = 0 \).
In either case, \( Q_{n+1} = 0 \) → For J = 0, K = 1, \( Q_{n+1} = 0 \).
JK flip-flop: introduction

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>(\overline{Q})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td>Q</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

Truth table for RS latch

When \(\text{CLK} = 0\), we have \(R = S = 1\), and the RS latch holds the previous \(Q\). In other words, nothing happens as long as \(\text{CLK} = 0\).

When \(\text{CLK} = 1\):
- \(J = K = 0 \rightarrow R = S = 1\), RS latch holds previous \(Q\), i.e., \(Q_{n+1} = Q_n\), where \(n\) denotes the \(n^{\text{th}}\) clock pulse. (This notation will become clear shortly).
- \(J = 0, K = 1 \rightarrow R = 1, S = \overline{Q_n}\).

Case (i): \(Q_n = 0 \rightarrow S = 1\) (i.e., \(R = S = 1\)) \(\rightarrow Q_{n+1} = Q_n = 0\).

Case (ii): \(Q_n = 1 \rightarrow S = 0\) (i.e., \(R = 1, S = 0\)) \(\rightarrow Q_{n+1} = 0\).

In either case, \(Q_{n+1} = 0 \rightarrow \) For \(J = 0, K = 1\), \(Q_{n+1} = 0\).
JK flip-flop: introduction

Truth table for RS latch

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>( \overline{Q} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td>previous</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td>invalid</td>
</tr>
</tbody>
</table>

Truth table for JK flip-flop

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>( Q_{(n+1)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>previous ( Q_n )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>previous ( Q_n )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Previous and invalid states:

- **Previous state**: For any initial state \( Q_n \), the next state \( Q_{(n+1)} \) is determined by the current state and inputs.
- **Invalid state**: For \( J = 1 \) and \( K = 0 \), the state is invalid, meaning \( Q_{(n+1)} \) is not defined.

For \( J = 1 \) and \( K = 1 \):

- **Case (i)**: If \( Q_n = 0 \), then \( R = 0 \) (i.e., \( R = 0, S = 1 \)) → \( Q_{n+1} = 1 \).
- **Case (ii)**: If \( Q_n = 1 \), then \( R = 1 \) (i.e., \( R = 1, S = 0 \)) → \( Q_{n+1} = 0 \).

- \( \overline{Q} \) toggles when \( CLK = 1 \):

  - Consider \( J = 1 \), \( K = 0 \) → \( S = 1, R = Q_n \).
  - Case (i): \( Q_n = 0 \) → \( R = 0 \) (i.e., \( R = 0, S = 1 \)) → \( Q_{n+1} = 1 \).
  - Case (ii): \( Q_n = 1 \) → \( R = 1 \) (i.e., \( R = 1, S = 0 \)) → \( Q_{n+1} = 0 \).
**JK flip-flop: introduction**

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>\bar{Q}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

Truth table for RS latch

* When \(\text{CLK} = 1\):
  - Consider \(J = 1, \ K = 0 \rightarrow S = 1, \ R = \overline{Q_n} = Q_n\).
JK flip-flop: introduction

Truth table for RS latch

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>(\overline{Q})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>Invalid</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

Truth table for RS latch

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>Q (Q_{n+1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>previous (Q_n)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>previous (Q_n)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

* When \(CLK = 1\):
  - Consider \(J = 1, K = 0 \rightarrow S = 1, R = \overline{Q_n} = Q_n\).

Case (i): \(Q_n = 0 \rightarrow R = 0\) (i.e., \(R = 0, S = 1\)) \(\rightarrow Q_{n+1} = 1\).
**JK flip-flop: introduction**

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>(\overline{Q})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

Truth table for RS latch

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>Q_{(Q_{n+1})}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>previous (Q_n)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>previous (Q_n)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Truth table for JK flip-flop

* When \(CLK = 1\):
  - Consider \(J = 1, K = 0\) \(\rightarrow S = 1, R = \overline{Q}_n = Q_n\).
    - Case (i): \(Q_n = 0\) \(\rightarrow R = 0\) (i.e., \(R = 0, S = 1\)) \(\rightarrow Q_{n+1} = 1\).
    - Case (ii): \(Q_n = 1\) \(\rightarrow R = 1\) (i.e., \(R = 1, S = 1\)) \(\rightarrow Q_{n+1} = Q_n = 1\).
* When $\text{CLK} = 1$:
  - Consider $J = 1, K = 0 \rightarrow S = 1, R = \overline{Q_n} = Q_n$.
    - Case (i): $Q_n = 0 \rightarrow R = 0$ (i.e., $R = 0, S = 1$) $\rightarrow Q_{n+1} = 1$.
    - Case (ii): $Q_n = 1 \rightarrow R = 1$ (i.e., $R = 1, S = 1$) $\rightarrow Q_{n+1} = Q_n = 1$.
  $\rightarrow$ For $J = 1, K = 0, Q_{n+1} = 1$. 

\begin{tabular}{|c|c|c|c|}
  \hline
  R & S & Q & $\overline{Q}$ \\
  \hline
  1 & 0 & 0 & 1 \\
  0 & 1 & 1 & 0 \\
  1 & 1 & previous & \ \\
  0 & 0 & invalid & \ \\
  \hline
\end{tabular}

Truth table for RS latch

\begin{tabular}{|c|c|c|c|c|}
  \hline
  CLK & J & K & $Q (Q_{n+1})$ \\
  \hline
  0 & X & X & previous $(Q_n)$ \\
  1 & 0 & 0 & previous $(Q_n)$ \\
  1 & 0 & 1 & 0 \\
  \hline
\end{tabular}

Truth table for JK flip-flop

M. B. Patil, IIT Bombay
**Truth table for RS latch**

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>(\overline{Q})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

Truth table for RS latch

* When CLK = 1:
  - Consider \(J = 1, K = 0\) \(\rightarrow S = 1, R = \overline{Q_n} = Q_n\).
  - Case (i): \(Q_n = 0\) \(\rightarrow R = 0\) (i.e., \(R = 0, S = 1\)) \(\rightarrow Q_{n+1} = 1\).
  - Case (ii): \(Q_n = 1\) \(\rightarrow R = 1\) (i.e., \(R = 1, S = 1\)) \(\rightarrow Q_{n+1} = Q_n = 1\).
  - For \(J = 1, K = 0, Q_{n+1} = 1\).

---

**Truth table for JK flip-flop**

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>(Q(Q_{n+1}))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>previous ((Q_n))</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>previous ((Q_n))</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

* When CLK = 1:
  - Consider \(J = 1, K = 1\) \(\rightarrow R = Q_n, S = Q_n\).
  - Case (i): \(Q_n = 0\) \(\rightarrow R = 0\), \(S = 1\) \(\rightarrow Q_{n+1} = 1\).
  - Case (ii): \(Q_n = 1\) \(\rightarrow R = 1\), \(S = 0\) \(\rightarrow Q_{n+1} = 0\).
  - For \(J = 1, K = 1, Q_{n+1} = Q_n\).
When CLK = 1:

- Consider $J = 1, K = 0 \rightarrow S = 1, R = \overline{Q_n} = Q_n$.
  
  Case (i): $Q_n = 0 \rightarrow R = 0$ (i.e., $R = 0, S = 1$) \rightarrow $Q_{n+1} = 1$.
  
  Case (ii): $Q_n = 1 \rightarrow R = 1$ (i.e., $R = 1, S = 1$) \rightarrow $Q_{n+1} = Q_n = 1$.
  
  \rightarrow For $J = 1, K = 0, Q_{n+1} = 1$.

- Consider $J = 1, K = 1 \rightarrow R = Q_n, S = \overline{Q_n}$.
When CLK = 1:

- Consider $J = 1$, $K = 0 \rightarrow S = 1$, $R = \overline{Q_n} = Q_n$.
  
  Case (i): $Q_n = 0 \rightarrow R = 0$ (i.e., $R = 0$, $S = 1$) $\rightarrow Q_{n+1} = 1$.
  
  Case (ii): $Q_n = 1 \rightarrow R = 1$ (i.e., $R = 1$, $S = 1$) $\rightarrow Q_{n+1} = Q_n = 1$.
  
  $\rightarrow$ For $J = 1$, $K = 0$, $Q_{n+1} = 1$.

- Consider $J = 1$, $K = 1 \rightarrow R = Q_n$, $S = \overline{Q_n}$.
  
  Case (i): $Q_n = 0 \rightarrow R = 0$, $S = 1 \rightarrow Q_{n+1} = 1$. 
JK flip-flop: introduction

**Truth table for RS latch**

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>\overline{Q}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

* invalid

Previous

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>\overline{Q}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>R</td>
</tr>
</tbody>
</table>

When CLK = 1:

- Consider \( J = 1, K = 0 \) \( \rightarrow S = 1, R = \overline{Q_n} = Q_n \).
  - Case (i): \( Q_n = 0 \) \( \rightarrow R = 0 \) (i.e., \( R = 0, S = 1 \)) \( \rightarrow Q_{n+1} = 1 \).
  - Case (ii): \( Q_n = 1 \) \( \rightarrow R = 1 \) (i.e., \( R = 1, S = 1 \)) \( \rightarrow Q_{n+1} = Q_n = 1 \).
  - For \( J = 1, K = 0 \), \( Q_{n+1} = 1 \).

- Consider \( J = 1, K = 1 \) \( \rightarrow R = Q_n, S = \overline{Q_n} \).
  - Case (i): \( Q_n = 0 \) \( \rightarrow R = 0, S = 1 \) \( \rightarrow Q_{n+1} = 1 \).
  - Case (ii): \( Q_n = 1 \) \( \rightarrow R = 1, S = 0 \) \( \rightarrow Q_{n+1} = 0 \).

**Truth table for JK flip-flop**

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>( Q(Q_{n+1}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>previous ( (Q_n) )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>previous ( (Q_n) )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

M. B. Patil, IIT Bombay
JK flip-flop: introduction

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>\overline{Q}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
<td>previous</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>invalid</td>
</tr>
</tbody>
</table>

Truth table for RS latch

*When CLK = 1:*

- Consider $J = 1$, $K = 0 \implies S = 1$, $R = \overline{Q_n} = Q_n$.
  - Case (i): $Q_n = 0 \implies R = 0$ (i.e., $R = 0$, $S = 1$) $\implies Q_{n+1} = 1$.
  - Case (ii): $Q_n = 1 \implies R = 1$ (i.e., $R = 1$, $S = 1$) $\implies Q_{n+1} = Q_n = 1$.
  $\implies$ For $J = 1$, $K = 0$, $Q_{n+1} = 1$.

- Consider $J = 1$, $K = 1 \implies R = Q_n$, $S = \overline{Q_n}$.
  - Case (i): $Q_n = 0 \implies R = 0$, $S = 1$ $\implies Q_{n+1} = 1$.
  - Case (ii): $Q_n = 1 \implies R = 1$, $S = 0$ $\implies Q_{n+1} = 0$.
  $\implies$ For $J = 1$, $K = 1$, $Q_{n+1} = \overline{Q_n}$.

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>Q (Q_{n+1})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>previous (Q_n)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>previous (Q_n)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Truth table for JK flip-flop
* When \( \text{CLK} = 1 \):

- Consider \( J = 1, K = 0 \) \( \rightarrow \) \( S = 1, R = \overline{Q_n} = Q_n \).
  - Case (i): \( Q_n = 0 \) \( \rightarrow \) \( R = 0 \) (i.e., \( R = 0, S = 1 \)) \( \rightarrow \) \( Q_{n+1} = 1 \).
  - Case (ii): \( Q_n = 1 \) \( \rightarrow \) \( R = 1 \) (i.e., \( R = 1, S = 1 \)) \( \rightarrow \) \( Q_{n+1} = Q_n = 1 \).
  \( \rightarrow \) For \( J = 1, K = 0 \), \( Q_{n+1} = 1 \).

- Consider \( J = 1, K = 1 \) \( \rightarrow \) \( R = Q_n, S = \overline{Q_n} \).
  - Case (i): \( Q_n = 0 \) \( \rightarrow \) \( R = 0, S = 1 \) \( \rightarrow \) \( Q_{n+1} = 1 \).
  - Case (ii): \( Q_n = 1 \) \( \rightarrow \) \( R = 1, S = 0 \) \( \rightarrow \) \( Q_{n+1} = 0 \).
  \( \rightarrow \) For \( J = 1, K = 1 \), \( Q_{n+1} = \overline{Q_n} \).

Truth table for RS latch

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>( \overline{Q} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>previous</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>invalid</td>
<td></td>
</tr>
</tbody>
</table>

Truth table for JK flip-flop

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>( Q(Q_{n+1}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>previous ( (Q_n) )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>previous ( (Q_n) )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>toggles ( (Q_n) )</td>
</tr>
</tbody>
</table>
Consider $J = K = 1$ and $CLK = 1$. 

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>$Q(Q_{n+1})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>previous ($Q_n$)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>previous ($Q_n$)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>toggles ($Q_n$)</td>
</tr>
</tbody>
</table>

Truth table for JK flip-flop
Consider $J = K = 1$ and $\text{CLK} = 1$.

As long as $\text{CLK} = 1$, $Q$ will keep toggling! (The frequency will depend on the delay values of the various gates).

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>$Q(Q_{n+1})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>X</td>
<td>previous ($Q_n$)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>previous ($Q_n$)</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>toggles ($Q_n$)</td>
</tr>
</tbody>
</table>

Truth table for JK flip-flop
Consider $J = K = 1$ and $CLK = 1$.
As long as $CLK = 1$, $Q$ will keep toggling! (The frequency will depend on the delay values of the various gates).
→ Use the “Master-slave” configuration.
When CLK goes high, only the first latch is affected; the second latch retains its previous value (because $\text{CLK} = 0 \rightarrow R_2 = S_2 = 1$).

When CLK goes low, the output of the first latch ($Q_1$) is retained (since $R_1 = S_1 = 1$), and $Q_1$ can now affect $Q$.

In other words, the effect of any changes in $J$ and $K$ appears at the output $Q$ only when CLK makes a transition from 1 to 0. This is therefore a negative edge-triggered flip-flop.

Note that the JK flip-flop allows all four input combinations.

M. B. Patil, IIT Bombay
* When CLK goes high, only the first latch is affected; the second latch retains its previous value (because $\overline{CLK} = 0 \rightarrow R_2 = S_2 = 1$).
* When CLK goes high, only the first latch is affected; the second latch retains its previous value (because \( \overline{\text{CLK}} = 0 \rightarrow R_2 = S_2 = 1 \)).

* When CLK goes low, the output of the first latch (\( Q_1 \)) is retained (since \( R_1 = S_1 = 1 \)), and \( Q_1 \) can now affect \( Q \).
* When CLK goes high, only the first latch is affected; the second latch retains its previous value (because \( \text{CLK} = 0 \rightarrow R_2 = S_2 = 1 \)).

* When CLK goes low, the output of the first latch \((Q_1)\) is retained (since \(R_1 = S_1 = 1\)), and \(Q_1\) can now affect \(Q\).

* In other words, the effect of any changes in \(J\) and \(K\) appears at the output \(Q\) only when CLK makes a transition from 1 to 0. This is therefore a negative edge-triggered flip-flop.
JK flip-flop (Master-Slave)

* When CLK goes high, only the first latch is affected; the second latch retains its previous value (because $CLK = 0 \rightarrow R_2 = S_2 = 1$).
* When CLK goes low, the output of the first latch ($Q_1$) is retained (since $R_1 = S_1 = 1$), and $Q_1$ can now affect $Q$.
* In other words, the effect of any changes in $J$ and $K$ appears at the output $Q$ only when CLK makes a transition from 1 to 0.
This is therefore a negative edge-triggered flip-flop.
When CLK goes high, only the first latch is affected; the second latch retains its previous value (because $CLK = 0 \rightarrow R_2 = S_2 = 1$).

When CLK goes low, the output of the first latch ($Q_1$) is retained (since $R_1 = S_1 = 1$), and $Q_1$ can now affect $Q$.

In other words, the effect of any changes in $J$ and $K$ appears at the output $Q$ only when CLK makes a transition from 1 to 0. This is therefore a negative edge-triggered flip-flop.

Note that the JK flip-flop allows all four input combinations.
JK flip-flop (Master-Slave)

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>Q_{n+1}</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
<td>0</td>
<td>0</td>
<td>Q_n</td>
</tr>
<tr>
<td>↓</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>↓</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>↓</td>
<td>1</td>
<td>1</td>
<td>Q_n</td>
</tr>
</tbody>
</table>
JK flip-flop (Master-Slave)

CLK       J       K   |   Q_{n+1}
\downarrow 0  0  0   | Q_n
\downarrow 0  1  0   | 0
\downarrow 1  0  1   | 1
\downarrow 1  1  0   | Q_n
JK flip-flop (Master-Slave)

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>Q_{n+1}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>Q_n</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>Q_n</td>
</tr>
</tbody>
</table>
JK flip-flop (Master-Slave)

CLK | J  | K  | Q_{n+1}  
----|----|----|----------
    | 0  | 0  | Q_n      
    | 0  | 1  | 0        
    | 1  | 0  | 1        
    | 1  | 1  | Q_n      

RS latch 1
RS latch 2

M. B. Patil, IIT Bombay
JK flip-flop (Master-Slave)

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>Q_{n+1}</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
<td>0</td>
<td>0</td>
<td>Q_n</td>
</tr>
<tr>
<td>↓</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>↓</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>↓</td>
<td>1</td>
<td>1</td>
<td>Q_n</td>
</tr>
</tbody>
</table>
JK flip-flop (Master-Slave)

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>$Q_{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
<td>0</td>
<td>0</td>
<td>$Q_n$</td>
</tr>
<tr>
<td>↓</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>↓</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>↓</td>
<td>1</td>
<td>1</td>
<td>$Q_n$</td>
</tr>
</tbody>
</table>
JK flip-flop (Master-Slave)

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>Q_{n+1}</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>Q_n</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>Q_n</td>
</tr>
</tbody>
</table>
JK flip-flop (Master-Slave)

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>Q_{n+1}</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
<td>0</td>
<td>0</td>
<td>Q_n</td>
</tr>
<tr>
<td>↓</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>↓</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>↓</td>
<td>1</td>
<td>1</td>
<td>Q_n</td>
</tr>
</tbody>
</table>
JK flip-flop (Master-Slave)

### Truth Table

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>Q_{n+1}</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
<td>0</td>
<td>0</td>
<td>Q_n</td>
</tr>
<tr>
<td>↓</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>↓</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>↓</td>
<td>1</td>
<td>1</td>
<td>Q_n</td>
</tr>
</tbody>
</table>

### Waveforms

- **CLK**
- **J**
- **K**
- **R_1**
- **S_1**
- **R_2**
- **S_2**
- **Q_1**
- **Q**

M. B. Patil, IIT Bombay
Both negative (e.g., 74ALS112A, CD54ACT112) and positive (e.g., 74ALS109A, CD4027) edge-triggered JK flip-flops are available as ICs.
Both negative (e.g., 74ALS112A, CD54ACT112) and positive (e.g., 74ALS109A, CD4027) edge-triggered JK flip-flops are available as ICs.
Consider a negative edge-triggered JK flip-flop.
Consider a negative edge-triggered JK flip-flop.

* As seen earlier, when CLK is high (i.e., $t_{1A} < t < t_{1B}$, etc.), the input $J$ and $K$ determine the Master latch output $Q_1$.

  During this time, *no change* is visible at the flip-flop output $Q$. 
Consider a negative edge-triggered JK flip-flop.

* As seen earlier, when CLK is high (i.e., $t_{1A} < t < t_{1B}$, etc.), the input $J$ and $K$ determine the Master latch output $Q_1$.
  During this time, no change is visible at the flip-flop output $Q$.

* When the clock goes low, the Slave flip-flop becomes active, making it possible for $Q$ to change.
Consider a negative edge-triggered JK flip-flop.

* As seen earlier, when CLK is high (i.e., $t_{1A} < t < t_{1B}$, etc.), the input $J$ and $K$ determine the Master latch output $Q_1$. During this time, no change is visible at the flip-flop output $Q$.

* When the clock goes low, the Slave flip-flop becomes active, making it possible for $Q$ to change.

* In short, although the flip-flop output $Q$ can only change after the active edge, $(t_{1B}, t_{2B}$, etc.), the new $Q$ value is determined by $J$ and $K$ values just before the active edge.

M. B. Patil, IIT Bombay
Consider a negative edge-triggered JK flip-flop.

* As seen earlier, when CLK is high (i.e., $t_{1A} < t < t_{1B}$, etc.), the input $J$ and $K$ determine the Master latch output $Q_1$. During this time, no change is visible at the flip-flop output $Q$.

* When the clock goes low, the Slave flip-flop becomes active, making it possible for $Q$ to change.

* In short, although the flip-flop output $Q$ can only change after the active edge, $(t_{1B}, t_{2B}$, etc.), the new $Q$ value is determined by $J$ and $K$ values just before the active edge. This is a very important point!
JK flip-flop

positive edge–triggered JK flip–flop

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>$Q_{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>$Q_n$</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>$\overline{Q_n}$</td>
</tr>
</tbody>
</table>
### JK flip-flop

#### Positive edge-triggered JK flip-flop

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>$Q_{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>$Q_n$</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>$\overline{Q_n}$</td>
</tr>
</tbody>
</table>

#### Negative edge-triggered JK flip-flop

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>$Q_{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
<td>0</td>
<td>0</td>
<td>$Q_n$</td>
</tr>
<tr>
<td>↓</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>↓</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>↓</td>
<td>1</td>
<td>1</td>
<td>$\overline{Q_n}$</td>
</tr>
</tbody>
</table>

M. B. Patil, IIT Bombay
$J_1 = K_1 = 1$. Assume $Q_1 = Q_2 = 0$ initially.

$$
\begin{array}{cccc}
\text{CLK} & J & K & Q_{n+1} \\
\uparrow & 0 & 0 & Q_n \\
\uparrow & 0 & 1 & 0 \\
\uparrow & 1 & 0 & 1 \\
\uparrow & 1 & 1 & \overline{Q_n} \\
\end{array}
$$
$J_1 = K_1 = 1$. Assume $Q_1 = Q_2 = 0$ initially.

\[
\begin{array}{c|c|c|c}
\text{CLK} & J & K & Q_{n+1} \\
\hline
\uparrow & 0 & 0 & Q_n \\
\uparrow & 0 & 1 & 0 \\
\uparrow & 1 & 0 & 1 \\
\uparrow & 1 & 1 & \overline{Q_n}
\end{array}
\]

* Since $J_1 = K_1 = 1$, $Q_1$ toggles after every active clock edge.
$J_1 = K_1 = 1$. Assume $Q_1 = Q_2 = 0$ initially.

* Since $J_1 = K_1 = 1$, $Q_1$ toggles after every active clock edge.

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>$Q_{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>$Q_n$</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>$\overline{Q_n}$</td>
</tr>
</tbody>
</table>
$J_1 = K_1 = 1$. Assume $Q_1 = Q_2 = 0$ initially.

Since $J_1 = K_1 = 1$, $Q_1$ toggles after every active clock edge.

$J_2 = Q_1$, $K_2 = \overline{Q_1}$. We need to look at $J_2$ and $K_2$ values just before the active edge, to determine the next value of $Q_2$.
$J_1 = K_1 = 1$. Assume $Q_1 = Q_2 = 0$ initially.

* Since $J_1 = K_1 = 1$, $Q_1$ toggles after every active clock edge.

* $J_2 = Q_1$, $K_2 = Q_1$. We need to look at $J_2$ and $K_2$ values just before the active edge, to determine the next value of $Q_2$.

* It is convenient to construct a table listing $J_2$ and $K_2$ to figure out the next $Q_2$ value.

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>$Q_{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>up</td>
<td>0</td>
<td>0</td>
<td>$Q_n$</td>
</tr>
<tr>
<td>up</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>up</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>up</td>
<td>1</td>
<td>1</td>
<td>$Q_n$</td>
</tr>
</tbody>
</table>
JK flip-flop

\[ J_1 = K_1 = 1. \] Assume \( Q_1 = Q_2 = 0 \) initially.

\[ \begin{array}{c|c|c|c}
CLK & J & K & Q_{n+1} \\
\hline
\uparrow & 0 & 0 & Q_n \\
\uparrow & 0 & 1 & 0 \\
\uparrow & 1 & 0 & 1 \\
\uparrow & 1 & 1 & \overline{Q_n}
\end{array} \]

* Since \( J_1 = K_1 = 1 \), \( Q_1 \) toggles after every active clock edge.
* \( J_2 = Q_1 \), \( K_2 = \overline{Q_1} \). We need to look at \( J_2 \) and \( K_2 \) values just before the active edge, to determine the next value of \( Q_2 \).
* It is convenient to construct a table listing \( J_2 \) and \( K_2 \) to figure out the next \( Q_2 \) value.

\[ \begin{array}{c|c|c|c}
t & J_2 (t = t_k^-) & K_2 (t = t_k^+) & Q_2 (t = t_k^+) \\
\hline
t_1 & 0 & 1 & 0 \\
t_2 & 1 & 0 & 1 \\
t_3 & 0 & 1 & 0 \\
t_4 & 1 & 0 & 1 \\
t_5 & 0 & 1 & 0
\end{array} \]
$J_1 = K_1 = 1$. Assume $Q_1 = Q_2 = 0$ initially.

Since $J_1 = K_1 = 1$, $Q_1$ toggles after every active clock edge.

$J_2 = Q_1$, $K_2 = \overline{Q_1}$. We need to look at $J_2$ and $K_2$ values just before the active edge, to determine the next value of $Q_2$.

* It is convenient to construct a table listing $J_2$ and $K_2$ to figure out the next $Q_2$ value.
<table>
<thead>
<tr>
<th>t</th>
<th>$t_k^-$</th>
<th>$t_k^+$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$Q_0$</td>
<td>$Q_1$</td>
</tr>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_4$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_5$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\text{CLK}$</th>
<th>$J$</th>
<th>$K$</th>
<th>$Q_{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\downarrow$</td>
<td>0</td>
<td>0</td>
<td>$Q_n$</td>
</tr>
<tr>
<td>$\downarrow$</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$\downarrow$</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$\downarrow$</td>
<td>1</td>
<td>1</td>
<td>$\overline{Q_n}$</td>
</tr>
</tbody>
</table>
M. B. Patil, IIT Bombay
<table>
<thead>
<tr>
<th>t</th>
<th>( t_k^- )</th>
<th>( t_k^+ )</th>
</tr>
</thead>
<tbody>
<tr>
<td>t1</td>
<td>0 0 0 1 0 1 1 0 1 1 1 0</td>
<td></td>
</tr>
<tr>
<td>t2</td>
<td>1 1 0 1 0 0 0 1 0 1 1 1</td>
<td></td>
</tr>
<tr>
<td>t3</td>
<td>1 1 1 1 1 0 0 1 0</td>
<td></td>
</tr>
<tr>
<td>t4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>t5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>( Q_{n+1} )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 0</td>
<td>0 0</td>
<td>( Q_n )</td>
</tr>
<tr>
<td></td>
<td>0 1</td>
<td>0 0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 0</td>
<td>1 0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1 1</td>
<td>1 0</td>
<td>( \overline{Q_n} )</td>
</tr>
<tr>
<td>t</td>
<td>$Q_0$</td>
<td>$Q_1$</td>
<td>$Q_2$</td>
</tr>
<tr>
<td>-----</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$t_4$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$t_5$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### CLak J K | $Q_{n+1}$

<table>
<thead>
<tr>
<th></th>
<th>0 0 0</th>
<th>$Q_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0 0 1</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1 0 1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>1 1 1</td>
<td>$\overline{Q}_n$</td>
</tr>
<tr>
<td>t</td>
<td>$Q_0$</td>
<td>$Q_1$</td>
</tr>
<tr>
<td>----</td>
<td>------</td>
<td>------</td>
</tr>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$t_3$</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$t_4$</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$t_5$</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>$Q_{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓</td>
<td>0</td>
<td>0</td>
<td>$Q_n$</td>
</tr>
<tr>
<td>↓</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>↓</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>↓</td>
<td>1</td>
<td>1</td>
<td>$\overline{Q_n}$</td>
</tr>
</tbody>
</table>

CLK

$Q_0$

$Q_1$

$Q_2$
<table>
<thead>
<tr>
<th>t</th>
<th>( Q_0 )</th>
<th>( Q_1 )</th>
<th>( Q_2 )</th>
<th>( J_0 )</th>
<th>( K_0 )</th>
<th>( J_1 )</th>
<th>( K_1 )</th>
<th>( J_2 )</th>
<th>( K_2 )</th>
<th>( Q_0 )</th>
<th>( Q_1 )</th>
<th>( Q_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t_1 )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>( t_2 )</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( t_3 )</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( t_4 )</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>( t_5 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CLK | J   | K   | \( Q_{n+1} \)
---|-----|-----|-----------
↓ 0 0 | 0   | 0
↓ 0 1 | 1   | 0
↓ 1 0 | 1   | 0
↓ 1 1 | \( Q_n \) | 0

CLK

\( Q_0 \)

\( Q_1 \)

\( Q_2 \)
\[ \begin{array}{c|c|c|c|c|c|c|c|c} 
\text{CLK} & J & K & Q_{n+1} \\
\hline 
\downarrow & 0 & 0 & Q_n \\
\downarrow & 0 & 1 & 0 \\
\downarrow & 1 & 0 & 1 \\
\downarrow & 1 & 1 & \overline{Q_n} \\
\end{array} \]

\[ \begin{array}{c|c|c|c|c|c|c|c|c} 
\text{t} & Q_0 & Q_1 & Q_2 & J_0 & K_0 & J_1 & K_1 & J_2 & K_2 & Q_0 & Q_1 & Q_2 \\
\hline 
t_1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\
t_2 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
t_3 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
t_4 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 \\
t_5 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & \\
\end{array} \]
The diagram illustrates a J-K flip-flop circuit with three stages, labeled as $Q_0$, $Q_1$, and $Q_2$. The inputs are labeled $J$ and $K$ for each stage. The circuit includes pull-up and pull-down resistors labeled $CLK$ and $Q$.

A truth table for the circuit is shown below, detailing the state transitions in columns $t_k^-$ and $t_k^+$ for time points $t_1$, $t_2$, $t_3$, $t_4$, and $t_5$. The table compares the current state $Q_n$ with the next state $Q_{n+1}$, along with the inputs $J$, $K$, and the clock state.

### Truth Table

<table>
<thead>
<tr>
<th>$t$</th>
<th>$Q_0$</th>
<th>$Q_1$</th>
<th>$Q_2$</th>
<th>$J_0$</th>
<th>$K_0$</th>
<th>$J_1$</th>
<th>$K_1$</th>
<th>$J_2$</th>
<th>$K_2$</th>
<th>$Q_0$</th>
<th>$Q_1$</th>
<th>$Q_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$t_3$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$t_4$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$t_5$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

The table also includes a waveform graph showing the clock (CLK) and the states of the flip-flops $Q_0$, $Q_1$, and $Q_2$ at different time points $t_1$, $t_2$, $t_3$, $t_4$, and $t_5$. The graph illustrates the state transitions over time.
<table>
<thead>
<tr>
<th>t</th>
<th>t_{k^-}</th>
<th>t_{k^+}</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>Q_0 Q_1 Q_2 J_0 K_0 J_1 K_1 J_2 K_2 Q_0 Q_1 Q_2</td>
<td></td>
</tr>
<tr>
<td>t_1</td>
<td>0 0 0 1 0 1 1 0 1 1 1 0</td>
<td></td>
</tr>
<tr>
<td>t_2</td>
<td>1 1 0 1 0 0 0 1 0 1 1 1</td>
<td></td>
</tr>
<tr>
<td>t_3</td>
<td>1 1 1 1 1 0 0 1 0 0 1 1</td>
<td></td>
</tr>
<tr>
<td>t_4</td>
<td>0 1 1 1 1 1 1 0 0 1 0 1</td>
<td></td>
</tr>
<tr>
<td>t_5</td>
<td>1 0 1 1 1 0 0 1 1 0 0 0</td>
<td></td>
</tr>
</tbody>
</table>
JK flip-flop: asynchronous inputs

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 (\( J, K, \) and \( \text{CLK} \)).

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.

---

**Table:**

<table>
<thead>
<tr>
<th>( S_d )</th>
<th>( R_d )</th>
<th>( \text{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>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>( \uparrow )</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>( \uparrow )</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>( \uparrow )</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>( \uparrow )</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

**Diagram:**

- Normal operation

---

M. B. Patil, IIT Bombay
* 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 $\overline{R_d}$. 

M. B. Patil, IIT Bombay
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 $S = D$, $R = D$. 

M. B. Patil, IIT Bombay
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 $S = D$, $R = D$. 

M. B. Patil, IIT Bombay
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 \( S = D, R = \overline{D} \).
Shift register

CLK  D  Q_{n+1}
\downarrow 0  0
\downarrow 1  1

M. B. Patil, IIT Bombay
Shift register

CLK | D | Q_{n+1}
---|---|---
↓ | 0 | 0
↓ | 1 | 1
Shift register

CLK D Q_1
0 0 1 0

CLK D Q_2
0 0 1 0

CLK D Q_3
0 0 1 0

CLK D Q_4
0 0 1 0

CLK D Q_{n+1}
0 0 1 1

CLK D Q_1
0 0 1 0

CLK D Q_2
0 0 1 0

CLK D Q_3
0 0 1 0

CLK D Q_4
0 0 1 0

CLK D Q_{n+1}
0 0 1 1

M. B. Patil, IIT Bombay
Shift register

Shift register circuit diagram with D-flip-flops and clock signals. The diagram shows the behavior of the shift register with inputs and outputs represented by Q1, Q2, Q3, Q4, and D. The table lists the input and output states for different clock phases (CLK).
Shift register

\[ \begin{array}{c c c c c}
   & D & Q & Q_1 & Q_2 & Q_3 & Q_4 \\
\hline
   \text{D} & 1 & 0 & 1 & 1 & 0 & 0 \\
   \overline{Q} & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array} \]

<table>
<thead>
<tr>
<th>CLK</th>
<th>D</th>
<th>Q_{n+1}</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>00</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>11</td>
</tr>
</tbody>
</table>

M.B. Patil, IIT Bombay
Shift register

CLK

<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>

D Q Q Q Q Q Q

0 1 0 0 1 1 0 1

0 0 1 0 0 0 0 0

0 0 0 1 0 0 0 0

0 0 0 0 0 0 0 0

SEQUEL file: ee101 shift reg 1.sqproj

M. B. Patil, IIT Bombay
Shift register

CLK   D   Q_{n+1}
↓  0   0
↓  1   1

SEQUEL file: ee101 shift reg 1.sqproj

M. B. Patil, IIT Bombay
Shift register

CLK

<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>

CLK

D

Q_1

Q_2

Q_3

Q_4
Shift register

- Diagram of a 4-bit shift register showing the shift operation with inputs D, Qn+1, and CLK.
- Truth table for the shift operation:
<table>
<thead>
<tr>
<th>CLK</th>
<th>D</th>
<th>Qn+1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

- Waveform diagrams show the CLK, D, Q1, Q2, Q3, and Q4 signals over time.
Shift register

CLK | D | Q_{n+1}  
---|---|---------
↓ | 0 | 0      
↓ | 1 | 1      

CLK
D
Q_1
Q_2
Q_3
Q_4

SEQUEL file: ee101_shift_reg_1.sqproj
Parallel transfer between shift registers

After the active clock edge, the contents of the A register (A3 A2 A1 A0) are copied to the B register.

M. B. Patil, IIT Bombay
* 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 \( \rightarrow \) shift right operation.
* M \( \rightarrow \) shift left operation.

M. B. Patil, IIT Bombay
When the mode input (M) is 1, we have
\(D_0 = D_R, \ D_1 = Q_0, \ D_2 = Q_1, \ D_3 = Q_2\).

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

M. B. Patil, IIT Bombay
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
\[ D_0 = Q_1, \quad D_1 = Q_2, \quad D_2 = Q_3, \quad D_3 = D_L. \]
Bidirectional shift register

When the mode input (M) is 1, we have
\( D_0 = D_R, \ D_1 = Q_0, \ D_2 = Q_1, \ D_3 = Q_2. \)

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

* \( M = 1 \rightarrow \) shift right operation.
* \( M = 0 \rightarrow \) shift left operation.

M. B. Patil, IIT Bombay
Shift left operation

original number

\[
\begin{array}{ccccccc}
2^7 & 2^6 & 2^5 & 2^4 & 2^3 & 2^2 & 2^1 & 2^0 \\
0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\
\end{array}
\]

dec. 13

Shift left \( \rightarrow \times 2 \)

M. B. Patil, IIT Bombay
### Shift Left Operation

<table>
<thead>
<tr>
<th>Original Number</th>
<th>2⁷</th>
<th>2⁶</th>
<th>2⁵</th>
<th>2⁴</th>
<th>2³</th>
<th>2²</th>
<th>2¹</th>
<th>2⁰</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Original Number (dec. 13):**

<table>
<thead>
<tr>
<th></th>
<th>2⁷</th>
<th>2⁶</th>
<th>2⁵</th>
<th>2⁴</th>
<th>2³</th>
<th>2²</th>
<th>2¹</th>
<th>2⁰</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec. 13</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**After Shift Left (dec. 26):**

<table>
<thead>
<tr>
<th></th>
<th>2⁷</th>
<th>2⁶</th>
<th>2⁵</th>
<th>2⁴</th>
<th>2³</th>
<th>2²</th>
<th>2¹</th>
<th>2⁰</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dec. 26</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Shift left operation

<table>
<thead>
<tr>
<th>2^7</th>
<th>2^6</th>
<th>2^5</th>
<th>2^4</th>
<th>2^3</th>
<th>2^2</th>
<th>2^1</th>
<th>2^0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

original number

dec. 13

<table>
<thead>
<tr>
<th>2^7</th>
<th>2^6</th>
<th>2^5</th>
<th>2^4</th>
<th>2^3</th>
<th>2^2</th>
<th>2^1</th>
<th>2^0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

after shift left

dec. 26

Shift left $\rightarrow \times 2$

M. B. Patil, IIT Bombay
## Multiplication using shift and add

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

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

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

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

since \( B_0 = 1 \)

since \( B_1 = 0 \)

addition

since \( B_2 = 1 \)

addition

since \( B_3 = 1 \)

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

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

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**

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

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

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

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

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

\(A_3A_2A_1A_0\) (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 zeros which are independent of the numbers being multiplied.

Initialize:
\[
\begin{array}{cccc}
Z & Z & Z & Z \\
\end{array}
\]

Load:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
\end{array}
\]

Add:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 & Z & Z & Z & Z \\
\end{array}
\]

Shift:
\[
\begin{array}{cccc}
Z & Z & Z & Z \\
\end{array}
\]

Load:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
\end{array}
\]

Add:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 & Z & Z & Z & Z \\
\end{array}
\]

Shift:
\[
\begin{array}{cccc}
Z & Z & Z & Z \\
\end{array}
\]

Load:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
\end{array}
\]

Add:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 & Z & Z & Z & Z \\
\end{array}
\]

Shift:
\[
\begin{array}{cccc}
Z & Z & Z & Z \\
\end{array}
\]

Load:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
\end{array}
\]

Add:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 & Z & Z & Z & Z \\
\end{array}
\]

Shift:
\[
\begin{array}{cccc}
Z & Z & Z & Z \\
\end{array}
\]

Load:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
\end{array}
\]

Add:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 & Z & Z & Z & Z \\
\end{array}
\]

Shift:
\[
\begin{array}{cccc}
Z & Z & Z & Z \\
\end{array}
\]

Load:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
\end{array}
\]

Add:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 & Z & Z & Z & Z \\
\end{array}
\]

Shift:
\[
\begin{array}{cccc}
Z & Z & Z & Z \\
\end{array}
\]

Load:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
\end{array}
\]

Add:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 & Z & Z & Z & Z \\
\end{array}
\]

Shift:
\[
\begin{array}{cccc}
Z & Z & Z & Z \\
\end{array}
\]

Load:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
\end{array}
\]

Add:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 & Z & Z & Z & Z \\
\end{array}
\]

Shift:
\[
\begin{array}{cccc}
Z & Z & Z & Z \\
\end{array}
\]

Load:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 \\
\end{array}
\]

Add:
\[
\begin{array}{cccc}
1 & 0 & 1 & 1 & Z & Z & Z & Z \\
\end{array}
\]

Shift:
\[
\begin{array}{cccc}
Z & Z & Z & Z \\
\end{array}
\]
Multiplication using shift and add

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

A_3A_2A_1A_0 (decimal 11)  
B_3B_2B_1B_0 (decimal 13)

+ 1 0 1 1 + 0 0 0 0 0 Z
+ 1 0 1 1 1 Z Z

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

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

\[
1 \ 0 \ 1 \ 1 \\
+ \ 0 \ 1 \ 0 \ 1 \ 1 \\
\hline
1 \ 1 \ 0 \ 1 \ 1 \ Z \ Z
\]

\[
1 \ 1 \ 0 \ 1 \ 1 \\
+ \ 1 \ 0 \ 1 \ 1 \ Z \ Z \ Z
\hline
1 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 1
\]

A_3A_2A_1A_0 (decimal 11)
B_3B_2B_1B_0 (decimal 13)

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

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

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

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

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

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

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

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

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

- **Register 1**
  - Initialize: \[Z \ Z \ Z \ Z\]
  - Load 1011: \[1 \ 0 \ 1 \ 1\]
  - Add: \[1 \ 0 \ 1 \ 1 \ Z \ Z \ Z \ Z\]
  - Shift: \[Z \ 1 \ 0 \ 1 \ 1 \ Z \ Z \ Z \ Z\]
  - Load 0000: \[0 \ 0 \ 0 \ 0\]
  - Add: \[0 \ 1 \ 0 \ 1 \ 1 \ Z \ Z \ Z \ Z\]
  - Shift: \[Z \ 0 \ 1 \ 0 \ 1 \ Z \ Z \ Z \ Z\]

- **Register 2**
  - Initialize: \[Z \ Z \ Z \ Z\]
  - Load 1011: \[1 \ 0 \ 1 \ 1\]
  - Add: \[1 \ 0 \ 1 \ 1 \ Z \ Z \ Z \ Z\]
  - Shift: \[Z \ 1 \ 0 \ 1 \ 1 \ Z \ Z \ Z \ Z\]

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

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

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

<table>
<thead>
<tr>
<th>A_3A_2A_1A_0 (decimal 11)</th>
<th>B_3B_2B_1B_0 (decimal 13)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 1 1</td>
<td>1 1 0 1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Register 2</th>
<th>Register 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Z) (Z) (Z) (Z)</td>
<td>(Z) (Z) (Z) (Z)</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>(Z) (Z) (Z) (Z)</td>
</tr>
<tr>
<td>1 0 1 1</td>
<td>1 0 1 1</td>
</tr>
<tr>
<td>(Z) 1 0 1</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0 0 0 0</td>
<td>0 1 0 1</td>
</tr>
<tr>
<td>0 1 0 1</td>
<td>1 1 Z Z</td>
</tr>
<tr>
<td>(Z) 0 1 0</td>
<td>1 0 1 1</td>
</tr>
</tbody>
</table>

- Initialize: \(Z\) \(Z\) \(Z\) \(Z\) \(Z\) \(Z\) \(Z\) \(Z\)
- Load: 1 0 1 1 since \(B_0 = 1\)
- Add: since \(B_1 = 0\)
- Shift: since \(B_2 = 1\)
- Load: 0 0 0 0 since \(B_1 = 0\)
- Add: since \(B_3 = 1\)
- Shift: since \(B_2 = 1\)
- Load: 1 0 1 1
Multiplication using shift and add

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

A3A2A1A0 (decimal 11)
B3B2B1B0 (decimal 13)

since B0 = 1
since B1 = 0

addition
since B2 = 1

addition
since B3 = 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

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

\(A_3A_2A_1A_0\) (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

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

\( A_3A_2A_1A_0 \) (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
(Note that \( Z = 0 \). We use \( Z \) to denote 0s which are independent of the numbers being multiplied.)
Multiplication using shift and add

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

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

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

Addition
Since \(B_2 = 1\)
Addition
Since \(B_3 = 1\)

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

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

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

1. Since \(B_0 = 1\)
2. Since \(B_1 = 0\)
3. Addition
4. Since \(B_2 = 1\)
5. Addition
6. Since \(B_3 = 1\)
7. Addition

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

Initialize:

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

Load 1011 since \(B_0 = 1\):

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

Add:

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

Shift:

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

Add:

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

Shift:

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

M. B. Patil, IIT Bombay
* 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.
* 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 \( R_d = \text{Clear} = 1, S_d = 0 \)).
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.
* 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.
* Subsequently, with every clock pulse, the data shifts right and appears \textit{serially} at the output $Q_0$. → parallel in-serial out data movement

$$Q_0 = A_0 \quad Q_0 = A_1 \quad Q_0 = A_2 \quad Q_0 = A_3$$
* 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.
* Subsequently, with every clock pulse, the data shifts right and appears serially at the output $Q_0$. $ightarrow$ 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.
* Subsequently, with every clock pulse, the data shifts right and appears serially at the output $Q_0$. \(\rightarrow\) 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.
* 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 \( R_d = \text{Clear} = 1 \), \( S_d = 0 \)).

When Load = 1, \( S_d = A_i \), \( R_d = 0 \) → \( A_i \) gets loaded into the \( i^{th} \) flip-flop.

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 $R_d = \text{Clear} = 1$, $S_d = 0$).
* When Load = 1, $S_d = A_i$, $R_d = 0 \rightarrow A_i$ gets loaded into the $i^{th}$ flip-flop.
* 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 $R_d = \text{Clear} = 1$, $S_d = 0$).

When Load = 1, $S_d = A_i$, $R_d = 0 \rightarrow A_i$ gets loaded into the $i^{th}$ flip-flop.

Subsequently, with every clock pulse, the data shifts right and appears serially at the output $Q_0$ -> parallel in-serial out data movement.
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. \( \rightarrow \) \( N \) 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.

M. B. Patil, IIT Bombay
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-\( k \) (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.

\[ \rightarrow N \text{ flip-flops can be used to make a mod-}k \text{ counter with } k \leq 2^N. \]
* 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.
  \( \rightarrow \) \( N \) 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.
State transition diagram

**State Transition Table**

<table>
<thead>
<tr>
<th>State</th>
<th>(Q_0)</th>
<th>(Q_1)</th>
<th>(Q_2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**State Transition Diagram**

Counters: example

M. B. Patil, IIT Bombay
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.

X is 1 for state 3; else, it is 0.
The counter outputs (i.e., the flip-flop outputs, $Q_0$, $Q_1$, \ldots $Q_{N-1}$) can be decoded using appropriate logic.

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

1
CLK

FF0
J
Q

K
Q

FF1
J
Q

K
Q

FF2
J
Q

K
Q

Q0
Q1
Q2

CLK

*J = K = 1 for all flip-flops. Let Q0 = Q1 = Q2 = 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, Q0 and Q1, 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.
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.
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 Q₀ = Q₁ = Q₂ = 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, Q₀ and Q₁, 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, $Q_2 Q_1 Q_0 = 000, 001, 010, 011, 100, 101, 110, 111$.
  → it is a mod-8 counter. In particular, it is a binary, mod-8, up counter (since it counts up from 000 to 111).
A binary ripple counter

The counter has 8 states, \( Q_2 Q_1 Q_0 = 000, 001, 010, 011, 100, 101, 110, 111 \).

Therefore, it is a \textit{mod-8} counter. In particular, it is a \textit{binary, mod-8, up} counter (since it counts \textit{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.

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

* The counter has 8 states, \( Q_2 Q_1 Q_0 = 000, 001, 010, 011, 100, 101, 110, 111 \).
  \( \rightarrow \) 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.

\[ \begin{array}{ccc}
\text{Q}_2 & \text{Q}_1 & \text{Q}_0 \\
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 1 & 0 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
1 & 1 & 1 \\
0 & 0 & 0 \uparrow \text{ repeats} \\
\end{array} \]
If positive edge-triggered flip-flops are used, we get a binary down counter (counting down from 111 to 000).
If positive edge-triggered flip-flops are used, we get a binary down counter (counting down from 111 to 000).
* Home work: Sketch the waveforms (CLK, $Q_0$, $Q_1$, $Q_2$), and tabulate the counter states in each case.
When Mode (M) = 1, the counter counts up; else, it counts down. (SEQUEL file: counter3.sqproj)
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."

M. B. Patil, IIT Bombay
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 $Q_3Q_2Q_1Q_0 = 0000$. 

M. B. Patil, IIT Bombay
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.”

<table>
<thead>
<tr>
<th>$Q_3$</th>
<th>$Q_2$</th>
<th>$Q_1$</th>
<th>$Q_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

SEQUEL file: ee101_counter_5.sqproj

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.

\[
\begin{align*}
J_0 &= K_0 = 1, \\
J_1 &= K_1 = Q_0, \\
J_2 &= K_2 = Q_1 Q_0, \\
J_3 &= K_3 = Q_2 Q_1 Q_0.
\end{align*}
\]

FF0 toggles after every active edge. FF1 toggles if \(Q_0 = 1\) (just before the active clock edge); else, it retains its previous state. (Similarly, for FF2 and FF3)

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

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

* FF0 toggles after every active edge.
* FF1 toggles if $Q_0 = 1$ (just before the active clock edge); else, it retains its previous state. (Similarly, for FF2 and FF3)

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

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

\[ J_0 = K_0 = 1, \quad J_1 = K_1 = Q_0, \quad J_2 = K_2 = Q_1 Q_0, \quad J_3 = K_3 = Q_2 Q_1 Q_0. \]
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 after every active edge.
* FF1 toggles if $Q_0 = 1$ (just before the active clock edge); else, it retains its previous state. (Similarly, for 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 after every active edge.

FF1 toggles if \( Q_0 = 1 \) (just before the active clock edge); else, it retains its previous state. (Similarly, for 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 after every active edge.

* FF1 toggles if $Q_0 = 1$ (just before the active clock edge); else, it retains its previous state. (Similarly, for 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 after every active edge.
- FF1 toggles if $Q_0 = 1$ (just before the active clock edge); else, it retains its previous state. (Similarly, for FF2 and FF3)

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

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

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

* FF0 toggles after every active edge.

* FF1 toggles if $Q_0 = 1$ (just before the active clock edge); else, it retains its previous state. (Similarly, for FF2 and FF3)
A synchronous counter

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

\[ J_0 = K_0 = 1, \quad J_1 = K_1 = Q_0, \quad J_2 = K_2 = Q_1 Q_0, \quad J_3 = K_3 = Q_2 Q_1 Q_0. \]

* FF0 toggles after every active edge.

* FF1 toggles if \( Q_0 = 1 \) (just before the active clock edge); else, it retains its previous state. (Similarly, for 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 \( Q_{n+1} = Q_n \) by making \( J = 0, K = 0 \).

- \( J = 0, K = \overline{X} \) (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?

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>$Q_{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>$Q_n$</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>$\overline{Q_n}$</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></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</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 $Q_{n+1} = Q_n$ 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 \( Q_{n+1} = Q_n \) by making \( J = 0, K = 0 \).

\[ J = 0, K = X \text{ (i.e., } K \text{ can be 0 or 1).} \]
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 $Q_{n+1} = Q_n$ by making $J = 0$, $K = 0$.
- $J = 0$, $K = X$ (i.e., $K$ can be 0 or 1).

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>$Q_{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>$Q_n$</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>$\overline{Q_n}$</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>
</tbody>
</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 $Q_{n+1} = Q_n$ 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 $Q_{n+1} = Q_n$ by making $J = 0$, $K = 0$. $ightarrow 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 $Q_{n+1} = Q_n$ 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 $Q_{n+1} = Q_n$ by making $J = 0, K = 0$.

$\rightarrow J = 0, K = X$ (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 $Q_{n+1} = Q_n$ by making $J = 0$, $K = 0$.
  $\rightarrow J = 0$, $K = X$ (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 $Q_{n+1} = Q_n$ 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.
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 $Q_{n+1} = Q_n$ by making $J = 0, K = 0$.
  \[ J = 0, K = X \text{ (i.e., } K \text{ can be 0 or 1).} \]

- Similarly, work out the other entries in the table.
Design of synchronous counters

<table>
<thead>
<tr>
<th>CLK</th>
<th>J</th>
<th>K</th>
<th>$Q_{n+1}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
<td>$Q_n$</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
<td>$\overline{Q_n}$</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>

* 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 $Q_{n+1} = Q_n$ 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 a synchronous mod-5 counter with the given state transition table.

<table>
<thead>
<tr>
<th>state</th>
<th>Q_2</th>
<th>Q_1</th>
<th>Q_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Design of synchronous counters

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_0, Q_1, Q_2.
- 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_0, Q_1, Q_2. This can be done with K-maps. (If the number of flip-flops is more than 4, other techniques can be employed.)

M. B. Patil, IIT Bombay
Design a synchronous mod-5 counter with the given state transition table.

Outline of method:

<table>
<thead>
<tr>
<th>state</th>
<th>Q_2</th>
<th>Q_1</th>
<th>Q_0</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</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 a synchronous mod-5 counter with the given state transition table.

Outline of method:

* State 1 → State 2 means
  \[ Q_2: 0 \rightarrow 0, \]
  \[ Q_1: 0 \rightarrow 0, \]
  \[ Q_0: 0 \rightarrow 1. \]
Design of synchronous counters

<table>
<thead>
<tr>
<th>state</th>
<th>$Q_2$</th>
<th>$Q_1$</th>
<th>$Q_0$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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

Outline of method:

* State 1 → 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 \( Q_0, Q_1, Q_2. \)
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 $Q_0$, $Q_1$, $Q_2$.

* 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 $Q_0$, $Q_1$, $Q_2$. This can be done with K-maps. (If the number of flip-flops is more than 4, other techniques can be employed.)
### 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></td>
<td></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>

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.
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.

---

<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></td>
<td></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>$\uparrow$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>$\uparrow$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>$\uparrow$</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>$\uparrow$</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>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.

We treat these as don’t care conditions.
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></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>
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.
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></td>
<td></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>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>0</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>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.
### 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></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>6</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>

#### Truth Tables

<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 $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, and $K_2$ in terms of $Q_0$, $Q_1$, and $Q_2$. 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 $Q_0$, $Q_1$, and $Q_2$ 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.

### 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></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>

### Clock 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>

M. B. Patil, IIT Bombay
### 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>1</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</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>1</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>X</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

#### Truth 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>$\uparrow$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$X$</td>
</tr>
<tr>
<td>$\uparrow$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$X$</td>
</tr>
<tr>
<td>$\uparrow$</td>
<td>1</td>
<td>0</td>
<td>$X$</td>
<td>1</td>
</tr>
<tr>
<td>$\uparrow$</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>$Q_2$</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>

### Truth Tables

<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>X</td>
<td>0</td>
<td></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>0</td>
<td></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>X</td>
<td>0</td>
<td></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>X</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

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.

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

State transition table for the synchronous counter:

<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>

Next step is to find the logical functions for each state.

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.

<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

<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>1</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

CLK  | Qₙ  | Qₙ₊₁ | J  | K  \\
---   | ---- | ------ | --- | ---
↑ 0   | 0    | 0     | 0  | X  \\
↑ 0   | 1    | 1     | 1  | X  \\
↑ 1   | 0    | X     | 1  |
↑ 1   | 1    | X     | 0  |
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>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></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.
We now have the truth tables for $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, and $K_2$ in terms of $Q_0$, $Q_1$, and $Q_2$. 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 $Q_0$, $Q_1$, and $Q_2$ which do not occur in the state transition table (such as $Q_2Q_1Q_0 = 110$). We treat these as don’t care conditions.
### 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>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>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>

#### Truth Tables

<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 $J_0$, $K_0$, $J_1$, $K_1$, $J_2$, and $K_2$ 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.

<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></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>
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.

**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>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>

**Truth 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.

<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>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>

<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>$\uparrow$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>$\uparrow$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>$\uparrow$</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>$\uparrow$</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>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>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>X</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>

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.

---

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 logical functions for each of them are:

<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>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>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></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.
## 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>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>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></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>$\uparrow$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$X$</td>
</tr>
<tr>
<td>$\uparrow$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$X$</td>
</tr>
<tr>
<td>$\uparrow$</td>
<td>1</td>
<td>0</td>
<td>$X$</td>
<td>1</td>
</tr>
<tr>
<td>$\uparrow$</td>
<td>1</td>
<td>1</td>
<td>$X$</td>
<td>0</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.
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>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>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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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, $Q_2Q_1Q_0 = 000$).

From the K-maps, $J_2 = Q_1Q_0$, $K_2 = 1$, $J_1 = Q_0$, $K_1 = Q_0$, $J_0 = Q_2$, $K_0 = 1$. 

M. B. Patil, IIT Bombay
### Design of Synchronous Counters

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.

<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>1</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>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>X</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>

* \(J_2 = Q_2Q_1Q_0\), \(K_2 = 0\)*
* \(J_1 = Q_2Q_1Q_0\), \(K_1 = 1\)*
* \(J_0 = Q_2Q_1Q_0\), \(K_0 = 1\)*
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>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>1</td>
<td>X</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>X</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>X</td>
<td>X</td>
<td>1</td>
<td>X</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>0</td>
<td>X</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 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, $Q_2 Q_1 Q_0 = 000$).
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>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>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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></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, $J_2 = \overline{Q_1} Q_0$, $K_2 = 1$, $J_1 = Q_0$, $K_1 = Q_0$, $J_0 = \overline{Q_2}$, $K_0 = 1$. 

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

\[
\begin{align*}
J_2 &= Q_1, \\
K_2 &= 1, \\
J_1 &= Q_0, \\
K_1 &= Q_0, \\
J_0 &= Q_2, \\
K_0 &= 1.
\end{align*}
\]

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

SEQUEL file: ee101_counter_6.sqproj
Design of synchronous counters: verification

\[ J_2 = Q_1 Q_0, \]
\[ K_2 = 1, \]
\[ J_1 = Q_0, \]
\[ K_1 = Q_0, \]
\[ J_0 = \overline{Q_2}, \]
\[ K_0 = 1. \]

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

\[ J_2 = Q_1 Q_0, \]
\[ K_2 = 1, \]
\[ J_1 = Q_0, \]
\[ K_1 = Q_0, \]
\[ J_0 = \overline{Q_2}, \]
\[ K_0 = 1. \]

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

SEQUEL file: ee101_counter_6.sqproj
Combination of counters: Approach 1

Clock 1 → Clock 2

Counter 1
mod-$k_1$

$k_1 = 4$

A$_1$
A$_2$
...
A$_{N_1}$

Counter 2
mod-$k_2$

$k_2 = 3$

B$_1$
B$_2$
...
B$_{N_2}$

→ the combined counter is a mod-$k_1$ mod-$k_2$ counter.

M. B. Patil, IIT Bombay
Combination of counters: Approach 1

Clock 1 ➔ Counter 1
mod-\(k_1\)
\(k_1 = 4\)

Clock 2 ➔ Counter 2
mod-\(k_2\)
\(k_2 = 3\)

Clock 1 ➔ Counter 1
mod-\(k_1\)

Clock 1 ➔ Counter 1 state
\[4 \ 1 \ 2 \ 3 \ 4 \ 1 \ 2 \ 3 \ 4 \ 1 \ 2 \ 3 \ 4 \ 1 \]

Counter 2 state
\[1 \ 2 \ 3 \ 1\]

\[\rightarrow\text{the combined counter is a mod-}\(k_1 \times k_2\) counter.\]
Combination of counters: Approach 1

Clock 1

Counter 1
mod-k_{1}

k_{1} = 4

\begin{align*}
A_{1} \\
A_{2} \\
\vdots \\
A_{N_{1}}
\end{align*}

Clock 2

Counter 2
mod-k_{2}

k_{2} = 3

\begin{align*}
B_{1} \\
B_{2} \\
\vdots \\
B_{N_{2}}
\end{align*}

Clock 1

Counter 1
mod-k_{1}

Decoding Logic

Counter 2
mod-k_{2}

Counter 1 state

\begin{align*}
4 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 1 \\
\end{align*}

Clock 2

Counter 2 state

\begin{align*}
1 & 2 & 3 & 1 \\
\end{align*}

Combined state

\begin{align*}
2 & 3 & 4 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 1 & 2 & 3 & 4 & 1 \\
\end{align*}

→ the combined counter is a mod-k_{1}k_{2} counter.

M. B. Patil, IIT Bombay
Combination of counters: Approach 1

Counter 1

\[ k_1 = 4 \]

Counter 2

\[ k_2 = 3 \]

Clock 1

Decoding Logic

Clock 2

→ the combined counter is a mod-\(k_1\)\(k_2\) counter.

M. B. Patil, IIT Bombay
Combination of counters: Approach 1

Clock 1

Counter 1
mod-\(k_1\)

\(k_1 = 4\)

Clock 2

Counter 2
mod-\(k_2\)

\(k_2 = 3\)

Counter 1 state 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 1

Clock 1

Counter 1
mod-\(k_1\)

Decoding
Logic

Clock 2

Counter 2
mod-\(k_2\)

Counter 2 state 1 2 3 1

Combined state 3 1 1 1 2 2 2 2 3 3 3 3 1 1 1 1 2 4 1 2 3 4 1 2 3 4 1 1

→ the combined counter is a mod-\(k_1 k_2\) counter.

M. B. Patil, IIT Bombay
Combination of counters: Approach 1

\[ \text{Counter 1} \quad \text{mod-}k_1 \quad k_1 = 4 \]

\[ \text{Counter 2} \quad \text{mod-}k_2 \quad k_2 = 3 \]

Clock 1

\[ A_1 \quad A_2 \quad \ldots \quad A_{N_1} \]

Clock 2

\[ B_1 \quad B_2 \quad \ldots \quad B_{N_2} \]

Clock 1

Counter 1 state

\[ 4 \quad 1 \quad 2 \quad 3 \quad 4 \quad 1 \quad 2 \quad 3 \quad 4 \quad 1 \quad 2 \quad 3 \quad 4 \quad 1 \quad 2 \quad 3 \quad 4 \quad 1 \quad 2 \quad 3 \quad 4 \quad 1 \quad 2 \quad 3 \quad 4 \quad 1 \quad 2 \quad 3 \quad 4 \quad 1 \]

Clock 2

Counter 2 state

\[ 1 \quad 2 \quad 3 \quad 1 \]

Combined state

\[ 3 \quad 1 \quad 1 \quad 1 \quad 2 \quad 2 \quad 2 \quad 2 \quad 3 \quad 3 \quad 3 \quad 3 \quad 3 \quad 1 \quad 1 \quad 1 \quad 1 \quad 2 \quad 4 \quad 1 \quad 2 \quad 3 \quad 4 \quad 1 \quad 2 \quad 3 \quad 4 \quad 1 \quad 2 \quad 3 \quad 4 \quad 1 \]

→ the combined counter is a mod-\(k_1k_2\) counter.

M. B. Patil, IIT Bombay
Combination of counters: example

mod−2 counter

mod−5 counter
Combination of counters: example

SEQUEL file: ee101_counter_7.sqproj

CLK 1 J JJ
51 2 3 4 5 6 7 8 9 10 1 2 3 4 6 7 8 9 10
QA Q0 Q1
Q2

M. B. Patil, IIT Bombay
Combination of counters: Approach 2

Clock 1

Counter 1
mod-$k_1$

$k_1 = 4$

A1

A2

\ldots

A_{N_1}

Clock 2

Counter 2
mod-$k_2$

$k_2 = 3$

B1

B2

\ldots

B_{N_2}

→ the combined counter is a mod-$k_1$-$k_2$ counter.
Combination of counters: Approach 2

Clock 1 → Counter 1 modulation $k_1$ = 4

Clock 2 → Counter 2 modulation $k_2$ = 3

Counter 1 state:
- $A_1$
- $A_2$
- ...
- $A_{N_1}$

Counter 2 state:
- $B_1$
- $B_2$
- ...
- $B_{N_2}$

Combined state:
- $2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4, 1, 1$
- $13, 2, 1, 3, 3, 3, 3, 3, 1, 1, 1, 12, 2, 2, 2, 2$

→ the combined counter is a mod-$k_1 k_2$ counter.

M. B. Patil, IIT Bombay
Combination of counters: Approach 2

Clock 1 → Counter 1 mod-$k_1$

- $k_1 = 4$
- $A_1$
- $A_2$
- $\ldots$
- $A_{N_1}$

Clock 2 → Counter 2 mod-$k_2$

- $k_2 = 3$
- $B_1$
- $B_2$
- $\ldots$
- $B_{N_2}$

→ the combined counter is a mod-$k_1$ mod-$k_2$ counter.

Clock 1

Counter 1 state

Counter 1 state

Counter 2 state

Combined state

M. B. Patil, IIT Bombay
Combination of counters: Approach 2

Clock 1 → Counter 1
mod-$k_1$

$A_1$

$A_2$

$\cdots$

$A_{N1}$

Clock 1

Counter 1 state

$4 \ 1 \ 2 \ 3 \ 4 \ 1 \ 2 \ 3 \ 4 \ 1 \ 2 \ 3 \ 4 \ 1 \ 2 \ 3 \ 4 \ 1$

$\cdots$

$k_1 = 4$

Clock 2 → Counter 2
mod-$k_2$

$B_1$

$B_2$

$\cdots$

$B_{N2}$

Counter 2 state

$1 \ 3 \ 2 \ 1 \ 3 \ 3 \ 3 \ 3 \ 1 \ 1 \ 1 \ 2 \ 2 \ 2 \ 2$

$\cdots$

$k_2 = 3$

→ the combined counter is a mod-$k_1$ $k_2$ counter.
Combination of counters: Approach 2

Counter 1
mod-\(k_1\)
\(k_1 = 4\)

Counter 2
mod-\(k_2\)
\(k_2 = 3\)

Combined state
2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 1

→ the combined counter is a mod-\(k_1 k_2\) counter.

M. B. Patil, IIT Bombay
Combination of counters: Approach 2

Counter 1
mod-$k_1$

$k_1 = 4$

Counter 2
mod-$k_2$

$k_2 = 3$

Clock 1

Counter 1 state

Clock 2

Counter 2 state

Combined state

→ the combined counter is a mod-$k_1 k_2$ counter.

M. B. Patil, IIT Bombay
Combination of counters: Approach 2

Clock 1 → Counter 1 mod-\(k_1\)  
\(k_1 = 4\)  
\[\begin{array}{c}
A_1 \\
A_2 \\
\vdots \\
A_{N_1}
\end{array}\]

Clock 2 → Counter 2 mod-\(k_2\)  
\(k_2 = 3\)  
\[\begin{array}{c}
B_1 \\
B_2 \\
\vdots \\
B_{N_2}
\end{array}\]

Counter 1 state  
4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1

Counter 2 state  
3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2

Combined state  
3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2

→ the combined counter is a mod-\(k_1 k_2\) counter.

M. B. Patil, IIT Bombay
Combination of counters: example (same as before)

- **mod-2 counter**
  - J0, K0
  - CLK

- **mod-5 counter**
  - J0, K0
  - CLK
Combination of counters: example

SEQUEL file: ee101_counter_8.sqproj

CLK

J_A

Q_A

K_A

J

Q

K

Q


Q_A

Q_0

Q_1

Q_2

CLK

1

2

3

4

5

6

7

8

9

10

1

2

3

4

5

6

7

8

9

10

M. B. Patil, IIT Bombay
The 555 timer is useful in timer, pulse generation, and oscillator applications. We will look at two common applications.
The 555 timer is useful in timer, pulse generation, and oscillator applications. We will look at two common applications.

* Monostable multivibrator

\[ V_{\text{out}} \] for \( T \) seconds after \( V_{\text{trigger}} \]
The 555 timer is useful in timer, pulse generation, and oscillator applications. We will look at two common applications.

* Monostable multivibrator

* Astable multivibrator
555 timer

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>(\overline{Q})</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>previous</td>
<td></td>
</tr>
</tbody>
</table>

\(V_{CC}\)

Threshold

Trigger

Discharge

Out

buffer
555 timer

<table>
<thead>
<tr>
<th>R</th>
<th>S</th>
<th>Q</th>
<th>$\overline{Q}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>previous</td>
<td>0</td>
</tr>
</tbody>
</table>
555 timer

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

VCC

STOP

M. B. Patil, IIT Bombay
555 monostable multivibrator

\[ V_C \left( t \right) = V_{CC} \left( 1 - e^{-t/\tau} \right) \rightarrow 2V_{CC}/3 = V_{CC}\left( 1 - e^{-T/\tau} \right) \rightarrow e^{-T/\tau} = \frac{1}{3} \rightarrow T = \tau \log 3 \approx 1.1 \tau \]
555 monostable multivibrator

\[ V_C \]

\[ V_{CC} \]

\[ R \]

\[ S \]

\[ Q \]

\[ \text{Trigger} \]

\[ \text{Discharge} \]

\[ \text{Out} \]

\[ 2V_{CC}/3 \]

\[ V_{CC}/3 \]

\[ V_{CC} \]

\[ T \]

\[ \text{Threshold} \]

\[ V_C \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]

\[ t \]
555 monostable multivibrator

\[ V_C(t) = V_{CC}(1 - e^{-t/\tau}) \rightarrow 2V_{CC}/3 = V_{CC}(1 - e^{-T/\tau}) \rightarrow e^{-T/\tau} = 1/3 \rightarrow T = \tau \log 3 \approx 1.1 \tau \]
555 monostable multivibrator

The 555 monostable multivibrator is a circuit that produces a single pulse when triggered. The circuit consists of two inverting amplifiers, a Schmitt trigger, and a flip-flop. The circuit is powered by a supply voltage, $V_{CC}$.

When triggered, the circuit output, $Q$, goes high for a duration of $T$, where $T$ is given by

$$T = \tau \log_3 \approx 1.1 \tau$$

where $\tau$ is the time constant of the RC network formed by the resistor, $R$, and the capacitor, $C$. The rise time of the output pulse is $\frac{V_{CC}}{3}$.

The discharge process begins when the output $Q$ returns to ground, and the circuit returns to its initial state.

The waveforms illustrate the timing parameters of the multivibrator, including the trigger, discharge, and output signals.

SEQUEL file: ic555 mono 1.sqproj

M. B. Patil, IIT Bombay
555 monostable multivibrator

\[ V_C(t) = V_{CC} \left(1 - e^{-t/\tau}\right) \]
555 monostable multivibrator

\[ V_C(t) = V_{CC} \left( 1 - e^{-t/\tau} \right) \]

\[ \Rightarrow \frac{2 V_{CC}}{3} = V_{CC} \left( 1 - e^{-T/\tau} \right) \]
555 monostable multivibrator

\[ V_C(t) = V_{CC} \left( 1 - e^{-t/\tau} \right) \]

\[ \rightarrow \frac{2 V_{CC}}{3} = V_{CC} \left( 1 - e^{-T/\tau} \right) \]

\[ \rightarrow e^{-T/\tau} = \frac{1}{3} \rightarrow T = \tau \log 3 \approx 1.1 \tau \]
555 monostable multivibrator

\[ V_C(t) = V_{CC} \left( 1 - e^{-t/\tau} \right) \]

\[ \rightarrow \frac{2 \ V_{CC}}{3} = V_{CC} \left( 1 - e^{-T/\tau} \right) \]

\[ \rightarrow e^{-T/\tau} = \frac{1}{3} \rightarrow T = \tau \log 3 \approx 1.1 \tau \]

SEQUEL file: ic555_mono_1.sqproj
Charging:
\[ V_C(0) = \frac{2V_{CC}}{3}, \quad V_C(\infty) = V_{CC}. \]

Let \( V_C(t) = A e^{-t/\tau_1} + B \rightarrow B = V_{CC}, \quad A = -\frac{2V_{CC}}{3} = -\frac{2V_{CC}}{3} e^{-T_H/\tau_1} + V_{CC} \rightarrow T_H = \tau_1 \log 2, \text{ with } \tau_1 = (R_a + R_b)C. \]
555 astable multivibrator

Charging:
\[ V_C(0) = \frac{2V_{CC}}{3}, \quad V_C(\infty) = V_{CC}. \]
Let \[ V_C(t) = A e^{-t/\tau_1} + B \rightarrow B = V_{CC}, \quad A = -\frac{2V_{CC}}{3} = -\frac{2V_{CC}}{3} e^{-T_H/\tau_1} + V_{CC} \rightarrow T_H = \tau_1 \log 2, \] with \( \tau_1 = (R_a + R_b)C. \)
Charging:

\[ V_C(0) = \frac{V_{CC}}{3}, \quad V_C(\infty) = V_{CC} \].

M. B. Patil, IIT Bombay
555 astable multivibrator

Charging:

\[ V_C(0) = \frac{V_{CC}}{3}, \quad V_C(\infty) = V_{CC}. \]

Let \( V_C(t) = A e^{-t/\tau_1} + B \)

\( \rightarrow B = V_{CC}, \quad A = -\frac{2 V_{CC}}{3} \)
555 astable multivibrator

Charging:

\[ V_C(0) = \frac{V_{CC}}{3}, \quad V_C(\infty) = V_{CC}. \]

Let \( V_C(t) = A e^{-t/\tau_1} + B \)

\[ \rightarrow B = V_{CC}, \quad A = -\frac{2}{3} V_{CC} \]

\[ \frac{2}{3} V_{CC} = -\frac{2}{3} V_{CC} e^{-T_H/\tau_1} + V_{CC} \]

M. B. Patil, IIT Bombay
555 astable multivibrator

Charging:
\[ V_C(0) = \frac{V_{CC}}{3}, \quad V_C(\infty) = V_{CC}. \]

Let \( V_C(t) = A e^{-t/\tau_1} + B \)

\[ \rightarrow B = V_{CC}, \quad A = -\frac{2 V_{CC}}{3} \]

\[ \frac{2 V_{CC}}{3} = -\frac{2 V_{CC}}{3} e^{-T_H/\tau_1} + V_{CC} \]

\[ \rightarrow T_H = \tau_1 \log 2, \quad \text{with} \quad \tau_1 = (R_a + R_b) C. \]

M. B. Patil, IIT Bombay
555 astable multivibrator

Discharging:

\[ V_C(0) = 2V_{CC}, \quad V_C(\infty) = 0. \]

\[ V_C(t) = 2V_{CC} e^{-t/\tau}, \quad 2V_{CC} = 2V_{CC} e^{-T_L/\tau}, \quad T_L = \tau \log 2, \quad \tau = R_b C. \]

SEQUEL file: ic555

M. B. Patil, IIT Bombay
555 astable multivibrator

Discharging:

\[ V_C(0) = \frac{2V_{CC}}{3}, \quad V_C(\infty) = 0. \]

\[ V_C(t) = \frac{2V_{CC}}{3} e^{-t/\tau}, \quad \tau_R = \frac{R_b C}. \]

SEQUEL file: ic555astable1.sqproj

M. B. Patil, IIT Bombay
555 astable multivibrator

Discharging: $V_C(0) = \frac{2V_{CC}}{3}$, $V_C(\infty) = 0$.

$\rightarrow V_C(t) = \frac{2V_{CC}}{3} e^{-t/\tau_2}$
Discharging: $V_C(0) = \frac{2V_{CC}}{3}$, $V_C(\infty) = 0$.

$\rightarrow V_C(t) = \frac{2V_{CC}}{3} e^{-t/\tau_2}$

$V_{CC} = \frac{2V_{CC}}{3} e^{-T_L/\tau_2}$
555 astable multivibrator

Discharging: $V_C(0) = \frac{2V_{CC}}{3}$, $V_C(\infty) = 0$.

$\rightarrow V_C(t) = \frac{2V_{CC}}{3} e^{-t/\tau_2}$

$\frac{V_{CC}}{3} = \frac{2V_{CC}}{3} e^{-T_L/\tau_2}$

$\rightarrow T_L = \tau_2 \log 2$, with $\tau_2 = R_b C$. 

M. B. Patil, IIT Bombay
Discharging: $V_C(0) = \frac{2V_{CC}}{3}$, $V_C(\infty) = 0$.

$\rightarrow V_C(t) = \frac{2V_{CC}}{3} e^{-t/\tau_2}$

$\frac{V_{CC}}{3} = \frac{2V_{CC}}{3} e^{-T_L/\tau_2}$

$\rightarrow T_L = \tau_2 \log 2$, with $\tau_2 = R_b C$.

SEQUEL file: ic555_able.1.sqproj