Counters

Counters are a class of sequential logic circuits that tally a series of input pulses; the input pulses may be regular or irregular in nature. The counter is a fundamental part of most digital logic applications. It is used in timing units, control circuits, signal generators, and numerous other devices.

- According to how they handle input transitions
  - Synchronous
  - Asynchronous
- According to counting sequence
  - Binary
  - Non-binary
Synchronous binary counter

• Synchronous because all flip-flop operate from the same clock
  – This one contains negative edge triggered FFs
Synchronous binary counter

• State sequence given in table
  – When does bit $X_i$ toggle?
  – What is the maximum frequency of operation?
  – What is the delay?

<table>
<thead>
<tr>
<th>$X_n$</th>
<th>$X_3$</th>
<th>$X_2$</th>
<th>$X_1$</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>...</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</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>
<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>
</tbody>
</table>
• Synchronous binary counter SN74163
Synchronous binary counter SN74163

• Function table:

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Clear</th>
<th>Load</th>
<th>ENT</th>
<th>ENP</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>×</td>
<td>×</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
<td>×</td>
<td>L</td>
<td>×</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
<td>H</td>
<td>L</td>
<td>×</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
<td>×</td>
<td>L</td>
<td>L</td>
</tr>
</tbody>
</table>

Mode
- Synchronous clear
- Synchronous load
- Count
- Hold
- Hold

• Logic of operation
  - KC ≡ K input of FF C
  - JC ≡ J input of FF C

\[
KC = \left[ \text{Load} + \text{Clear} + Q_B \cdot Q_A \cdot \text{ENT} \cdot \text{ENP} \right] \cdot \left[ (\text{Load} + \text{Clear}) \cdot \text{Clear} \cdot \text{Data}_{C} \right] \\
= \text{Clear} + \text{Load} \cdot Q_B \cdot Q_A \cdot \text{ENT} \cdot \text{ENP} + \text{Load} \cdot \text{Data}_{C} \\
\]

\[
JC = \text{Clear} \cdot \left[ \text{Load} \cdot Q_B \cdot Q_A \cdot \text{ENT} \cdot \text{ENP} + \text{Load} \cdot \text{Data}_{C} \right] \\
\]
Synchronous binary counter SN74163

\[ \text{KC} = \text{Clear} + \text{Load} \cdot Q_B \cdot Q_A \cdot \text{ENT} \cdot \text{ENP} + \text{Load} \cdot Q_B \cdot Q_A \cdot \text{ENT} \cdot \text{ENP} \]

\[ \text{JC} = \text{Clear} \cdot \left[ \text{Load} \cdot Q_B \cdot Q_A \cdot \text{ENT} \cdot \text{ENP} + \text{Load} \cdot Q_B \cdot Q_A \cdot \text{ENT} \cdot \text{ENP} \right] \]

– a) Synchronous clear:
  Clear = 1; Load = 0  \implies JC = 0; KC = 1  \implies FF reset on next CK↓

– b) Synchronous load:
  Clear = 0; Load = 1  \implies JC = \text{Data}_C; KC = \text{Data}_C \implies FF latches data on next CK↓

– c) Count mode:
  Clear = 0; Load = 0  \implies JC = KC = Q_A \cdot Q_B \cdot \text{ENT} \cdot \text{ENP}  
  \implies FF toggles or not on next CK↓, depending on Q_A \cdot Q_B

– d) Hold mode:
  ENT = ENP = 0
Synchronous binary counter SN74163

- Timing diagram
Asynchronous binary counter

- Asynchronous because the flip-flops do not operate from the same clock
  - Therefore, not all FFs change at the same time
Asynchronous binary counter

- Toggle of $X_i$ is caused by a $1 \rightarrow 0$ transition in $X_{i-1}$
- Can we use master-slave FFs instead of edge triggered?

- On wraparound from $2^n-1$ to 0, state passes ("ripples") through $2^n-2$, $2^n-4$, etc. to 0, instead of a direct transition $2^n-1 \rightarrow 0$
- Called "ripple" counter
Counters

• Let the delay of 1 stage of a n-bit binary counter be $\Delta$
  – The synchronous counter must have a clock pulse period $T \geq n\Delta$
  – The asynchronous counter can have a pulse period $T \geq \Delta$

• If we don’t care about intermediate states, after starting in the same state, and applying
  – k pulses $n\Delta$ wide to the synchronous counter
  – k pulses $\Delta$ wide to the asynchronous counter

Both counters will be in the same final state (after hold)
  – The asynchronous counter stages will trigger each other properly, even if multiple pulses present & racing through
74293 asynchronous binary counter

- J and K are internally held high
- When $R_{0(1)}$ and $R_{0(2)}$ are both 1, all flip-flops are reset (0)
- Connecting $Q_A$ to Input B gives a 4 bit counter
- Count pulses with $T$ longer than the propagation delay are required if a stable output of the different states is desired
- On wraparound from $2^n-1$ to 0, state passes ("ripples") through $2^n-2$, $2^n-4$, etc. to 0, instead of a direct transition $2^n-1 \rightarrow 0$
74293 asynchronous binary counter

- Counter passes through intermediate transient states (small circles) between the steady states (the large circles)
- This is the picture only if \( T \) is longer than the propagation delay
74177 asynchronous binary counter

- Has asynchronous load mode

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clear</td>
<td>Count/Load</td>
</tr>
<tr>
<td>L</td>
<td>×</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
</tr>
</tbody>
</table>

Asynchronous common clear
Asynchronous load
Count
74177 asynchronous binary counter

– Functions

\[ \text{Preset}_B = \text{Data}_B \cdot (\text{Count/Load}) \cdot \text{Clear} \]

\[ \text{Clear}_B = (\text{Clear}) + (\text{Count/Load}) \cdot (\text{Data}_B) \]

Clear is low, inside the flip-flop, Preset\textsubscript{B} is low and Clear\textsubscript{B} is high. This is the asynchronous common clear mode of operation. Setting the external Clear line high (inactive) allows the Count/Load external line to control the device.

If the Clear line is high and Count/Load is low, then Preset\textsubscript{B} = Data\textsubscript{B} and Clear\textsubscript{B} = (Data\textsubscript{B}). This is the asynchronous load mode of operation because the value of Data\textsubscript{B} will be forced into the flip-flop. If the external Clear and Count/Load lines are both high, then Preset\textsubscript{R} = Clear\textsubscript{R} = 0; this is the count mode for the device.
A *down*, or *backward*, counter is one whose state transitions are reversed from those of the standard counter, which is also known as an *up*, or *forward*, counter.

<table>
<thead>
<tr>
<th>$X_n$</th>
<th>...</th>
<th>$X_3$</th>
<th>$X_2$</th>
<th>$X_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>...</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>...</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Up count mode

<table>
<thead>
<tr>
<th>$X_n$</th>
<th>...</th>
<th>$X_3$</th>
<th>$X_2$</th>
<th>$X_1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>...</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>...</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>...</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>...</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>...</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Down count mode

Question: In down count mode, when do we toggle (complement) a bit?
Down counters

• Answer
  – (1) Only if the all the lower-order bits are 0 AND we decrement the counter (i.e. subtract 1)
    • Borrow propagates as long as the bits are zero: this feature is useful for the synchronous design
  – (2) LSB toggles on every count pulse
  – (3) A 0 → 1 transition in bit i toggles bit i+1
    • This feature is useful in the asynchronous design
  – (4) If the up counter starts from state 0, and the down counter from $2^n-1$, every state bit in the one is the complement of the corresponding bit in the other
    • This feature is useful in up/down counter design
Down counters

- Asynchronous down counter
  - Notice the use of feature (3)
Up/down counters

• Synchronous up/down counter
  – Notice the use of features (1) and (4)
  – Would a synchronous or asynchronous up/down be faster and use less hardware?
Up/down counters

- Asynchronous up/down counter
  - Notice the use of features (3) and (4)
SN74191
• Synchronous 4-bit up/down counter with asynchronous load, enable, ripple clock and maximum state output
Inspect the S and R terminals of the four flip-flops. These correspond to the Preset and Clear terminals in previous example chips. The logic driving the S and R terminals resembles the logic for Preset and Clear in the SN74177

\[
S_C = Data_C \cdot (\text{Load})
\]
\[
R_C = [Data_C \cdot (\text{Load})] \cdot (\text{Load})
\]
\[
= Data_C \cdot (\text{Load})
\]

When \(\text{Load}\) is low, then \(S_C = Data_C\) and \(R_C = (Data_C)\), so the value of \(Data_C\) is loaded into the flip-flop asynchronously. When \(\text{Load}\) is high, \(S_C = R_C = 0\), so the counter is controlled by its other inputs, \(\overline{CTEN}, D/\bar{U}\), and Clock.

Next examine the \(J\) and \(K\) inputs of flip-flop C:

\[
J_C = K_C = (\overline{CTEN}) \cdot [Q_B \cdot Q_A \cdot (D/\bar{U}) + \bar{Q}_B \cdot \bar{Q}_A \cdot (D/\bar{U})]
\]
When the count enable input signal $\overline{CTEN}$ is high, both $J_C$ and $K_C$ are logic 0, so no changes will occur in the flip-flop outputs. When $\overline{CTEN}$ is low, the counter enters the up or down counting mode depending on the value of $D/\overline{U}$. If $D/\overline{U}$ is low, then $Q_B$ and $Q_A$ determine the toggling of $Q_C$ so that the counter is in the \textit{up} mode. When $D/\overline{U}$ is high, $\bar{Q}_B$ and $\bar{Q}_A$ determine the toggling of $Q_C$, and so the counter sequences backward, or \textit{down}, through its states.

- Function table

<table>
<thead>
<tr>
<th>Inputs</th>
<th>Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>$\overline{CTEN}$</td>
</tr>
<tr>
<td>L</td>
<td>$\times$</td>
</tr>
<tr>
<td>H</td>
<td>H</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
</tr>
<tr>
<td>H</td>
<td>L</td>
</tr>
</tbody>
</table>
SN74191

- Timing diagram