Module-3 Counter

Input: \( x, x \in \{0, 1\} \)
Output: \( \{Z_1, Z_0\}, Z_i \in \{0, 1\} \)

<table>
<thead>
<tr>
<th>( PS )</th>
<th>( x = 0 )</th>
<th>( x = 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>01</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( NS )</th>
<th>( Output )</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>01</td>
<td>01</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
</tbody>
</table>

For SR flip-flop:

<table>
<thead>
<tr>
<th>( PS )</th>
<th>( NS )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0-</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
</tr>
</tbody>
</table>

and output is equal to present state.

For \( S_1 \):

\[ S_1 = Q_0 x \]

For \( R_1 \):

\[ R_1 = Q_0 ' x \]

For \( S_0 \):

\[ S_0 = Q_1 ' Q_0 ' x \]

For \( R_0 \):

\[ R_0 = Q_0 x \]
2. **Full adder**

\[ Z_i = p_i \oplus c_i \]
\[ c_{i+1} = g_i + p_i \cdot c_i \]

![Diagram of full adder circuit](image)

- [X1], [X0], [Y1], [Y0]: Inputs
- [C0]: Carry-in
- [Z0]: Sum
- [C1]: Carry-out

---

2. **Full adder**

\[ Z_i = p_i \oplus c_i \]
\[ c_{i+1} = g_i + p_i \cdot c_i \]

![Diagram of full adder circuit](image)

- [X1], [X0], [Y1], [Y0]: Inputs
- [C0]: Carry-in
- [Z0]: Sum
- [C1]: Carry-out
### 2-out-of-5 Code

<table>
<thead>
<tr>
<th>Input $x_i$</th>
<th>Output $z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>00011</td>
<td>0</td>
</tr>
<tr>
<td>00101</td>
<td>0</td>
</tr>
<tr>
<td>01001</td>
<td>0</td>
</tr>
<tr>
<td>10001</td>
<td>0</td>
</tr>
<tr>
<td>00110</td>
<td>0</td>
</tr>
<tr>
<td>01010</td>
<td>0</td>
</tr>
<tr>
<td>10010</td>
<td>0</td>
</tr>
<tr>
<td>01100</td>
<td>0</td>
</tr>
<tr>
<td>10100</td>
<td>0</td>
</tr>
<tr>
<td>11000</td>
<td>0</td>
</tr>
</tbody>
</table>

$z = \left( x_4 + x_3 + x_2 + x_1 + x_0 \right) \left( x_4 + x_3 + x_2 + x_1 + x_0 \right)$

\[
\begin{align*}
\text{Input } & = (x_4, x_3, x_2, x_1, x_0), \quad x_i \in \{0, 1\}, \quad \text{Output } z \in \{0, 1\} \\
(x_4', x_3', x_2', x_1', x_0') & = x_4 + x_3 + x_2 + x_1 + x_0' \\
(x_4', x_3', x_2', x_1', x_0') & = x_4 + x_3 + x_2' + x_1 + x_0' \\
(x_4', x_3', x_2', x_1', x_0') & = x_4 + x_3 + x_2 + x_1' + x_0' \\
(x_4', x_3', x_2', x_1', x_0') & = x_4 + x_3 + x_2 + x_1 + x_0 \\
(x_4', x_3', x_2', x_1', x_0') & = x_4 + x_3 + x_2 + x_1 + x_0' \\
(x_4', x_3', x_2', x_1', x_0') & = x_4 + x_3 + x_2' + x_1 + x_0 \\
(x_4', x_3', x_2', x_1', x_0') & = x_4 + x_3 + x_2' + x_1 + x_0' \\
(x_4', x_3', x_2', x_1', x_0') & = x_4 + x_3' + x_2 + x_1 + x_0 \\
(x_4', x_3', x_2', x_1', x_0') & = x_4 + x_3' + x_2' + x_1 + x_0 \\
(x_4', x_3', x_2', x_1', x_0') & = x_4 + x_3 + x_2' + x_1 + x_0' \\
(x_4', x_3', x_2', x_1', x_0') & = x_4 + x_3' + x_2 + x_1 + x_0 \\
(x_4', x_3', x_2', x_1', x_0') & = x_4 + x_3 + x_2 + x_1 + x_0 \\
\end{align*}
\]
4d) Shift register:

Since it is a three-bit shift register, so we need 2 D-flip-flop to implement it.

4c) Comparator:
5. **FPGA**s is field-programmable gate arrays which can be programmed to implement a digital system which have thousands of gate inside.

- ROM is called read-only memory. It is corresponds to a tabular representation of a combinational system in which the input bit vector is used to identify which row in the array is accessed.
- PLA consists of a stage register which stores states and a PLA which is used to implement state transition and output.
- PLA consists of a set of arrays of AND gates and or gates which can be used to implement different functions.
- PALs is similar to PLAs but the size is bigger and easier to implement or modify. Both PLAs and PALs are programmable.
- RTL is a network that consists of the standard sequential and combinational modules.

- RTL is register transfer level. It consists of data subsystem and control subsystem.
- Conditions are exchanged between data and control system, which implements flow logic function.