6.12 (State minimization) Derive the minimal-state FSM from the state/output table shown in Figure P6.12.

<table>
<thead>
<tr>
<th>Present state</th>
<th>next-state</th>
<th>x = 0</th>
<th>x = 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>s₀</td>
<td>s₀/1</td>
<td>s₄/0</td>
<td></td>
</tr>
<tr>
<td>s₁</td>
<td>s₀/0</td>
<td>s₄/0</td>
<td></td>
</tr>
<tr>
<td>s₂</td>
<td>s₁/0</td>
<td>s₅/0</td>
<td></td>
</tr>
<tr>
<td>s₃</td>
<td>s₁/0</td>
<td>s₅/0</td>
<td></td>
</tr>
<tr>
<td>s₄</td>
<td>s₂/0</td>
<td>s₆/1</td>
<td></td>
</tr>
<tr>
<td>s₅</td>
<td>s₂/0</td>
<td>s₆/1</td>
<td></td>
</tr>
<tr>
<td>s₆</td>
<td>s₃/0</td>
<td>s₇/1</td>
<td></td>
</tr>
<tr>
<td>s₇</td>
<td>s₃/0</td>
<td>s₇/1</td>
<td></td>
</tr>
</tbody>
</table>
### Implication table

\[ G_0 = \{s_0\}, \ G_1 = \{s_1\}, \ G_2 = \{s_2, s_3\}, \ G_3 = \{s_4, s_5, s_6, s_7\} \]

<table>
<thead>
<tr>
<th>State</th>
<th>Next state</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(X = 0)</td>
</tr>
<tr>
<td>(G_0)</td>
<td>(G_0/1)</td>
</tr>
<tr>
<td>(G_2)</td>
<td>(G_0/0)</td>
</tr>
<tr>
<td>(G_3)</td>
<td>(G_1/0)</td>
</tr>
<tr>
<td>(G_4)</td>
<td>(G_2/0)</td>
</tr>
</tbody>
</table>
6.14 (State encoding) For the state diagram below, derive the state encodings using (a) the minimum-bit-change heuristic, (b) the prioritized adjacency heuristic and (c) hot-one encoding.

(a) Minimum-bit-change results in:

\[ \begin{align*}
  s_0 &= 000 \\
  s_1 &= 001 \\
  s_2 &= 010 \\
  s_3 &= 011 \\
  s_4 &= 101
\end{align*} \]
(b) Prioritized adjacency results in:

priority 1 = \{s_1, s_2, s_3\}, \{s_2, s_4\}
priority 2 = \{s_2, s_3\}, \{s_0, s_1\}, \{s_0, s_3\}, \{s_3, s_4\}, \{s_0, s_3\}
priority 3 = \{s_0, s_1, s_2\}, \{s_3, s_4\}

(c) Hot-one encoding results in:

\begin{align*}
s_0 &= 00001 \\
s_1 &= 00010 \\
s_2 &= 00100 \\
s_3 &= 01000 \\
s_4 &= 10000
\end{align*}
6.17 (Sequential Synthesis) Design a counter that counts in the sequence 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, ..., using natural binary encoding and D-type flip-flops.

In order to implement this modulo-10 counter, we will need four flip-flops labeled: $Q_3$, $Q_2$, $Q_1$ and $Q_0$. Using natural binary coding, we can derive a state transitions table:

<table>
<thead>
<tr>
<th>$Q_3Q_2$</th>
<th>00</th>
<th>01</th>
<th>11</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00</td>
<td>01</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>01</td>
<td>10</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>11</td>
<td>01</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>10</td>
<td>01</td>
<td>1</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

We obtain the following equations for the implementation:

\[
\begin{align*}
D_0 &= Q_2Q_1Q_0 + Q_3Q'_1Q'_0 \\
D_1 &= Q'_2Q_1Q_0 + Q_2Q'_1 + Q_2Q'_0 \\
D_2 &= Q'_3Q'_1Q_0 + Q_1Q'_0 \\
D_3 &= Q'_0
\end{align*}
\]
Design a simplified traffic-light controller that switches traffic lights on a crossing where a north-south (NS) street intersects an east-west (EW) street. The input to the controller is the \textit{WALK} button pushed by pedestrians who want to cross the street. The outputs are two signals \textit{NS} and \textit{EW} that control the traffic lights in NS and EW directions. When \textit{NS} or \textit{EW} are 0, the red light is on and when they are 1, the green light is on. When there are no pedestrians, \textit{NS} = 0 and \textit{EW} = 1 for 1 minute, followed by \textit{NS} = 1 and \textit{EW} = 0 for 1 minute and so on. When a \textit{WALK} button is pushed, \textit{NS} and \textit{EW} both become 1 for a minute when the present minute expires. After that the \textit{NS} and \textit{EW} signals continue alternating. For the traffic-light controller, (a) develop a state diagram and state/output table, (b) minimize the number of states, (c) encode the states and (d) derive a logic schematic.

(a) We can define three states, \( s_0 \), \( s_1 \) and \( s_2 \) which represent:

\[
\begin{align*}
  s_0 &= (NS = 0, EW = 1) \\
  s_1 &= (NS = 1, EW = 0) \\
  s_2 &= (NS = 1, EW = 1)
\end{align*}
\]

The description may be described in the following next-state output table:

<table>
<thead>
<tr>
<th>State</th>
<th>next-state</th>
<th>( NS )</th>
<th>( EW )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>\textit{walk} = 0</td>
<td>\textit{walk} = 1</td>
<td></td>
</tr>
<tr>
<td>( s_0 )</td>
<td>( s_1 )</td>
<td>( s_2 )</td>
<td>0</td>
</tr>
<tr>
<td>( s_1 )</td>
<td>( s_0 )</td>
<td>( s_2 )</td>
<td>1</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>( s_0 )</td>
<td>( s_2 )</td>
<td>1</td>
</tr>
</tbody>
</table>

The state diagram could be described as:
(b) As you can see below, the number of states in minimal

(c) The encoding would follow:

\[
\begin{align*}
    s_0 &= 00 \\
    s_1 &= 01 \\
    s_2 &= 10
\end{align*}
\]

(d) Using D-type flip-flops, the states can be mapped as follows:

\[
\begin{array}{c|c|c|c}
  Walk & 0 & 1 \\
  \hline
  00 & 0 & 1 & 10 \\
  01 & 0 & 0 & 10 \\
  11 & XX & XX \\
  10 & 0 & 0 & 10 \\
\end{array}
\]

Resulting in the equations:

\[
\begin{align*}
    D_0 &= Q'_1 Q'_0 \text{walk}' \\
    D_1 &= \text{walk} \\
    NS &= Q_1 + Q_0
\end{align*}
\]
\[ EW = Q'_0 \]

Giving us the following implementation:

Logic schematic