One-Hot Encoded
Finite State Machines
Example State Machine

- State A = 00
- State B = 01
- State C = 10
- State D = 11

<table>
<thead>
<tr>
<th>InA</th>
<th>InB</th>
<th>CS</th>
<th>NS</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>A</td>
<td>A</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>A</td>
<td>B</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>B</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>B</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>C</td>
<td>C</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>C</td>
<td>D</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>D</td>
<td>A</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>InA</th>
<th>InB</th>
<th>CS</th>
<th>NS</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>00</td>
<td>00</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>00</td>
<td>01</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>01</td>
<td>00</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>01</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>10</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>11</td>
<td>00</td>
<td>0</td>
</tr>
</tbody>
</table>
Example Machine Implementation

\[
N_1 = Q_1 \cdot Q_0' + Q_1' \cdot Q_0 \cdot \text{InA} \\
N_0 = Q_1 \cdot Q_0' \cdot \text{InB} + Q_1' \cdot Q_0' \cdot \text{InA} \\
Z = Q_1' \cdot Q_0 + Q_1 \cdot Q_0'
\]

9 gates
21 gate inputs
Choose a Different Encoding

- \( A = 1000, \ B = 0100, \ C = 0010, \ D = 0001 \)

<table>
<thead>
<tr>
<th>InA</th>
<th>InB</th>
<th>CS</th>
<th>NS</th>
<th>Z</th>
<th>InA</th>
<th>InB</th>
<th>CS</th>
<th>NS</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-</td>
<td>1000</td>
<td>1000</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>1___</td>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>1000</td>
<td>0100</td>
<td>0</td>
<td>1</td>
<td>-</td>
<td>1___</td>
<td>0100</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-</td>
<td>0100</td>
<td>1000</td>
<td>1</td>
<td>0</td>
<td>-</td>
<td>-1-</td>
<td>1000</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>-</td>
<td>0100</td>
<td>0010</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-1-</td>
<td>0010</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>0010</td>
<td>0010</td>
<td>1</td>
<td>-</td>
<td>0</td>
<td>--1-</td>
<td>0010</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>1</td>
<td>0010</td>
<td>0001</td>
<td>1</td>
<td>-</td>
<td>1</td>
<td>--1-</td>
<td>0001</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>-</td>
<td>0001</td>
<td>1000</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>---1</td>
<td>1000</td>
<td>0</td>
</tr>
</tbody>
</table>

This is called a **one-hot** encoding.

Because of the state encodings, there are many illegal states.

Only one state bit is on at a time

This TT with all these input don't cares is the result
One-Hot Encoding Results

- Will require 4 flip flops
  - One per state
  - Call the current state bits A, B, C, and D
  - Call the next state bits NA, NB, NC, and ND

By inspection we see:

\[
\begin{align*}
NA &= A \cdot \text{In}A' + B \cdot \text{In}A' + D \\
NB &= A \cdot \text{In}A \\
NC &= B \cdot \text{In}A + C \cdot \text{In}B' \\
ND &= C \cdot \text{In}B \\
Z &= B + C
\end{align*}
\]
One-Hot Implementation

9 gates
19 gate inputs
One-Hot - Observations

• Choosing a one-hot encoding results in many, many don’t cares in transition table

• Minimization results in simpler IFL and OFL

• Can do one-hot design by inspection
  – without using transition tables...
Another One-Hot Example

State Encoding

<table>
<thead>
<tr>
<th>State</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>100</td>
</tr>
<tr>
<td>B</td>
<td>010</td>
</tr>
<tr>
<td>C</td>
<td>001</td>
</tr>
</tbody>
</table>
State Encoding and Structure

With one-hot encoding, each state has its own flip flop.

Note: 'A' is the name of a state. It is also the name of the wire coming out from the flip flop for state 'A'.

The same holds true for states 'B' and 'C'.

<table>
<thead>
<tr>
<th>State</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>100</td>
</tr>
<tr>
<td>B</td>
<td>010</td>
</tr>
<tr>
<td>C</td>
<td>001</td>
</tr>
</tbody>
</table>
When is A the next state?

Look at the arcs entering state A

\[ NA = A \cdot x' + B \cdot y'z + C \]
When is A the next state?

Look at the arcs entering state A

NA = A\cdot x' + B\cdot y'z + C
One-Hot Encodings By Inspection

When is A the next state?

Look at the arcs entering state A

\[ NA = A \cdot x' + B \cdot y'z + C \]
When is A the next state?

Look at the arcs entering state A

\[ NA = A \cdot x' + B \cdot y'z + C \]

Similar reasoning leads to:
\[ NB = A \cdot x + B \cdot y'z' \]
\[ NC = B \cdot y \]
The Key Lock Problem - One-Hot Version

There will be 6 flip flops
One for each state
Key Lock Input Forming Logic
Key Lock Input Forming Logic

\[ A^+ = (PUSHED' \cdot A) + (ECNT3' \cdot E) + (WAITDONE \cdot F) + (LOCKED \cdot D) \]
Key Lock Input Forming Logic

\[
A^+ = (PUSHED' \cdot A) + (ECNT3' \cdot E) + (WAITDONE \cdot F) + (LOCKED \cdot D)
\]

\[
B^+ = (PUSHED' \cdot B) + (PUSHED \cdot 7 \cdot A)
\]
Key Lock Input Forming Logic

\[ A^+ = (PUSHED' \cdot A) + (ECNT3' \cdot E) \]
\[ + (WAITDONE \cdot F) \]
\[ + (LOCKED \cdot D) \]

\[ B^+ = (PUSHED' \cdot B) + (PUSHED \cdot 7 \cdot A) \]

\[ C^+ = (PUSHED' \cdot C) + (PUSHED \cdot 8 \cdot B) \]
Key Lock Input Forming Logic

\[
A^+ = (\text{PUSHED}^\prime \bullet A) + (\text{ECNT3}^\prime \bullet E) + (\text{WAITDONE} \bullet F) + (\text{LOCKED} \bullet D)
\]

\[
B^+ = (\text{PUSHED}^\prime \bullet B) + (\text{PUSHED} \bullet 7 \bullet A)
\]

\[
C^+ = (\text{PUSHED}^\prime \bullet C) + (\text{PUSHED} \bullet 8 \bullet B)
\]

\[
D^+ = (\text{LOCKED}^\prime \bullet D) + (\text{PUSHED} \bullet 9 \bullet C)
\]
Key Lock Input Forming Logic

\[
A^+ = (\text{PUSHED} \cdot A') + (\text{ECNT3} \cdot E') + (\text{WAITDONE} \cdot F) + (\text{LOCKED} \cdot D)
\]

\[
B^+ = (\text{PUSHED} \cdot B') + (\text{PUSHED} \cdot 7 \cdot A)
\]

\[
C^+ = (\text{PUSHED} \cdot C') + (\text{PUSHED} \cdot 8 \cdot B)
\]

\[
D^+ = (\text{LOCKED} \cdot D') + (\text{PUSHED} \cdot 9 \cdot C)
\]

\[
E^+ = (\text{PUSHED} \cdot 7' \cdot A) + (\text{PUSHED} \cdot 8' \cdot B) + (\text{PUSHED} \cdot 9' \cdot C)
\]
Key Lock Input Forming Logic

\[ A^+ = (\text{PUSHED}' \bullet A) + (\text{ECNT3}' \bullet E) \]
\[ + (\text{WAITDONE} \bullet F) \]
\[ + (\text{LOCKED} \bullet D) \]
\[ B^+ = (\text{PUSHED}' \bullet B) + (\text{PUSHED} \bullet 7 \bullet A) \]
\[ C^+ = (\text{PUSHED}' \bullet C) + (\text{PUSHED} \bullet 8 \bullet B) \]
\[ D^+ = (\text{LOCKED}' \bullet D) + (\text{PUSHED} \bullet 9 \bullet C) \]
\[ E^+ = (\text{PUSHED} \bullet 7' \bullet A) \]
\[ + (\text{PUSHED} \bullet 8' \bullet B) \]
\[ + (\text{PUSHED} \bullet 9' \bullet C) \]
\[ F^+ = (\text{ECNT3} \bullet E) \]
\[ + (\text{WAITDONE}' \bullet F) \]
Key Lock Output Forming Logic

ERROR = E
Key Lock Output Forming Logic

\[
\text{ERROR} = E \\
\text{INC} = (\text{PUSHED} \cdot 7' \cdot A) + (\text{PUSHED} \cdot 8' \cdot B) + (\text{PUSHED} \cdot 9' \cdot C)
\]
Key Lock Output Forming Logic

ERROR = E
INC = (PUSHED\textbullet7\textbullet A) 
+ (PUSHED\textbullet8\textbullet B) 
+ (PUSHED\textbullet9\textbullet C)
CLRTIMER = ECNT3\textbullet E
Key Lock Output Forming Logic

ERROR = E
INC = (PUSHED\circ7'\circ A)
    + (PUSHED\circ8'\circ B)
    + (PUSHED\circ9'\circ C)
CLRTIMER = ECNT3\circ E
CLRCNTR = (WAITDONE\circ F)
    + ((LOCKED\circ D)
Key Lock Output Forming Logic

ERROR = E
INC = (PUSHED\(7\)'A) + (PUSHED\(8\)'B) + (PUSHED\(9\)'C)
CLRTIMER = ECNT3.E
CLRCNTR = (WAITDONE.F) + ((LOCKED.D)
UNLOCK = (PUSHED\(9\)'C)
Other State Encoding Techniques

• You have learned the 2 extremes
  - Fully encoded (8 states ↔ 3 state bits)
  - One-hot encoded (8 states ↔ 8 state bits)
• A range of options exist in between

• A good choice of encoding
  - Can minimize IFL and OFL complexity
  - Algorithms have been developed for this...
  - Beyond the scope of this class