K-map minimization examples
POS minimization with K-map
Design example
"Switching network" logic blocks (multiplexers/demultiplexers)

Example: BCD increment-by-1

\[
\begin{align*}
O_8 &= I_1I_3I_4 + I_2I_4' \\
O_4 &= I_1I_3' + I_2I_3 + I_2I_1'I_4 \\
O_2 &= I_1I_3I_4' + I_4'I_2 \\
O_1 &= I_4'
\end{align*}
\]

We greatly simplify the logic by using the don't cares.

We need a 4-variable Karnaugh map for each of the 4 output functions.

Example: Two-bit multiplier

\[
\begin{align*}
P_8 &= A_2A_1B_2B_1 \\
P_4 &= A_2B_2B_1B_1' + A_2B_2'B_1 + A_2B_2'B_1' \\
P_2 &= A_2A_1B_1'B_1 \\
P_1 &= A_1B_1
\end{align*}
\]

POS minimization with K-maps

Encircle the zeros in the map
Interpret indices complementary to SOP form

\[
F = (B'+C+D)(B+C+D')(A'+B'+C)
\]

Same idea as with truth tables.
Design example: Comparator

- Comparator: K-maps
  - K-map for LT
  - K-map for EQ
  - K-map for GT
  - LT = A'B'D + A'C + B'CD
  - EQ = A'B'C'D' + A'BC'D + ABCD + AB'CD'
  - GT = ABC'D + AC + ABD'

Comparator: Implementing EQ

- Option 1:
  - EQ = A'B'C'D' + A'BC'D + ABCD + AB'CD'
  - 5 gates but they require lots of inputs
- Option 2:
  - EQ = (A xnor C) • (B xnor D)
  - XNOR is constructed from 3 simple gates
  - 7 gates but they all have 2 inputs each

Comparator: Circuit schematics

Switching networks logic blocks

- Multiplexer (MUX)
  - Routes one of many inputs to a single output
  - Also called a selector

- Demultiplexer (DEMUX)
  - Routes a single input to one of many outputs
  - Also called a decoder

Multiplexers

- Basic concept
  - 2^n data inputs; n control inputs ("selects"); 1 output
  - Connects one of 2^n inputs to the output
  - "Selects" decide which input connects to output
Multiplexers: Truth tables

- Two alternative truth-tables: **Functional** and **Logical**

**Example: A 2:1 Mux**

<table>
<thead>
<tr>
<th>S</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Functional truth table**

<table>
<thead>
<tr>
<th>S</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Logical truth table**

Multiplexers: Implementation

- 2:1 mux: \( Z = S_0 \overline{I} + S_1 I \)
- 4:1 mux: \( Z = S_1 S_0 \overline{I} + S_0 S_1 I + S_2 S_0 \overline{I} + S_1 S_2 I + \ldots \)
- 8:1 mux: \( Z = S_2 S_1 \overline{I} + S_1 S_2 I + S_0 S_2 \overline{I} + \ldots \)

Cascading multiplexers

- Can form large multiplexers from smaller ones (many implementation options)

Multiplexer as logic block

- A 2^n:1 mux can implement any function of n variables as a lookup table

\[
F(A,B,C) = m_0 + m_2 + m_6 + m_7 = A'B'C' + A'B'C + ABC' + ABC
\]

- Can also use a 2^n-1:1 mux to implement a function of n variables
  - (n-1) mux control variables \( S_0 \) – \( S_{n-2} \)
  - One data variable \( S_{n-1} \)
  - Four possible values for each data input: 0, 1, \( S_{n-1} \), \( S_{n-1}' \)
Multiplexer as logic block

- F(A,B,C,D) implemented using an 8:1 mux

Choose A, B, C as control variables
Choose D as a data variable