Karnaugh Map (k-Map)

- A graphical technique for representing & simplifying a Boolean expression from a circuit (as truth table)
- Two-dimensional form of the truth table.
- Karnaugh map provides a neat way to derive minimum-cost implementations of simple logic functions.

Two-variable k-Map

**Step 1:** Truth Table → Karnaugh Map

*Plot all the 1s (minterm) in the function's truth table on the Karnaugh Map.*

Three-variable

**Step 2:** Group the adjacent 1s on the map.

(In groups of 1, 2, 4, 8, 16, 2^n)

**Step 3:** Read the groups of 1s on the map. Express each group as a product term.

\[ f = x_1 \overline{x_2} + \overline{x_1} (\overline{x_2} + x_2) \]

\[ = x_2 + \overline{x_1} \]

Three-variable

Be careful! (Adjacent columns differ by only one variable.)

**Group:** Vertically or horizontally, not diagonally.

Four-variable

\[ f = x_3 + x_1 x_4 \]
Example 1.

Find the minimum-cost sum of products (SOP) form for the function

\[ f(x_1, x_2, x_3) = \Sigma m(1, 4, 5, 7) \]

Using k-Map

\[ f = x_1x_3 + x_1\overline{x_2} + \overline{x_1}x_3 \]

Without using k-map

<table>
<thead>
<tr>
<th>X_1</th>
<th>X_2</th>
<th>X_3</th>
<th>f(x_1, x_2, x_3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>m_0 \overline{x_1}x_2x_3 = 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>m_1 \overline{x_1}x_2x_3 = 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>m_2 \overline{x_1}x_2x_3</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>m_3 \overline{x_1}x_2x_3 = 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>m_4 x_1x_2x_3 = 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>m_5 x_1x_2x_3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>m_6 x_1x_2x_3</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>m_7 x_1x_2x_3 = 1</td>
</tr>
</tbody>
</table>

Originally

\[ f = \overline{x_1}x_2x_3 + \overline{x_1}x_3 + x_1\overline{x_2}x_3 + x_1x_2x_3 + x_1x_2x_3 \]

Example 2.

\[ f = x_1\overline{x_2} + \overline{x_1}x_3 + \overline{x_1}x_2x_4 \]

or

\[ f = x_1\overline{x_3} + \overline{x_1}x_3 + x_2\overline{x_3}x_4 \]
Sequential Logic

- Previously, combinational circuits are only functions of current inputs.
- The output of a sequential circuit is a function of both its current inputs and its past inputs; that is, a sequential circuit has memory.
- The building blocks used to construct devices that store data are called flip-flops.
- Learn basic sequential elements (latch, flip-flop), important to sequential logic.
- Know the applications: counter, registers, and shifters constructed from flip-flops.
- Clock input, which triggers the transition from the current state to the next state.

- Basic Latch: a feedback connection of two NOR gates or two NAND gates (can store 1 bit)
- Gated Latch: basic Latch + clock (control input signal)
- Gated SR Latch
- Gated D Latch
- Level sensitive
- Flip-flop: basic latch + clock
  - SR Flip-flop
  - D Flip-flop
  - T Flip-flop
  - JK Flip-flop

Basic Latch bistable.

- Two cross-coupled NOR gates $X = \overline{A + B}$  (if $A = 0$, $X = \overline{B}$, if $A = 1$, $X = 0$)

- Analyzing a sequential circuit by assuming initial conditions

- When $R = S = 0$, the new output $Q^+$ is simply the old output $Q$.
- The output doesn't change state.

  - When $S = 1, R = 0$, the output is set to 1
  - When $R = 1, S = 0$, the output is reset to 0
  - When $S = R = 1$, undefined condition, should avoid!
Lecture Notes (04/23/2015)

Basic Latch = feedback connection of two NOR gates
Gated Latch = basic latch + clock (control signal)

Gated SR Latch

\[ \text{Characteristic Table} \]
\[
\begin{array}{cccc}
\text{clk} & S & R & Q(t+1) \\
0 & X & X & \text{X (no change)} \\
1 & 0 & 0 & Q(t) \\
1 & 0 & 1 & 0 \leftarrow \text{Reset} \\
1 & 1 & 0 & 1 \leftarrow \text{set} \\
1 & 1 & 1 & \text{X} \leftarrow \text{forbidden} \\
\end{array}
\]

Graphical Symbol

\[ \text{f} \]

or the clock can be expressed as "enable" input

Gated D Latch

\[ \text{Characteristic Table} \]
\[
\begin{array}{cccc}
\text{clk} & D & Q(t+1) \\
0 & X & 0 \text{ Reset} \\
1 & 1 & 1 \text{ Set} \\
\end{array}
\]

Graphical Symbol

\[ \text{f} \]

△ One data input, (avoid forbidden condition)

\[ \begin{align*}
D = 0 \Rightarrow S = 0, R = 1 & \Rightarrow \text{Reset to 0} \\
D = 1 \Rightarrow S = 1, R = 0 & \Rightarrow \text{Set to 1}
\end{align*} \]

(4)
Flip-flop: based on the gated latch principle, output state changed only on the edge of the clock (controlling clock signal)

- Edge-trigger flip-flop
- Master-slave flip-flop (will not talk about in our class)

D flip-flop

JK flip-flop

T flip-flop

**Edge-trigger D Flip-flop**

<table>
<thead>
<tr>
<th>Clk</th>
<th>D</th>
<th>Q(t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>⊕(t)</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**D-Latch vs Edge-triggered D Flip-flop**

Clock

(D-Latch) Qa

(D flip-flop) Qb

Change to 1 according to D state at clock = 1

Change to 0 according to D state at clock ↑

**Time Diagram**

T Flip-flop  \( T \rightarrow \text{Toggle} \)

<table>
<thead>
<tr>
<th>Clk</th>
<th>T</th>
<th>Q(t+1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>X</td>
<td>⊕(t)</td>
</tr>
<tr>
<td>↑</td>
<td>0</td>
<td>⊕(t)</td>
</tr>
<tr>
<td>↑</td>
<td>1</td>
<td>⊕(t)</td>
</tr>
</tbody>
</table>

Q retain its present state if \( T = 0 \)

reverse its present state if \( T = 1 \)

Q: What if \( T \) always equals to 1?

What if \( T \) always equals to 0?
**Lecture Notes (04/23/2015)**

**JK Flip Flop**

\[
\begin{array}{c|c|c}
J & K & Q(t+1) \\
\hline
0 & 0 & Q(t) \\
0 & 1 & 0 \\
1 & 0 & 1 \\
1 & 1 & \bar{Q} \rightarrow T \\
\end{array}
\]

- It combines the behaviors of SR and T flip-flops in a useful way.
- It behaves as the SR flip-flop, where $J=\bar{S}$, $K=R$ for all inputs, except $J=K=1$ for $S=R=1$ (normal condition for SR flip-flop), JK flip flop toggles its state like T flip-flop.

**Register**

- Flip-flop: store a single bit
- Registers: store multiple bits (a set of $n$ flip-flops)

\[
\begin{align*}
\text{In} & \quad D & \quad D & \quad D & \quad D & \quad \overline{D} & \quad \text{Out} \\
\text{clk} & \quad \overline{Q_4} & \quad \overline{Q_3} & \quad \overline{Q_2} & \quad \overline{Q_1} & \quad Q_4
\end{align*}
\]

**4-bit shift register**

<table>
<thead>
<tr>
<th>In</th>
<th>$Q_1$</th>
<th>$Q_2$</th>
<th>$Q_3$</th>
<th>$Q_4$ (Out)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_0$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_1$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_2$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_3$</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$t_4$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$t_5$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
A synchronous counters

\[ \text{frequency divider} \]