Binary Logic and Gates

• Binary variables take on one of two values.
• Logical operators operate on binary values and binary variables.
• Basic logical operators are the logic functions AND, OR and NOT.
• Logic gates implement logic functions.
Logical Operations

- The three basic logical operations are:
  - AND
  - OR
  - NOT

- AND is denoted by a dot (·).
- OR is denoted by a plus (+).
- NOT is denoted by an overbar (‾), a single quote mark (') after, or (~) before the variable.
NOR

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>out = x NOR y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Output value is the complemented output from an “OR” function.

amplitude

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Y(t)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>X(t)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>out(t) = x(t) nor y(t)</td>
</tr>
</tbody>
</table>
XOR (EXCLUSIVE OR)

- The number of inputs that are 1 matter.
- More than two values can be “xor-ed” together.
- General rule: the output is equal to 1 if an odd number of input values are 1 and 0 if an even number of input values are 1.

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>out = x ⊕ y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

amplitude

\[
\begin{array}{c}
0 \ 1 \ 0 \ 1 \\
0 \ 0 \ 1 \ 1 \\
0 \ 1 \ 1 \ 0 \\
\end{array}
\rightarrow
Y(t) \ \ X(t) \ \ out(t) = x(t) \ xor \ y(t)
\]
Truth Tables

- Used to evaluate any logic function
- Consider $F(X, Y, Z) = XY + YZ$

<table>
<thead>
<tr>
<th>$X$</th>
<th>$Y$</th>
<th>$Z$</th>
<th>$XY$</th>
<th>$\overline{Y}$</th>
<th>$\overline{Y}Z$</th>
<th>$F = XY + \overline{Y}Z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
REALIZATION USING NAND

• NOT GATE
  All NAND input pins connect to the input signal A gives an output A’.

One NAND input pin is connected to the input signal A while all other input pins are connected to logic 1. The output will be A’.

(A.A)’=A’

(A.1)’=A’

A’
DeMorgan’s theorems provide mathematical verification of:

– the equivalency of the NAND and negative-OR gates

– the equivalency of the NOR and negative-AND gates.
DEMORGAN’S THEOREMS

• The complement of two or more ANDed variables is equivalent to the OR of the complements of the individual variables.

\[
\overline{X \cdot Y} = \overline{X} + \overline{Y}
\]

• The complement of two or more ORed variables is equivalent to the AND of the complements of the individual variables.

\[
\overline{X + Y} = \overline{X} \cdot \overline{Y}
\]
BOOLEAN OPERATOR PRECEDENCE

- The order of evaluation is:
  1. Parentheses
  2. NOT
  3. AND
  4. OR

- Consequence: Parentheses appear around OR expressions

- Example: \( F = A(B + C)(C + \overline{D}) \)
Expression Simplification

• An application of Boolean algebra
• Simplify to contain the smallest number of literals (variables that may or may not be complemented)

\[ AB + \overline{A}CD + \overline{A}BD + \overline{A}C\overline{D} + ABCD \]

\[ = AB + ABCD + \overline{A}C D + \overline{A}C\overline{D} + \overline{A}B D \]

\[ = AB + AB(CD) + \overline{A}C(D + \overline{D}) + \overline{A}B D \]

\[ = AB + \overline{A}C + \overline{A}B D = B(A + \overline{AD}) + \overline{AC} \]

\[ = B(A + D) + \overline{A}C \text{ (has only 5 literals)} \]
A Simplification Example:

\[ F(A, B, C) = \sum (1, 4, 5, 6, 7) \]

Writing the minterm expression:

\[ F = A \overline{B} C + A \overline{B} C + A \overline{B} C + A B \overline{C} + A B C \]

Simplifying:

\[ F = A \overline{B} C + A (\overline{B} \overline{C} + \overline{B} C + B \overline{C} + B C) \]
\[ F = A \overline{B} C + A (B (C + C) + B (C + C)) \]
\[ F = A \overline{B} C + A (B + B) \]
\[ F = A \overline{B} C + A \]
\[ F = B \overline{C} + A \]

Simplified \( F \) contains 3 literals
FOUR-VARIABLE K-MAPS

• We can do four-variable expressions too!
  – The minterms in the third and fourth columns, and in the third and fourth rows, are switched around.
  – Again, this ensures that adjacent squares have common literals.

• Grouping minterms is similar to the three-variable case, but:
  – You can have rectangular groups of 1, 2, 4, 8 or 16 minterms.
  – You can wrap around all four sides.
**EXAMPLE: SIMPLIFY**

\[ M_0 + M_2 + M_5 + M_8 + M_{10} + M_{13} \]

- The expression is already a sum of minterms, so here’s the K-map:

  ![K-map diagram]

- We can make the following groups, resulting in the MSP \( x'z' + xy'z \).
**K-MAPS CAN BE TRICKY!**

- There may not necessarily be a *unique* MSP. The K-map below yields two valid and equivalent MSPs, because there are two possible ways to include minterm \(m_7\).

- \(y'z + yz' + xy\)

- \(y'z + yz' + xz\)

- Remember that overlapping groups is possible, as shown above.
Maxterm Example

\[ f(A, B, C) = \prod M(1, 2, 4, 6, 7) \]


Note that the complements are \((0, 3, 5)\) which are the minterms of the previous example.
FOUR VARIABLE EXAMPLE
(A) MINTERM FORM. (B) MAXTERM FORM.

\[ f(a, b, Q, G) = \sum m(0, 3, 5, 7, 10, 11, 12, 13, 14, 15) = \prod M(1, 2, 4, 6, 8, 9) \]