CSEE 3827: Fundamentals of Computer Systems

Boolean Logic & Boolean Algebra
Agenda (M&K 2.1-2.2, 2.8-2.9)

- Terminology
- Boolean algebra
- Logic gates
- Circuit fabrication
  - NAND, NOR
- DUAL
- XOR
**Terminology**

- **Digital / Binary / Boolean**: $0 = \text{False}, \ 1 = \text{True}$

- **Binary Variable**: a symbolic representation of a value that might be 0 or 1, e.g., $X, Y, A, B$

- **Complement** (e.g., of a variable $X$): written $\overline{X}$: the opposite value of $X$

<table>
<thead>
<tr>
<th>$X$</th>
<th>$\overline{X}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- **Literal**: a boolean variable or its complement (e.g., $X, \overline{X}, \overline{Y}$)
Boolean Logic

- All logical functions can be implemented in terms of three logical operations:

\[
\begin{array}{c|c}
 \text{NOT} & \bar{x} \\
\hline
 0 & 1 \\
 1 & 0 \\
\end{array}
\quad
\begin{array}{c|c|c}
 \text{AND} & x \cdot y \\
\hline
 0 & 0 & 0 \\
 0 & 1 & 0 \\
 1 & 0 & 0 \\
 1 & 1 & 0 \\
\end{array}
\quad
\begin{array}{c|c|c}
 \text{OR} & x + y \\
\hline
 0 & 0 & 0 \\
 0 & 1 & 1 \\
 1 & 0 & 1 \\
 1 & 1 & 1 \\
\end{array}
\]
Boolean Logic 2

• Precedence rules just like decimal system

• Implied precedence: NOT > AND > OR

• Use parentheses as necessary

\[ AB + C = (AB) + C \]

\[ (\overline{A} + B)C = ((\overline{A}) + B)C \]
Terminology cont’d

- **Expression**: a set of literals (possibly with repeats) combined with logic operations (and possibly ordered by parentheses)

  - e.g., 4 expressions: $AB + C, (AB) + C, (A + B)C, ((A) + B)C$

  - Note: can compliment expressions, too, e.g., $((A) + B)C$

- **Equation**: expression1 = expression2

  - e.g., $\overline{(A + B)C} = \overline{(\overline{A} + B)C}$

- **Function** of (possibly several) variables: an equation where the lefthand side is defined by the righthand side

  $F(A,B,C) = ((\overline{A}) + B)C$
Boolean Logic: Example

Truth Table: all combinations of input variables

\( k \) variables \( \rightarrow 2^k \) input combinations

<table>
<thead>
<tr>
<th>D</th>
<th>X</th>
<th>A</th>
<th>DX + A</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Boolean Logic: Example 2

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>Y</td>
<td>XY + XY</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---------</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
## Boolean Algebra: Identities and Theorems

<table>
<thead>
<tr>
<th>OR</th>
<th>AND</th>
<th>NOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>X + 0 = X</td>
<td>X1 = X</td>
<td>(identity)</td>
</tr>
<tr>
<td>X + 1 = 1</td>
<td>X0 = 0</td>
<td>(null)</td>
</tr>
<tr>
<td>X + X = X</td>
<td>X X = X</td>
<td>(idempotent)</td>
</tr>
<tr>
<td>X + X = 1</td>
<td>X X = 0</td>
<td>(complementarity)</td>
</tr>
<tr>
<td>X = X</td>
<td></td>
<td>(involution)</td>
</tr>
<tr>
<td>X + Y = Y + X</td>
<td>X Y = Y X</td>
<td>(commutativity)</td>
</tr>
<tr>
<td>X + (Y + Z) = (X + Y) + Z</td>
<td>X(YZ) = (XY)Z</td>
<td>(associativity)</td>
</tr>
<tr>
<td>X(Y + Z) = X Y + X Z</td>
<td>X + Y Z = (X + Y)(X + Z)</td>
<td>(distributive)</td>
</tr>
<tr>
<td>X = X Y</td>
<td>X Y = X + Y</td>
<td>(DeMorgan’s theorem)</td>
</tr>
</tbody>
</table>
Boolean Algebra: Example

*Simplify this equation using algebraic manipulation.*

\[ F = \overline{XYZ} + \overline{XYZ} + XZ \]
Boolean Algebra: Example 2

Find the complement of $F$.

\[ F = AB + \overline{AB} \]

\[ \overline{F} = \]
DeMorgan’s Theorem

• Procedure for complementing expressions

• Remove the “big bar” over AND or OR of 2 (or more) functions (e.g., $F \& G$) and replace...

  • AND with OR, OR with AND

  • 1 with 0, 0 with 1

  • function $F$ with $\overline{F}$, $\overline{F}$ with $F$

\[
\overline{FG} = \overline{F} + \overline{G}
\]

\[
\overline{F + G} = \overline{FG}
\]
DeMorgan’s Practice

\[ ABC + ACD + BC \]
Circuit Representation

- Information flows from left to right

- Input(s) all the way on the left, output(s) on the right

These circuits consume area, power, and time

Goal: minimize the amount of circuitry to compute the desired function
F = \overline{XYZ} + \overline{XZ}Y + XZ

\overline{XY}(Z + \overline{Z}) + XZ \quad \text{(by reverse distribution)}

\overline{XY}i + XZ \quad \text{(by complementarity)}

\overline{XY} + XZ \quad \text{(by identity)}
Circuit view

wire connector: black dot signifies wires are connected

(a) $F = \overline{X}YZ + \overline{X}Y\overline{Z} + XZ$

(b) $F = \overline{X}Y + XZ$
Universal gates: NAND, NOR

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z = \overline{xy}</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</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>

Note: the “o” in a circuit represents a NOT (inverter)

Different from “●” which represents wire connector

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z = x+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>
NAND and NOR universal because...

- NOT, AND, OR can each be implemented using only NAND gates
- NOT, AND, OR can each be implemented using only NOR gates

<table>
<thead>
<tr>
<th></th>
<th>NOT Operation</th>
<th>AND Operation</th>
<th>OR Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{A} = A \text{ NAND } A$</td>
<td>$\bar{A} = A \text{ NOR } A$</td>
<td>$AB = A \text{ NAND } B$</td>
<td>$A+B = A \text{ NOR } B$</td>
</tr>
<tr>
<td>$A+B = \bar{A} \text{ NAND } \bar{B}$</td>
<td></td>
<td>$AB = \bar{A} \text{ NOR } \bar{B}$</td>
<td></td>
</tr>
</tbody>
</table>
Duals
Duals

• All boolean expressions have duals

• Any theorem you can prove, you can also prove for its dual

• To form a dual...
  
  • replace AND with OR, OR with AND

  • replace 1 with 0, 0 with 1
What is the dual of this theorem?

\[ \bar{X} + \bar{Y} = \bar{X} \bar{Y} \]
Duals and Complements

\( \overline{X + Y} = \overline{XY} \)  \( \overline{XY} = \overline{X + Y} \)

\( X + Y = \overline{\overline{X + Y}} \)  \( \overline{X + Y} = \overline{\overline{X + Y}} \)

Note: to complement a function, compute its dual and complement literals
“Complement using Dual” example

\[ F = X + A (Z + \overline{X} (Y + W) + \overline{Y} (Z + W)) \]
Can be used for gate manipulation.

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

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

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

\[ X + Y = \overline{XY} \]
Converting circuits to all-NAND (or all-NOR)

- Work from right to left
- When manipulating an (AND or OR) gate, stick in pairs of NOT gates to get it in “appropriate” form
- Isolated NOT gates are easily implemented as a NAND (NOR) gate
- Example manipulations (for NAND gates)
Convert-to-all-NAND example
Convert-to-all-NAND example

Each “o” by itself represents a NOT gate
XOR: the parity operation

- \( X \oplus Y = X\overline{Y} + \overline{X}Y \)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>X \oplus 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>

- In general, represents parity, i.e.,

- \( X_1 \oplus X_2 \oplus X_3 \oplus \ldots \oplus X_k = 1 \) when an odd number of \( X_i = 1 \)