The Place where
$1 + 1 = 1$....
~~or~~

Boolean Functions and Truth Tables
Review of Objectives

After this lecture, you should be able to........

- Simplify Boolean Algebra expressions
- Develop combinatorial logic solutions:
  - Sum of Product
  - Product of Sum
- Configure positive and negative logic circuits
- Identify critical parameters for logic gates from their datasheets
- Carry out “bubble matching”
# The Basic Properties of Boolean Algebra

## Postulates

<table>
<thead>
<tr>
<th>Relationship</th>
<th>Dual</th>
<th>Property</th>
</tr>
</thead>
<tbody>
<tr>
<td>$AB = BA$</td>
<td>$A + B = B + A$</td>
<td>Commutative</td>
</tr>
<tr>
<td>$A (B + C) = AB + AC$</td>
<td>$A + BC = (A + B) (A + C)$</td>
<td>Distributive</td>
</tr>
<tr>
<td>$1A = A$</td>
<td>$0 + A = A$</td>
<td>Identity</td>
</tr>
<tr>
<td>$\overline{A}A = 0$</td>
<td>$A + \overline{A} = 1$</td>
<td>Complement</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Theorems</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$0A = 0$</td>
<td>$1 + A = 1$</td>
<td>Zero and one theorems</td>
</tr>
<tr>
<td>$AA = A$</td>
<td>$A + A = A$</td>
<td>Idempotence</td>
</tr>
<tr>
<td>$A (B C) = (A B) C$</td>
<td>$A + (B + C) = (A + B) + C$</td>
<td>Associative</td>
</tr>
<tr>
<td>$\overline{A} = A$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\overline{AB} = \overline{A} + \overline{B}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\overline{AB + \overline{AC} + BC} = A\overline{B} + \overline{AC}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$A (A + B) = A$</td>
<td>$A + AB = A$</td>
<td>Absorption Theorem</td>
</tr>
</tbody>
</table>

Principle of duality: The dual of a Boolean function is gotten by replacing AND with OR and OR with AND, constant 1s by 0s, and 0s by 1s.

A, B, etc. are Literals; 0 and 1 are constants.
### DeMorgan’s Theorem

<table>
<thead>
<tr>
<th></th>
<th>( A )</th>
<th>( B )</th>
<th>( \overline{A \cdot B} = \overline{A} + \overline{B} )</th>
<th>( \overline{A + B} = \overline{A} \cdot \overline{B} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</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>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

DeMorgan’s theorem: \( A + B = \overline{A + B} = \overline{A} \cdot \overline{B} \)

**Diagram**: $A \overline{B} \implies F = A + B \iff A \overline{B} \overline{F} = \overline{A} \cdot \overline{B}$

**Discussion**: Applying DeMorgan’s theorem by “pushing the bubbles,” and “bubble tricks.”
The Sum-of-Products (SOP) Form

Fig. A.15—Truth Table for The Majority Function

- Transform the function into a two-level AND-OR equation
- Implement the function with an arrangement of logic gates from the set {AND, OR, NOT}
- F is true when A=0, B=1, and C=1, or when A=1, B=0, and C=1, and so on for the remaining cases.
- Represent logic equations by using the sum-of-products (SOP) form
The SOP Form of the Majority Gate

- The SOP form for the 3-input majority gate is:

\[ M = ABC + ABC + ABC + ABC = m3 + m5 + m6 + m7 = \sum (3, 5, 6, 7) \]

- Each of the \(2^n\) terms are called minterms, running from 0 to \(2^n - 1\)

- Note the relationship between minterm number and boolean value.

- Discuss: common-sense interpretation of equation.
A 2-Level AND-OR Circuit that Implements the Majority Function

Discuss: What is the Gate Count?
Notation Used at Circuit Intersections

Connection

No connection

Connection

No connection
A 2-Level OR-AND Circuit that Implements the Majority Function

"Product of Sums" realization

\[ F = (A + B + C) \land (A + B + \overline{C}) \land (A + \overline{B} + C) \land (\overline{A} + B + C) \]
Positive vs. Negative Logic

- Positive logic: truth, or assertion is represented by logic 1, higher voltage; falsity, de- or unassertion, logic 0, is represented by lower voltage.
- Negative logic: truth, or assertion is represented by logic 0, lower voltage; falsity, de- or unassertion, logic 1, is represented by lower voltage.

**Gate Logic: Positive vs. Negative Logic**

**Normal Convention: Positive Logic/Active High**
Low Voltage = 0; High Voltage = 1

**Alternative Convention sometimes used: Negative Logic/Active Low**

Voltage Truth Table

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>low</td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td>high</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>high</td>
<td>high</td>
<td>high</td>
</tr>
</tbody>
</table>

Positive Logic

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</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>1</td>
</tr>
</tbody>
</table>

Negative Logic

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Behavior in terms of Electrical Levels

Two Alternative Interpretations
Positive Logic AND
Negative Logic OR

**Dual Operations**
Positive and Negative Logic (Cont’d.)

Voltage Levels

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>low</td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td>high</td>
<td>low</td>
<td>low</td>
</tr>
<tr>
<td>high</td>
<td>high</td>
<td>high</td>
</tr>
</tbody>
</table>

Positive Logic Levels

\[
\begin{array}{ccc}
A & B & F \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

Negative Logic Levels

\[
\begin{array}{ccc}
A & B & F \\
1 & 1 & 1 \\
1 & 0 & 1 \\
0 & 1 & 1 \\
0 & 0 & 0 \\
\end{array}
\]

Physical AND gate

\[F = A \cdot B\]

Physical NAND gate

\[F = \overline{A \cdot B}\]

Voltage Levels

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>low</td>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>low</td>
<td>high</td>
<td>high</td>
</tr>
<tr>
<td>high</td>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>high</td>
<td>high</td>
<td>low</td>
</tr>
</tbody>
</table>

Positive Logic Levels

\[
\begin{array}{ccc}
A & B & F \\
0 & 0 & 1 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}
\]

Negative Logic Levels

\[
\begin{array}{ccc}
A & B & F \\
1 & 1 & 0 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{array}
\]

\[F = A + B\]
Bubble Matching

- Active low signals are signified by a prime or overbar or /.
- Active high: enable ______
- Active low: enable’, enable, enable/
- Discuss microwave oven control:
- Active high: Heat = DoorClosed \cdot Start
- Active low: ? (hint: begin with AND gate as before.)
Bubble Matching (Cont’d.)

(a) Positive logic $x_0$
Positive logic $x_1$
Positive Logic

(b) Negative logic $x_0$
Negative logic $x_1$
Negative Logic

(c) Negative logic $x_0$
Negative logic $x_1$
Bubble mismatch

(d) Negative logic $x_0$
Negative logic $x_1$
Bubble match

Bubble match
Digital Components

- High level digital circuit designs are normally made using collections of logic gates referred to as components, rather than using individual logic gates. The majority function can be viewed as a component.
- Levels of integration (numbers of gates) in an integrated circuit (IC):
  - Small scale integration (SSI): 10-100 gates.
  - Medium scale integration (MSI): 100 to 1000 gates.
  - Large scale integration (LSI): 1000-10,000 logic gates.
  - Very large scale integration (VLSI): 10,000-upward.
- These levels are approximate, but the distinctions are useful in comparing the relative complexity of circuits.
- Let us consider several useful MSI components:
Objectives Completed

- Reviewed rules of Boolean algebra
- Investigated two combinational logic forms:
  - Sum of Product : SOP
  - Product of Sum : POS
- Distinguished between positive and negative logic
- Identified critical parameters for logic gates from datasheets
- Carried out “bubble matching”
Next time we will....

• Examine logic components

• Describe and apply typical component functions

• Develop a ripple carry adder using logic components