Chapter #2: Two-Level Combinational Logic

Section 2.1, 2.2 -- Logic Functions and Gates
Two-Level Combinational Logic

- Gates and Truth Tables
- Switches
- Minimization
- Boolean Algebra
- Canonical Forms
- Waveforms
- Positive/Negative Logic
### Logic Functions: Boolean Algebra

#### Description
- **NOT**
  - If $X = 0$ then $X' = 1$
  - If $X = 1$ then $X' = 0$

<table>
<thead>
<tr>
<th>$X$</th>
<th>$X'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

#### Description
- **AND**
  - $Z = 1$ if $X$ and $Y$ are both 1

<table>
<thead>
<tr>
<th>$X$</th>
<th>$Y$</th>
<th>$Z$</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>

#### Description
- **OR**
  - $Z = 1$ if $X$ or $Y$ (or both) are 1

<table>
<thead>
<tr>
<th>$X$</th>
<th>$Y$</th>
<th>$Z$</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>1</td>
</tr>
</tbody>
</table>
**Logic Functions: NAND, NOR, XOR, XNOR**

### NAND

**Description**
- $Z = 1$ if $X$ is 0 or $Y$ is 0.

<table>
<thead>
<tr>
<th>Truth Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

### NOR

**Description**
- $Z = 1$ if both $X$ and $Y$ are 0.

<table>
<thead>
<tr>
<th>Truth Table</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>X</strong></td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>
Any Boolean expression can be implemented by NAND, NOR, NOT gates.

In fact, NOT is superfluous.
(NOT = NAND or NOR with both inputs tied together)

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X NOR Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>X NAND Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Logic Functions: XOR, XNOR

**XOR:** X or Y but not both ("inequality", "difference")

**XNOR:** X and Y are the same ("equality", "coincidence")

<table>
<thead>
<tr>
<th>Description</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Z = 1$ if X has a different value than Y</td>
<td>$Z = 1$ if X has the same value as Y</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Gates</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="XOR gate" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Truth Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

(a) XOR

\[ X \oplus Y = XY' + X'Y \]

<table>
<thead>
<tr>
<th>Gates</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="XNOR gate" /></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Truth Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>$X$</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>0</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>1</td>
</tr>
</tbody>
</table>

(b) XNOR

\[ X \oplus Y = X'Y' + XY \]
Logic Functions: Boolean Algebra

Switches

True

False

X

NOT

Switches

false

true

X \cdot Y

AND

Switches

False

True

X + Y

OR
Logic Functions: NAND, NOR, XOR, XNOR

**NAND**

Switches

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>X • Y</td>
</tr>
<tr>
<td>False</td>
<td>X • Y</td>
</tr>
</tbody>
</table>

**NOR**

Switches

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>X + Y</td>
</tr>
<tr>
<td>False</td>
<td>X + Y</td>
</tr>
</tbody>
</table>
Logic Functions: From Expressions to Gates

More than one way to map an expression to gates

\[ T_1 \]

E.g., \[ Z = A' \cdot B' \cdot (C + D) = (A' \cdot (B' \cdot (C + D))) \]

\[ T_2 \]

3-input gate

2-input gates

**Literal**: each appearance of a variable or its complement in an expression

E.g., \[ Z = A \cdot B' \cdot C + A' \cdot B + A' \cdot B \cdot C' + B' \cdot C \]

3 variables, 10 literals
Logic Functions: Rationale for Simplification

Logic Minimization: reduce complexity of the gate level implementation

- reduce number of literals (gate inputs)
- reduce number of gates
- reduce number of levels of gates

fewer inputs implies faster gates in some technologies
fan-ins (number of gate inputs) are limited in some technologies
fewer levels of gates implies reduced signal propagation delays
minimum delay configuration typically requires more gates
number of gates (or gate packages) influences manufacturing costs

Traditional methods:
reduce delay at expense of adding gates

New methods:
trade off between increased circuit delay and reduced gate count
Logic Functions: Alternative Gate Realizations

\[
Z = A'B'C + A'BC + AB'C + ABC'
\]

\[
ABC' + A'C + B'C
\]

Two-Level Realization
(inverters don't count)

\[
((AB) \cdot C') + ((AB)' \cdot C)
\]

Multi-Level Realization

Advantage: Reduced Gate Fan-ins

\[
(AB) \oplus C
\]

Complex Gate: XOR
Advantage: Fewest Gates

TTL Package Counts:
Z1 - three packages (1x 6-inverters, 1x 3-input AND, 1x 3-input OR)
Z2 - three packages (1x 6-inverters, 1x 2-input AND, 1x 2-input OR)
Z3 - two packages (1x 2-input AND, 1x 2-input XOR)
Logic Functions: Boolean Algebra

Algebraic structure consisting of:

set of elements $B$

binary operations $\{+, \cdot\}$

unary operation $\{\}'\$

such that the following axioms hold:

1. $B$ contains at least two elements, $a$, $b$, such that $a \neq b$

2. Closure $a, b$ in $B$,
   (i) $a + b$ in $B$
   (ii) $a \cdot b$ in $B$

3. Commutative Laws: $a, b$ in $B$,
   (i) $a + b = b + a$
   (ii) $a \cdot b = b \cdot a$

4. Identities: $0, 1$ in $B$
   (i) $a + 0 = a$
   (ii) $a \cdot 1 = a$

5. Distributive Laws:
   (i) $a + (b \cdot c) = (a + b) \cdot (a + c)$
   (ii) $a \cdot (b + c) = a \cdot b + a \cdot c$

6. Complement:
   (i) $a + a' = 1$
   (ii) $a \cdot a' = 0$
Gate Logic: Laws of Boolean Algebra

Duality: a dual of a Boolean expression is derived by replacing AND operations by ORs, OR operations by ANDs, constant 0s by 1s, and 1s by 0s (literals are left unchanged).

Any statement that is true for an expression is also true for its dual!

Useful Laws/Theorems of Boolean Algebra:

Operations with 0 and 1:

1. \( X + 0 = X \)
2. \( X + 1 = 1 \)
3. \( X + X = X \)
4. \( (X')' = X \)
5. \( X + X' = 1 \)
6. \( X + Y = Y + X \)

Idempotent Law:

1D. \( X \cdot 1 = X \)
2D. \( X \cdot 0 = 0 \)
3D. \( X \cdot X = X \)

Involution Law:

4D. \( X \cdot X' = 0 \)

Laws of Complementarity:

5D. \( X \cdot X' = 0 \)

Commutative Law:

6D. \( X \cdot Y = Y \cdot X \)
Gate Logic: Laws of Boolean Algebra (cont)

**Associative Laws:**
1. \((X + Y) + Z = X + (Y + Z) = X + Y + Z\)
2. \((X \cdot Y) \cdot Z = X \cdot (Y \cdot Z) = X \cdot Y \cdot Z\)

**Distributive Laws:**
3. \(X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)\)
4. \(X + (Y \cdot Z) = (X + Y) \cdot (X + Z)\)

**Simplification Theorems:**
5. \(X \cdot Y + X \cdot Y' = X\)
6. \(X + X \cdot Y = X\)
7. \((X + Y') \cdot Y = X \cdot Y\)
8. \((X + Y) \cdot (X + Y') = X\)
9. \(X \cdot (X + Y) = X\)
10. \((X \cdot Y') + Y = X + Y\)

**DeMorgan's Law:**
11. \((X + Y + Z + ...)' = X' \cdot Y' \cdot Z' \cdot ...\)
12. \((X \cdot Y \cdot Z \cdot ...) ' = X' + Y' + Z' + ...
13. \{F(X1,X2,...,Xn,0,1,+,\cdot)\}' = \{F(X1',X2',...,Xn',1,0,\cdot,+)\}

**Duality:**
14. \((X + Y + Z + ...)^D = X \cdot Y \cdot Z \cdot ...
15. \{F(X1,X2,...,Xn,0,1,+,\cdot)\}^D = \{F(X1,X2,...,Xn,1,0,\cdot,+)\}

**Theorems for Multiplying and Factoring:**
16. \((X + Y) \cdot (X' + Z) = X \cdot Z + X' \cdot Y\)
17. \((X \cdot Y) + (Y \cdot Z) + (X' \cdot Z) = X \cdot Y + X' \cdot Z\)
18. \((X + Y) \cdot (Y + Z) \cdot (X' + Z) = (X + Y) \cdot (X' + Z)\)

**Consensus Theorem:**
19. \((X \cdot Y) + (Y \cdot Z) + (X' \cdot Z) =\)
Gate Logic: Laws of Boolean Algebra

**DeMorgan's Law**

\[(X + Y)' = X' \cdot Y'\]

NOR is equivalent to AND with inputs complemented

\[(X \cdot Y)' = X' + Y'\]

NAND is equivalent to OR with inputs complemented

DeMorgan's Law can be used to convert AND/OR expressions to OR/AND expressions
Gate Logic: Laws of Boolean Algebra

Apply the laws and theorems to simplify Boolean equations

Example: full adder's carry out function

\[
C_{\text{out}} = A' B \text{ Cin} + A B' \text{ Cin} + A B \text{ Cin}' + A B \text{ Cin}
\]

=
Gate Logic: Laws of Boolean Algebra

Apply the laws and theorems to simplify Boolean equations

Example: full adder's carry out function

\[
\begin{align*}
  \text{Cout} &= A' \cdot B \cdot \text{Cin} + A \cdot B' \cdot \text{Cin} + A \cdot B \cdot \text{Cin}' + A \cdot B \cdot \text{Cin} \\
  &= A' \cdot B \cdot \text{Cin} + A \cdot B' \cdot \text{Cin} + A \cdot B \cdot \text{Cin}' + A \cdot B \cdot \text{Cin} + A \cdot B \cdot \text{Cin} \\
  &= A' \cdot B \cdot \text{Cin} + A \cdot B \cdot \text{Cin} + A \cdot B' \cdot \text{Cin} + A \cdot B \cdot \text{Cin}' + A \cdot B \cdot \text{Cin} \\
  &= (A' + A) \cdot B \cdot \text{Cin} + A \cdot B' \cdot \text{Cin} + A \cdot B \cdot \text{Cin}' + A \cdot B \cdot \text{Cin} \\
  &= (1) \cdot B \cdot \text{Cin} + A \cdot B' \cdot \text{Cin} + A \cdot B \cdot \text{Cin}' + A \cdot B \cdot \text{Cin} \\
  &= B \cdot \text{Cin} + A \cdot B' \cdot \text{Cin} + A \cdot B \cdot \text{Cin}' + A \cdot B \cdot \text{Cin} \\
  &= B \cdot \text{Cin} + A \cdot B \cdot (\text{Cin}' + \text{Cin}) \\
  &= B \cdot \text{Cin} + A \cdot \text{Cin} + A \cdot B \cdot (1) \\
  &= B \cdot \text{Cin} + A \cdot \text{Cin} + A \cdot B
\end{align*}
\]
Gate Logic: Laws of Boolean Algebra

Proving theorems via axioms of Boolean Algebra:

E.g., prove the theorem: \( X \cdot Y + X \cdot Y' = X \)

E.g., prove the theorem: \( X + X \cdot Y = X \)
Gate Logic: Laws of Boolean Algebra

Example: -- Apply DeMorgan’s Theorem

\[ Z = A' B' C + A' B C + A B' C + A B C' \]

\[ Z' = \]

\[ Z' = (A + B + C') \cdot (A + B' + C') \cdot (A' + B + C') \cdot (A' + B' + C) \]
Gate Logic: 2-Level Canonical Forms

Truth table is the unique signature of a Boolean function

Many alternative expressions (and gate realizations) may have the same truth table

Canonical form: standard form for a Boolean expression provides a unique algebraic signature

Sum of Products Form
also known as disjunctive normal form, minterm expansion

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

\[ F = \] 
\[ F' = \]
### Gate Logic: Two Level Canonical Forms

**Sum of Products**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>Minterms</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( \overline{A} \overline{B} \overline{C} = m_0 )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( \overline{A} \overline{B} C = m_1 )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( \overline{A} B \overline{C} = m_2 )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( \overline{A} B C = m_3 )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( A \overline{B} \overline{C} = m_4 )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( A \overline{B} C = m_5 )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( A B \overline{C} = m_6 )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( A B C = m_7 )</td>
</tr>
</tbody>
</table>

**product term / minterm:**

ANDed product of literals in which each variable appears exactly once, in true or complemented form (but not both!)

**F in canonical form:**

\[
F(A,B,C) = \sum m(3,4,5,6,7) = m_3 + m_4 + m_5 + m_6 + m_7 = A' B C + A B' C' + A B' C + A B C'
\]

**F'(A,B,C) = \sum m(0,1,2)**

**Shorthand Notation for Minterms of 3 Variables**
canonical form/minimal form

\[ F = A' \, B \, C \, + \, A \, B' \, C' \, + \, A \, B' \, C \, + \, A \, B \, C' \, + \, A \, B \, C \]

\[ F' = \]
Gate Logic: 2 Level Canonical Forms

Product of Sums / Conjunctive Normal Form / Maxterm Expansion

Maxterm:
ORed sum of literals in which each variable appears exactly once in either true or complemented form, but not both!

Maxterm form:
Find truth table rows where F is 0
0 in input column implies true literal
1 in input column implies complemented literal

Maxterm Shorthand Notation for a Function of Three Variables

F(A,B,C) = ΠM(0,1,2) = M0 • M1 • M2
= (A + B + C) (A + B + C') (A + B' + C)

F'(A,B,C) = ΠM(3,4,5,6,7) = M3 • M4 • M5 • M6 • M7
= (A + B' + C') (A' + B + C) (A' + B + C') (A' + B' + C) (A' + B' + C')
Gate Logic: Incompletely Specified Functions

n input functions have $2^n$ possible input configurations

for a given function, not all input configurations may be possible

this fact can be exploited during circuit minimization!

E.g., Binary Coded Decimal Digit Increment by 1

BCD digits encode the decimal digits 0 - 9 in the bit patterns $0000_2 - 1001_2$

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>W</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<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>1</td>
<td>1</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>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</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>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
</tbody>
</table>

These input patterns should never be encountered in practice associated output values are "Don't Cares"
Gate Logic: Incompletely Specified Functions

Don't Cares and Canonical Forms

Canonical Representations of the BCD Increment by 1 Function:

\[ Z = m_0 + m_2 + m_4 + m_6 + m_8 + d_{10} + d_{11} + d_{12} + d_{13} + d_{14} + d_{15} \]

\[ Z = \sum m(0, 2, 4, 6, 8) + \sum d(10, 11, 12, 13, 14, 15) \]

\[ Z = M_1 \cdot M_3 \cdot M_5 \cdot M_7 \cdot M_9 \cdot D_{10} \cdot D_{11} \cdot D_{12} \cdot D_{13} \cdot D_{14} \cdot D_{15} \]

\[ Z = \Pi M(1, 3, 5, 7, 9) + \Pi D(10, 11, 12, 13, 14, 15) \]
Gate Logic: Two-Level Canonical Forms

Mapping Between Forms

1. Minterm to Maxterm conversion:
   rewrite minterm shorthand using maxterm shorthand
   replace minterm indices with the indices not already used

   E.g., \( F(A,B,C) = \Sigma m(3,4,5,6,7) = \Pi M(0,1,2) \)

2. Maxterm to Minterm conversion:
   rewrite maxterm shorthand using minterm shorthand
   replace maxterm indices with the indices not already used

   E.g., \( F(A,B,C) = \Pi M(0,1,2) = \Sigma m(3,4,5,6,7) \)

3. Minterm expansion of \( F \) to Minterm expansion of \( F' \):
   in minterm shorthand form, list the indices not already used in \( F \)

   E.g., \( F(A,B,C) = \Sigma m(3,4,5,6,7) \) \( \longrightarrow \) \( F'(A,B,C) = \Sigma m(0,1,2) \)
   \( = \Pi M(0,1,2) \) \( \longrightarrow \) \( = \Pi M(3,4,5,6,7) \)

4. Minterm expansion of \( F \) to Maxterm expansion of \( F' \):
   rewrite in Maxterm form, using the same indices as \( F \)

   E.g., \( F(A,B,C) = \Sigma m(3,4,5,6,7) \) \( \longrightarrow \) \( F'(A,B,C) = \Pi M(3,4,5,6,7) \)
   \( = \Pi M(0,1,2) \) \( \longrightarrow \) \( = \Sigma m(0,1,2) \)
Gate Logic: Two-Level Canonical Forms

Four Alternative Implementations of F:

**Canonical Sum of Products**

\[ F_1 = A'BC + AB'C' + AB'C + ABC' + ABC \]
\[ \Sigma m(3,4,5,6,7) \]

**Minimized Sum of Products**

\[ F_2 = BC + A \]

**Canonical Products of Sums**

\[ F_3 = (A+B+C) \cdot (A+B+C') \cdot (A+B'+C) \]
\[ \Pi M(0,1,2) \]

**Minimized Products of Sums**

\[ F_4 = (A+B) \cdot (A+C) \]
Eight Unique Combinations of Three Inputs

Except for timing glitches, output waveforms of the three implementations are essentially identical.
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

© R.H. Katz Transparency No. 3-29
**Gate Logic: Positive vs. Negative Logic**

**Conversion from Positive to Negative Logic**

<table>
<thead>
<tr>
<th>Voltage Truth Table</th>
<th>Positive Logic</th>
<th>Negative Logic</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>F</td>
</tr>
<tr>
<td>low</td>
<td>low</td>
<td>high</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>low</td>
</tr>
</tbody>
</table>

- Positive Logic NOR:
  - Truth Table

- Negative Logic NAND:
  - Truth Table

**Dual operations:**
AND becomes OR, OR becomes AND
Complements remain unchanged

© R.H. Katz  Transparency No. 3-30
Homework Assignment

HW #3 -- Chapter 2: Sections 2.1 and 2.2