L2: Combinational Logic Design
(Construction and Boolean Algebra)

The Inverter

Truth Table

<table>
<thead>
<tr>
<th>IN</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

- Large noise margins protect against various noise sources

\[ N_{M_L} = V_{IL} - V_{OL} \]
\[ N_{M_H} = V_{OH} - V_{IH} \]
TTL Logic Style (1970’s-early 80’s)

74LS04
(courtesy TI)
MOS Technology: The NMOS Switch

Switch Model

ON: $V_{GS} > V_T$

OFF: $V_{GS} < V_T$

$V_T = 1V$

NMOS ON when Switch Input is High
PMOS: The Complementary Switch

Switch Model

OFF

$V_{GS} > V_T$

$R_{PMOS}$

ON

$V_{GS} < V_T$

$R_{PMOS}$

$V_T = -1V$

PMOS ON when Switch Input is Low
The CMOS Inverter

Switch Model

Rail-to-rail Swing in CMOS
There are 16 possible functions of 2 input variables:

### Truth Table

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0000</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0011</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1100</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1111</td>
</tr>
</tbody>
</table>

**Possible Functions:**
- **X AND Y**
- **X OR Y**
- **X XOR Y**
- **X NOR Y**
- **NOT (X OR Y)**
- **NOT X**
- **NOT Y**
- **X NAND Y**
- **X = Y**

### General Formula

In general, there are \(2^{(2^n)}\) functions of \(n\) inputs.

---

L2: 6.111 Spring 2004

Introductory Digital Systems Laboratory
## Common Logic Gates

<table>
<thead>
<tr>
<th>Gate</th>
<th>Symbol</th>
<th>Truth-Table</th>
<th>Expression</th>
</tr>
</thead>
</table>
| NAND | ![NAND Symbol](image1) | $\begin{array}{ccc} 
X & Y & Z \\
0 & 0 & 1 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{array}$ | $Z = X \cdot Y$ |
| AND | ![AND Symbol](image2) | $\begin{array}{ccc} 
X & Y & Z \\
0 & 0 & 0 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}$ | $Z = X \cdot Y$ |
| NOR | ![NOR Symbol](image3) | $\begin{array}{ccc} 
X & Y & Z \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 0 \\
\end{array}$ | $Z = X + Y$ |
| OR | ![OR Symbol](image4) | $\begin{array}{ccc} 
X & Y & Z \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 1 \\
\end{array}$ | $Z = X + Y$ |
Exclusive (N)OR Gate

**XOR**

\[(X \oplus Y)\]

- X
- Y
- Z

<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>0</td>
</tr>
</tbody>
</table>

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

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

**XNOR**

\[(X \oplus Y)\]

- X
- Y
- Z

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>Z</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>1</td>
</tr>
</tbody>
</table>

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

X and Y the same ("equality")

*Widely used in arithmetic structures such as adders and multipliers*
**Generic CMOS Recipe**

Note: CMOS gates result in inverting functions! (easier to build NAND vs. AND)

---

**How do you build a 2-input NOR Gate?**

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>PDN</th>
<th>PUN</th>
<th>O</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0ff</td>
<td>0n</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0ff</td>
<td>0n</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0ff</td>
<td>0n</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0n</td>
<td>0ff</td>
<td>0</td>
</tr>
</tbody>
</table>
Theorems of Boolean Algebra (I)

- **Elementary**
  1. $X + 0 = X$
  2. $X + 1 = 1$
  3. $X + X = X$
  4. $(\overline{X}) = X$
  5. $X + \overline{X} = 1$
  6. $X + 1 = 1$
  7. $X \cdot 1 = X$
  8. $X \cdot 0 = 0$
  9. $X \cdot X = X$
  10. $X \cdot X = X$
  11. $(\overline{X}) = X$

- **Commutativity:**
  6. $X + Y = Y + X$
  6D. $X \cdot Y = Y \cdot X$

- **Associativity:**
  7. $(X + Y) + Z = X + (Y + Z)$
  7D. $(X \cdot Y) \cdot Z = X \cdot (Y \cdot Z)$

- **Distributivity:**
  8. $X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$
  8D. $X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$

- **Uniting:**
  9. $X \cdot Y + X \cdot \overline{Y} = X$
  9D. $(X + Y) \cdot (X + \overline{Y}) = X$

- **Absorption:**
  10. $X + X \cdot Y = X$
  10D. $X \cdot (X + Y) = X$
  11. $(X + \overline{Y}) \cdot Y = X \cdot Y$
  11D. $(X \cdot \overline{Y}) + Y = X + Y$
Theorems of Boolean Algebra (II)

- **Factoring:**
  12. \((X \cdot Y) + (X \cdot Z) = X \cdot (Y + Z)\)
  12D. \((X + Y) \cdot (X + Z) = X + (Y \cdot Z)\)

- **Consensus:**
  13. \((X \cdot Y) + (Y \cdot Z) + (X \cdot Z) = X \cdot Y \cdot Z\)
  13D. \((X + Y) \cdot (Y + Z) \cdot (X + Z) = (X + Y) \cdot (X + Z)\)

- **De Morgan's:**
  14. \((X + Y + \ldots) = \overline{X} \cdot \overline{Y} \cdot \ldots\)
  14D. \((X \cdot Y \cdot \ldots) = \overline{X} + \overline{Y} + \ldots\)

- **Generalized De Morgan's:**
  15. \(f(X_1,X_2,\ldots,X_n,0,1,+,\cdot) = f(X_1,X_2,\ldots,X_n,1,0,\cdot,+)\)

- **Duality**
  - Dual of a Boolean expression is derived by replacing \(\cdot\) by +, + by \(\cdot\), 0 by 1, and 1 by 0, and leaving variables unchanged
  - \(f(X_1,X_2,\ldots,X_n,0,1,+,\cdot) \Leftrightarrow f(X_1,X_2,\ldots,X_n,1,0,\cdot,+)\)
Simple Example: One Bit Adder

1-bit binary adder
- inputs: A, B, Carry-in
- outputs: Sum, Carry-out

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>Cin</th>
<th>S</th>
<th>Cout</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</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>0</td>
<td>1</td>
</tr>
<tr>
<td>1</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>1</td>
</tr>
</tbody>
</table>

Sum-of-Products Canonical Form

- \( S = \overline{A} \overline{B} \overline{Cin} + \overline{A} B \overline{Cin} + A \overline{B} \overline{Cin} + A B \overline{Cin} \)
- \( Cout = \overline{A} B \overline{Cin} + A \overline{B} \overline{Cin} + A B \overline{Cin} + A B \overline{Cin} \)

Product term (or minterm)
- ANDed product of literals – input combination for which output is true
- Each variable appears exactly once, in true or inverted form (but not both)
Simplify Boolean Expressions

\[ C_{out} = \overline{A} \overline{B} C_{in} + A \overline{B} C_{in} + \overline{A} B C_{in} + A B C_{in} \]
\[ = \overline{A} \overline{B} C_{in} + A \overline{B} C_{in} + \overline{A} B C_{in} + A B C_{in} \]
\[ = (\overline{A} + A) B C_{in} + A (\overline{B} + B) C_{in} + A B (C_{in} + C_{in}) \]
\[ = B C_{in} + A C_{in} + A B \]
\[ = (B + A) C_{in} + A B \]

\[ S = \overline{A} \overline{B} C_{in} + \overline{A} B C_{in} + \overline{A} B C_{in} + A B C_{in} \]
\[ = (A B + A B) C_{in} + (A B + A B) C_{in} \]
\[ = (A \oplus B) C_{in} + (A \oplus B) C_{in} \]
\[ = A \oplus B \oplus C_{in} \]
# Sum-of-Products & Product-of-Sum

## Product term (or minterm): ANDed product of literals – input combination for which output is true

<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} ) m0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( \overline{A} \overline{B} \overline{C} ) m1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( \overline{A} \overline{B} \overline{C} ) m2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( \overline{A} \overline{B} \overline{C} ) m3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( \overline{A} \overline{B} \overline{C} ) m4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( \overline{A} \overline{B} \overline{C} ) m5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( \overline{A} \overline{B} \overline{C} ) m6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( \overline{A} \overline{B} \overline{C} ) m7</td>
</tr>
</tbody>
</table>

F in canonical form:

\[ F(A, B, C) = \Sigma m(1,3,5,6,7) = m1 + m3 + m5 + m6 + m7 \]

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

Canonical form ≠ minimal form

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

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

## Sum term (or maxterm) - ORed sum of literals – input combination for which output is false

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>maxterms</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( A + B + C ) M0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( A + B + \overline{C} ) M1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( A + \overline{B} + C ) M2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( A + \overline{B} + \overline{C} ) M3</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( A + B + C ) M4</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( \overline{A} + B + \overline{C} ) M5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( \overline{A} + \overline{B} + C ) M6</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( \overline{A} + \overline{B} + \overline{C} ) M7</td>
</tr>
</tbody>
</table>

F in canonical form:

\[ F(A, B, C) = \Pi M(0,2,4) = M0 \cdot M2 \cdot M4 \]

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

Canonical form ≠ minimal form

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

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

\[ F(A, B, C) = (A + C)(B + C) \]
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., \( F(A,B,C) = \Sigma m(3,4,5,6,7) \quad \rightarrow \quad F'(A,B,C) = \Sigma m(0,1,2) = \Pi M(3,4,5,6,7) \)

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

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

- Key tool to simplification: $A (\overline{B} + B) = A$

- Essence of simplification of two-level logic
  - Find two element subsets of the ON-set where only one variable changes its value – this single varying variable can be eliminated and a single product term used to represent both elements

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

- $B$ has the same value in both on-set rows
- $B$ remains

- $A$ has a different value in the two rows
- $A$ is eliminated
Boolean Cubes

- Just another way to represent truth table
- Visual technique for identifying when the unifying theorem can be applied
- n input variables = n-dimensional "cube"
## Uniting theorem

Circled group of the on-set is called the \textit{adjacency} plane. Each adjacency plane corresponds to a product term.

- ON-set = solid nodes
- OFF-set = empty nodes

\[ A \text{ varies within face, } B \text{ does not} \]

\begin{center}
\begin{tabular}{c|c|c}
A & B & F \\
\hline
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
\end{tabular}
\end{center}

## Three variable example: Binary full-adder carry-out logic

The on-set is completely covered by the combination (OR) of the subcubes of lower dimensionality - note that “111” is covered three times.

\[ A(B + \overline{B})Cin \]

\[ AB(\overline{Cin} + Cin) \]

\[ Cout = BCin + AB + ACin \]
Higher Dimension Cubes

F(A,B,C) = Σm(4,5,6,7)

- on-set forms a square
  - i.e., a cube of dimension 2 (2-D adjacency plane)
  - represents an expression in one variable
    - i.e., 3 dimensions – 2 dimensions
  - A is asserted (true) and unchanged
  - B and C vary

This subcube represents the literal A

- In a 3-cube (three variables):
  - 0-cube, i.e., a single node, yields a term in 3 literals
  - 1-cube, i.e., a line of two nodes, yields a term in 2 literals
  - 2-cube, i.e., a plane of four nodes, yields a term in 1 literal
  - 3-cube, i.e., a cube of eight nodes, yields a constant term "1"

- In general,
  - m-subcube within an n-cube (m < n) yields a term with n – m literals
Karnaugh Maps

- Alternative to truth-tables to help visualize adjacencies
  - Guide to applying the uniting theorem - On-set elements with only one variable changing value are adjacent unlike in a linear truth-table

- Numbering scheme based on Gray–code
  - e.g., 00, 01, 11, 10 (only a single bit changes in code for adjacent map cells)
K-Map Examples

Cout =

F(A,B,C) = \Sigma m(0,4,5,7)

F =

F(A,B,C) =

F'(A,B,C) = \Sigma m(1,2,3,6)

F' =

F' simply replace 1's with 0's and vice versa
Four Variable Karnaugh Map

F(A,B,C,D) = \Sigma m(0,2,3,5,6,7,8,10,11,14,15)

F = C + \overline{A} B D + B \overline{D}

Find the smallest number of the largest possible subcubes that cover the ON-set

K-map Corner Adjacency Illustrated in the 4-Cube
K-Map Example: Don't Care

Don't Cares can be treated as 1's or 0's if it is advantageous to do so.

\[ F(A,B,C,D) = \Sigma m(1,3,5,7,9) + \Sigma d(6,12,13) \]

Without don't cares:

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

With don't cares:

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

By treating this DC as a "1", a 2-cube can be formed rather than one 0-cube.

In PoS form: \( F = D (\overline{A} + \overline{C}) \)

Equivalent answer as above, but fewer literals.
Hazards

Static Hazards: Consider this function:

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

![Gate Diagram]

\( A = B = 1 \)

\( C \)

\( 1 \)

\( 2 \)

\( F \)

\[ \begin{array}{c|c|c|c|c}
    A \ B & 00 & 01 & 11 & 10 \\
    \hline
    0 \ C & 0 & 0 & 1 & 1 \\
    1 & 0 & 1 & 1 & 0 \\
\end{array} \]

Implemented with MSI gates:

![Implemented Gate Diagram]

\( + \)

\( F \)

\( \rightarrow \) Gate Delay

\( \leftarrow \) Glitch
The glitch is the result of timing differences in parallel data paths. It is associated with the function jumping between groupings or product terms on the K–map. To fix it, cover it up with another grouping or product term.

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

- In general, it is difficult to avoid hazards – need a robust design methodology to deal with hazards.