L2: Combinational Logic Design
(Construction and Boolean Algebra)

Acknowledgements:


Lecture Based on Notes by Professor Anantha Chandrakasan
Large noise margins protect against various noise sources

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>

NML = VIL - VOL
NMH = VOH - VIH
TTL Logic Style (1970’s-early 80’s)

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

NMOS ON when Switch Input is High

Switch Model

OFF

ON

R_{NMOS}

V_{GS} < V_T

V_{GS} > V_T

V_T = 0.5V

N+ N+

P-substrate

source

drain

gate

D

G

S
NMOS Device Characteristics

- MOS is a very non-linear.
- Switch-resistor model sufficient for first order analysis.
PMOS: The Complementary Switch

Switch Model

PMOS ON when Switch Input is Low

RPMOS

VGS > VT

ON

VDD

VT = -0.5V

OFF

VGS < VT

RPMOS
The CMOS Inverter

Switch Model

Rail-to-rail Swing in CMOS
CMOS gates have:
- Rail-to-rail swing (0V to $V_{DD}$)
- Large noise margins
- “zero” static power dissipation
There are 16 possible functions of 2 input variables:

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>16 possible functions (F_0–F_{15})</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0 0 0 0 1 1 1 1 0 0 0 0 0 0 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0 0 1 0 0 0 1 1 0 0 0 0 1 1 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0 0 1 1 1 0 0 1 0 1 0 1 1 1 1</td>
</tr>
</tbody>
</table>

In general, there are 2^{2^n} functions of n inputs
Common Logic Gates

**Gate** | **Symbol** | **Truth-Table** | **Expression**
---|---|---|---
NAND | ![NAND Symbol](image) | ![NAND Truth-Table](image) | \( Z = X \cdot Y \)
AND | ![AND Symbol](image) | ![AND Truth-Table](image) | \( Z = X \cdot Y \)
NOR | ![NOR Symbol](image) | ![NOR Truth-Table](image) | \( Z = X + Y \)
OR | ![OR Symbol](image) | ![OR Truth-Table](image) | \( Z = X + Y \)
Exclusive (N)OR Gate

**XOR**

\[ (X \oplus Y) \]

- **Symbol:** \( \oplus \)
- **Truth Table:**

<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**

\[ (X \oplus Y) \]

- **Symbol:** \( \overline{\oplus} \)
- **Truth Table:**

<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 multiplexers**
**Generic CMOS Recipe**

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>Off</td>
<td>Off</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Off</td>
<td>On</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Off</td>
<td>On</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>On</td>
<td>Off</td>
<td>0</td>
</tr>
</tbody>
</table>

Note: CMOS gates result in inverting functions! (easier to build NAND vs. AND)
Theorems of Boolean Algebra (I)

**Elementary**

1. $X + 0 = X$
2. $X + 1 = 1$
3. $X + X = X$
4. $(\overline{X}) = X$
5. $X + \overline{X} = 1$

1D. $X \cdot 1 = X$
2D. $X \cdot 0 = 0$
3D. $X \cdot X = X$
5D. $X \cdot \overline{X} = 0$

**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. $X + X \cdot Y = X$
11. $(X + \overline{Y}) \cdot Y = X \cdot Y$

10D. $X \cdot (X + Y) = X$
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. \((X \cdot Y) + (Y \cdot Z) + (X \cdot Z) = X \cdot Y + X \cdot Z\)
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

\[ S = \overline{A} \overline{B} \text{ Cin} + \overline{A} \overline{B} \overline{\text{ Cin}} + A \overline{B} \overline{\text{ Cin}} + A \overline{B} \text{ Cin} \]

\[ \text{Cout} = \overline{A} \overline{B} \text{ Cin} + A \overline{B} \overline{\text{ Cin}} + A \overline{B} \overline{\text{ Cin}} + A \overline{B} \text{ Cin} \]

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

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

\[
S = \overline{A} \overline{B} \overline{C_{\text{in}}} + \overline{A} \overline{B} \overline{C_{\text{in}}} + A \overline{B} \overline{C_{\text{in}}} + A \overline{B} \overline{C_{\text{in}}}
\]
\[
= (\overline{A} \overline{B} + A \overline{B}) \overline{C_{\text{in}}} + (A \overline{B} + \overline{A} \overline{B}) \overline{C_{\text{in}}}
\]
\[
= (A \oplus B) \overline{C_{\text{in}}} + (A \oplus B) \overline{C_{\text{in}}}
\]
\[
= A \oplus B \oplus C_{\text{in}}
\]
**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} )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( \overline{A} \overline{B} \overline{C} )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( \overline{A} \overline{B} \overline{C} )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( \overline{A} \overline{B} \overline{C} )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( \overline{A} \overline{B} \overline{C} )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( \overline{A} \overline{B} \overline{C} )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( \overline{A} \overline{B} \overline{C} )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( \overline{A} \overline{B} \overline{C} )</td>
</tr>
</tbody>
</table>

**F in canonical form:**

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

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

**Product 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 )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( A + B + \overline{C} )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>( A + \overline{B} + C )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>( A + \overline{B} + \overline{C} )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( A + B + C )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>( \overline{A} + B + \overline{C} )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>( \overline{A} + \overline{B} + C )</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>( \overline{A} + \overline{B} + \overline{C} )</td>
</tr>
</tbody>
</table>

**F in canonical form:**

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

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

**Sum-of-Products & Product-of-Sum**

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

**F in canonical form:**

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

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

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

**F in canonical form:**

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

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

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

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

- B has the same value in both on-set rows
- B remains
- A has a different value in the two rows
- A is eliminated
- Just another way to represent truth table
- Visual technique for identifying when the uniting theorem can be applied
- \( n \) input variables = \( n \)-dimensional "cube"
Uniting theorem

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

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

A varies within face, B does not

This face represents the literal $\overline{B}$

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.
Higher Dimension Cubes

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

$F = \sum m(1, 2, 3, 6)$

$F' = F'$ simply replace 1's with 0's and vice versa

$F'(A, B, C) = \sum m(1, 2, 3, 6)$
Four Variable Karnaugh Map

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

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

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 Cares

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)

F = \overline{A} \ D + \overline{B} \ \overline{C} \ D \ w/o \ don't \ cares

F = \overline{C} \ D + \overline{A} \ D \ w/ \ don't \ cares

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:

\[ F = A \cdot C + B \cdot C \]

A \hspace{1cm} \hspace{1cm} C \hspace{1cm} \hspace{1cm} B \hspace{1cm} \hspace{1cm} F

\begin{array}{c|cccc}
C & 00 & 01 & 11 & 10 \\
\hline
0 & 0 & 0 & 1 & 1 \\
1 & 0 & 1 & 1 & 0 \\
\end{array}

Implemented with MSI gates:

A \hspace{1cm} C \hspace{1cm} B \hspace{1cm} F

\rightarrow \hspace{1cm} \text{Gate Delay} \hspace{1cm} \text{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!

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