



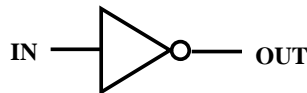
L2: Combinational Logic Design: Construction and Boolean Algebra



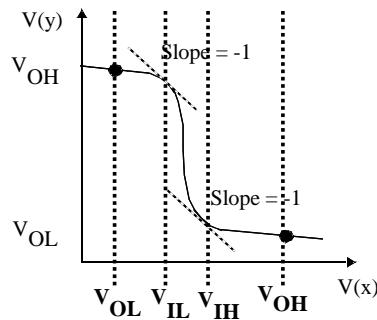
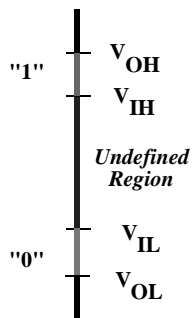
Lecture material derived from R. Katz, "Contemporary Logic Design", Addison Wesley Publishing Company, Reading, MA, 1993.



The Inverter



IN	OUT
0	1
1	0



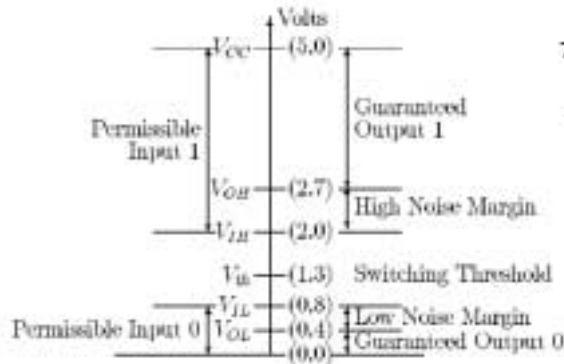
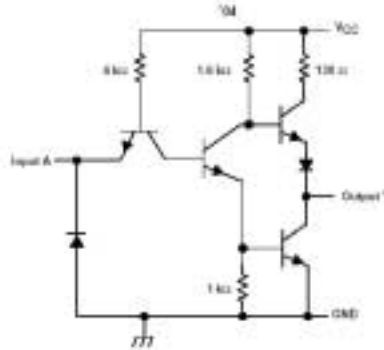
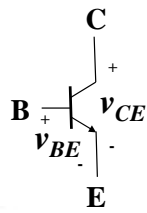
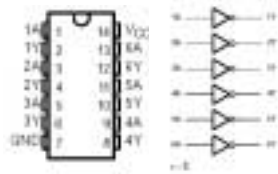
$$NM_L = V_{IL} - V_{OH}$$

$$NM_H = V_{OL} - V_{IH}$$

- Large noise margins protect against various noise sources



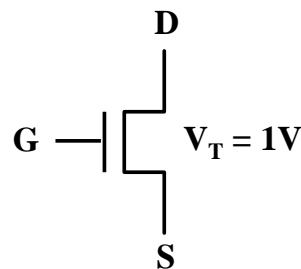
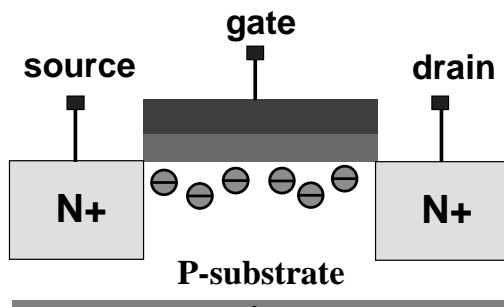
TTL Logic Style (1970's-early 80's)



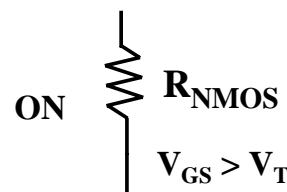
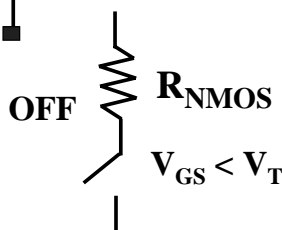
74LS04 (courtesy TI)



MOS Technology: The NMOS Switch



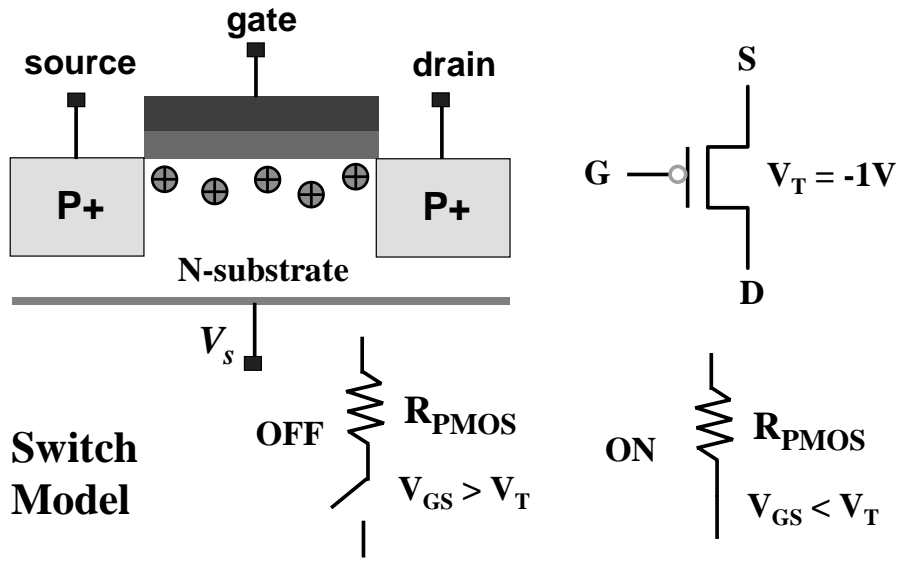
Switch Model



NMOS ON when Switch Input is High



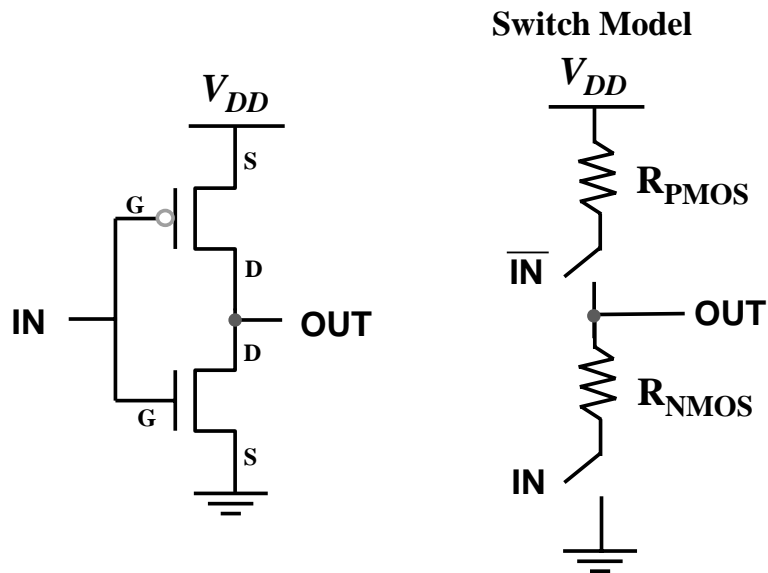
PMOS: The Complementary Switch



PMOS ON when Switch Input is Low



The CMOS Inverter

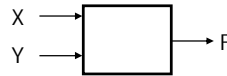




Possible Function of Two Inputs



There are 16 possible functions of 2 input variables:



X	Y	16 possible functions (F_0 - F_{15})															
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	1	1	1	1	0	0	0	1	1	0	1	1	1
1	0	0	0	1	1	0	0	1	1	0	1	0	0	1	0	1	1
1	1	0	1	0	0	0	1	0	1	0	1	0	1	0	1	1	

Labels for functions (from left to right):
 X AND Y, X, Y, X XOR Y, X OR Y, X NOR Y, NOT (X OR Y), X = Y, NOT Y, NOT X, X NAND Y, NOT (X AND Y)

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



Common Logic Gates



Gate	Symbol	Truth-Table	Expression															
NAND		<table border="1"> <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>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>	X	Y	Z	0	0	1	0	1	1	1	0	1	1	1	0	$Z = \overline{X \cdot Y}$
X	Y	Z																
0	0	1																
0	1	1																
1	0	1																
1	1	0																
AND		<table border="1"> <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>	X	Y	Z	0	0	0	0	1	0	1	0	0	1	1	1	$Z = X \cdot Y$
X	Y	Z																
0	0	0																
0	1	0																
1	0	0																
1	1	1																
NOR		<table border="1"> <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>0</td></tr> </tbody> </table>	X	Y	Z	0	0	1	0	1	0	1	0	0	1	1	0	$Z = \overline{X + Y}$
X	Y	Z																
0	0	1																
0	1	0																
1	0	0																
1	1	0																
OR		<table border="1"> <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>	X	Y	Z	0	0	0	0	1	1	1	0	1	1	1	1	$Z = X + Y$
X	Y	Z																
0	0	0																
0	1	1																
1	0	1																
1	1	1																



Exclusive (N)OR Gate



XOR
($X \oplus Y$)



X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

$Z = X\bar{Y} + \bar{X}Y$
X or Y but not both
("inequality", "difference")

XNOR
 $\overline{(X \oplus Y)}$



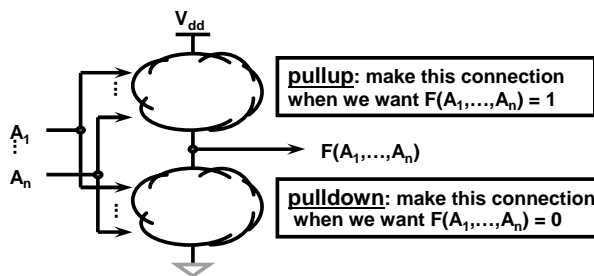
X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

$Z = \bar{X}Y + X\bar{Y}$
X or Y but not both
("inequality", "difference")

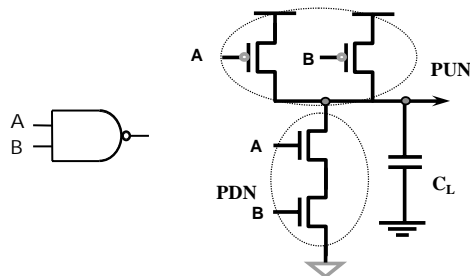
Widely used in arithmetic structures such as adders and multipliers



Generic CMOS Recipe



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



A	B	PDN	PUN	O
0	0	Off	On	1
0	1	Off	On	1
1	0	Off	On	1
1	1	On	Off	0

How do you build a 2-input NOR Gate?



Theorems of Boolean Algebra (I)



- **Elementary**

1. $X + 0 = X$

2. $X + 1 = 1$

3. $X + X = X$

4. $\overline{\overline{X}} = X$

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. $X + X \cdot Y = X$

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

12D. $X \cdot Y + \overline{X} \cdot Z = (X + Z) \cdot (\overline{X} + Y)$

- **Consensus:**

13. $(X \cdot Y) + (Y \cdot Z) + (\overline{X} \cdot Z) = X \cdot Y + \overline{X} \cdot Z$

13D. $(X + Y) \cdot (Y + Z) \cdot (\overline{X} + Z) = (X + Y) \cdot (\overline{X} + Z)$

- **De Morgan's:**

14. $\overline{(X + Y + \dots)} = \overline{X} \cdot \overline{Y} \cdot \dots$

14D. $\overline{(X \cdot Y \cdot \dots)} = \overline{X} + \overline{Y} + \dots$

- **generalized De Morgan's:**

15. $\overline{f(X_1, X_2, \dots, X_n, 0, 1, +, \cdot)} = f(\overline{X_1}, \overline{X_2}, \dots, \overline{X_n}, 1, 0, \cdot, +)$

- **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

□ $f(X_1, X_2, \dots, X_n, 0, 1, +, \cdot) \Leftrightarrow f(X_1, X_2, \dots, X_n, 1, 0, \cdot, +)$

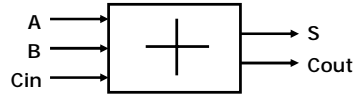


Simple Example: One Bit Adder



■ 1-bit binary adder

- inputs: A, B, Carry-in
- outputs: Sum, Carry-out



A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Sum-of-Products Canonical Form

$$S = \bar{A} \bar{B} \text{Cin} + \bar{A} B \bar{\text{Cin}} + A \bar{B} \bar{\text{Cin}} + A B \text{Cin}$$

$$\text{Cout} = \bar{A} B \text{Cin} + A \bar{B} \text{Cin} + A B \bar{\text{Cin}} + A B \text{Cin}$$

■ 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



$$\begin{aligned} \text{Cout} &= \bar{A} B \text{Cin} + A \bar{B} \text{Cin} + A B \bar{\text{Cin}} + A B \text{Cin} \\ &= \bar{A} B \text{Cin} + A B \text{Cin} + A \bar{B} \text{Cin} + A B \text{Cin} + A B \bar{\text{Cin}} + A B \text{Cin} \\ &= (\bar{A} + A) B \text{Cin} + A (\bar{B} + B) \text{Cin} + A B (\bar{\text{Cin}} + \text{Cin}) \\ &= B \text{Cin} + A \text{Cin} + A B \\ &= (B + A) \text{Cin} + A B \end{aligned}$$

$$\begin{aligned} S &= \bar{A} \bar{B} \text{Cin} + \bar{A} B \bar{\text{Cin}} + A \bar{B} \bar{\text{Cin}} + A B \text{Cin} \\ &= (\bar{A} \bar{B} + A B) \text{Cin} + (\bar{A} B + A \bar{B}) \bar{\text{Cin}} \\ &= (A \oplus B) \text{Cin} + (A \oplus B) \bar{\text{Cin}} \\ &= A \oplus B \oplus \text{Cin} \end{aligned}$$



Sum-of-Products & Product-of-Sum



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

A	B	C	minterms	
0	0	0	$\bar{A} \bar{B} \bar{C}$	m0
0	0	1	$\bar{A} \bar{B} C$	m1
0	1	0	$\bar{A} B \bar{C}$	m2
0	1	1	$\bar{A} B C$	m3
1	0	0	$A \bar{B} \bar{C}$	m4
1	0	1	$A \bar{B} C$	m5
1	1	0	$A B \bar{C}$	m6
1	1	1	$A B C$	m7

short-hand notation form in terms of 3 variables

F in canonical form:

$$\begin{aligned}
 F(A, B, C) &= \Sigma m(1,3,5,6,7) \\
 &= m1 + m3 + m5 + m6 + m7 \\
 F &= \bar{A} \bar{B} C + \bar{A} B C + A \bar{B} C + A B \bar{C} + ABC \\
 \text{canonical form} &\neq \text{minimal form} \\
 F(A, B, C) &= \bar{A} \bar{B} C + \bar{A} B C + A \bar{B} C + ABC + ABC \\
 &= (\bar{A} \bar{B} + \bar{A} B + A \bar{B} + AB)C + ABC \\
 &= ((\bar{A} + A)(\bar{B} + B))C + ABC \\
 &= C + ABC = ABC + C = AB + C
 \end{aligned}$$

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

A	B	C	maxterms	
0	0	0	$A + B + C$	M0
0	0	1	$A + B + \bar{C}$	M1
0	1	0	$A + \bar{B} + C$	M2
0	1	1	$A + \bar{B} + \bar{C}$	M3
1	0	0	$\bar{A} + B + C$	M4
1	0	1	$\bar{A} + B + \bar{C}$	M5
1	1	0	$\bar{A} + \bar{B} + C$	M6
1	1	1	$\bar{A} + \bar{B} + \bar{C}$	M7

short-hand notation for maxterms of 3 variables

F in canonical form:

$$\begin{aligned}
 F(A, B, C) &= \Pi M(0,2,4) \\
 &= M0 \cdot M2 \cdot M4 \\
 &= (A + B + C)(A + \bar{B} + C)(\bar{A} + B + C) \\
 \text{canonical form} &\neq \text{minimal form} \\
 F(A, B, C) &= (A + B + C)(A + \bar{B} + C)(\bar{A} + B + C) \\
 &= (A + B + C)(A + \bar{B} + C) \\
 &= (A + C)(B + C)
 \end{aligned}$$



Mapping Between Forms



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

E.g., $F(A,B,C) = \Sigma m(3,4,5,6,7) = \Pi M(0,1,2)$
- Maxterm to Minterm conversion:**
rewrite maxterm shorthand using minterm shorthand
replace maxterm indices with the indices not already used

E.g., $F(A,B,C) = \Pi M(0,1,2) = \Sigma m(3,4,5,6,7)$
- Minterm expansion of F to Minterm expansion of F':**
in minterm shorthand form, list the indices not already used in F

E.g., $F(A,B,C) = \Sigma m(3,4,5,6,7) = \Pi M(0,1,2)$ \longrightarrow $F'(A,B,C) = \Sigma m(0,1,2) = \Pi M(3,4,5,6,7)$
- Minterm expansion of F to Maxterm expansion of F':**
rewrite in Maxterm form, using the same indices as F

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



The Uniting Theorem



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

$$F = \bar{A}\bar{B} + A\bar{B} = (\bar{A} + A)\bar{B} = \bar{B}$$

A	B	F
0	0	1
0	1	0
1	0	1
1	1	0

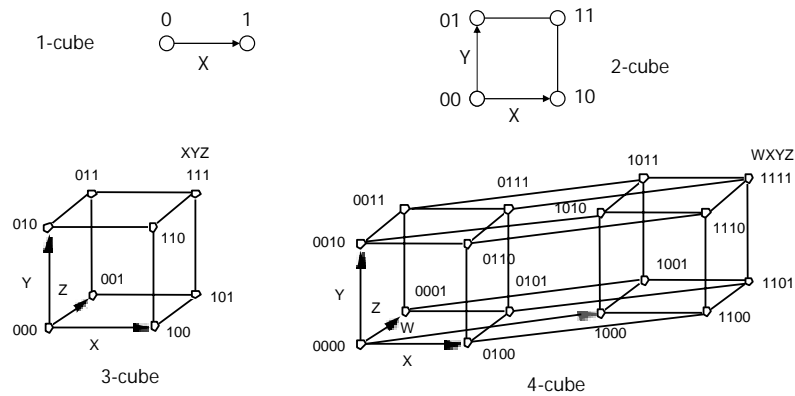
B has the same value in both on-set rows
 - B remains
 A has a different value in the two rows
 - A is eliminated



Boolean Cubes



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



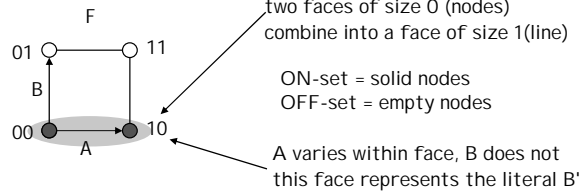


Mapping truth tables onto Boolean cubes



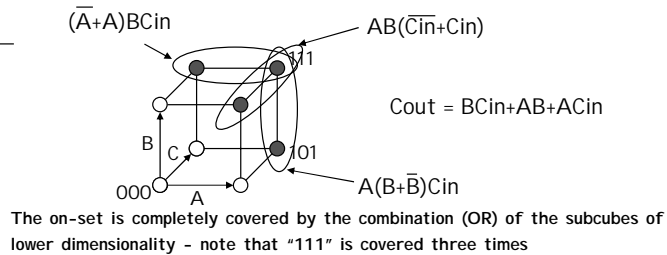
- **Uniting theorem** combines two "faces" of a cube into a larger "face"

A	B	F
0	0	1
0	1	0
1	0	1
1	1	0

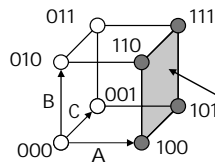


- **Three variable example: Binary full-adder carry-out logic**

A	B	Cin	Cout
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



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

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

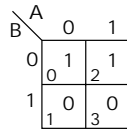


Karnaugh Maps



Alternative to truth-tables to help visualize adjacencies

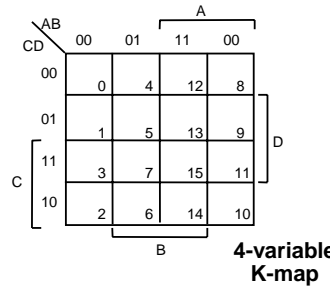
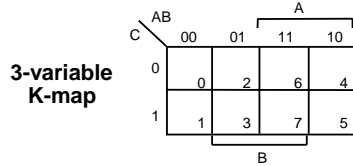
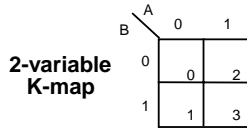
- Guide to applying the uniting theorem - On-set elements with only one variable changing value are adjacent unlike in a linear truth-table



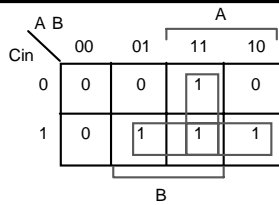
A	B	F
0	0	1
0	1	0
1	0	1
1	1	0

Numbering scheme based on Gray-code

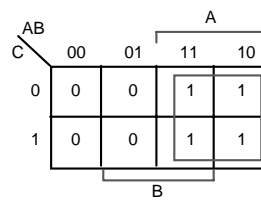
- e.g., 00, 01, 11, 10 (only a single bit changes in code for adjacent map cells)



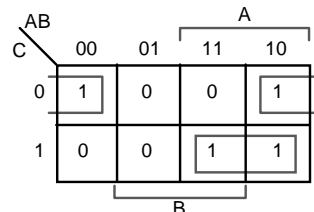
K-Map Examples



$C_{out} =$

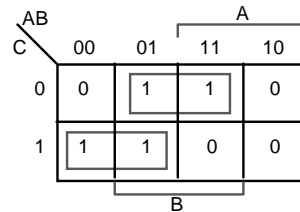


$F(A,B,C) =$



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

$F =$



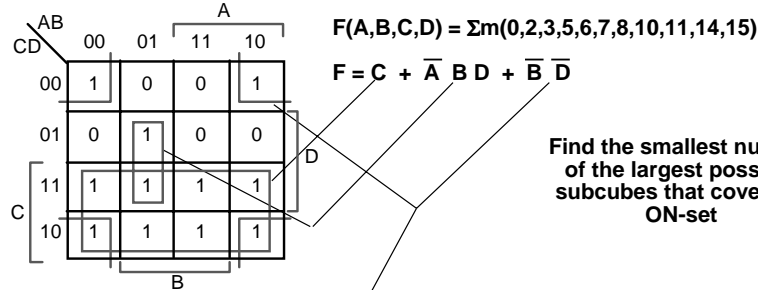
F' simply replace 1's with 0's and vice versa

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

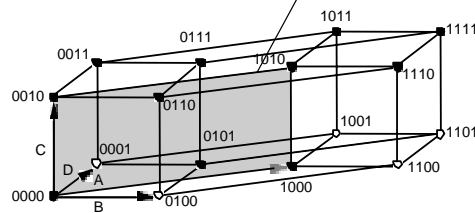
$F' =$



Four Variable Karnaugh Map



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



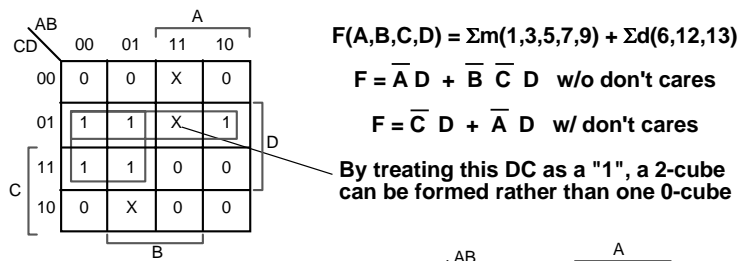
K-map Corner Adjacency Illustrated in the 4-Cube



K-Map Example: Don't Cares



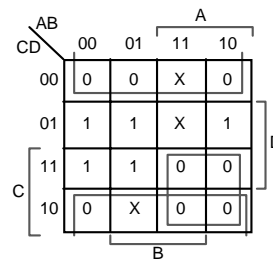
Don't Cares can be treated as 1's or 0's if it is advantageous to do so



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

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

Equivalent answer as above, but fewer literals



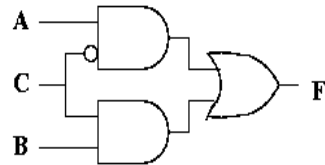


Hazards



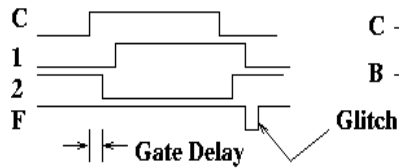
Static Hazards: Consider this function:

$$F = A * \bar{C} + B * C$$

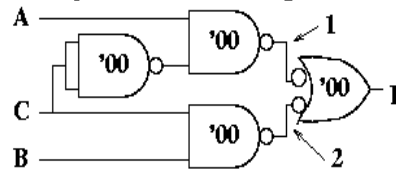


	AB			
C	00	01	11	10
0	0	0	1	1
1	0	1	1	0

A = B = 1



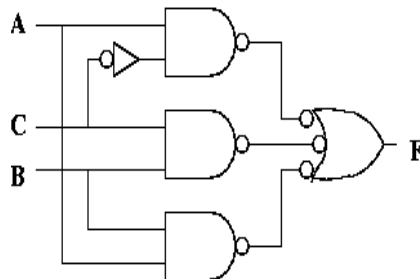
Implemented with MSI gates:



Fixing Hazards



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!



	AB			
C	00	01	11	10
0	0	0	1	1
1	0	1	1	0

$$F = A * \bar{C} + B * C + A * B$$