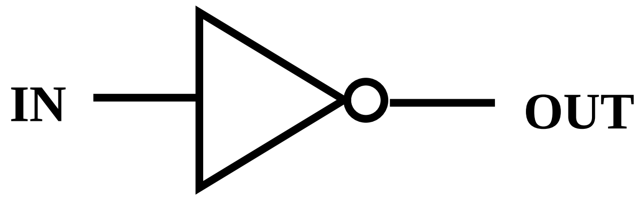
L2: Combinational Logic Design:

Construction and Boolean Algebra

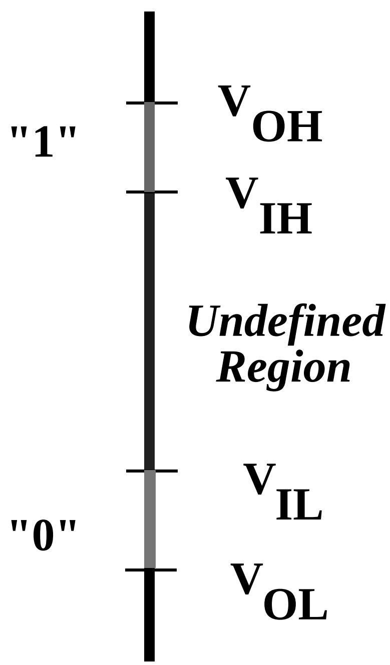
Corrected version – Sept. 7, 2003

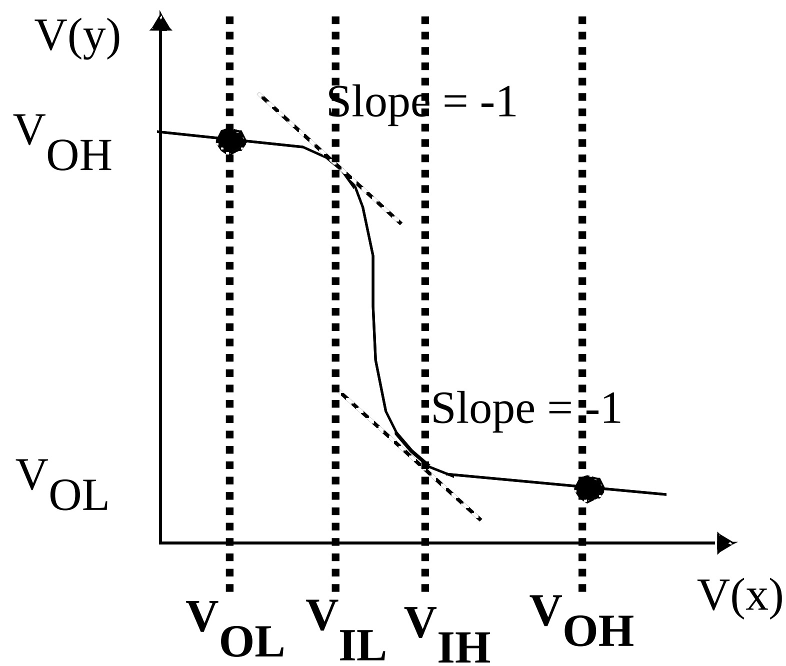
Some (most) lecture material derived from R. Katz, *"Contemporary Logic Design"*, Addison Wesley Publishing Company, Reading, MA, 1993. *Some slides are derived from slides used in past terms of 6.111*

The Inverter



IN	OUT
0	1
1	0

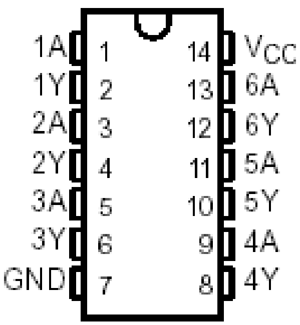


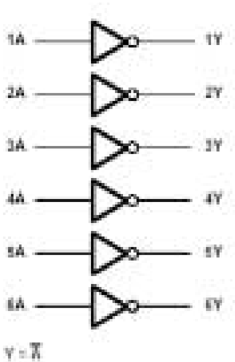


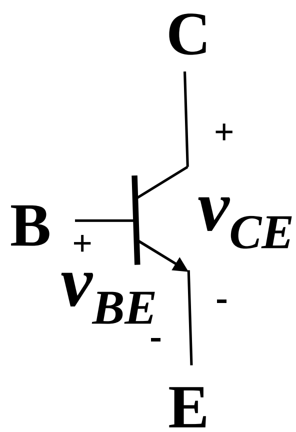
$$NM_L = V_{IL} - V_{OL}$$
$$NM_H = V_{OH} - V_{IH}$$

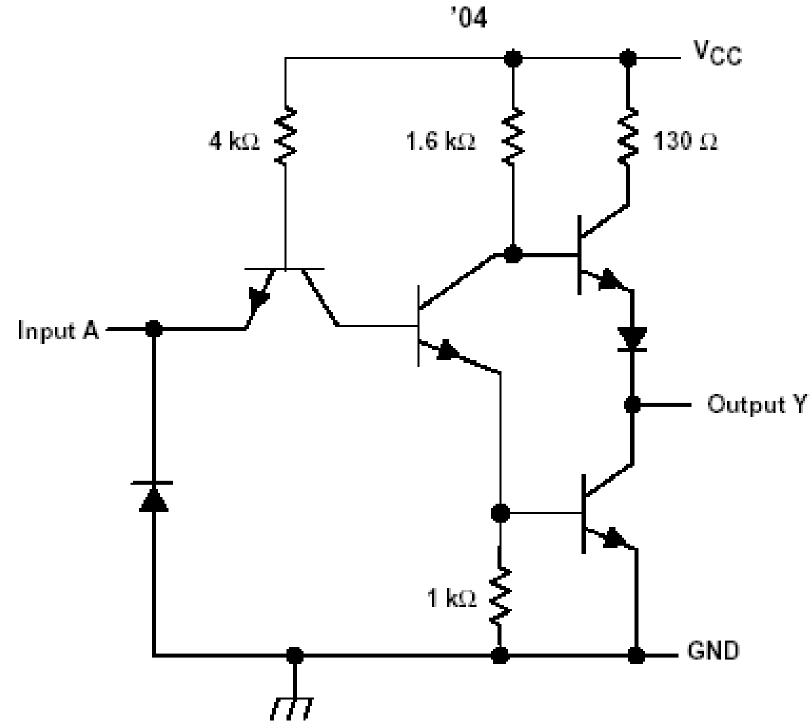
- **Large noise margins protect against various noise sources.**

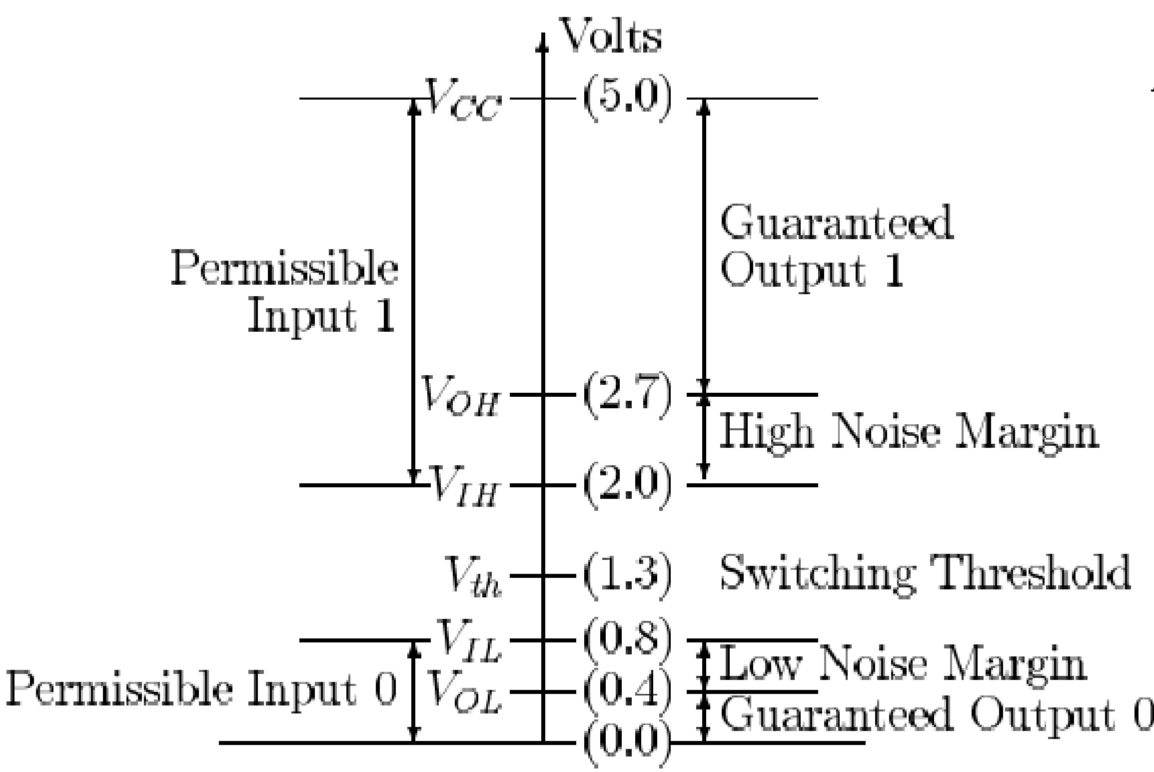
TTL Logic Style (1970's early 80's)





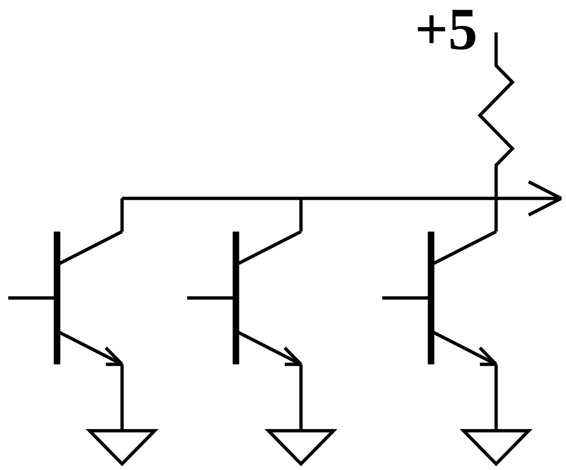






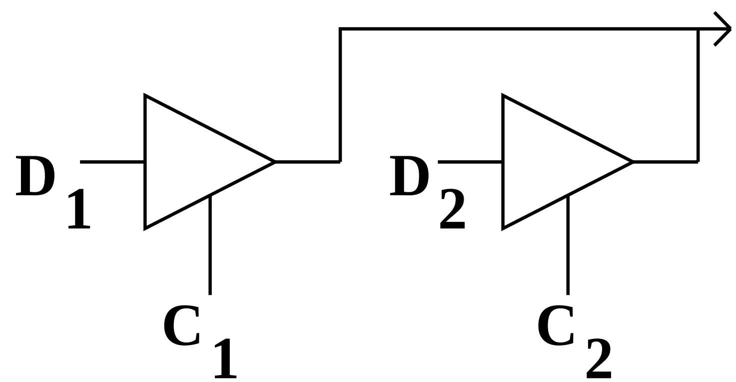
74LS04
(courtesy TI)

Busses

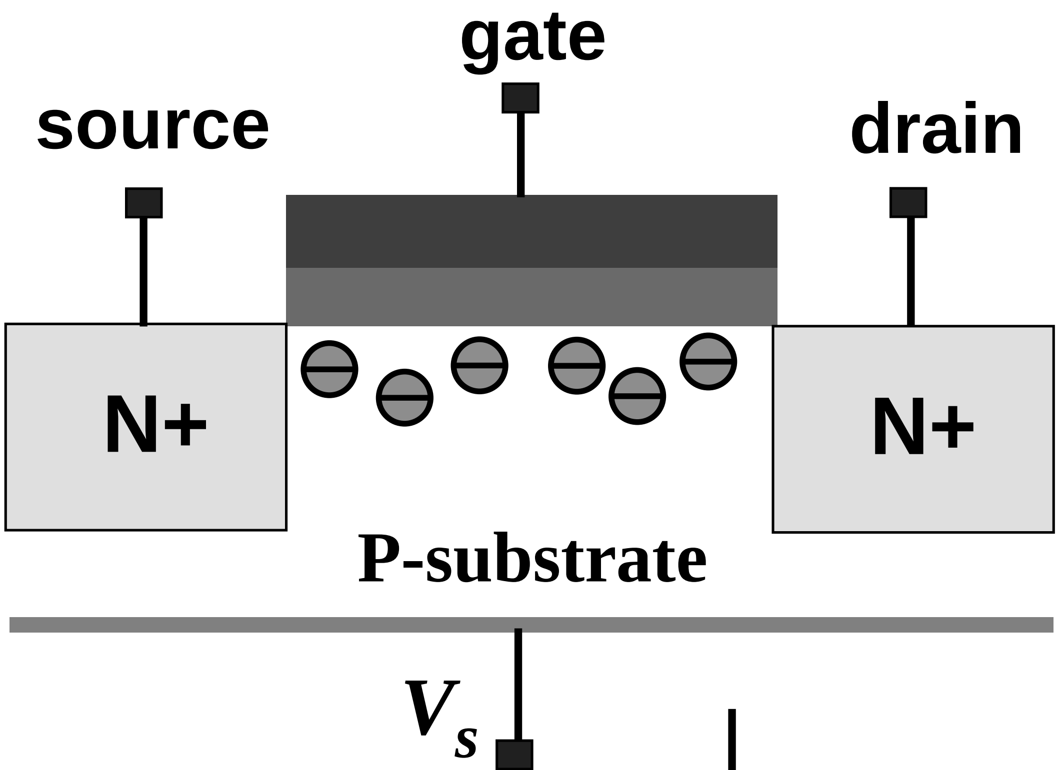


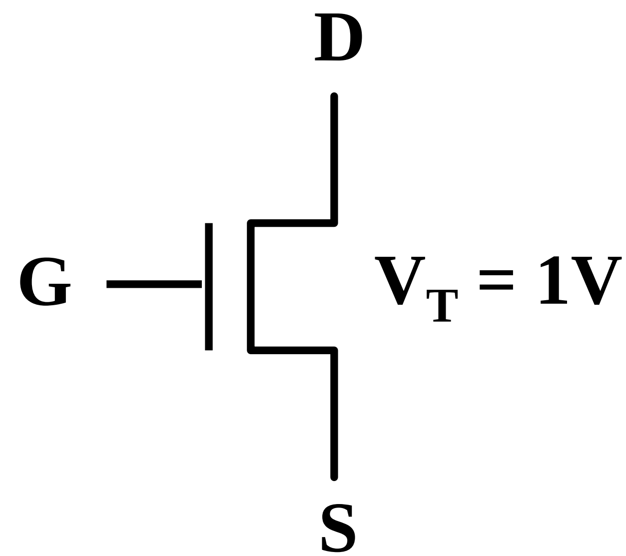
Open collector gates can be wired together like this to make wired ANDs.
This is a bus because it can be driven by more than one source.
You can't do this with Totem Pole outputs!

By controlling the gates on both transistors of a Totem Pole to be open, a high impedance is created (this is a tri−state). Control inputs C_1 and C_2 are output enables.

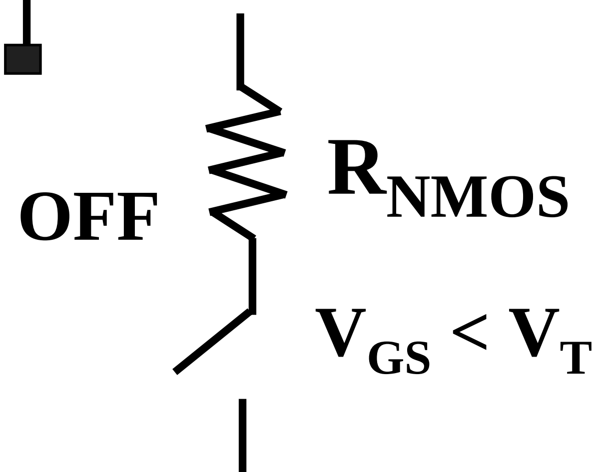


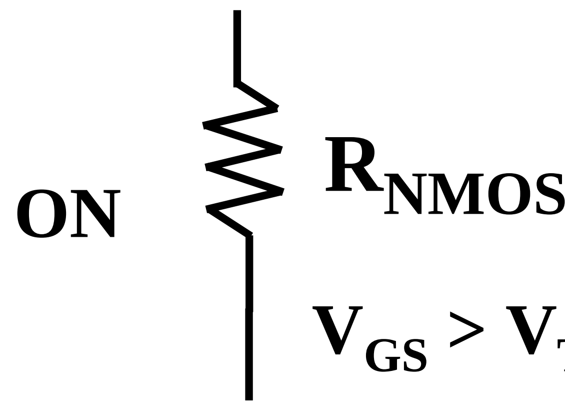
MOS Technology: The NMOS Switch





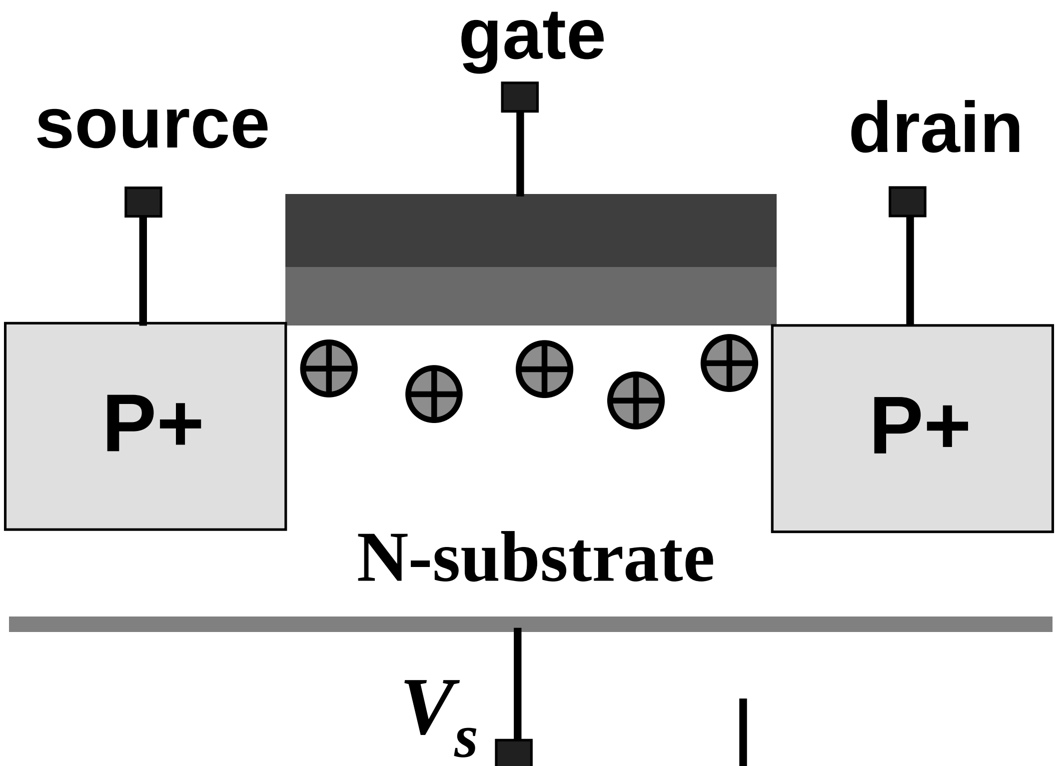
Switch Model

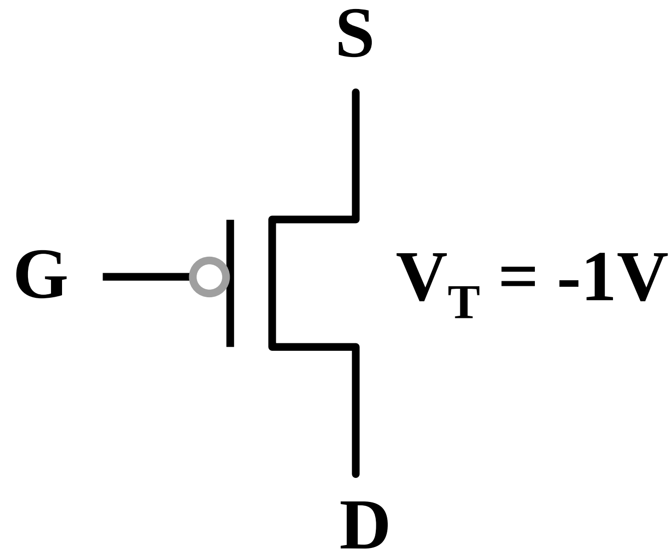




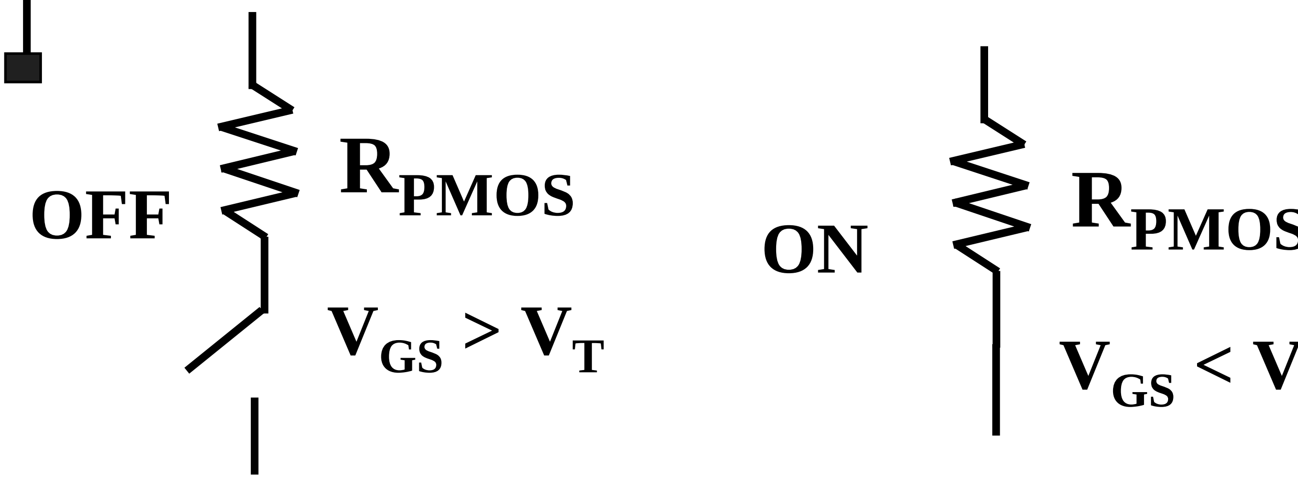
NMOS ON when Switch Input is High

PMOS: The Complementary Switch



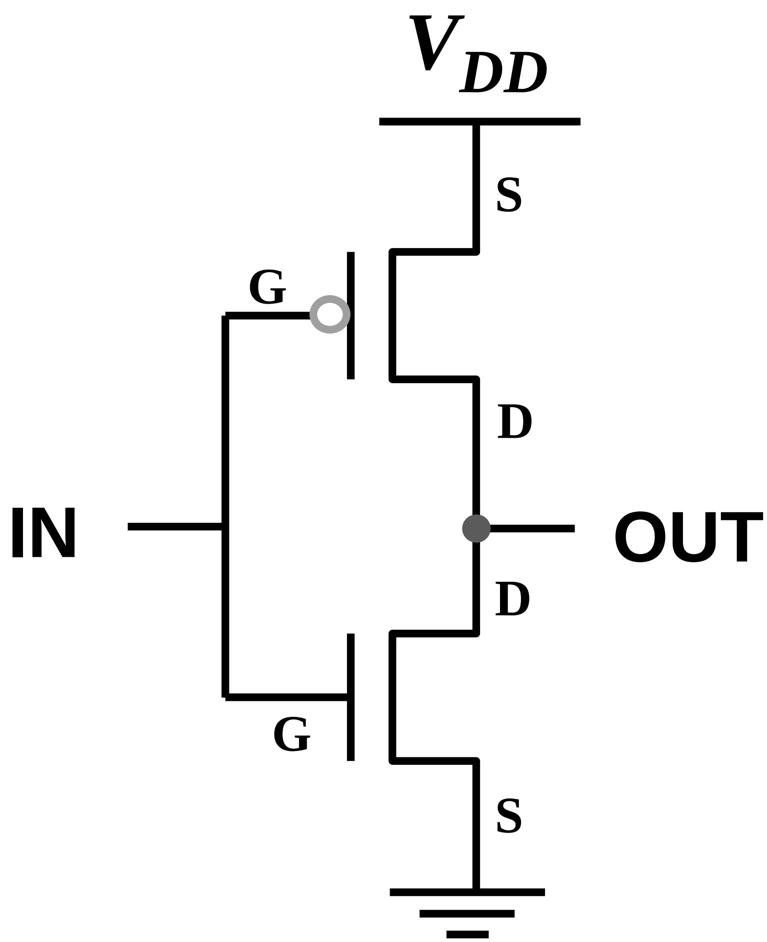


Switch Model

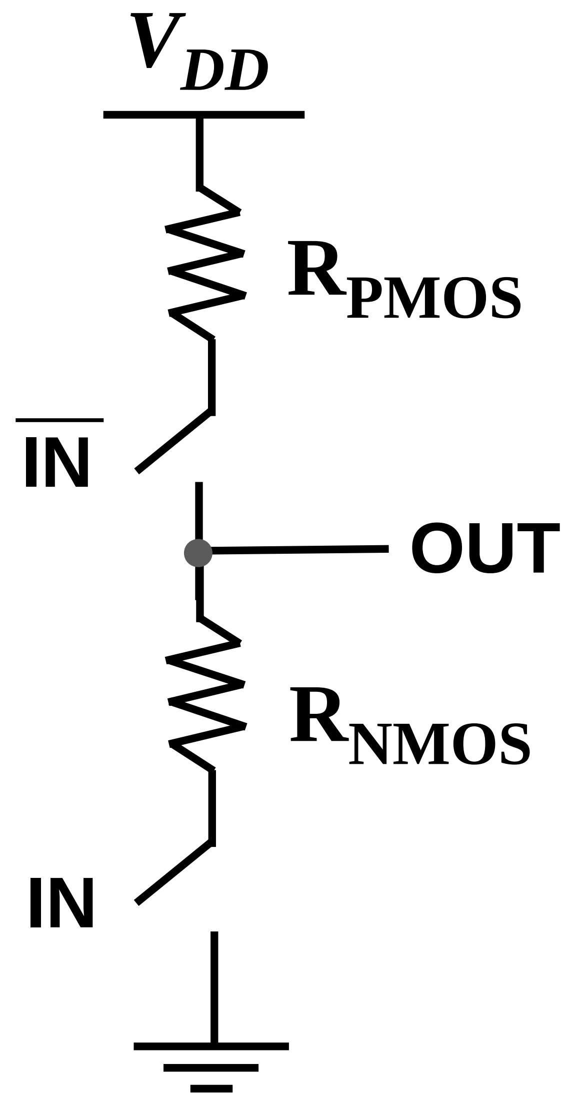


PMOS ON when Switch Input is Low

The CMOS Inverter

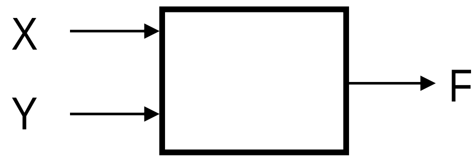


Switch Model



Possible Function of Two Inputs

There are 16 possible functions of 2 input variables:



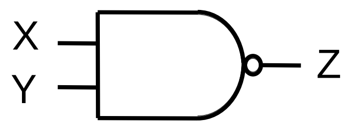
X	Y	16 possible functions (F_0–F_{15})															
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
		0	X AND Y		X		Y	X XOR Y	X OR Y	X NOR Y NOT (X OR Y)	X = Y	NOT Y		NOT X		X NAND Y NOT (X AND Y)	1

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

Common Logic Gates

Gate	Symbol	Truth Table	Expression

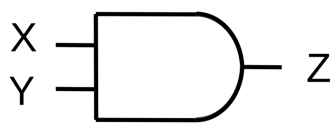
NAND



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

$$Z = \overline{X \bullet Y}$$

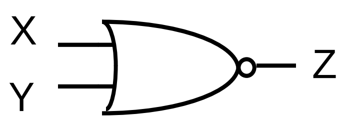
AND



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

$$Z = X \bullet Y$$

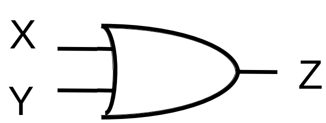
NOR



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

$$Z = \overline{X + Y}$$

OR

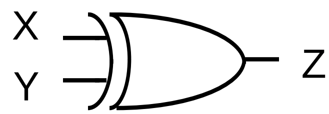


X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

$$Z = X + Y$$

Exclusive (N)OR Gate

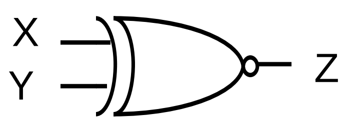
XOR
$(X \oplus Y)$



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

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

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

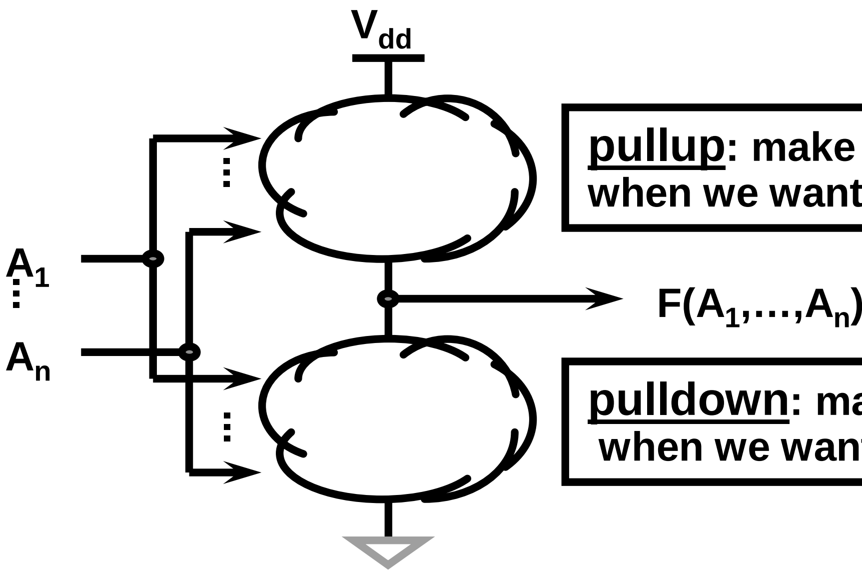


X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

$Z = \overline{X}\,\overline{Y} + X\,Y$
X and Y the same
("equality")

Widely used in arithmetic structures such as adders and multipliers

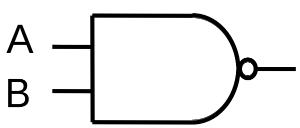
Generic CMOS Recipe

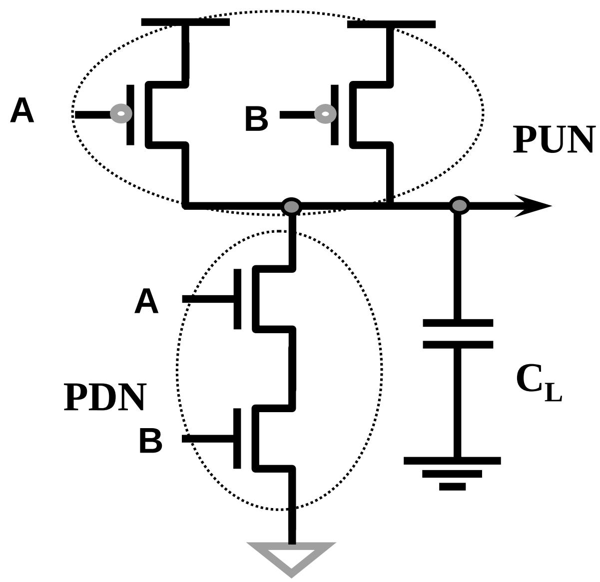


pullup: make this connection when we want $F(A_1,\ldots,A_n) = 1$

pulldown: make this connection when we want $F(A_1,\ldots,A_n) = 0$

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





A	B	PDN	PUN	O
0	0	0ff	0n	1
0	1	0ff	0n	1
1	0	0ff	0n	1
1	1	0n	0ff	0

How do you build a 2 input NOR Gate?

Theorems of Boolean Algebra (I)

- **Elementary**
  1. $X + 0 = X$ — 1D. $X \bullet 1 = X$
  2. $X + 1 = 1$ — 2D. $X \bullet 0 = 0$
  3. $X + X = X$ — 3D. $X \bullet X = X$
  4. $(\overline{\overline{X}}) = X$
  5. $X + \overline{X} = 1$ — 5D. $X \bullet \overline{X} = 0$
- **Commutativity:**
  6. $X + Y = Y + X$ — 6D. $X \bullet Y = Y \bullet X$
- **Associativity:**
  7. $(X + Y) + Z = X + (Y + Z)$ — 7D. $(X \bullet Y) \bullet Z = X \bullet (Y \bullet Z)$
- **Distributivity:**
  8. $X \bullet (Y + Z) = (X \bullet Y) + (X \bullet Z)$ — 8D. $X + (Y \bullet Z) = (X + Y) \bullet (X + Z)$
- **Uniting:**
  9. $X \bullet Y + X \bullet \overline{Y} = X$ — 9D. $(X + Y) \bullet (X + \overline{Y}) = X$
- **Absorption:**
  10. $X + X \bullet Y = X$ — 10D. $X \bullet (X + Y) = X$
  11. $(X + \overline{Y}) \bullet Y = X \bullet Y$ — 11D. $(X \bullet \overline{Y}) + Y = X + Y$

Theorems of Boolean Algebra (II)

- **Factoring:**

  12. $(X \bullet Y) + (X \bullet Z) = X \bullet (Y + Z)$

  12D. $(X + Y) \bullet (X + Z) = X + (Y \bullet Z)$

- **Consensus:**

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

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

- **De Morgan's:**

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

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

- **Generalized De Morgan's:**

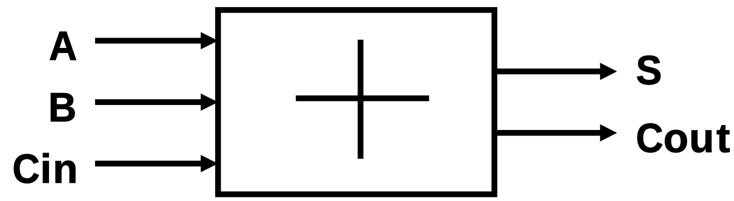
  15. $\overline{f}(X1,X2,...,Xn,0,1,+,\bullet) = f(\overline{X1},\overline{X2},...,\overline{Xn},1,0,\bullet,+)$

- **Duality**

  - Dual of a Boolean expression is derived by replacing • by +, + by •, 0 by 1, and 1 by 0, and leaving variables unchanged
  - $f(X1,X2,...,Xn,0,1,+,\bullet) \Leftrightarrow f(X1,X2,...,Xn,1,0,\bullet,+)$

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

$$S = \bar{A}\,\bar{B}\,Cin + \bar{A}\,B\,\overline{Cin} + A\,\bar{B}\,\overline{Cin} + A\,B\,Cin$$

$$Cout = \bar{A}\,B\,Cin + A\,\bar{B}\,Cin + A\,B\,\overline{Cin} + A\,B\,Cin$$

- 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

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

$$= \bar{A}\,B\,\text{Cin} + A\,B\,\text{Cin} + A\,\bar{B}\,\text{Cin} + A\,B\,\text{Cin} + A\,B\,\overline{\text{Cin}} + A\,B\,\text{Cin}$$

$$= (\bar{A} + A)\,B\,\text{Cin} + A\,(\bar{B} + B)\,\text{Cin} + A\,B\,(\overline{\text{Cin}} + \text{Cin})$$

$$= B\,\text{Cin} + A\,\text{Cin} + A\,B$$

$$= (B + A)\,\text{Cin} + A\,B$$

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

$$= (\bar{A}\,\bar{B} + A\,B)\,\text{Cin} + (A\,\bar{B} + \bar{A}\,B)\,\overline{\text{Cin}}$$

$$= \overline{(A \oplus B)}\,\text{Cin} + (A \oplus B)\,\overline{\text{Cin}}$$

$$= A \oplus B \oplus \text{Cin}$$

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:

$F(A, B, C) = \Sigma m(1,3,5,6,7)$

$= m1 + m3 + m5 + m6 + m7$

$F = \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$

canonical form ≠ minimal form

$F(A, B, C) = \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + ABC + AB\bar{C}$

$= (\bar{A}\bar{B} + \bar{A}B + A\bar{B} + AB)C + AB\bar{C}$

$= ((\bar{A} + A)(\bar{B} + B))C + AB\bar{C}$

$= C + AB\bar{C} = AB\bar{C} + C = AB + C$

- **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:

$F(A, B, C) = \Pi M(0,2,4)$

$= M0 \bullet M2 \bullet M4$

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

canonical form ≠ minimal form

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

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

$(A + B + C)\ (\bar{A} + B + C)$

$= (A + C)\ (B + C)$

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

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

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

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**

$$F = \overline{A}\,\overline{B} + A\overline{B} = (\overline{A} + A)\overline{B} = \overline{B}$$

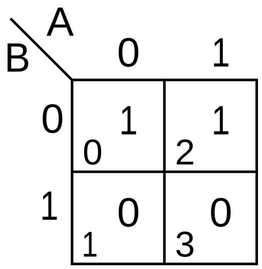
A	B	F
0	0	1
0	1	0
1	0	1
1	1	0

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

A has a different value in the two rows – A is eliminated

Karnaugh Maps

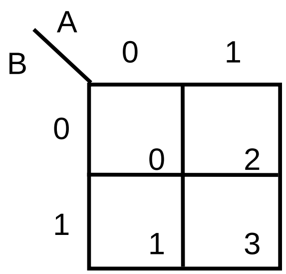
- Alternative to truth-tables to help visualize adjacencies
  - Guide to applying the uniting theorem - On-set elements with only one variable changing value are adjacent unlike in a linear truth-table

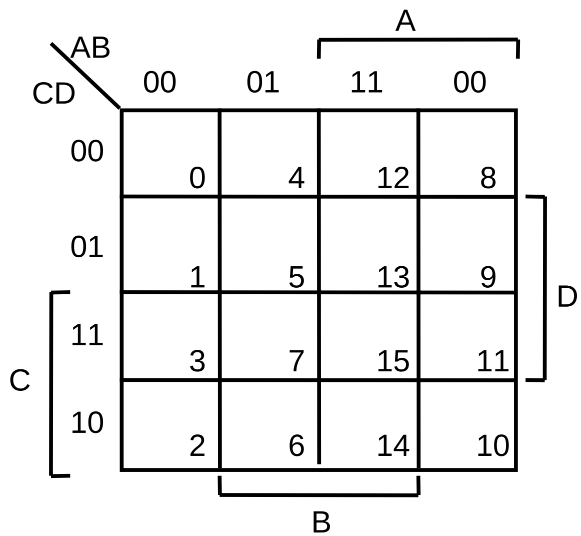


A	B	F
0	0	1
0	1	0
1	0	1
1	1	0

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

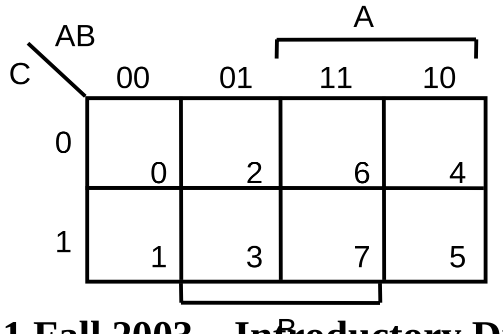
2-variable K-map



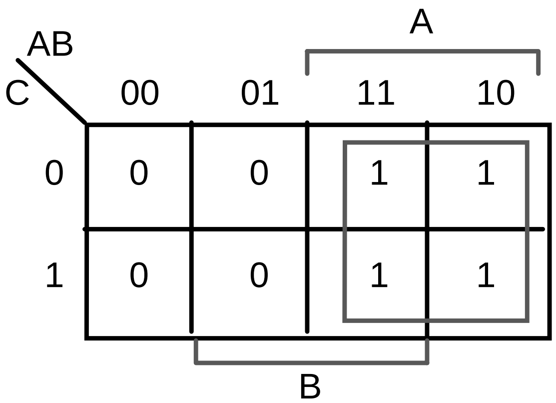


4-variable K-map

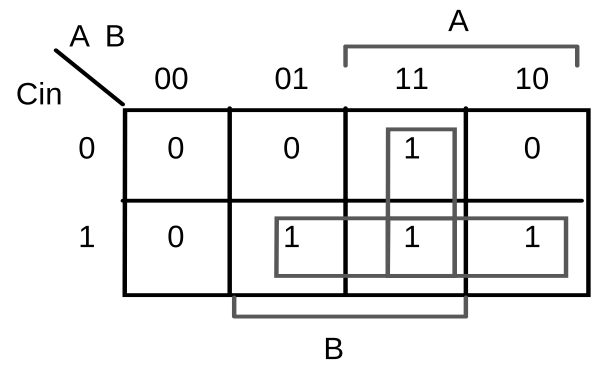
3-variable K-map



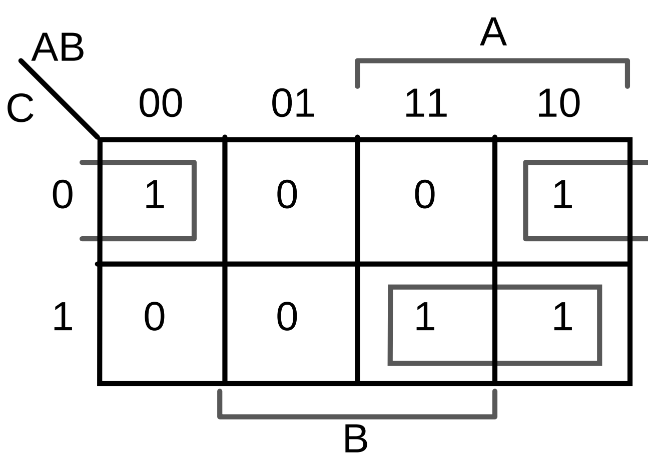
K-Map Examples



F(A,,C) =

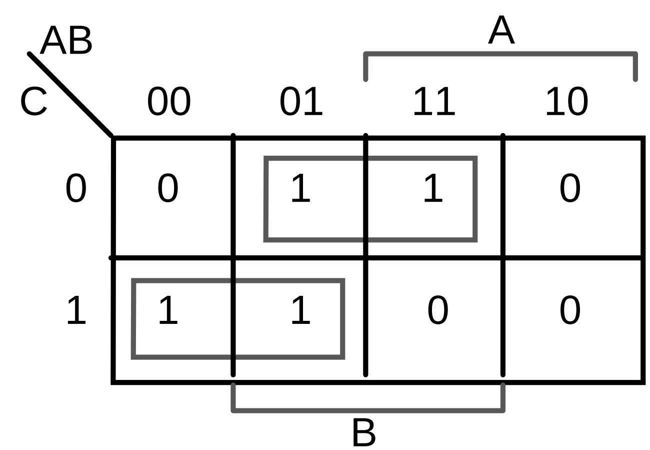


Cout =



F(A,B,C) = Σm(0,4,5,7)

F =

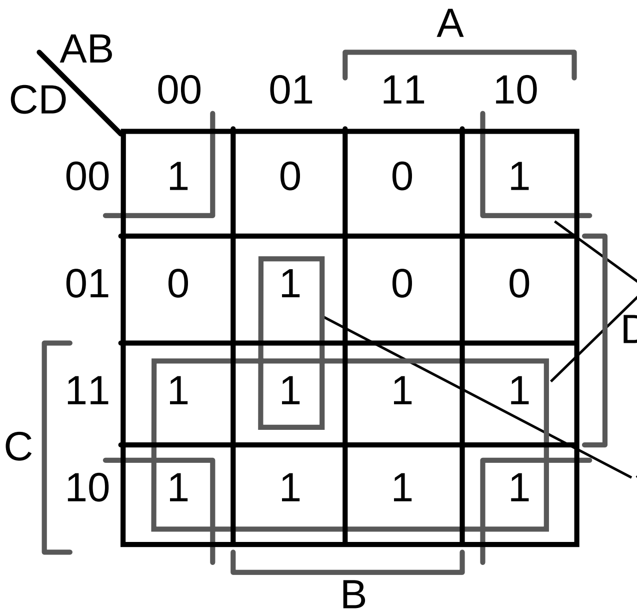


F' simply replace 1s with 0s and vice versa

F'(A,B,C) = Σm(1,2,3,6)

F' =

Four Variable Karnaugh Map



CD \ AB	00	01	11	10
00	1	0	0	1
01	0	1	0	0
11	1	1	1	1
10	1	1	1	1

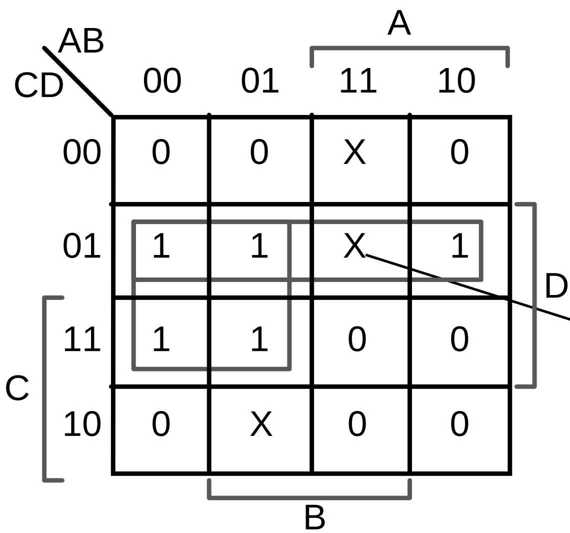
$$F(A,B,C,D) = \Sigma m(0,2,3,5,6,7,8,10,11,14,15)$$

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

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

K-Map Example: Don't Cares

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



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

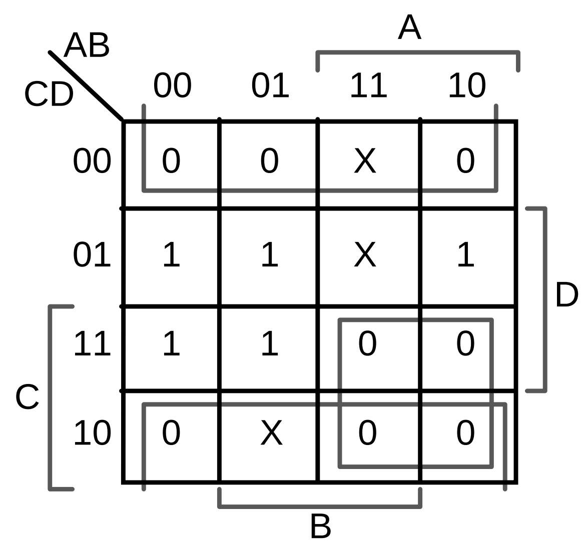
$F = \overline{A}D + \overline{B}\,\overline{C}D$ w/o don't cares

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

By treating this don't care as a "1", a group of 4 can be used instead of a group of 2.

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

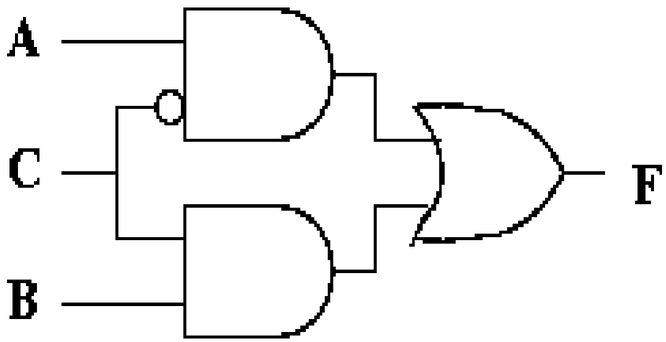
Equivalent answer as above, but fewer literals



Hazards

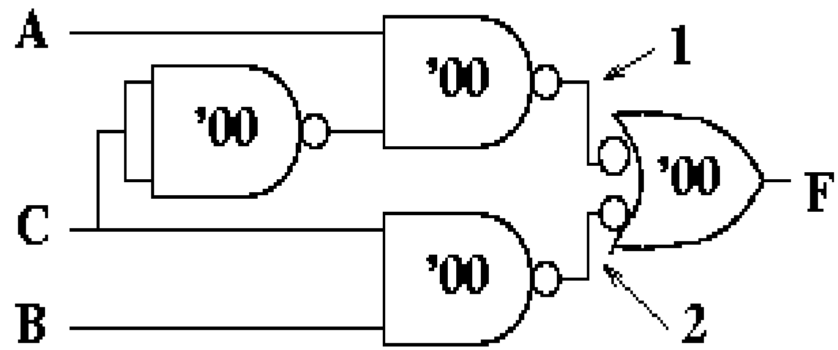
Static Hazards: Consider this function:

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

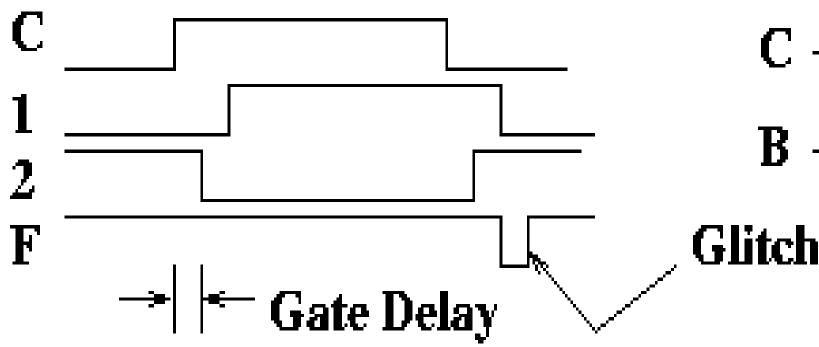


C \ AB	00	01	11	10
0	0	0	1	1
1	0	1	1	0

Implemented with MSI gates:

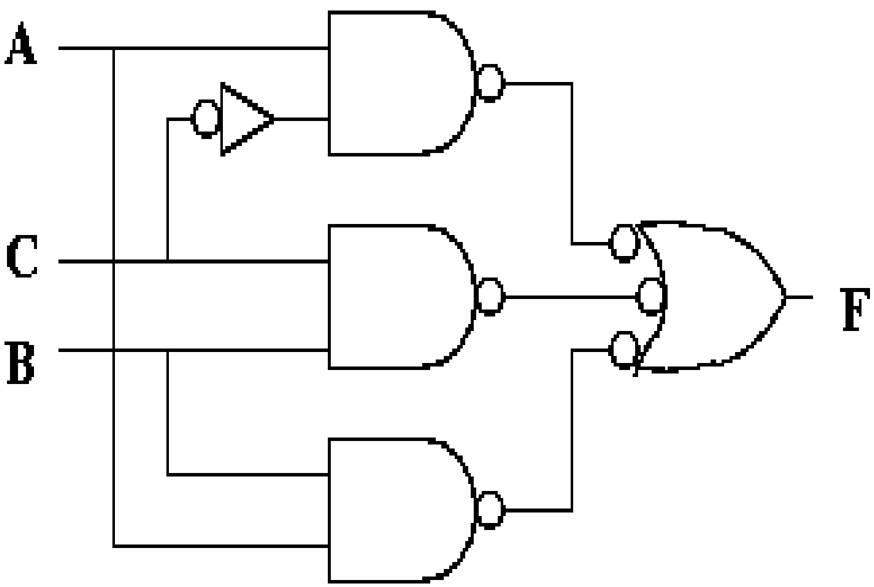


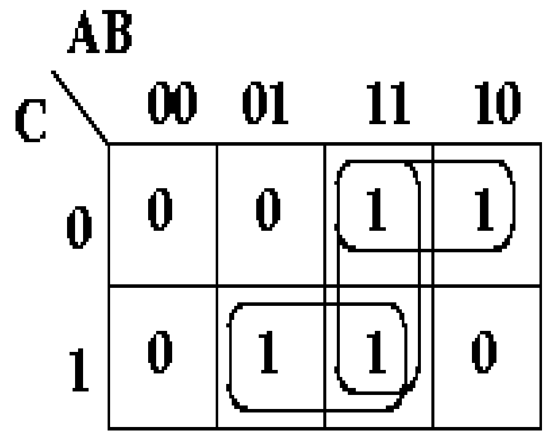
A = B = 1



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!





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