

Some Definitions

- **Combinational logic:** a digital logic circuit in which logical decisions are made based only on combinations of the inputs (e.g., an adder).
- **Sequential logic:** a circuit in which decisions are made based on combinations of the current inputs as well as the past history of inputs (e.g., a memory unit).
- **Finite state machine:** a circuit which has an internal state, and whose outputs are functions of both current inputs and its internal state (e.g., a vending machine controller).

The Combinational Logic Unit

- Translates a set of inputs into a set of outputs according to one or more mapping functions.
- Inputs and outputs for a CLU normally have two distinct (binary) values: high and low, 1 and 0, 0 and 1, or 5 v and 0 v, for example.
- The outputs of a CLU are strictly functions of the inputs, and the outputs are updated immediately after the inputs change. A set of inputs i_0 – i_n are presented to the CLU, which produces a set of outputs according to mapping functions f_0 – f_m .

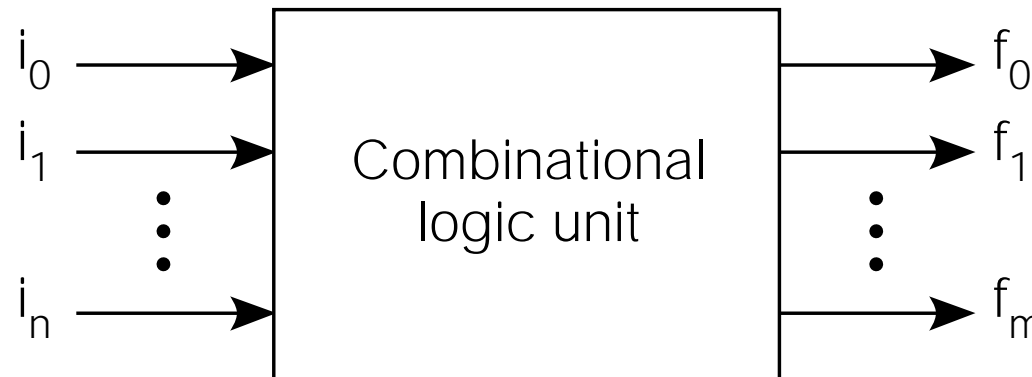


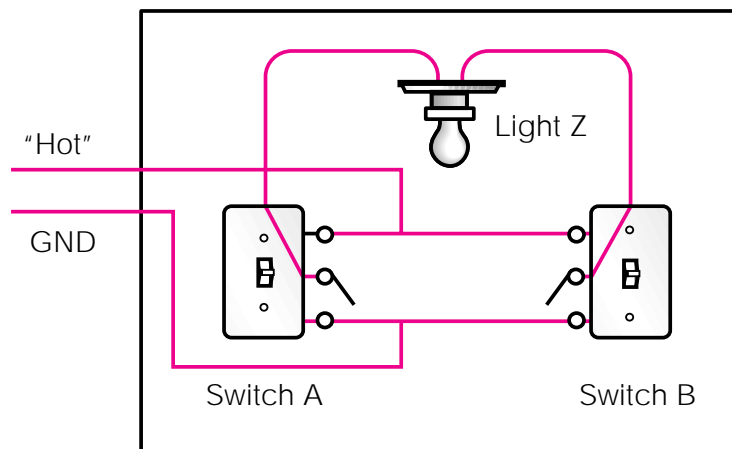
Fig A.1

Truth Tables

- Developed in 1854 by George Boole
- Further developed by Claude Shannon (Bell Labs)
- Outputs are computed for all possible input combinations (how many input combinations are there?)

Consider a room with two light switches. How must they work†?

Fig. A.2



Inputs		Output
A	B	Z
0	0	0
0	1	1
1	0	1
1	1	0

†Don't show this to your electrician, or wire your house this way. This circuit definitely violates the electric code. The practical circuit never leaves the lines to the light "hot" when the light is turned off. Can you figure how?

Fig A.4 Truth Tables Showing All Possible Functions of Two Binary Variables

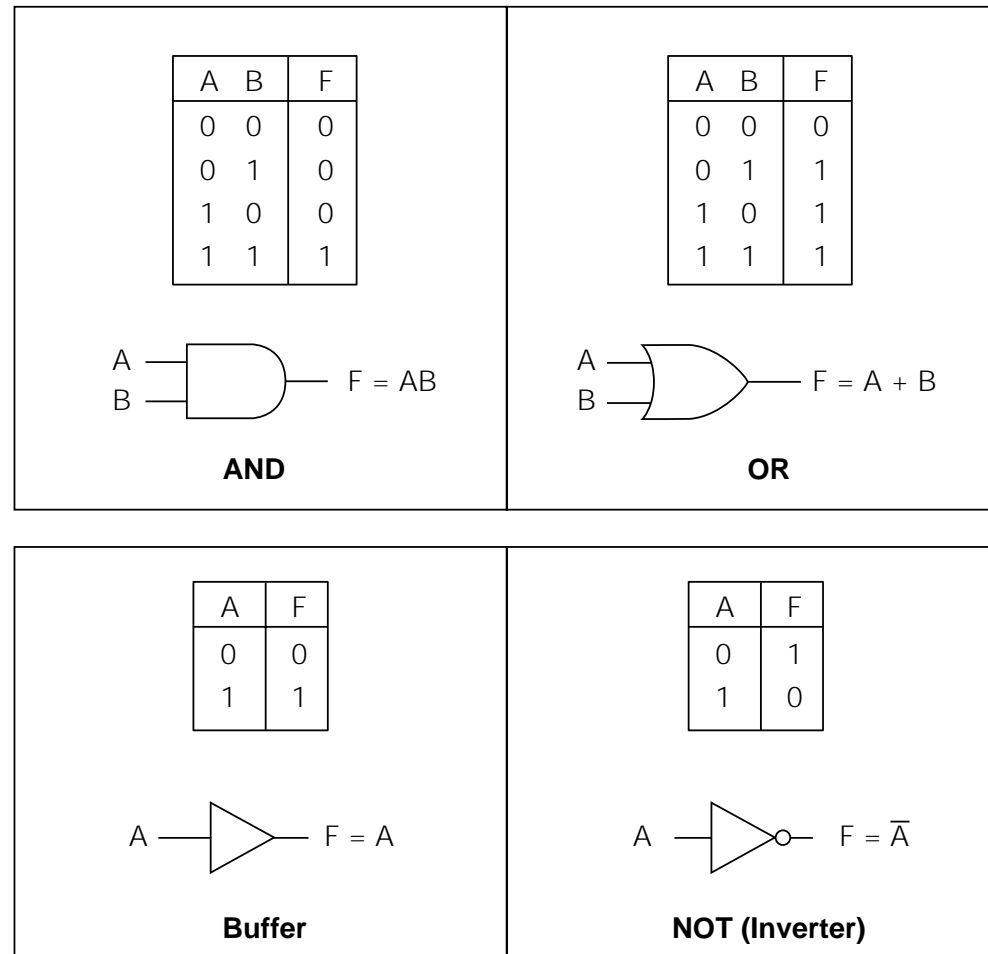
<i>A</i>	<i>B</i>	<i>False</i>	<i>AND</i>	$\overline{A\overline{B}}$	<i>A</i>	$\overline{A}B$	<i>B</i>	<i>XOR</i>	<i>OR</i>
0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

<i>A</i>	<i>B</i>	<i>NOR</i>	<i>XNOR</i>	\overline{B}	$A + \overline{B}$	\overline{A}	$\overline{A} + B$	<i>NAND</i>	<i>True</i>
0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1

- The more frequently used functions have names: AND, XOR, OR, NOR, XNOR, and NAND. (Always use upper-case spelling.)

Logic Gates and Their Symbols

Fig. A.5 Logic Gate Symbols for *AND*, *OR*, Buffer, and *NOT* Boolean functions



- Note the use of the “inversion bubble.”
- Be careful about the “nose” of the gate when drawing **AND** vs. **OR**.

Fig A.6 Logic Gate Symbols for *NAND*, *NOR*, *XOR*, and *XNOR* Boolean functions

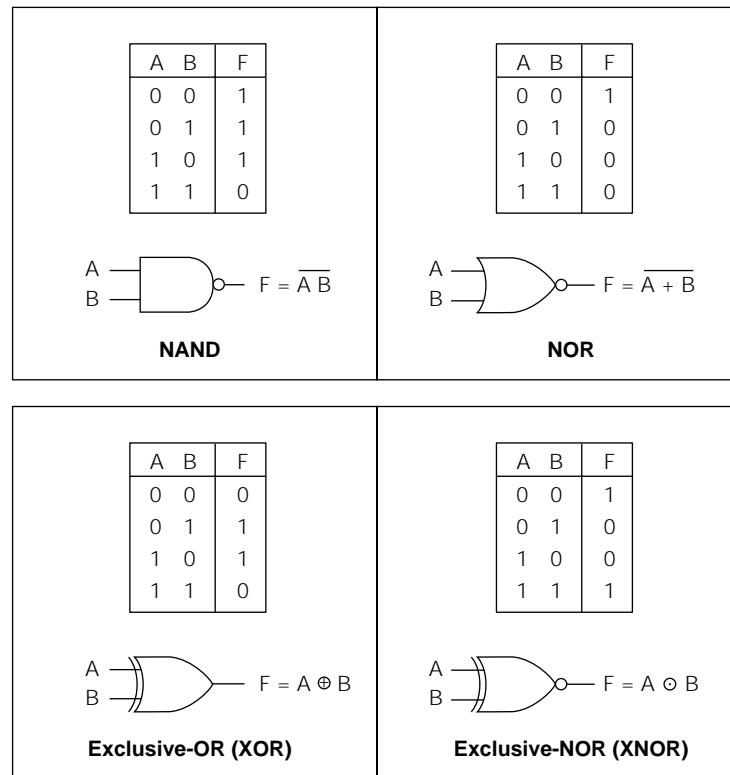
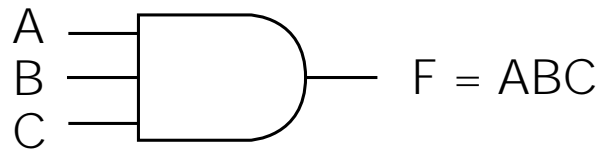
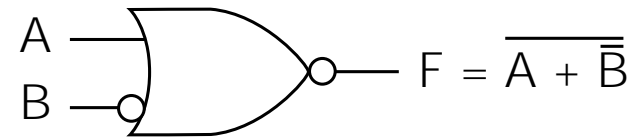


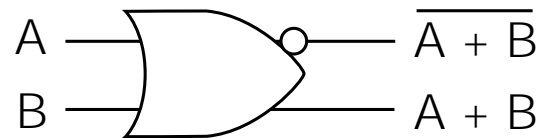
Fig A.7 Variations of Basic Logic Gate Symbols



(a)



(b)



(c)

(a) 3 inputs

(b) A negated input

(c) Complementary outputs

Tbl A.1 The Basic Properties of Boolean Algebra

Principle of duality: The dual of a Boolean function is gotten by replacing AND with OR and OR with AND, constant 1s by 0s, and 0s by 1s

Relationship	Dual	Property
$A B = B A$	$A + B = B + A$	Commutative
$A (B + C) = A B + A C$	$A + B C = (A + B) (A + C)$	Distributive
$1 A = A$	$0 + A = A$	Identity
$A \bar{A} = 0$	$A + \bar{A} = 1$	Inverse
$0 A = 0$	$1 + A = 1$	Null
$A A = A$	$A + A = A$	Idempotence
$A (B C) = (A B) C$	$A + (B + C) = (A + B) + C$	Associative
$\overline{\bar{A}} = A$		Complement
$\overline{A B} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A} \bar{B}$	DeMorgan's Theorem
$AB + \bar{A}C + BC = AB + \bar{A}C$	$(A + B)(\bar{A} + C)(B + C) = (A + B)(\bar{A} + C)$	Consensus Theorem

Postulates



Theorems

A, B, etc. are Literals; 0 and 1 are constants.

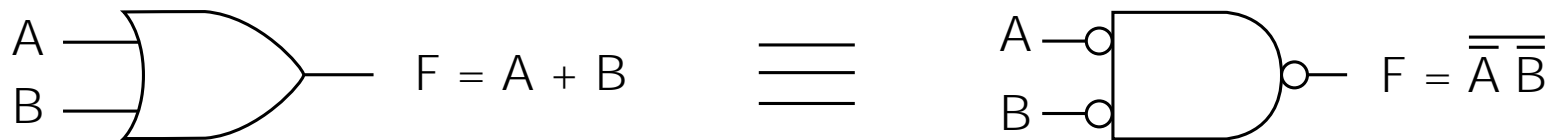
DeMorgan's Theorem

Fig A.11

A	B	$\overline{A B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} \overline{B}$
0	0	1	1
0	1	1	0
1	0	1	0
1	1	0	0

DeMorgan's theorem: $A + B = \overline{\overline{A + B}} = \overline{\overline{A} \overline{B}}$

Fig A.12

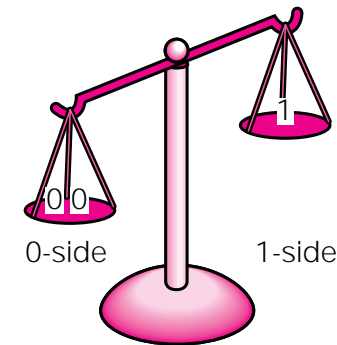


Discuss: Applying DeMorgan's theorem by "pushing the bubbles" and "bubble tricks."

The Sum-of-Products (SOP) Form

Fig. A.14 Truth Table for the Majority Function

Minterm Index	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



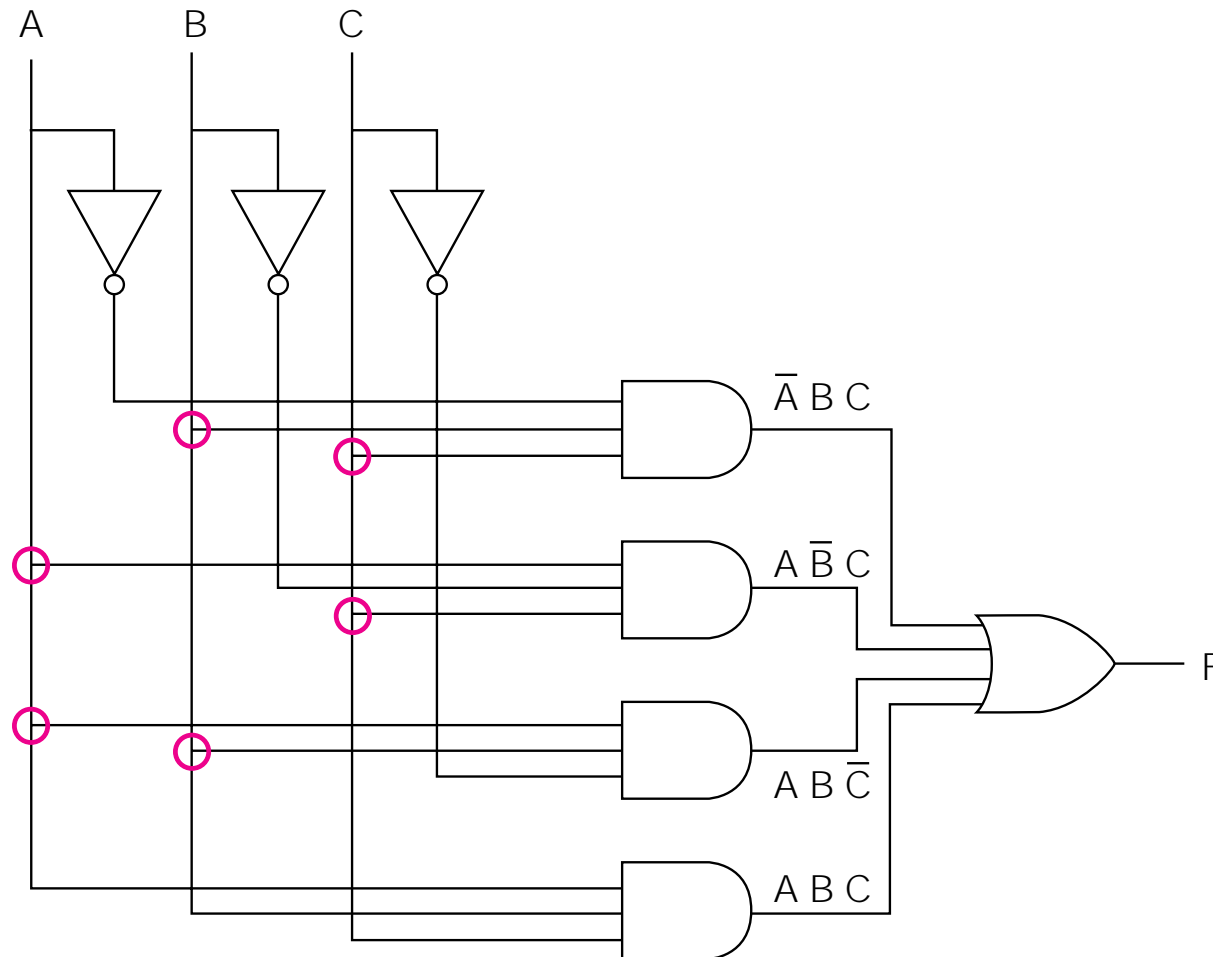
A balance tips to the left or right depending on whether there are more 0's or 1's.

- Transform the function into a two-level AND-OR equation
- Implement the function with an arrangement of logic gates from the set {AND, OR, NOT}
- M is true when $A = 0$, $B = 1$, and $C = 1$, or when $A = 1$, $B = 0$, and $C = 1$, and so on for the remaining cases.
- Represent logic equations by using the sum-of-products (SOP) form

The SOP Form of the Majority Gate

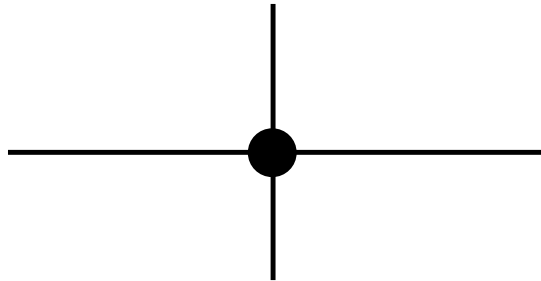
- The SOP form for the 3-input majority gate is:
- $M = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC = m_3 + m_5 + m_6 + m_7 = \Sigma(3, 5, 6, 7)$
- Each of the 2^n terms are called minterms, running from 0 to $2^n - 1$
- Note the relationship between minterm number and Boolean value.
- Discuss: common-sense interpretation of equation.

Fig A.15 A Two-Level AND-OR Circuit Implements the Majority Function

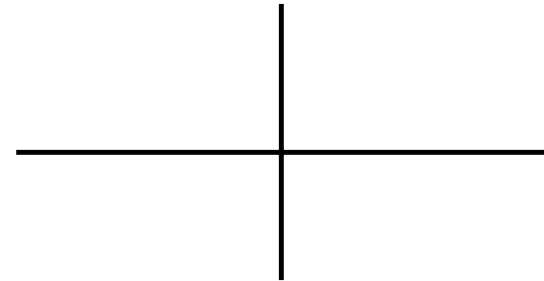


Discuss: what is the gate count?

Fig A.16 Four Notations Used at Circuit Intersections



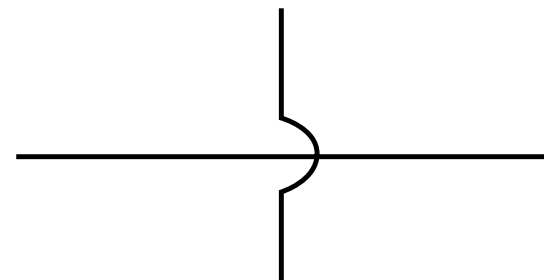
Connection



No connection

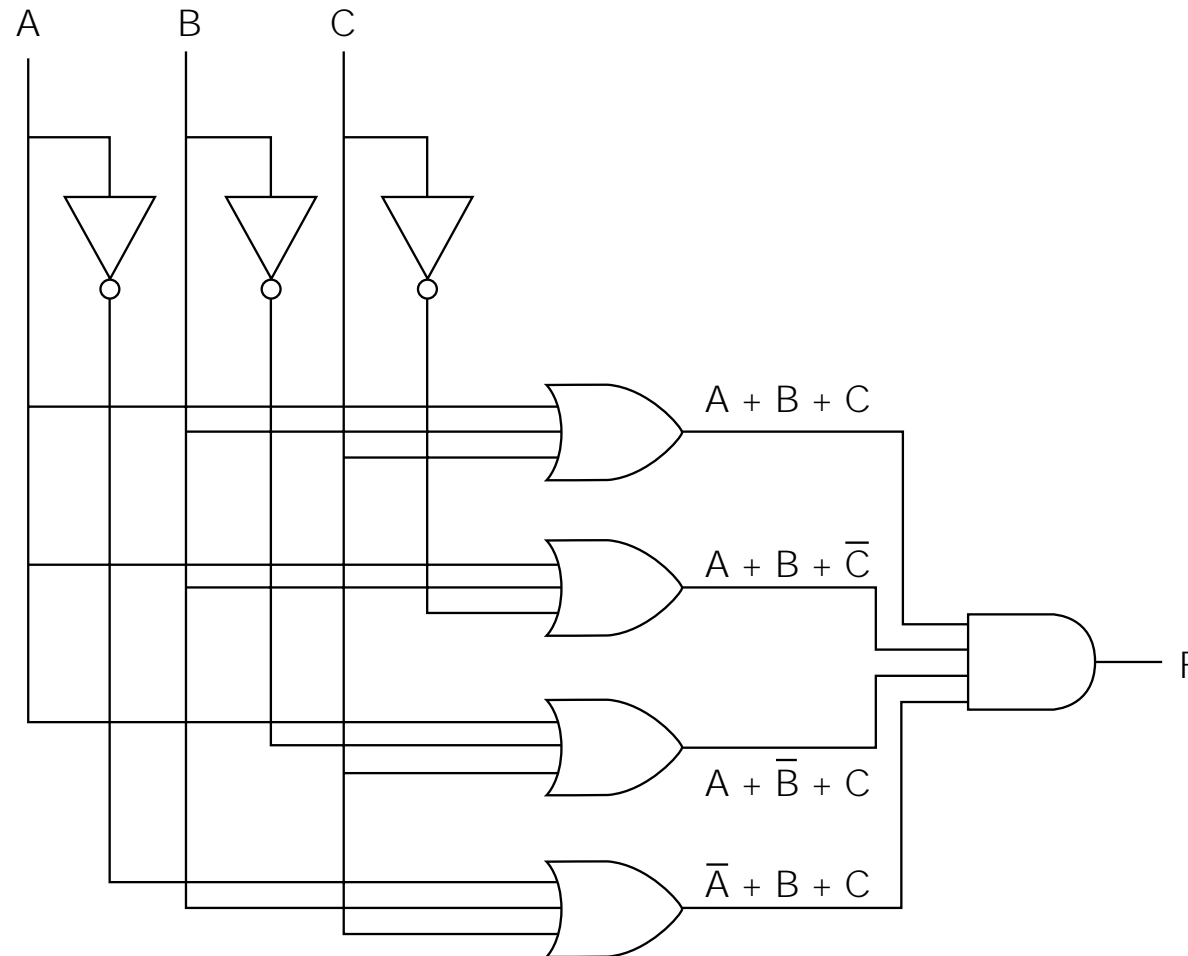


Connection



No connection

Fig A.17 A Two-Level OR-AND Circuit that Implements the Majority Function



Reduction (Simplification) of Boolean Expressions

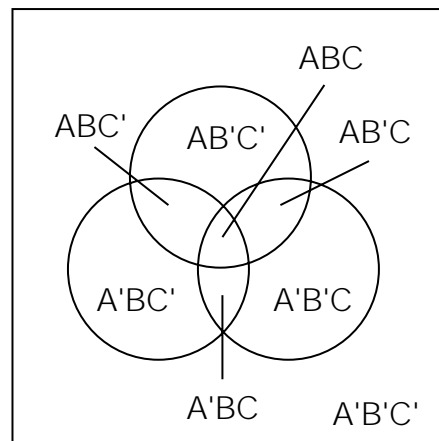
- It may be possible to simplify the canonical SOP or POS forms.
- A smaller Boolean equation translates to a lower gate count in the target circuit.
- We discuss two methods: algebraic reduction and Karnaugh map reduction.

The Algebraic Method

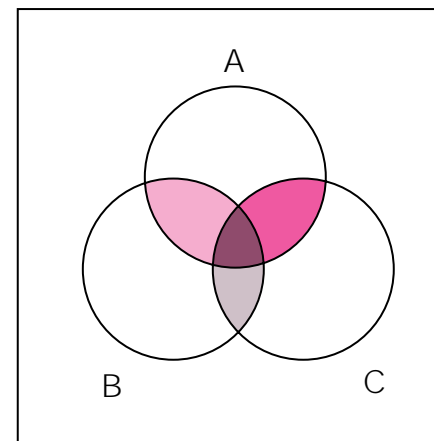
Consider the majority function, F :

$F = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$	
$F = \bar{A}BC + A\bar{B}C + AB(\bar{C} + C)$	Distributive property
$F = \bar{A}BC + A\bar{B}C + AB(1)$	Complement property
$F = \bar{A}BC + A\bar{B}C + AB$	Identity property
$F = \bar{A}BC + A\bar{B}C + AB + ABC$	Idempotence property
$F = \bar{A}BC + AC(\bar{B} + B) + AB$	Identity property
$F = \bar{A}BC + AC + AB$	Complement and identity properties
$F = \bar{A}BC + AC + AB + ABC$	Idempotence property
$F = BC(\bar{A} + A) + AC + AB$	Distributive property
$F = BC + AC + AB$	Complement and identity properties

Fig A.40 Venn Diagrams



3 binary variables



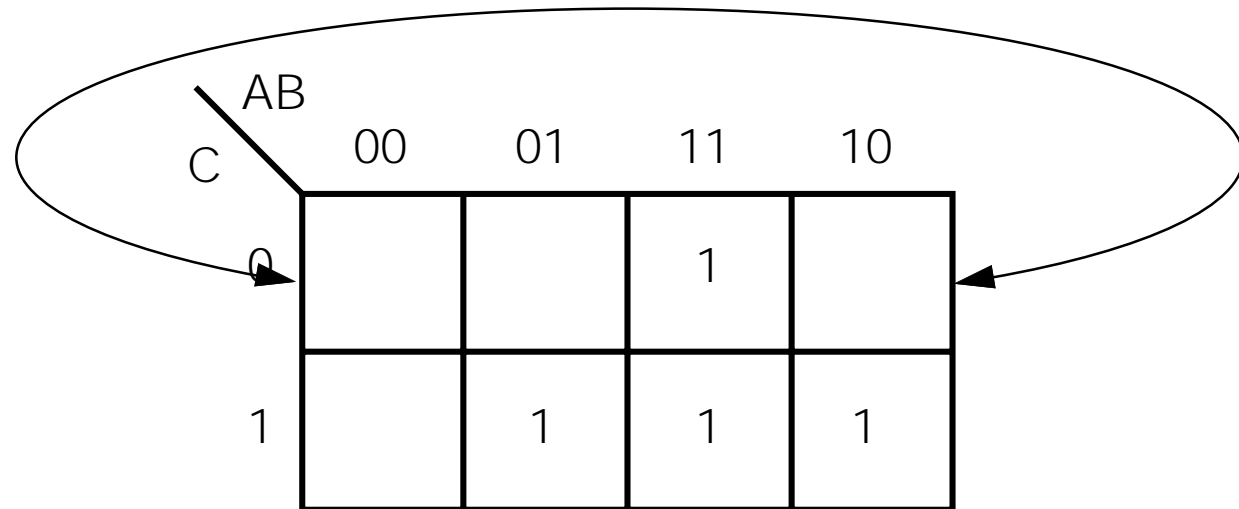
The majority function

**Each distinct region in the “universe” represents a minterm.
This diagram can be transformed into a Karnaugh Map.**

Fig A.41 A K-Map of the Majority Function

Place a “1” in each cell that has a that minterm.
Cells on the outer edge of the map “wrap around”

Minterm Index	A	B	C	F
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1



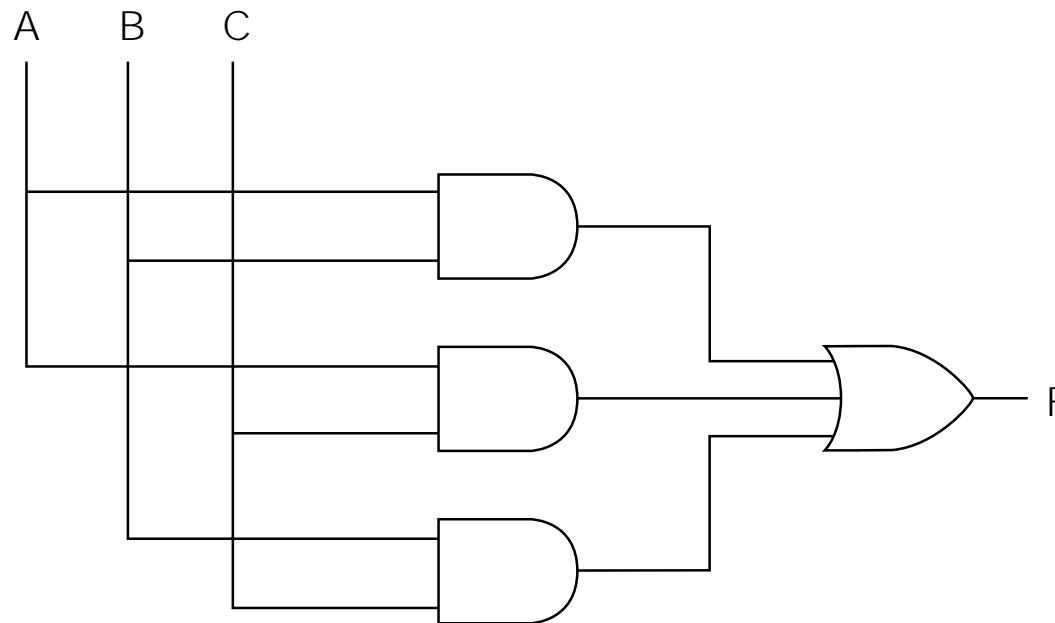
The map contains all the minterms. Adjacent 1's in the K-map satisfy the complement property of Boolean algebra.

Fig A.42 Adjacency Groupings for the Majority Function

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

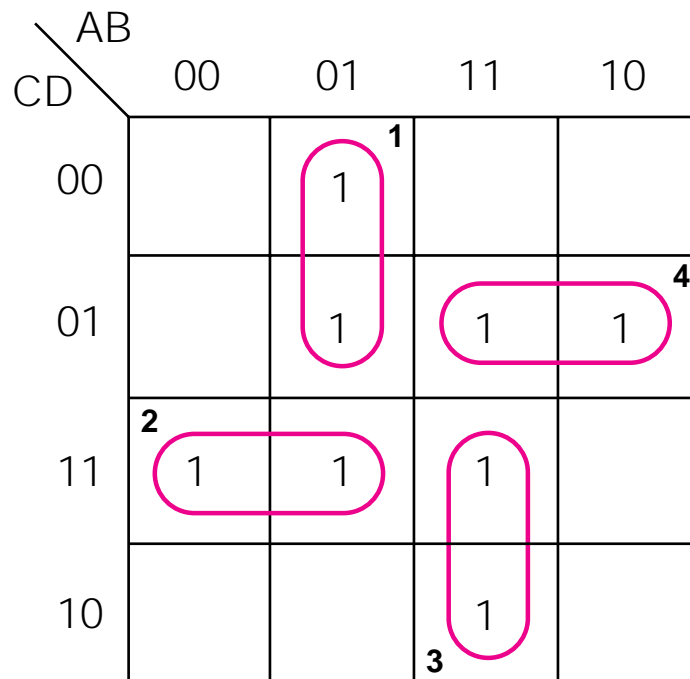
$$M = BC + AC + AB$$

A.43 Minimized AND-OR Circuit for the Majority Function

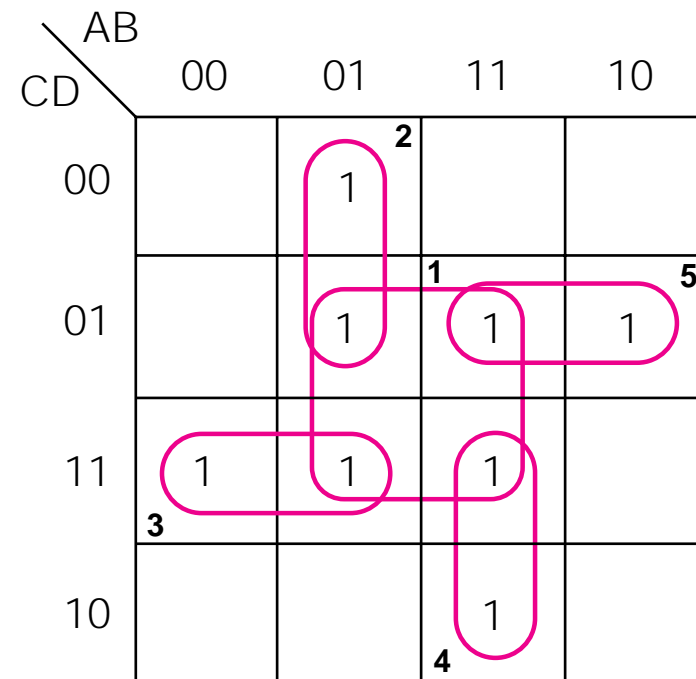


$$M = BC + AC + AB$$

Fig A.44 Minimal and Not-Minimal K-Map Groupings

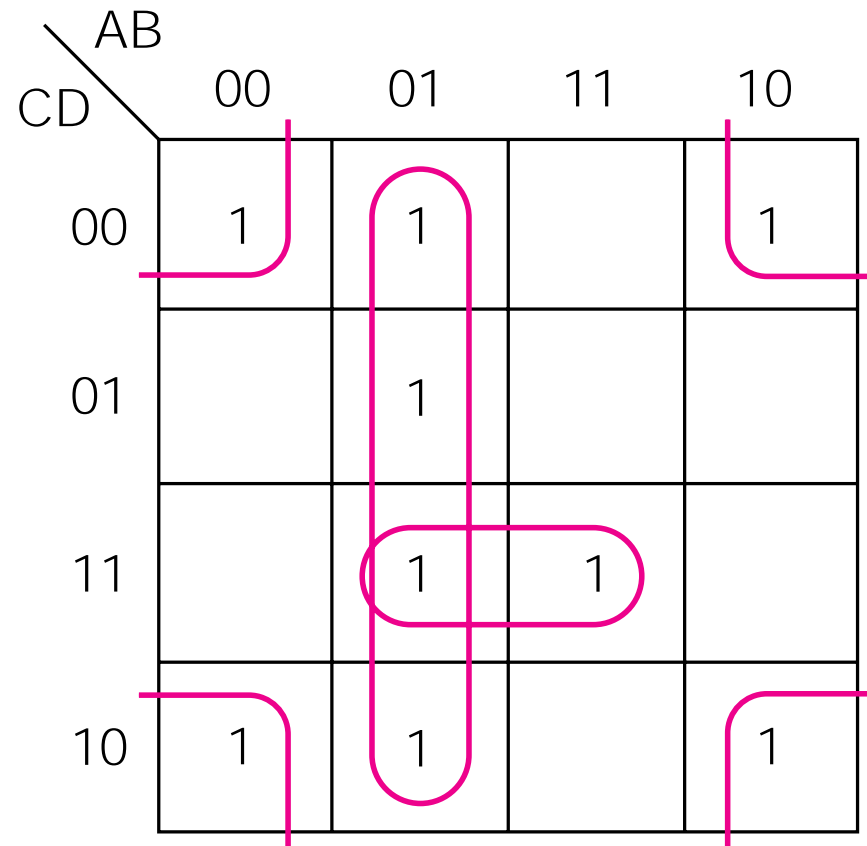


$$F = \bar{A}\bar{B}\bar{C} + \bar{A}CD + ABC + A\bar{C}\bar{D}$$



$$F = BD + \bar{A}\bar{B}\bar{C} + \bar{A}CD + ABC + A\bar{C}\bar{D}$$

Fig A.45 The Corners of a K-Map Are Logically Adjacent



$$F = BCD + \bar{B}\bar{D} + \bar{A}B$$

A.46 Two Different Minimized Equations Are Produced from the Same K-Map

		AB			
		00	01	11	10
CD	00	1			d
	01		1	1	
	11		1	1	
	10	d			

$$F = \overline{B}\overline{C}\overline{D} + BD$$

		AB			
		00	01	11	10
CD	00	1			d
	01		1	1	
	11		1	1	
	10	d			

$$F = \overline{A}\overline{B}\overline{D} + BD$$