Sum-of-Products (SOP) v.s. Product-of-Sums (POS)

- All Boolean equations can be written in SOP form
- Each row has a **minterm**
- A minterm is a **product** (AND) of literals
- Each minterm is **TRUE** for that row (and only that row)
- Form function by **ORing minterms** where **output is 1**
- Thus, a **sum** (OR) of **products** (AND terms)

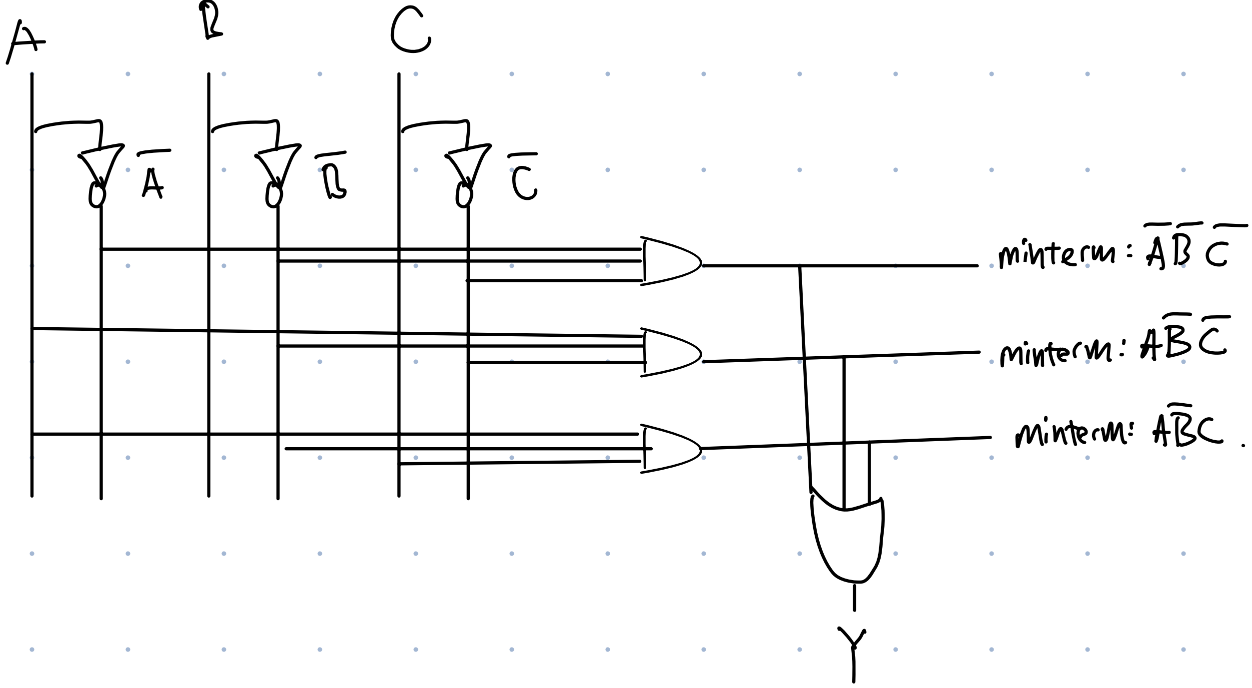
- All Boolean equations can be written in POS form
- Each row has a **maxterm**
- A maxterm is a **sum** (OR) of literals
- Each maxterm is **FALSE** for that row (and only that row)
- Form function by **ANDing maxterms** where **output is 0**
- Thus, a **product** (AND) of **sums** (OR terms)

A	*B*	minterm	minterm name	*Y*	maxterm	maxterm name
0	0	$\bar{A}\bar{B}$	m_0	0	$A+B$	M_0
0	1	$\bar{A}B$	m_1	1	$A+\bar{B}$	M_1
1	0	$A\bar{B}$	m_2	0	$\bar{A}+B$	M_2
1	1	AB	m_3	1	$\bar{A}+\bar{B}$	M_3

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

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

From logic to gates: $Y = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$



Boolean Theorems of Several Vars

#	Theorem	Dual	Name
T6	B•C = C•B	B+C = C+B	Commutativity
T7	(B•C) • D = B • (C•D)	(B + C) + D = B + (C + D)	Associativity
T8	B • (C + D) = (B•C) + (B•D)	B + (C•D) = (B+C) (B+D)	Distributivity
T9	B • (B+C) = B	B + (B•C) = B	Covering
T10	(B•C) + (B•$\overline{C}$) = B	(B+C) • (B+$\overline{C}$) = B	Combining

Axioms and theorems are useful for *simplifying* equations.

Different from traditional algebra.

ELSEVIER

DeMorgan's Theorem

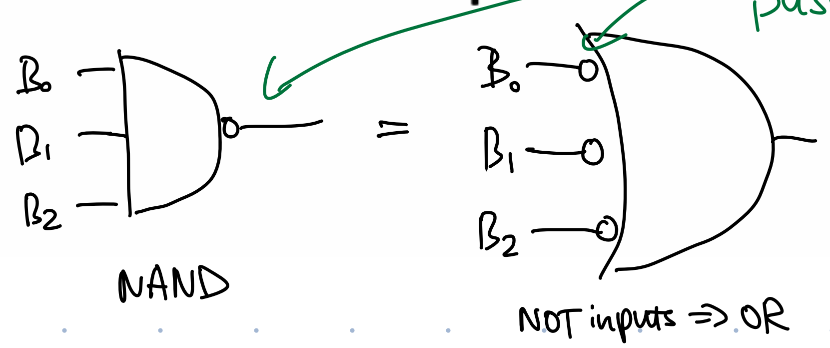
Number	Theorem	Name
T12	$\overline{B_0 \bullet B_1 \bullet B_2 \ldots} = \overline{B_0} + \overline{B_1} + \overline{B_2} \ldots$	DeMorgan's Theorem

The **complement** of the **product**

is the

sum of the **complements**

bubble pushing

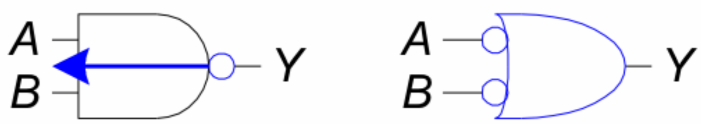


Convincing yourself of DeMorgan's Theorem.

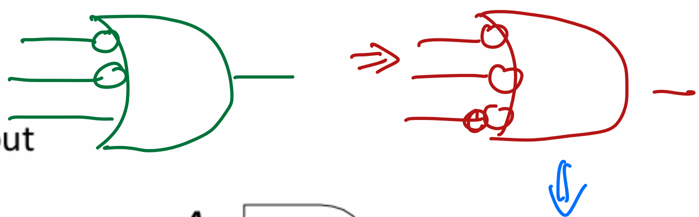
A	B	$\overline{AB}$	$\overline{A}+\overline{B}$
0	0	1	1
0	1	1	1
1	0	1	1
1	1	0	0

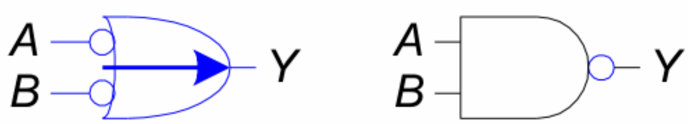
Bubble Pushing

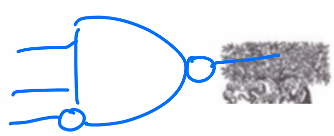
- **Backward:**
  - Body changes
  - Adds bubbles to inputs



- **Forward:**
  - Body changes
  - Adds bubble to output



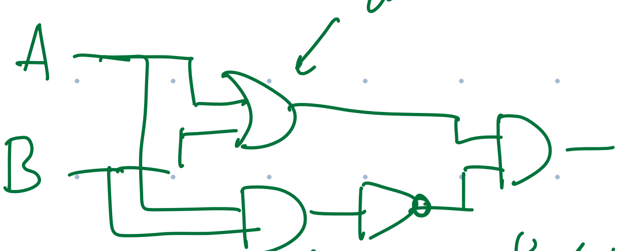




XOR

	A	B	Y
all 0 ✗ 1 0 ✓	0	0	0
all 0 ✗ 1 0 ✓	1	0	1
all 0 ✗ 1 0 ✓	0	1	1
all 0 ✗ 1 0 ✗	1	1	0

check if both.



check if 1 is 0.

K-maps are useful for simplifying equations

K-Map Rules

- **Every 1 must be circled** at least once
- Each circle must span a **power of 2** (i.e. 1, 2, 4) squares in each direction
- Each circle must be as **large** as possible
- A circle may **wrap around the edges**
- Circle a "**don't care**" (X) **only if it helps** minimize the equation

ELSEVIER