Combinational Logic

Jia Chen
jiac@ucr.edu

Slides adopted from Hung-Wei Tseng

Outline

- Two types of logics
- The **theory** behind combinational logics
- The building blocks of combinational logics

Types of circuits

Combinational v.s. sequential logic

- Combinational logic
  - The output is a pure function of its current inputs
  - The output doesn't change regardless how many times the logic is triggered — Idempotent
- Sequential logic
  - The output depends on current inputs, previous inputs, their history

When to use combinational logic?

- How many of the following can we simply use combinational logics to accomplish?
  ① Counters
  ② Adders
  ③ Memory cells
  ④ Decimal to 7-segment LED-decoders
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

When to use combinational logic?

- How many of the following can we simply use combinational logics to accomplish?
  ① Counters — You need the previous input
  ② Adders
  ③ Memory cells — You need to keep the current state
  ④ Decimal to 7-segment LED-decoders

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Theory behind each

- A **Combinational logic** is the implementation of a **Boolean Algebra** function with only Boolean Variables as their inputs
- A **Sequential logic** is the implementation of a **Finite-State Machine**

Boolean Algebra

Boolean algebra

- Boolean algebra — George Boole, 1815—1864
  - Introduced binary variables
  - Introduced the three fundamental logic operations: AND, OR, and NOT
  - Extended to abstract algebra with set operations: intersect, union, complement

Basic Boolean Algebra Concepts

- {0, 1}: The only two possible values in inputs/outputs
- Basic operators
  - AND (•) — a • b
    - returns 1 only if both a **and** b are 1s
    - otherwise returns 0
  - OR (+) — a + b
    - returns 1 if a **or** b is 1
    - returns 0 if none of them are 1s
  - NOT (‘) — a’
    - returns 0 if a is 1
    - returns 1 if a is 0

Truth tables

- A table sets out the functional values of logical expressions on each of their functional arguments, that is, for each combination of values taken by their logical variables

AND

Input		Output
A	B	
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

OR

Input		Output
A	B	
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

NOT

Input	Output
A	
0	1
0	1
1	0
1	0

Let's practice!

- X, Y are two Boolean variables. Consider the following function:
  $X \cdot Y + X$
  How many of the following the input values of X and Y can lead to an output of 1
  ① X = 0, Y = 0
  ② X = 0, Y = 1
  ③ X = 1, Y = 0
  ④ X = 1, Y = 1
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Let's practice!

- X, Y are two Boolean variables. Consider the following function:
  X • Y' + X
  How many of the following the input values of X and Y can lead to an output of 1
  1. X = 0, Y = 0
  2. X = 0, Y = 1
  3. X = 1, Y = 0
  4. X = 1, Y = 1

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Input					Output
X	Y	Y'	XY'	XY' + X	
0	0	1	0	0	**0**
0	1	0	0	0	**0**
1	0	1	1	1	**1**
1	1	0	0	1	**1**

Derived Boolean operators

- NAND — $(a \cdot b)'$
- NOR — $(a + b)'$
- XOR — $(a + b) \cdot (a' + b')$ or $ab' + a'b$
- XNOR — $(a + b') \cdot (a' + b)$ or $ab + a'b'$

NAND

Input		Output
A	B	
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

NOR

Input		Output
A	B	
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

XOR

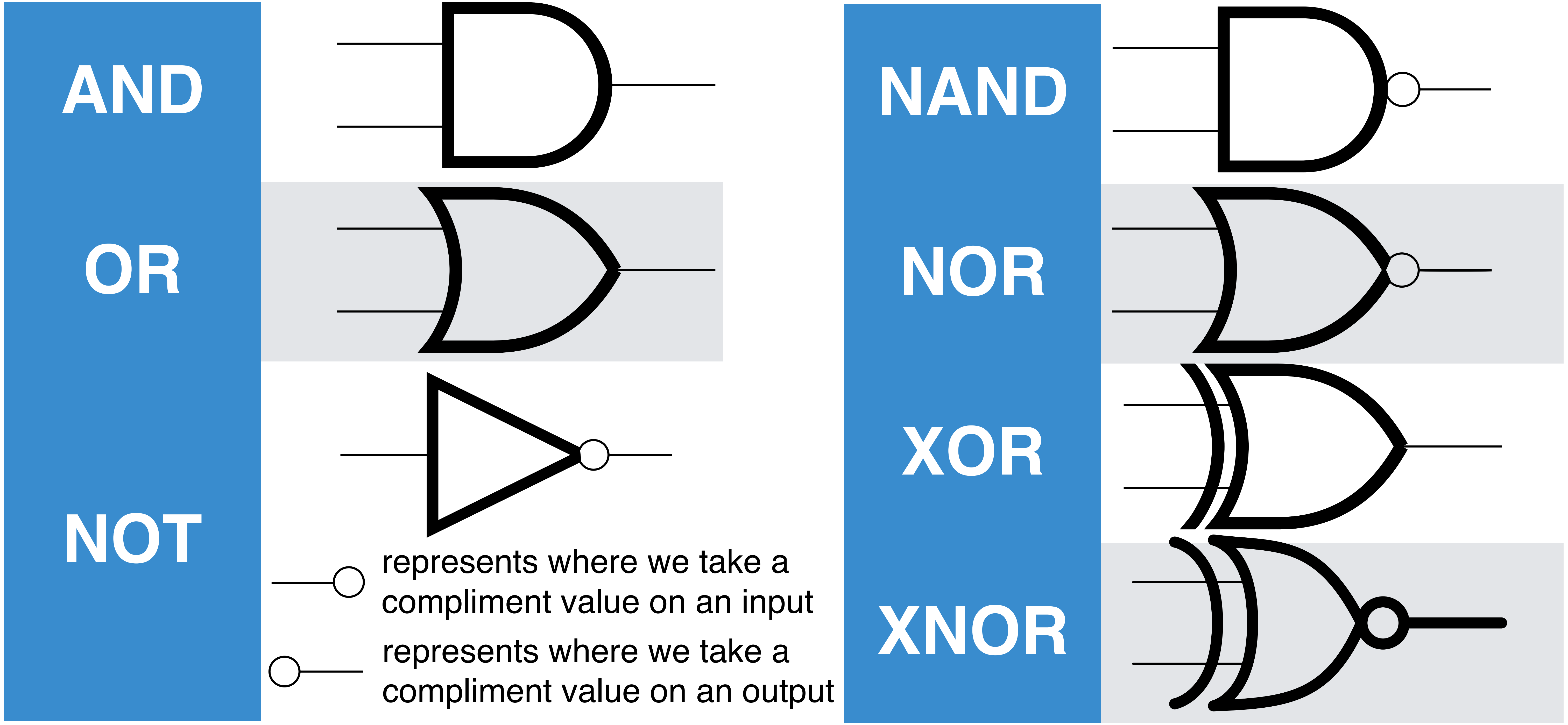
Input		Output
A	B	
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

XNOR

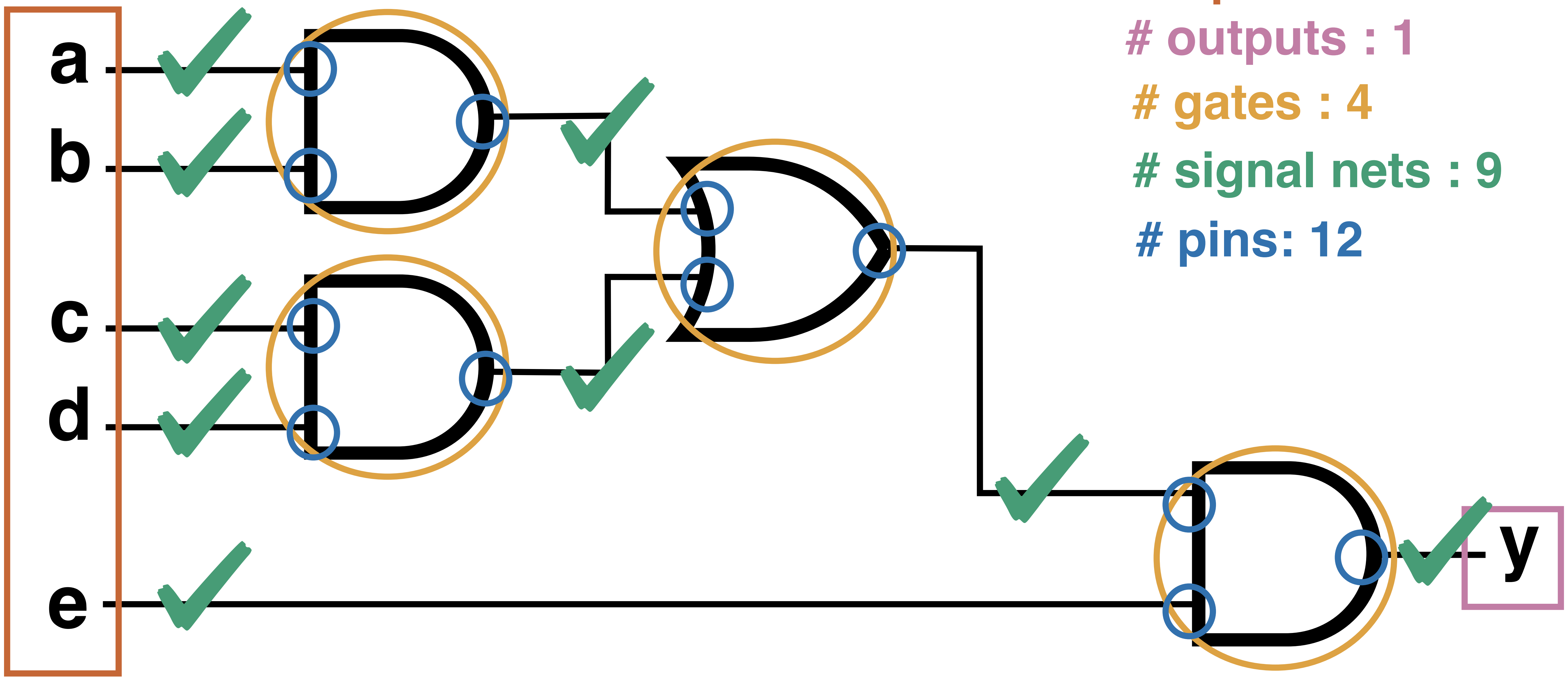
Input		Output
A	B	
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

Express Boolean Operators/Functions in Circuit "Gates"

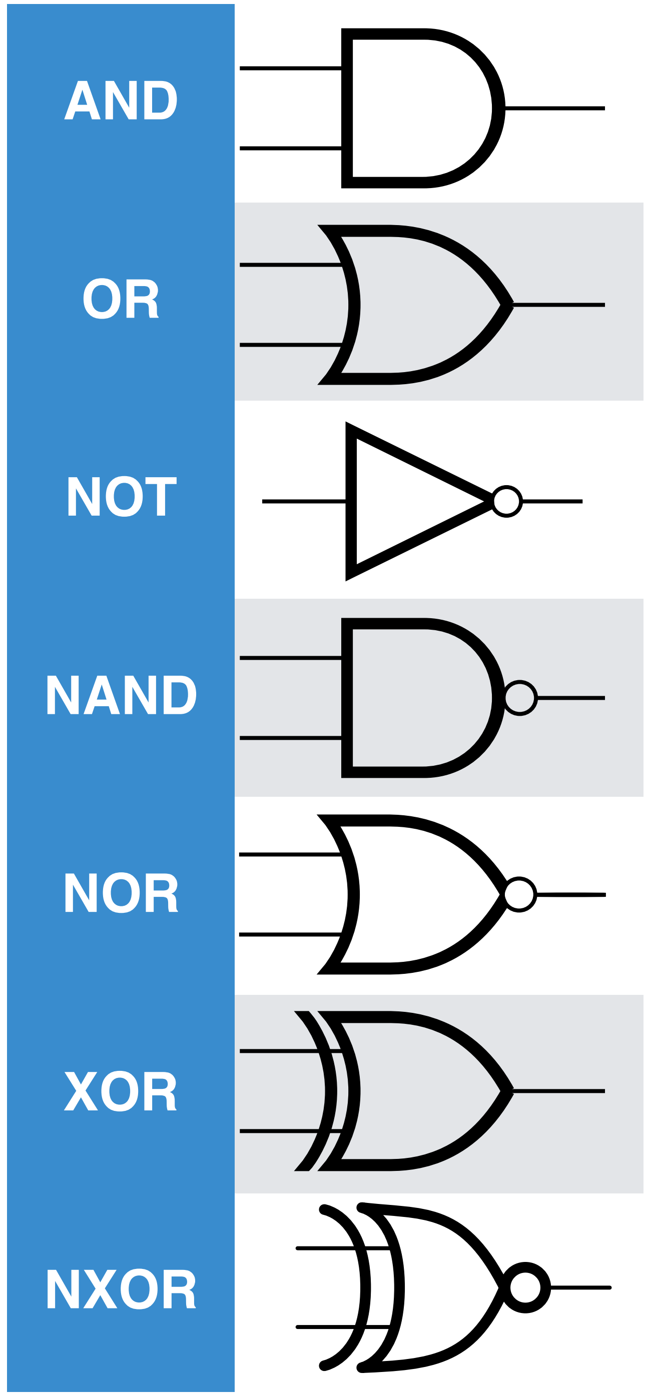
Boolean operators their circuit “gate” symbols



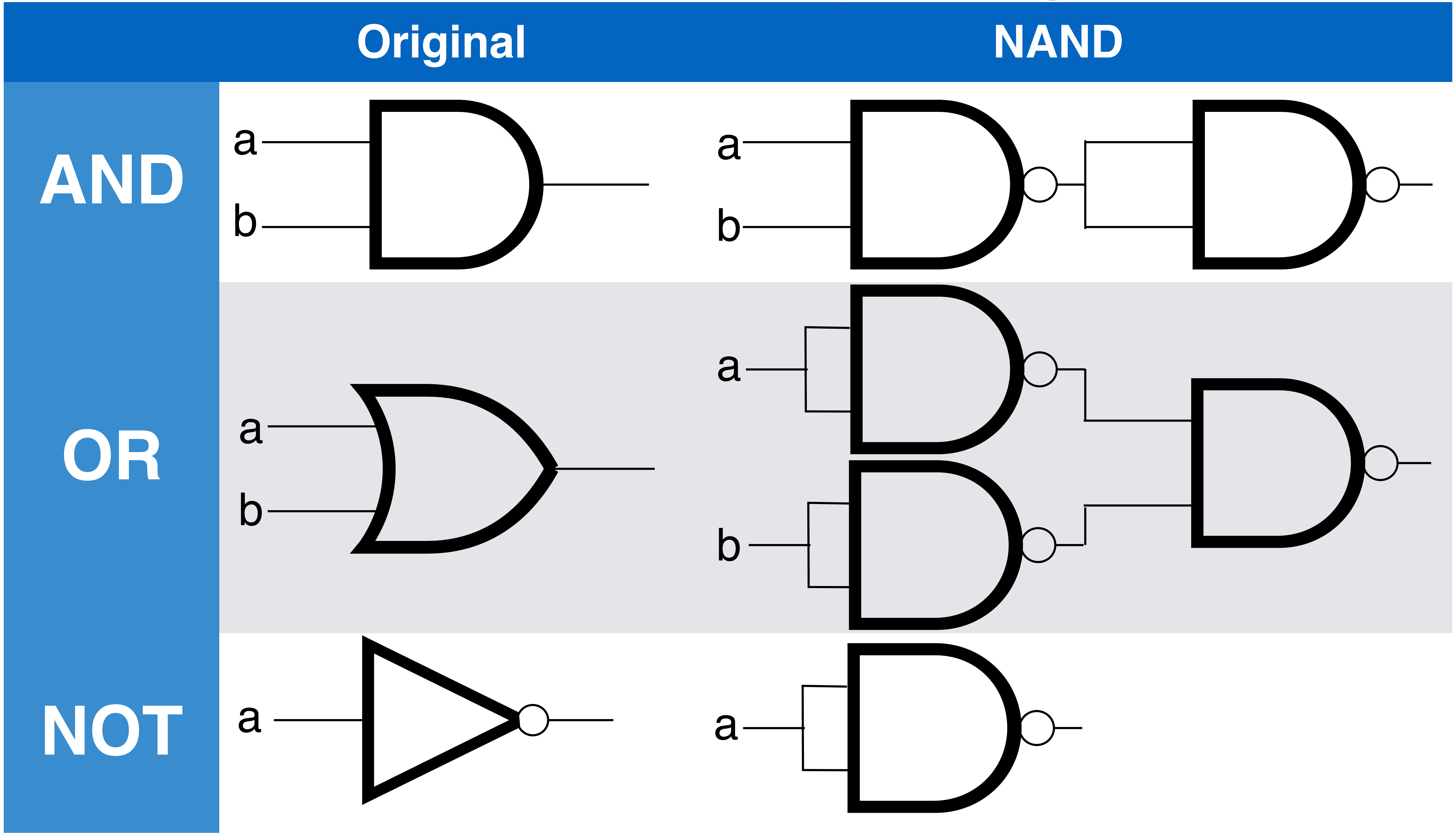
How to express y = e(ab+cd)



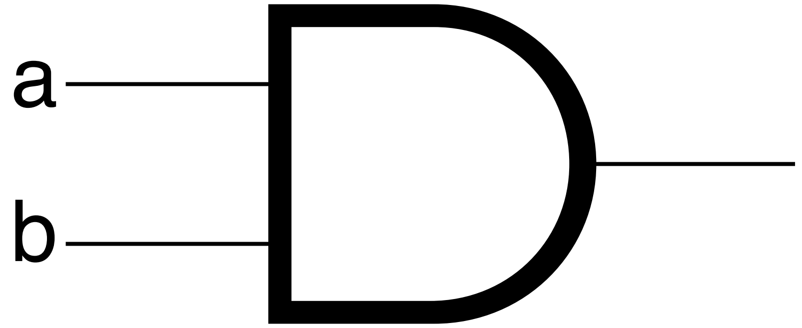
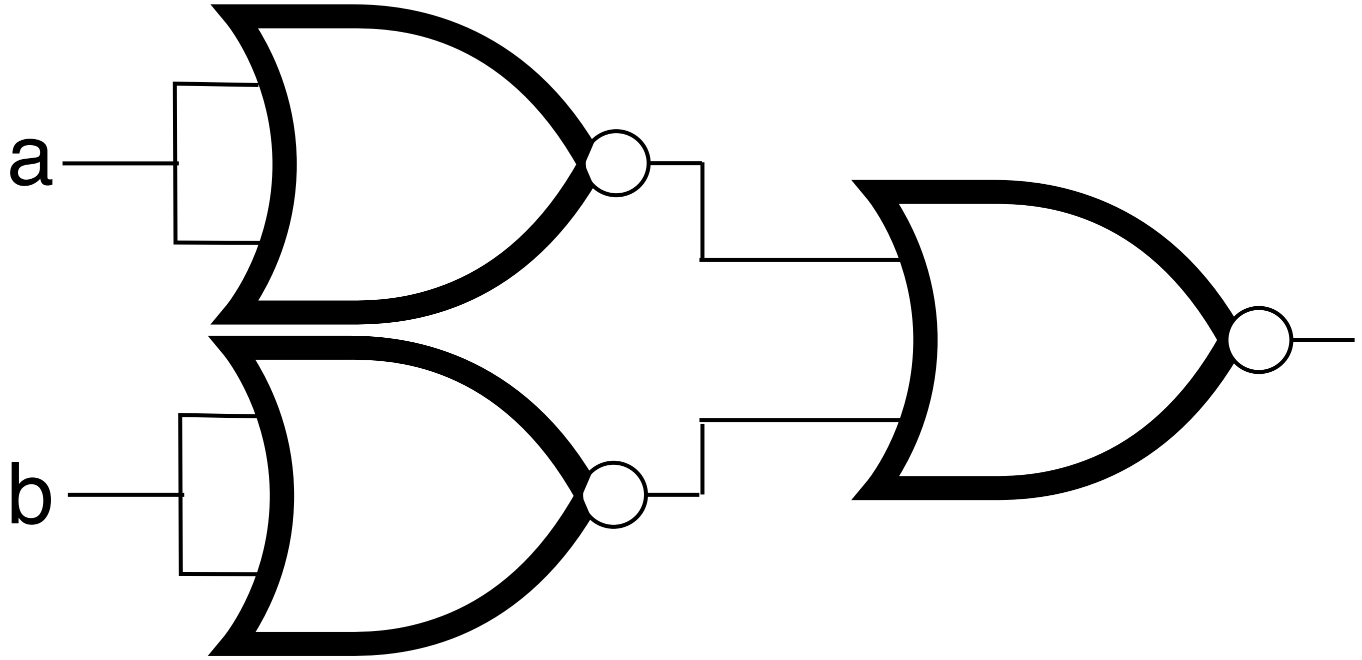
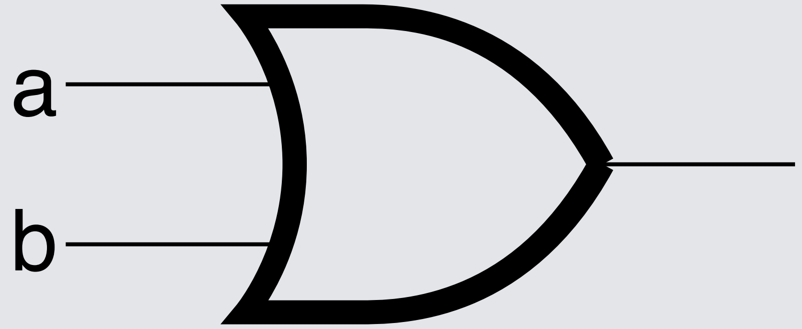
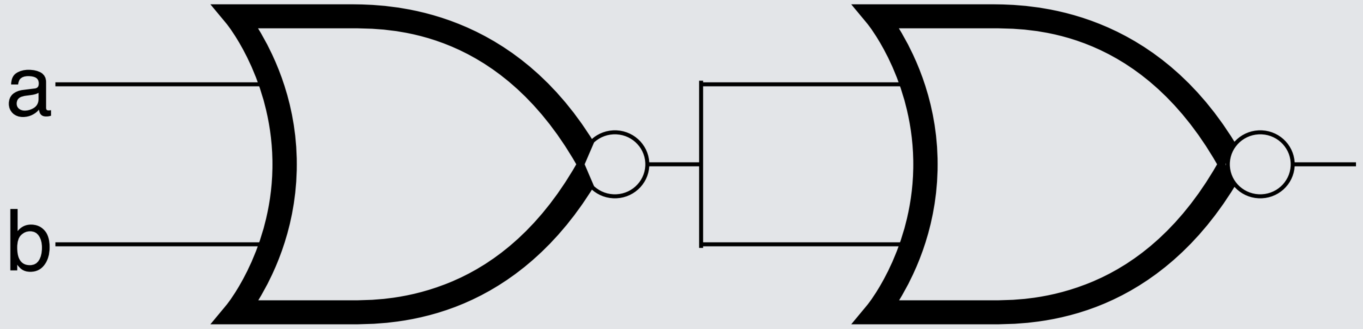
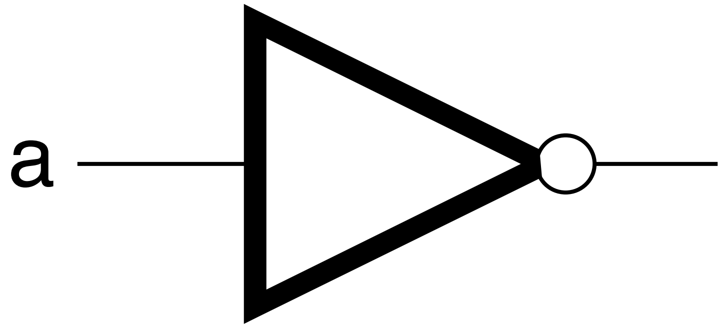
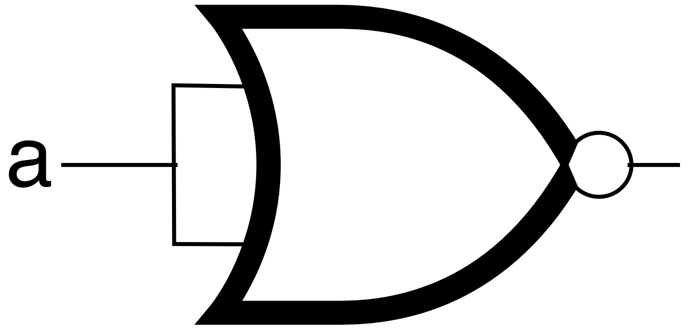
inputs : 5
outputs : 1
gates : 4
signal nets : 9
pins: 12



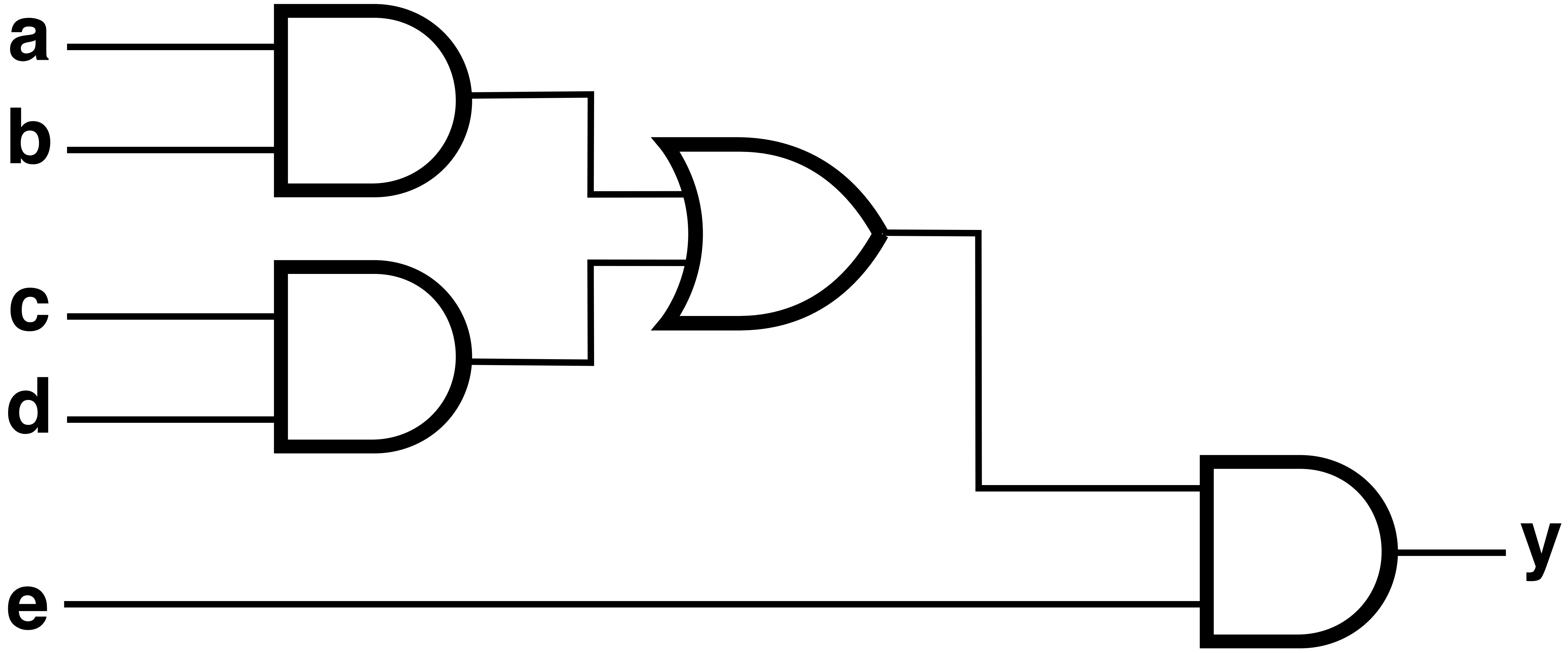
We can make everything NAND!



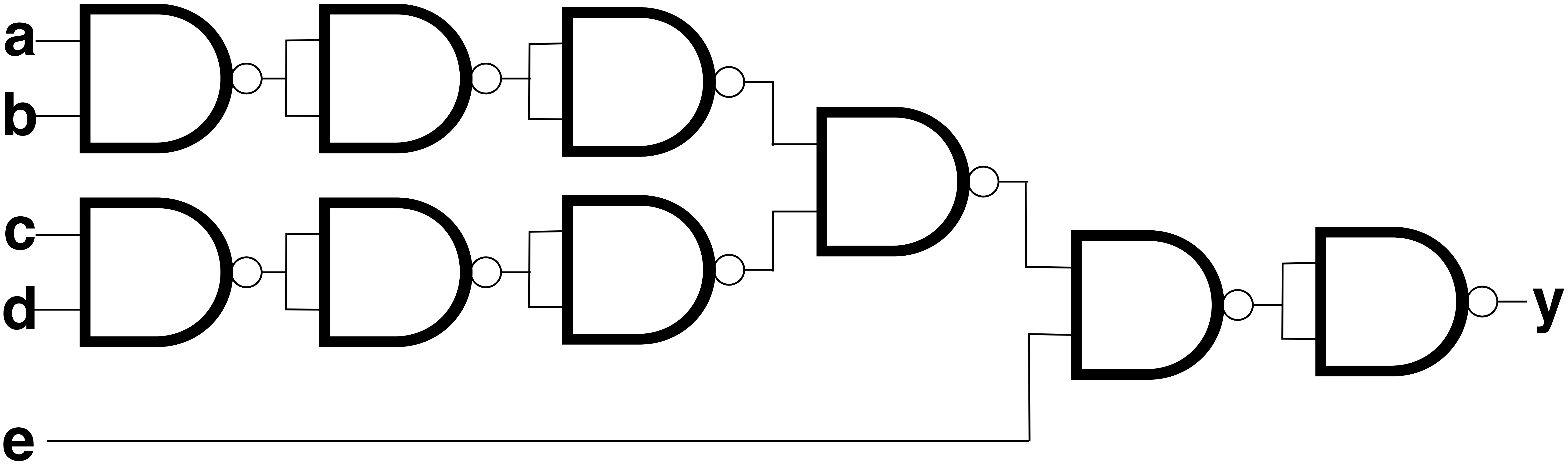
We can also make everything NOR!

	Original	NAND
AND		
OR		
NOT		

How to express y = e(ab+cd)



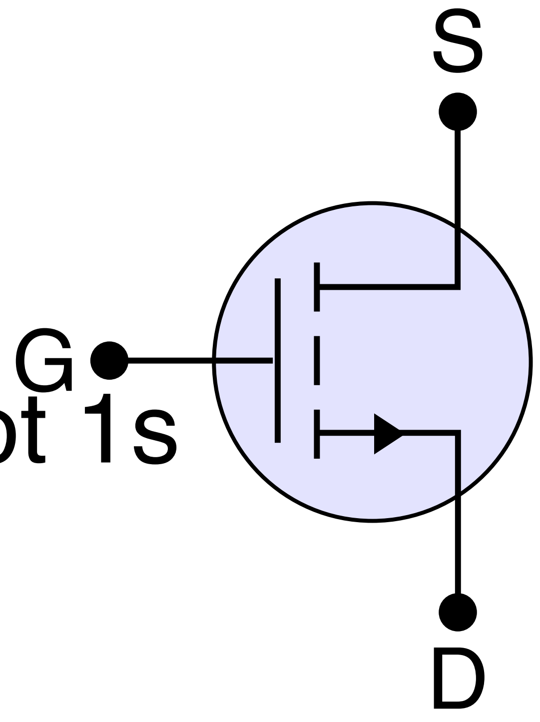
How to express y = e(ab+cd)



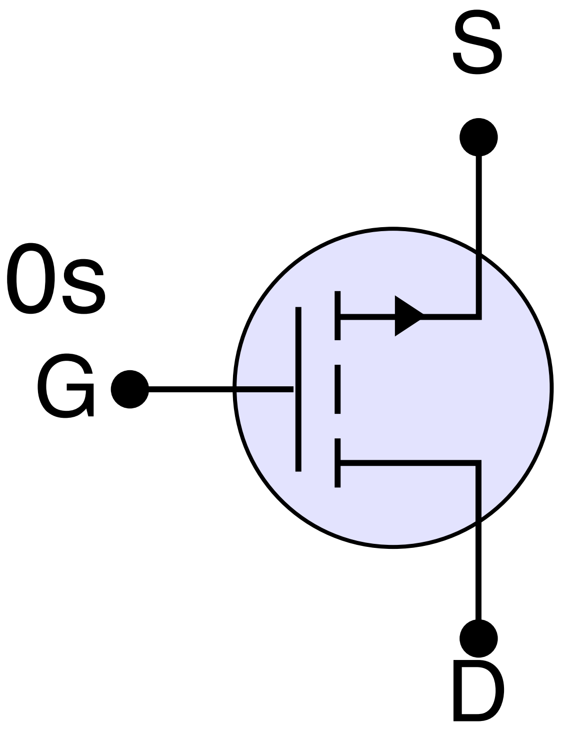
How gates are implemented?

Two type of CMOSs

- nMOS
  - Turns on when G = 1
  - When it's on, passes 0s, but not 1s
  - Connect S to ground (0)
  - Pulldown network

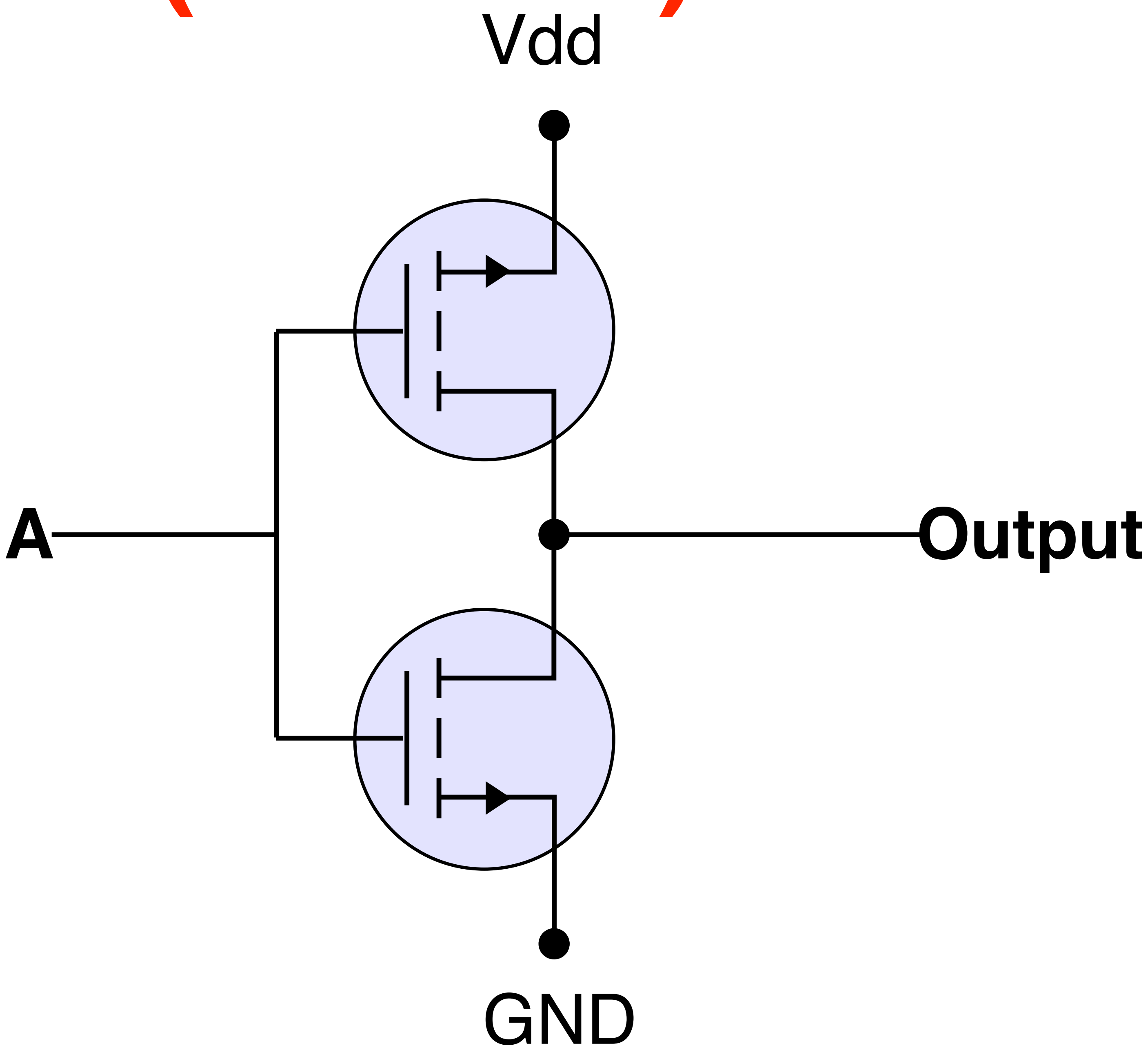


- pMOS
  - Turns on when G = 0
  - When it's on, passes 1s, but not 0s
  - Connect S to Vdd (1)
  - Pullup network

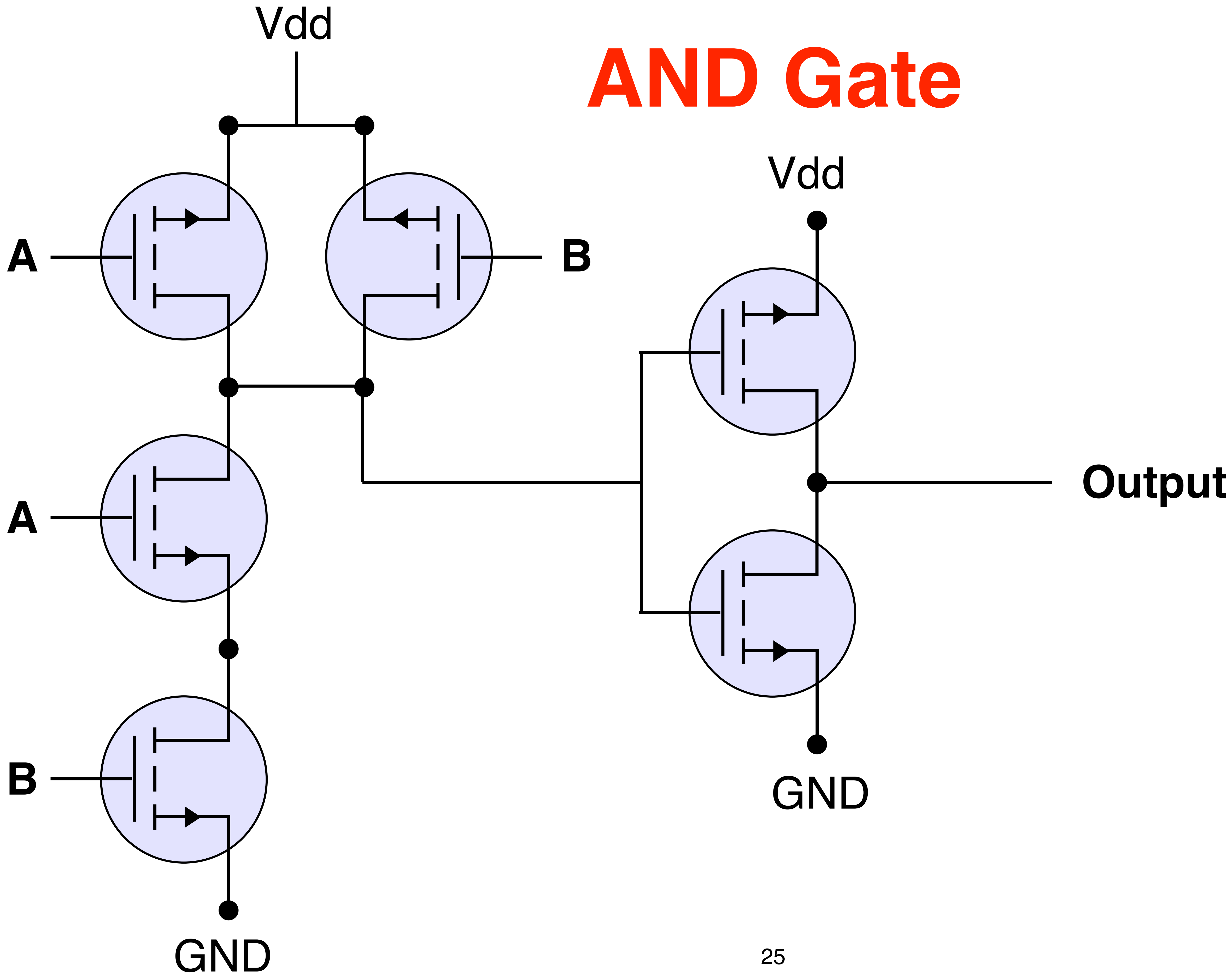


NOT Gate (Inverter)

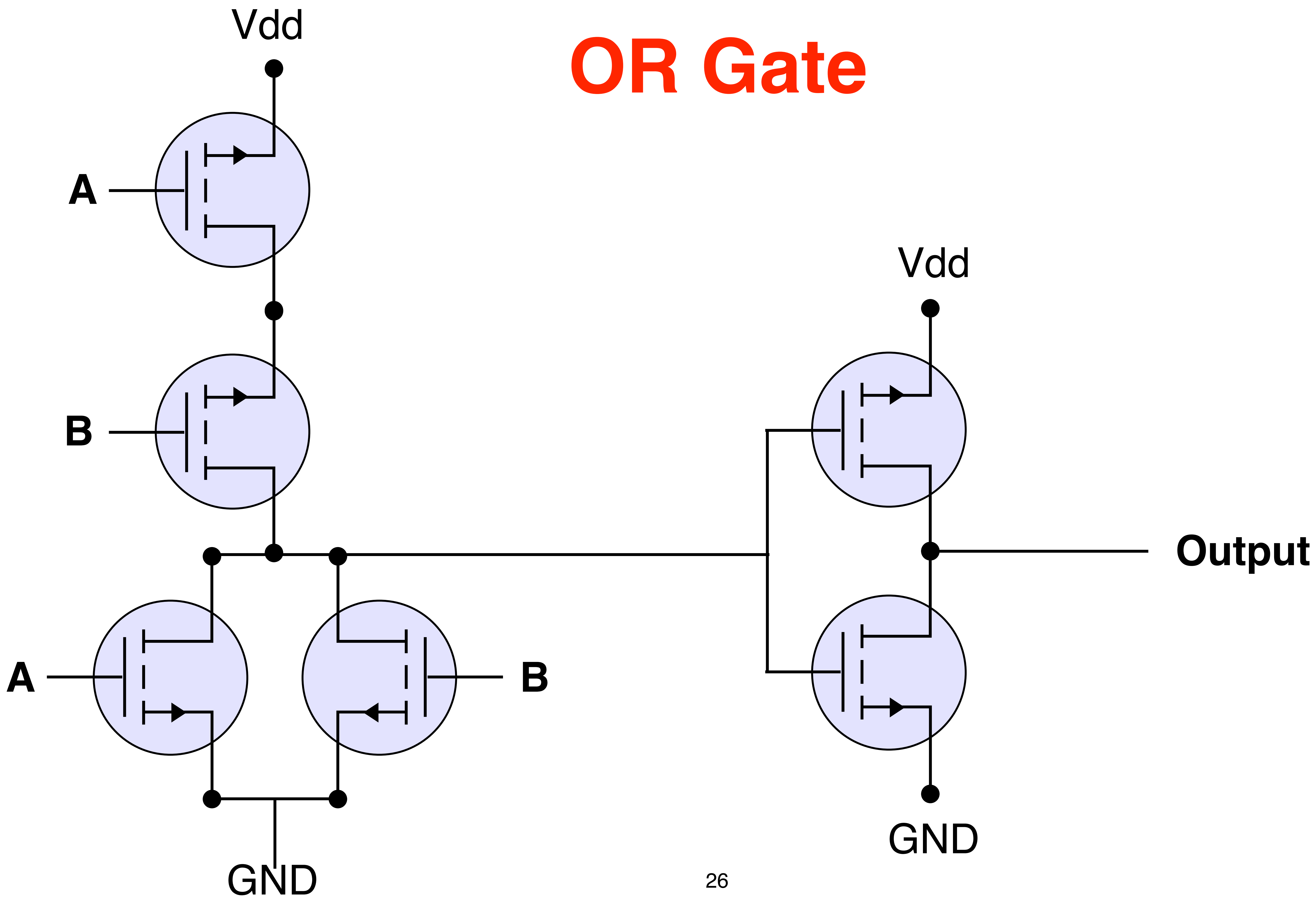
Input A	NMOS (passes 0 when on G=1)	PMOS (passes 1 when on G=0)	Output
0	OFF	ON	1
1	ON	OFF	0



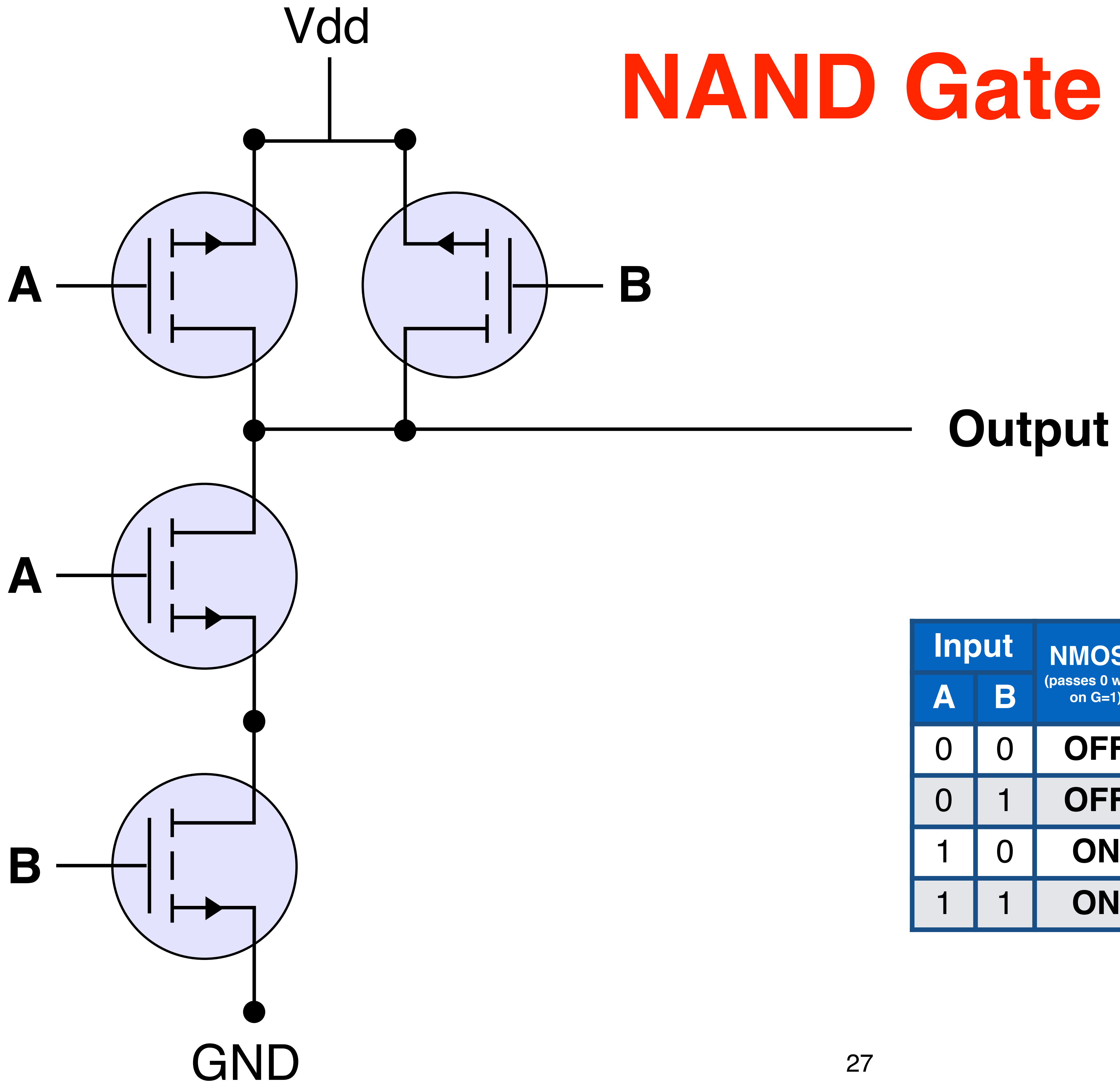
AND Gate



OR Gate



NAND Gate

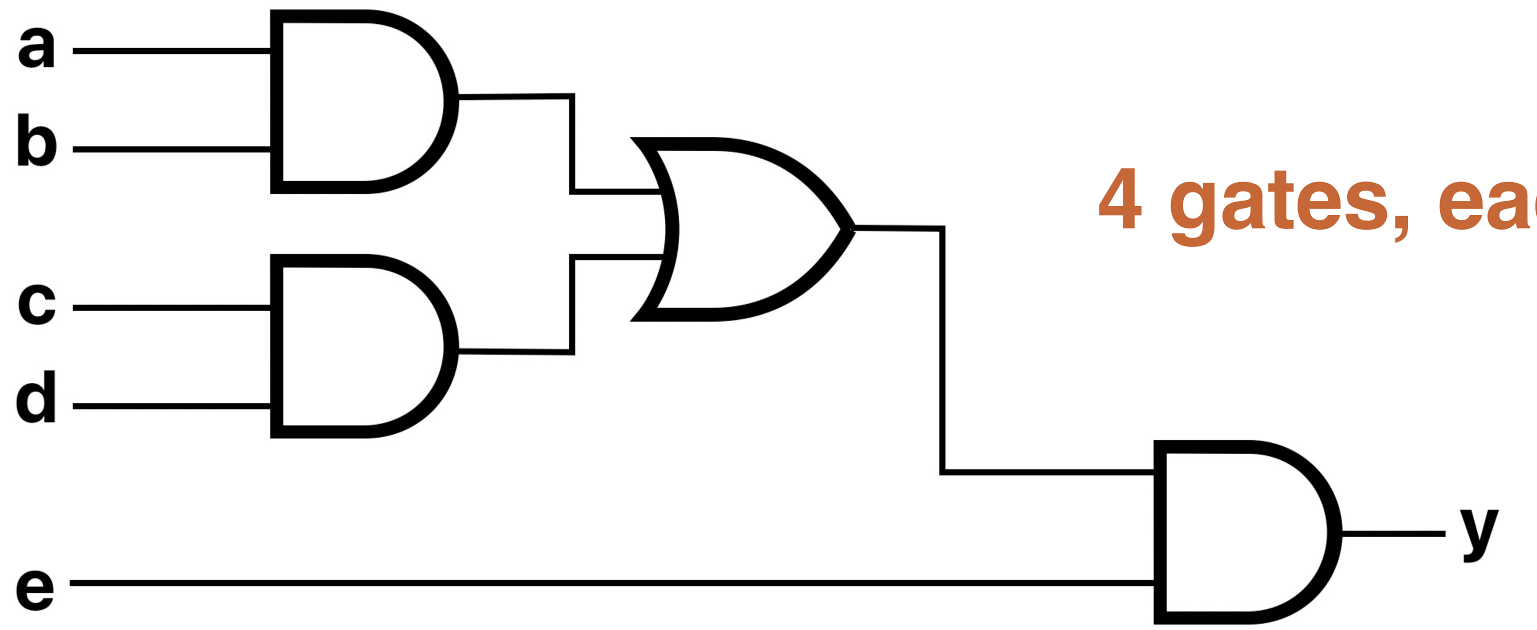


Input		NMOS1 (passes 0 when on G=1)	PMOS1 (passes 1 when on G=0)	NMOS2 (passes 0 when on G=1)	PMOS2 (passes 1 when on G=0)	Output
A	B					
0	0	OFF	ON	OFF	ON	1
0	1	OFF	ON	ON	OFF	1
1	0	ON	OFF	OFF	ON	1
1	1	ON	OFF	ON	OFF	0

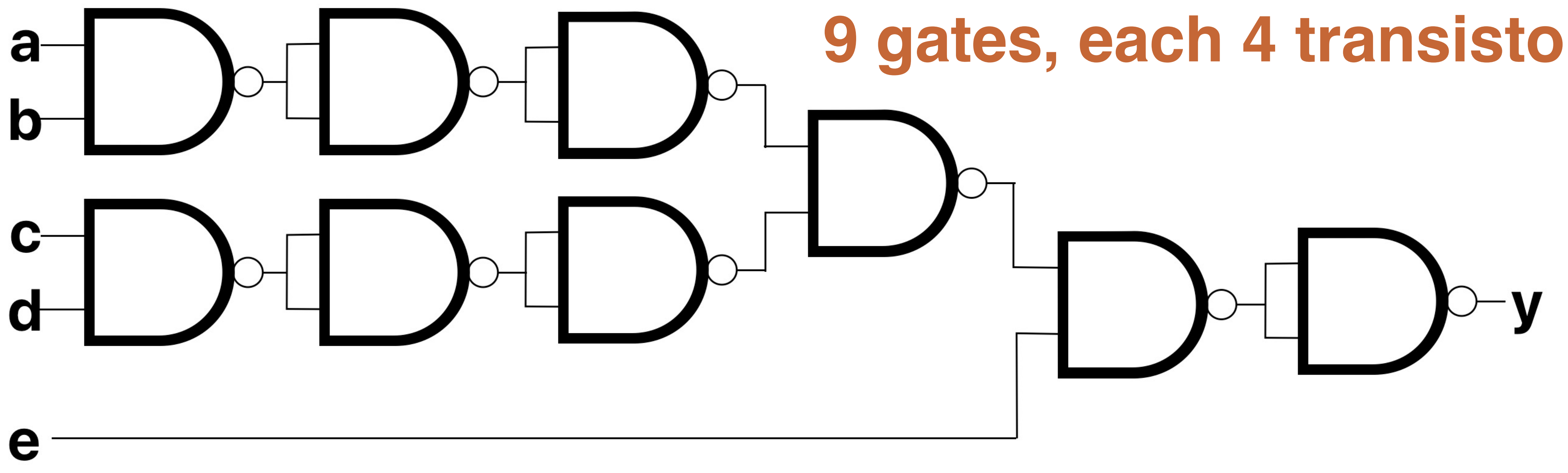
Why use NAND?

- NAND and NOR are “universal gates” — you can build any circuit with everything NAND or NOR
- Simplifies the design as you only need one type of gate
- NAND only needs 4 transistors — gate delay is smaller than OR/AND that needs 6 transistors
- NAND is slightly faster than NOR due to the physics nature

How about total number of transistors?



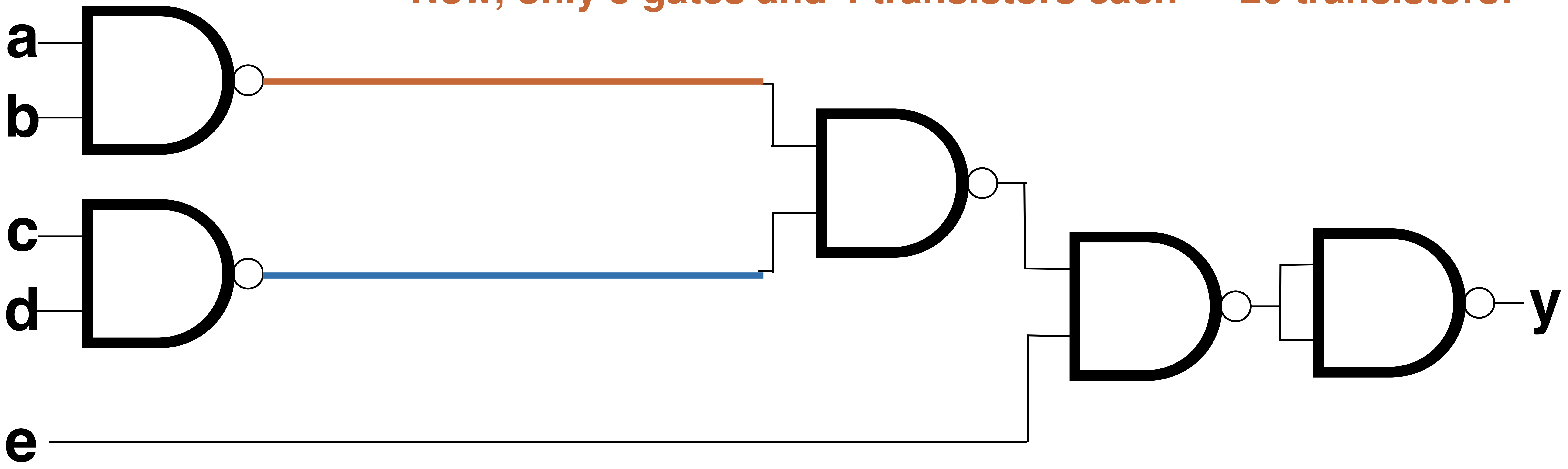
4 gates, each 6 transistors : total 24 transistors



9 gates, each 4 transistors : total 36 transistors

However …

Now, only 5 gates and 4 transistors each — 20 transistors!



How big is the truth table of $y = e(ab+cd)$

- How many rows do we need to express the circuit represented by $y = e(ab+cd)$
  A. 5
  B. 9
  C. 25
  D. 32
  E. 64

How big is the truth table of y = e(ab+cd)

- How many rows do we need to express the circuit represented by y = e(ab+cd)
    A. 5
    B. 9
    C. 25
    D. $\boxed{32}$
    E. 64

$$2 \times 2 \times 2 \times 2 \times 2 = 2^5 = 32$$

Boolean expression is a lot more compact than a truth table!

Can We Get the Boolean Equation from a Truth Table?

Definitions of Boolean Function Expressions

- Complement: variable with a bar over it or a ‘ — A’, B’, C’
- Literal: variable or its complement — A, A’, B, B’, C, C’
- Implicant (Product term): product of literals — ABC, AC, BC
- Implicate (Sum terms): sum of literals — (A+B+C), (A+C), (B+C)
- Minterm: AND that includes all input variables — ABC, A’BC, AB’C
- Maxterm: OR that includes all input variables — (A+B+C), (A’+B+C), (A’+B’+C)

Canonical form — Sum of "Minterms"

Input		Output
X	Y	
0	0	**0**
0	1	**0**
1	0	**1**
1	1	**1**

A minterm

f(X,Y) = XY' + XY

Sum (OR) of "product" terms

XNOR

Input		Output
A	B	
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

f(A,B) = A'B' + AB

Canonical form — Product of "Maxterms"

Input		Output
X	Y	
0	0	**0**
0	1	**0**
1	0	**1**
1	1	**1**

A "maxterm

f(X,Y) =(X+Y) (X + Y') ← Product of maxterms

XNOR

Input		Output
A	B	
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

f(A,B) = (A+B') (A'+B)

Minterms and Maxterms

Row number	x_1	x_2	x_3	Minterm	Maxterm
0	0	0	0	$m_0 = \overline{x}_1\overline{x}_2\overline{x}_3$	$M_0 = x_1 + x_2 + x_3$
1	0	0	1	$m_1 = \overline{x}_1\overline{x}_2x_3$	$M_1 = x_1 + x_2 + \overline{x}_3$
2	0	1	0	$m_2 = \overline{x}_1x_2\overline{x}_3$	$M_2 = x_1 + \overline{x}_2 + x_3$
3	0	1	1	$m_3 = \overline{x}_1x_2x_3$	$M_3 = x_1 + \overline{x}_2 + \overline{x}_3$
4	1	0	0	$m_4 = x_1\overline{x}_2\overline{x}_3$	$M_4 = \overline{x}_1 + x_2 + x_3$
5	1	0	1	$m_5 = x_1\overline{x}_2x_3$	$M_5 = \overline{x}_1 + x_2 + \overline{x}_3$
6	1	1	0	$m_6 = x_1x_2\overline{x}_3$	$M_6 = \overline{x}_1 + \overline{x}_2 + x_3$
7	1	1	1	$m_7 = x_1x_2x_3$	$M_7 = \overline{x}_1 + \overline{x}_2 + \overline{x}_3$

Sum-of-Products Form

- Represent a function f by a sum of minterms
- Each minterm is ANDed with the value of f

Row number	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

$f = m_1 + m_4 + m_5 + m_6$

$= \sum(m_1, m_4, m_5, m_6)$

$= \sum m(1, 4, 5, 6)$

$= \sum_{x_1, x_2, x_3}(1, 4, 5, 6)$

$= x_1' x_2' x_3 + x_1 x_2' x_3' + x_1 x_2' x_3 + x_1 x_2 x_3'$

- A logic expression in the sum-of-products (SoP) form:
  - Consisting of product (AND) terms that are summed (ORed)
- Canonical SoP: each product term is a minterm

Product-of-Sums Form

Row number	x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	0

$$f = \mathrm{M}_0\ \mathrm{M}_2\ \mathrm{M}_3\ \mathrm{M}_7$$

$$= \Pi\ (\mathrm{M}_0,\ \mathrm{M}_2,\ \mathrm{M}_3,\ \mathrm{M}_7)$$

$$= \Pi\ \mathrm{M}(0, 2, 3, 7)$$

$$= \Pi_{x_1, x_2, x_3}(0, 2, 3, 7)$$

$$= (x_1 + x_2 + x_3)(x_1 + x_2' + x_3)(x_1 + x_2' + x_3')(x_1' + x_2' + x_3')$$

- A logic expression in the product-of-sums (PoS) form:
  - Consisting of sum (OR) terms that are the factors of a logical product
- Canonical PoS: each sum term is a maxterm

Let’s design a circuit!

Binary addition

3 + 2 = 5

$$
\begin{array}{r}
\scriptstyle{1\ \ \ \ \ } \text{ carry} \\
0\,0\,1\,1 \\
+\ 0\,0\,1\,0 \\
\hline
0\,1\,0\,1
\end{array}
$$

3 + 3 = 6

$$
\begin{array}{r}
\scriptstyle{1\ \ 1\ \ \ \ } \\
0\,0\,1\,1 \\
+\ 0\,0\,1\,1 \\
\hline
0\,1\,1\,0
\end{array}
$$

Input			Output	
A	B	Cin	Out	Cout
0	0	0	**0**	**0**
0	1	0	**1**	**0**
1	0	0	**1**	**0**
1	1	0	**0**	**1**
0	0	1	**1**	**0**
0	1	1	**0**	**1**
1	0	1	**0**	**1**
1	1	1	**1**	**1**

full adder — adder with a carry as an input

half adder — adder without a carry as an input

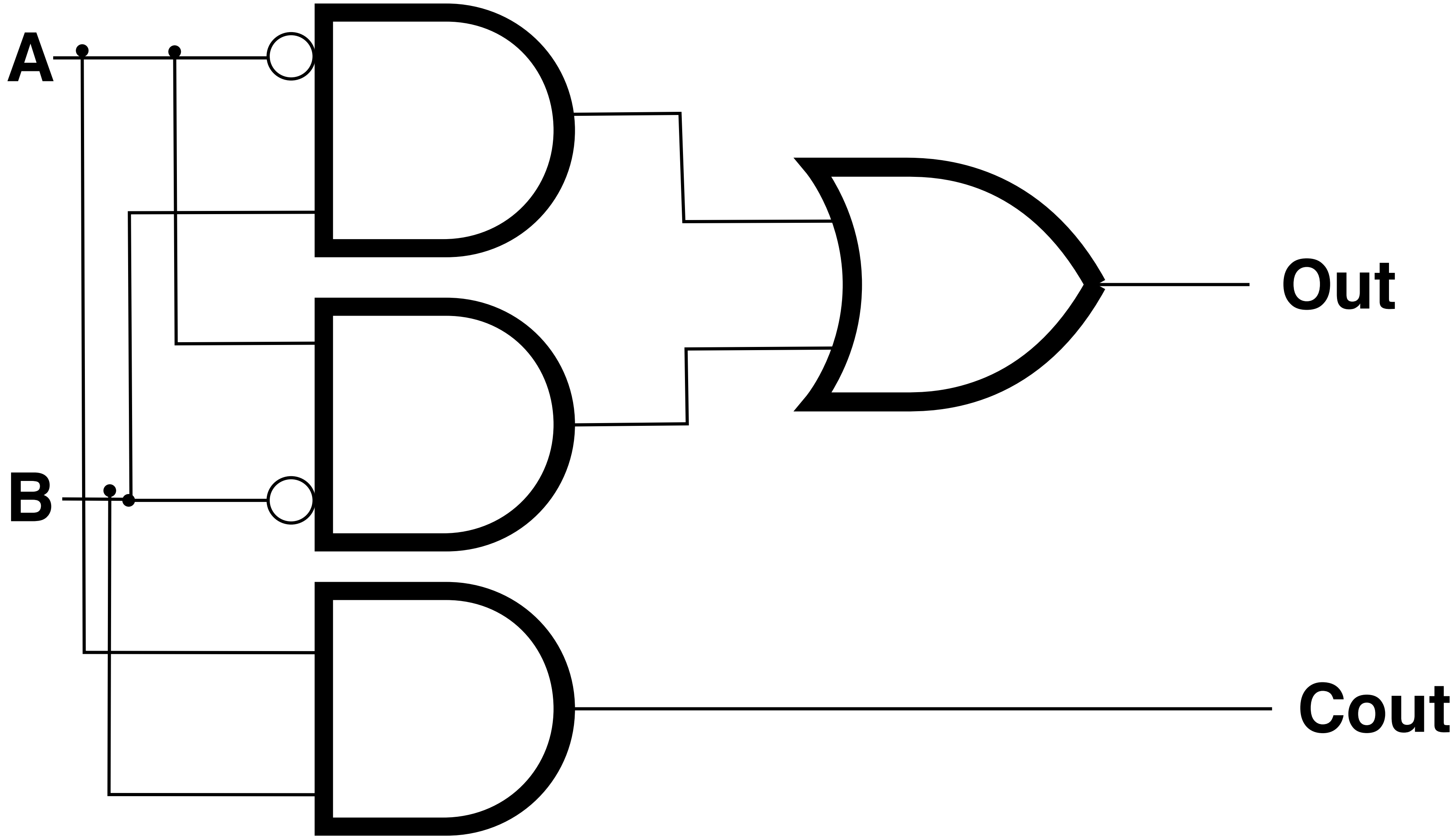
Input		Output	
A	B	Out	Cout
0	0	**0**	**0**
0	1	**1**	**0**
1	0	**1**	**0**
1	1	**0**	**1**

Half adder

Input		Output	
A	B	Out	Cout
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Out = A’B + AB’

Cout = AB



The sum-of-product form of the full adder

- How many of the following minterms are part of the sum-of-product form of the full adder in generating the output bit?
  ① A'B'Cin'
  ② A'BCin'
  ③ AB'Cin'
  ④ ABCin'
  ⑤ A'B'Cin
  ⑥ A'BCin
  ⑦ AB'Cin
  ⑧ ABCin
  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

2 Mins Poll

The sum-of-product form of the full adder

- How many of the following minterms are part of the sum-of-product form of the full adder in generating the output bit?
  1. A'B'Cin'
  2. A'BCin'
  3. AB'Cin'
  4. ABCin'
  5. A'B'Cin
  6. A'BCin
  7. AB'Cin
  8. ABCin

  A. 0
  B. 1
  C. 2
  D. 3
  E. 4

Input			Output	
A	B	Cin	Out	Cout
0	0	0	**0**	**0**
0	1	0	**1**	**0**
1	0	0	**1**	**0**
1	1	0	**0**	**1**
0	0	1	**1**	**0**
0	1	1	**0**	**1**
1	0	1	**0**	**1**
1	1	1	**1**	**1**

Out = A'BCin' + AB'Cin' + A'B'Cin + ABCin

Cout = ABCin' + A'BCin + AB'Cin + ABCin

sum-of-products/product-of-sums

- They can be used interchangeably
- Depends on if the truth table has more 0s or 1s in the result
- Neither forms give you the “optimized” equation. By optimized, we mean — minimize the number of operations

Can we simplify these functions?

Sum-of-minterms

- Different equations may represent the same function. Ex: y = a + b, and y = a + a'b, represent the same function. The sameness is not obvious, so a standard equation form is desirable.
- A **canonical form** of a Boolean equation is a standard equation form for a function.
- **Sum-of-minterms** form is a canonical form of a Boolean equation where the right-side expression is a **sum-of-products** with each product a unique minterm.
- A **minterm** is a product term having **exactly one literal** for **every** function **variable**.
- A **literal** is a variable appearance, in true or complemented form, in an expression, such as b, or b'.

Laws in Boolean Algebra

	OR	AND
Associative laws	$(a+b)+c=a+(b+c)$	$(a\cdot b)\cdot c=a\cdot(b\cdot c)$
Commutative laws	$a+b=b+a$	$a\cdot b=b\cdot a$
Distributive laws	$a+(b\cdot c)=(a+b)\cdot(a+c)$	$a\cdot(b+c)=a\cdot b+a\cdot c$
Identity laws	$a+0=a$	$a\cdot 1=a$
Complement laws	$a+a'=1$	$a\cdot a'=0$

Duality: We swap all operators between (+,.) and interchange all elements between (0,1).
For a theorem if the statement can be proven with the laws of Boolean algebra, then the duality of the statement is also true.

Some more tools

	OR	AND
DeMorgan's Theorem	$(a + b)' = a'b'$	$a'b' = (a + b)'$
Covering Theorem	$a(a+b) = a+ab = a$	$ab + ab' = (a+b)(a+b') = a$
Consensus Theorem	$ab+ac+b'c = ab+b'c$	$(a+b)(a+c)(b'+c) = (a+b)(b'+c)$
Uniting Theorem	$a\,(b + b') = a$	$(a+b)\cdot(a+b')=a$
Shannon's Expansion	$f(a,b,c) = a'b' + bc + ab'c$ $f(a,b,c) = a\,f(1, b, c) + a'\,f(0,b,c)$	

Applying Theorems

- Which of the following represents CB+BA+C'A?
  A. AB+AC'
  B. BC+AC'
  C. AB+BC
  D. AB+AC
  E. None of the above

	OR	AND
Associative laws	$(a+b)+c=a+(b+c)$	$(a\cdot b)\cdot c=a\cdot(b\cdot c)$
Commutative laws	$a+b=b+a$	$a\cdot b=b\cdot a$
Distributive laws	$a+(b\cdot c)=(a+b)\cdot(a+c)$	$a\cdot(b+c)=a\cdot b+a\cdot c$
Identity laws	$a+0=a$	$a\cdot 1=a$
Complement laws	$a+a'=1$	$a\cdot a'=0$
DeMorgan's Theorem	$(a+b)' = a'b'$	$a'b' = (a+b)'$
Covering Theorem	$a(a+b) = a+ab = a$	$ab + ab' = (a+b)(a+b') = a$
Consensus Theorem	$ab+ac+b'c = ab+b'c$	$(a+b)(a+c)(b'+c) = (a+b)(b'+c)$
Uniting Theorem	$a(b+b') = a$	$(a+b)\cdot(a+b')=a$
Shannon's Expansion	$f(a,b,c) = a'b' + bc + ab'c$ $f(a,b,c) = a\, f(1,b,c) + a'\, f(0,b,c)$	

Applying Theorems

2 Mins Poll

- Which of the following represents BC+BA+C'A?
  A. AB+AC'
  B. $\boxed{\text{BC+AC'}}$
  C. AB+BC
  D. AB+AC
  E. None of the above

Consensus Theorem

ab+ac+b'c = ab+b'c

CB + BA1 + C'A
= BC + BA (C'+C) + C'A
= BC + BAC' + BAC + C'A
= (1+A)BC + (B+1)AC'
= BC + AC'

How many "OR"s?

- For the truth table shown on the right, what's the minimum number of "OR" gates we need?
  A. 1
  B. 2
  C. 3
  D. 4
  E. 5

Input			Output
A	B	C	
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**0**
1	0	0	**1**
1	1	1	**0**

How many "OR"s?

- For the truth table shown on the right, what's the minimum number of "OR" gates we need?
   A. 1
   B. 2
   C. 3
   D. 4
   E. 5

$F(A, B, C) =$

$\boxed{A'B'C' + A'B'C} + \boxed{A'BC' + A'BC} + \boxed{AB'C' + ABC'}$

$= A'B'(C'+C) + A'B(C'+C) + AC'(B'+B)$

$= A'B' + A'B + AC'$

$= A' + AC' = A'(1+C')+AC'$ Distributive Laws

$= A' + A'C' + AC'$

$= A' + (A'+A)C'$

$= A' + C'$

Input			Output
A	B	C	
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**0**
1	1	0	**1**
1	1	1	**0**