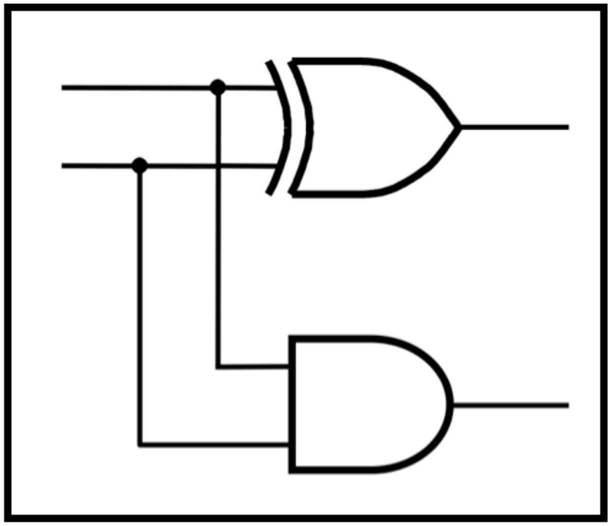


CprE 281: Digital Logic

Instructor: Alexander Stoytchev

http://www.ece.iastate.edu/~alexs/classes/

NAND and NOR Logic Networks

CprE 281: Digital Logic
Iowa State University, Ames, IA

Administrative Stuff

- **HW2 is due on Monday Aug 31 @ 4pm**

Administrative Stuff

- **HW3 is due on Monday Sep 7 @ 4pm**

- **Please write clearly on the first page the following three things:**

  - **Your First and Last Name**
  - **Your Student ID Number**
  - **Your Lab Section Letter**

- **Submit on Canvas as *one* PDF file.**

- **Please orient your pages such that the text can be read without the need to rotate the page.**

Quick Review

Four Basis Functions

x	y	f_{00}
0	0	1
0	1	0
1	0	0
1	1	0

x	y	f_{01}
0	0	0
0	1	1
1	0	0
1	1	0

x	y	f_{10}
0	0	0
0	1	0
1	0	1
1	1	0

x	y	f_{11}
0	0	0
0	1	0
1	0	0
1	1	1

$f_{00}(x, y)$ $f_{01}(x, y)$ $f_{10}(x, y)$ $f_{11}(x, y)$

Four Basis Functions

x	y	f_{00}
0	0	1
0	1	0
1	0	0
1	1	0

$f_{00}(x, y)$

x	y	f_{01}
0	0	0
0	1	1
1	0	0
1	1	0

$f_{01}(x, y)$

x	y	f_{10}
0	0	0
0	1	0
1	0	1
1	1	0

$f_{10}(x, y)$

x	y	f_{11}
0	0	0
0	1	0
1	0	0
1	1	1

$f_{11}(x, y)$

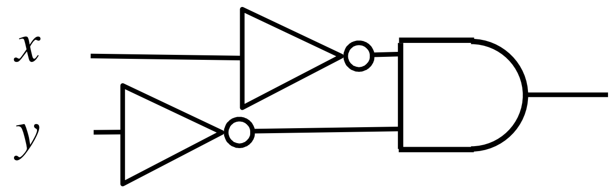
Four Basis Functions

x	y		$f_{00}(x, y)$	$f_{01}(x, y)$	$f_{10}(x, y)$	$f_{11}(x, y)$
0	0		1	0	0	0
0	1		0	1	0	0
1	0		0	0	1	0
1	1		0	0	0	1

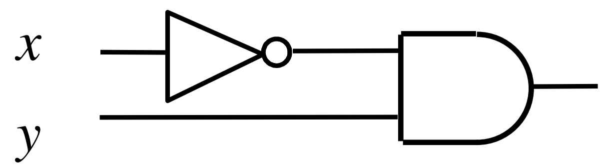
Four Basis Functions

x	y		$\bar{x}\,\bar{y}$	$\bar{x}\,y$	$x\,\bar{y}$	$x\,y$
0	0		1	0	0	0
0	1		0	1	0	0
1	0		0	0	1	0
1	1		0	0	0	1

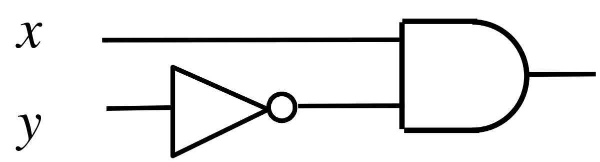
Circuits for the four basis functions



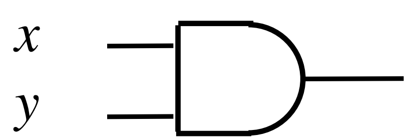
$$f_{00}(x, y) = \overline{x}\ \overline{y}$$



$$f_{01}(x, y) = \overline{x}\ y$$



$$f_{10}(x, y) = x\ \overline{y}$$



$$f_{11}(x, y) = x\ y$$

Four Basis Functions

x	y	f_{00}
0	0	1
0	1	0
1	0	0
1	1	0

x	y	f_{01}
0	0	0
0	1	1
1	0	0
1	1	0

x	y	f_{10}
0	0	0
0	1	0
1	0	1
1	1	0

x	y	f_{11}
0	0	0
0	1	0
1	0	0
1	1	1

$f_{00}(x, y) = \overline{x}\,\overline{y}$ $f_{01}(x, y) = \overline{x}\,y$ $f_{10}(x, y) = x\,\overline{y}$ $f_{11}(x, y) = x\,y$

Four Basis Functions

x	y	f_{00}
0	0	1
0	1	0
1	0	0
1	1	0

$f_{00}(x, y) = \overline{x}\,\overline{y}$

m_0

x	y	f_{01}
0	0	0
0	1	1
1	0	0
1	1	0

$f_{01}(x, y) = \overline{x}\,y$

m_1

x	y	f_{10}
0	0	0
0	1	0
1	0	1
1	1	0

$f_{10}(x, y) = x\,\overline{y}$

m_2

x	y	f_{11}
0	0	0
0	1	0
1	0	0
1	1	1

$f_{11}(x, y) = x\,y$

m_3

Minterms and Maxterms

Row number	x_1	x_2	Minterm	Maxterm
0	0	0	$m_0 = \overline{x}_1\overline{x}_2$	$M_0 = x_1 + x_2$
1	0	1	$m_1 = \overline{x}_1 x_2$	$M_1 = x_1 + \overline{x}_2$
2	1	0	$m_2 = x_1\overline{x}_2$	$M_2 = \overline{x}_1 + x_2$
3	1	1	$m_3 = x_1 x_2$	$M_3 = \overline{x}_1 + \overline{x}_2$

Minterms and Maxterms

Row number	x_1	x_2	Minterm	Maxterm
0	0	0	$m_0 = \overline{x}_1\overline{x}_2$	$M_0 = x_1 + x_2$
1	0	1	$m_1 = \overline{x}_1 x_2$	$M_1 = x_1 + \overline{x}_2$
2	1	0	$m_2 = x_1\overline{x}_2$	$M_2 = \overline{x}_1 + x_2$
3	1	1	$m_3 = x_1 x_2$	$M_3 = \overline{x}_1 + \overline{x}_2$

Use these for Sum-of-Products Minimization (1's of the function)

Use these for Product-of-Sums Minimization (0's of the function)

Sum-of-Products Form

(uses the ones of the function)

Sum-of-Products Form

Row number	x_1	x_2	Minterm	$f(x_1, x_2)$
0	0	0	$m_0 = \overline{x}_1\overline{x}_2$	1
1	0	1	$m_1 = \overline{x}_1 x_2$	1
2	1	0	$m_2 = x_1\overline{x}_2$	0
3	1	1	$m_3 = x_1 x_2$	1

Sum-of-Products Form

Row number	x_1	x_2	Minterm	$f(x_1, x_2)$
0	0	0	$m_0 = \overline{x}_1\overline{x}_2$	1
1	0	1	$m_1 = \overline{x}_1 x_2$	1
2	1	0	$m_2 = x_1\overline{x}_2$	0
3	1	1	$m_3 = x_1 x_2$	1

Sum-of-Products Form

Row number	x_1	x_2	Minterm	$f(x_1, x_2)$
0	0	0	$m_0 = \overline{x}_1\overline{x}_2$	1
1	0	1	$m_1 = \overline{x}_1 x_2$	1
2	1	0	$m_2 = x_1\overline{x}_2$	0
3	1	1	$m_3 = x_1 x_2$	1

$$\begin{aligned} f &= m_0 \cdot 1 + m_1 \cdot 1 + m_2 \cdot 0 + m_3 \cdot 1 \\ &= m_0 + m_1 + m_3 \\ &= \overline{x}_1\overline{x}_2 + \overline{x}_1 x_2 + x_1 x_2 \end{aligned}$$

Product-of-Sums Form

(uses the zeros of the function)

Product-of-Sums Form
(for this logic function)

Row number	x_1	x_2	Maxterm	$f(x_1, x_2)$
0	0	0	$M_0 = x_1 + x_2$	0
1	0	1	$M_1 = x_1 + \overline{x_2}$	1
2	1	0	$M_2 = \overline{x_1} + x_2$	0
3	1	1	$M_3 = \overline{x_1} + \overline{x_2}$	1

Product-of-Sums Form
(for this logic function)

Row number	x_1	x_2	Maxterm	$f(x_1, x_2)$
0	0	0	$M_0 = x_1 + x_2$	0
1	0	1	$M_1 = x_1 + \overline{x_2}$	1
2	1	0	$M_2 = \overline{x_1} + x_2$	0
3	1	1	$M_3 = \overline{x_1} + \overline{x_2}$	1

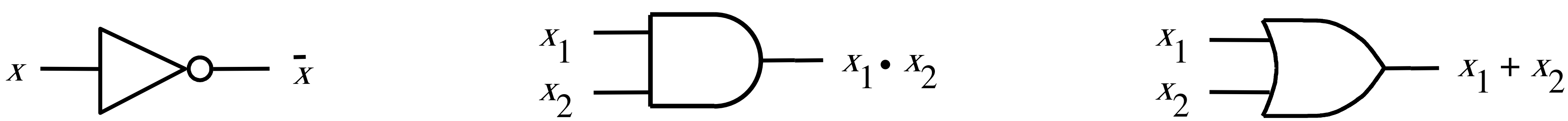
Product-of-Sums Form
(for this logic function)

Row number	x_1	x_2	Maxterm	$f(x_1, x_2)$
0	0	0	$M_0 = x_1 + x_2$	0
1	0	1	$M_1 = x_1 + \overline{x}_2$	1
2	1	0	$M_2 = \overline{x}_1 + x_2$	0
3	1	1	$M_3 = \overline{x}_1 + \overline{x}_2$	1

$$f(x_1, x_2) = M_0 \bullet M_2 = (x_1 + x_2) \bullet (\overline{x}_1 + x_2)$$

Two New Logic Gates

The Three Basic Logic Gates



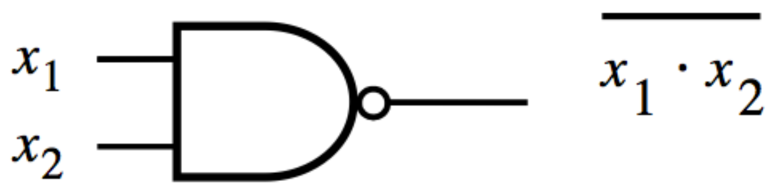
NOT gate

AND gate

OR gate

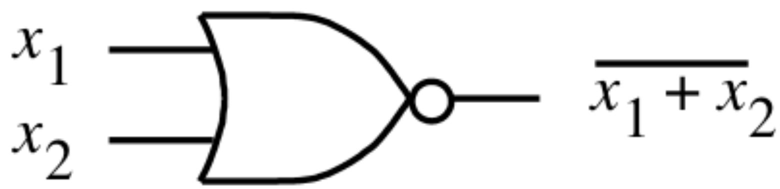
[Figure 2.8 from the textbook]

NAND Gate



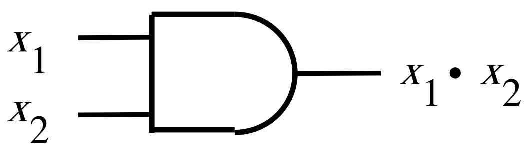
x_1	x_2	f
0	0	1
0	1	1
1	0	1
1	1	0

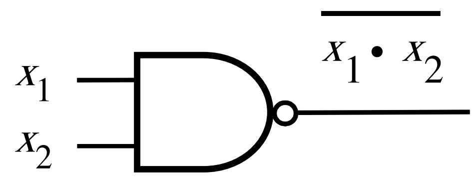
NOR Gate



x_1	x_2	f
0	0	1
0	1	0
1	0	0
1	1	0

AND vs NAND

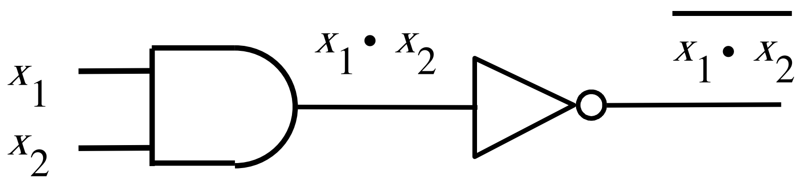


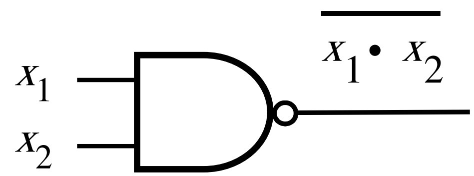


x_1	x_2	f
0	0	0
0	1	0
1	0	0
1	1	1

x_1	x_2	f
0	0	1
0	1	1
1	0	1
1	1	0

AND followed by NOT = NAND

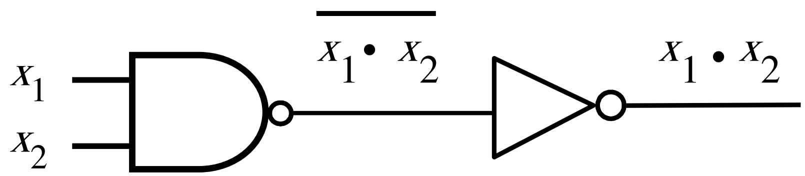


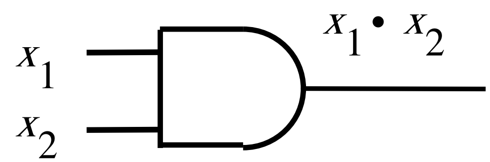


x_1	x_2	f	f
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

x_1	x_2	f
0	0	1
0	1	1
1	0	1
1	1	0

NAND followed by NOT = AND

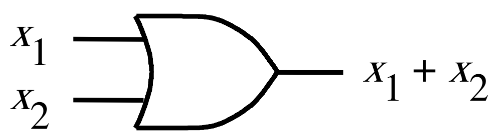


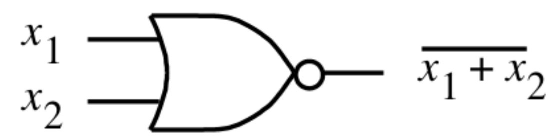


x_1	x_2	f	f
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

x_1	x_2	f
0	0	0
0	1	0
1	0	0
1	1	1

OR vs NOR

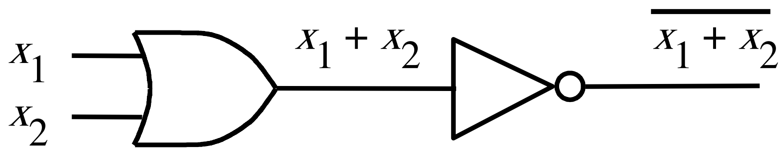


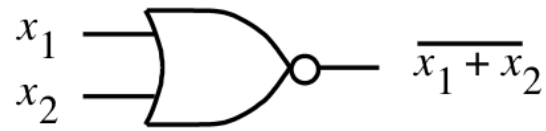


x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	1

x_1	x_2	f
0	0	1
0	1	0
1	0	0
1	1	0

OR followed by NOT = NOR



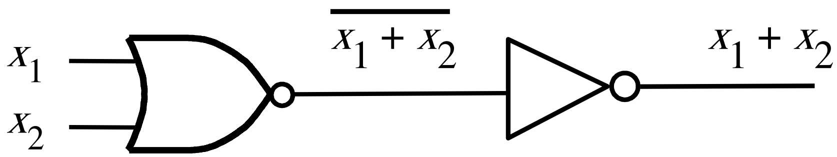


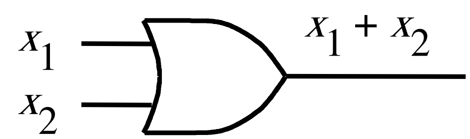
x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	1

f
1
0
0
0

x_1	x_2	f
0	0	1
0	1	0
1	0	0
1	1	0

NOR followed by NOT = OR





x_1	x_2	f	f
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	1

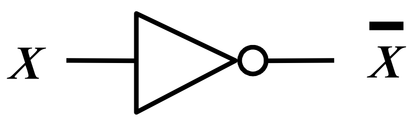
Why do we need two more gates?

Why do we need two more gates?

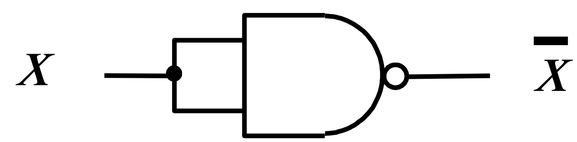
They can be implemented with fewer transistors.

They are simpler to implement, but are they also useful?

Building a NOT Gate with NAND

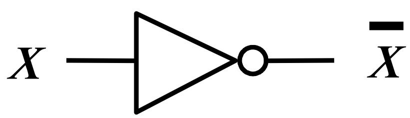


x	$\bar{x}$
0	1
1	0

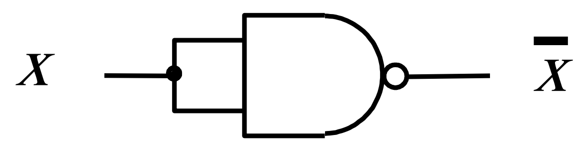


x	x	f
0	0	1
0	1	1
1	0	1
1	1	0

Building a NOT Gate with NAND



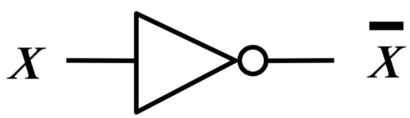
x	$\overline{x}$
0	1
1	0



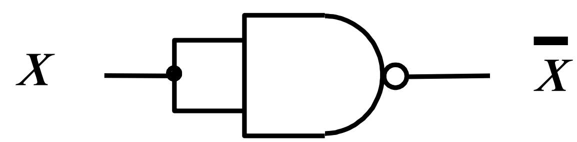
x	x	f
0	0	1
[redacted]		
1	1	0

impossible combinations

Building a NOT Gate with NAND



x	$\overline{x}$
0	1
1	0



x	x	f
0	0	1
[redacted]		
1	1	0

impossible combinations

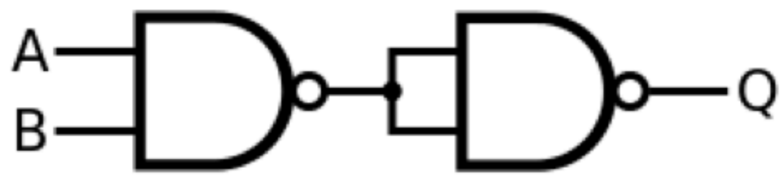
Thus, the two truth tables are equal!

Building an AND gate with NAND gates

Desired AND Gate

Q = **A** AND **B**

NAND Construction



= NOT(NOT(**A** AND **B**))

Truth Table

Input A	Input B	Output Q
0	0	0
0	1	0
1	0	0
1	1	1

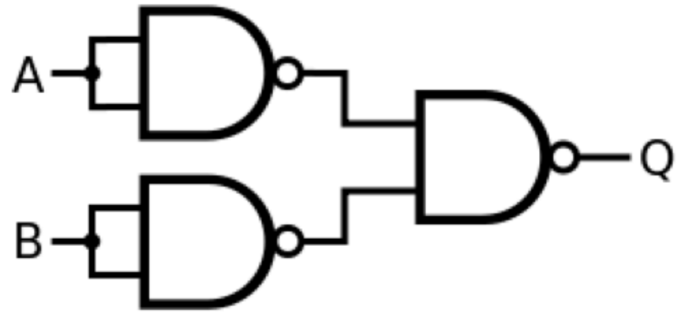
[http://en.wikipedia.org/wiki/NAND_logic]

Building an OR gate with NAND gates

Desired OR Gate

Q = **A** OR **B**

NAND Construction



= NOT[NOT(**A** AND **A**) AND NOT(**B** AND **B**)]

Truth Table

Input A	Input B		Output Q
0	0		0
0	1		1
1	0		1
1	1		1

[http://en.wikipedia.org/wiki/NAND_logic]

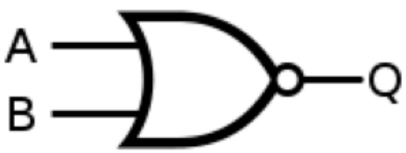
Implications

Implications

Any Boolean function can be implemented with only NAND gates!

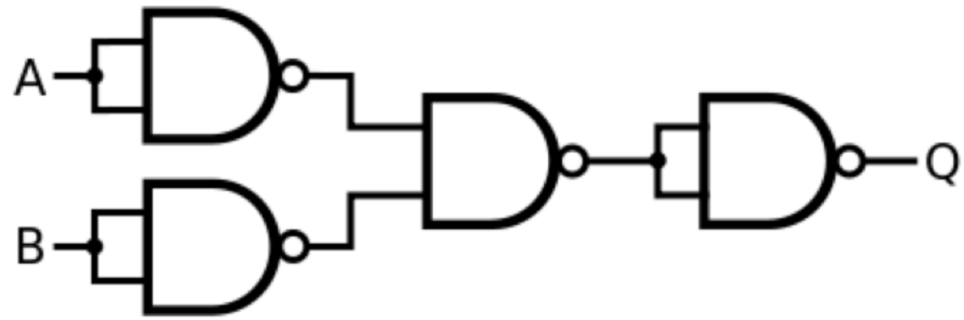
NOR gate with NAND gates

Desired NOR Gate



Q = NOT(**A** OR **B**)

NAND Construction



= NOT{ NOT[NOT(**A** AND **A**) AND NOT(**B** AND **B**)]}

Truth Table

Input A	Input B		Output Q
0	0		1
0	1		0
1	0		0
1	1		0

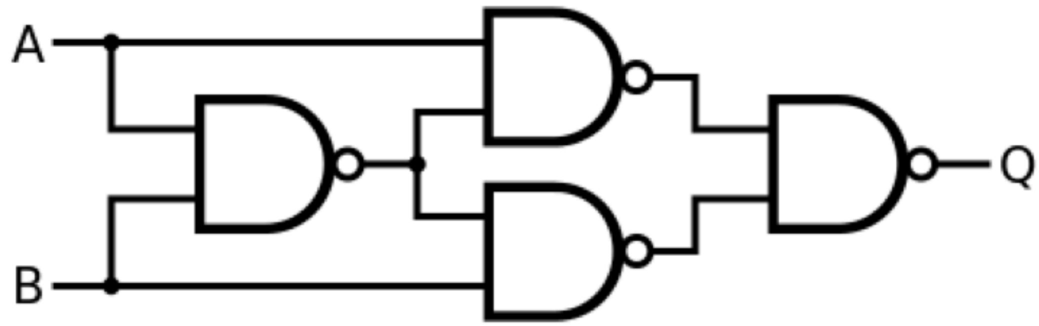
[http://en.wikipedia.org/wiki/NAND_logic]

XOR gate with NAND gates

Desired XOR Gate

Q = **A** XOR **B**

NAND Construction



= NOT[NOT{**A** AND NOT(**A** AND **B**)} AND NOT{**B** AND NOT(**A** AND **B**)}]

Truth Table

Input A	Input B		Output Q
0	0		0
0	1		1
1	0		1
1	1		0

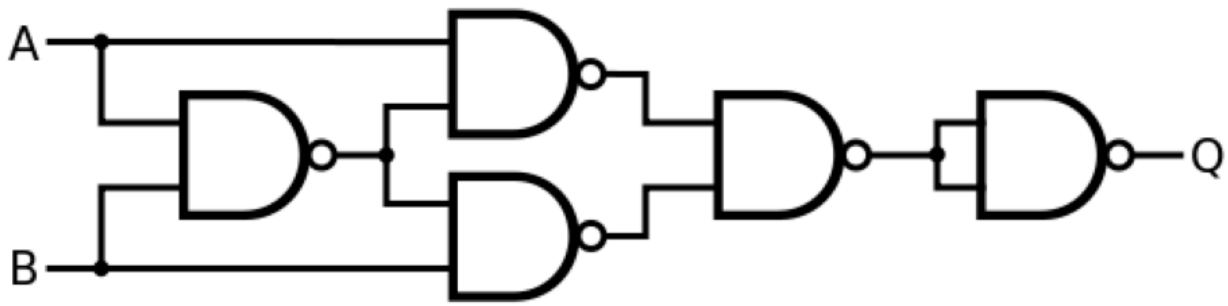
[http://en.wikipedia.org/wiki/NAND_logic]

XNOR gate with NAND gates

Desired XNOR Gate

Q = NOT(**A** XOR **B**)

NAND Construction



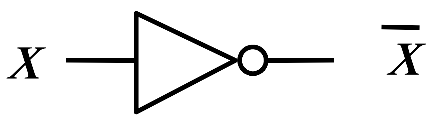
= NOT[NOT[NOT{**A** AND NOT(**A** AND **B**)} AND NOT{**B** AND NOT(**A** AND **B**)}]]

Truth Table

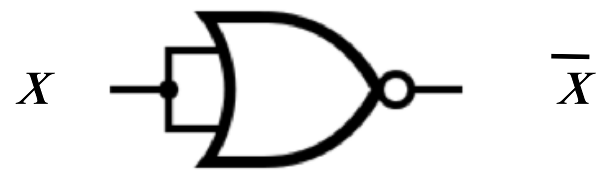
Input A	Input B		Output Q
0	0		1
0	1		0
1	0		0
1	1		1

[http://en.wikipedia.org/wiki/NAND_logic]

Building a NOT Gate with NOR

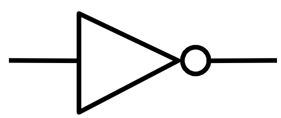


x	$\overline{x}$
0	1
1	0



x	x	f
0	0	1
0	1	0
1	0	0
1	1	0

Building a NOT Gate with NOR

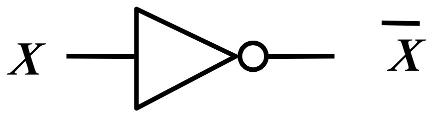


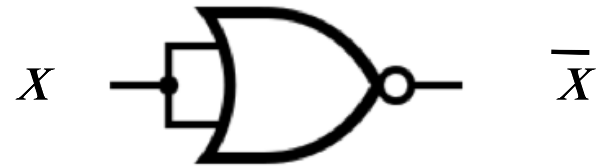
x	$\bar{x}$
0	1
1	0

x	x	f
0	0	1
[redacted]		
1	1	0

impossible combinations

Building a NOT Gate with NOR





x	$\overline{x}$
0	1
1	0

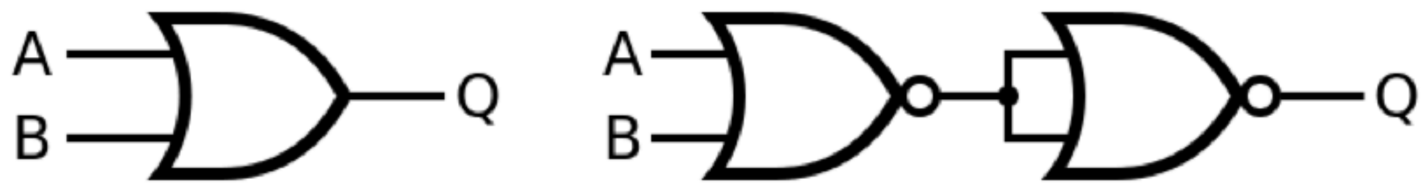
x	x	f
0	0	1
[redacted]		
1	1	0

impossible combinations

Thus, the two truth tables are equal!

Building an OR gate with NOR gates

Desired Gate **NOR Construction**



Truth Table

Input A	Input B	Output Q
0	0	0
0	1	1
1	0	1
1	1	1

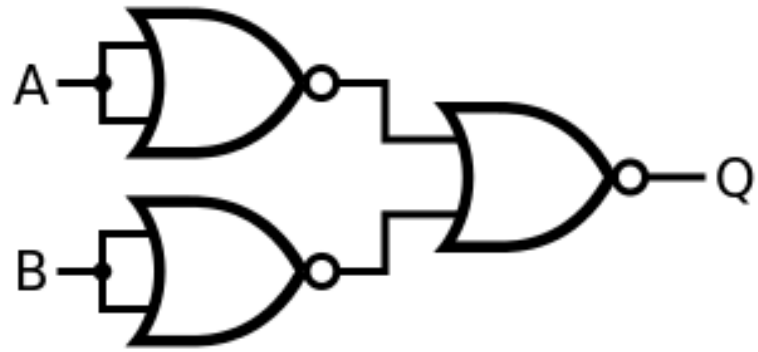
[http://en.wikipedia.org/wiki/NOR_logic]

Let’s build an AND gate with NOR gates

Let’s build an AND gate with NOR gates

Desired Gate

NOR Construction



Truth Table

Input A	Input B		Output Q
0	0		0
0	1		0
1	0		0
1	1		1

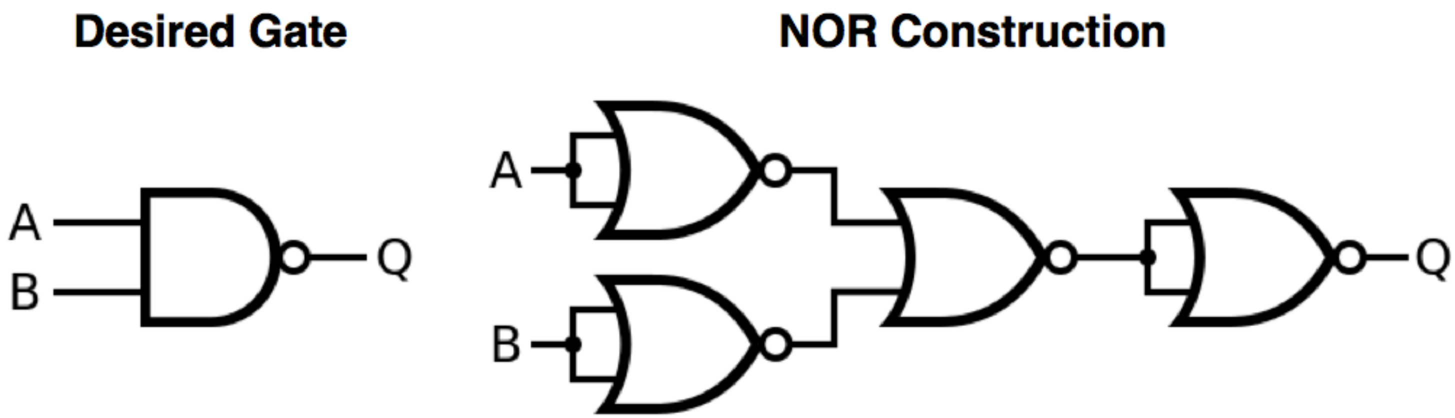
[http://en.wikipedia.org/wiki/NOR_logic]

Implications

Implications

Any Boolean function can be implemented with only NOR gates!

NAND gate with NOR gates

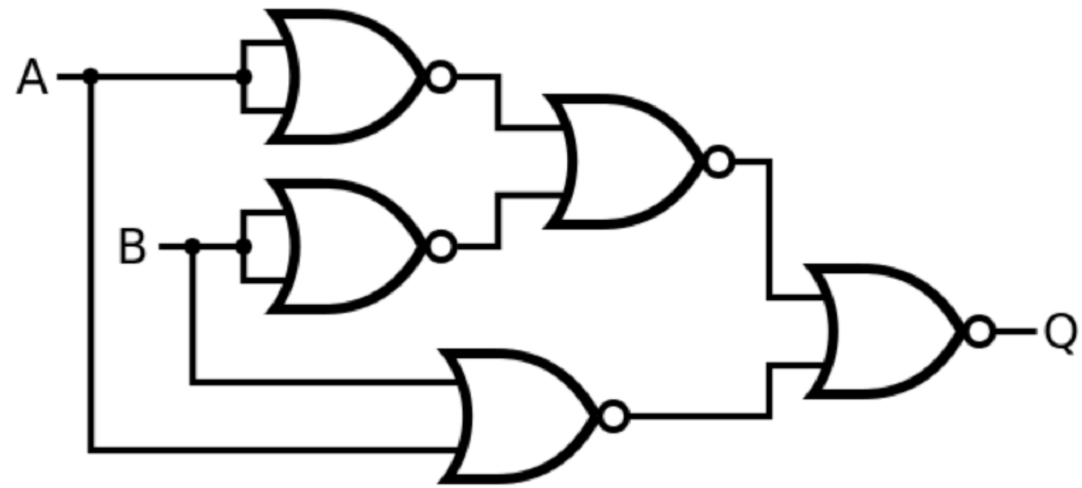


Truth Table

Input A	Input B		Output Q
0	0		1
0	1		1
1	0		1
1	1		0

[http://en.wikipedia.org/wiki/NOR_logic]

XOR gate with NOR gates



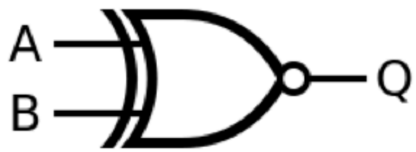
Truth Table

Input A	Input B		Output Q
0	0		0
0	1		1
1	0		1
1	1		0

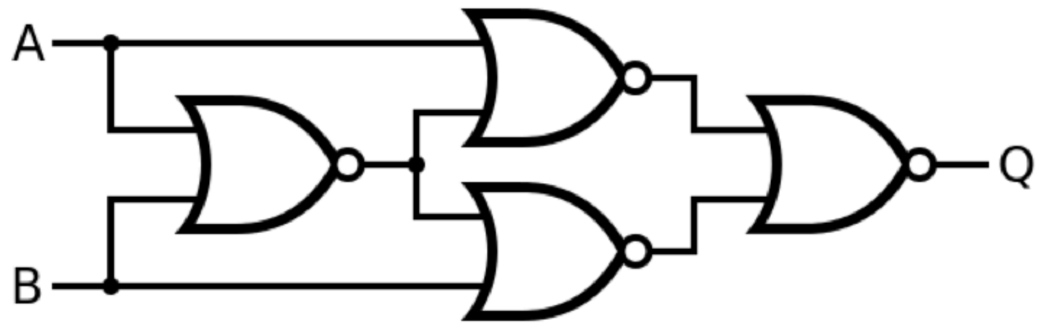
[http://en.wikipedia.org/wiki/NOR_logic]

XNOR gate with NOR gates

Desired XNOR Gate



NOR Construction

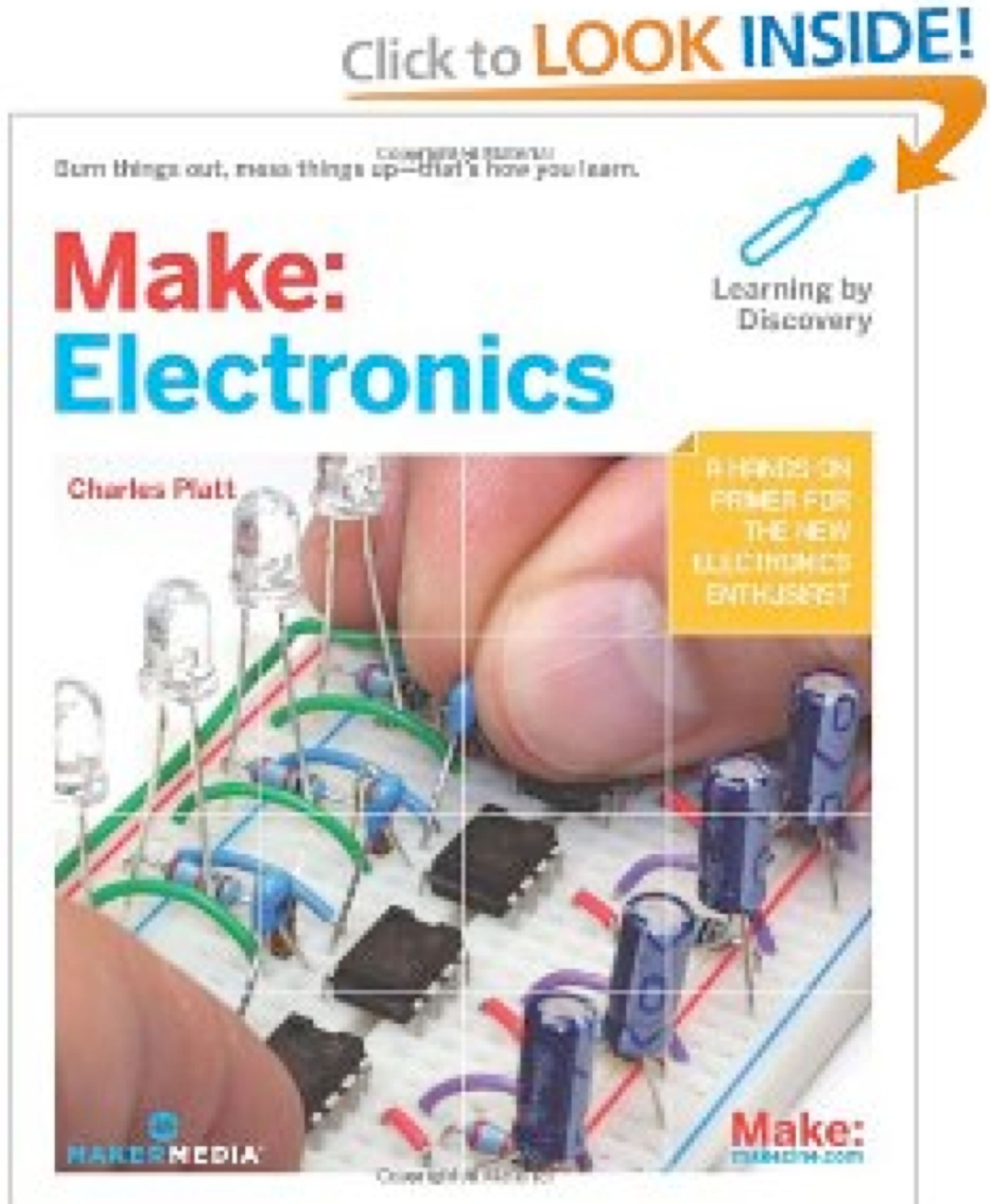


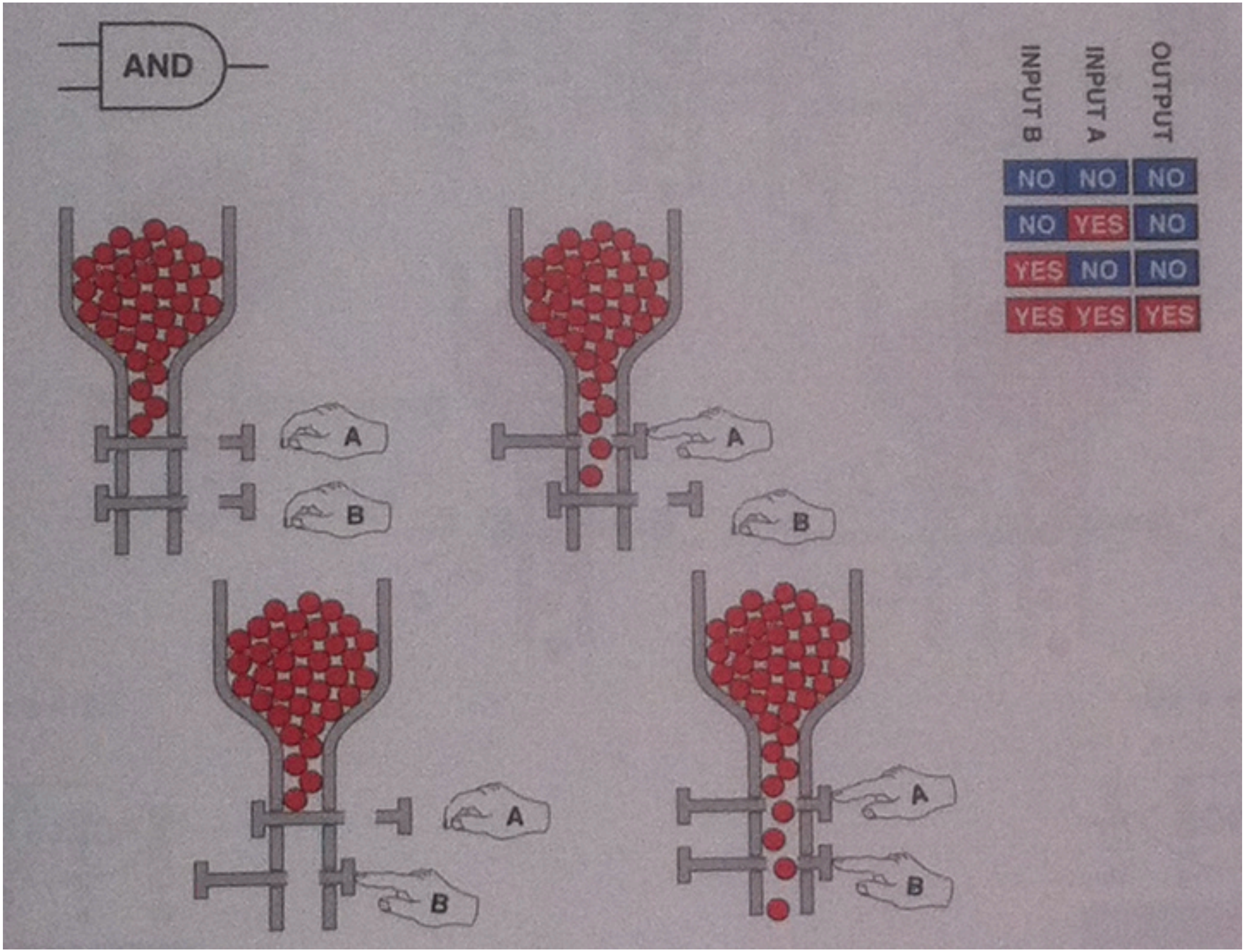
Truth Table

Input A	Input B		Output Q
0	0		1
0	1		0
1	0		0
1	1		1

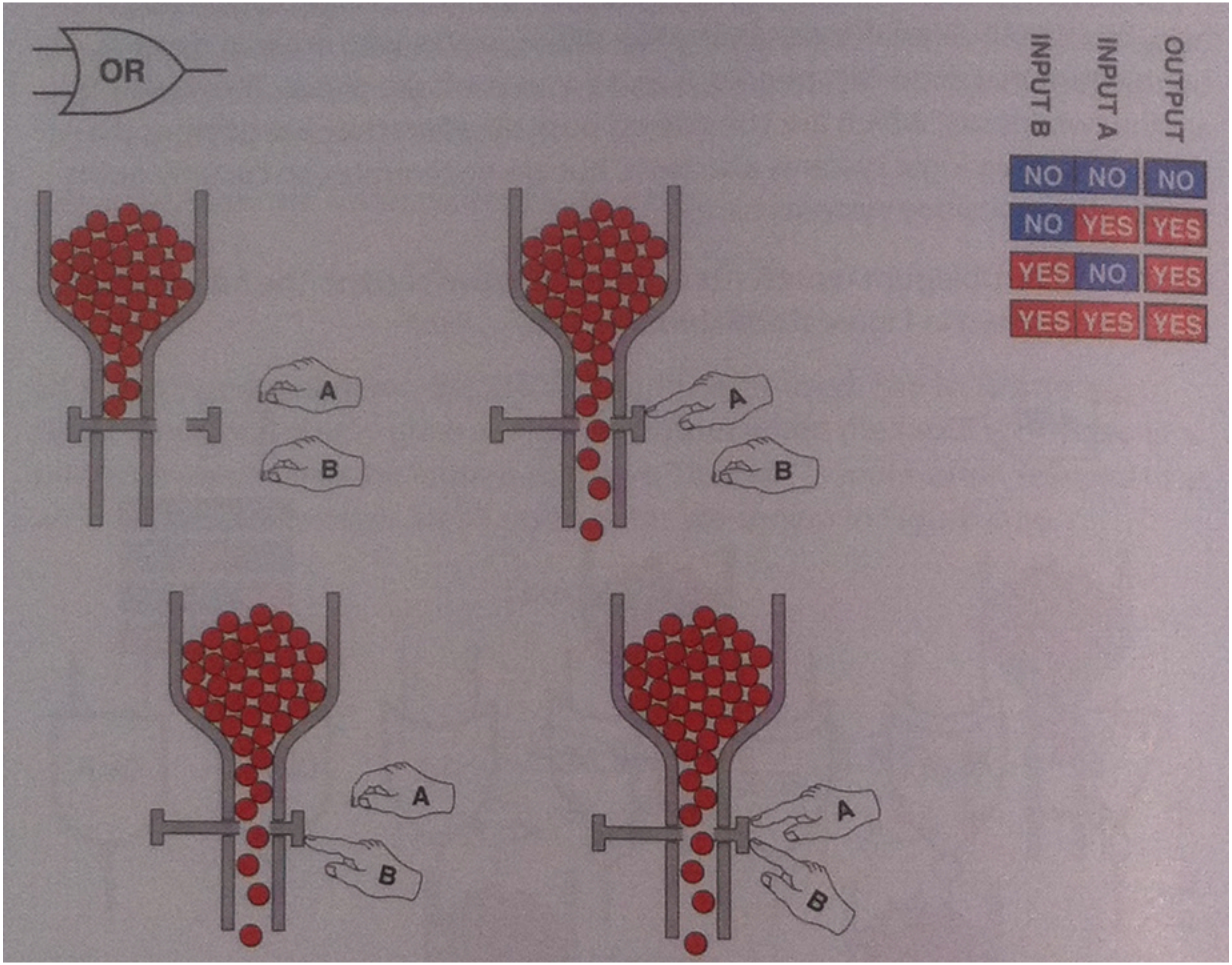
[http://en.wikipedia.org/wiki/NOR_logic]

The following examples came from this book

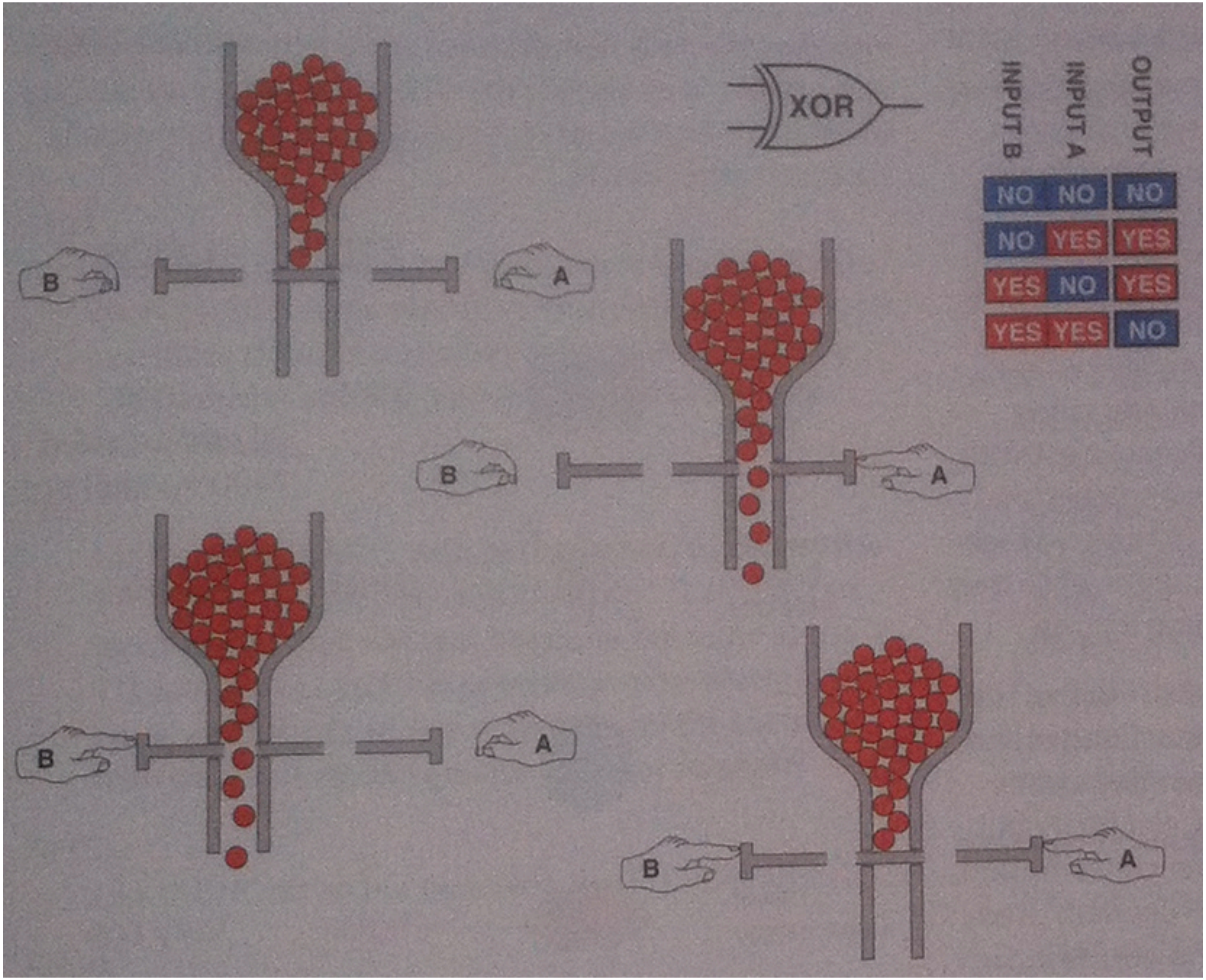



AND
INPUT B
INPUT A
OUTPUT
NO NO NO
NO YES NO
YES NO NO
YES YES YES
A
B
A
B
A
B
A
B

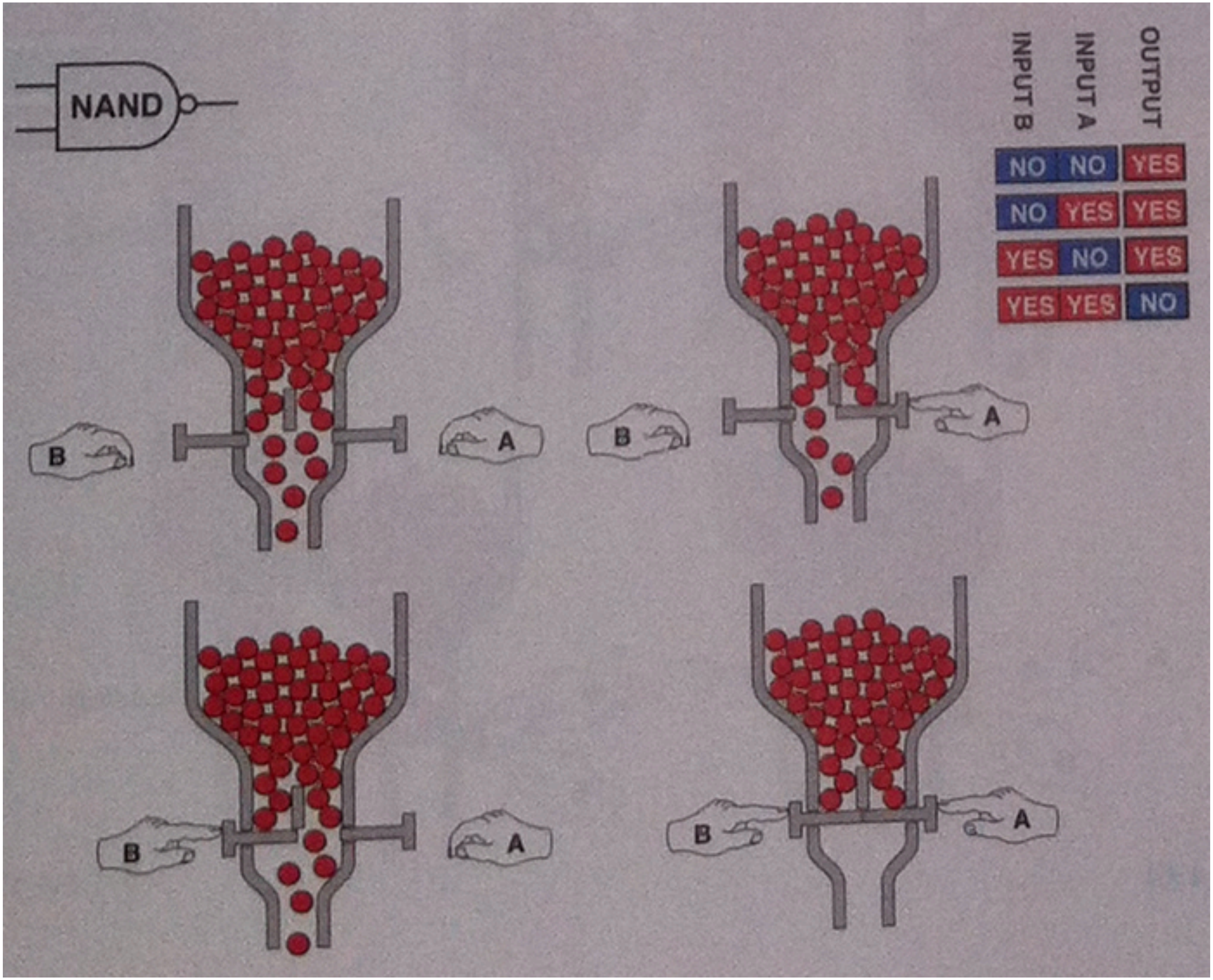
[Platt 2009]


OR
INPUT B
INPUT A
OUTPUT
NO NO NO
NO YES YES
YES NO YES
YES YES YES
A
B

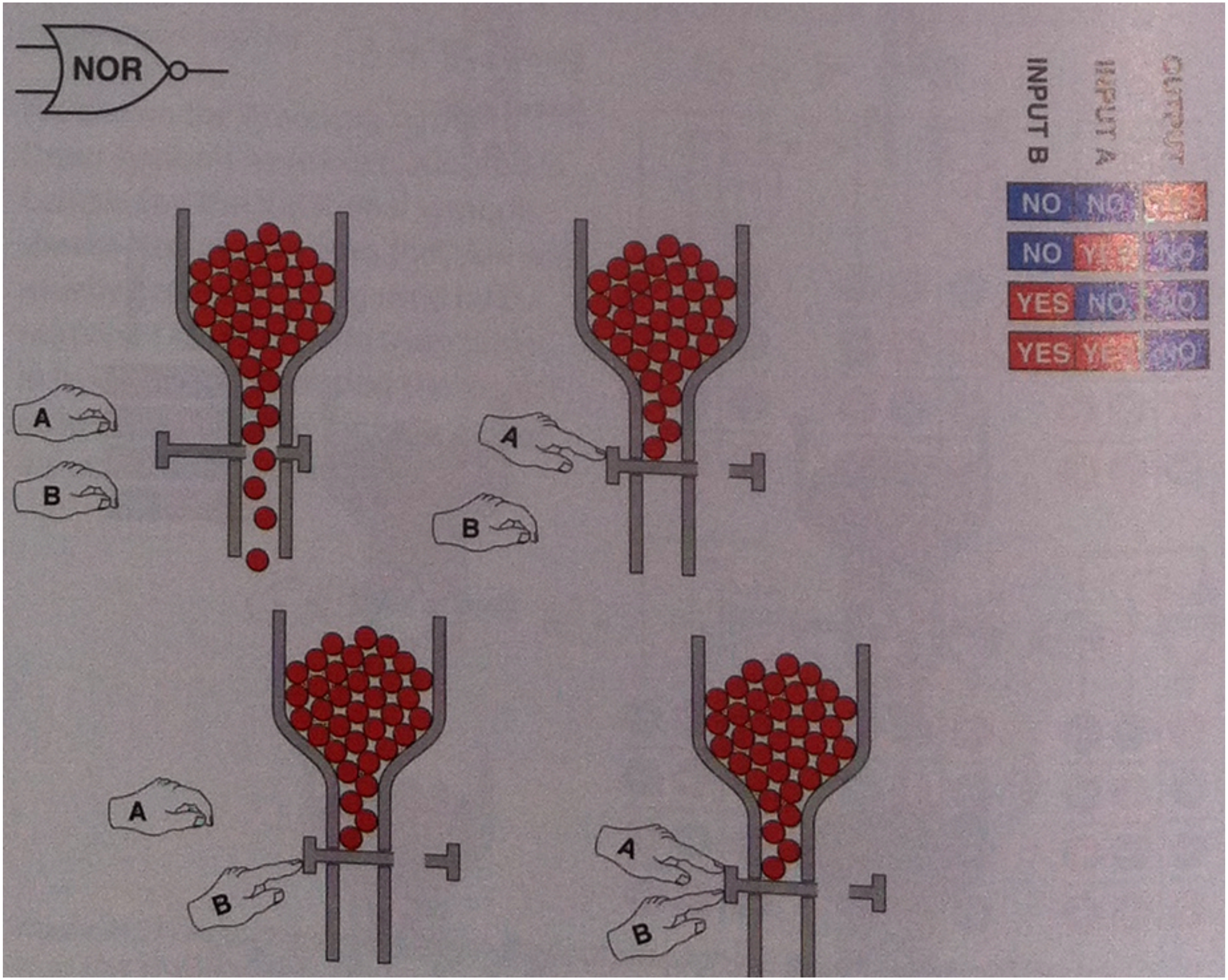
[Platt 2009]


XOR
INPUT B
INPUT A
OUTPUT
NO NO NO
NO YES YES
YES NO YES
YES YES NO
B
A
B
A
B
A
B
A

[Platt 2009]

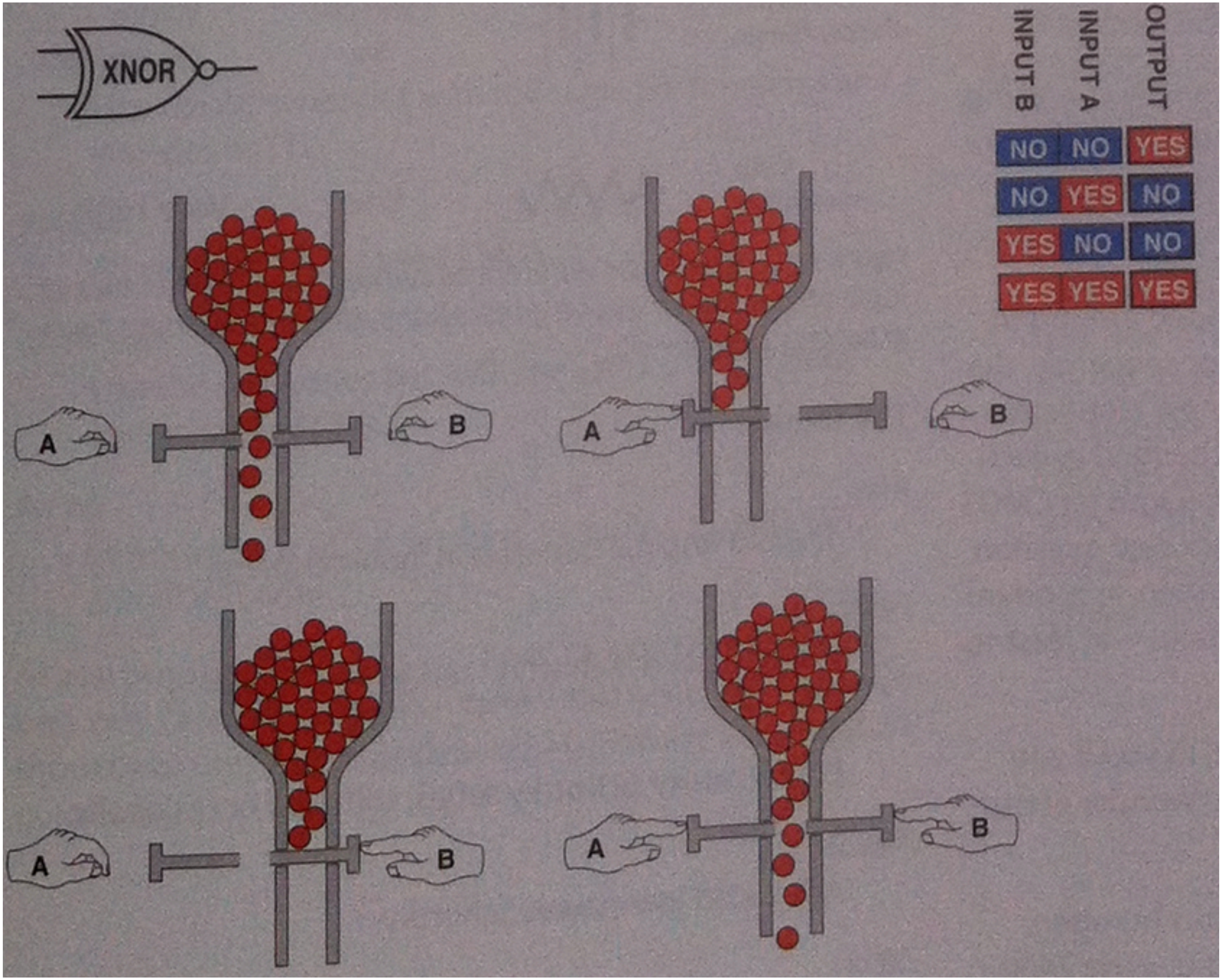

NAND
INPUT B
INPUT A
OUTPUT
NO NO YES
NO YES YES
YES NO YES
YES YES NO
B
A
B
A
B
A
B
A

[Platt 2009]



NOR
INPUT B
INPUT A
OUTPUT
NO
NO
NO
YES
YES
NO
YES
YES
NO
NO
A
B
A
B
A
B
A
B

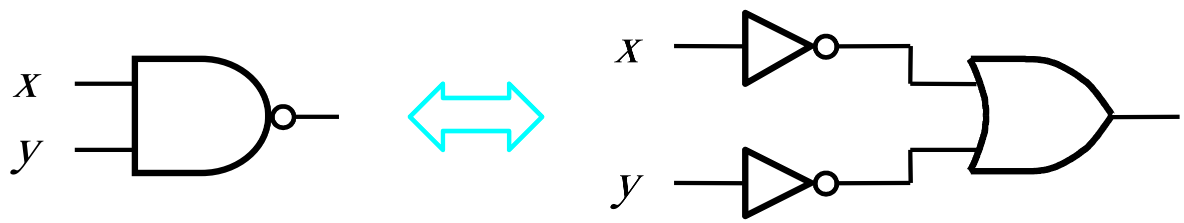
[Platt 2009]


XNOR
INPUT B
INPUT A
OUTPUT
NO NO YES
NO YES NO
YES NO NO
YES YES YES
A
B

[Platt 2009]

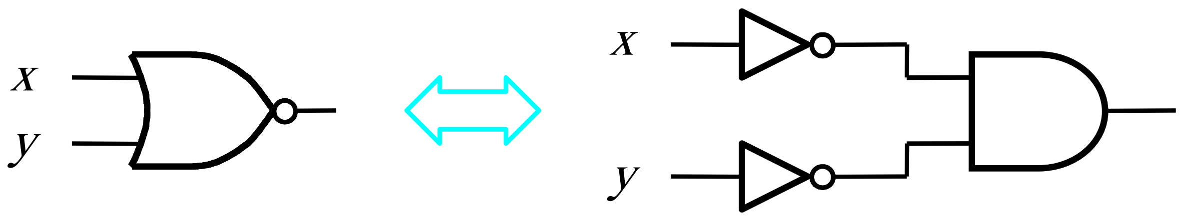
DeMorgan’s Theorem Revisited

DeMorgan’s theorem
(in terms of logic gates)



$$\overline{x \bullet y} = \overline{x} + \overline{y}$$

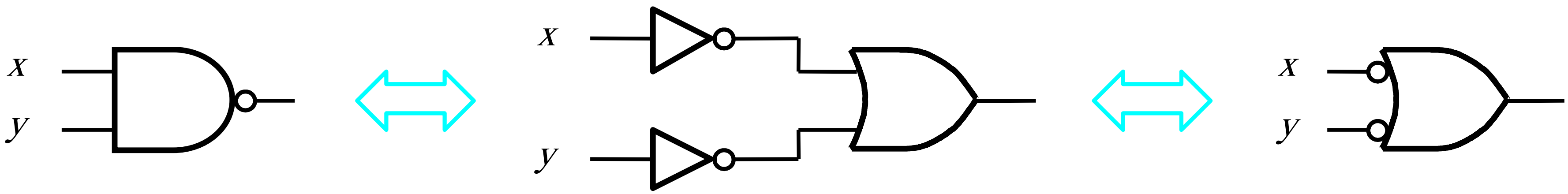
The other DeMorgan's theorem (in terms of logic gates)



$$\overline{x + y} = \overline{x} \bullet \overline{y}$$

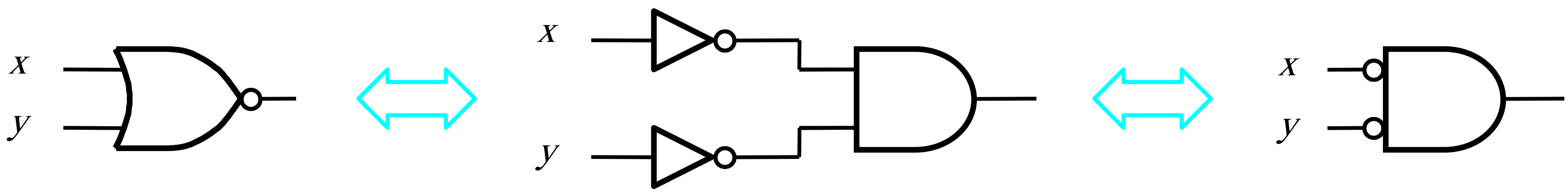
Shortcut Notation

DeMorgan's theorem in terms of logic gates



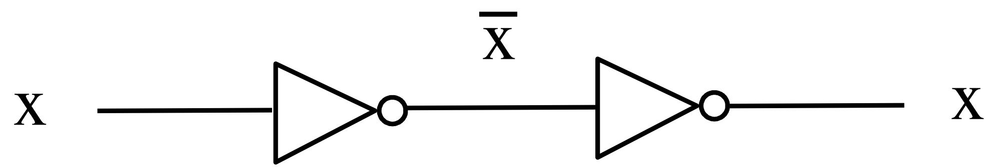
(Theorem 15.a) $\overline{x \bullet y} = \overline{x} + \overline{y}$

DeMorgan's theorem in terms of logic gates

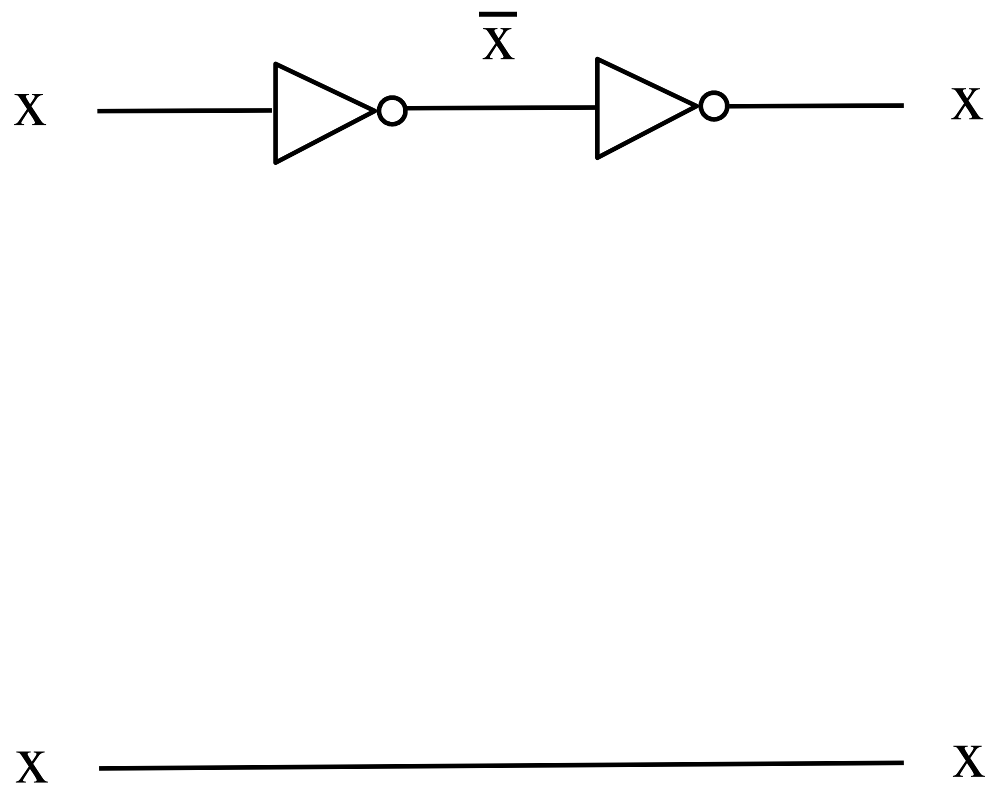


(Theorem 15.b) $\overline{x + y} = \overline{x}\,\overline{y}$

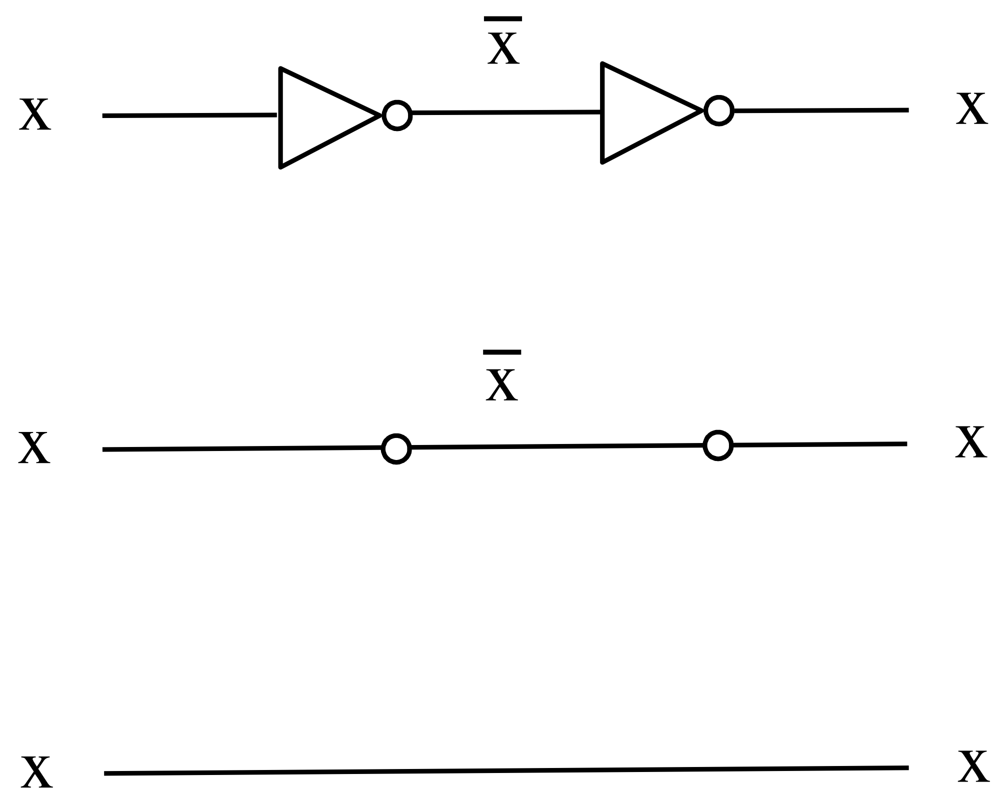
Two NOTs in a row



Two NOTs in a row

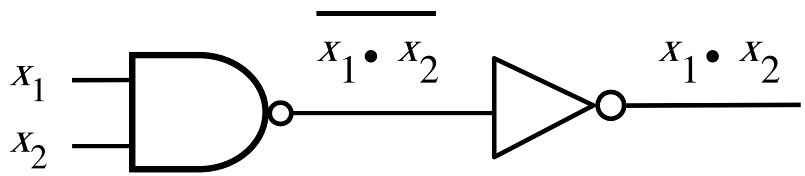


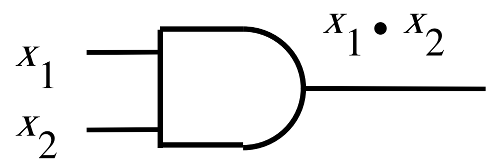
Two NOTs in a row



NAND-NAND Implementation of Sum-of-Products Expressions

NAND followed by NOT = AND





x_1	x_2	f
0	0	1
0	1	1
1	0	1
1	1	0

f
0
0
0
1

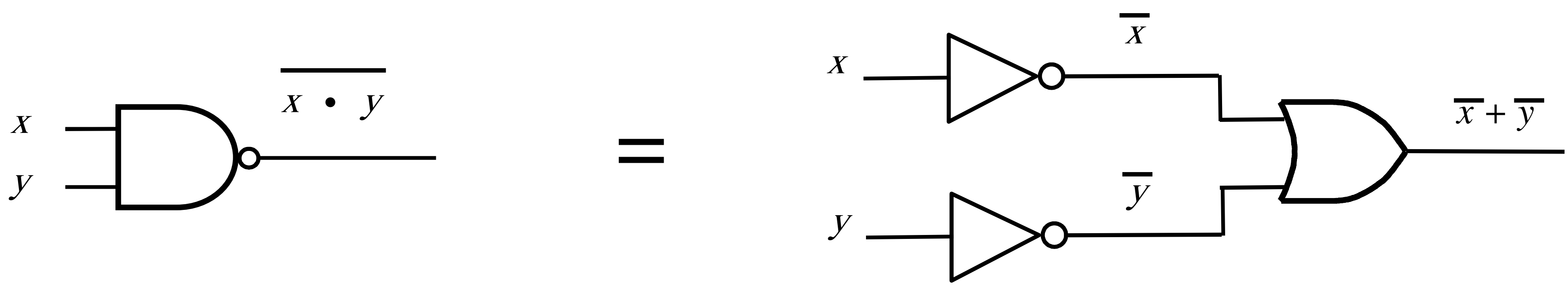
x_1	x_2	f
0	0	0
0	1	0
1	0	0
1	1	1

DeMorgan's Theorem

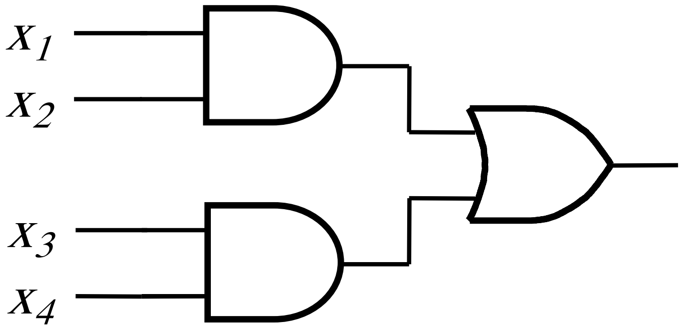
15a. $\overline{x \bullet y} = \overline{x} + \overline{y}$

DeMorgan’s Theorem

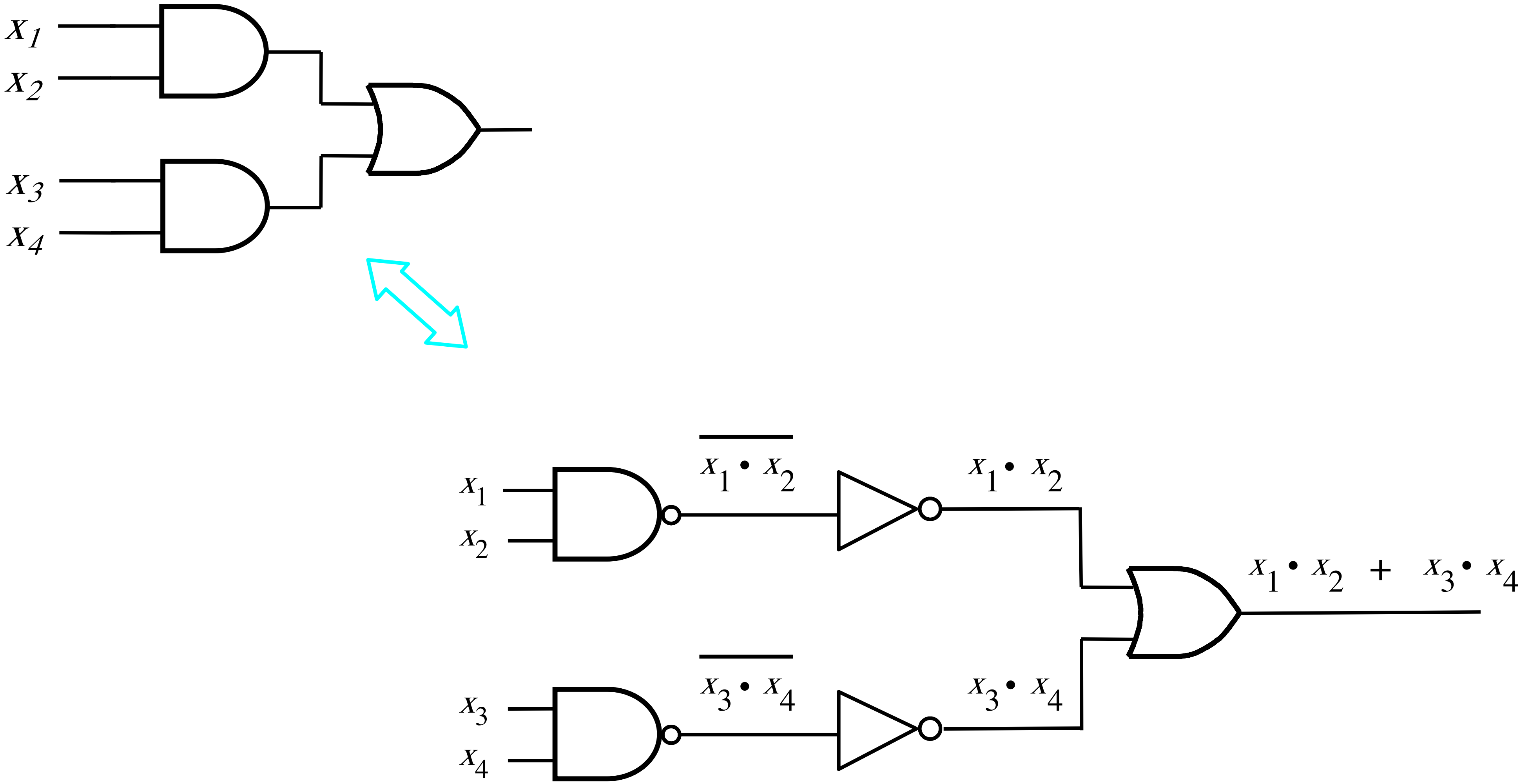
15a. $\overline{x \cdot y} = \overline{x} + \overline{y}$



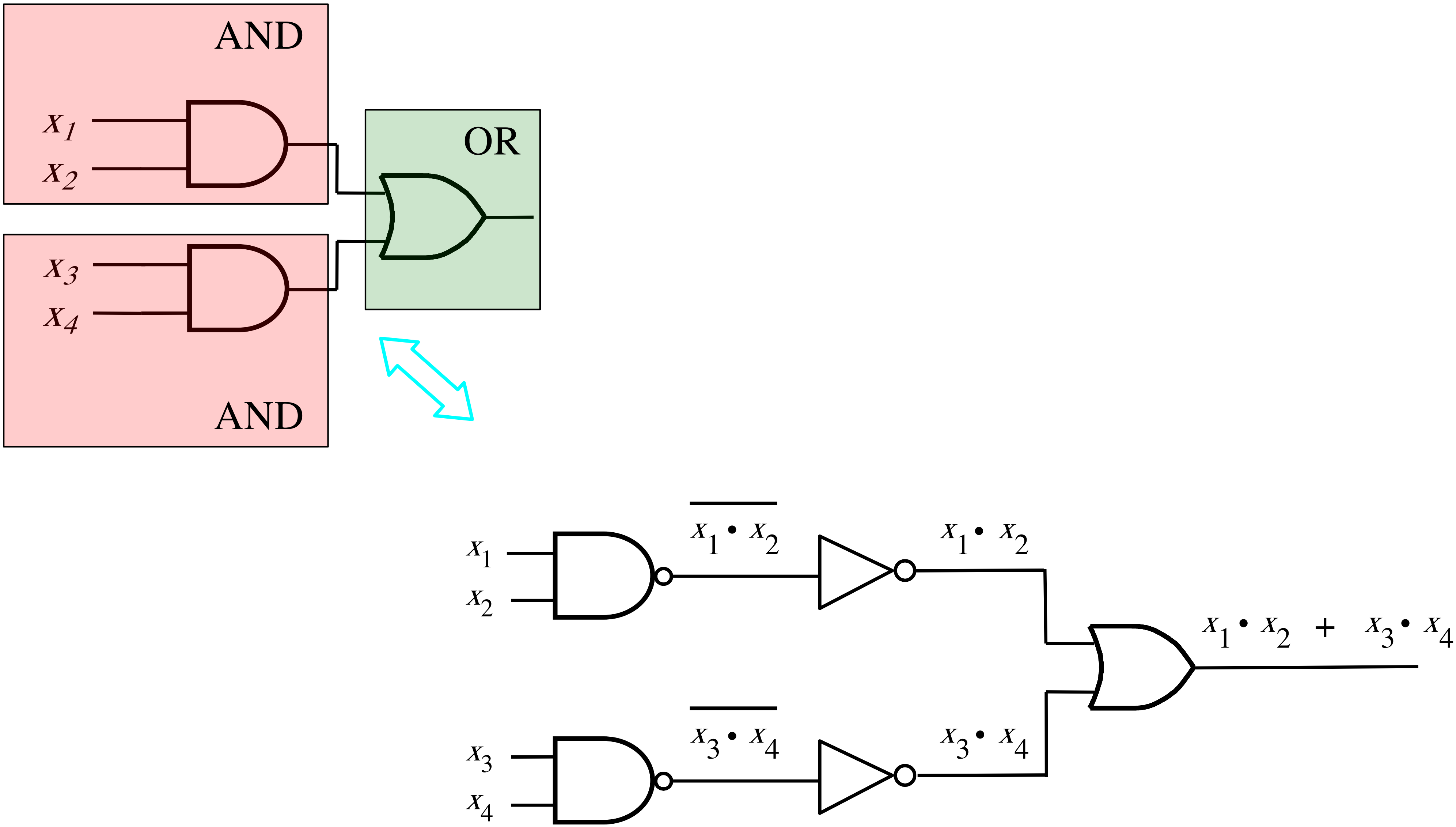
Sum-Of-Products



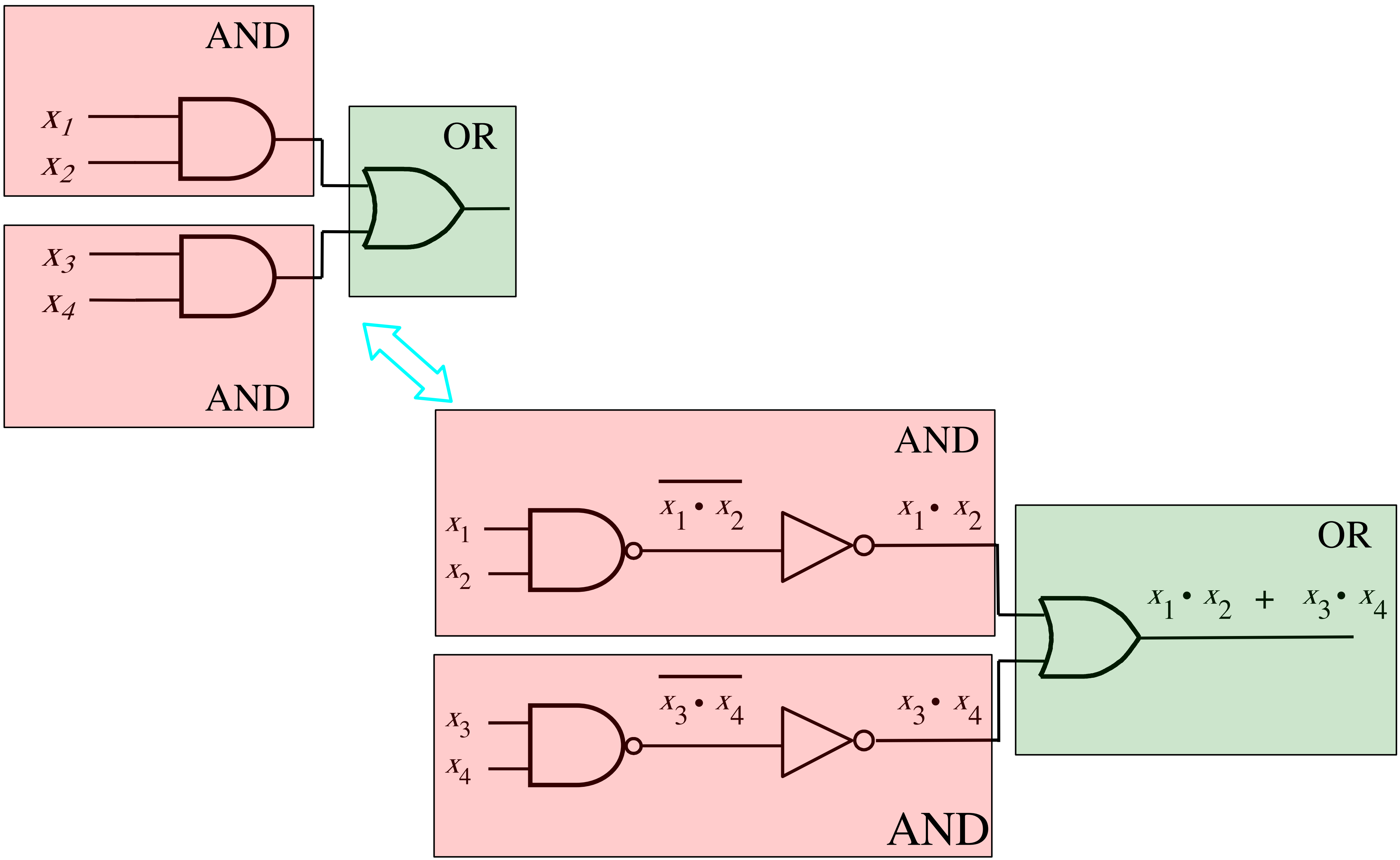
Sum-Of-Products



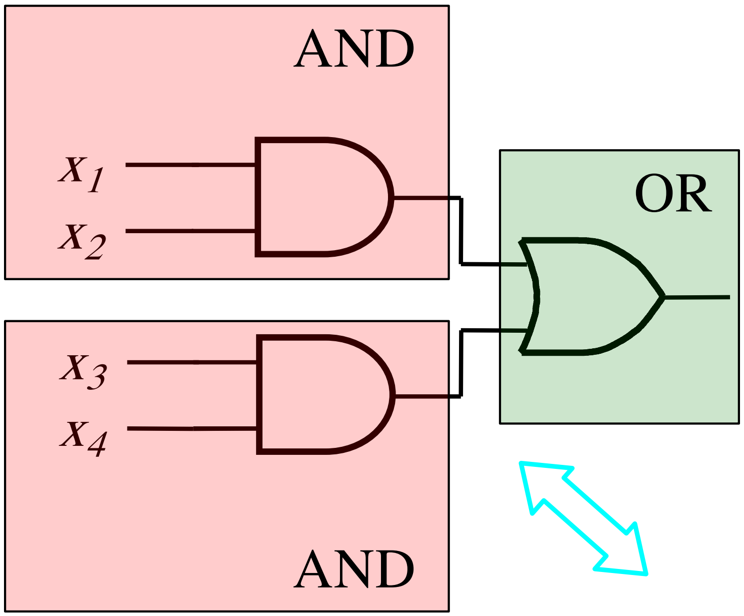
Sum-Of-Products

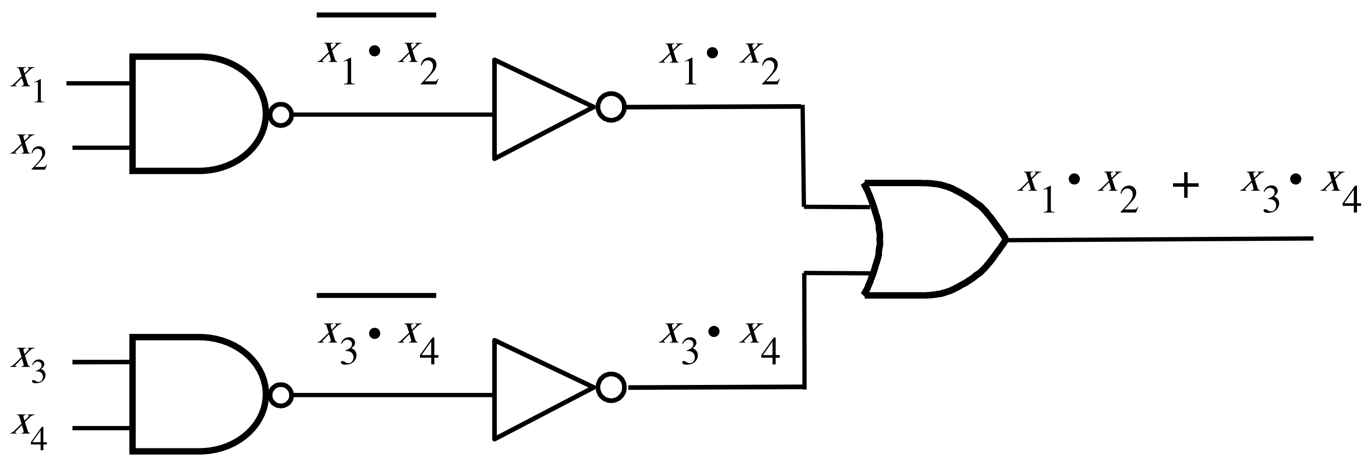


Sum-Of-Products

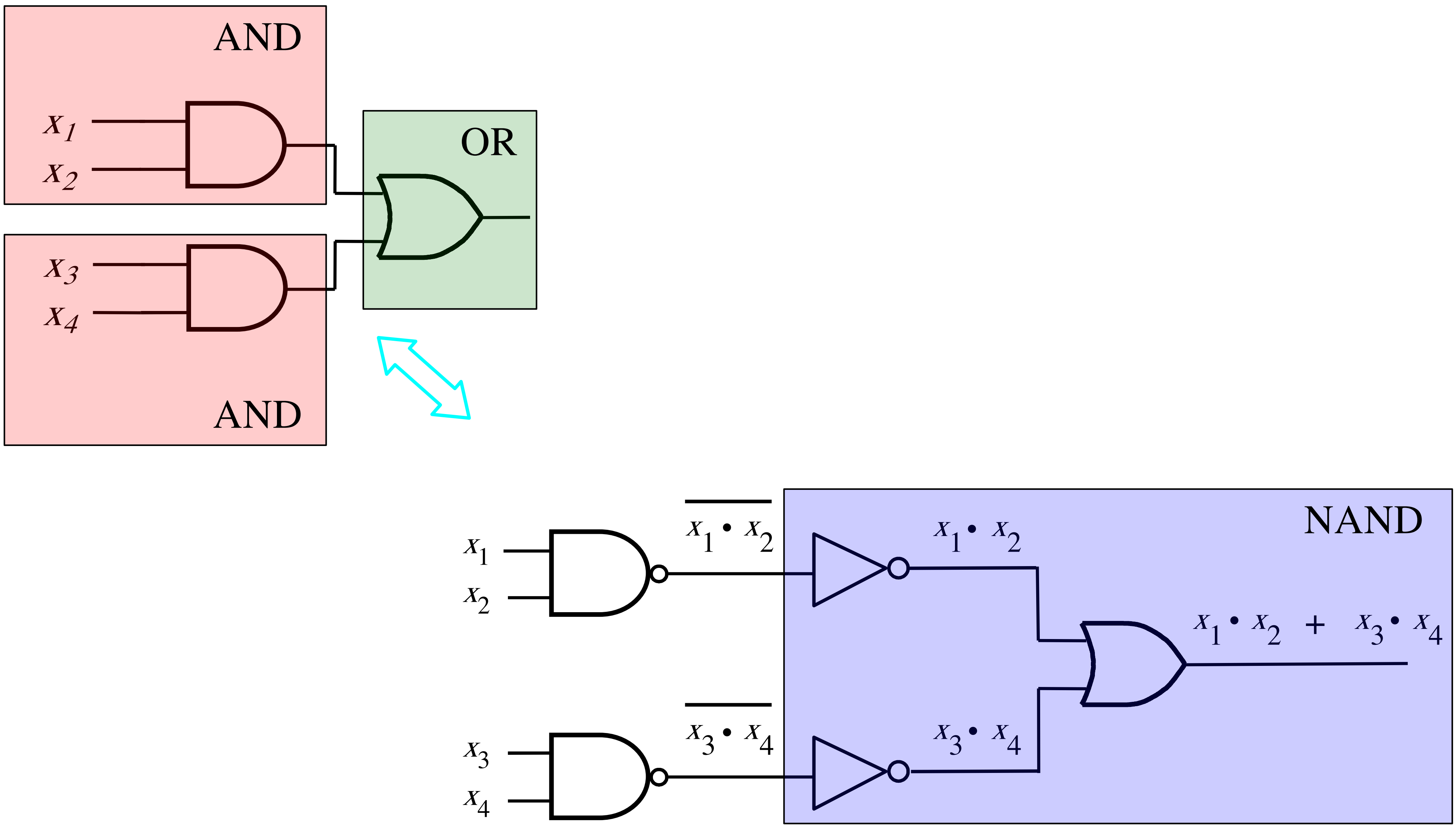


Sum-Of-Products

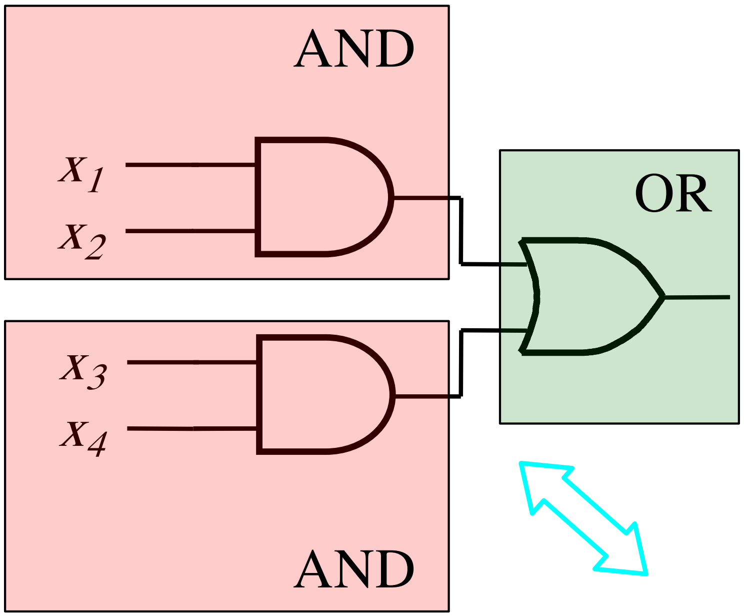


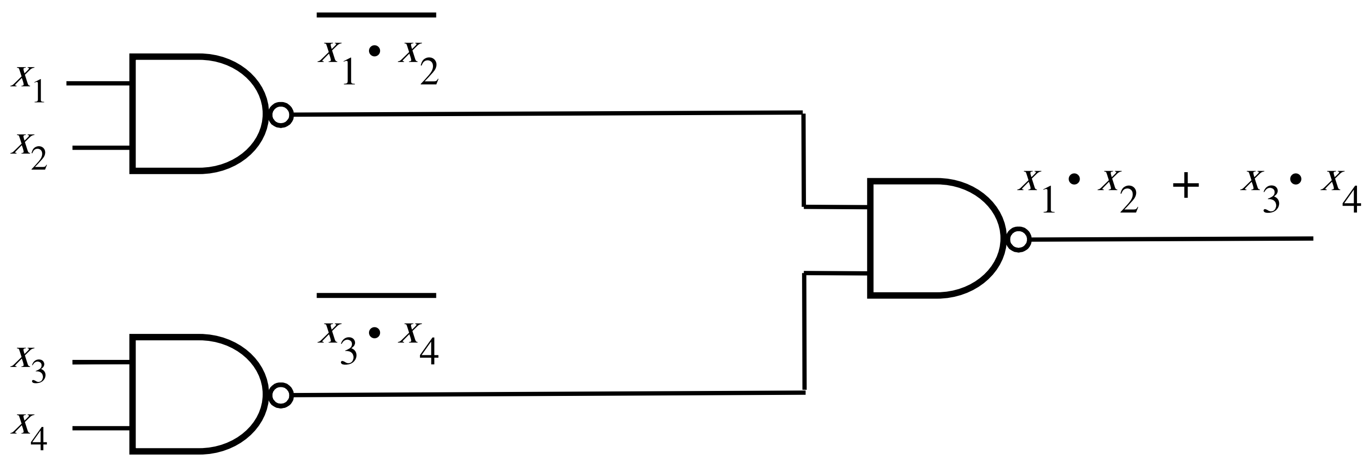


Sum-Of-Products

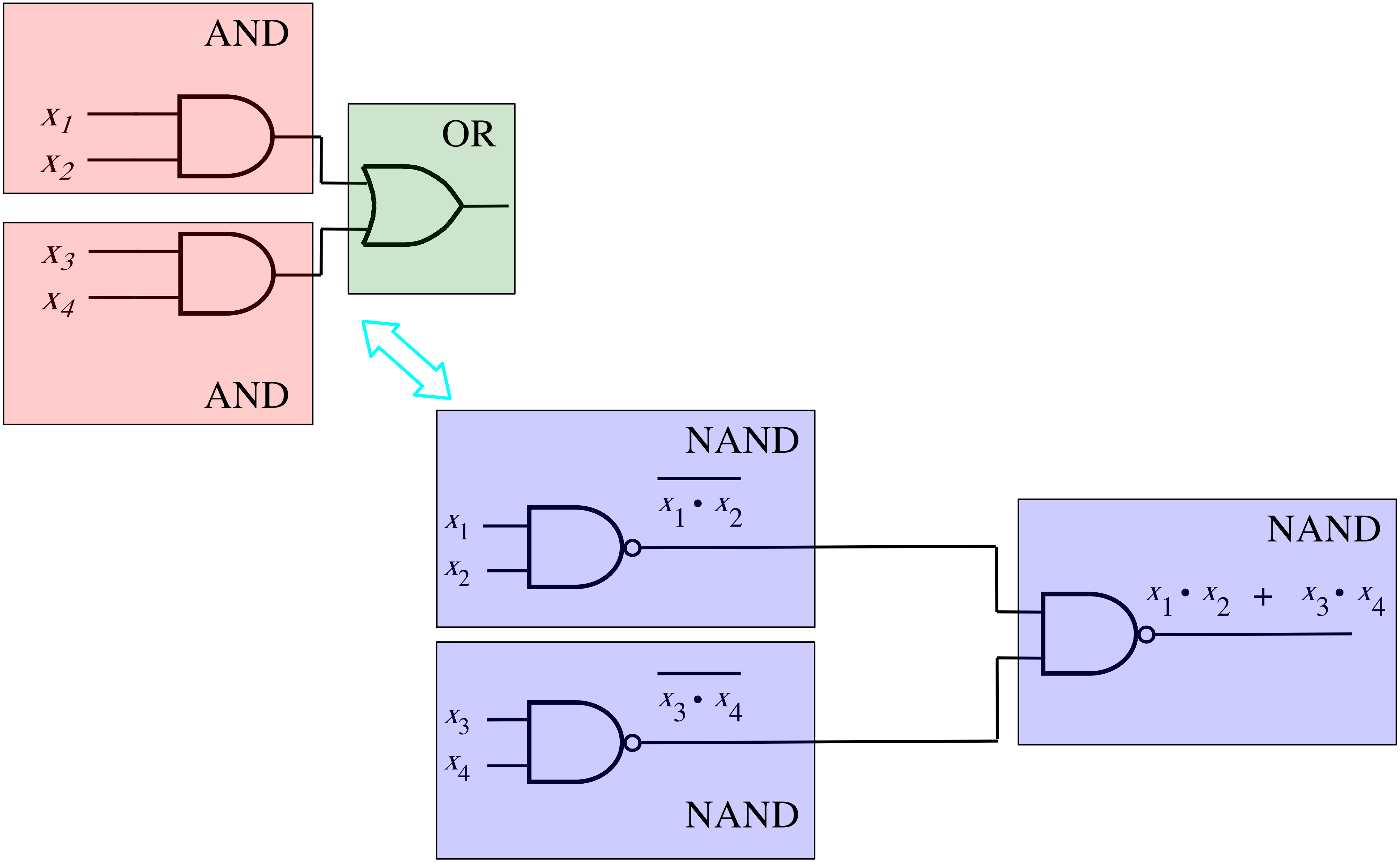


Sum-Of-Products



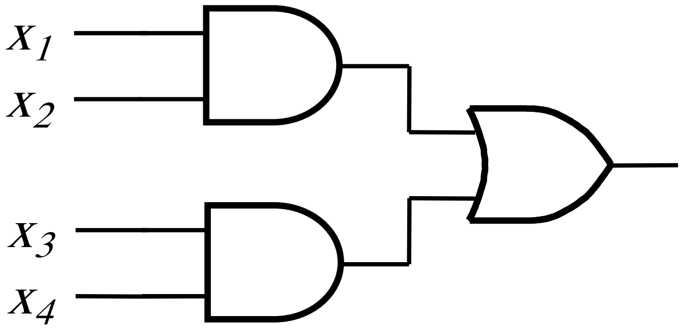


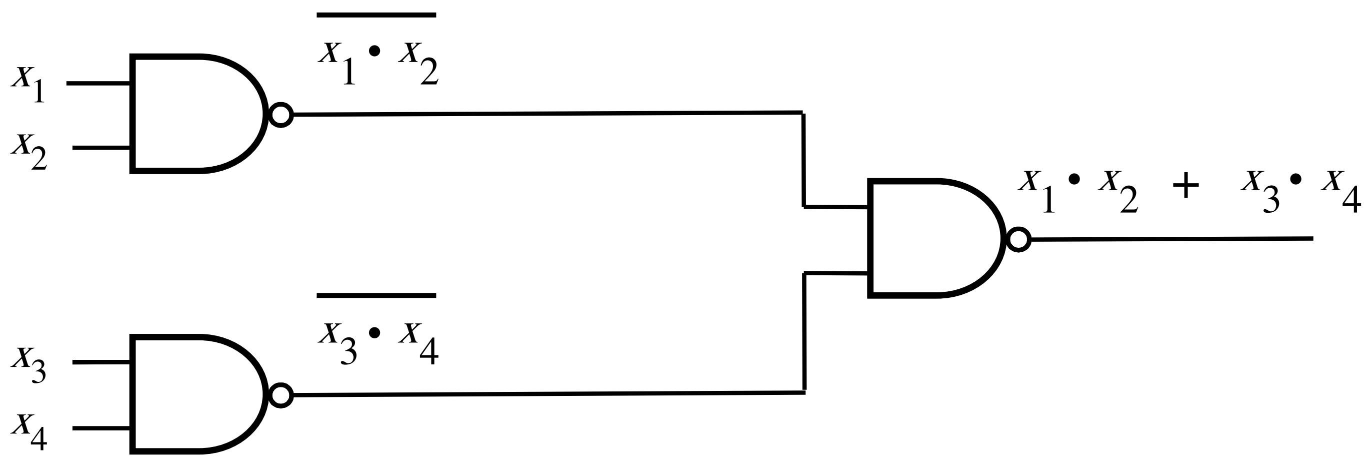
Sum-Of-Products



This circuit uses only NANDs

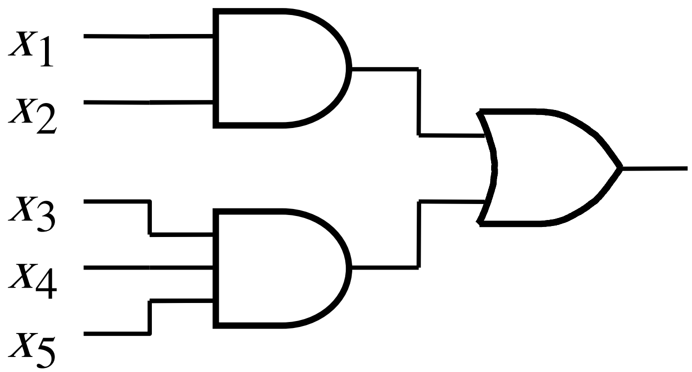
Sum-Of-Products



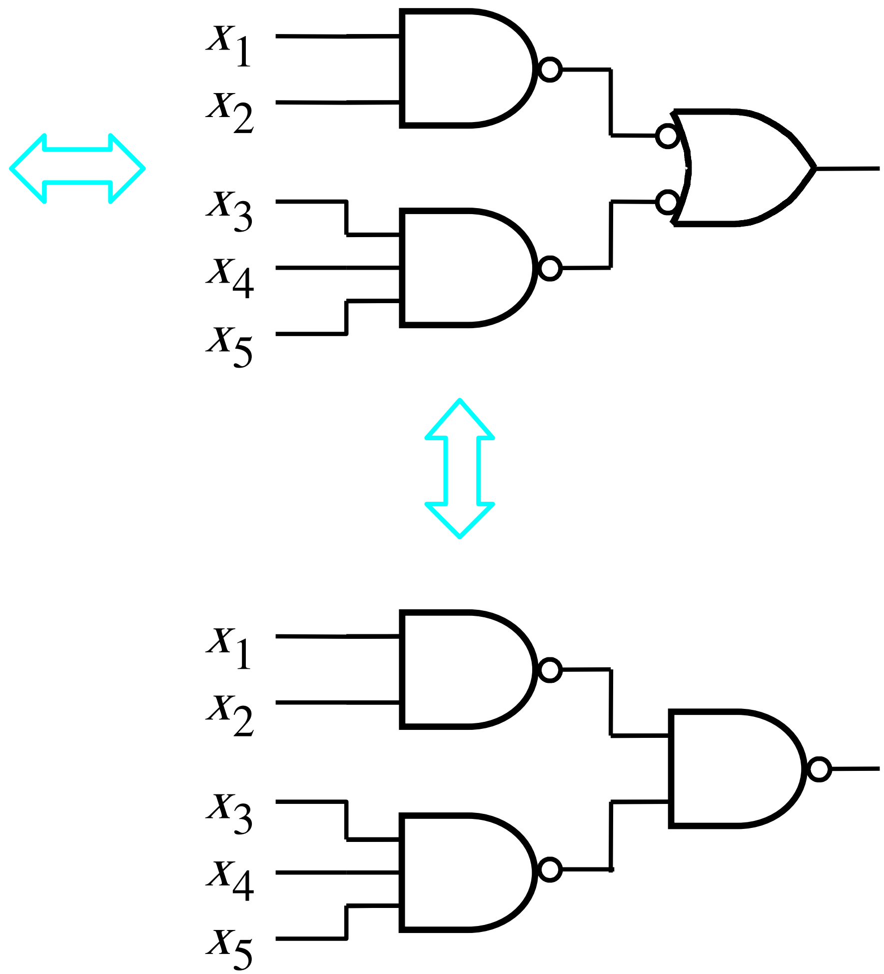


This circuit uses only NANDs

Another SOP Example



This circuit uses ANDs & OR

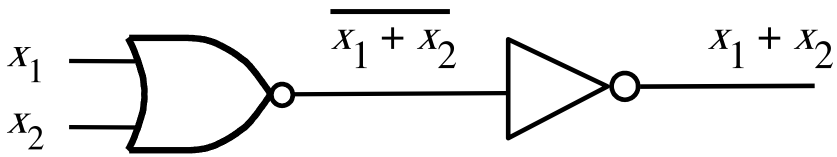


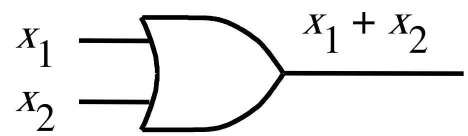
This circuit uses only NANDs

[Figure 2.27 from the textbook]

NOR-NOR Implementation of Product-of-Sums Expressions

NOR followed by NOT = OR





x_1	x_2	f	f
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

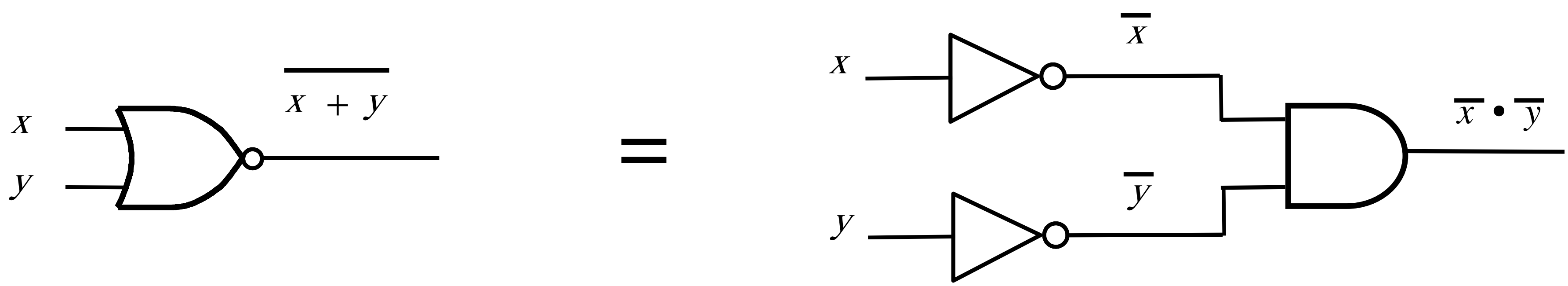
x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	1

DeMorgan's Theorem

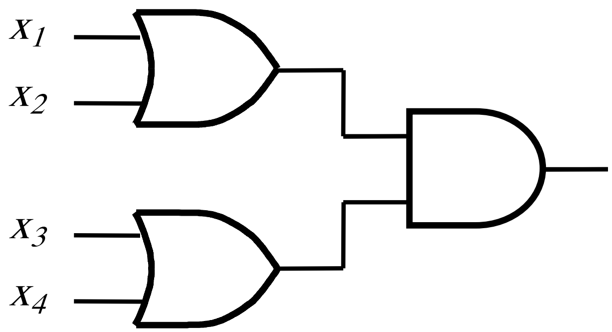
15b. $\overline{x + y} = \overline{x} \bullet \overline{y}$

DeMorgan's Theorem

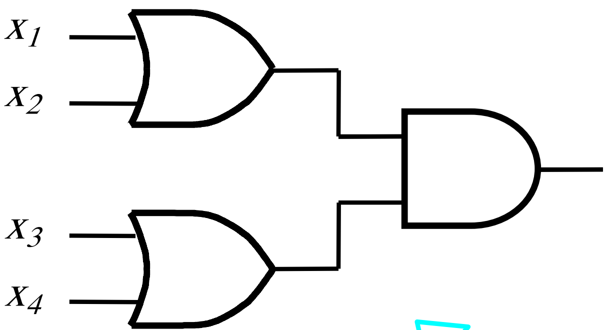
15b. $\overline{x + y} = \overline{x} \bullet \overline{y}$

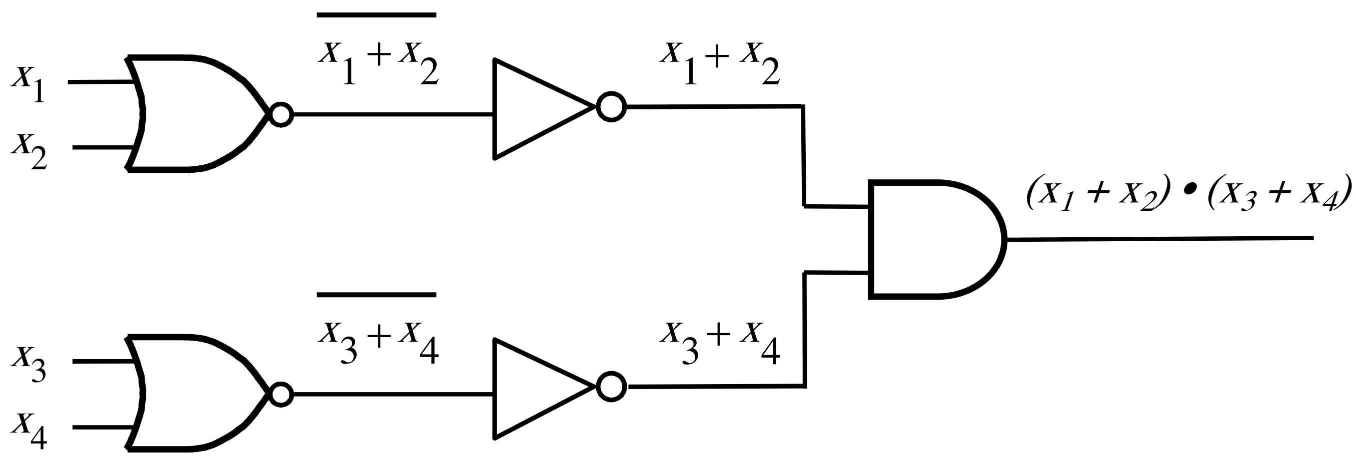


Product-Of-Sums

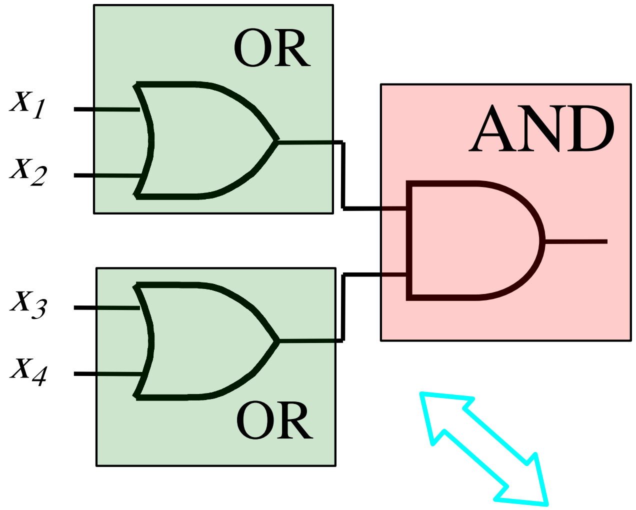


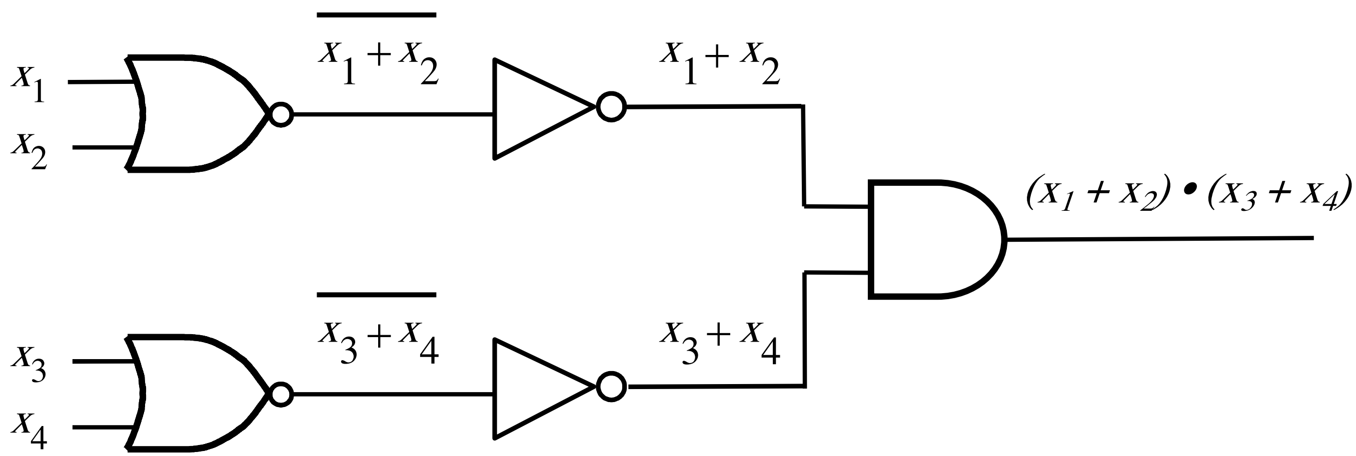
Product-Of-Sums



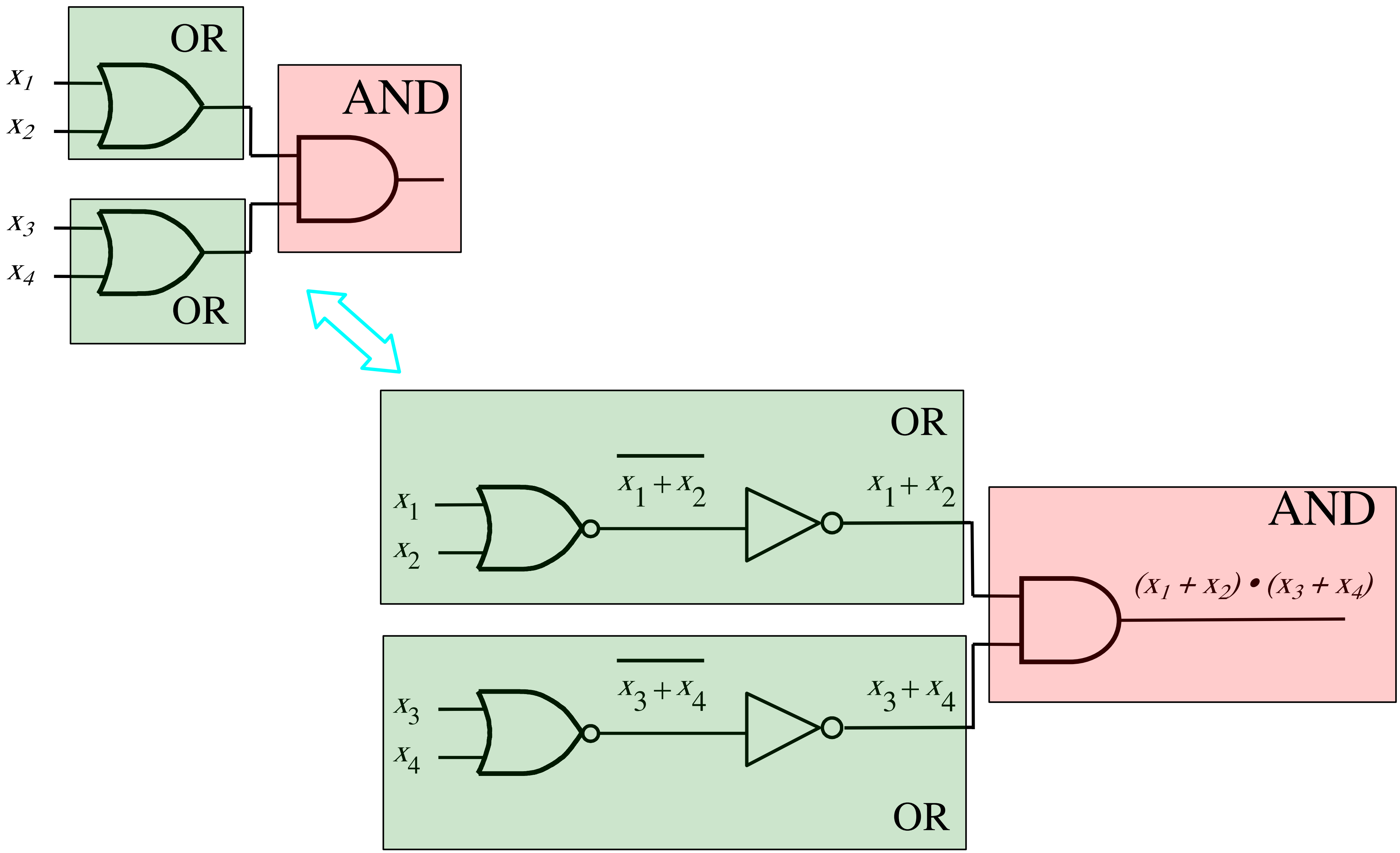


Product-Of-Sums

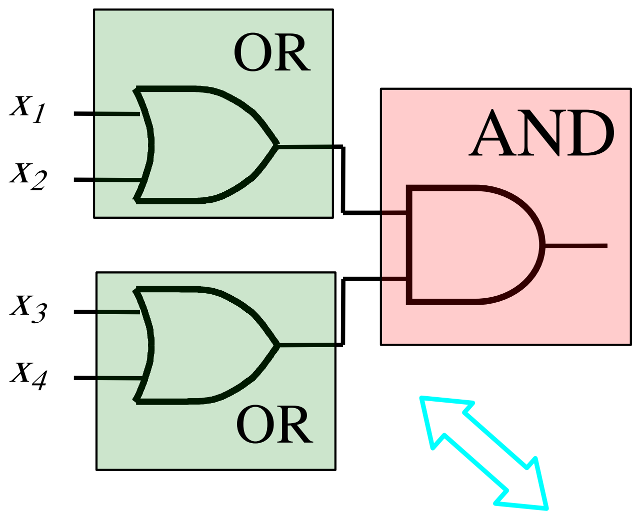


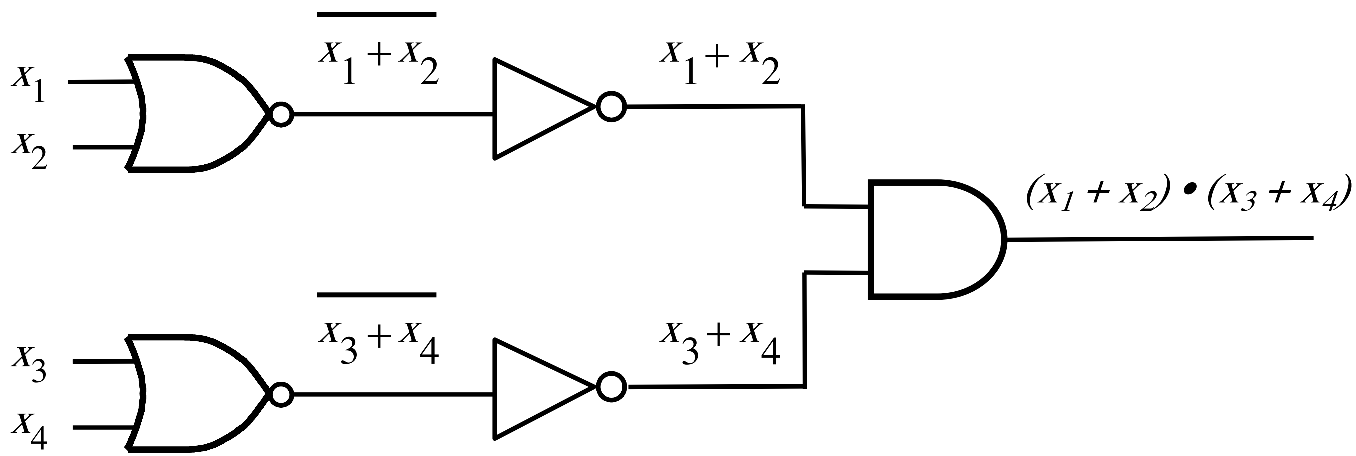


Product-Of-Sums

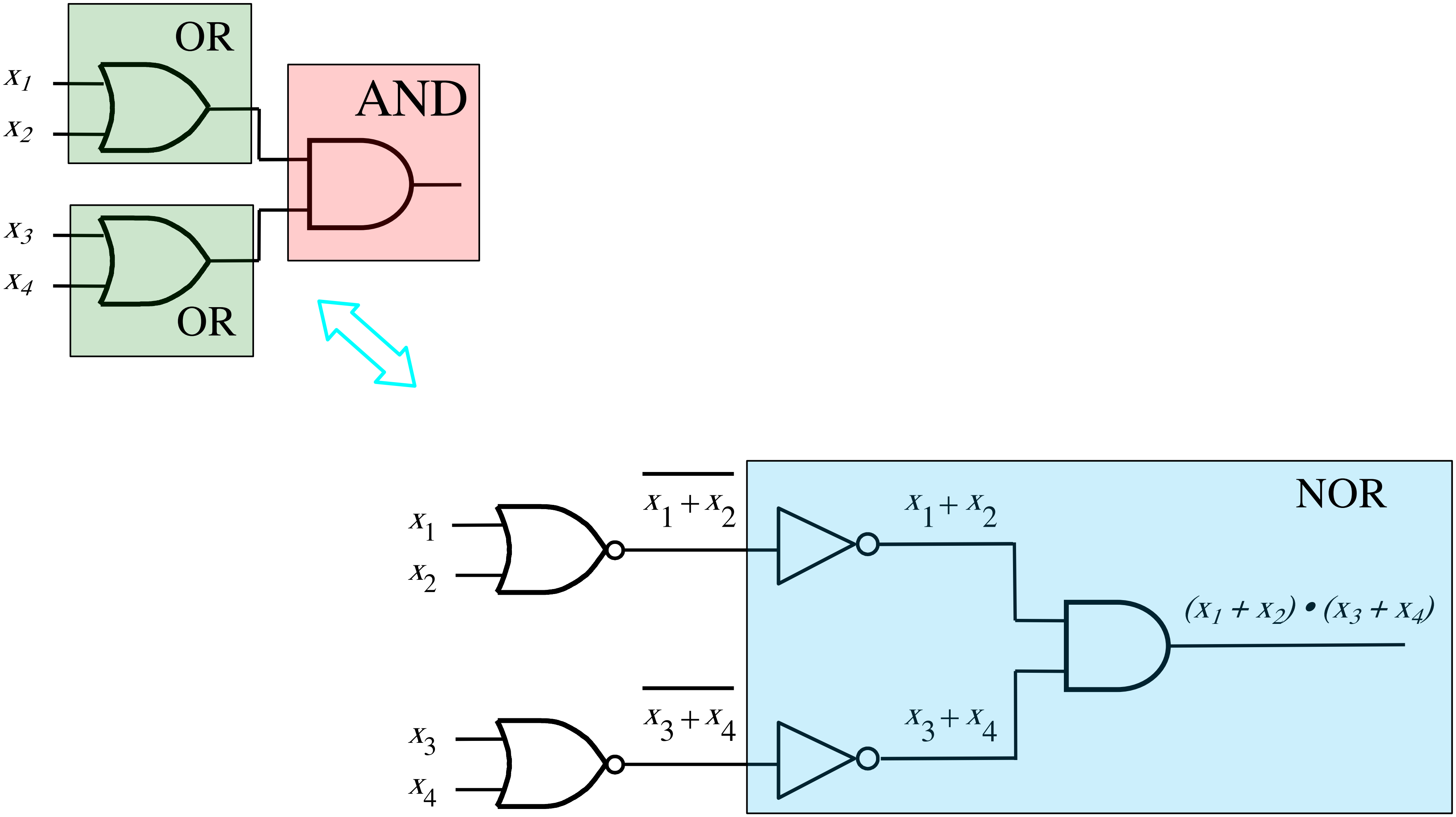


Product-Of-Sums

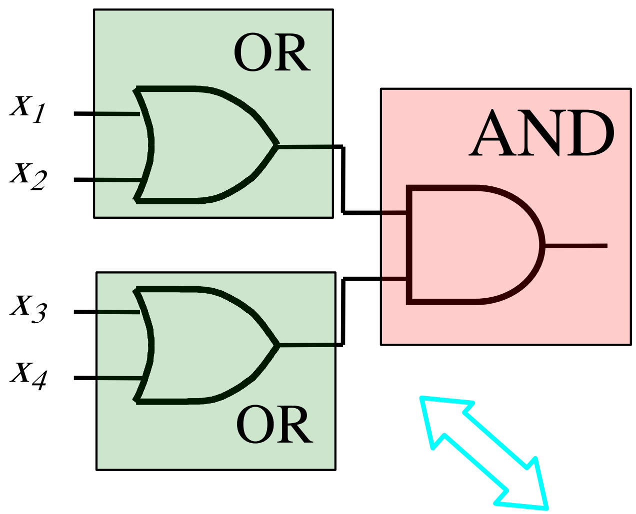


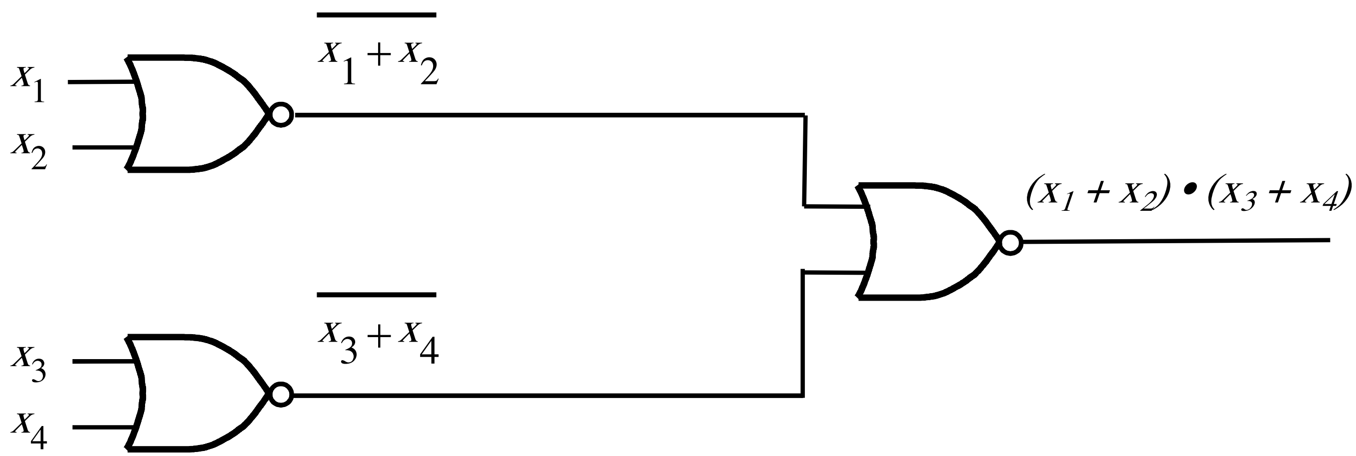


Product-Of-Sums

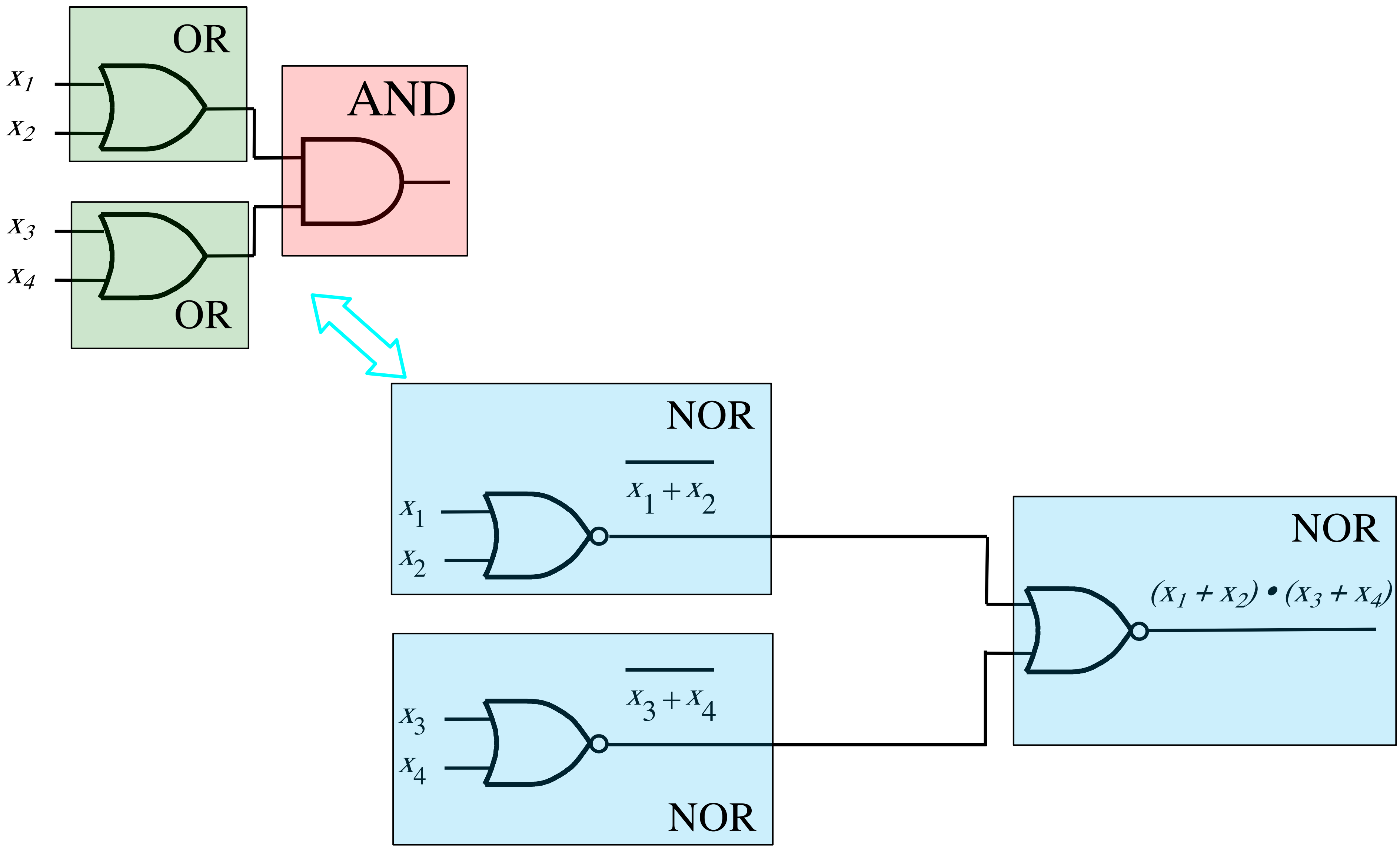


Product-Of-Sums



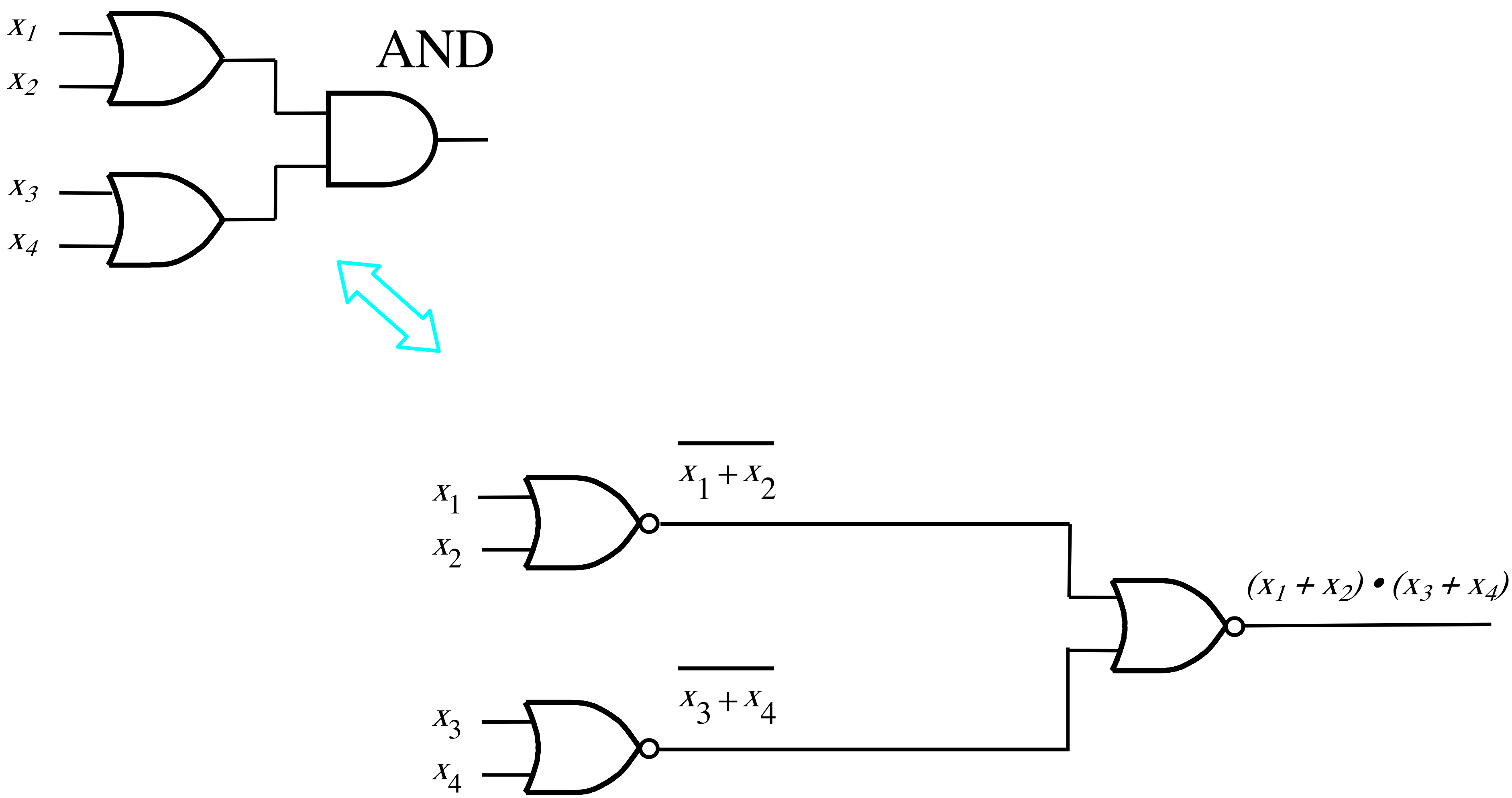


Product-Of-Sums



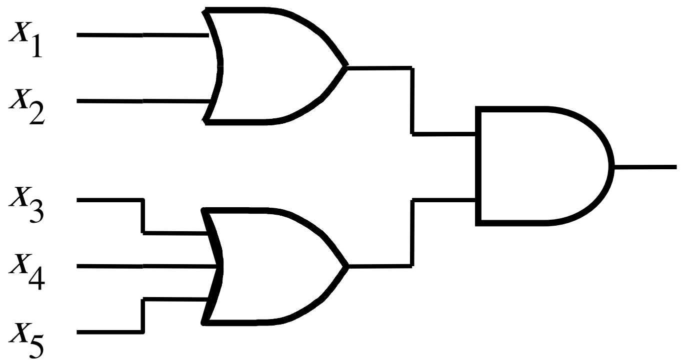
This circuit uses only NORs

Product-Of-Sums

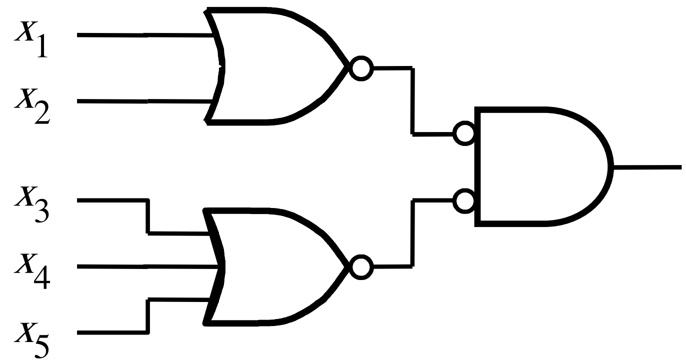


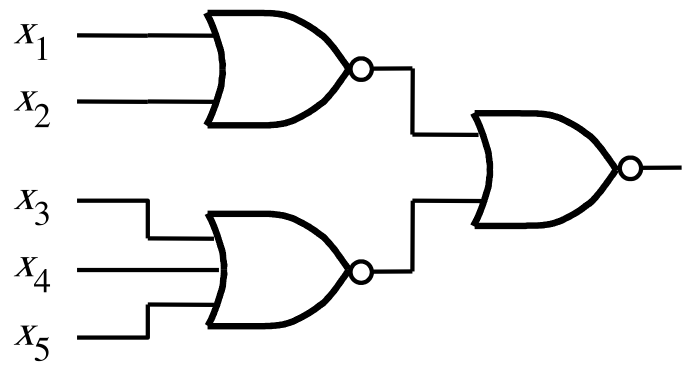
This circuit uses only NORs

Another POS Example



This circuit uses ORs & AND





This circuit uses only NORs

[Figure 2.28 from the textbook]

Questions?

THE END