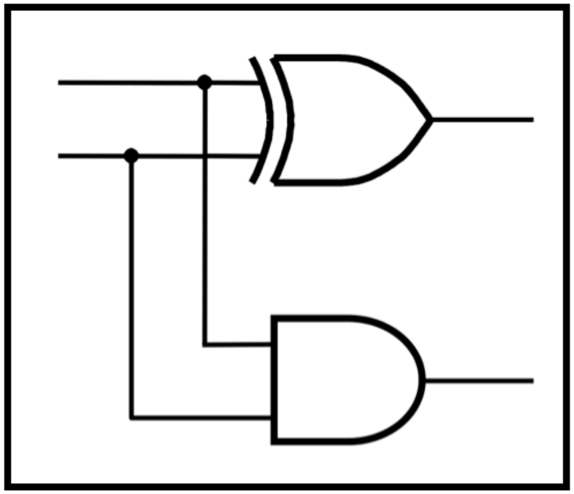


CprE 281: Digital Logic

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 Wednesday Sep 7

Administrative Stuff

- HW3 is out

- It is due on Monday Sep 12 @ 4pm.

- Please write clearly on the first page (in BLOCK CAPITAL letters) the following three things:

  - Your First and Last Name
  - Your Student ID Number
  - Your Lab Section Letter
  - Staple all of your pages together

- If any of these are missing, then you will lose 10% of your grade for that homework.

Labs Next Week

- **If your lab is on Mondays, i,e.,**

- **Section N: Mondays, 9:00 - 11:50 am (Coover Hall, room 1318)**
- **Section P: Mondays, 12:10 - 3:00 pm (Coover Hall, room 1318)**
- **Section R: Mondays, 5:10 - 8:00 pm (Coover Hall, room 1318)**

- **You will have 2 labs in one on September 12.**

- **That is, Lab #2 and Lab #3.**

Labs Next Week

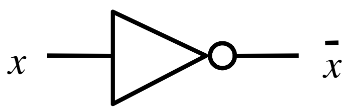
- **If your recitation is on Mondays, please go to one of the other 9 recitations next week:**

- **Section U: Tuesday 11:00 AM - 1:50 PM (Coover Hall, room 2050)**
  Section M: Tuesday 2:10 PM - 5:00 PM (Coover Hall, room 2050)
  Section J: Wednesday 8:00 AM - 10:50 AM (Coover Hall, room 2050)
  Section T: Wednesday 6:10 PM - 9:00 PM (Coover Hall, room 1318)
  Section Q: Thursday 11:00 AM - 1:50 PM (Coover Hall, room 2050)
  Section V: Thursday 11:00 AM - 1:50 PM (Coover Hall, room 1318)
  Section L: Thursday 2:10 PM - 5:00 PM (Coover Hall, room 2050)
  Section K: Thursday 5:10 PM - 8:00 PM (Coover Hall, room 2050)
  Section G: Friday 11:00 AM - 1:50 PM (Coover Hall, room 2050)

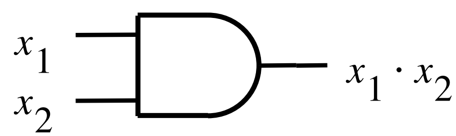
- **This is only for next week. And only for the recitation (first hour). You won't be able to stay for the lab as the sections are full.**

Quick Review

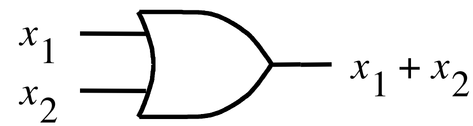
The Three Basic Logic Gates



NOT gate



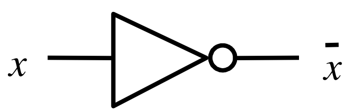
AND gate



OR gate

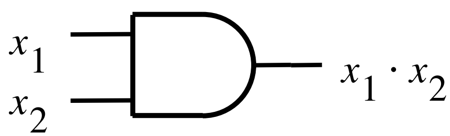
[Figure 2.8 from the textbook]

Truth Table for NOT



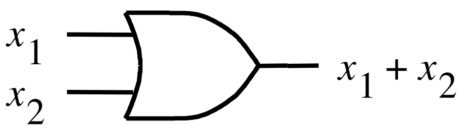
x	$\overline{x}$
0	1
1	0

Truth Table for AND



x_1	x_2	$x_1 \cdot x_2$
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table for OR



x_1	x_2	$x_1 + x_2$
0	0	0
0	1	1
1	0	1
1	1	1

DeMorgan's Theorem

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

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

Synthesize the Following Function

x_1	x_2	$f(x_1,x_2)$
0	0	1
0	1	1
1	0	0
1	1	1

Split the function into 4 functions

x_1	x_2	$f_{00}(x_1,x_2)$	$f_{01}(x_1,x_2)$	$f_{10}(x_1,x_2)$	$f_{11}(x_1,x_2)$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	0	0
1	1	0	0	0	1

Split the function into 4 functions

x_1	x_2	$f_{00}(x_1,x_2)$	$f_{01}(x_1,x_2)$	$f_{10}(x_1,x_2)$	$f_{11}(x_1,x_2)$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	0	0
1	1	0	0	0	1

Write Expressions for all four

x_1	x_2	$f_{00}(x_1,x_2)$	$f_{01}(x_1,x_2)$	$f_{10}(x_1,x_2)$	$f_{11}(x_1,x_2)$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	0	0
1	1	0	0	0	1
		$\bar{x}_1\bar{x}_2$	$\bar{x}_1 x_2$	0	$x_1 x_2$

Then just add them together

x_1	x_2	$f_{00}(x_1,x_2)$	$f_{01}(x_1,x_2)$	$f_{10}(x_1,x_2)$	$f_{11}(x_1,x_2)$
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	0	0
1	1	0	0	0	1

$$f(x_1, x_2) = \bar{x}_1\bar{x}_2 + \bar{x}_1 x_2 + 0 + x_1 x_2$$

Example 2.10

Implement the function $f(x_1, x_2, x_3) = \Sigma\, m(2, 3, 4, 6, 7)$

Minterms and Maxterms (with three variables)

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	$m_7 = x_1x_2x_3$	$M_7 = \overline{x}_1 + \overline{x}_2 + \overline{x}_3$

[Figure 2.22 from the textbook]

Minterms and Maxterms (with three variables)

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	$m_7 = x_1x_2x_3$	$M_7 = \overline{x}_1 + \overline{x}_2 + \overline{x}_3$

$$f(x_1, x_2, x_3) = \sum m(2, 3, 4, 6, 7)$$

- **The SOP expression is:**

$$\begin{aligned} f &= m_2 + m_3 + m_4 + m_6 + m_7 \\ &= \bar{x}_1 x_2 \bar{x}_3 + \bar{x}_1 x_2 x_3 + x_1 \bar{x}_2 \bar{x}_3 + x_1 x_2 \bar{x}_3 + x_1 x_2 x_3 \end{aligned}$$

- **This could be simplified as follows:**

$$\begin{aligned} f &= \bar{x}_1 x_2 (\bar{x}_3 + x_3) + x_1 (\bar{x}_2 + x_2) \bar{x}_3 + x_1 x_2 (\bar{x}_3 + x_3) \\ &= \bar{x}_1 x_2 + x_1 \bar{x}_3 + x_1 x_2 \\ &= (\bar{x}_1 + x_1) x_2 + x_1 \bar{x}_3 \\ &= x_2 + x_1 \bar{x}_3 \end{aligned}$$

Example 2.12

Implement the function $f(x_1, x_2, x_3) = \Pi\, M(0, 1, 5)$,

which is equivalent to $f(x_1, x_2, x_3) = \Sigma\, m(2, 3, 4, 6, 7)$

Minterms and Maxterms (with three variables)

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	$m_7 = x_1x_2x_3$	$M_7 = \overline{x}_1 + \overline{x}_2 + \overline{x}_3$

$$f(x_1, x_2, x_3) = \Pi M(0, 1, 5)$$

- **The POS expression is:**

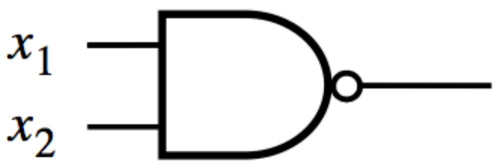
$$\begin{aligned} f &= M_0 \cdot M_1 \cdot M_5 \\ &= (x_1 + x_2 + x_3)(x_1 + x_2 + \overline{x}_3)(\overline{x}_1 + x_2 + \overline{x}_3) \end{aligned}$$

- **This could be simplified as follows:**

$$\begin{aligned} f &= (x_1 + x_2 + x_3)(x_1 + x_2 + \overline{x}_3)(x_1 + x_2 + \overline{x}_3)(\overline{x}_1 + x_2 + \overline{x}_3) \\ &= ((x_1 + x_2) + x_3)((x_1 + x_2) + \overline{x}_3)(x_1 + (x_2 + \overline{x}_3))(\overline{x}_1 + (x_2 + \overline{x}_3)) \\ &= ((x_1 + x_2) + x_3\overline{x}_3)(x_1\overline{x}_1 + (x_2 + \overline{x}_3)) \\ &= (x_1 + x_2)(x_2 + \overline{x}_3) \end{aligned}$$

Two New Logic Gates

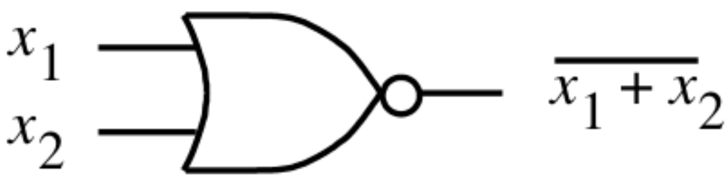
NAND Gate



$$\overline{x_1 \cdot x_2}$$

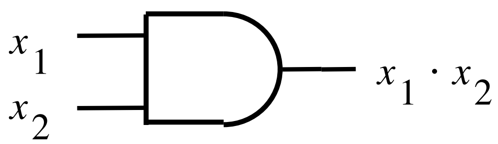
x_1	x_2	f
0	0	1
0	1	1
1	0	1
1	1	0

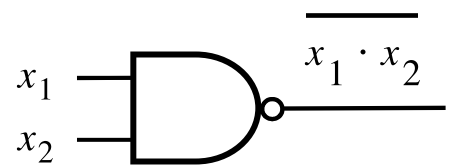
NOR Gate



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

AND vs NAND

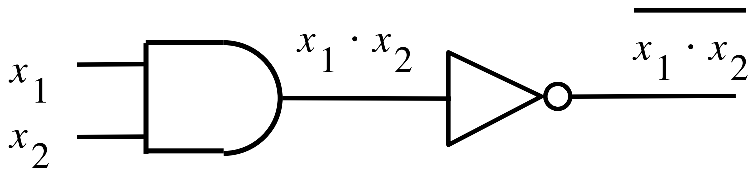


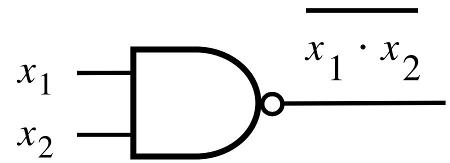


x_1	x_2	f
0	0	0
0	1	0
1	0	0
1	1	1

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

AND followed by NOT = NAND



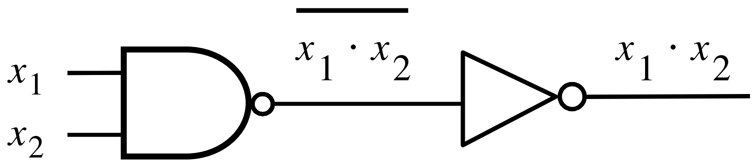


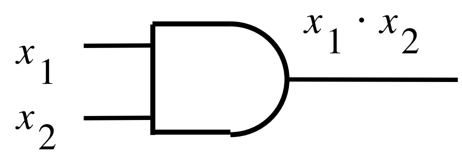
x_1	x_2	f
0	0	0
0	1	0
1	0	0
1	1	1

f
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



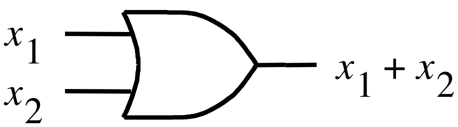


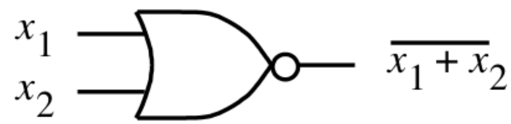
x_1	x_2	f
0	0	1
0	1	1
1	0	1
1	1	0

f
0
0
0
1

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

OR vs NOR

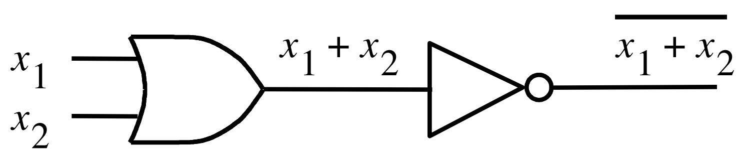


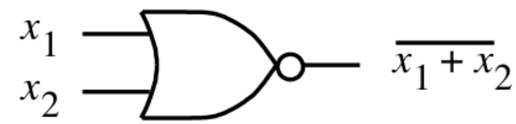


x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	1

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

OR followed by NOT = NOR



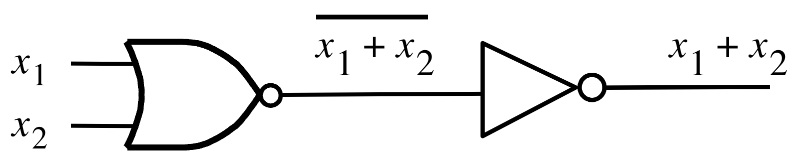


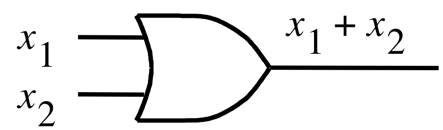
x_1	x_2	f
0	0	0
0	1	1
1	0	1
1	1	1

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
0	0	1
0	1	0
1	0	0
1	1	0

f
0
1
1
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?

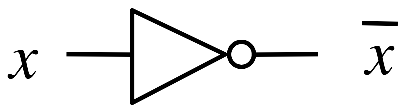
Why do we need two more gates?

They can be implemented with fewer transistors.

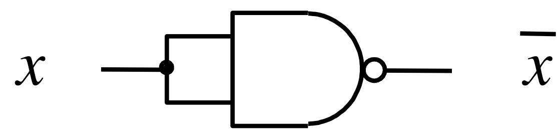
(more about this later)

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

Building a NOT Gate with NAND

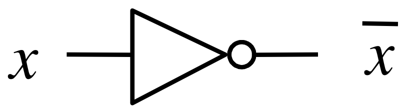


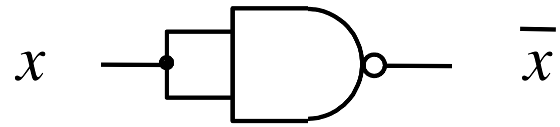
x	$\overline{x}$
0	1
1	0



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

Building a NOT Gate with NAND



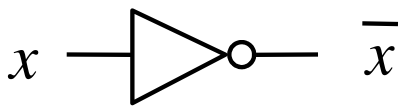


x	$\overline{x}$
0	1
1	0

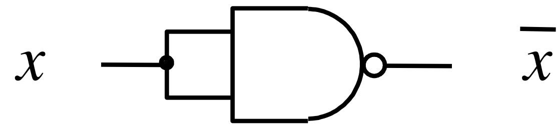
x	x	f
0	0	1
[redacted]		
1	1	0

impossible combinations

Building a NOT Gate with NAND



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



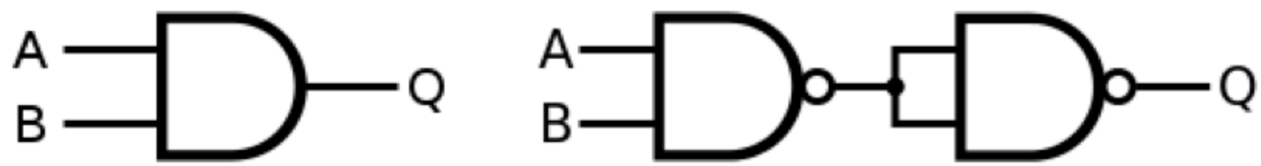
x	x	f
0	0	1
1	1	0

impossible combinations

Thus, the two truth tables are equal!

Building an AND gate with NAND gates

Desired AND Gate | **NAND Construction**



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

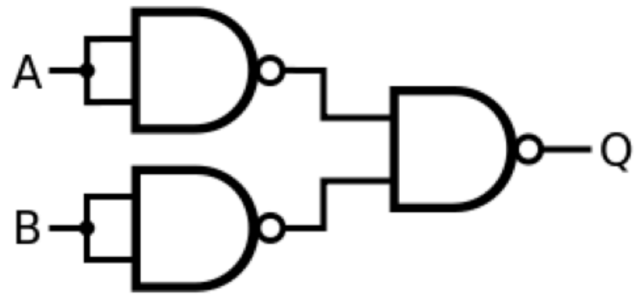
[http://en.wikipedia.org/wiki/NAND_logic]

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

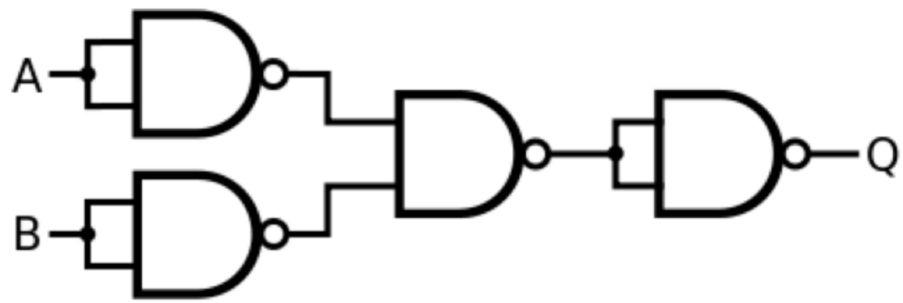
Any Boolean function can be implemented with only NAND gates!

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

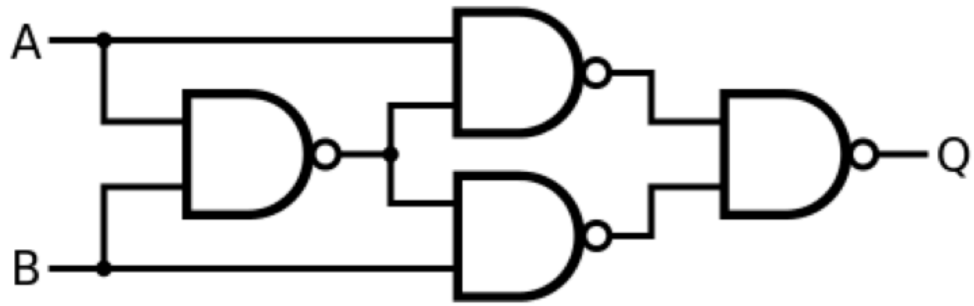
Input A	Input B	Output Q
0	0	1
0	1	0
1	0	0
1	1	0

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

XOR gate with NAND gates

Desired XOR Gate

NAND Construction



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

= 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

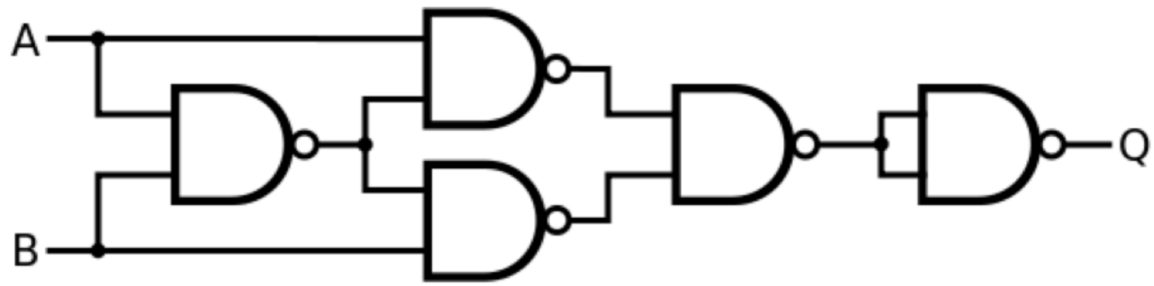
[http://en.wikipedia.org/wiki/NAND_logic]

XNOR gate with NAND gates

Desired XNOR Gate

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

NAND Construction



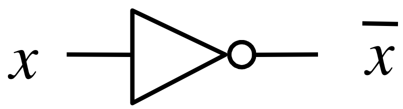
= NOT[NOT[NOT{**A** AND NOT(**A** AND **B**)} AND NOT{**B** AND NOT(**A** AND **B**)}]]

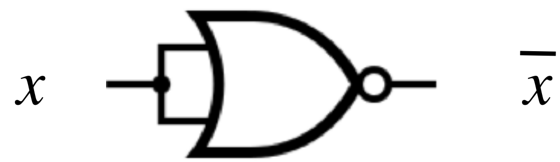
Truth Table

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

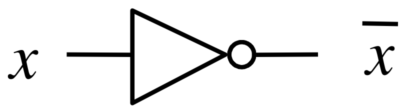




x	$\overline{x}$
0	1
1	0

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

Building a NOT Gate with NOR

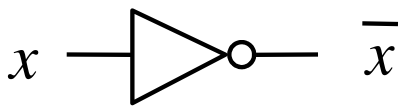


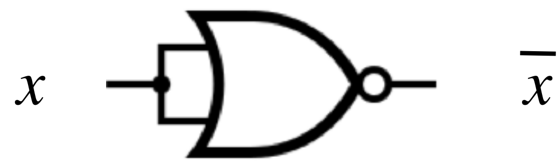
x	$\overline{x}$
0	1
1	0

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

impossible combinations

Building a NOT Gate with NOR





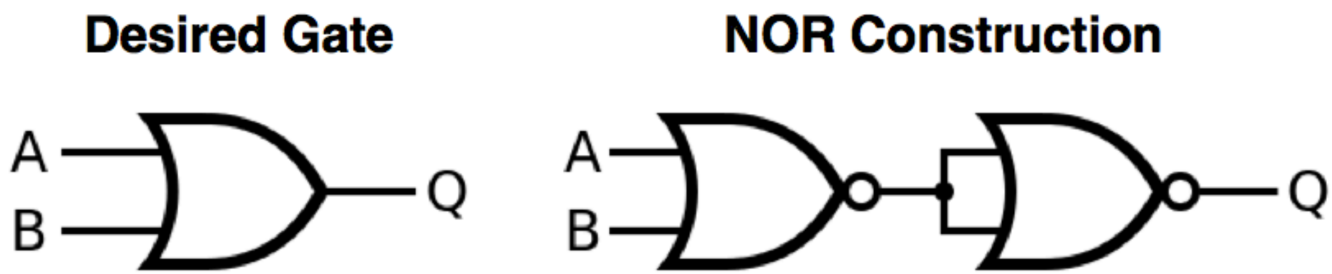
x	$\overline{x}$
0	1
1	0

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



Truth Table

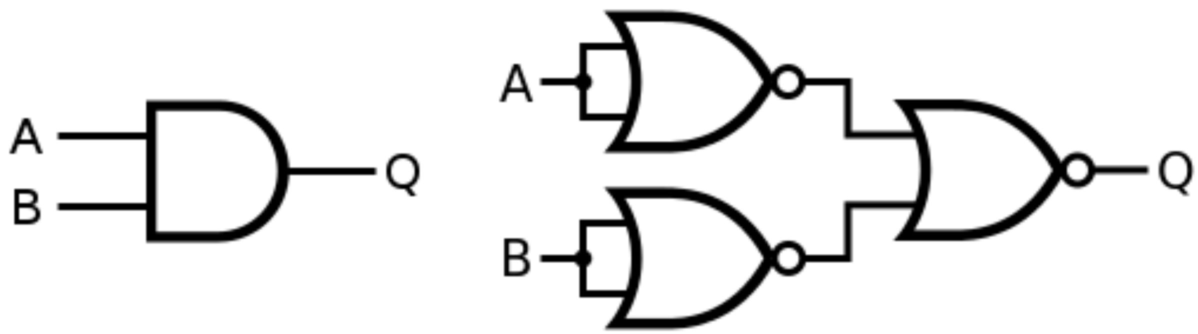
Input A	Input B		Output Q
0	0		0
0	1		1
1	0		1
1	1		1

[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]

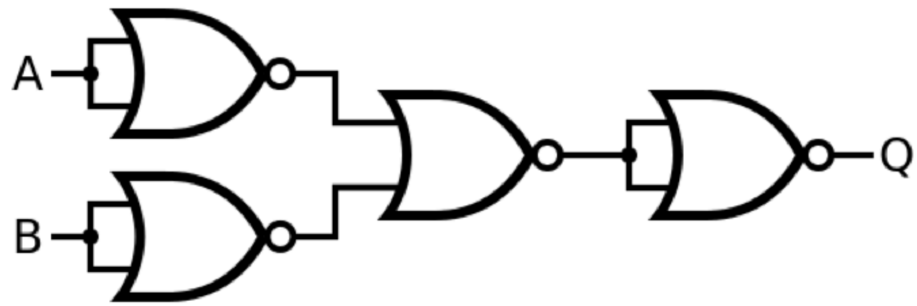
Implications

Any Boolean function can be implemented with only NOR gates!

NAND gate with NOR gates

Desired Gate

NOR Construction

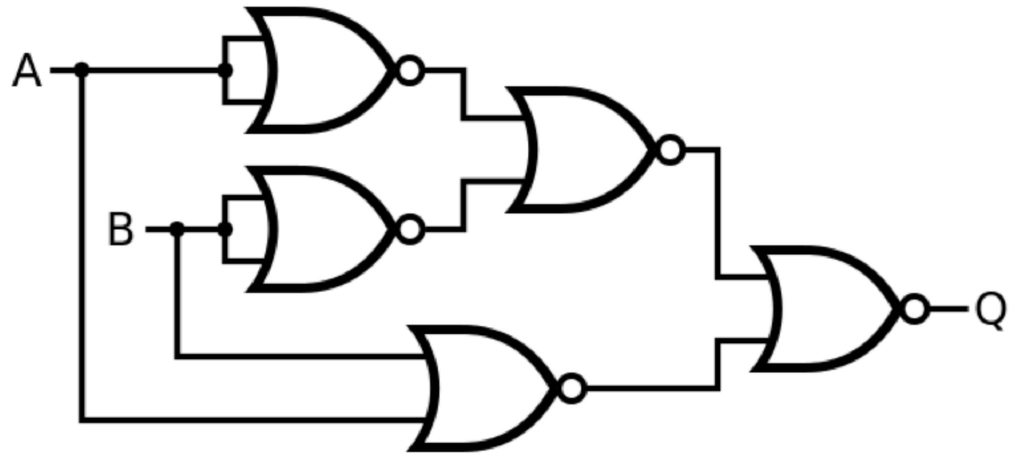


Truth Table

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



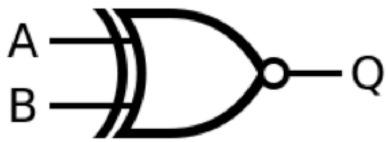
Truth Table

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

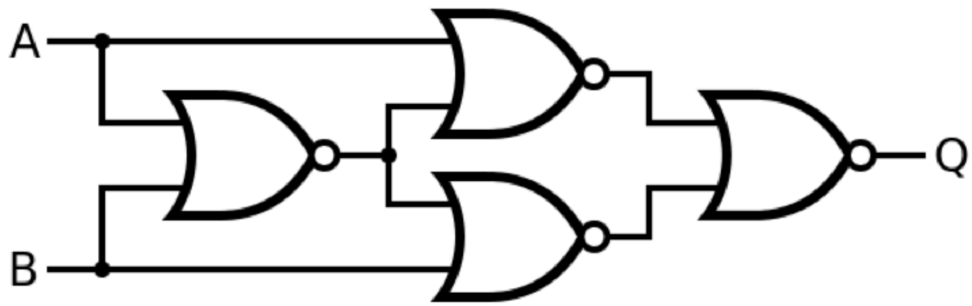
[http://en.wikipedia.org/wiki/NOR_logic]

XNOR gate with NOR gates

Desired XNOR Gate



NOR Construction

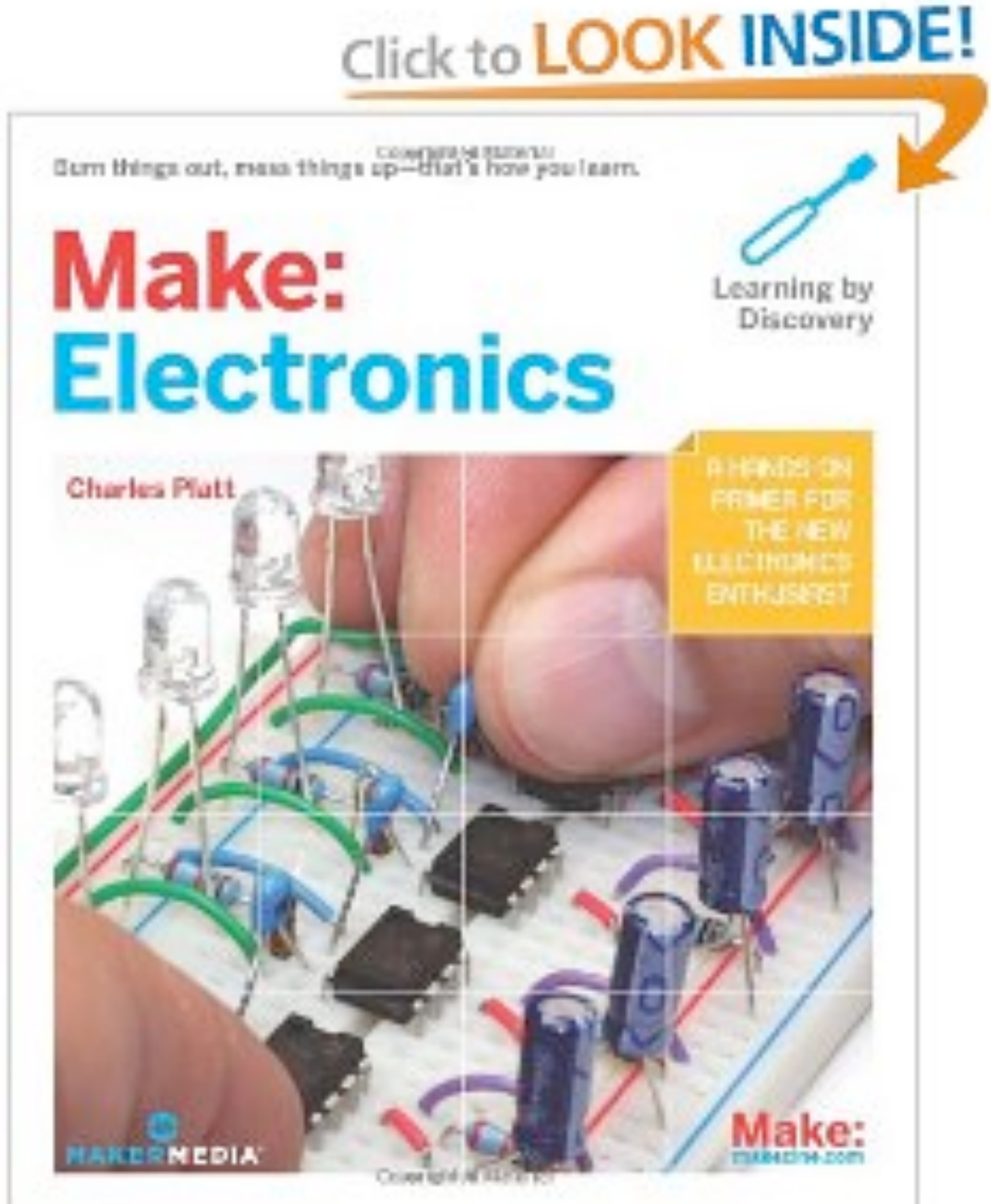


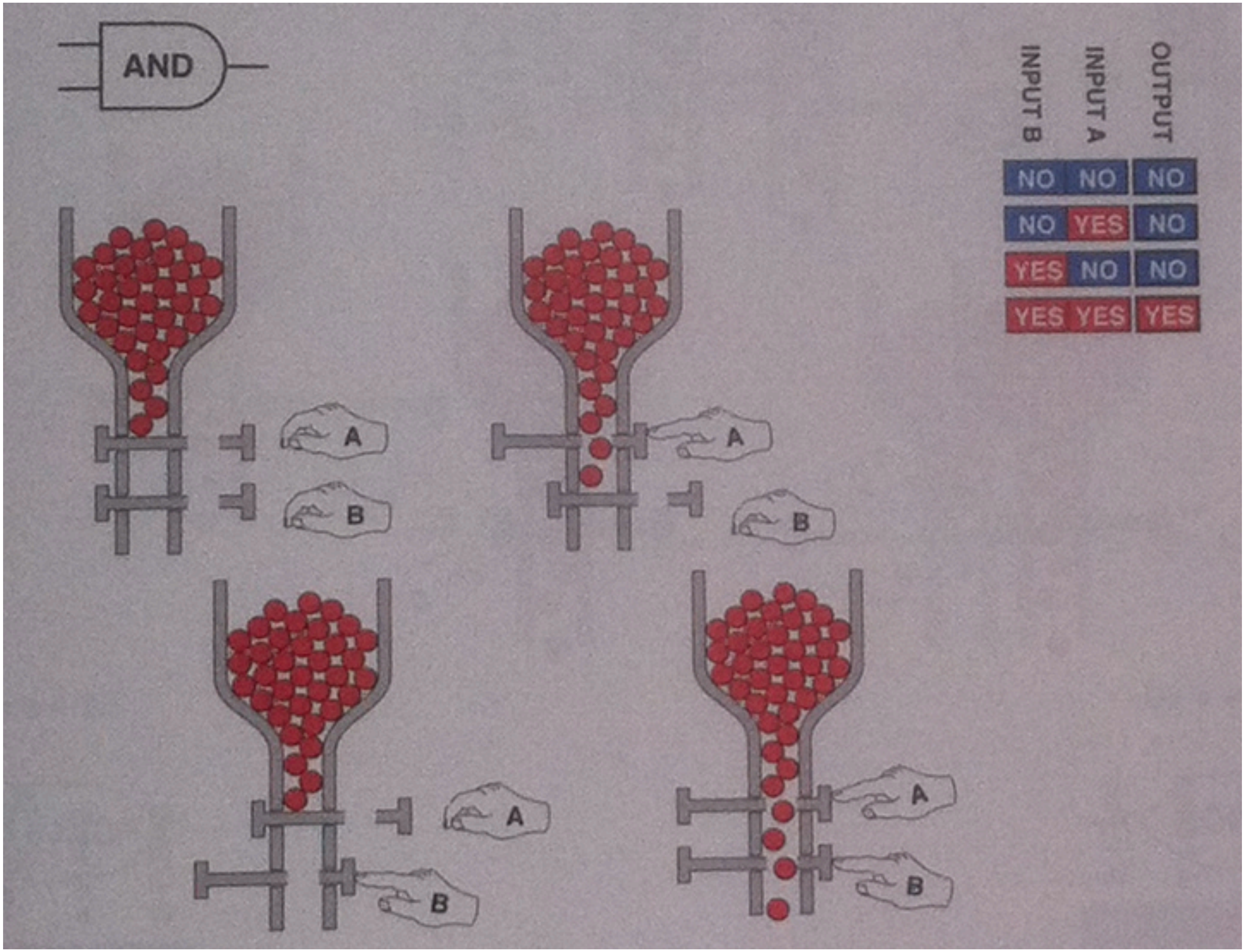
Truth Table

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

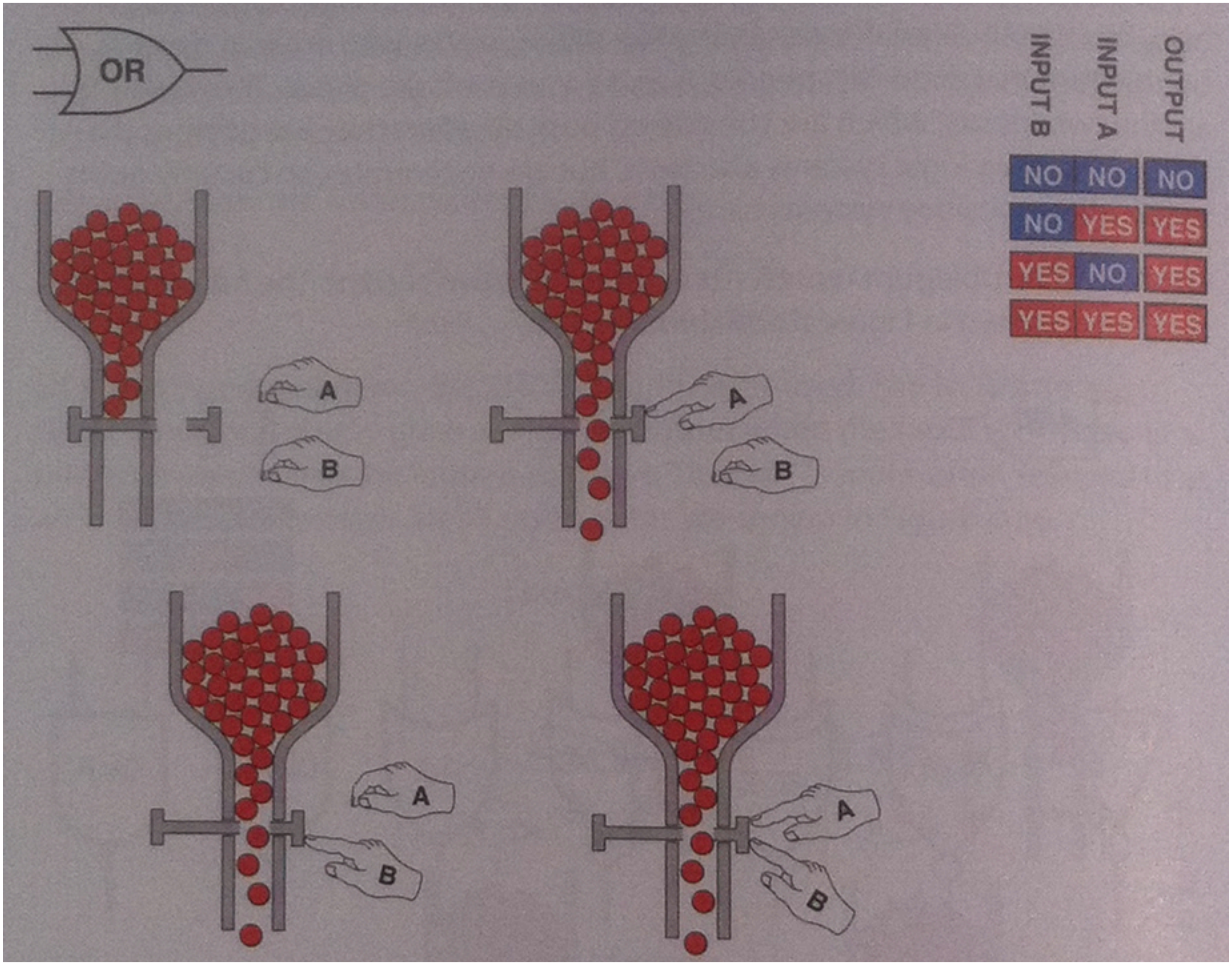
[http://en.wikipedia.org/wiki/NOR_logic]

The following examples came from this book

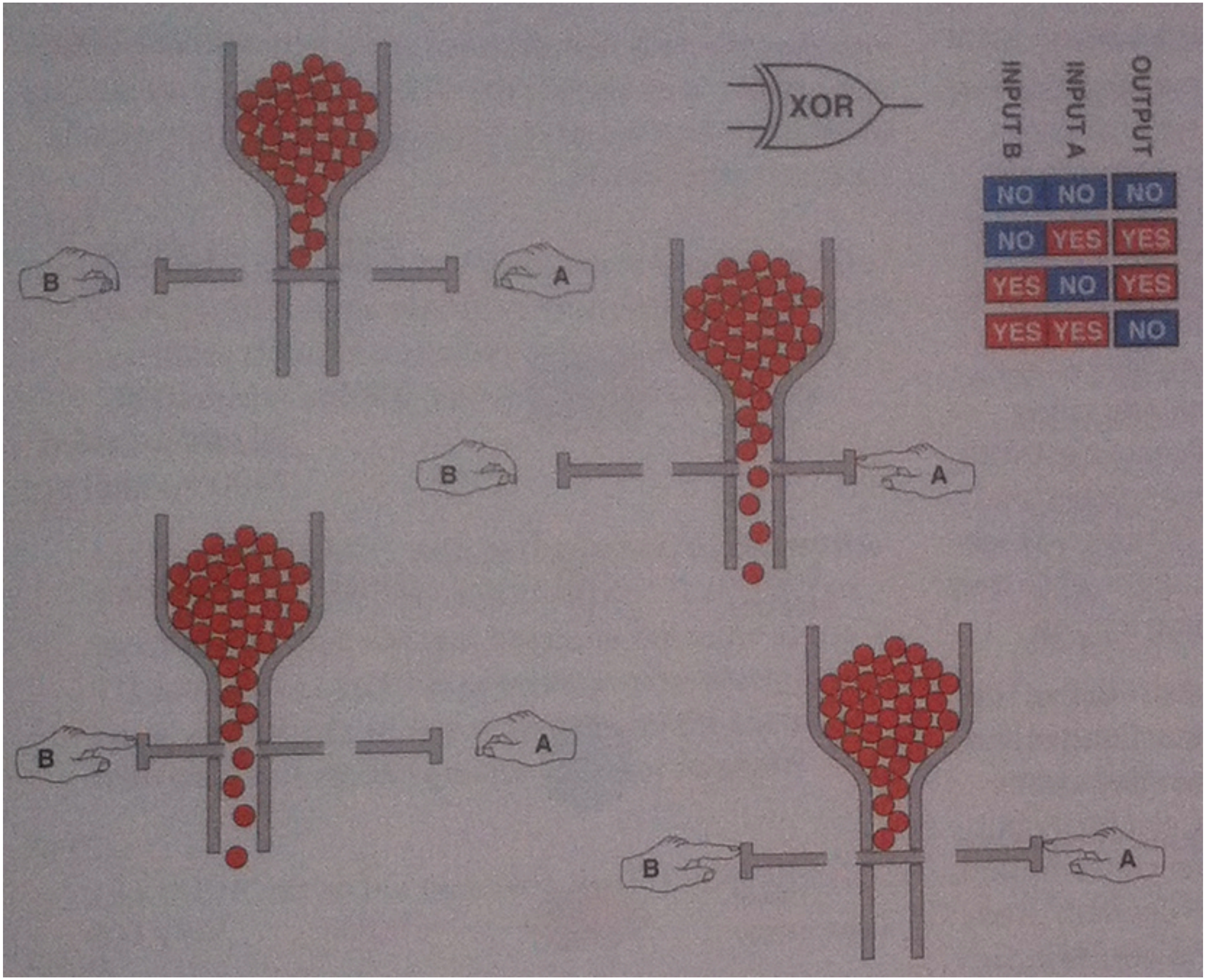



AND
INPUT B
INPUT A
OUTPUT
NO NO NO
NO YES NO
YES NO NO
YES YES YES
A
B
A
B
A
B
A
B

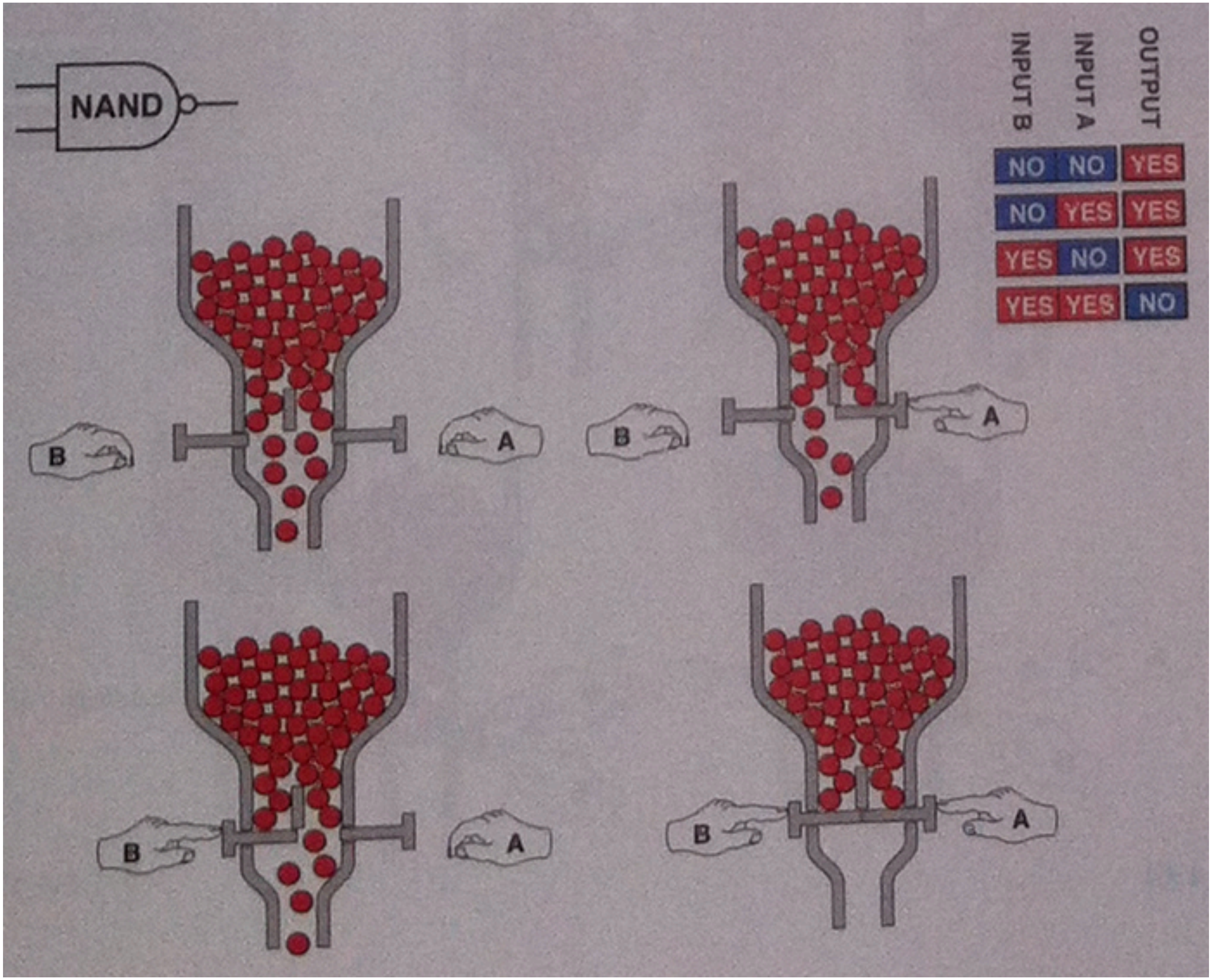
[Platt 2009]


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

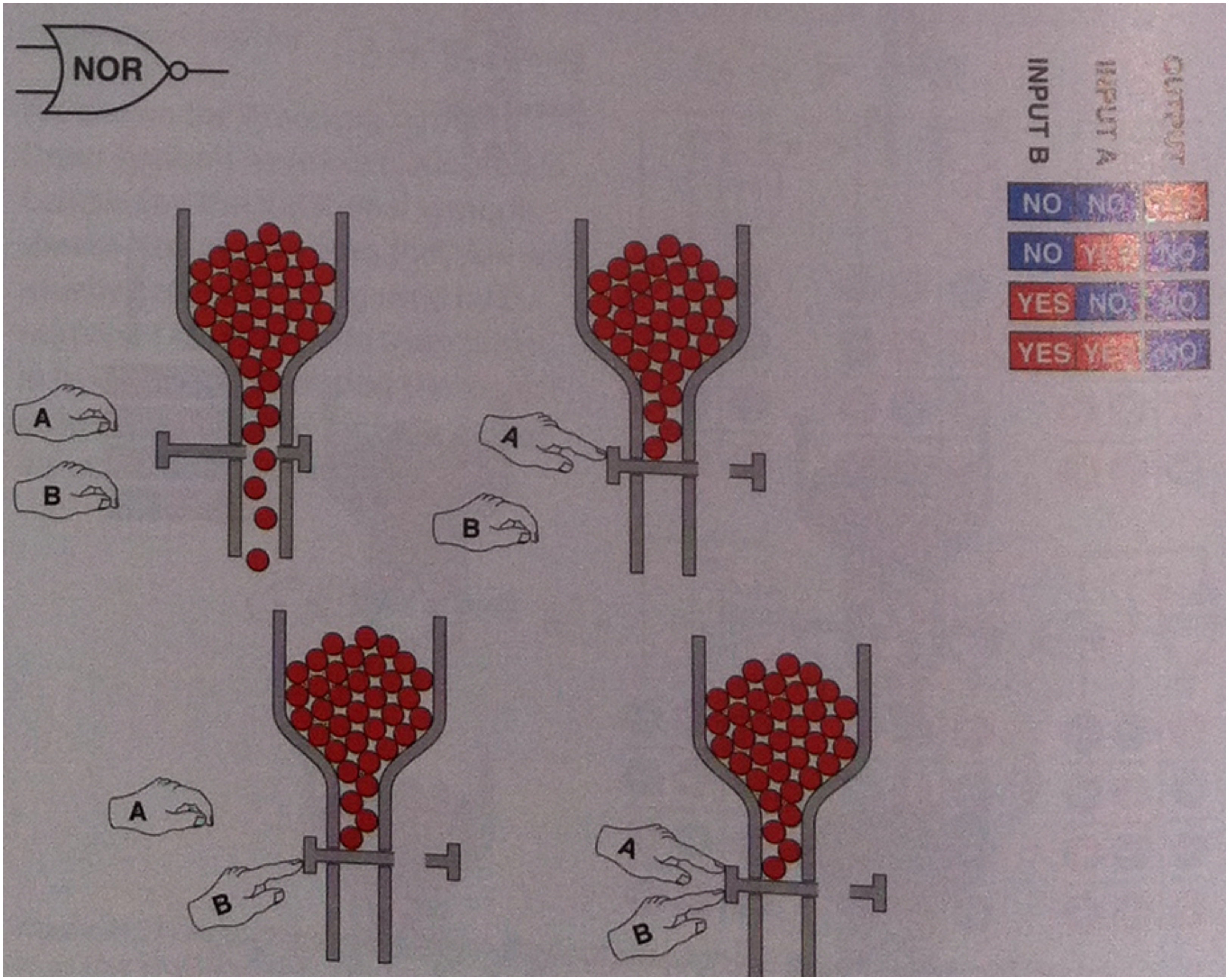
[Platt 2009]


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

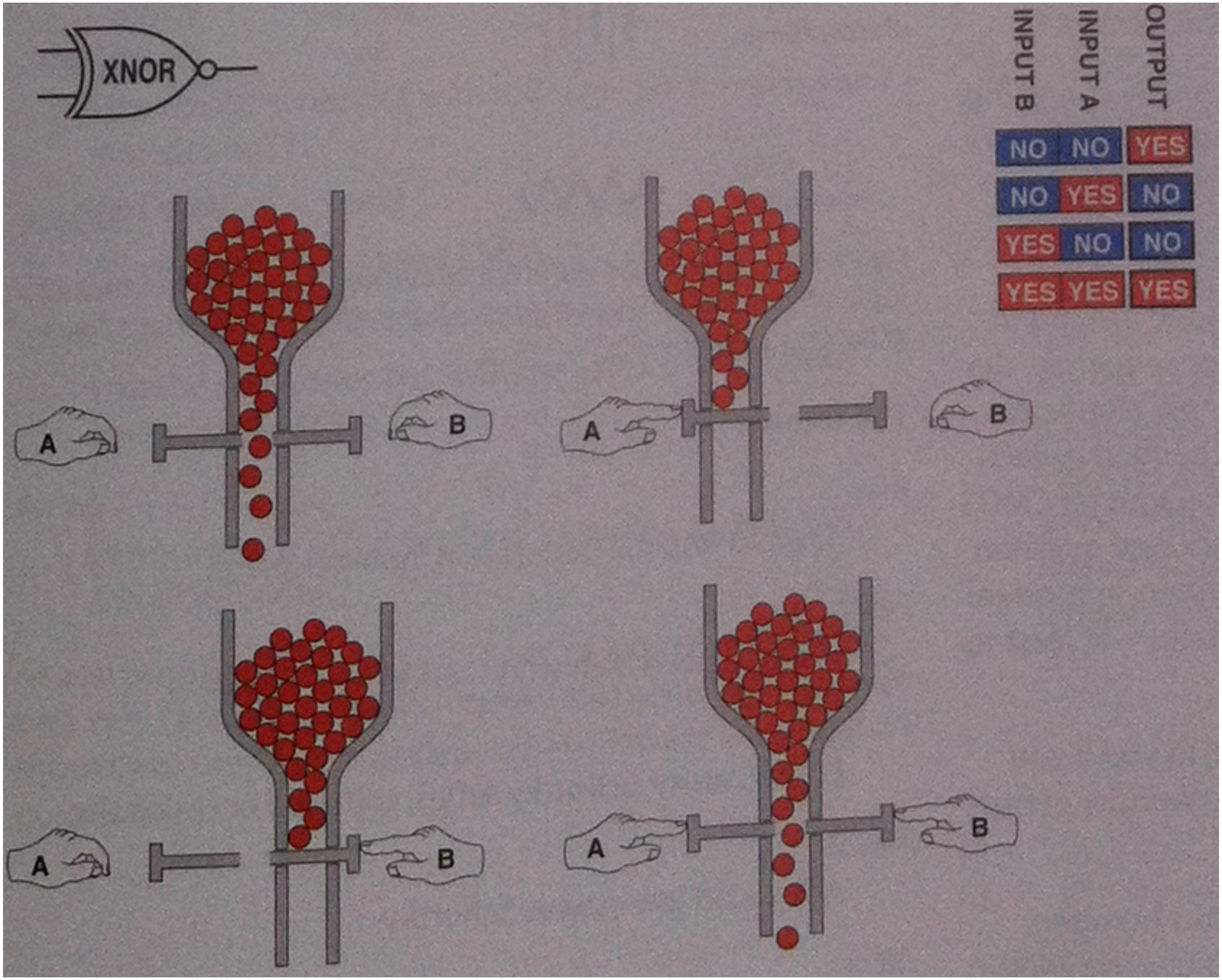
[Platt 2009]


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

[Platt 2009]

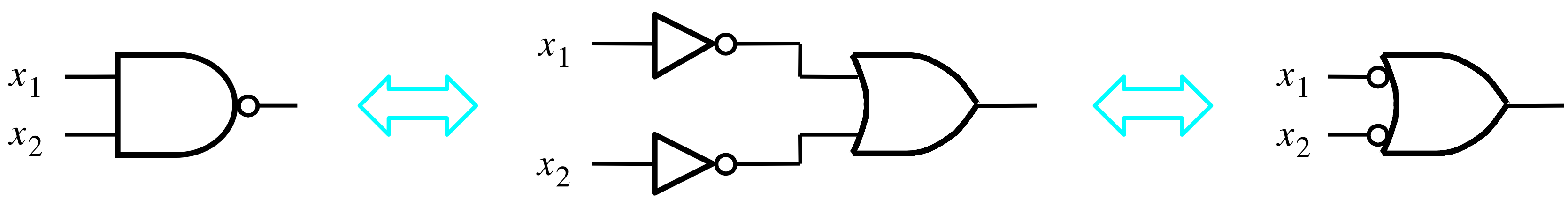


[Platt 2009]


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

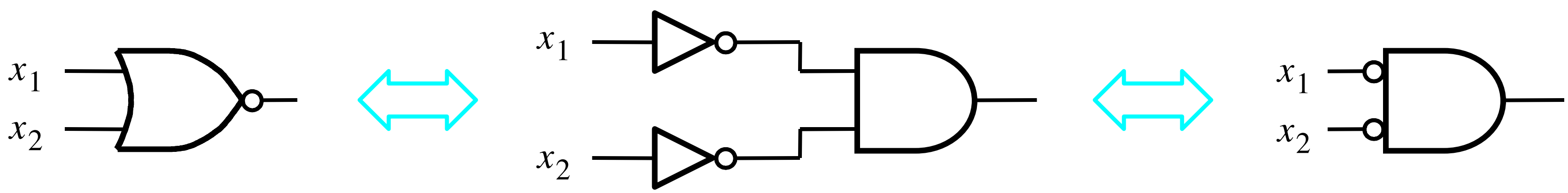
[Platt 2009]

DeMorgan's theorem in terms of logic gates



(a) $\overline{x_1 x_2} = \bar{x}_1 + \bar{x}_2$

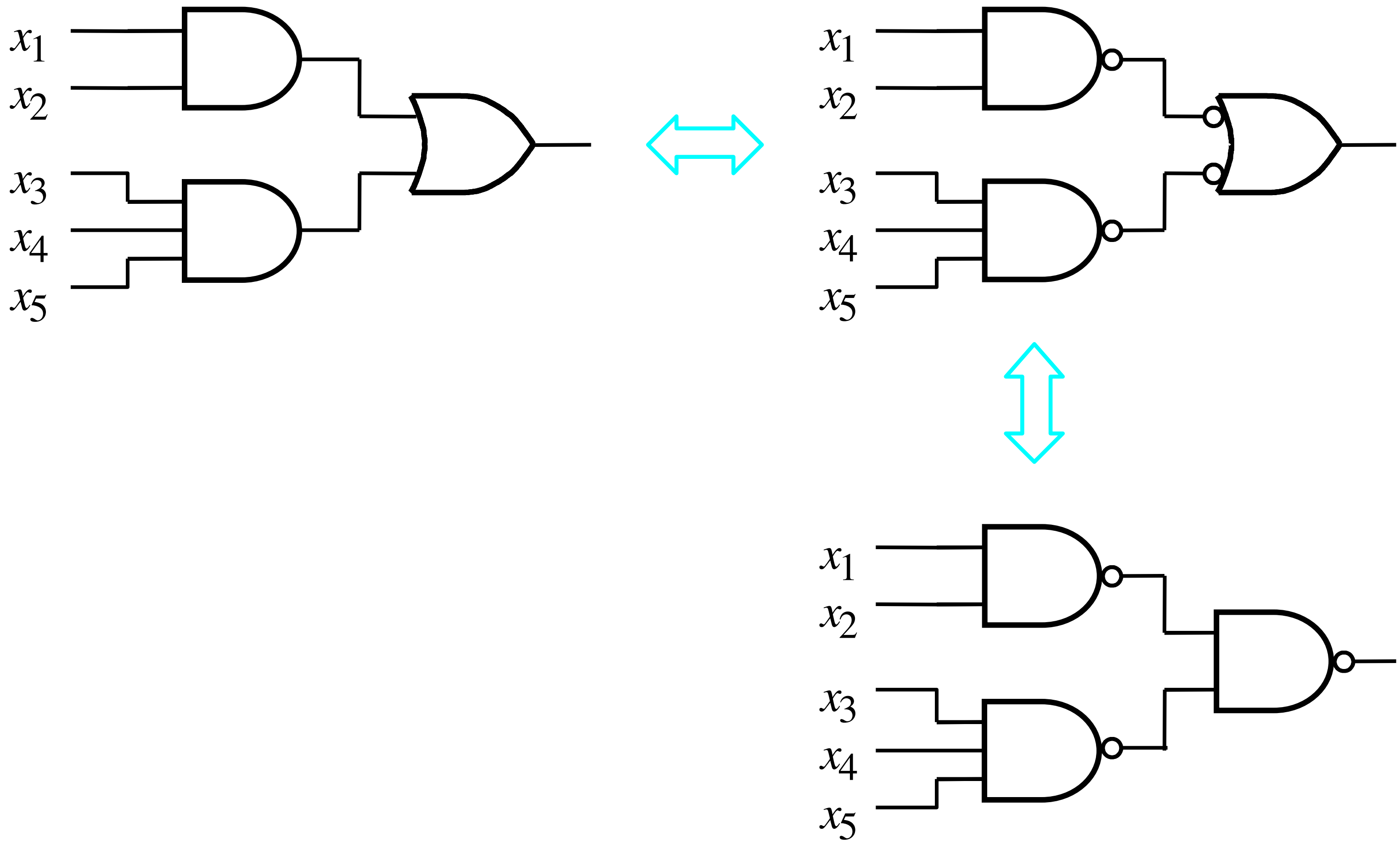
DeMorgan’s theorem in terms of logic gates



(b) $\overline{x_1 + x_2} = \bar{x}_1 \bar{x}_2$

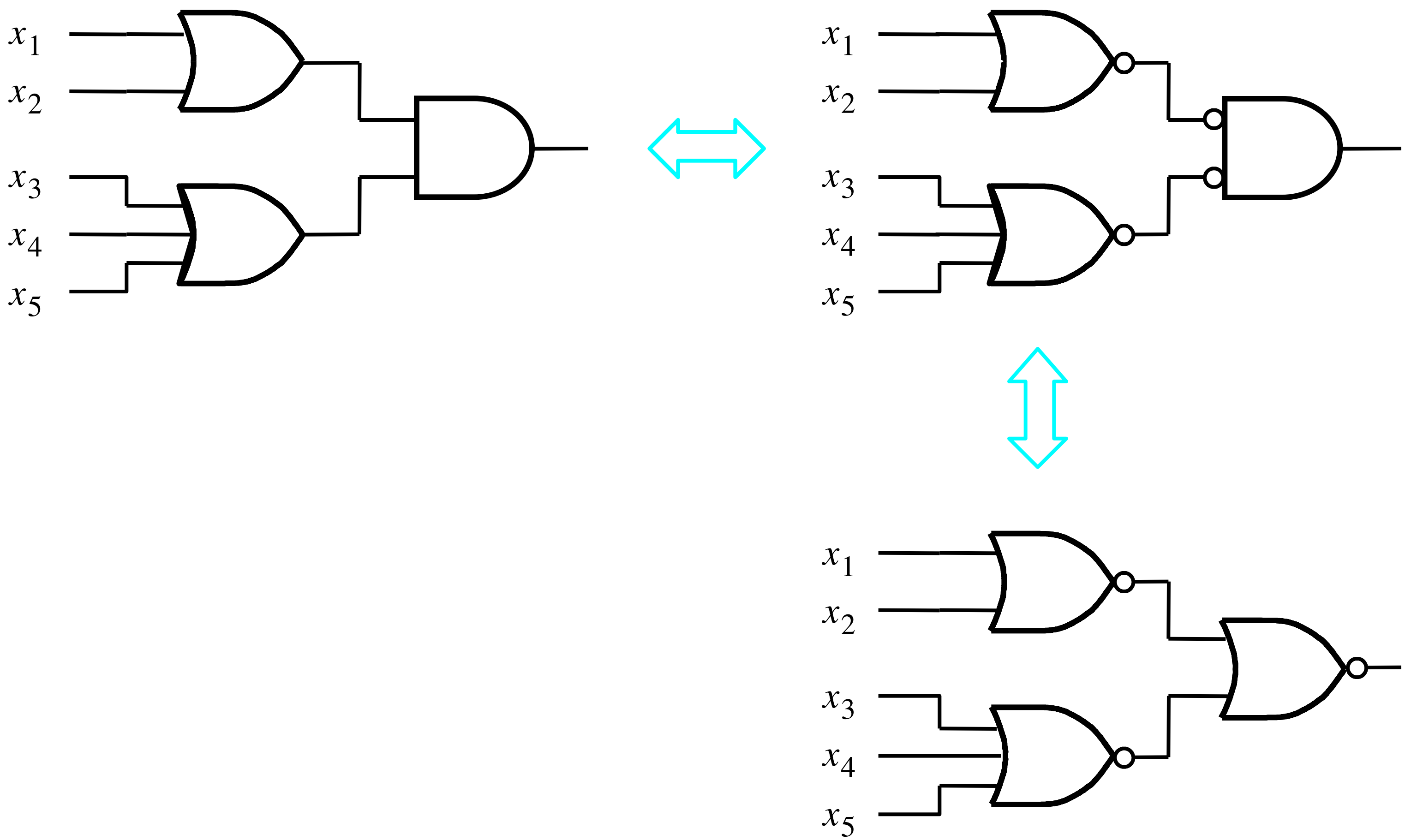
Function Synthesis

Using NAND gates to implement a sum-of-products



[Figure 2.27 from the textbook]

Using NOR gates to implement a product-of sums



[Figure 2.28 from the textbook]

Example 2.13

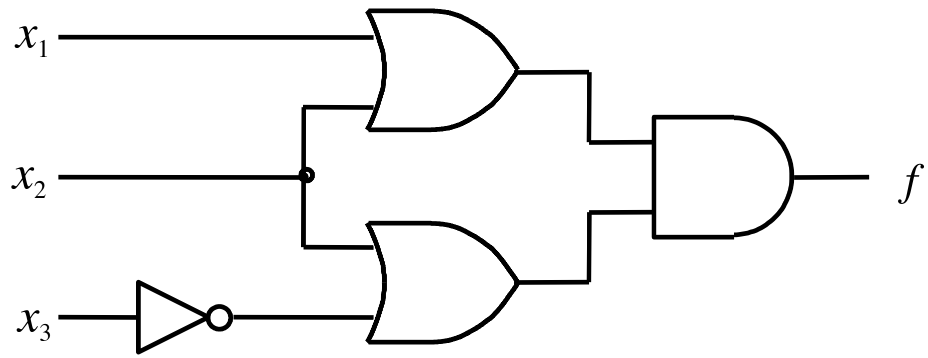
Implement the function $f(x_1, x_2, x_3) = \Sigma\, m(2, 3, 4, 6, 7)$ using only NOR gates.

Example 2.13

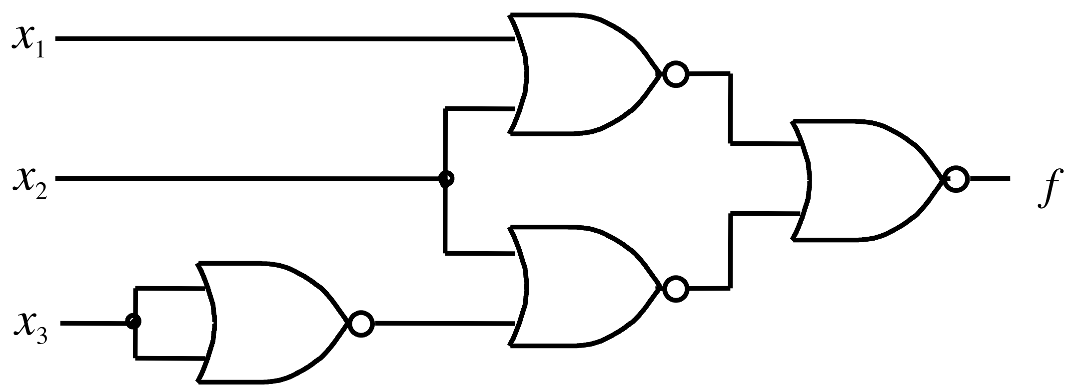
Implement the function $f(x_1, x_2, x_3) = \Sigma\, m(2, 3, 4, 6, 7)$ using only NOR gates.

The POS expression is: $f = (x_1 + x_2)\,(x_2 + \overline{x_3})$

NOR-gate realization of the function



(a) POS implementation



(b) NOR implementation

[Figure 2.29 from the textbook]

Example 2.14

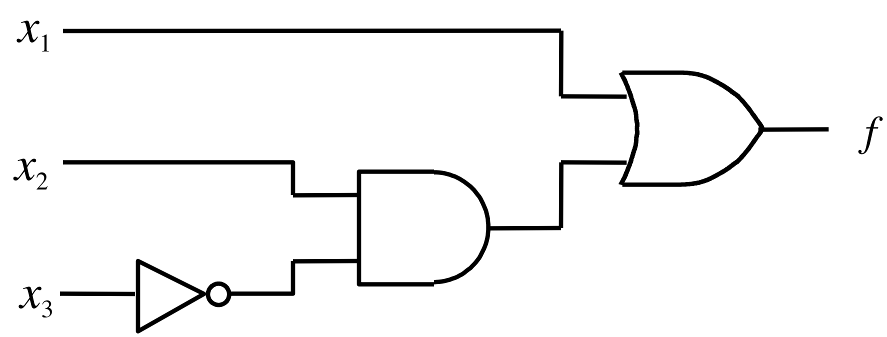
Implement the function $f(x_1, x_2, x_3) = \Sigma\, m(2, 3, 4, 6, 7)$ using only NAND gates.

Example 2.14

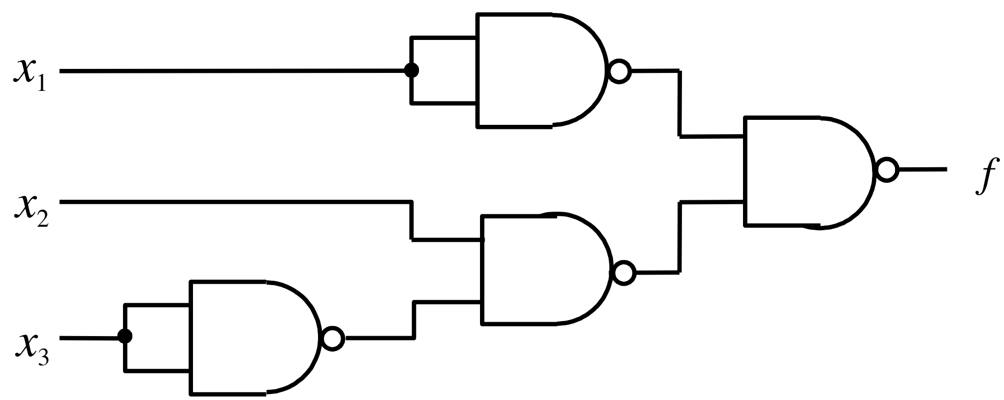
Implement the function $f(x_1, x_2, x_3) = \Sigma\, m(2, 3, 4, 6, 7)$ using only NAND gates.

The SOP expression is: $f = x_2 + x_1\overline{x}_3$

NAND-gate realization of the function



(a) SOP implementation



(b) NAND implementation

[Figure 2.30 from the textbook]

Questions?

THE END