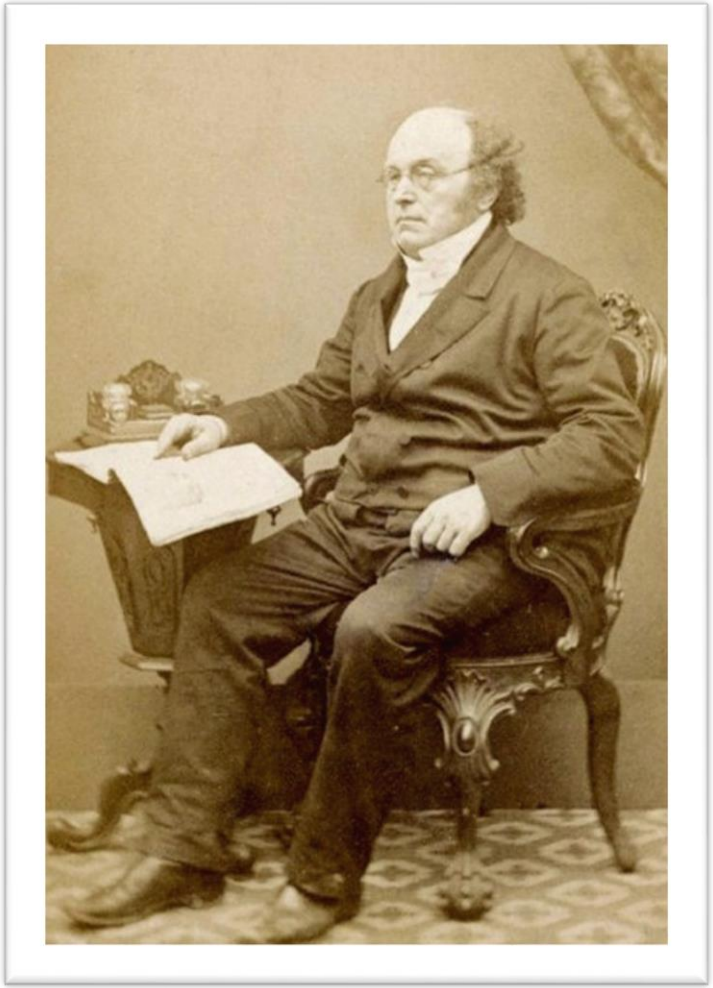
Binary Logic

DeMorgan Theorems

Augustus DeMorgan (1806-1871)



British mathematician born in India

https://en.wikipedia.org/wiki/Augustus_De_Morgan

DeMorgan's Theorems

- Are used to simplify complex negated binary logical expressions.

Example:

- $Z = \overline{ABC + A + B + C + AB + AC + BC}$

DeMorgan's Theorems

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

First DeMorgan's Theorem

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

Second DeMorgan's Theorem

Proof of the DeMorgan's theorems

Proof of the DeMorgan's theorem

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

x	y	$\overline{x}$	$\overline{y}$	$x\,y$	$\overline{x\,y}$	$\overline{x}+\overline{y}$	$\overline{x}\ \overline{y}$	$x + y$	$\overline{x + y}$
0	0								
0	1								
1	0								
1	1								

Proof of the DeMorgan's theorems

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

x	y	$\overline{x}$	$\overline{y}$	$x\,y$	$\overline{x\,y}$	$\overline{x}+\overline{y}$	$\overline{x}\ \overline{y}$	$x + y$	$\overline{x + y}$
0	0	1	1	0					
0	1	1	0	0					
1	0	0	1	0					
1	1	0	0	1					

Proof of the first DeMorgan's theorem

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

x	y	$\overline{x}$	$\overline{y}$	$x\,y$	$\overline{x\,y}$	$\overline{x}+\overline{y}$	$\overline{x}\ \overline{y}$	$x + y$	$\overline{x + y}$
0	0	1	1	0	1	1			
0	1	1	0	0	1	1			
1	0	0	1	0	1	1			
1	1	0	0	1	0	0			

Proof of both DeMorgan's theorems

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

x	y	$\overline{x}$	$\overline{y}$	$x\,y$	$\overline{x\,y}$	$\overline{x}+\overline{y}$	$\overline{x}\;\overline{y}$	$x + y$	$\overline{x + y}$
0	0	1	1	0	1	1	1	0	1
0	1	1	0	0	1	1	0	1	0
1	0	0	1	0	1	1	0	1	0
1	1	0	0	1	0	0	0	1	0

Perfect Induction

Example: Using DeMorgan's

$$\overline{A \bullet \overline{B} + C}$$

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

First application of the theorem

$$\overline{A \cdot \overline{B} + C} = \overline{A \cdot \overline{B}} \cdot \overline{C}$$

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

First application of the theorem

$$\overline{A \cdot \overline{B} + C} = \boxed{\overline{A \cdot \overline{B}} \cdot \boxed{\overline{C}}}$$

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

Second application of theorem

$$\overline{A \cdot \overline{B} + C} = \boxed{\overline{A \cdot \overline{B}}} \cdot \overline{C}$$

$$= \boxed{(\overline{A} + \overline{\overline{B}})} \cdot \overline{C}$$

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

Distribute…

$$\overline{A \cdot \overline{B} + C} = \overline{A \cdot \overline{B}} \cdot \overline{C}$$

$$= (\overline{A} + \overline{\overline{B}}) \cdot \overline{C}$$

$$= (\overline{A} + B) \cdot \overline{C}$$

$$= \overline{A} \cdot \overline{C} + B \cdot \overline{C}$$

Sum-of-products (SOP) form

$$\overline{A \cdot \overline{B} + C} = \overline{A \cdot \overline{B}} \cdot \overline{C}$$

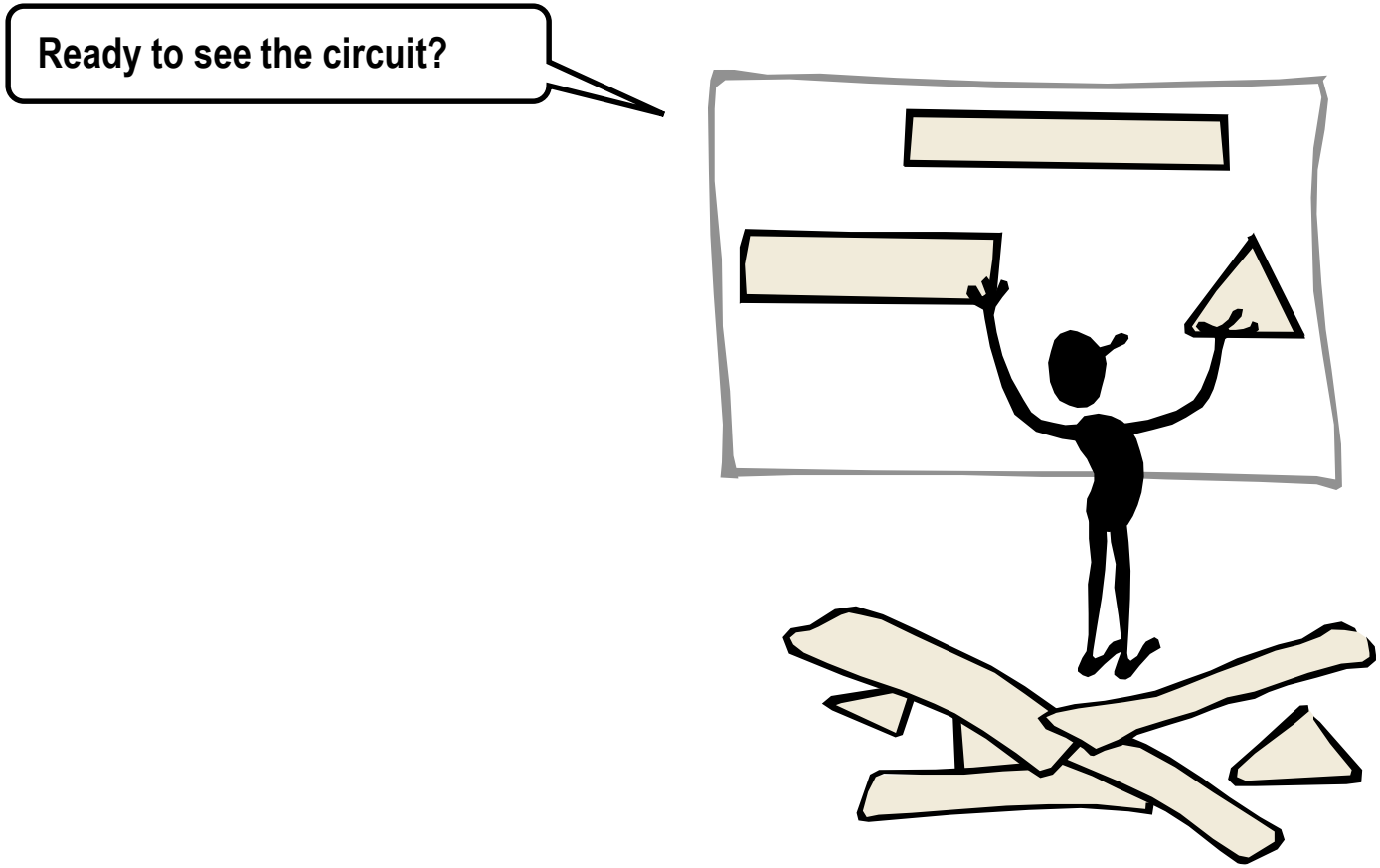
$$= (\overline{A} + \overline{\overline{B}}) \cdot \overline{C}$$

$$= (\overline{A} + B) \cdot \overline{C}$$

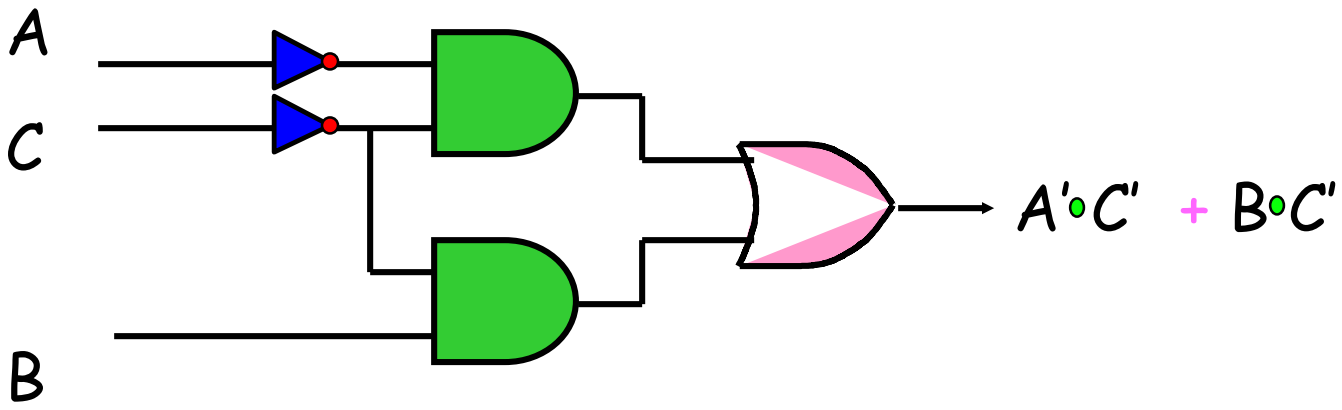
$$= \overline{A} \cdot \overline{C} + B \cdot \overline{C}$$

Implement with gates

Sum-of-Products (SOP) form

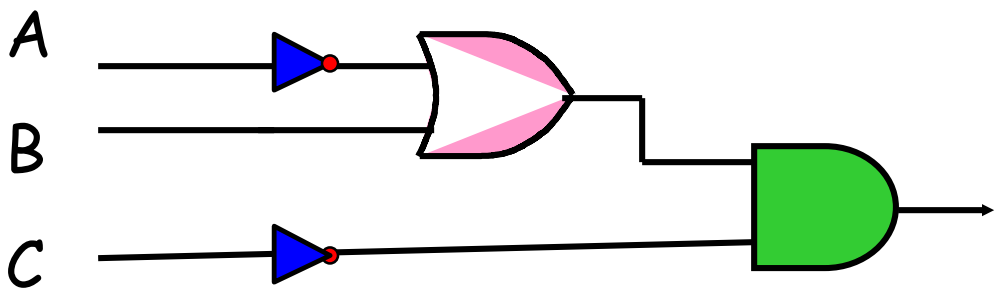


The SOP leads to “two-level-realization”



factor … “multi-level-realization”

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



More gates….

More Gates

- NOR (Not OR)
- NAND (Not AND)

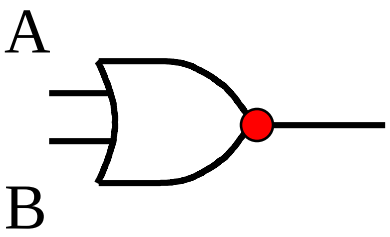
OR

A	B	OR	NOR
0	0	0	
0	1	1	
1	0	1	
1	1	1	

NOR

A	B	OR	NOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

NOR



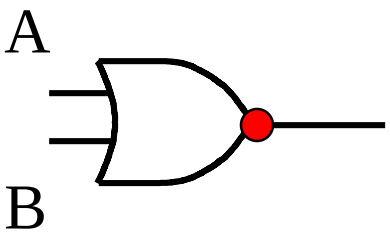
$$\overline{A + B}$$

Using DeMorgan's theorem

A	B	OR	NOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

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

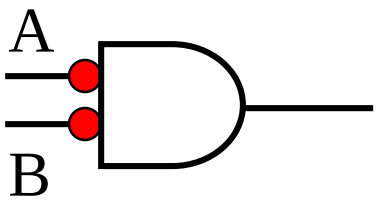
NOR



$$\overline{A + B}$$

Using DeMorgan's theorem

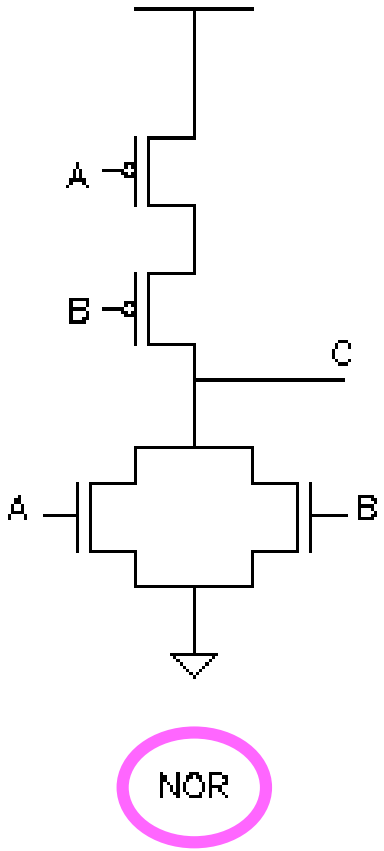
A	B	OR	NOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0



$$\overline{A} \bullet \overline{B}$$

$$\overline{A} \bullet \overline{B} = \overline{A + B}$$

NOR: CMOS and gate layout



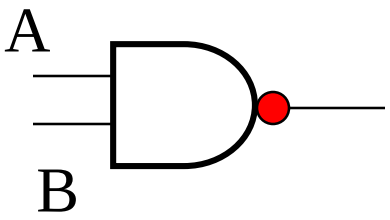
AND … Not AND

A	B	AND	NAND
0	0	0	
0	1	0	
1	0	0	
1	1	1	

NAND

A	B	AND	NAND
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

NAND



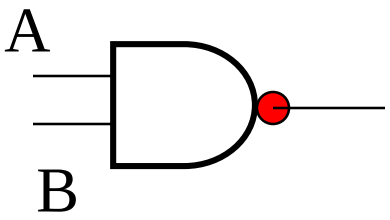
$\overline{A \bullet B}$

Using DeMorgan's theorem

A	B	AND	NAND
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

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

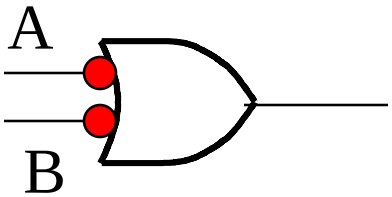
NAND



$$\overline{A \bullet B}$$

Using DeMorgan's theorem

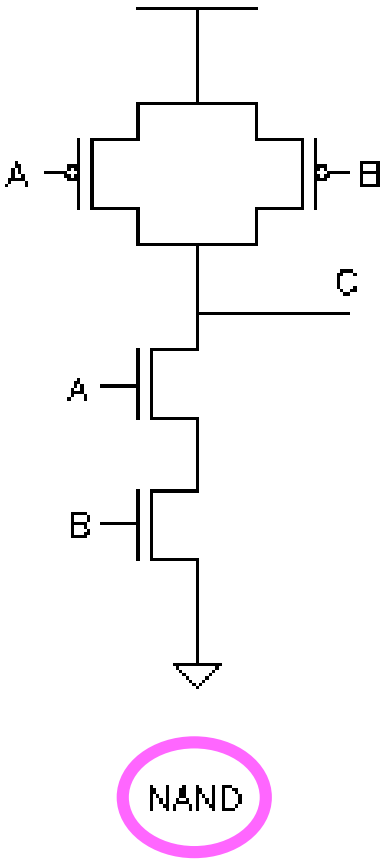
A	B	AND	NAND
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

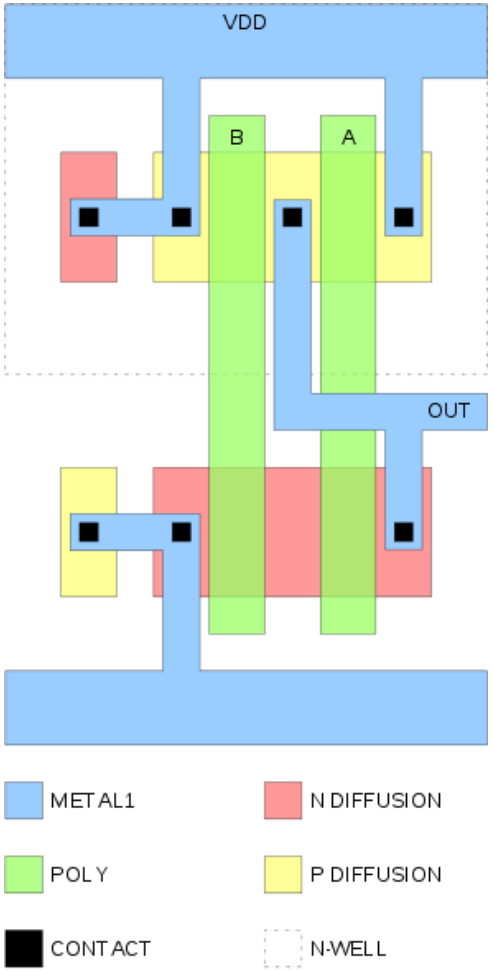


$$\overline{A} + \overline{B}$$

$$\overline{A \bullet B} = \overline{A} + \overline{B}$$

NAND: CMOS and gate layout



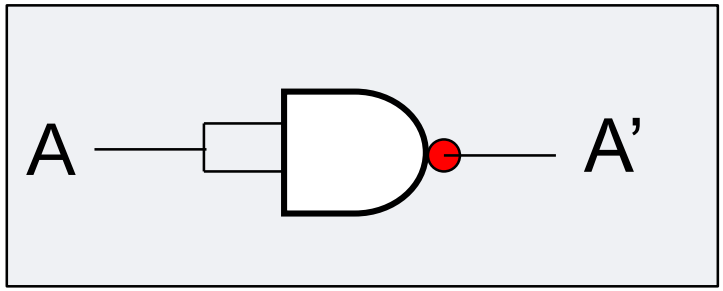


Universality of NAND and NOR Gates

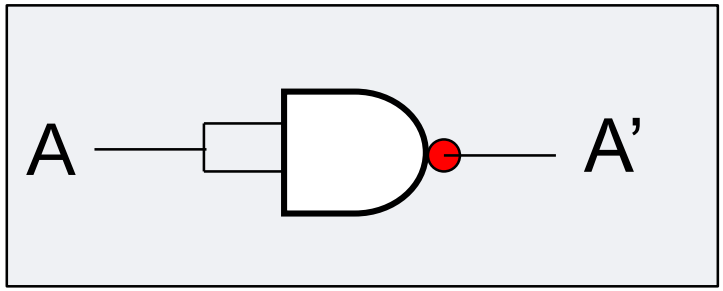
NAND and NOR gates …

- Can implement any Boolean expression
- Can simulate all three basic gates (AND, OR, NOT)

Universality of NAND: NOT

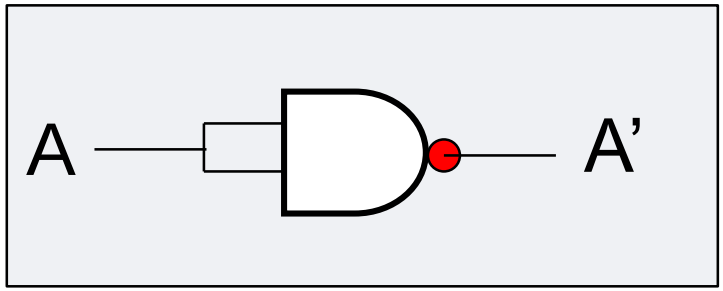


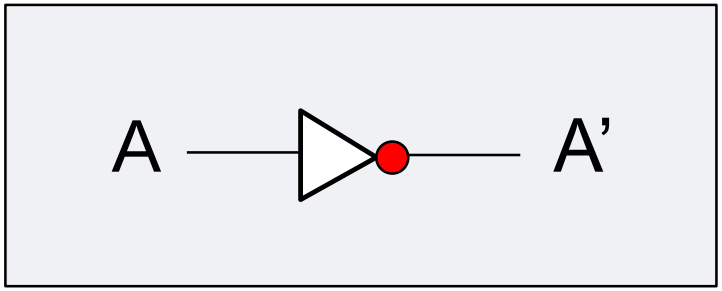
Proof



(AA)’ = A’

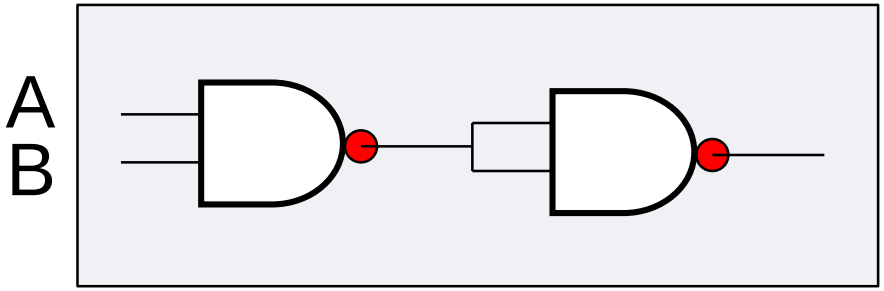
Proof



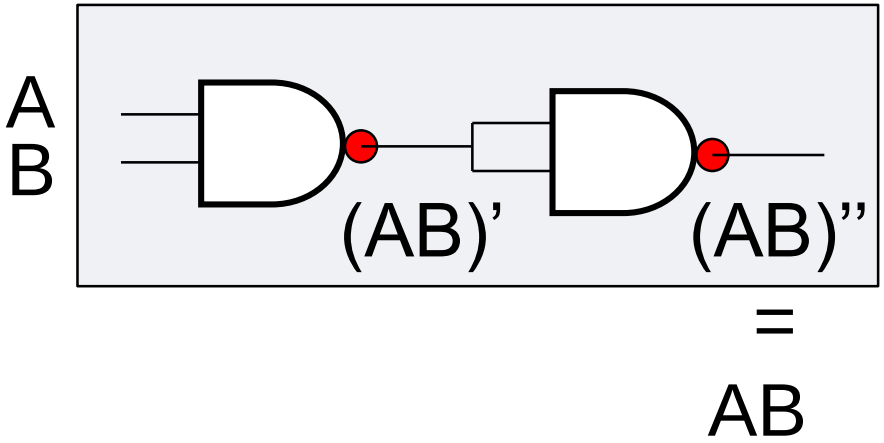


(AA)’ = A’

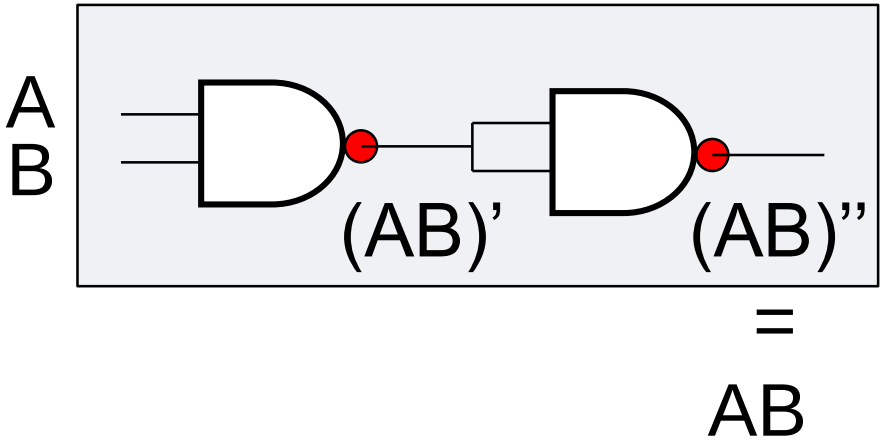
Universality of NAND: AND

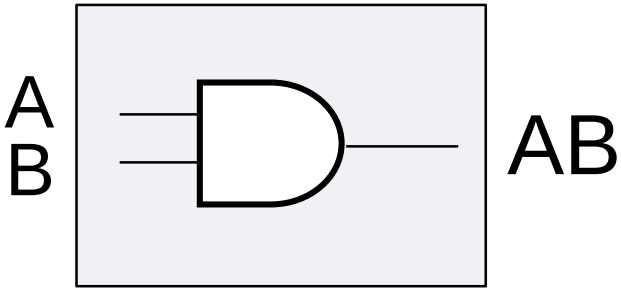


Proof

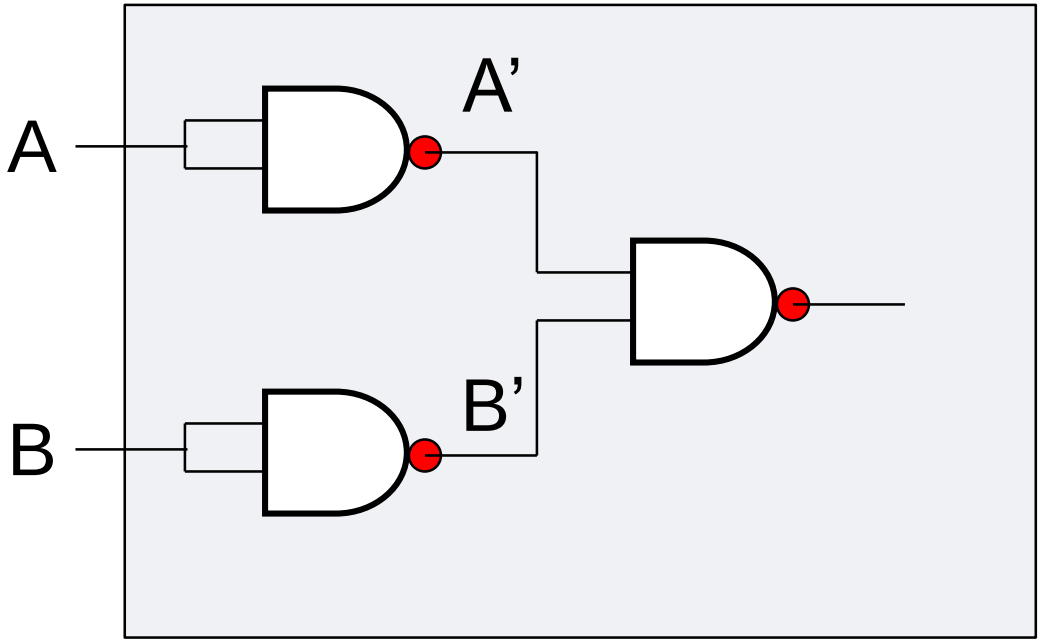


Proof

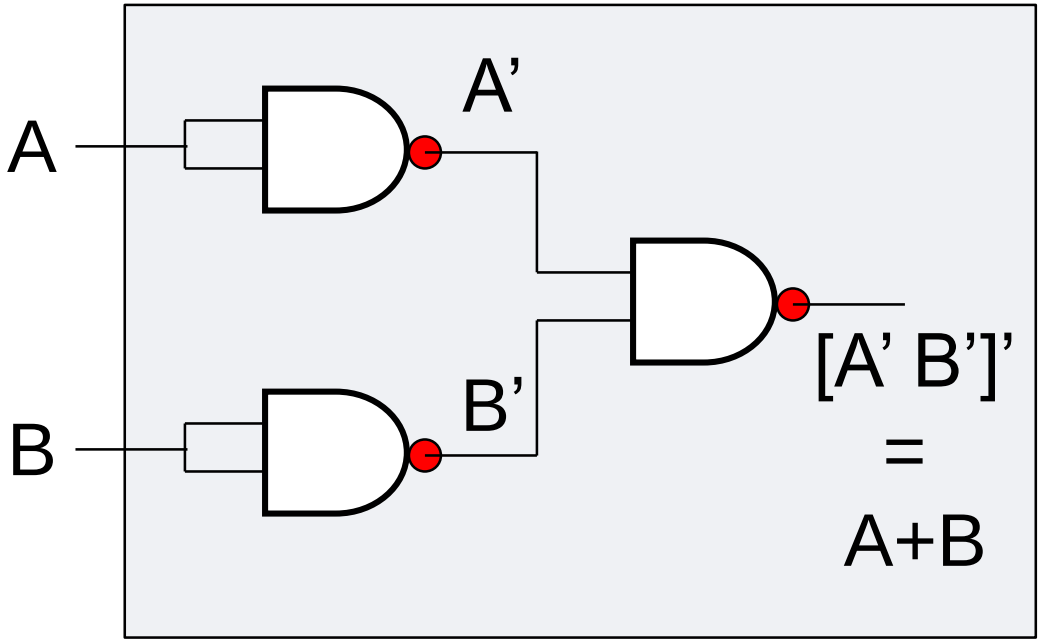




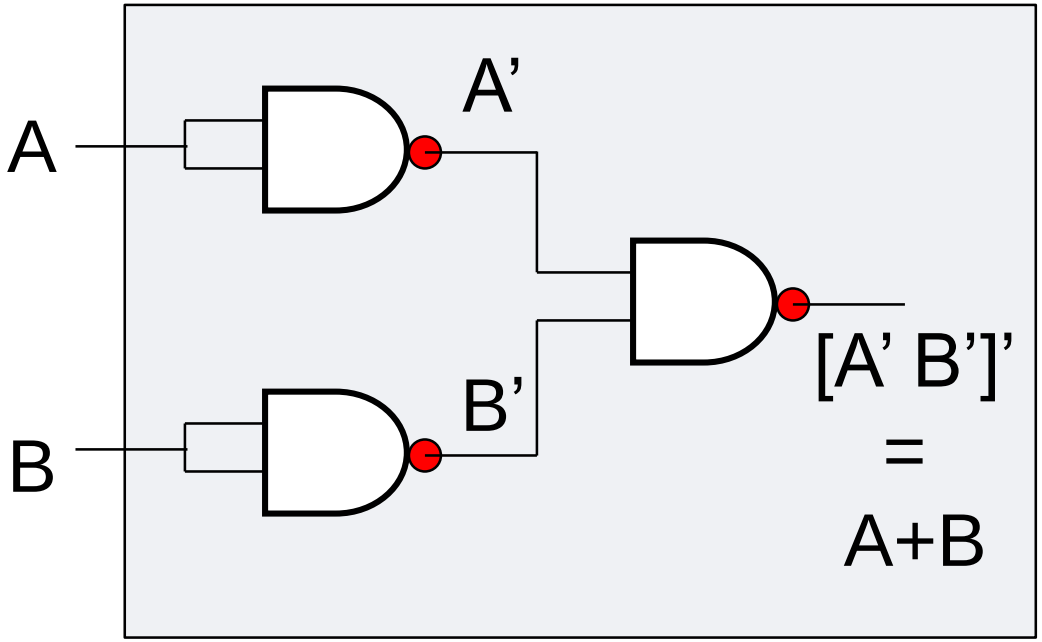
Universality of NAND: OR

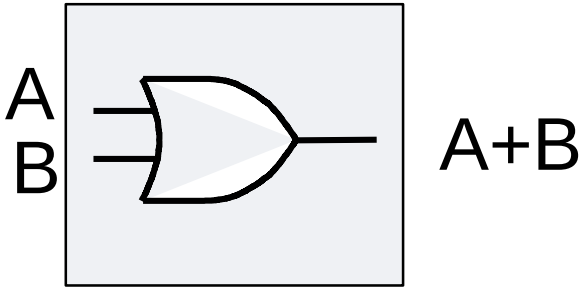


Universality of NAND: OR

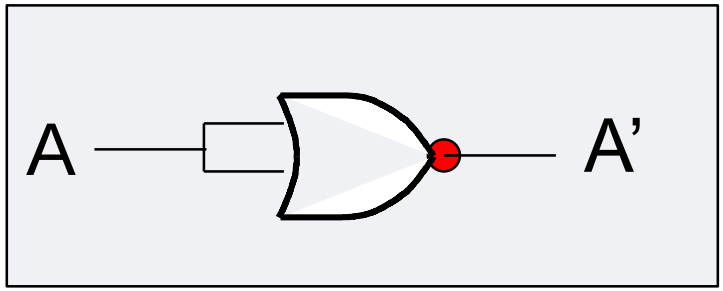


Universality of NAND: OR

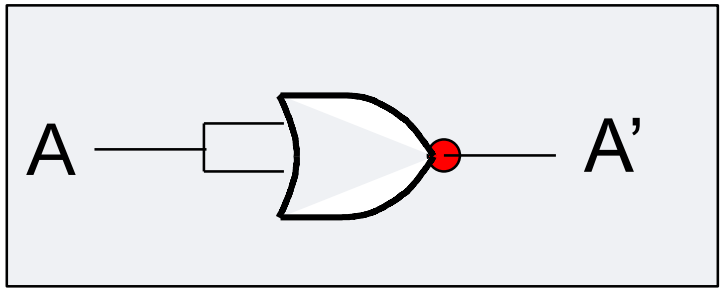




Universality of NOR: NOT

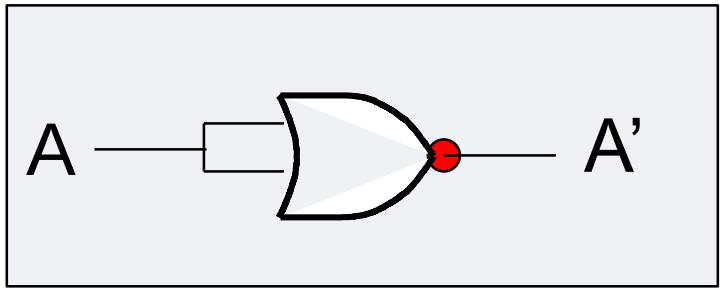


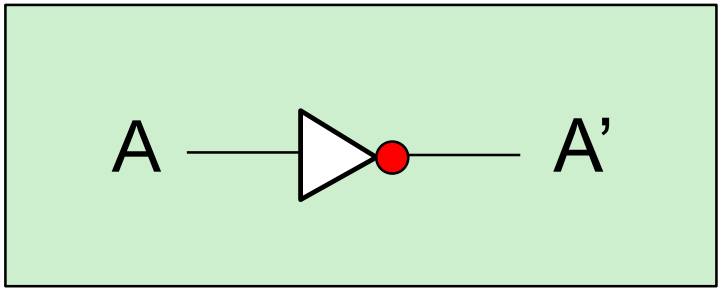
Proof



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

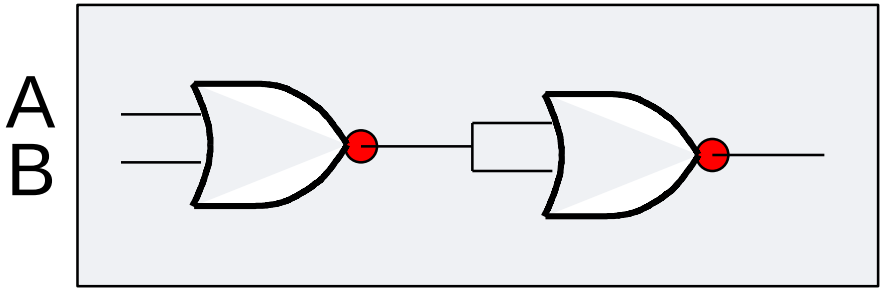
Proof



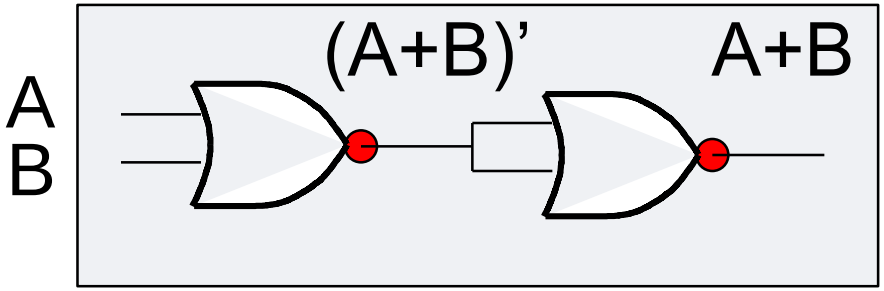


$$(A+A)' = A'$$

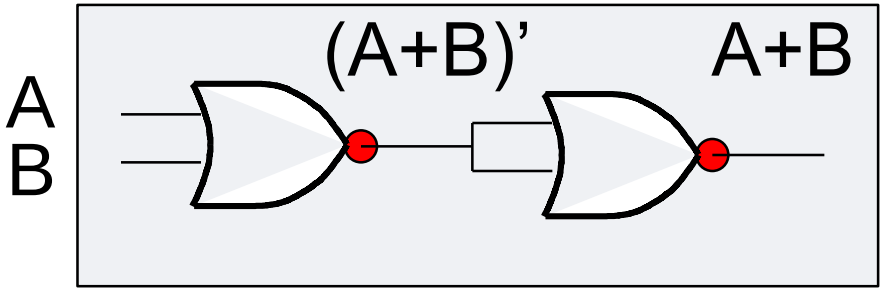
Universality of NOR: OR

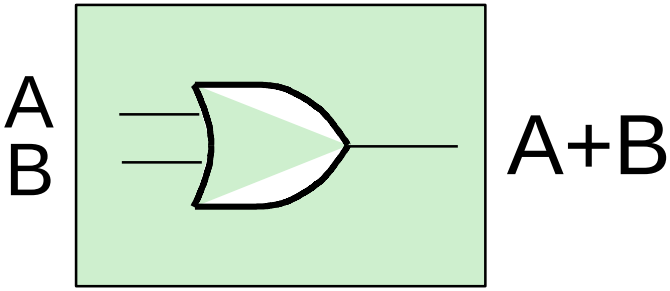


Proof

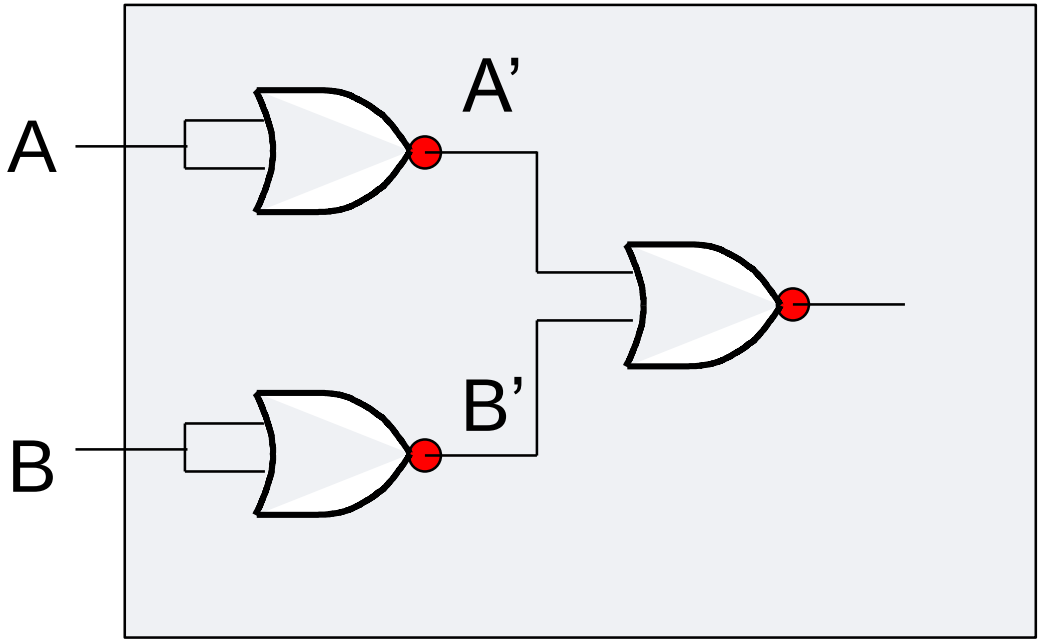


Proof

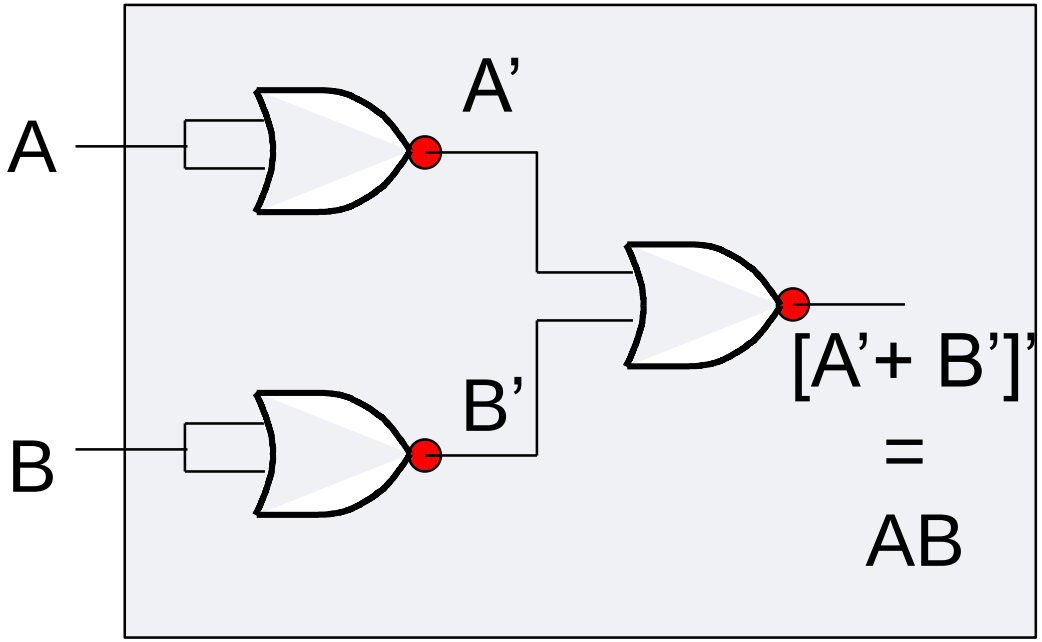




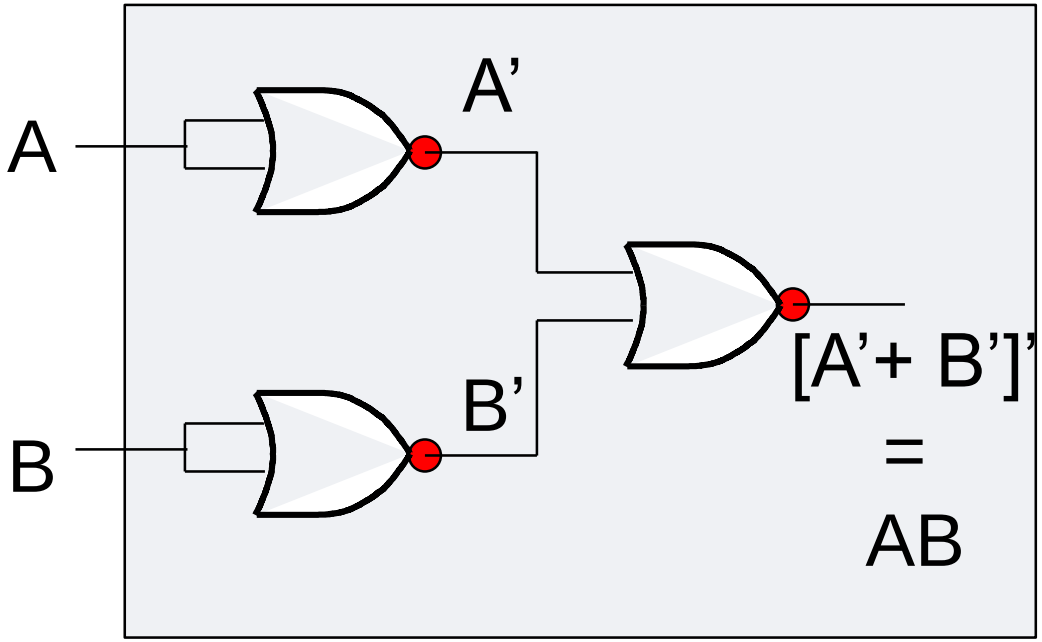
Universality of NOR: AND

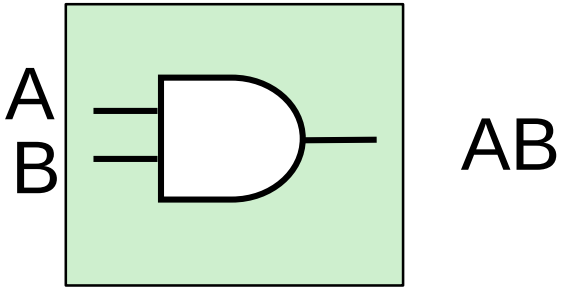


Proof



Proof





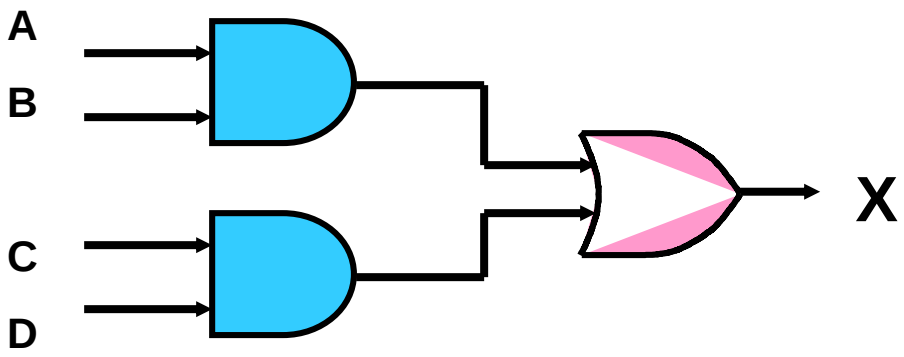
Example

Implement the Boolean function: X = AB + CD

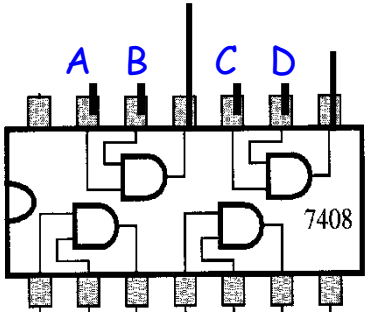
1. Using: AND,OR,NOT gates
2. Using: NAND gates

You have … 5 minutes …

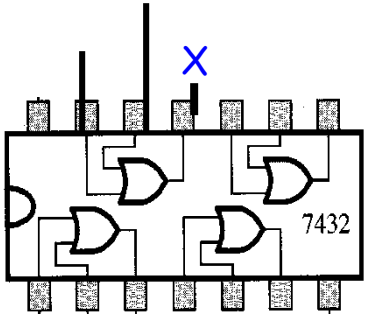
1) X = AB + CD; Using AND, OR, NOT gates



X = AB + CD; Using Chips (7408-7432)

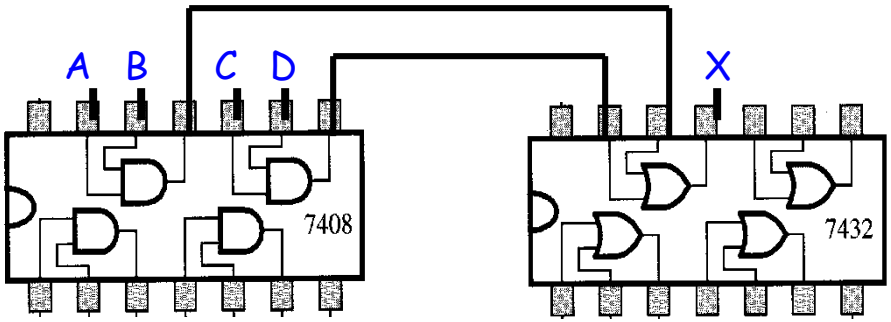


AND gates chip



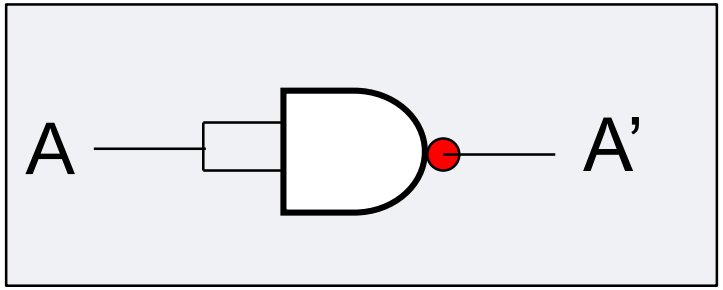
OR gates chip

2 Chips are used

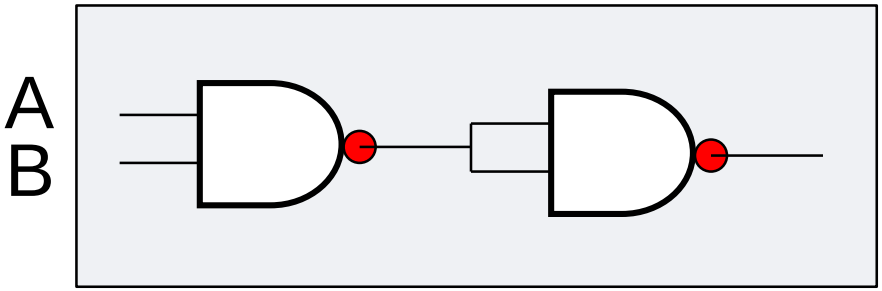


We use 2 different chips

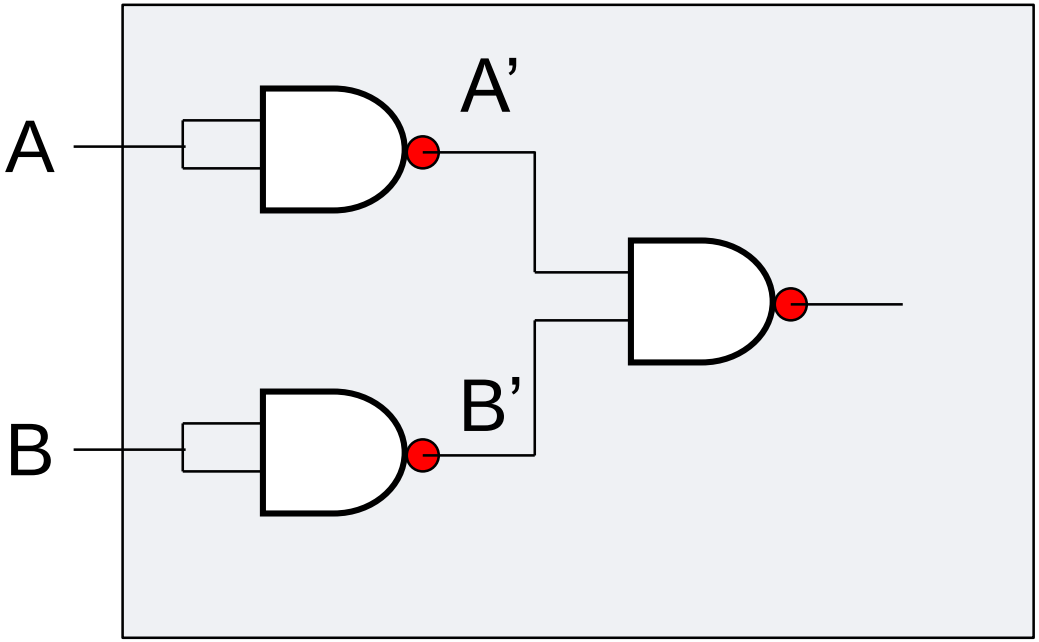
NAND gates



NOT

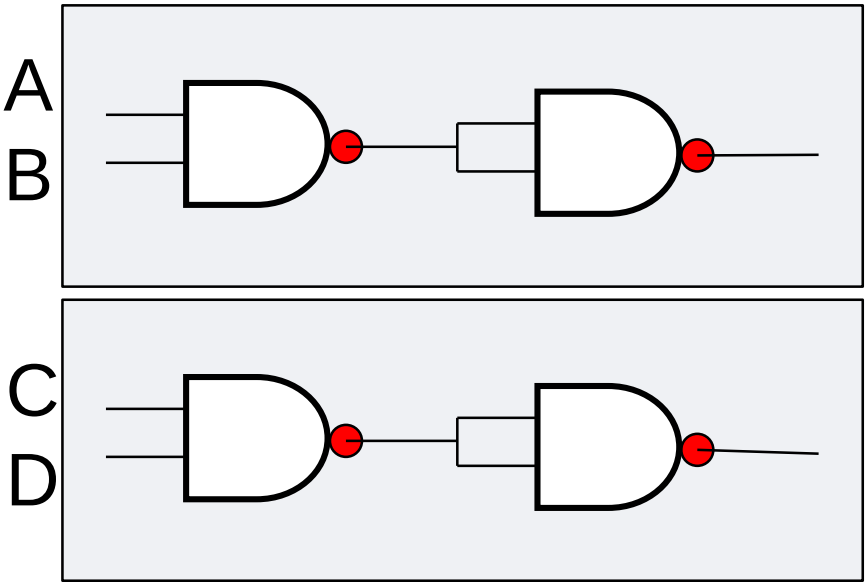


AND

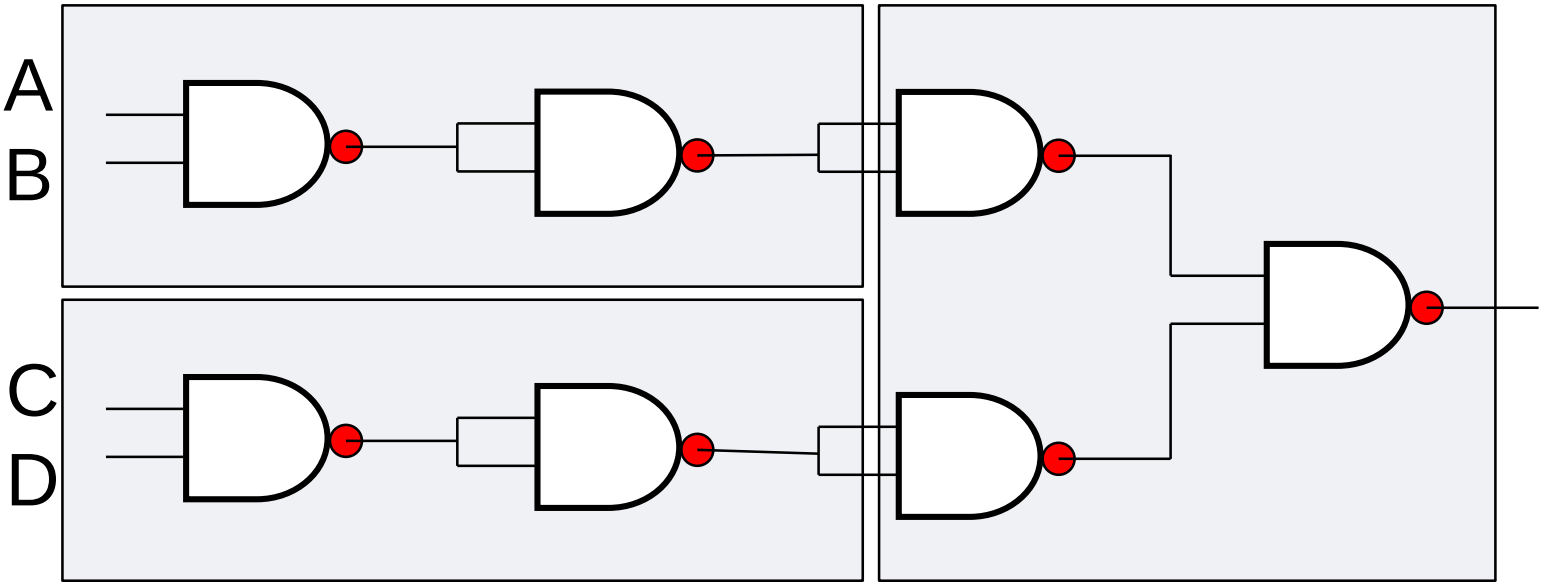


OR

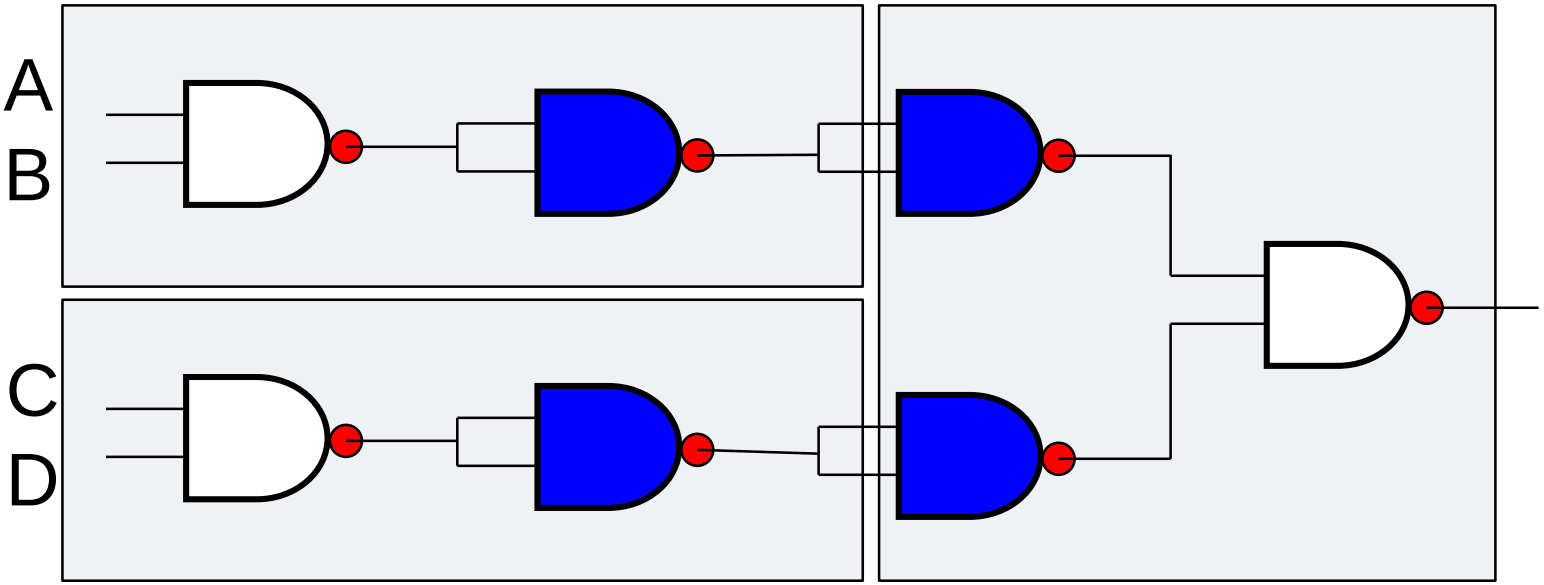
2) X = AB + CD; Using NAND gates



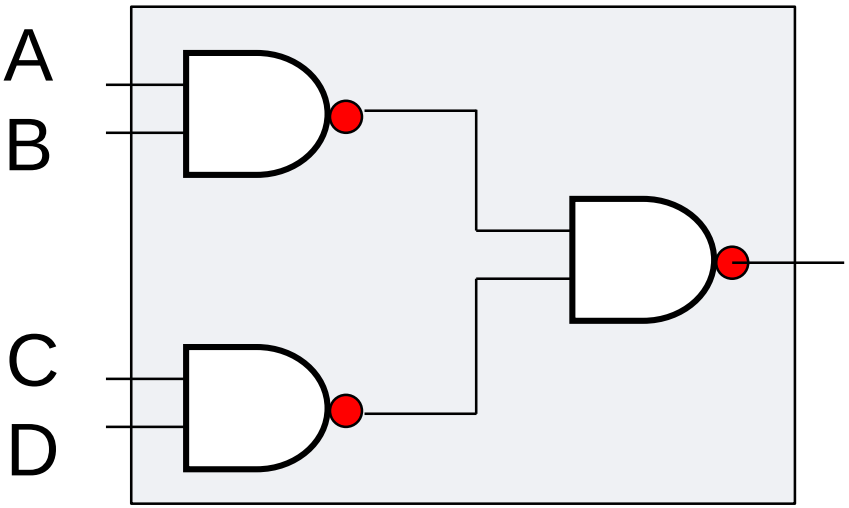
X = AB + CD; Using NAND gates



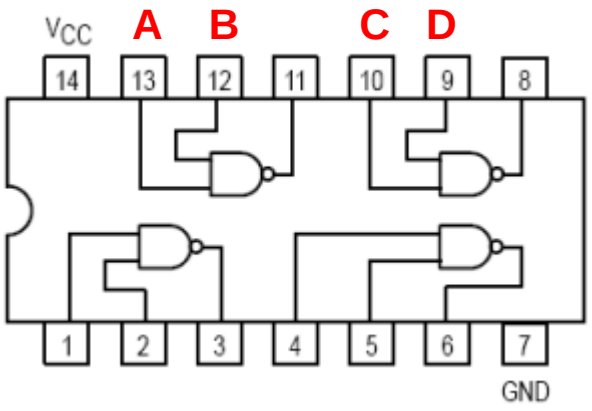
Two inverts cancel each other



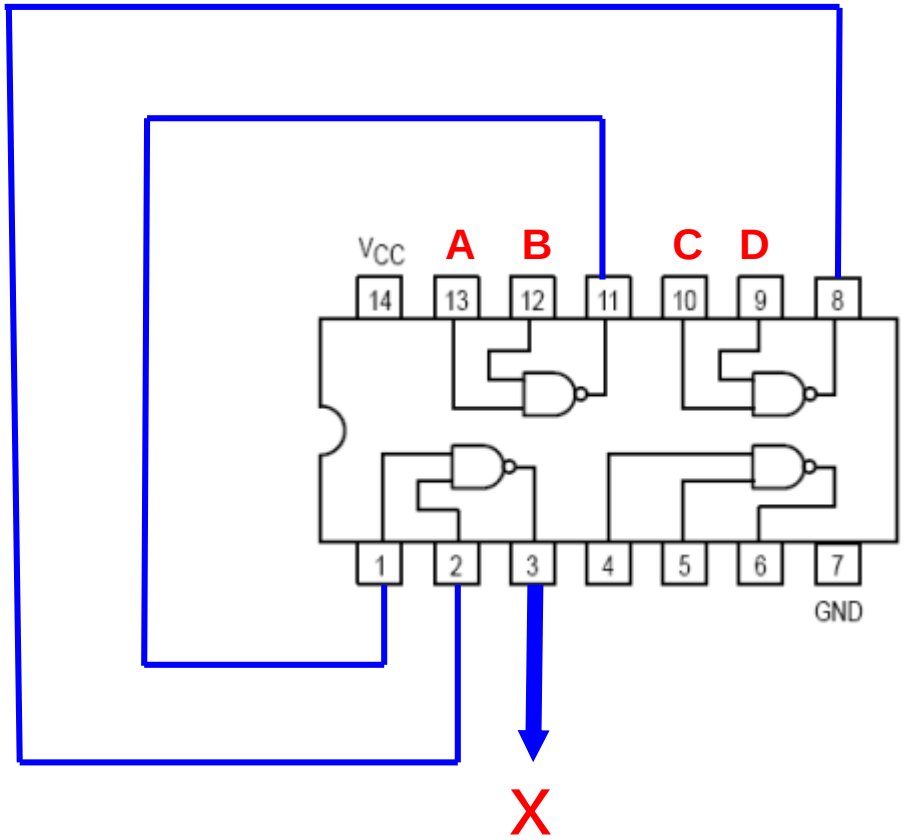
X = AB + CD; Using NAND gates



X = AB + CD; Using NAND Chip (74LS00)



X = AB + CD; Using NAND Chip (74LS00)



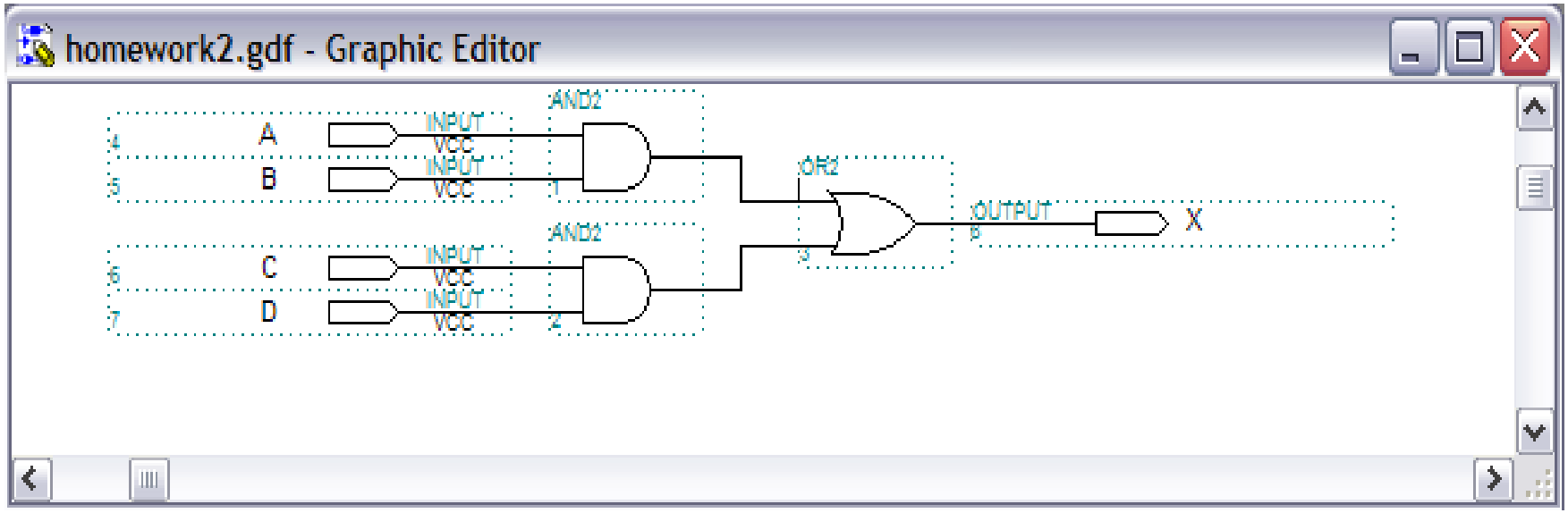
We only use 1 chip

NAND technology allows us to use less chips

Why today’s technology uses NAND gates?

- Reduces the integrated circuit complexity since NAND gates can be implemented with less transistors than the basic AND/OR gates
- Increases integrated circuit's speed
- Minimizes:
  - Production chip cost
  - Packing density of the chip.

X = AB + CD; Using VHDL



VHDL Code: X = AB + CD

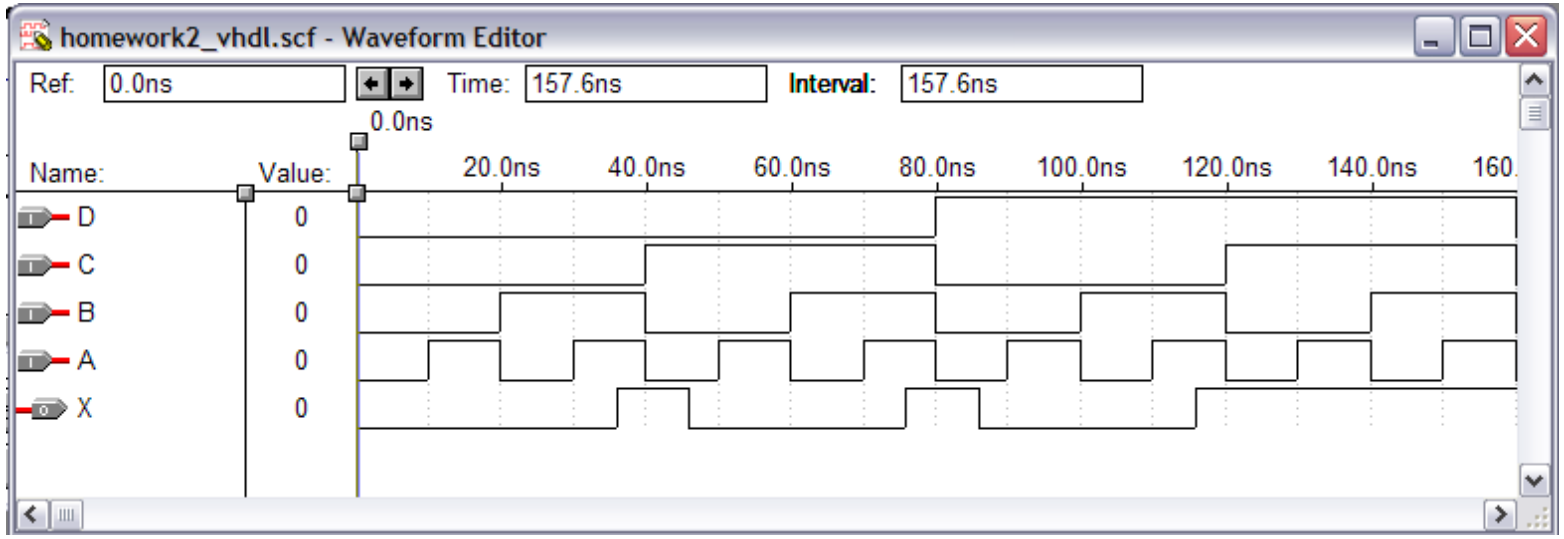
homework2_vhdl.vhd - Text Editor

```
entity homework2_vhdl is
    port (a,b,c,d: in bit;
          x:  out bit);
end homework2_vhdl;

architecture dataflow of homework2_vhdl is
begin
    x<=(a and b) or (c and d);
end dataflow;
```

Line 3 | Col 24 | INS

Waveform


homework2_vhdl.scf - Waveform Editor
Ref: 0.0ns
Time: 157.6ns
Interval: 157.6ns
0.0ns
Name:
Value:
20.0ns
40.0ns
60.0ns
80.0ns
100.0ns
120.0ns
140.0ns
D 0
C 0
B 0
A 0
X 0

… Two more gates

- ✓ XOR
- ✓ XNOR

OR gate

A	B	OR gate
0	0	0
0	1	1
1	0	1
1	1	1

XOR (eXclusiveOR) gate

A	B	OR gate	XOR gate
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

XOR (eXclusiveOR) gate

A	B	OR gate	XOR gate
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

$$A \text{ XOR } B = A \oplus B$$

$$= \overline{A}\,B + A\,\overline{B}$$

It produces a high output whenever the two inputs are at opposite levels

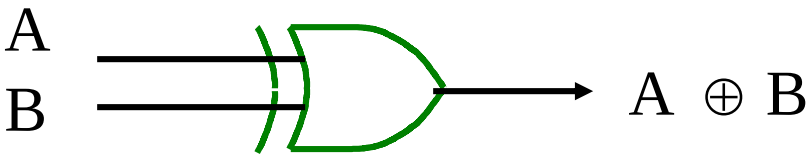
XOR (eXclusiveOR) gate

A	B	OR gate	XOR gate
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

$$A \text{ XOR } B = A \oplus B$$

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

It produces a high output whenever the two inputs are at opposite levels



Another gate … XNOR

$$\overline{A \oplus B} = ?$$

XNOR (eXclusiveNOR) gate

A	B	XOR gate	XNOR gate
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

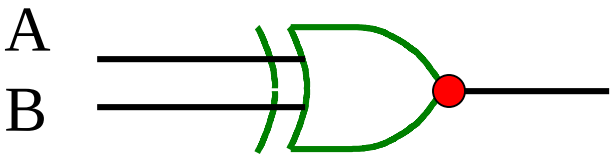
$\odot$

XNOR (eXclusiveNOR) gate

A	B	XOR gate	XNOR gate
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

$$A \text{ XNOR } B = \overline{A \oplus B}$$

$$= \overline{A}\,\overline{B} + A\,B$$



$$\overline{A \oplus B} = A \odot B$$

Total we have $2^4 = 16$ gates …

AB
00
01
10
11

Total we have $2^4 = 16$ gates …

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

Total we have $2^4 = 16$ gates …

AB	0	AND		A		B	XOR	OR	NOR	XNOR	NOTB		NOTA		NAND	1
00	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
01	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
10	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
11	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Only 7 gates are useful

AND, XOR, OR, NOR, XNOR, NOT, NAND