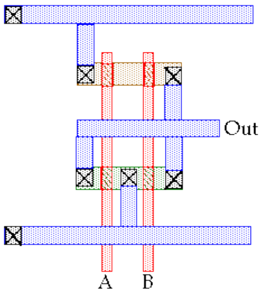
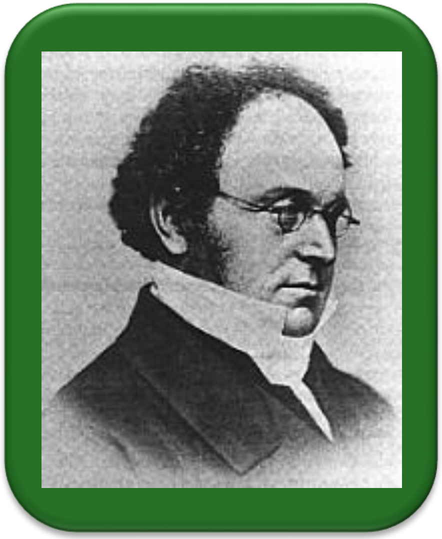


Binary Logic → continue

DeMorgan Theorems

Augustus DeMorgan (1806-1871)



British mathematician born in India

DeMorgan's theorems

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

➢ breaking the bar changes the logic operation (OR) under the bar

➢ breaking the bar changes the logic operation (AND) under the bar

Proof of the DeMorgan's theorem

Proof of the DeMorgan's theorem

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 theorem

x	y	$\bar{x}$	$\bar{y}$	$x\,y$	$\overline{x\,y}$	$\bar{x}+\bar{y}$	$\bar{x}\ \bar{y}$	$x + y$	$\overline{x + y}$
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

x	y	$\bar{x}$	$\bar{y}$	$x\,y$	$\overline{x\,y}$	$\bar{x}+\bar{y}$	$\bar{x}\;\bar{y}$	$x+y$	$\overline{x+y}$
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

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

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

First application of the theorem

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

Second application of theorem

$$\overline{A\,\overline{B} + C} = \boxed{\overline{A\,\overline{B}}}\;\overline{C}$$

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

Distribute…

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

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

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

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

Sum-of-products form

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

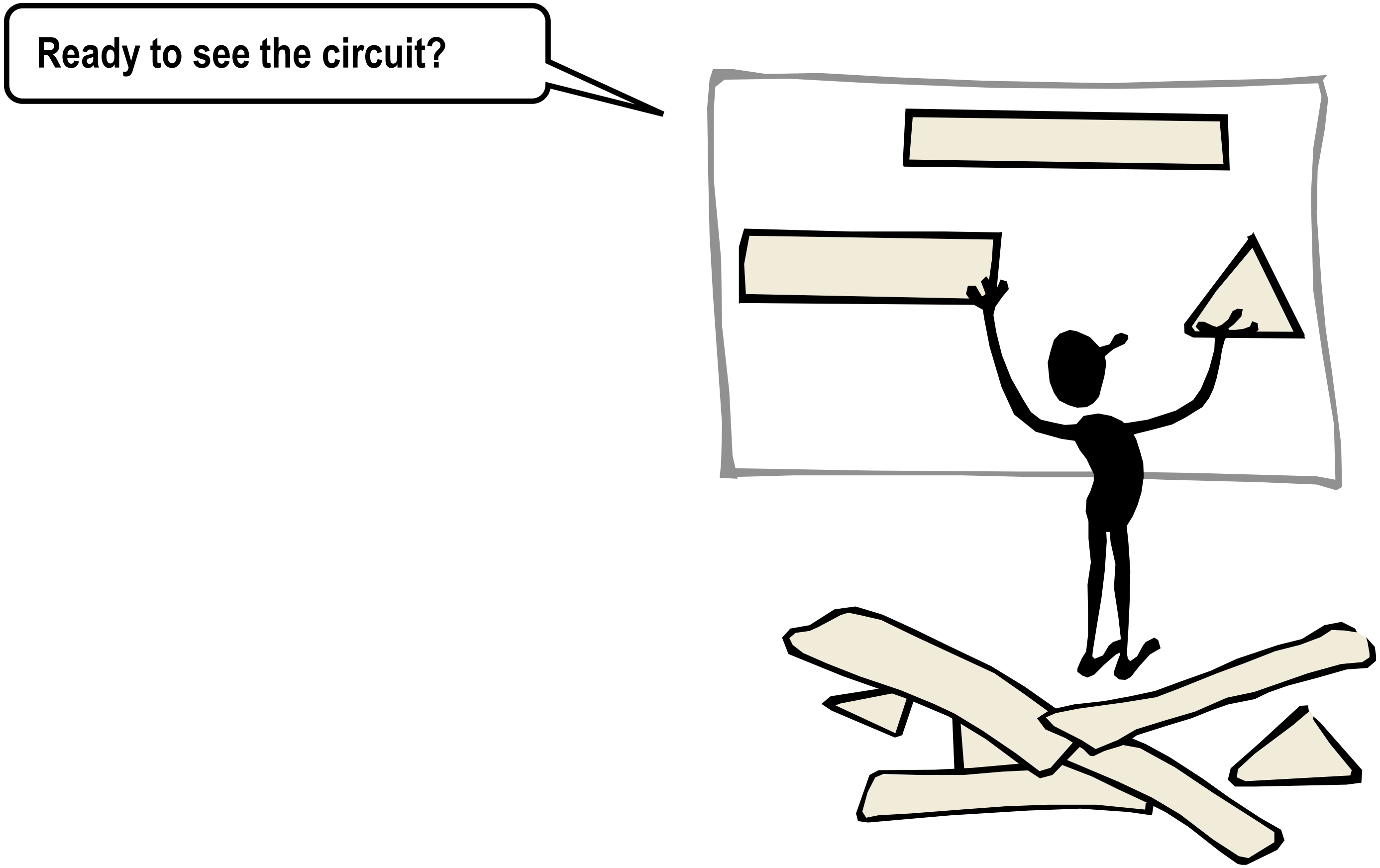
$$= (\overline{A} + \overline{\overline{B}})\;\overline{C}$$

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

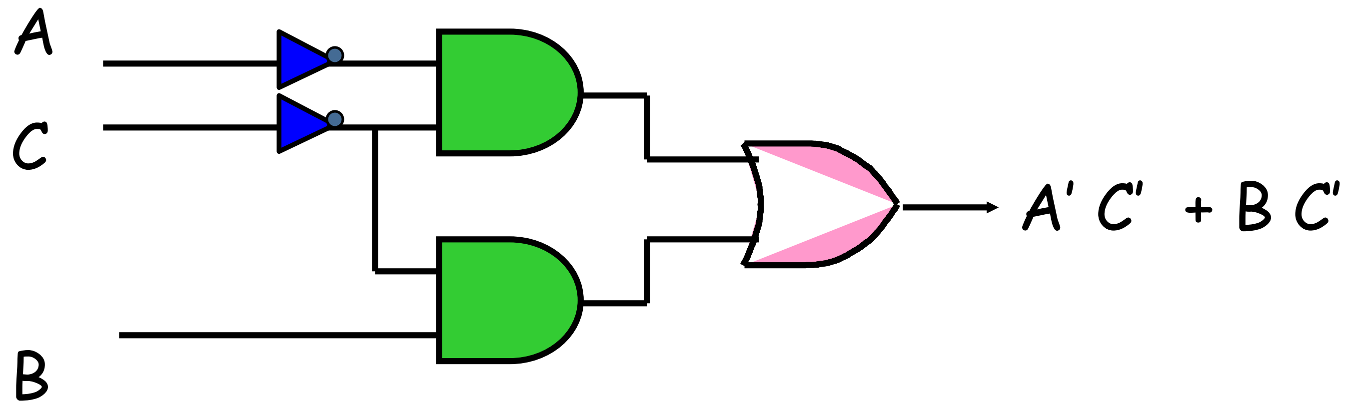
$$= \overline{A}\,\overline{C} + B\,\overline{C}$$

$\boxed{\text{Implement with gates}}$

Sum-of-Products form

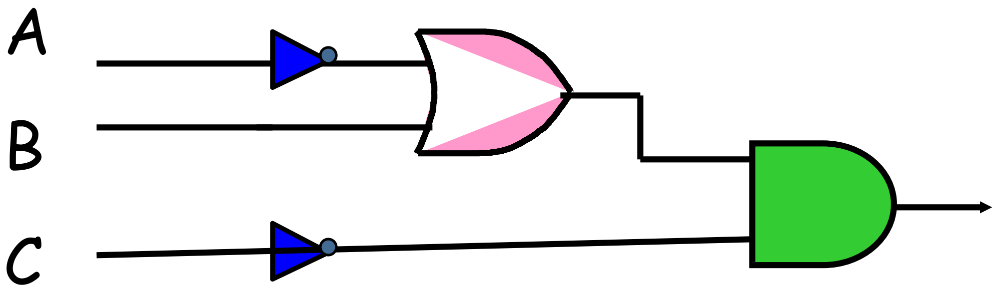


The SOP leads to “two-level-realization”

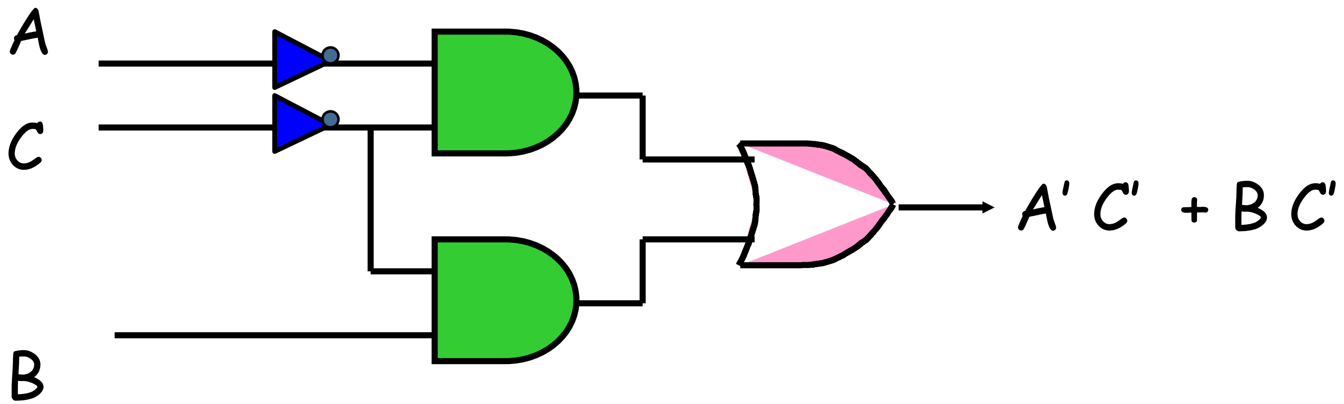


factor … “multi-level-realization”

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



Fun-in & Fan-out



Fan-in: max number of inputs a gate can accept

Fan-out: max number of inputs a gate can drive

More gates….

More Gates

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

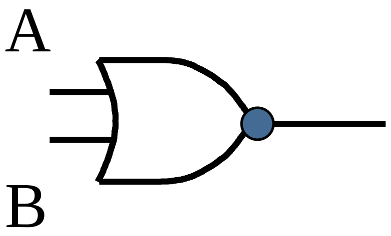
OR

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



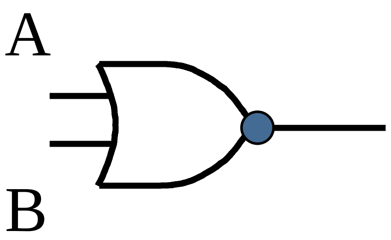
$\overline{A + B}$

Using DeMorgan's

theorem

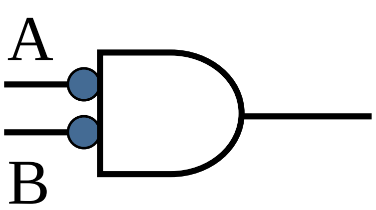
A	B	OR	NOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

NOR



$\overline{A + B}$

Using DeMorgan's theorem



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

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

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

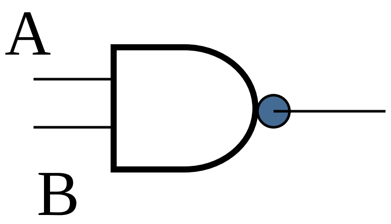
AND … Not AND

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

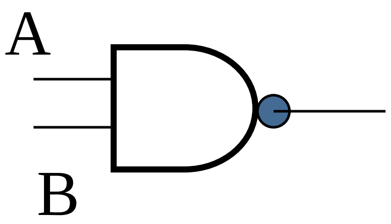


$$\overline{A \bullet B}$$

Using DeMorgan's theorem

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

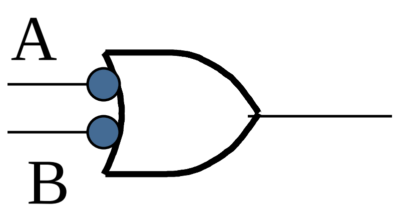
NAND



$\overline{A \bullet B}$

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

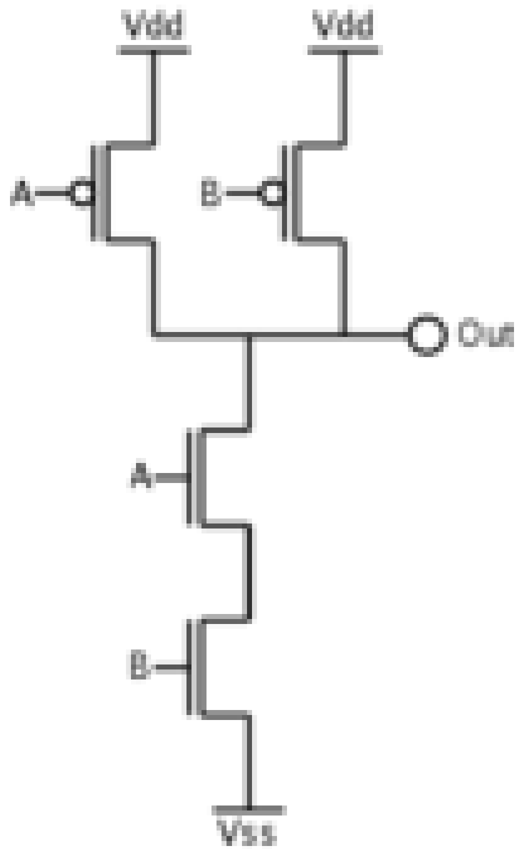
Using DeMorgan's theorem

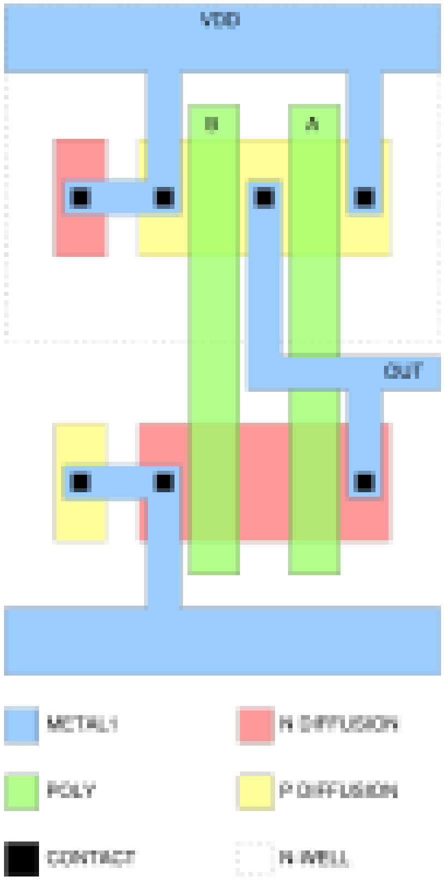


$\overline{A} + \overline{B}$

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

NAND: CMOS and gate layout

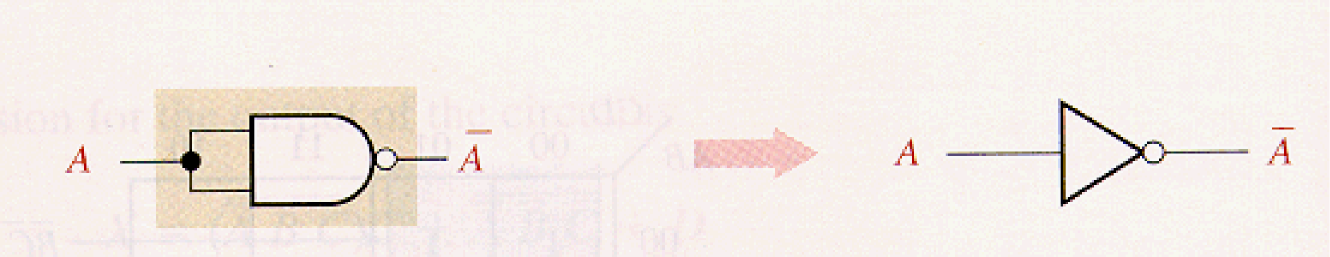




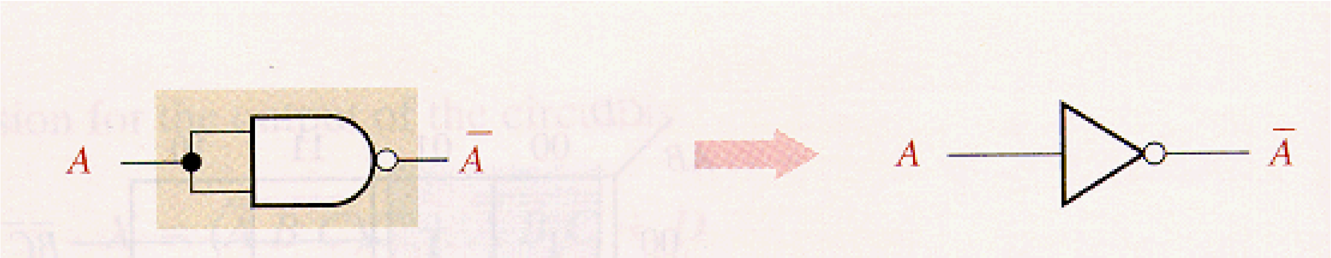
Universality of NAND and NOR Gates

…can implement any Boolean expression

Universality of NAND: NOT

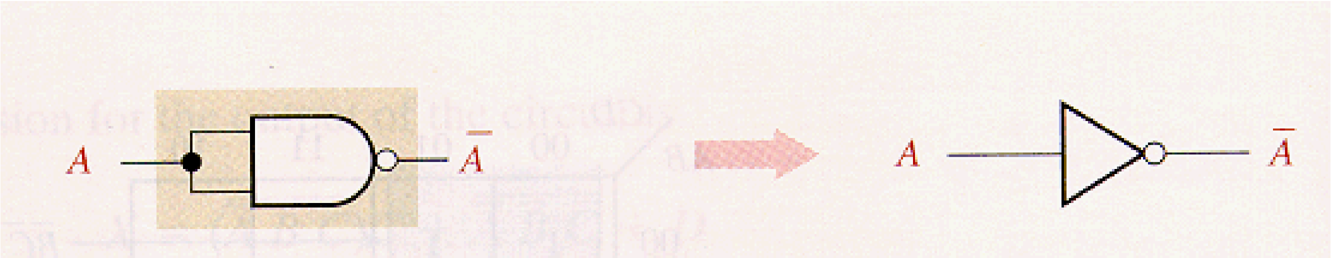


Proof



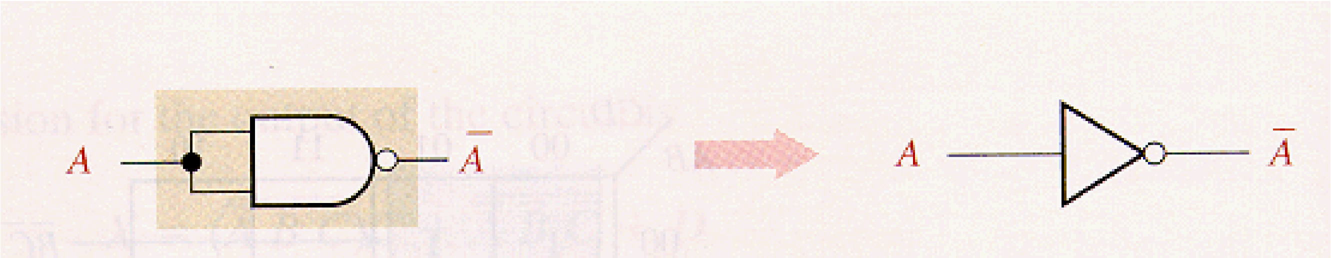
A	AND	NAND
0	0	1

Proof



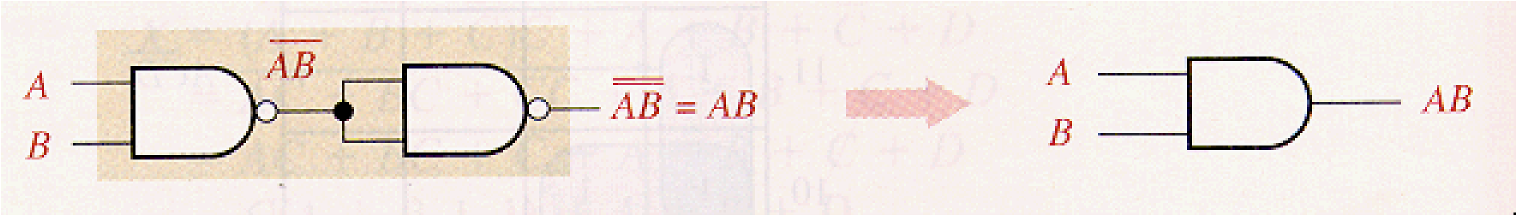
A	AND	NAND
0	0	1
1	1	0

Proof

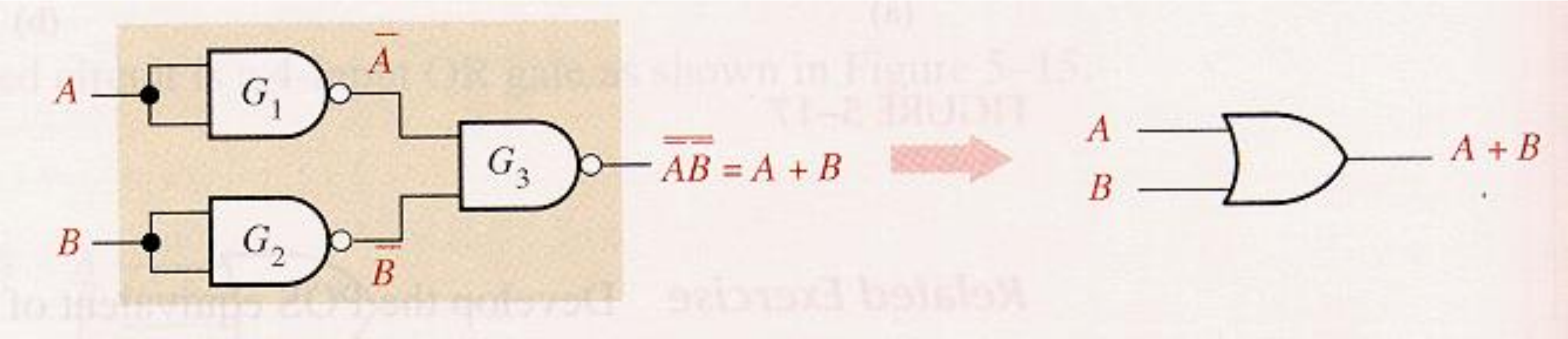


A	AND	NAND
0	0	1
1	1	0

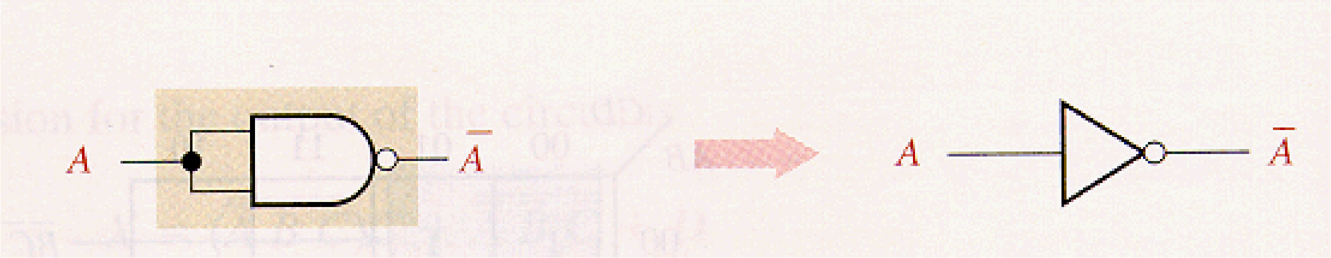
Universality of NAND: AND

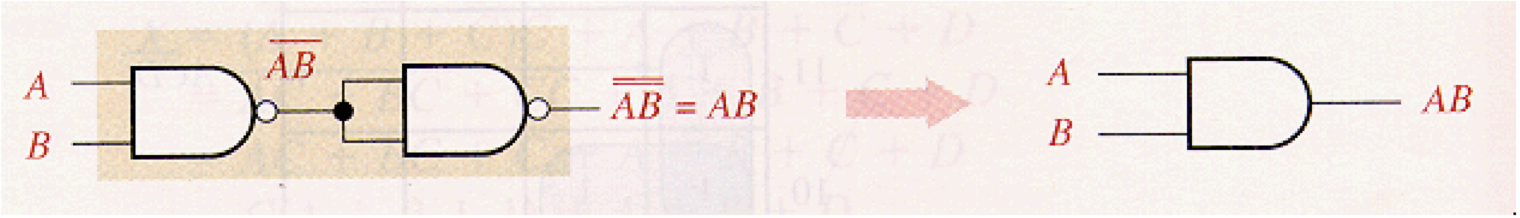


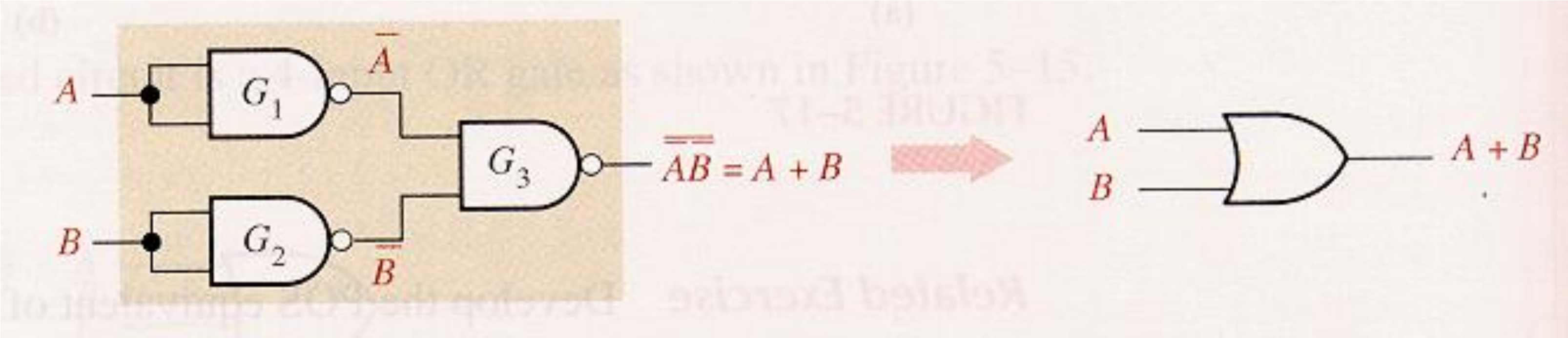
Universality of NAND: OR



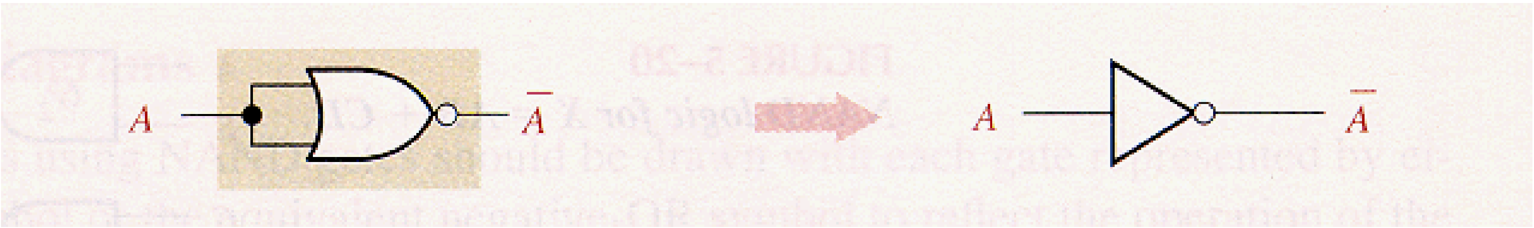
All


A
$\bar{A}$
A
$\bar{A}$

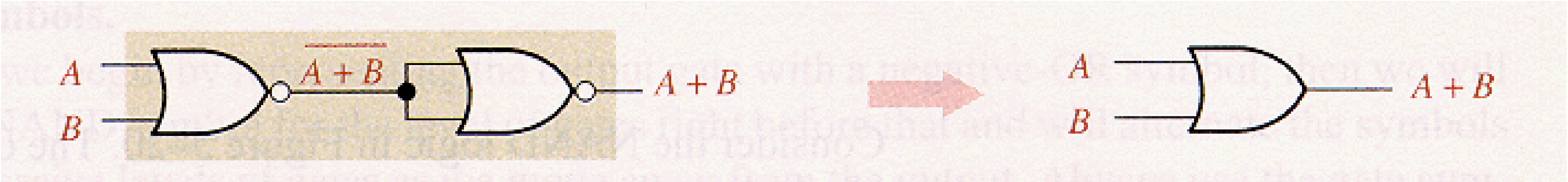

A
B
$\overline{AB}$
$\overline{\overline{AB}} = AB$
A
B
AB


A
B
G_1
G_2
G_3
$\bar{A}$
$\bar{B}$
$\overline{\bar{A}\bar{B}} = A + B$
A
B
A + B

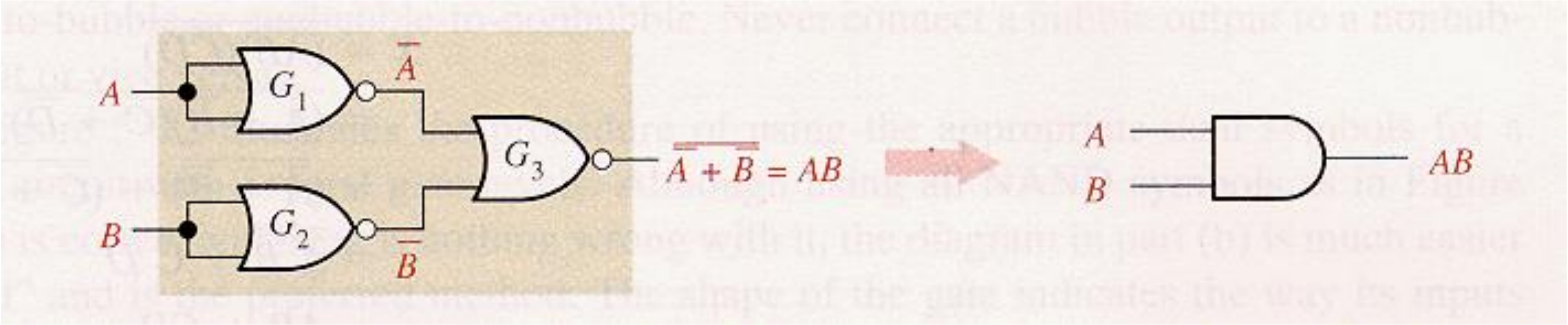
Universality of NOR: NOT



Universality of NOR: OR



Universality of NOR: AND

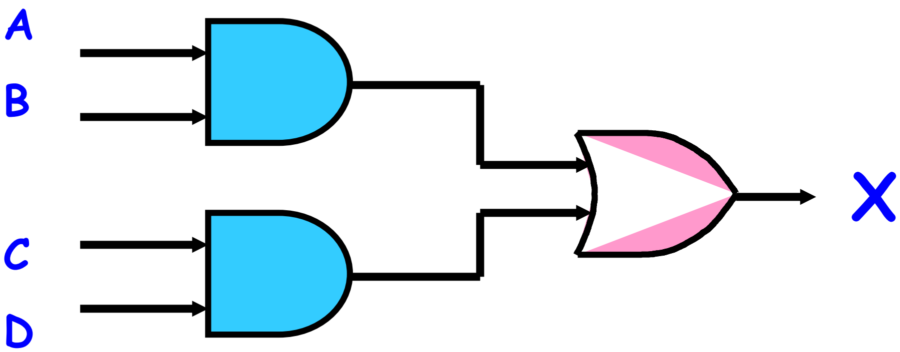


Example: implement (two-level-realization) the Boolean function: $X = AB + CD$, using:

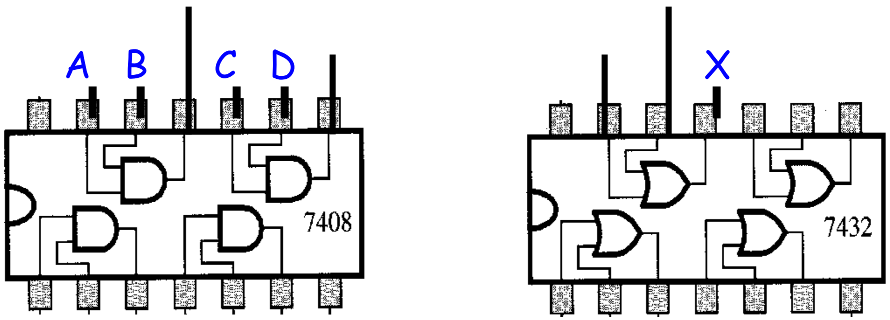
1. AND,OR,NOR gates
2. NAND gates

You have ...5 minutes ...

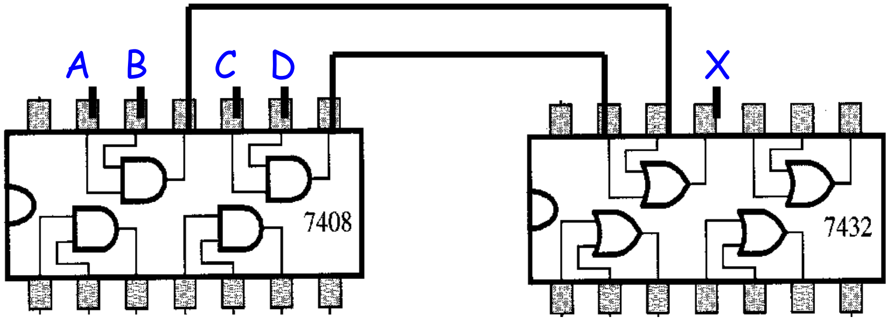
1) X = AB + CD; Using AND, OR, NOT gates



X = AB + CD; Using Chips



X = AB + CD; Using Chips



X = AB + CD; Using NAND gates

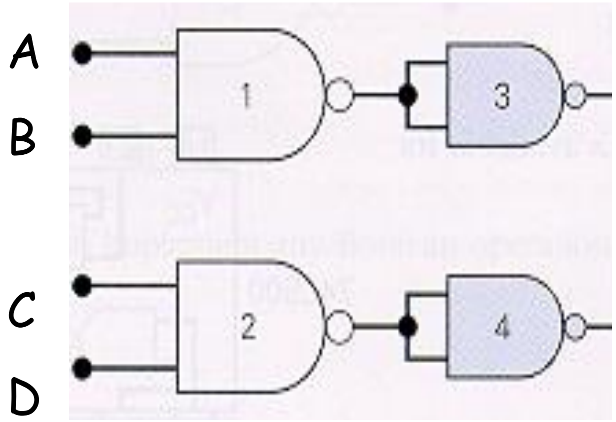
A

B

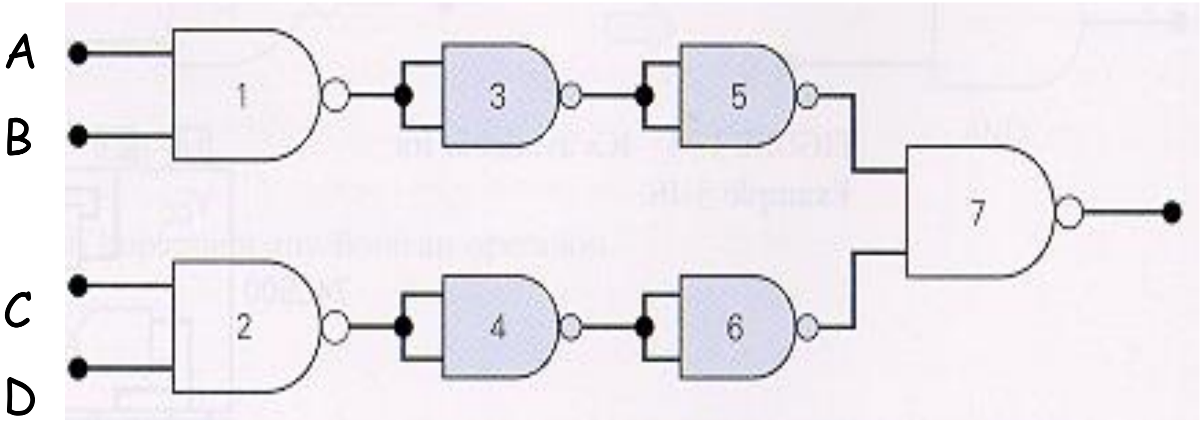
C

D

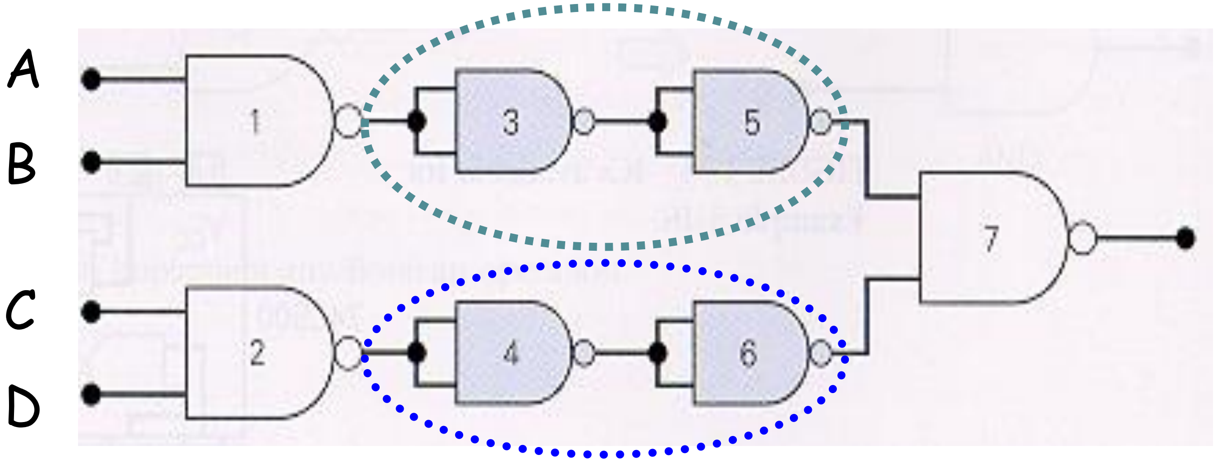
X = AB + CD; Using NAND gates



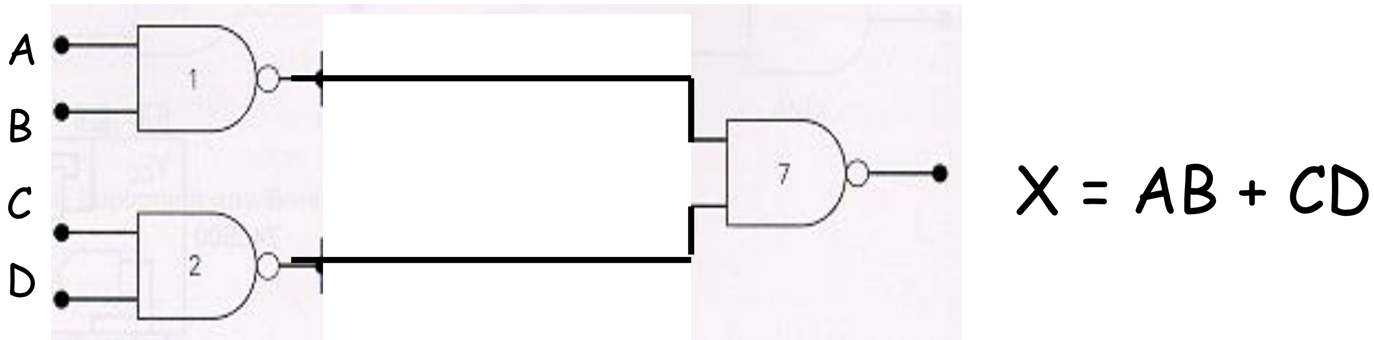
X = AB + CD; Using NAND gates

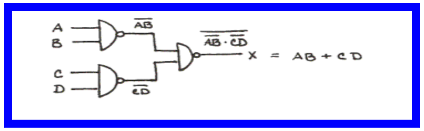


Two inverts cancel each other



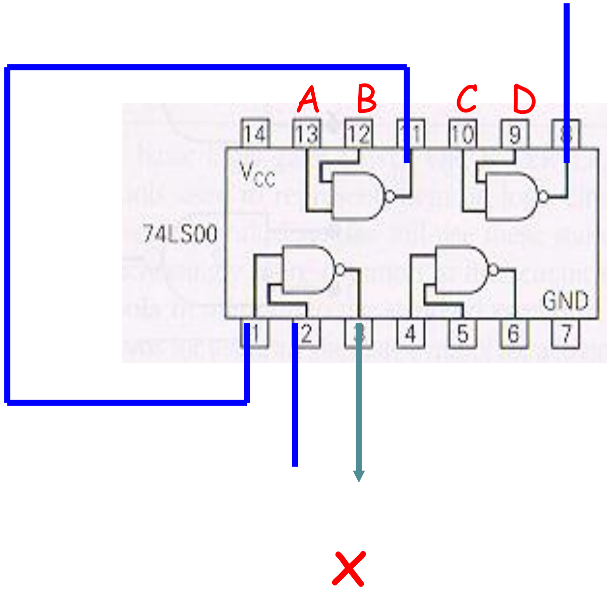
X = AB + CD; Using NAND gates



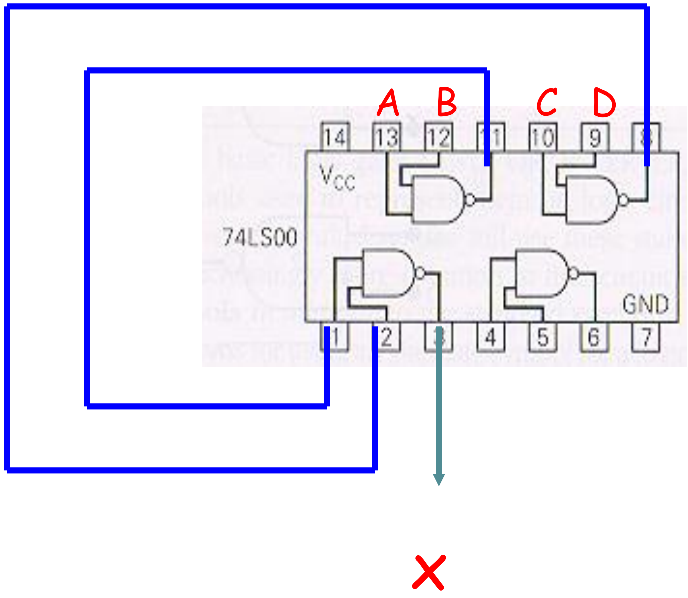


Note: 2 inverts in series cancel each other

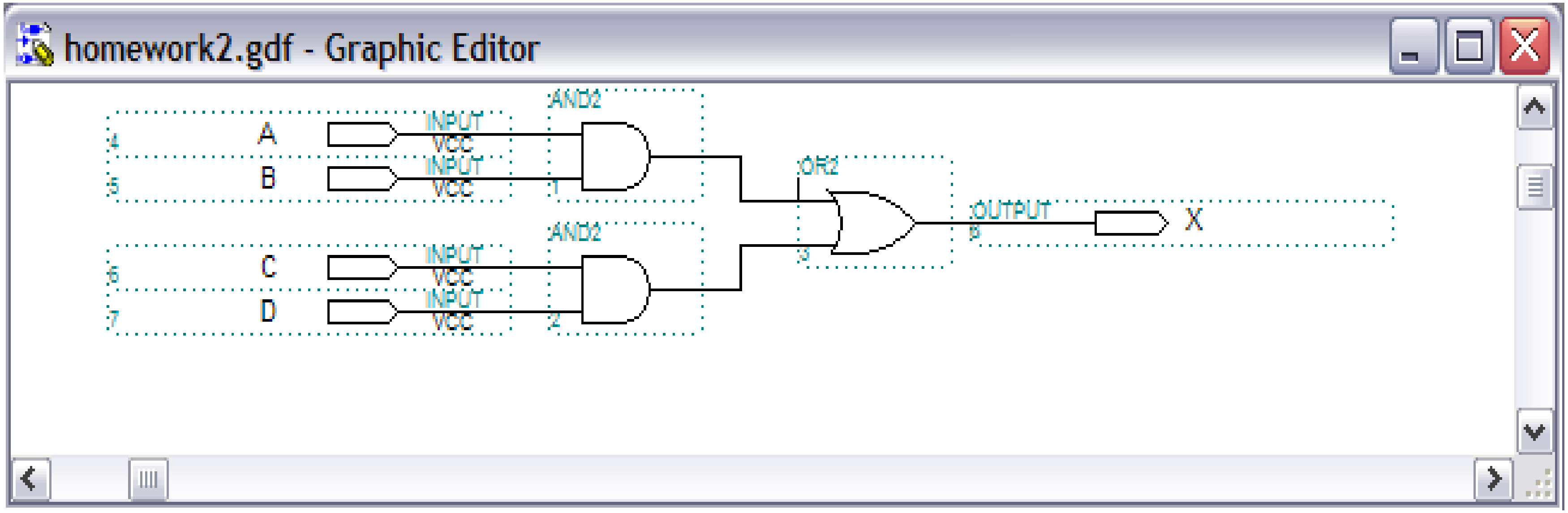
X = AB + CD; Using NAND chips



X = AB + CD; Using NAND chips



X = AB + CD; Using VHDL



VHDL Code: X = AB + CD

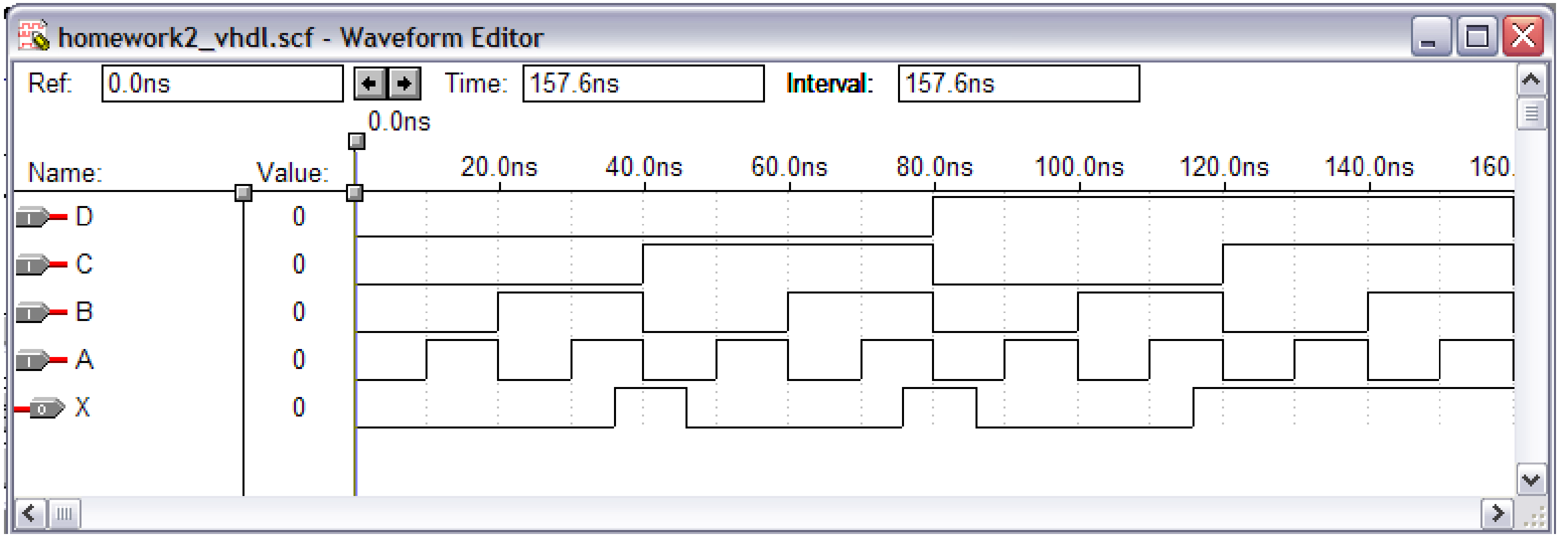
homework2_vhdl.vhd - Text Editor

```
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

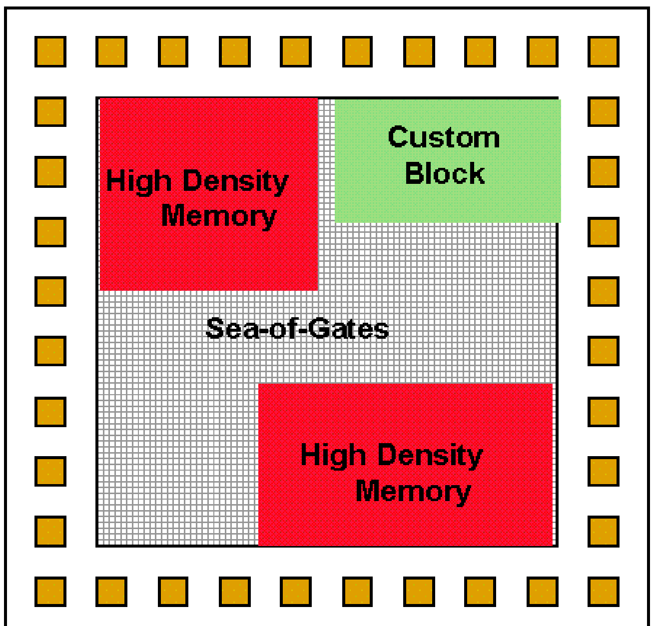
Waveform



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
160.
D 0
C 0
B 0
A 0
X 0

An array with many NAND gates

Sea-of-Gates Array Technology



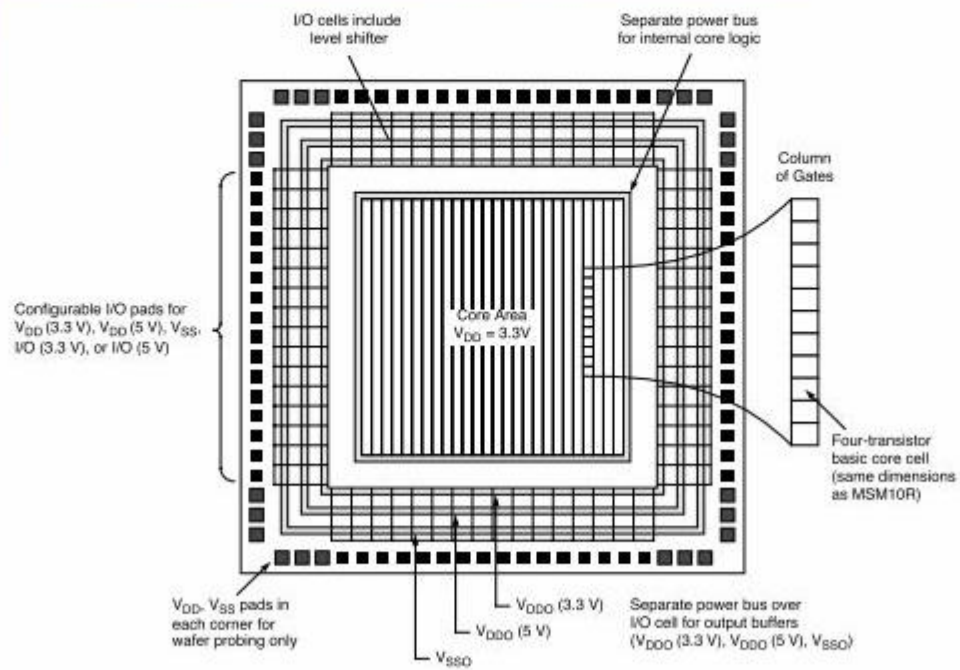
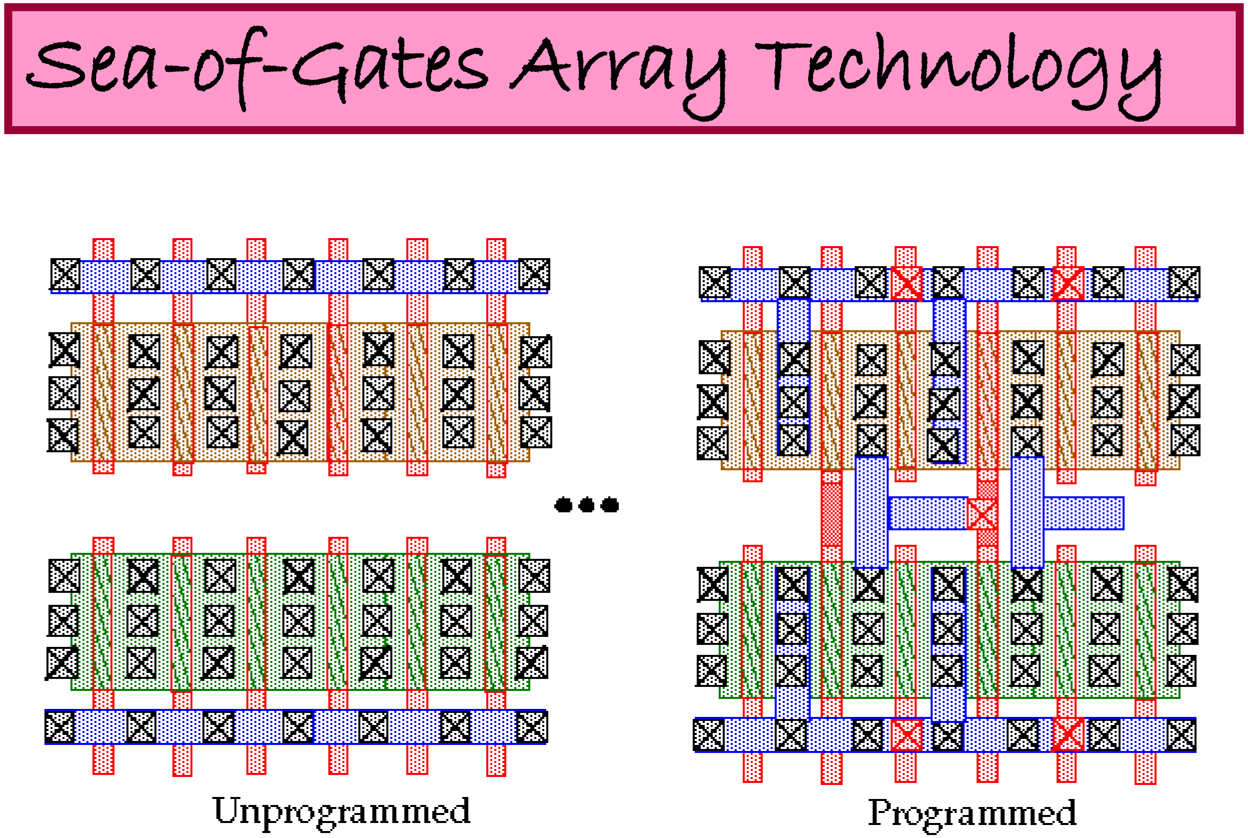


Figure 7. MSM13R0000 Array Architecture

An array with many NAND gates



http://www.ece.unm.edu/~jimp/vlsi/slides/chap5_1.html

… 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

$\oplus$

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

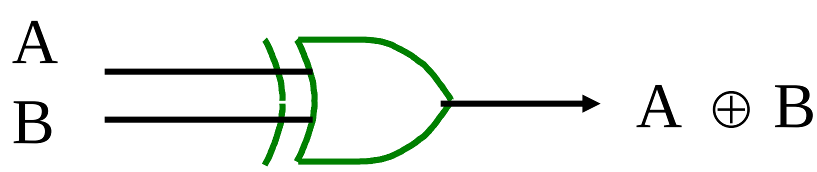
XOR (eXclusiveOR) gate

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

$\odot$

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

XNOR (eXclusiveNOR) gate

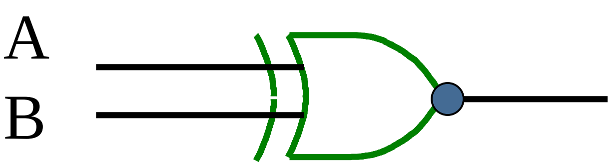
⊙

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

⊙

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

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



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