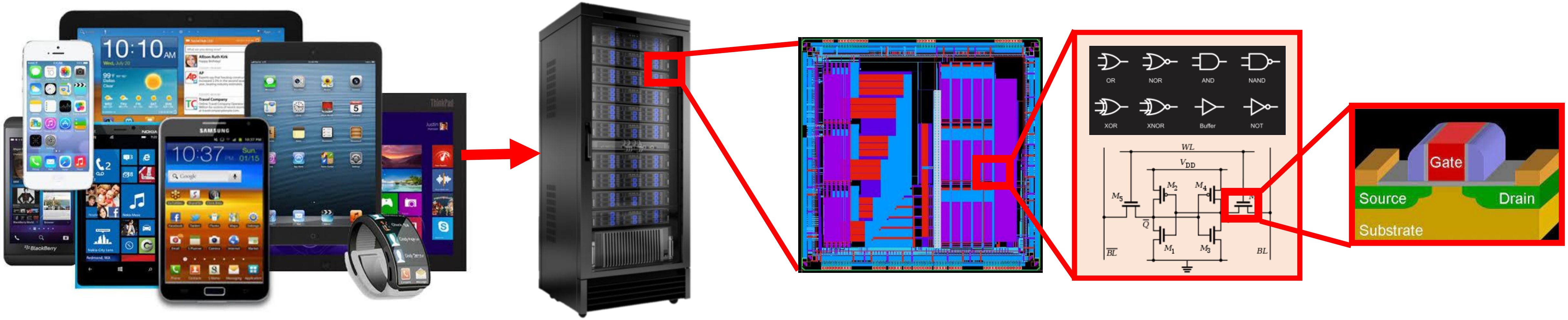
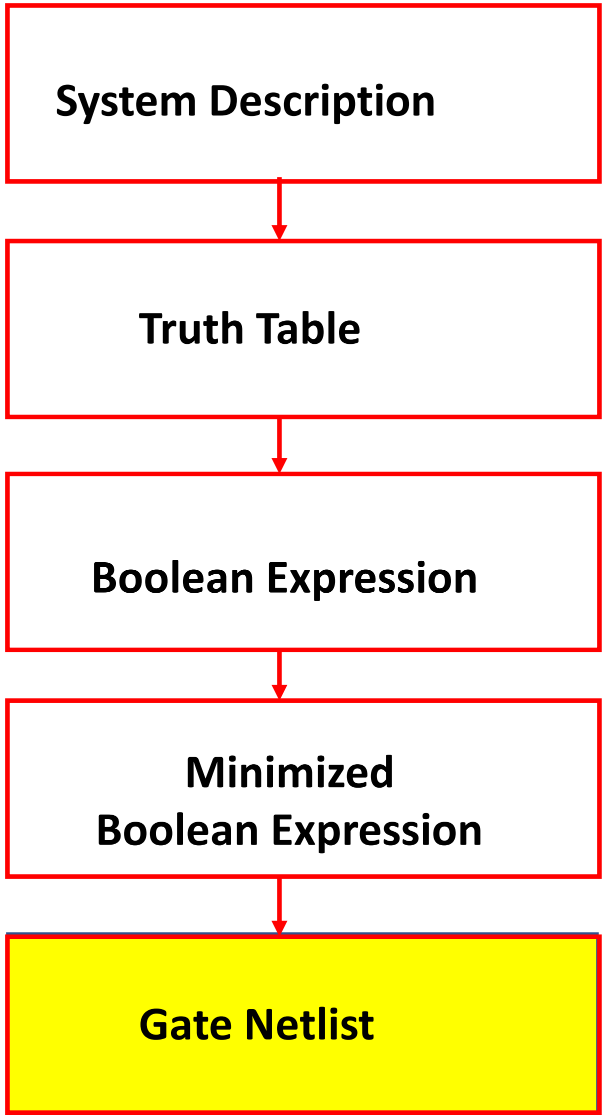


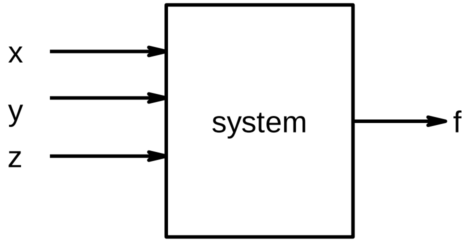
ESC201: INTRODUCTION TO ELECTRONICS

MODULE 6: DIGITAL CIRCUITS

Dr. Shubham Sahay,
Associate Professor,
Department of Electrical Engineering,
IIT Kanpur

Design Flow

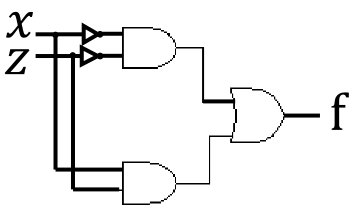




x	y	z	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

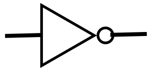
$$\mathrm{f} = \overline{x}.\overline{y}.\overline{z} + \overline{x}.y.\overline{z} + x.\overline{y}.z + x.y.z$$

$$\Rightarrow \mathrm{f} = \overline{x}.\overline{z} + x.z$$



Basic Boolean Operations with NAND

NOT operation

x  $\overline{x}$

x

$\overline{x.x} = \overline{x}$

AND operation

x

y

$x \cdot y$

x

y

$\overline{x.y}$

$x.y$

OR operation

x

y

$x + y$

x

y

$\overline{x}$

$\overline{y}$

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

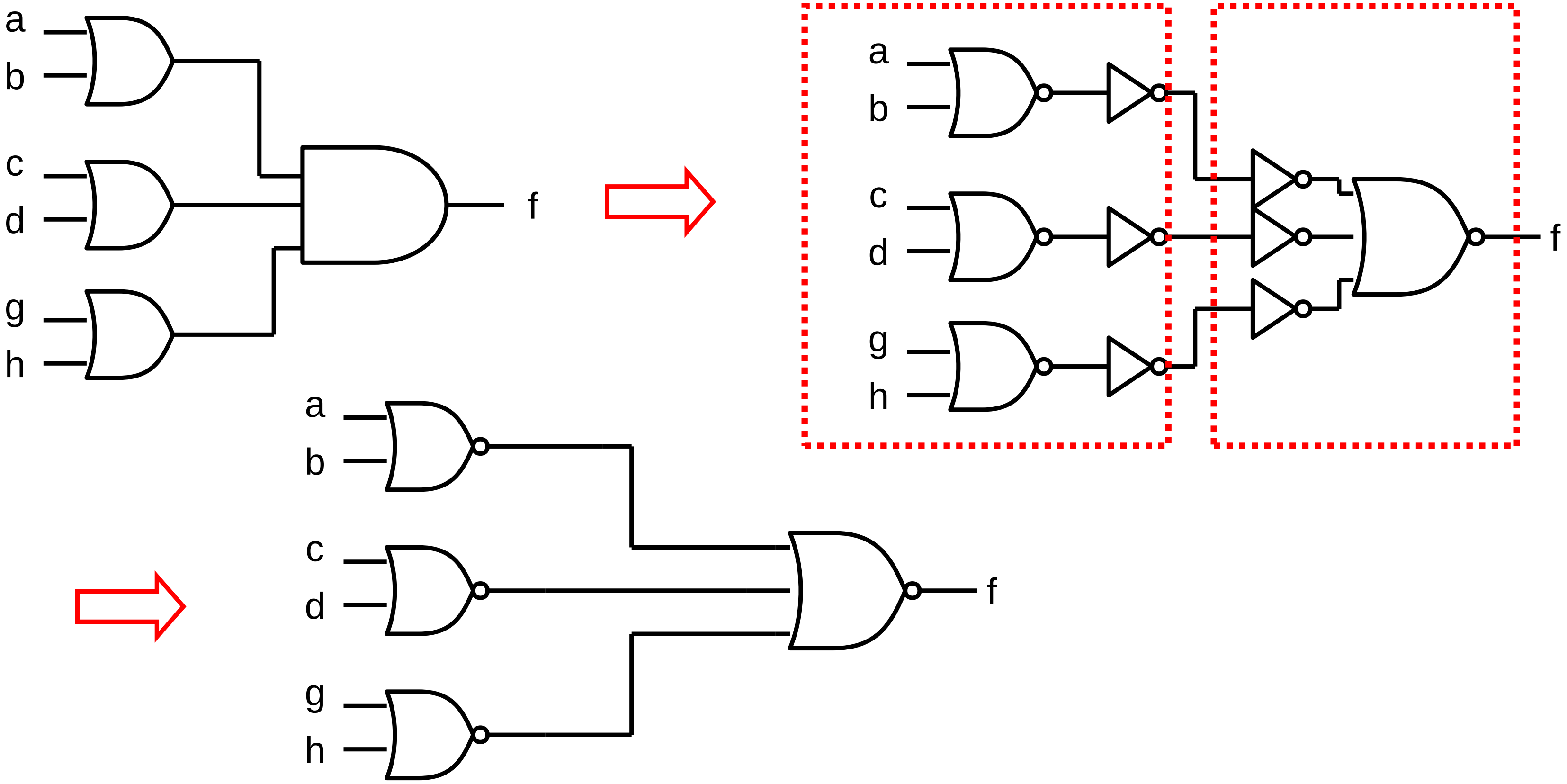
Exercise

Implement NOT, AND and OR with NOR gates

Implementing Boolean Function with Universal Gates

To implement using NOR gates, it is easiest to start with minimized Boolean expression in POS form

$$f = (a+b).(c+d).(g+h)$$

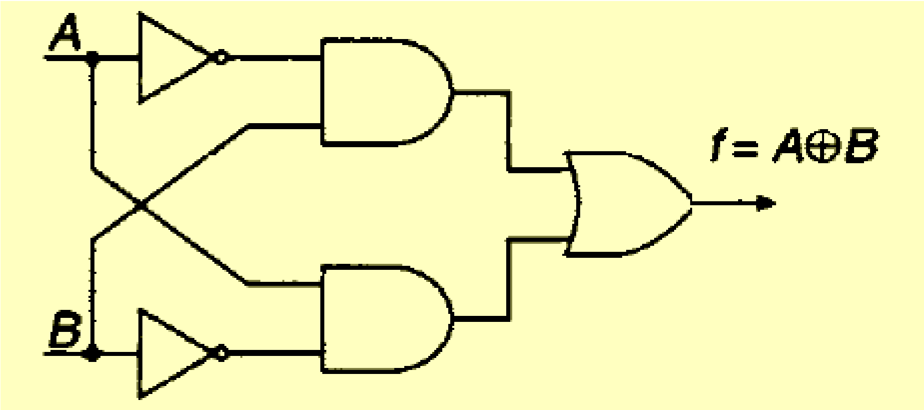


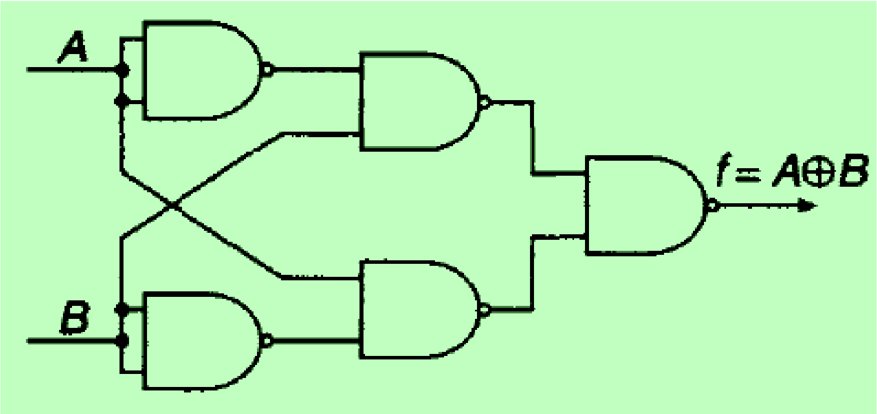
There is one-to-one mapping between OR-AND network and NOR network.

Similarly, there is a one-to-one mapping between AND-OR network and NAND network.

Example

Implement XOR function with NAND gates: $f = \overline{A}.B + A.\overline{B}$ Already in SoP form

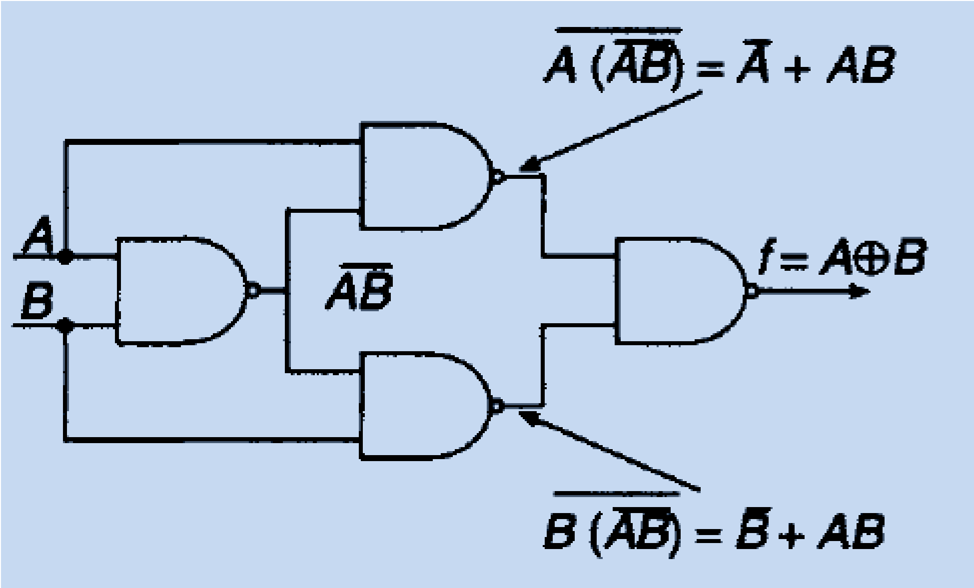




Example

$$f = \overline{A}.B + B.\overline{B} + A.\overline{B} + A.\overline{A}$$
$$= B(\overline{A} + \overline{B}) + A(\overline{A} + \overline{B})$$

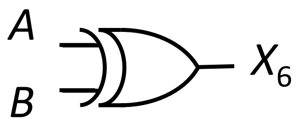
Going as per algorithm:
8 two I/P and 1 four I/P NAND
versus
4 two I/P NAND
for ckt. to the right

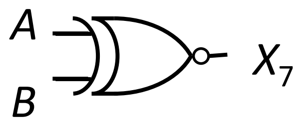


Popular and Useful Gates

Two gates are popular for useful in Boolean Logic implementation in hardware

Gate





Operation

XOR | XNOR

Algebraic Represetnation

$$X_6 = \overline{A} \cdot B + A \cdot \overline{B} = \mathrm{A} \oplus \mathrm{B}$$

$$X_7 = A \cdot B + \overline{A} \cdot \overline{B} = A \odot B = A \equiv B$$

Truth Table

XOR:

A	***B***	***X_1***
0	0	0
0	1	1
1	0	1
1	1	0

XNOR:

A	***B***	***X_1***
0	0	1
0	1	0
1	0	0
1	1	1

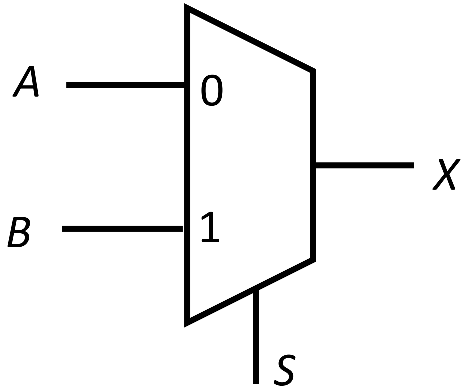
These gates are useful for many operations including <u>addition</u> and <u>comparing</u>.
They are **not** Universal Gates for implementing Boolean functions.

More than two inputs XOR and XNOR gates is a possibility and are often used.

Some Other Methods of Implanting Boolean Functions

- Circuits implementing certain functions may also be used as universal gates.

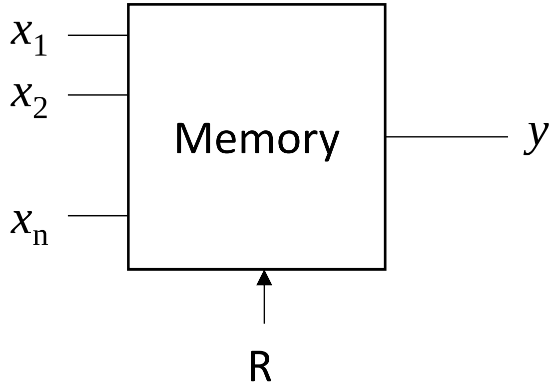
Example: MUX (or multiplexer)



For the two input MUX, $X = A \cdot S + \overline{B} \cdot S$

By choosing inputs A, B and select S as Boolean variables or Boolean constants of 0 or 1, one can implement all Basis functions AND, OR and NOT.

- <u>Look up tables</u> (LUT) or memories



Values of y^s corresponding x_i^s are stored in memory.

Recall y value based on x_i inputs and read signal R

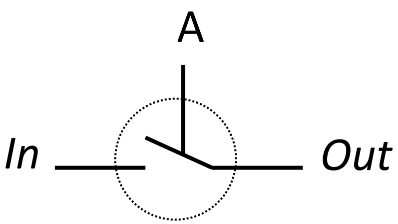
There may be many more approaches to implement Boolean Functions.
The quest is on!

Positive and Negative Switch

Define high voltage ≡ logic 1

Define low voltage ≡ logic 0

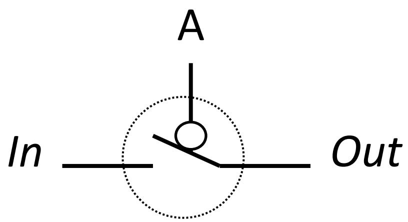
Positive Switch



A is logic 1 - Switch is closed

A is logic 0 - Switch is open

Negative Switch

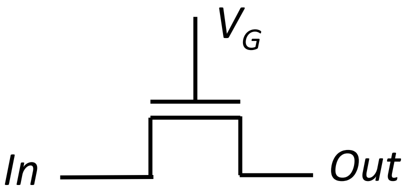


A is logic 1 - Switch is open

A is logic 0 - Switch is closed

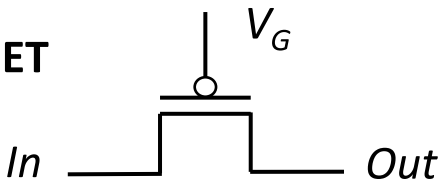
The MOSFET behaves this way and has been popular to build logic circuits

N-MOSFET



V_G is high (logic 1)
low resistance between In and Out
V_G is low (logic 0)
high resistance between In and Out

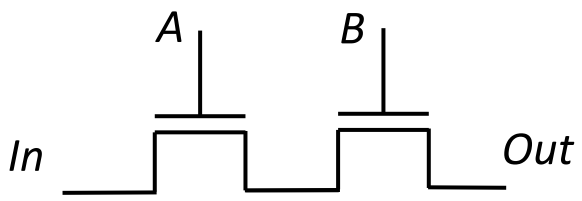
P-MOSFET



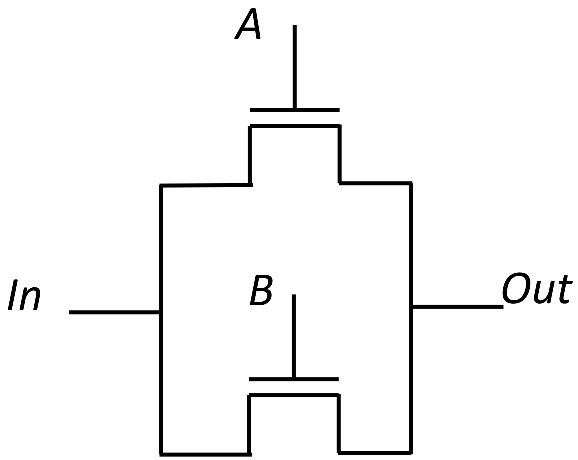
V_G is high (logic 1)
high resistance between In and Out
V_G is low (logic 0)
low resistance between In and Out

Combining Switches

N-MOSFET (<u>positive</u>) switches

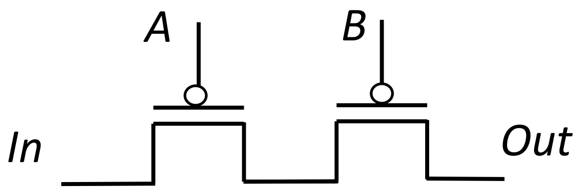


Out transparent to *In* for $A \cdot B = 1$

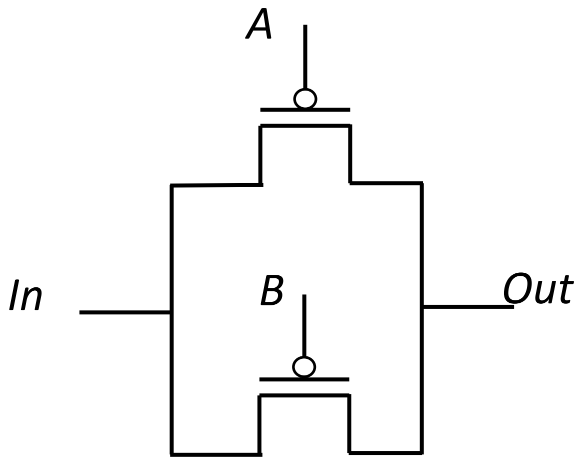


Out transparent to *In* for $A + B = 1$

P-MOSFET (<u>negative</u>) switches

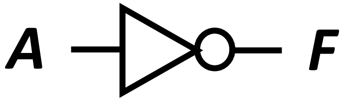


Out transparent to *In* for $\overline{A} \cdot \overline{B} = 1$



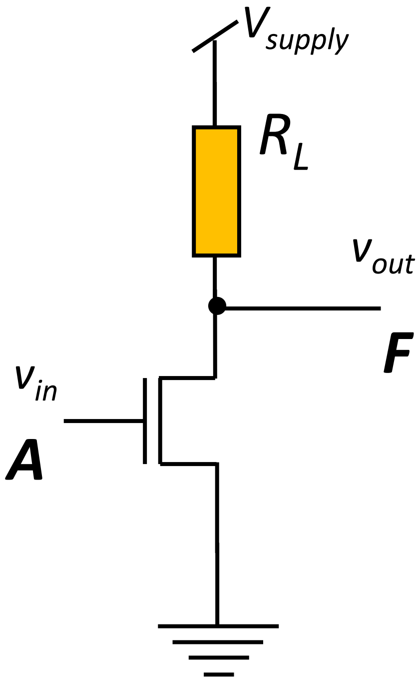
Out transparent to *In* for $\overline{A} + \overline{B} = 1$

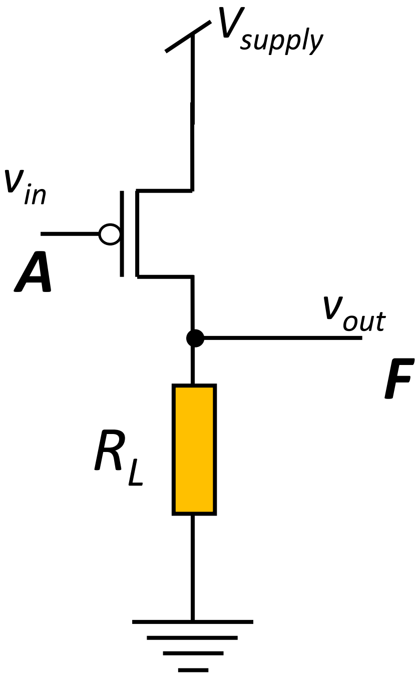
Inverters or NOT Gate

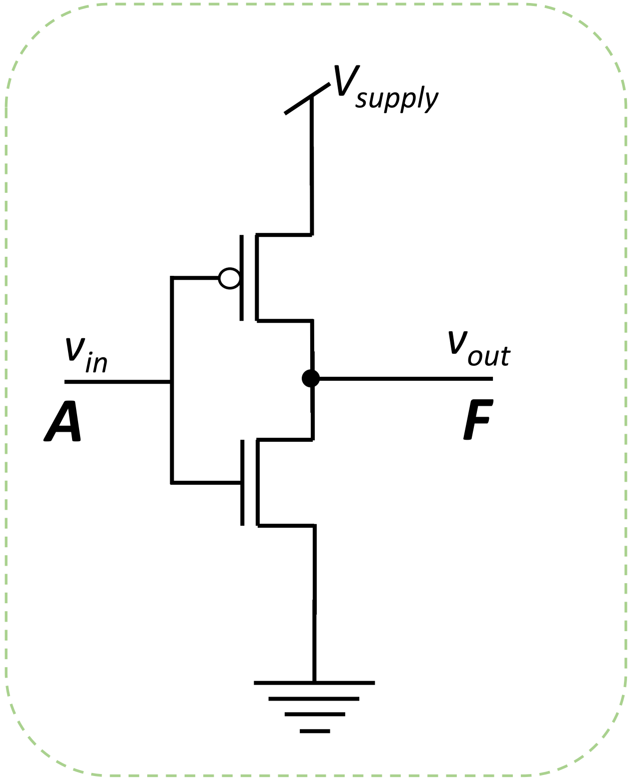


$F = \overline{A}$

A popular solution:
CMOS inverter





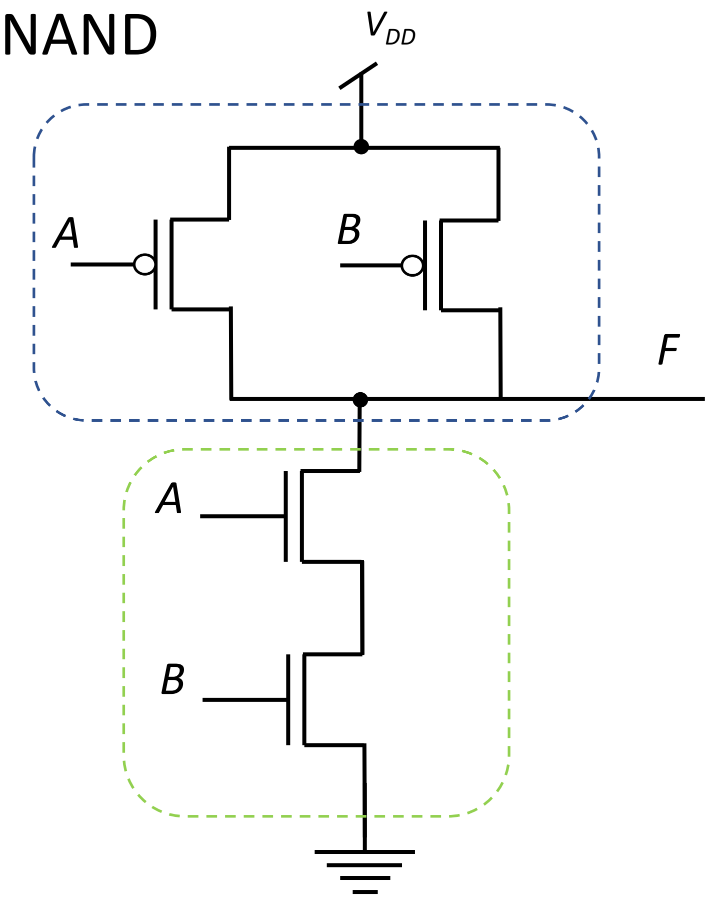


V_{in} and V_{out} are analogue values of input and output voltage

A and *B* are <u>Boolean values</u> of input and output.

Popular Two Input Universal Gates

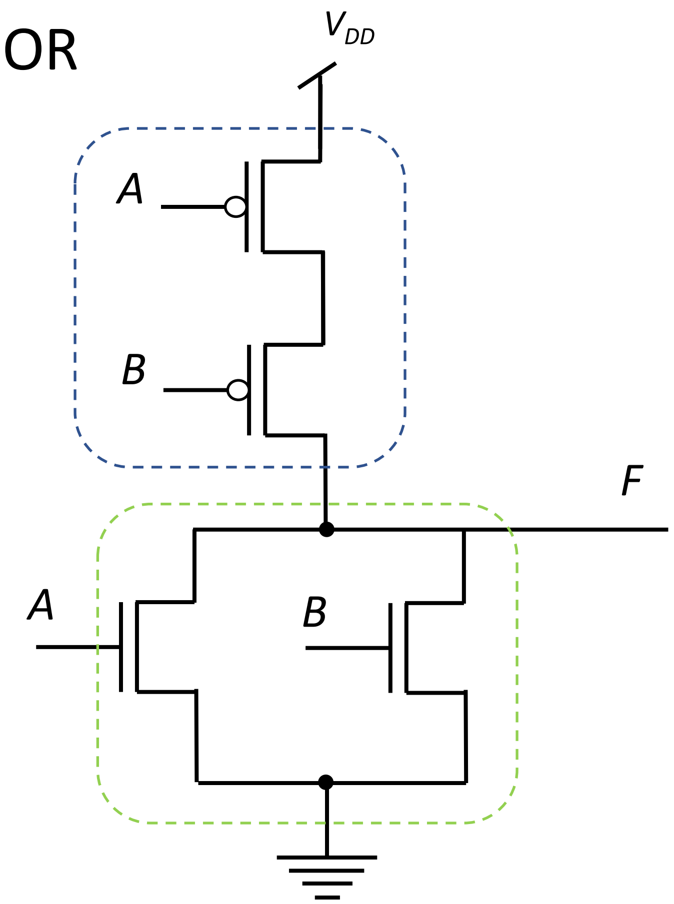
NAND



Two input NAND Gate

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

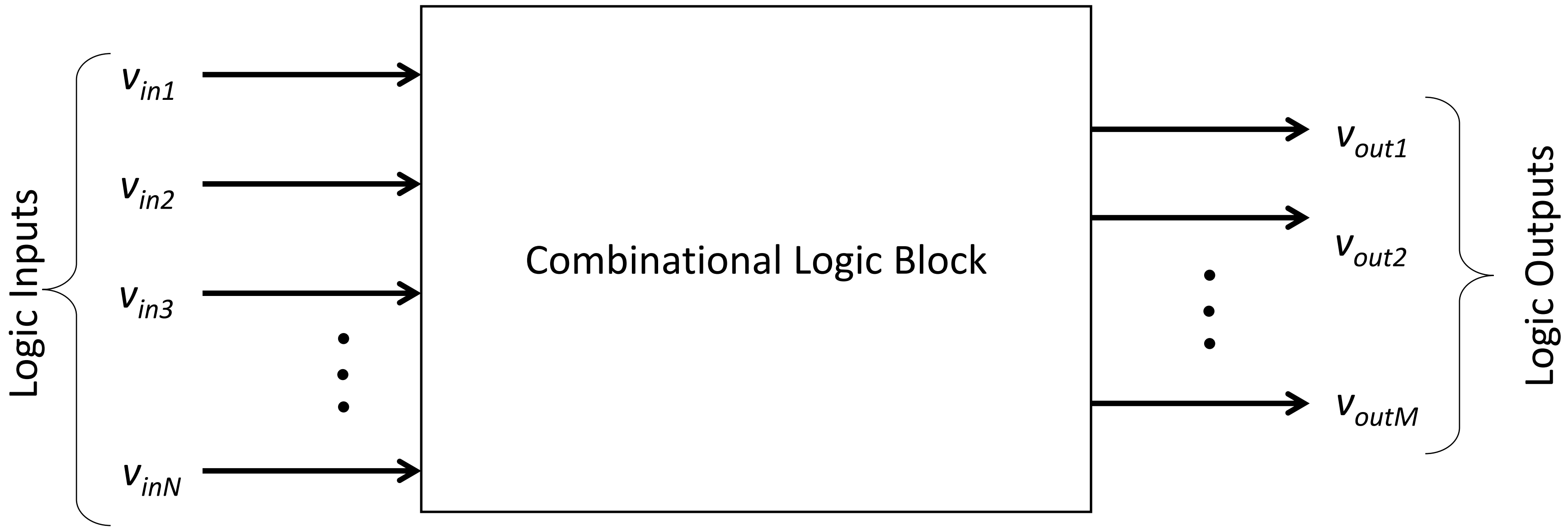
NOR



Two input NOR Gate

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

Combinational Logic



$v_{outi} = f_i\,(v_{in1}, v_{in2}, \ldots, v_{inN})$ for $i = 1$ to M

Here the f_i's are Boolean functions

The functions are typically built with logic gates

Any circuit that implements a combinational logic is a combinational circuit.