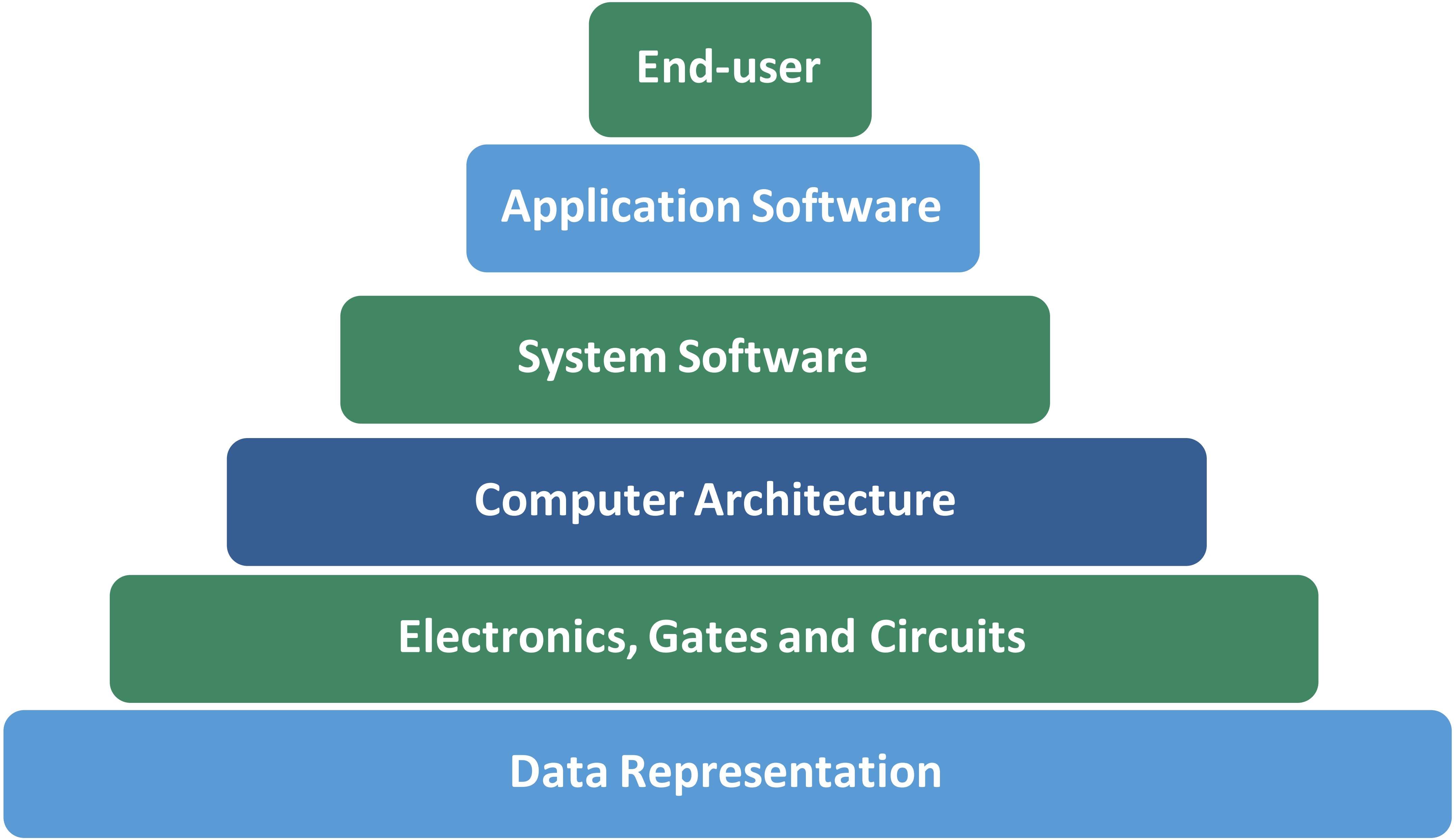
Layers of a Computing System



Data Representation

A Bit (Binary digIT) has two possible values – 0 and 1

Used to represent one of two discrete states.

0 (OFF) or 1 (ON)

0 (`False`) or 1 (`True`)

Two bits can represent four different things.

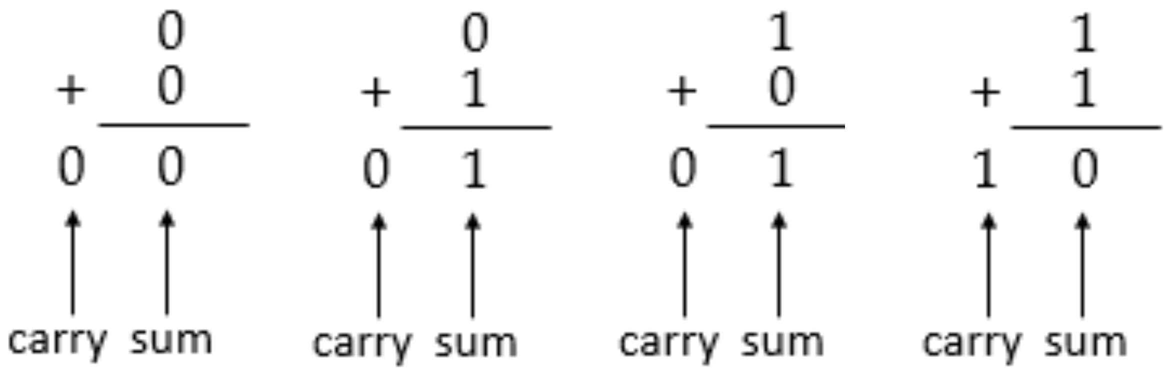
Three bits can represent eight different things.

How many things can *n* bits represent?

Bits are used to represent numbers, text characters, images, sound, etc.

Binary (bit) Addition

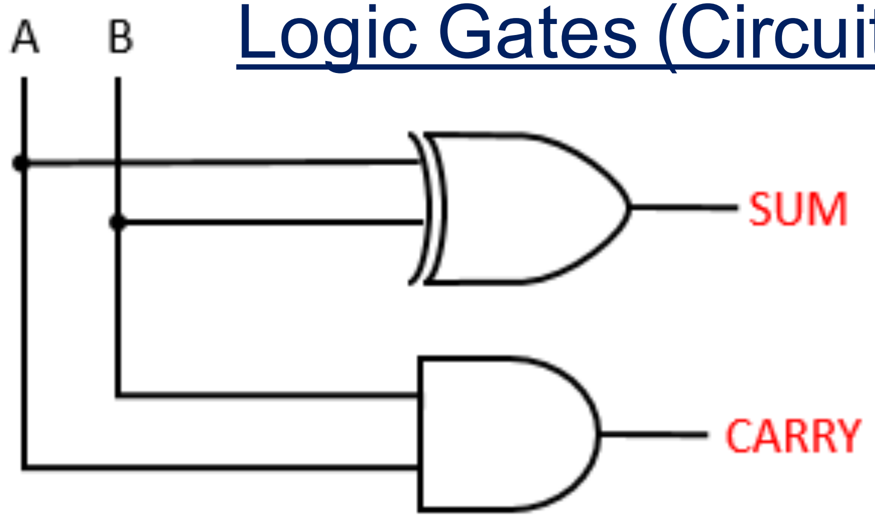
Only four possible combinations of input:



A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

<u>Truth Table</u>

<u>Logic Gates (Circuit)</u>

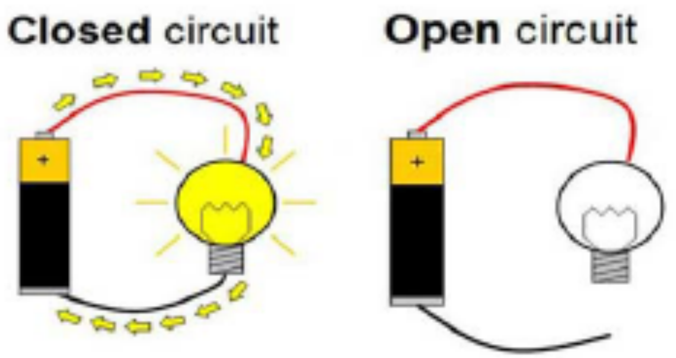


<u>Boolean Expressions</u>

$$SUM = A.\bar{B} + \bar{A}.B$$

$$CARRY = A.B$$

Logic Gates



Bits can be used to represent electrical signals:
0 (0V) or 1 (5V)

A gate is a device that performs a logical operation on electrical signals

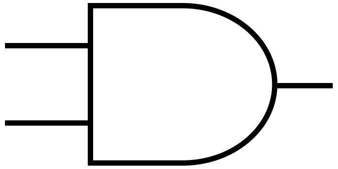
These logical operations were defined by the mathematician George Boole (1815-64)

The most common logic (Boolean) operations are:

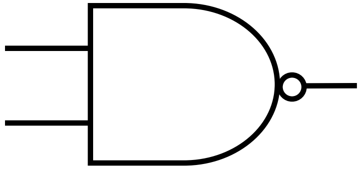
NOT	XOR
AND	NAND
OR	NOR

Logic Gates Symbols

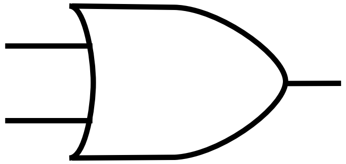
Each gate has its own logic symbol which allows circuits to be represented by a logic diagram



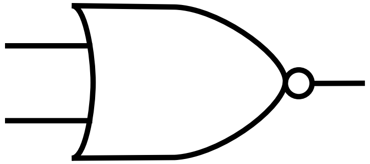
AND



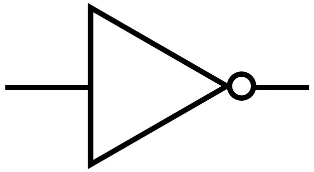
NAND



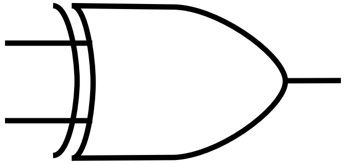
OR



NOR



NOT



Exclusive OR (XOR)

Logic gates have one or more inputs and a single output

Logic Gates

The behaviour of gates (and circuits) are commonly represented in any of the following ways:

Boolean Expressions
Uses Boolean algebra, a mathematical notation for expressing two-valued logic

Logic Diagrams
A graphical representation of a circuit; each gate has its own symbol

Truth Tables
A table showing all possible input values and the associated output values

The AND operation

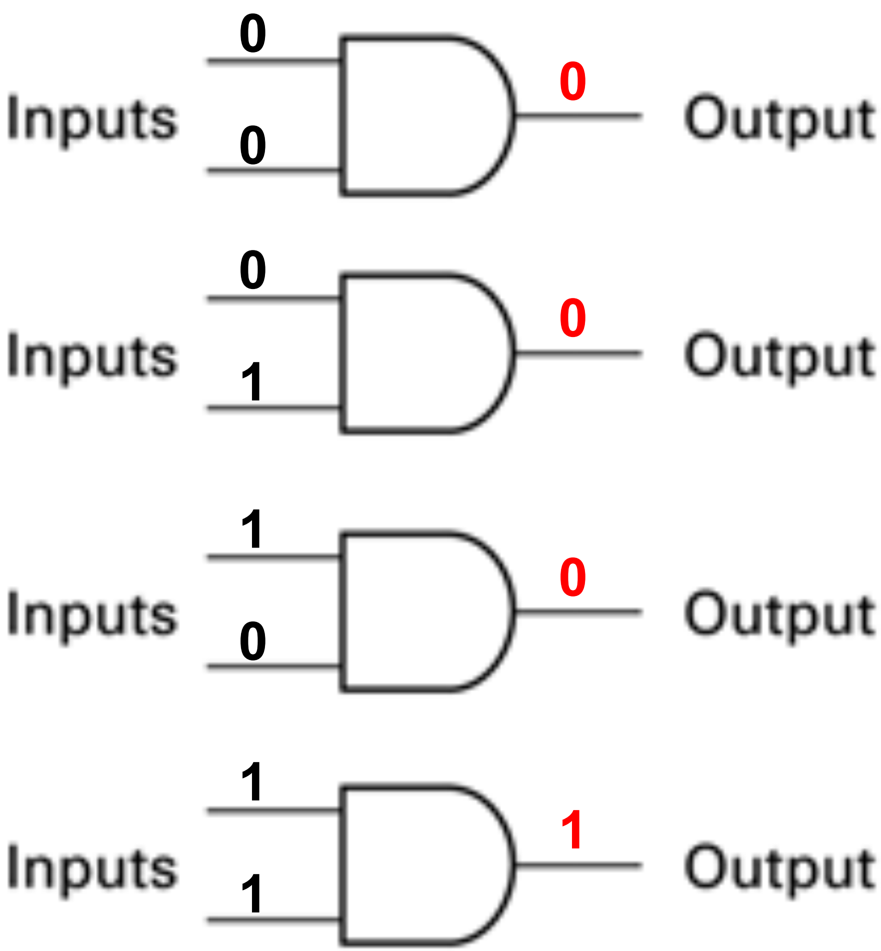
Logic Gate Symbol

A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table

In order for the output to be 1 both inputs must be 1

The AND operation



Truth Table

Inputs		**Output**
0	0	0
0	1	0
1	0	0
1	1	1

0 = FALSE

1 = TRUE

AND operation

- Both input values must be TRUE for output to be TRUE

Rick Graziani graziani@cabrillo.edu

The OR operation

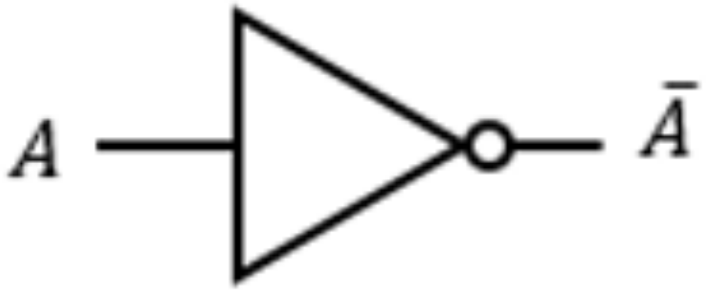
Logic Gate Symbol

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

Truth Table

In order for the output to be 1 either input must be 1.

The NOT operation



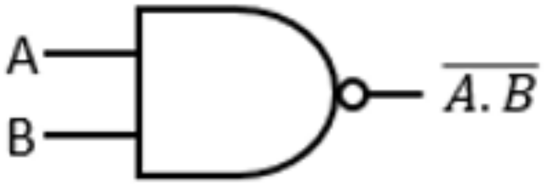
Logic Gate Symbol

A	$\bar{A}$
0	1
1	0

Truth Table

Inverts a single input. Also called an inverter.

NAND



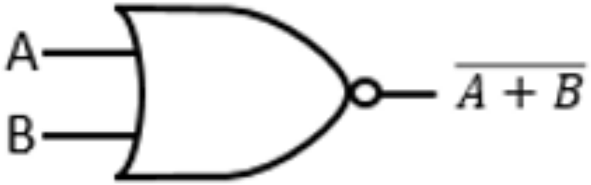
Logic Gate Symbol

A	B	$\overline{A.B}$
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table

A NAND B = NOT (A AND B)

NOR



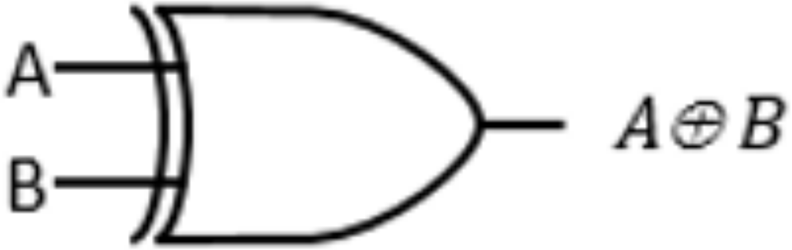
Logic Gate Symbol

A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

Truth Table

A NOR B = NOT (A OR B)

The XOR (eXclusive OR) operation



Logic Gate Symbol

A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	0

Truth Table

In order for the output to be 1 either (but not both) inputs must be 1.

(A AND NOT B) OR (NOT A AND B) = A XOR B

Boolean Algebra

Boolean Constants: these are ‘0’ (false) and ‘1’ (true)

Boolean Variables: variables that can only take the vales ‘0’ or ‘1’

Boolean Functions: such as NOT, AND and OR (in that order)

Boolean Theorems: a set of identities and laws

Law	AND	OR
Commutative	$A.B = B.A$	$A + B = B + A$
Associative	$A.(B.C) = (A.B).C$	$A + (B + C) = (A + B) + C$
Absorption	$A.(A + B) = A$	$A + (A.B) = A$
Distributive	$A.(B + C) = (A.B) + (B.C)$	$A + (B.C) = (A + B).(B + C)$
De Morgan’s Law	$\overline{A.B} = \bar{A} + \bar{B}$	$\overline{A + B} = \bar{A}.\bar{B}$

Complete the truth table for a Boolean expression

$A + \bar{B}$

A	B	$\bar{B}$	$A + \bar{B}$
0	0	1	1
0	1	0	0
1	0	1	1
1	1	0	1

Complete the truth table for a Boolean expression

$\overline{A.B}$

A	B	A.B	$\overline{A.B}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Using truth tables to verify identities

Let's say we wanted to investigate whether the identity holds

A	B	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$	A.B	$\overline{A.B}$
0	0	1	1	1	0	1
0	1	1	0	0	0	1
1	0	0	1	0	0	1
1	1	0	0	0	1	0

Is NOT A AND NOT B = NOT (A AND B) ?

Using truth tables to verify identities

Investigate whether the identity holds

$$A + (A.B) = A$$

A	B	$A.B$	$A + (A.B)$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Using truth tables to verify identities

A	B	C	B+C	$A.(B+C)$	$A.B$	$A.C$	$(A.B)+(B.C)$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

Distribution : $A.(B+C) = (A.B) + (A.C)$

Using truth tables to verify identities

A	B	C	B.C	$A + (B.C)$	$A + B$	$A + C$	$(A + B).(A + C)$
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

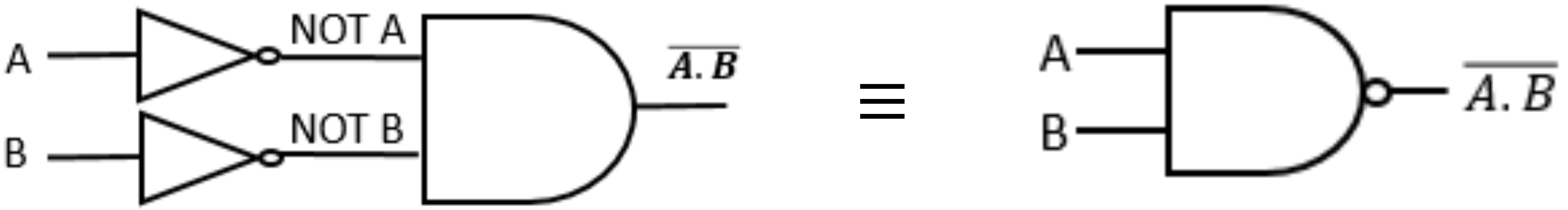
Distribution : $A + (B.C) = (A + B).(A + C)$

Using truth tables to verify identities - exercise

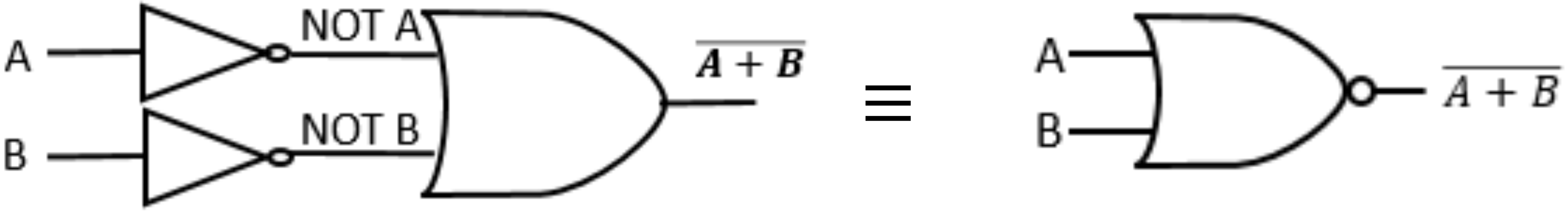
De Morgan's laws are used to simplify Boolean equations so that you can build equations only involving one sort of gate.

Verify De Morgan's laws using truth tables

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



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



Using truth tables to verify identities - SOLUTIONS

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

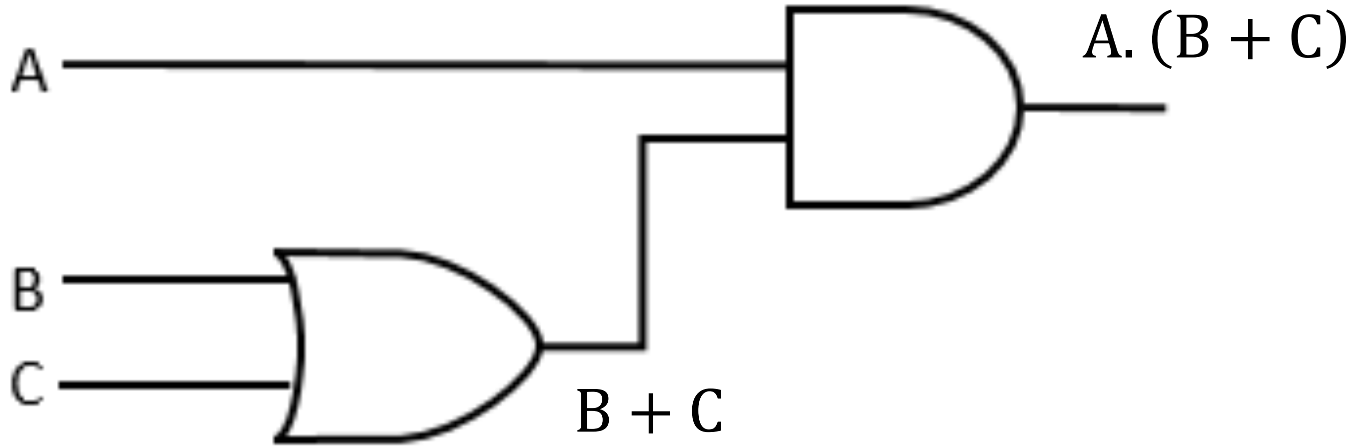
A	B	$A.B$	$\overline{A.B}$	$\bar{A}$	$\bar{B}$	$\bar{A} + \bar{B}$
0	0	0	1	1	1	1
0	1	0	1	1	0	1
1	0	0	1	0	1	1
1	1	1	0	0	0	0

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

A	B	$A + B$	$\overline{A + B}$	$\bar{A}$	$\bar{B}$	$\bar{A}.\bar{B}$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

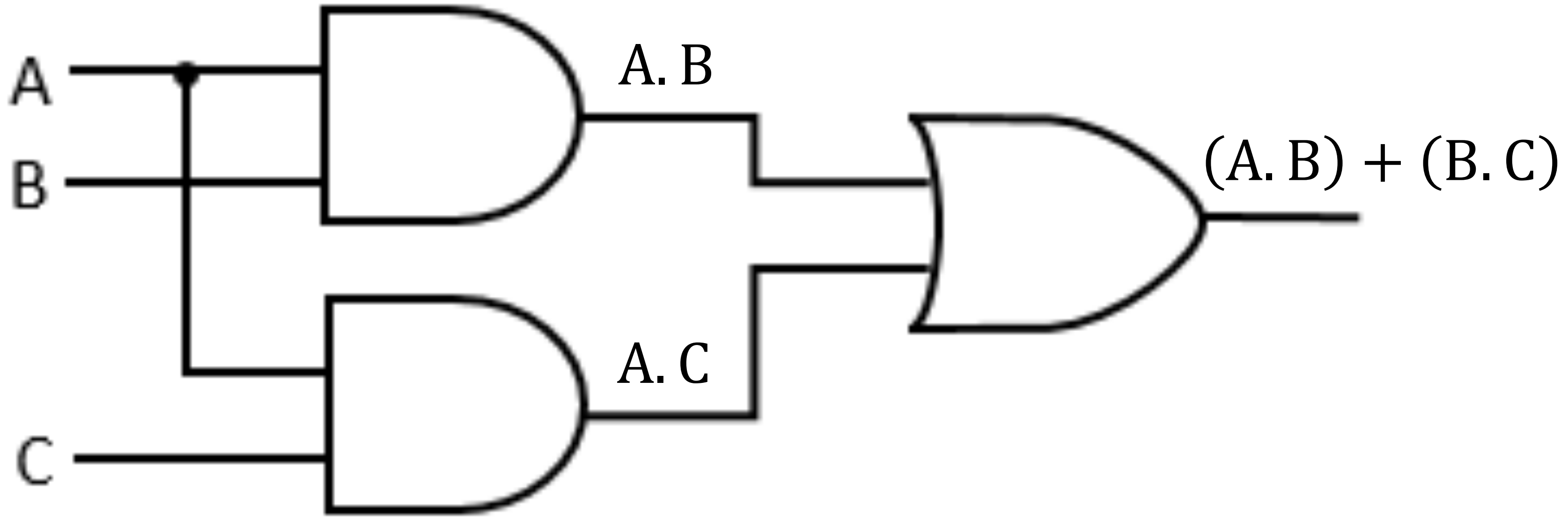
Create a Boolean Expression from a logic diagram

Work progressively from the inputs to the output adding logic expressions to the output of each gate in turn

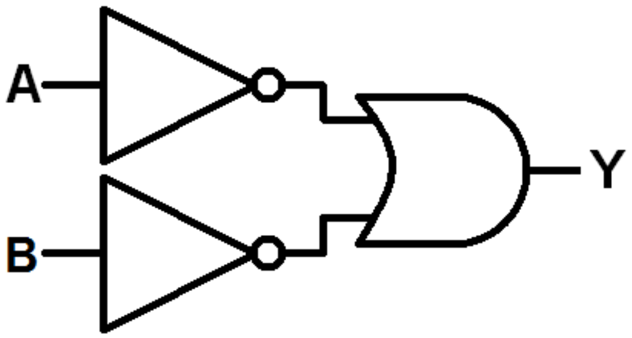


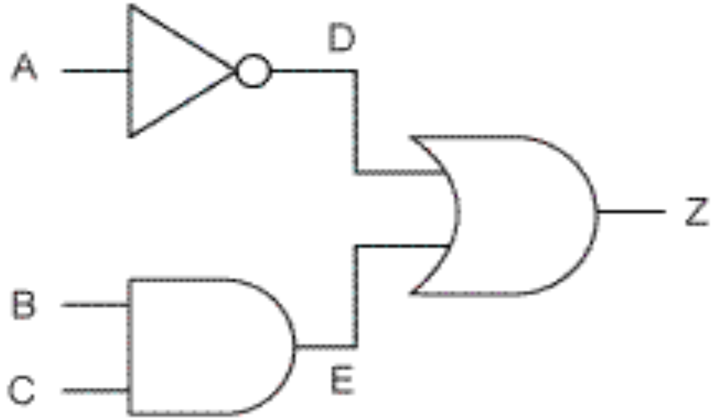
Create a Boolean Expression from a logic diagram

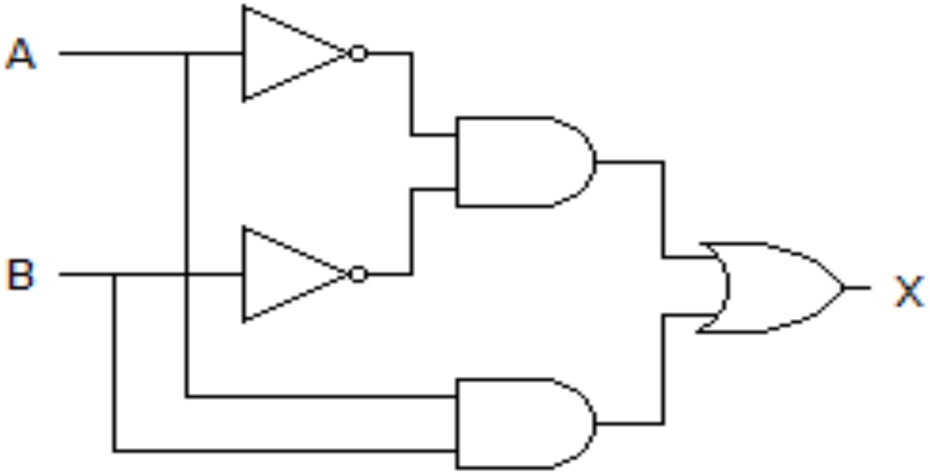
Work progressively from the inputs to the output adding logic expressions to the output of each gate in turn

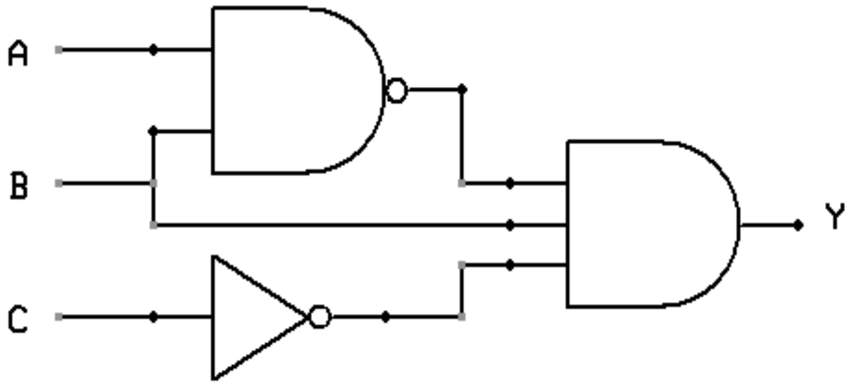


Create a Boolean Expression from a logic diagram







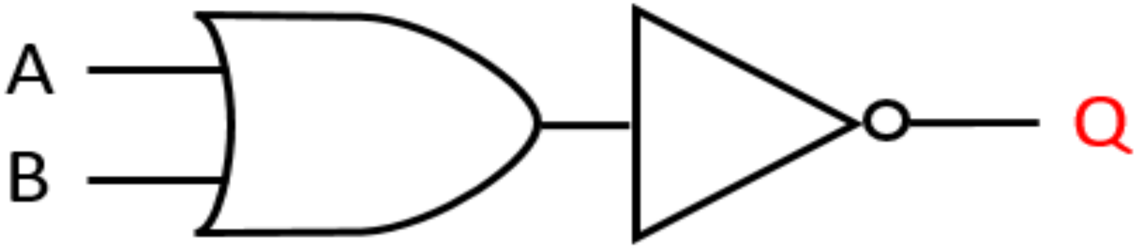


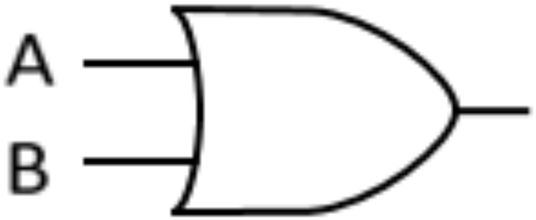
Connect Logic Gates (to create circuits)

Logic gates may be combined by using the output of one gate as the input to another.

A	B	A OR B	Q = NOT (A OR B)
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

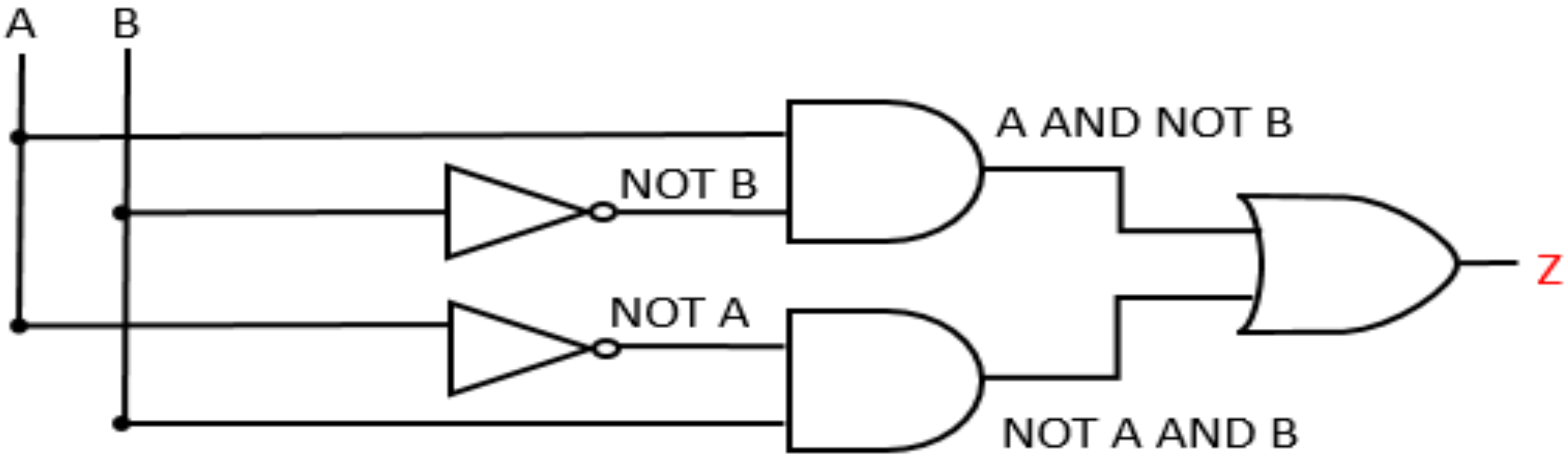
A NOR B
1
0
0
0



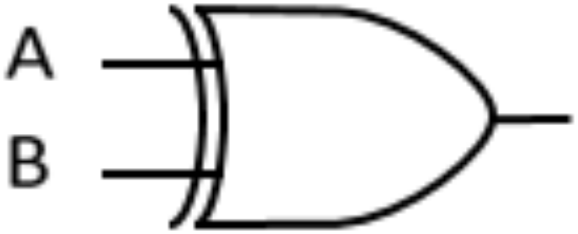


Connect Logic Gates (to create circuits)

Circuits in which the output is determined solely by the current inputs are termed **combinational logic circuits.**



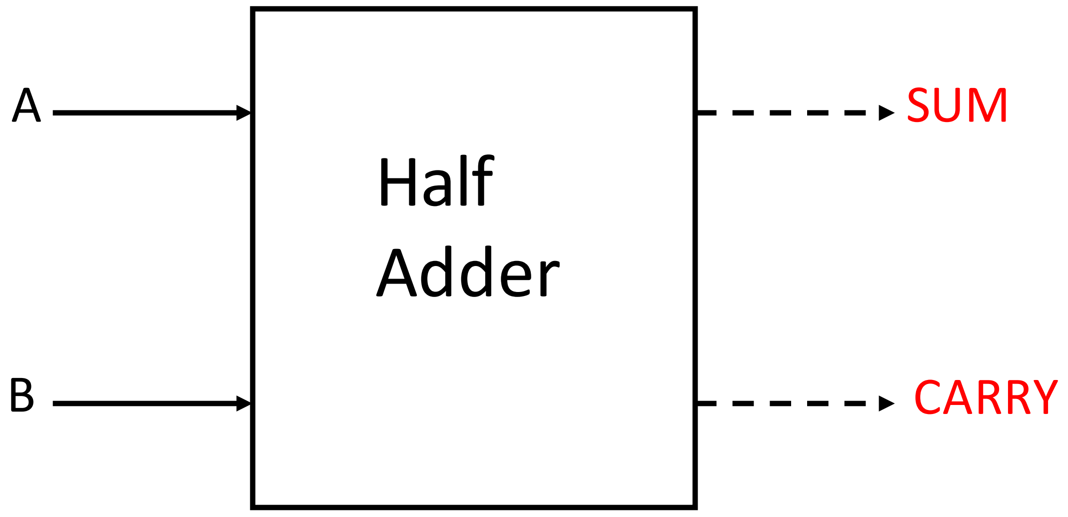
A AND NOT B OR NOT A AND B = A XOR B



Half-adder

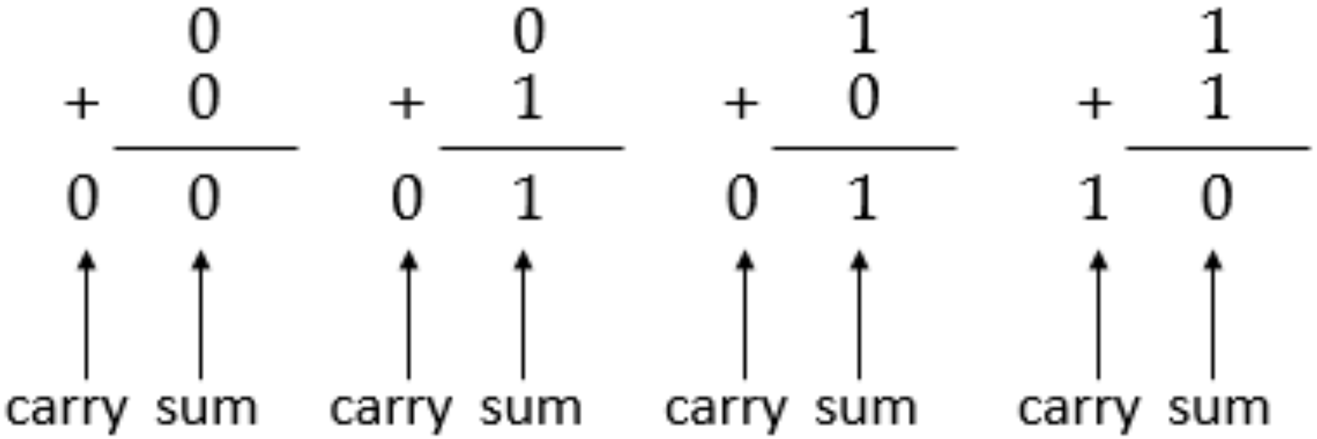
An adder is a digital circuit that performs addition of numbers.

A half adder adds two binary digits and produces two outputs – the sum and the carry



Half-adder

Only four possible combinations of input:



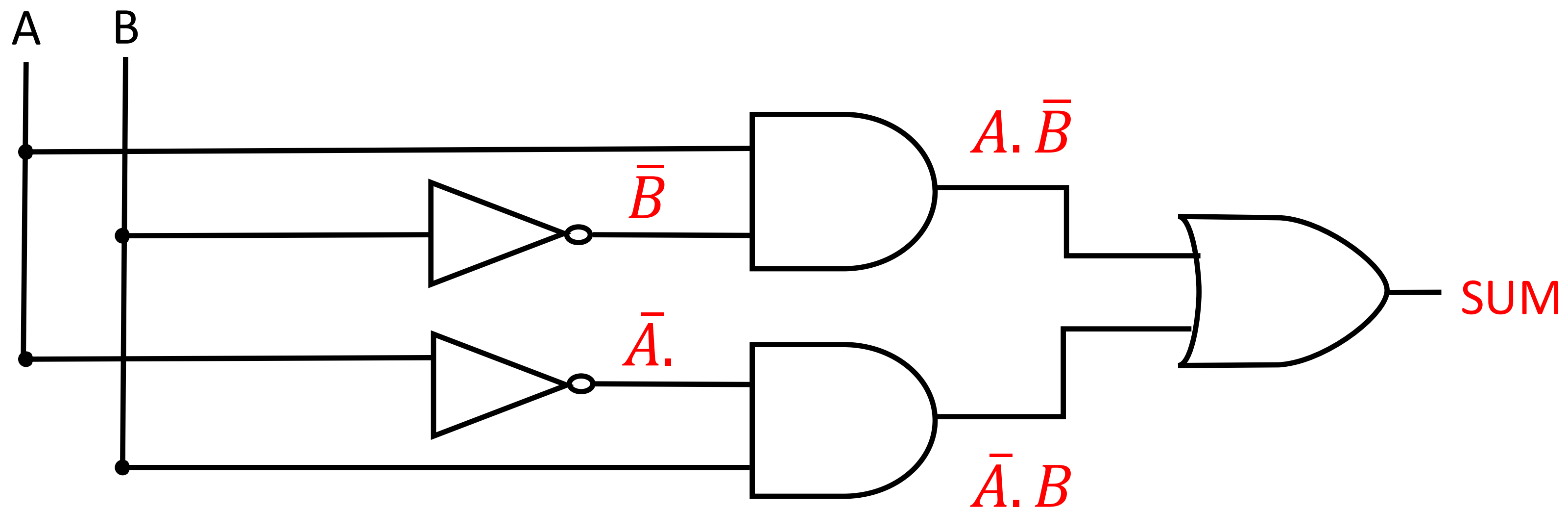
$$SUM = A.\bar{B} + \bar{A}.B$$

$$CARRY = A.B$$

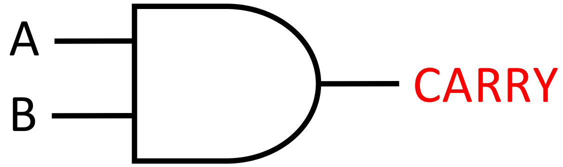
A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half-adder

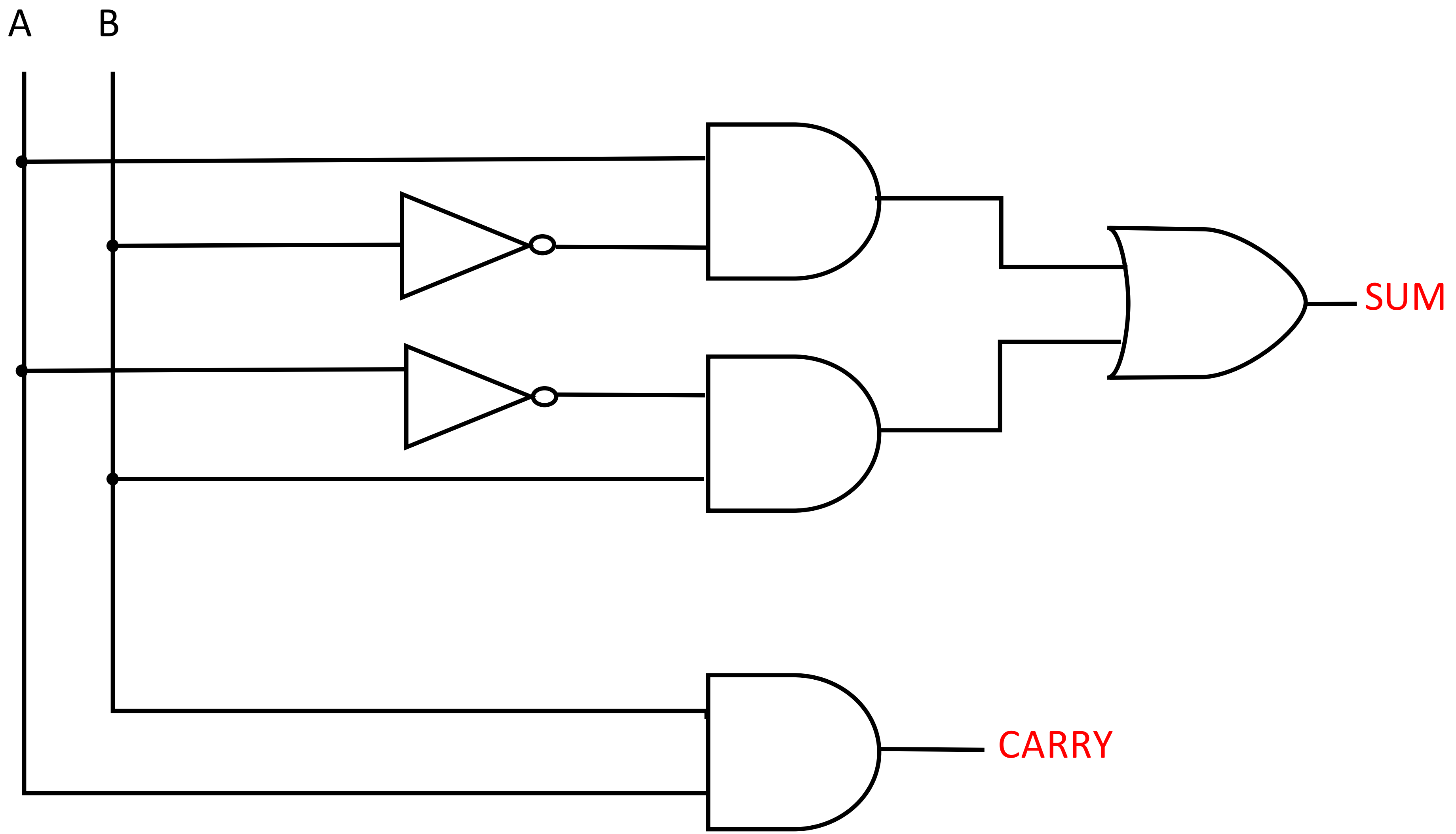
$$SUM = A.\bar{B} + \bar{A}.B$$



$$CARRY = A.B$$



Half-adder



Half-adder (refined)

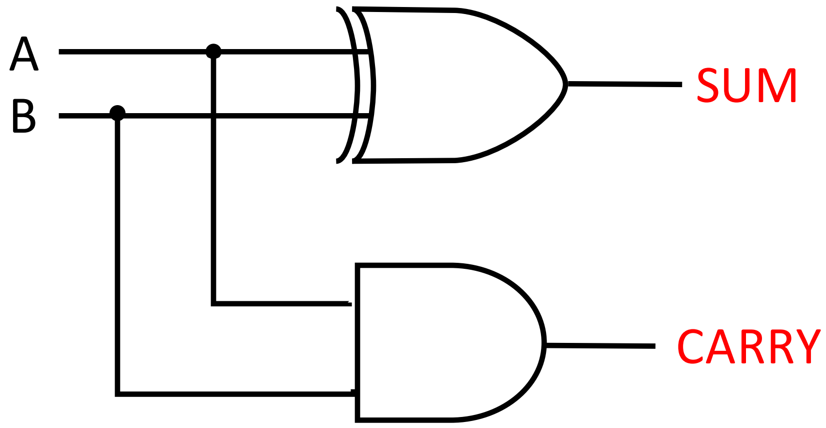
The circuit for SUM can be simplified as follows:

$$SUM = A.\bar{B} + \bar{A}.B$$

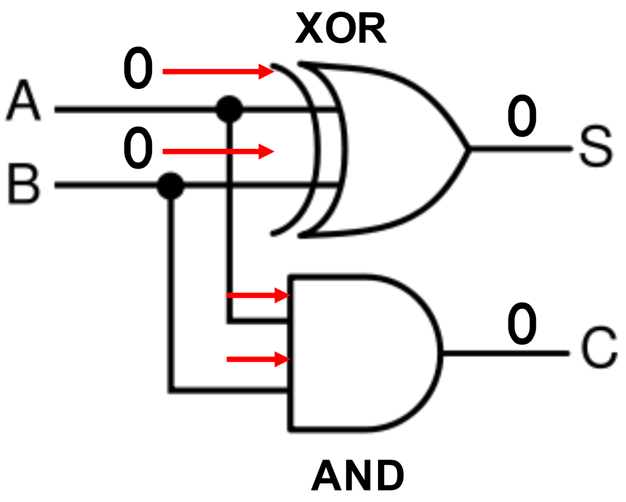
$$= A \oplus B$$ (i.e. A XOR B)

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

The half-adder can now be realised with fewer gates as shown

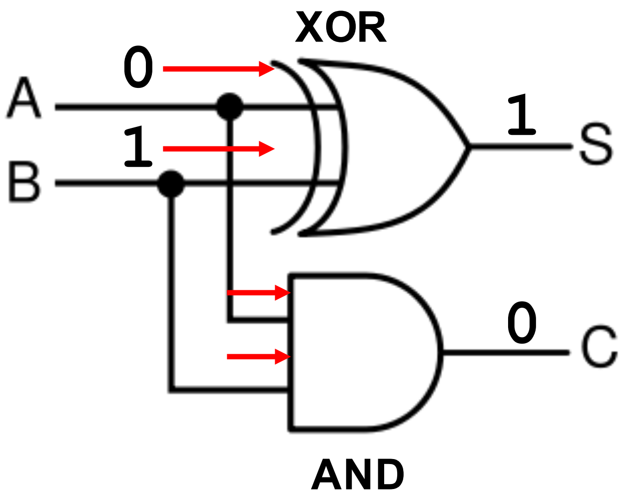


Half-adder (operation)



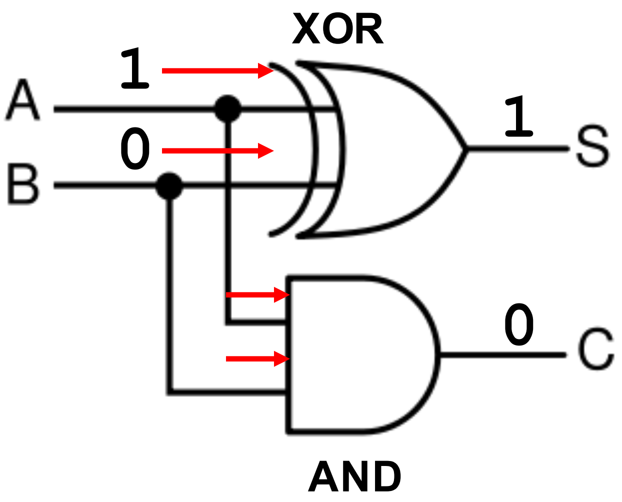
Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	0

Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1



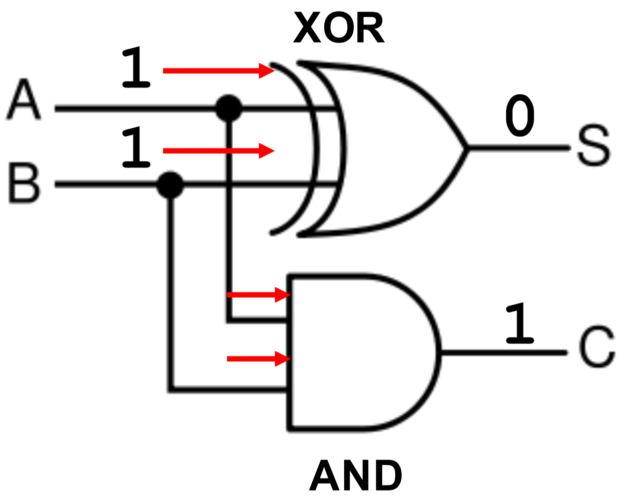
Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	0

Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1



Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	0

Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1

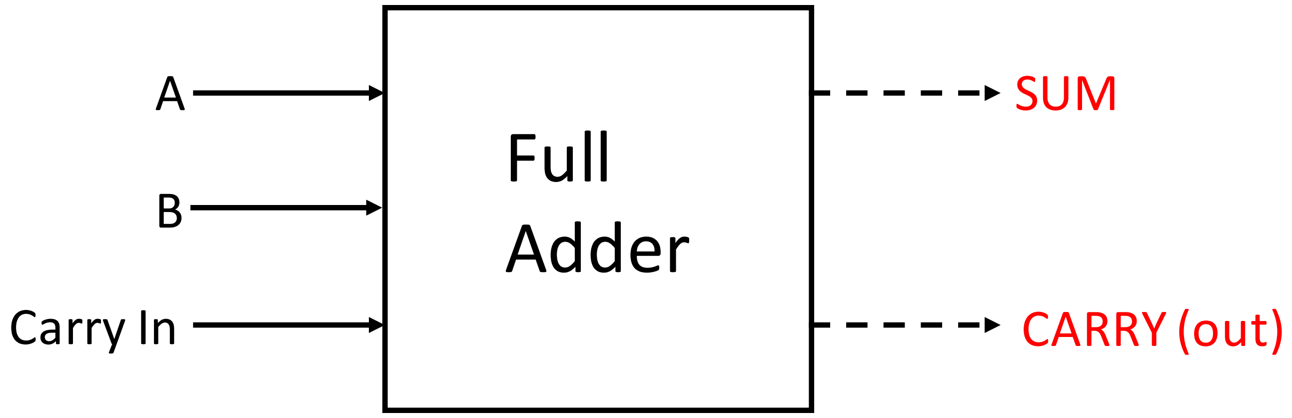


Inputs	Output
0 0	0
0 1	1
1 0	1
1 1	0

Inputs	Output
0 0	0
0 1	0
1 0	0
1 1	1

Full-adder

A full adder adds the carry in bit to two binary digits and produces two outputs – the sum and carry out.



Full-adder

A full adder adds the carry in bit to two binary digits and produces two outputs – the sum and carry out

A	B	Carry In	Sum	Carry Out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$SUM = \bar{A}.\bar{B}.C + \bar{A}.B.\bar{C} + A.\bar{B}.\bar{C} + A.B.C$$

$$CARRY = \bar{A}.B.C + A.\bar{B}.C + A.B.\bar{C} + A.B.C$$

Full-adder *(circuit for SUM)*

$$SUM = \bar{A}.\bar{B}.C + \bar{A}.B.\bar{C} + A.\bar{B}.\bar{C} + A.B.C$$

$$SUM = \bar{A}\,(\bar{B}.C + B.\bar{C}) + A(\bar{B}.\bar{C} + B.C)$$

$$\bar{B}.\bar{C} + B.C = \overline{B \oplus C}$$

$$\bar{B}.C + B.\bar{C} = B \oplus C$$

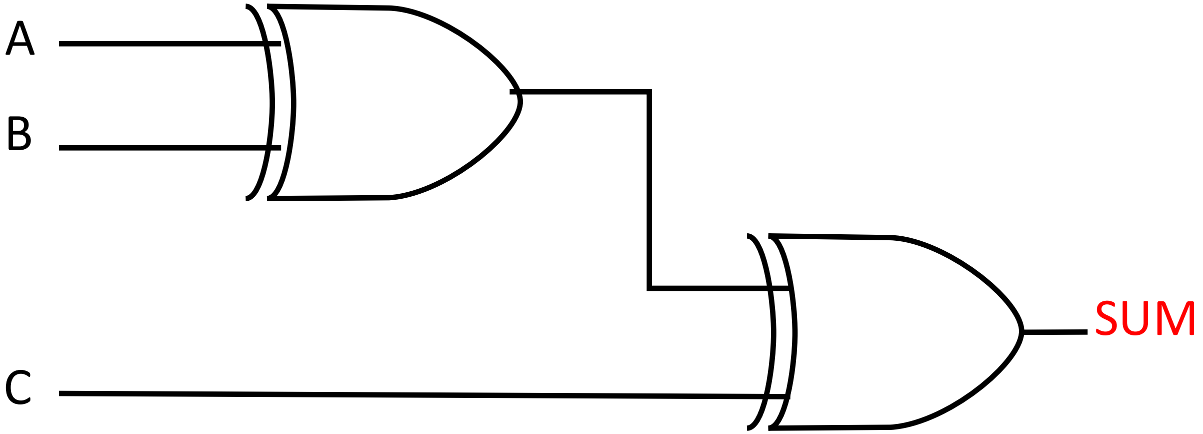
Taking,

$$X = B \oplus C$$

$$SUM = \bar{A}.X + A.\bar{X}$$

$$SUM = A \oplus X$$

$$SUM = A \oplus B \oplus C$$

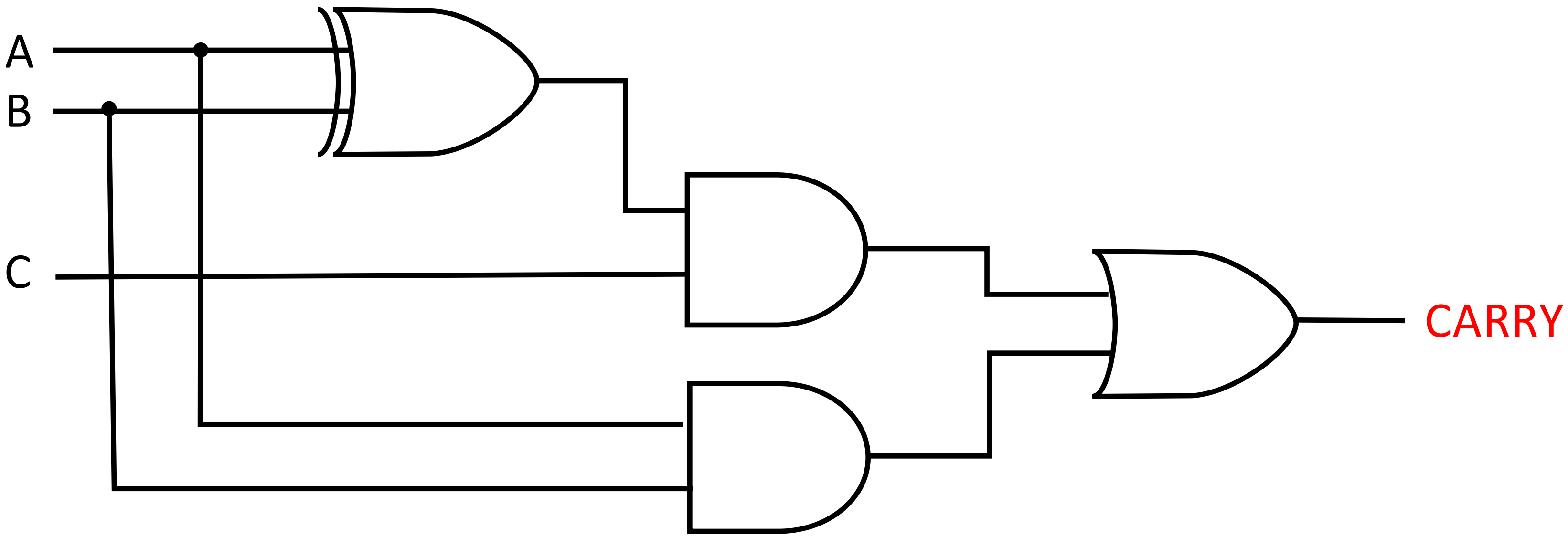


Full-adder *(circuit for CARRY)*

$$CARRY = \bar{A}.B.C + A.\bar{B}.C + A.B.\bar{C} + A.B.C$$

$$CARRY = C(\bar{A}.B + A.\bar{B}) + A.B(C + \bar{C})$$

$$CARRY = C(A\square B) + A.B$$



Full-adder

$SUM = A \oplus B \oplus C$

$CARRY = C.(A \oplus B) + A.B$