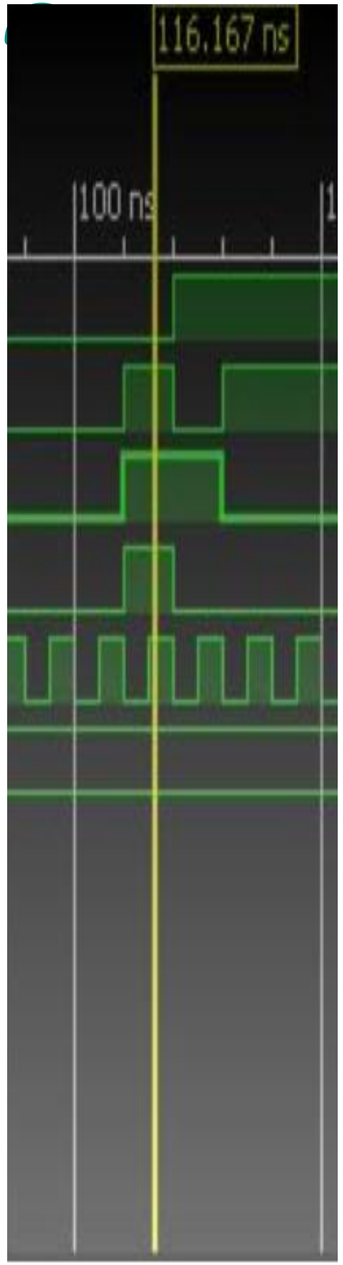
Combinational logic I

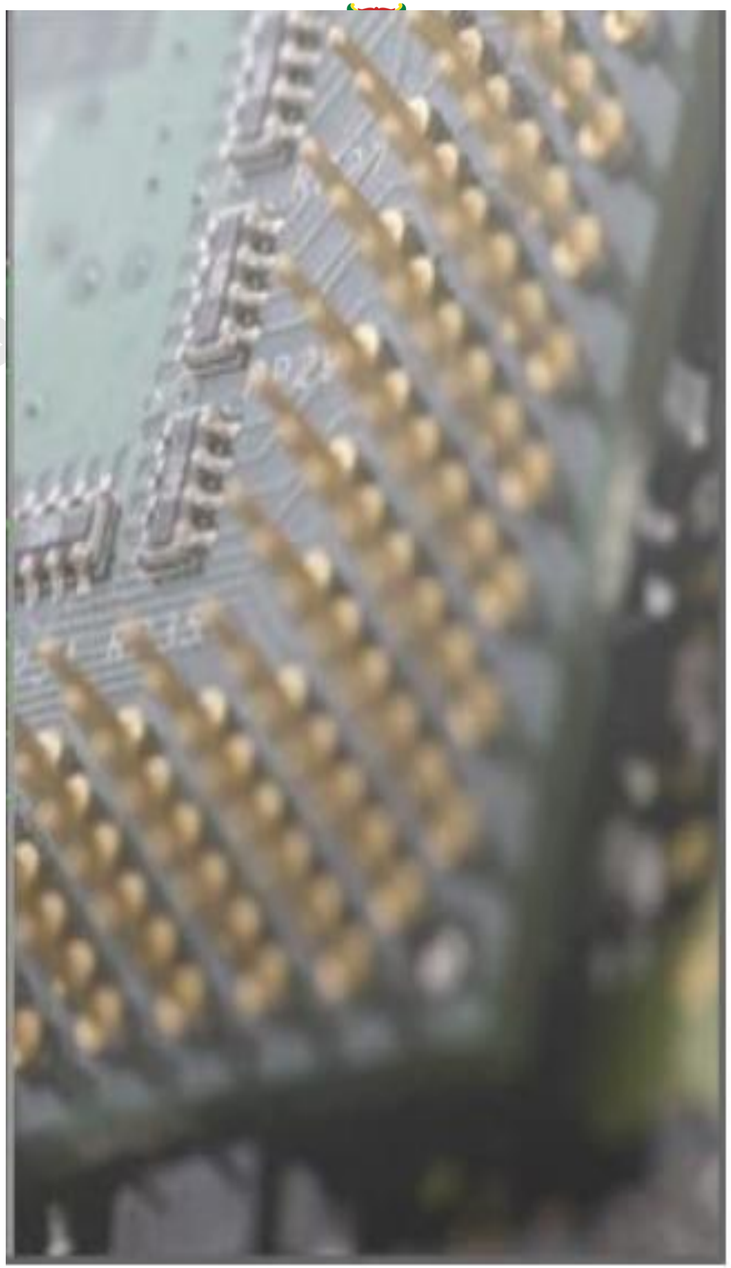
MC. Martin González Pérez

Agenda

- Logic gates
- Combinational logic
- Boolean equations
- Boolean algebra
- Karnaugh maps
- From logic to gates



Logic gates

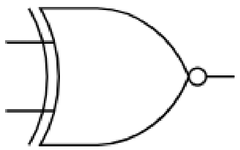


Logic gates

Logic gates are simple digital circuits that take one or more binary inputs to produce a binary output. Logic gates are represented with a symbol that shows the inputs and the outputs. Inputs are usually drawn on the left and outputs on the right.

The logic gate behavior can be described with a truth table or a Boolean equation. Truth table has one row for each possible combination of inputs at the left side and the corresponding output at right side. A Boolean equation is a mathematical expression using binary variables.

Symbol



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

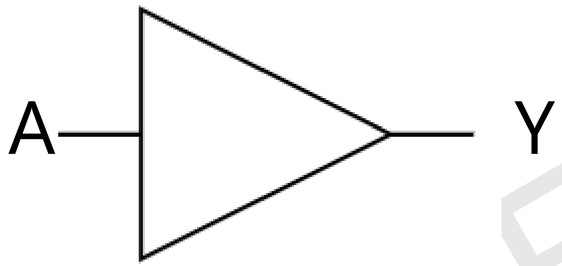
Truth table

Boolean equation

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

Single-input logic gates

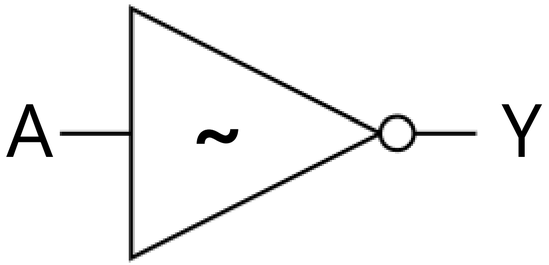
BUFF



$$Y = A$$

A	Y
0	0
1	1

NOT

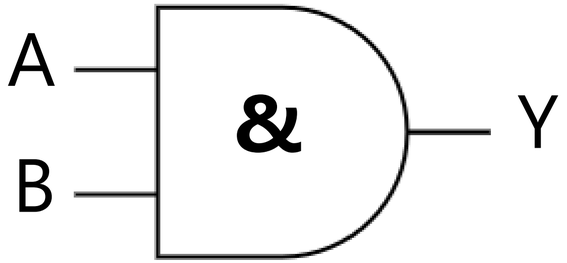


$$Y = \overline{A}$$

A	Y
0	1
1	0

Two-input logic gates

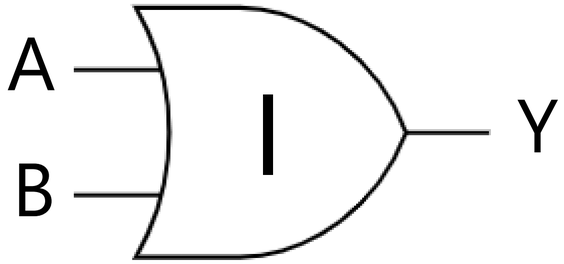
AND



$$Y = AB$$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR

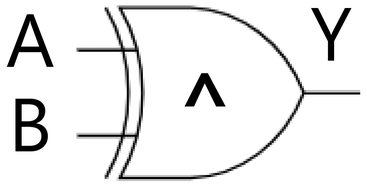


$$Y = A + B$$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Other logic gates

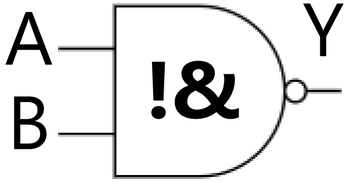
XOR



$Y = A \oplus B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

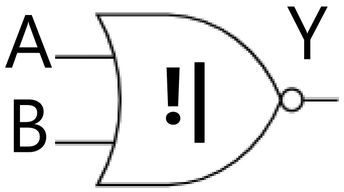
NAND



$Y = \overline{AB}$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

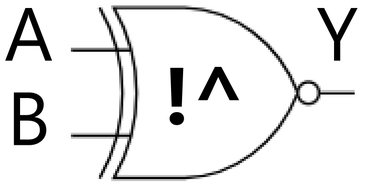
NOR



$Y = \overline{A + B}$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

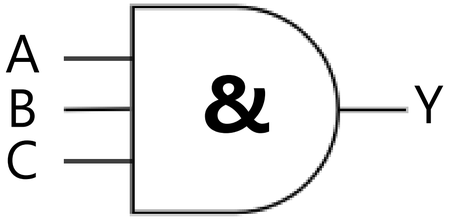
XNOR



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

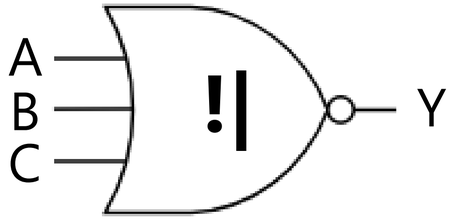
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Multiple-input logic gates



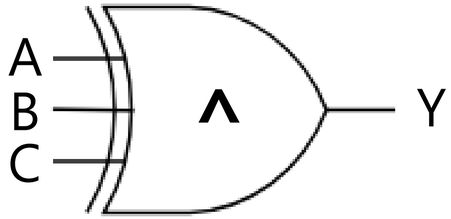
$$Y = ABC$$

A	B	C	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	



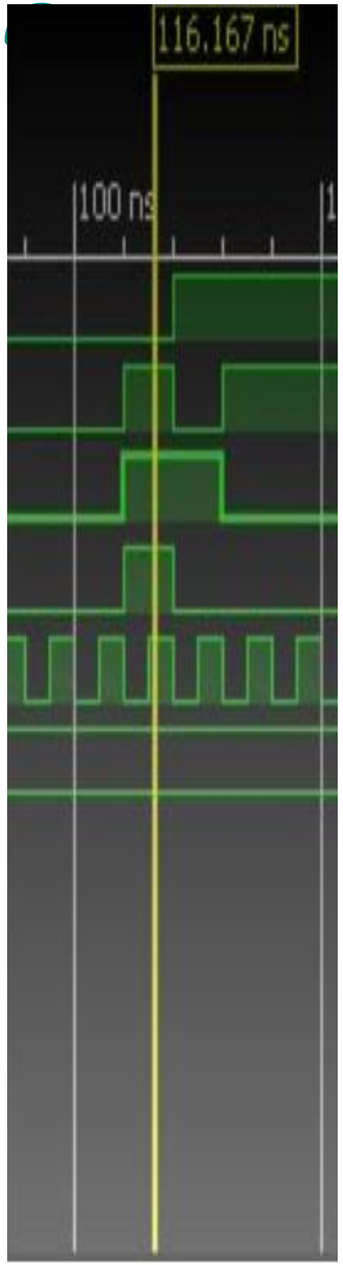
$$Y = \overline{A + B + C}$$

A	B	C	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

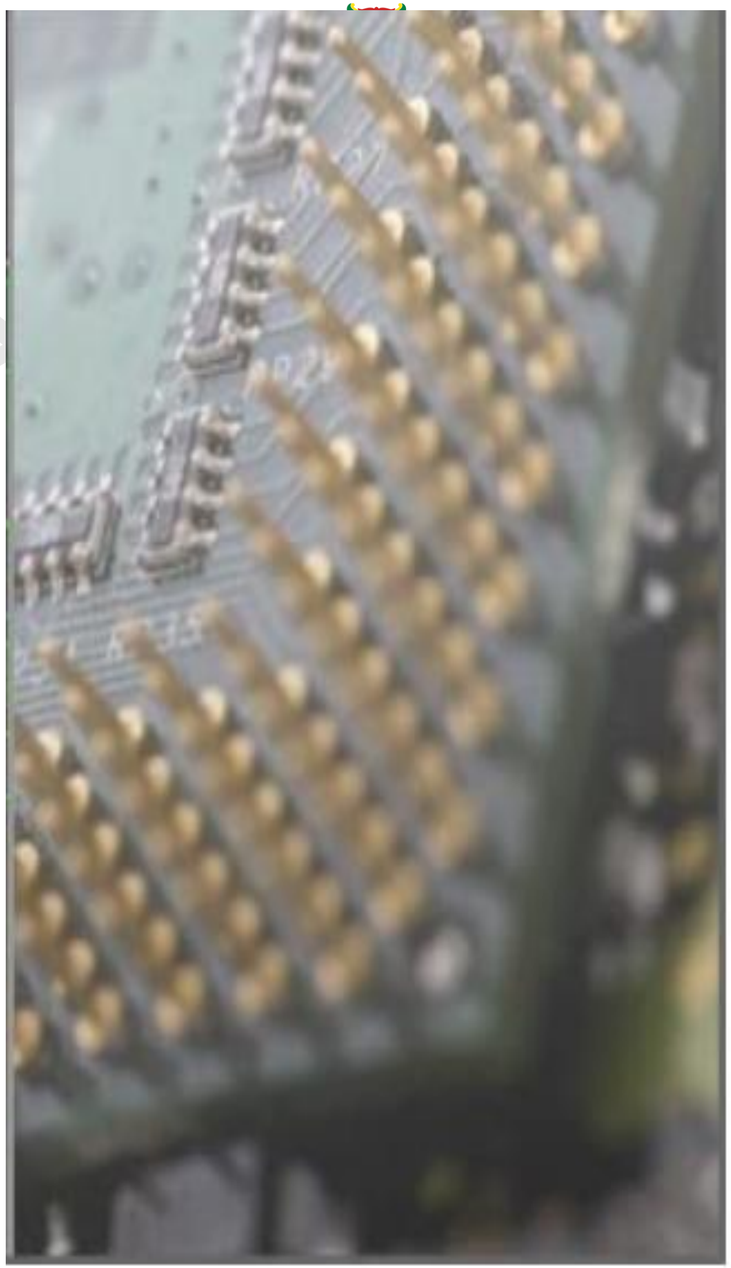


$$Y = A \oplus B \oplus C$$

A	B	C	Y
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	



Combinational logic



Logic circuits

A logic circuit is composed of:

- Inputs.
- Outputs.
- Functional specification (behavior).
- Timing specification.

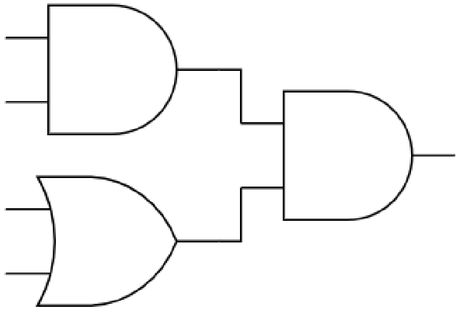
Type of circuits

- **Combinational logic**
  - Memoryless.
  - Output determined by current values of inputs.
- **Sequential logic**
  - Has memory.
  - Outputs determined by previous and current values of inputs.

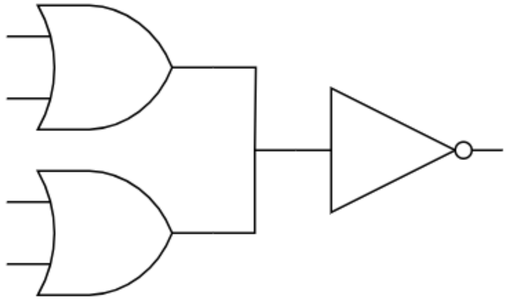
Combinational circuits

A circuit is combinational if:

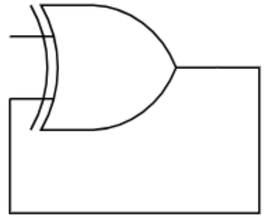
- Every circuit element is combinational.
- Every node designated as an input or connect to exactly one output.
- The circuit contain no cyclic paths.



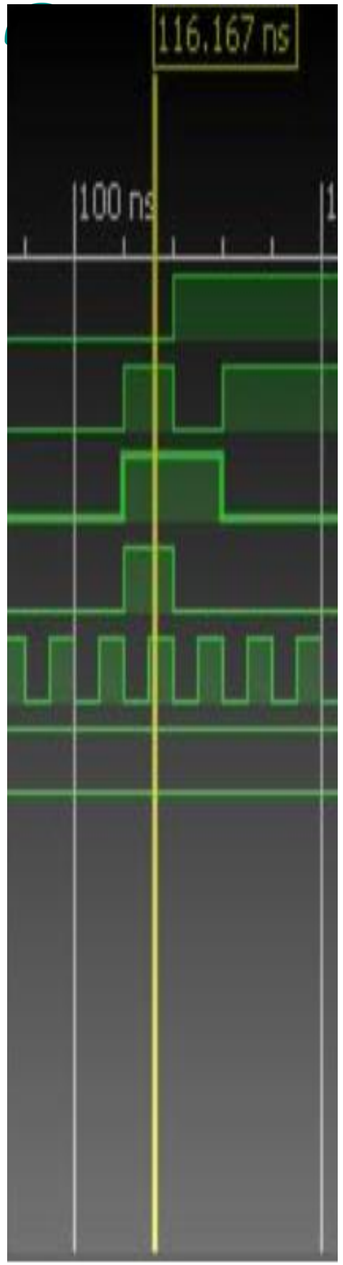
combinational



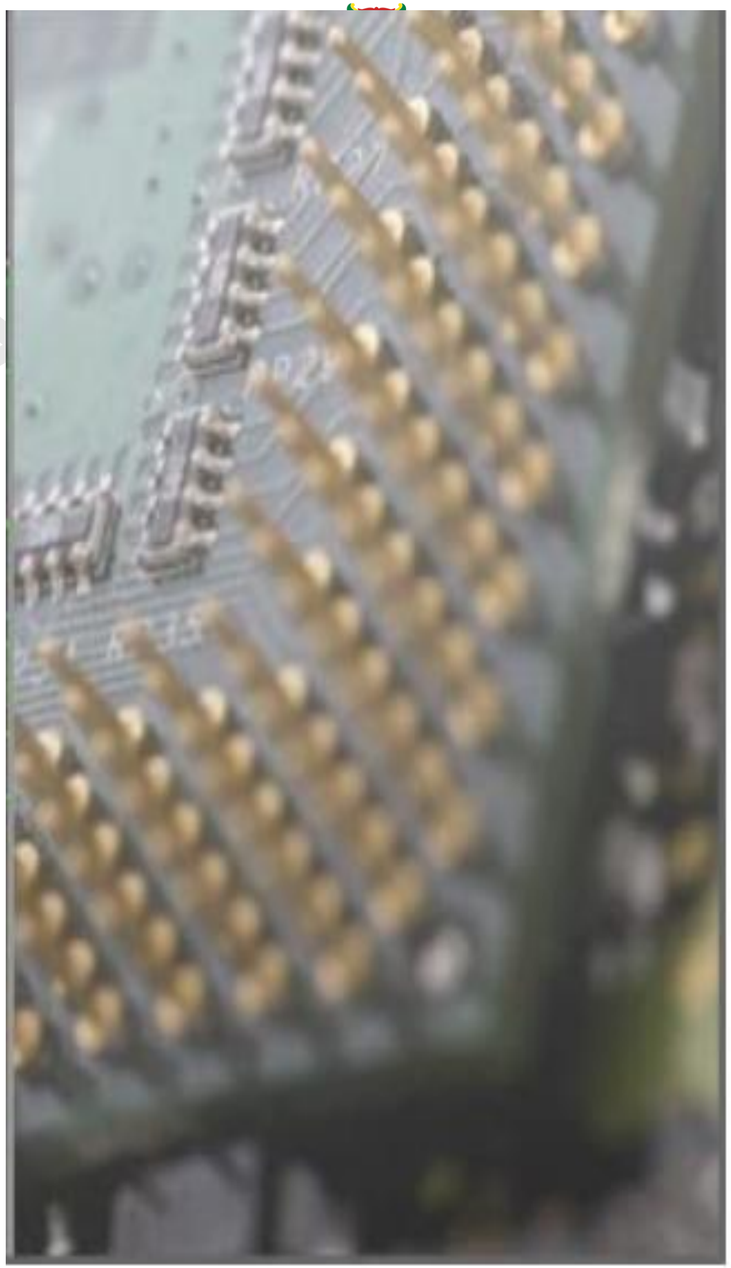
Multiple outputs conected to a single input



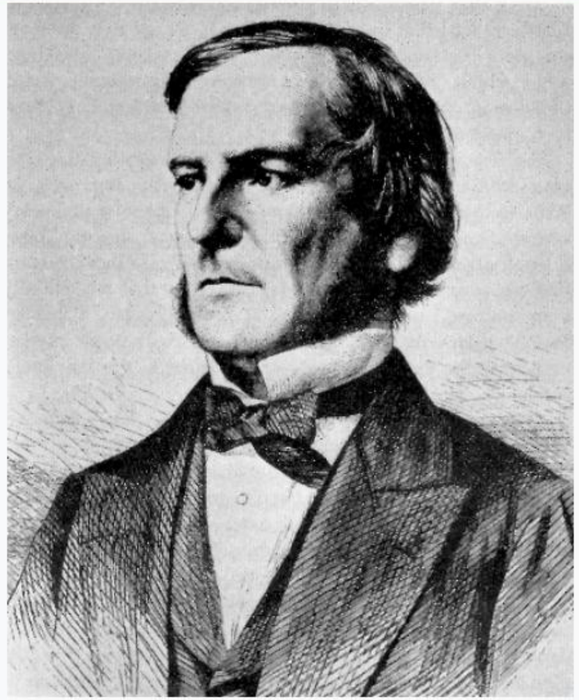
Cyclic path



Boolean equations



George Boole



George Boole (2 November 1815 – 8 December 1864) was a largely self-taught English mathematician, philosopher and logician, most of whose short career was spent as the first professor of mathematics at Queen's College, Cork in Ireland. He worked in the fields of differential equations and algebraic logic, and is best known as the author of *The Laws of Thought* (1854), which contains Boolean algebra.

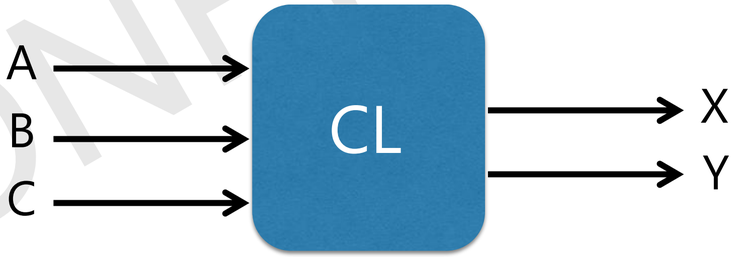
Boolean equations

Boolean equation describe outputs in terms of inputs.

For example:

X = F(A,B,C)

Y = F(A,B,C)



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

$$Y = ABC$$

Terminology

- **Complement:** Variable with a bar over it

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

- **Literal:** Variable or its complement

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

- **Minterm:** product that include all input variables

  $\overline{A}C\overline{B}, ABC$

- **Maxterm:** sum that include all input variables

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

Sum of products (SOP)

- All Boolean equation can be written in SOP form.
- Each row has a **minterm.**
- A minterm is **product** (AND) of literals.
- Each minterm is **true** for that row (only that row).
- Form the function by **adding** (OR) **minterms** where output is 1.

A	B	Y	minterm
0	0	**0**	$\bar{A}\bar{B}$
0	1	**1**	$\bar{A}B$
1	0	**0**	$A\bar{B}$
1	1	**1**	AB

Sum of products (SOP)

- All Boolean equation can be written in SOP form
- Each row has a **minterm**
- A minterm is **product** (AND) of literals.
- Each minterm is **true** for that row (only that row).
- Form the function by **adding** (OR) **minterms** where output is 1.

A	B	Y	minterm
0	0	**0**	$\bar{A}\bar{B}$
0	1	**1**	$\bar{A}B$
1	0	**0**	$A\bar{B}$
1	1	**1**	AB

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

Product of sums (POS)

- All Boolean equation can be written in POS form.
- Each row has a **maxterms.**
- A maxterms is **sum** (OR) of literals.
- Each maxterms is **False** for that row (only that row).
- Form the function by **adding** (OR) **maxterms** where output is 0.

A	B	Y	Maxterm
0	0	**0**	$A + B$
0	1	**1**	$A + \overline{B}$
1	0	**0**	$\overline{A} + B$
1	1	**1**	$\overline{A} + \overline{B}$

Product of sums (POS)

- All Boolean equation can be written in POS form.
- Each row has a **maxterms.**
- A maxterms is **sum** (OR) of literals.
- Each maxterms is **False** for that row (only that row).
- Form the function by **adding** (OR) **maxterms** where output is 0.

A	B	Y	Maxterm
0	0	**0**	$\boldsymbol{A + B}$
0	1	**1**	$\boldsymbol{A + \overline{B}}$
1	0	**0**	$\boldsymbol{\overline{A} + B}$
1	1	**1**	$\boldsymbol{\overline{A} + \overline{B}}$

$$Y = (A + B)\left(\overline{A} + B\right)$$

Exercises

- Get the SOP and POS of the next truth tables.

A	B	C	Y
0	0	0	**1**
0	0	1	**0**
0	1	0	**0**
0	1	1	**0**
1	0	0	**1**
1	0	1	**1**
1	1	0	**0**
1	1	1	**0**

A	B	C	Y
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**0**
1	0	0	**1**
1	0	1	**1**
1	1	0	**0**
1	1	1	**1**

Exercises

- Get the SOP and POS of the next truth tables.

A	B	C	Y	minterm	maxterm
0	0	0	**1**	$\bar{A}\bar{B}\bar{C}$	$A+B+C$
0	0	1	**0**	$\bar{A}\bar{B}C$	$A+B+\bar{C}$
0	1	0	**0**	$\bar{A}B\bar{C}$	$A+\bar{B}+C$
0	1	1	**0**	$\bar{A}BC$	$A+\bar{B}+\bar{C}$
1	0	0	**1**	$A\bar{B}\bar{C}$	$\bar{A}+B+C$
1	0	1	**1**	$A\bar{B}C$	$\bar{A}+B+\bar{C}$
1	1	0	**0**	$AB\bar{C}$	$\bar{A}+\bar{B}+C$
1	1	1	**0**	ABC	$\bar{A}+\bar{B}+\bar{C}$

$$Y = \bar{A}\bar{B}\bar{C}+A\bar{B}\bar{C}+A\bar{B}C$$

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

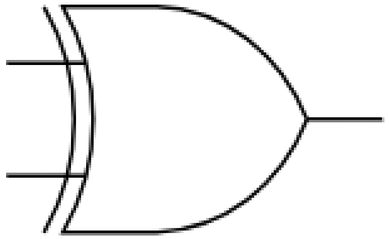
A	B	C	Y	minterm	maxterm
0	0	0	**1**	$\bar{A}\bar{B}\bar{C}$	$A+B+C$
0	0	1	**1**	$\bar{A}\bar{B}C$	$A+B+\bar{C}$
0	1	0	**1**	$\bar{A}B\bar{C}$	$A+\bar{B}+C$
0	1	1	**0**	$\bar{A}BC$	$A+\bar{B}+\bar{C}$
1	0	0	**1**	$A\bar{B}\bar{C}$	$\bar{A}+B+C$
1	0	1	**1**	$A\bar{B}C$	$\bar{A}+B+\bar{C}$
1	1	0	**0**	$AB\bar{C}$	$\bar{A}+\bar{B}+C$
1	1	1	**1**	ABC	$\bar{A}+\bar{B}+\bar{C}$

$$Y = \bar{A}\bar{B}\bar{C}+\bar{A}\bar{B}C+\bar{A}B\bar{C}+A\bar{B}\bar{C}+A\bar{B}C+ABC$$

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

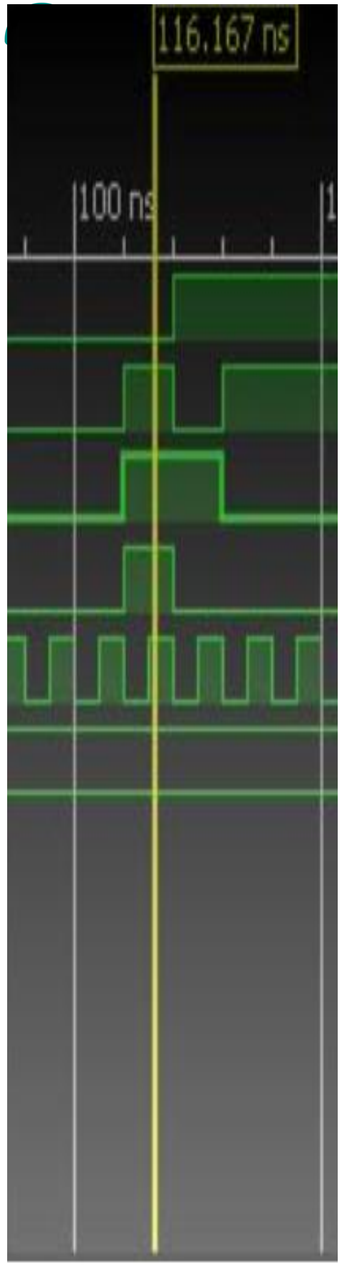
XOR Equation

XOR logic gate can be represented in Boolean equations as $\oplus$

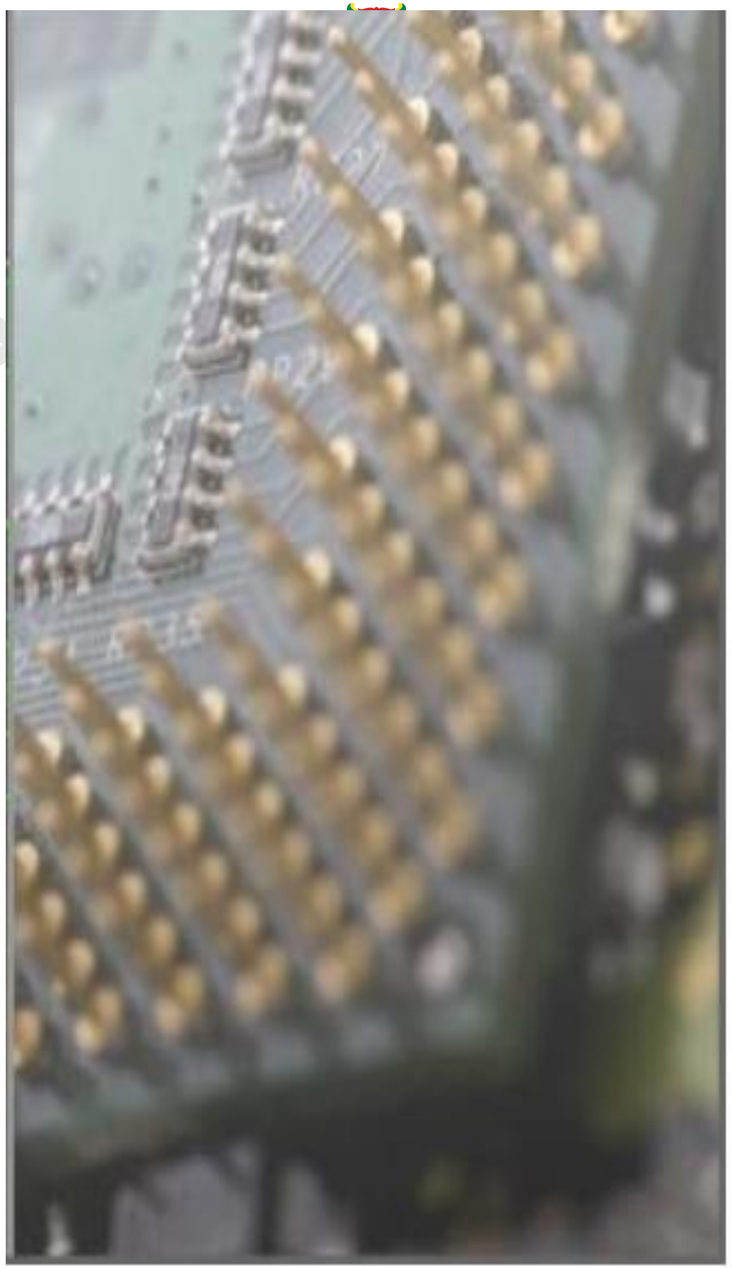


A	B	Y	maxterm	minterm
0	0	**0**	$\boldsymbol{A + B}$	$\boldsymbol{\overline{A}\,\overline{B}}$
0	1	**1**	$\boldsymbol{A + \overline{B}}$	$\boldsymbol{\overline{A}B}$
1	0	**1**	$\boldsymbol{\overline{A} + B}$	$\boldsymbol{A\overline{B}}$
1	1	**0**	$\boldsymbol{\overline{A} + \overline{B}}$	$\boldsymbol{AB}$

$$Y = \overline{A}\mathrm{B} + A\overline{B} = A \oplus \mathrm{B}$$



Boolean algebra



Boolean algebra

Now we know how to write Boolean expressions given a truth table. However, that expression does not necessarily lead to the simplest set of logic gates. Just like algebra is used to simplify equations, Boolean algebra is used to simplify Boolean equations.

Boolean Axioms

Boolean algebra is based on a set of **axioms**. Axioms and theorems obey the principle of **duality**. If symbols **0** and **1** and the operators •(**AND)** and +(**OR)** are interchanged, the statement still will be correct.

Number	Axiom	Dual	Name
A1	$B = 0$ *if* $B \neq 1$	$B = 1$ *if* $B \neq 0$	Binary Field
A2	$\overline{0} = 1$	$\overline{1} = 0$	NOT
A3	$0 \cdot 0 = 0$	$1 + 1 = 1$	AND/OR
A4	$1 \cdot 1 = 1$	$0 + 0 = 0$	AND/OR
A5	$0 \cdot 1 = 1 \cdot 0 = 0$	$1 + 0 = 0 + 1 = 1$	AND/OR

Theorems of one variable

Number	Theorem	Dual	Name
T1	$B \cdot 1 = B$	$B + 0 = B$	Identity
T2	$B \cdot 0 = 0$	$B + 1 = 1$	Null Element
T3	$B \cdot B = B$	$B + B = B$	Idempotency
T4	$\bar{\bar{B}} = B$		Involution
T5	$B \cdot \bar{B} = 0$	$B + \bar{B} = 1$	Complements

Theorems of several variables

Number	Theorem	Dual	Name
T6	$B \cdot C = C \cdot B$	$B + C = C + B$	Commutativity
T7	$(B \cdot C) \cdot D = B \cdot (C \cdot D)$	$(B + C) + D = B + (C + D)$	Associativity
T8	$B \cdot (C + D) = (B \cdot C) + (B \cdot D)$	$B + (C \cdot D) = (B + C)\,(B + D)$	Distributivity
T9	$B \cdot (B + C) = B$	$B + (B \cdot C) = B$	Covering
T10	$(B \cdot C) + (B \cdot \bar{C}) = B$	$(B + C) \cdot (B + \bar{C}) = B$	Combining
T11	$(B \cdot C) + (\bar{B} \cdot D) + (C \cdot D) = (B \cdot C) + (\bar{B} \cdot D)$	$(B + C) \cdot (\bar{B} + D) \cdot (C + D) = (B + C) \cdot (\bar{B} + D)$	Consensus

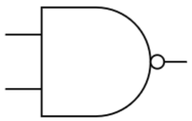
Warning: Dual T8 differs from traditional algebra:
OR (+) distributes over AND (•)

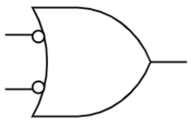
De Morgan’s theorem

De Morgan’s theorem is a particularly powerful tool in digital design. The theorem explain that The **complement** of the **product** is the **sum** of the **complements**. Likewise, the **complement** of the **sum** is the **product** of the **complements**.

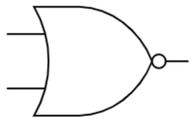
According to De Morgan’s theorem, NAND gate is equivalent to OR gate with inverted inputs. Similarly, NOR gate is equivalent to AND gate with inverted inputs.

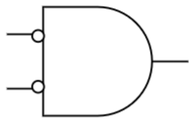
Number	Theorem	Dual	Name
T12	$\overline{B \bullet C \bullet D \ldots} = \overline{B}+\overline{C}+\overline{D}\ldots$	$\overline{B+C+D\ldots} = \overline{B}\bullet\overline{C}\bullet\overline{D}\ldots$	De Morgan’s Theorem





A	B	Y
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**





A	B	Y
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

Simplifying Boolean equations

$\mathbf{Y = \bar{A}B + AB}$

Y = B T10: Combining

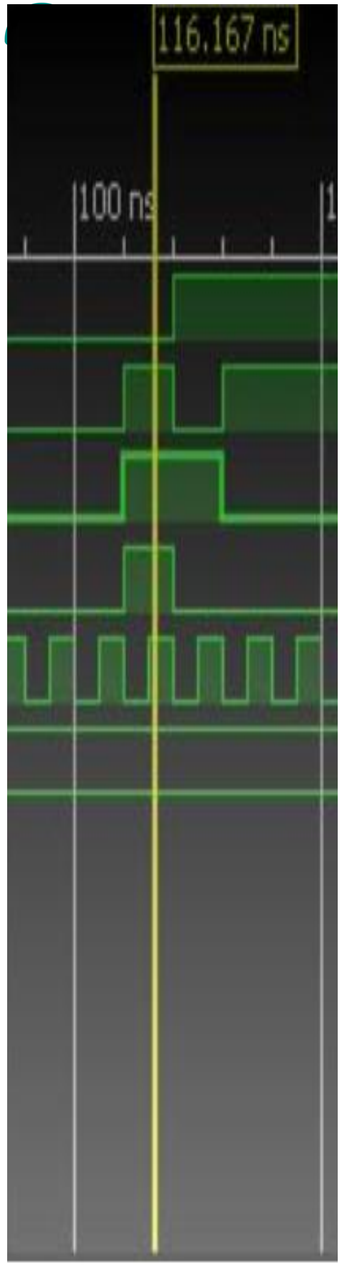
or

Y = B(A + $\bar{A}$) T8: Distributivity

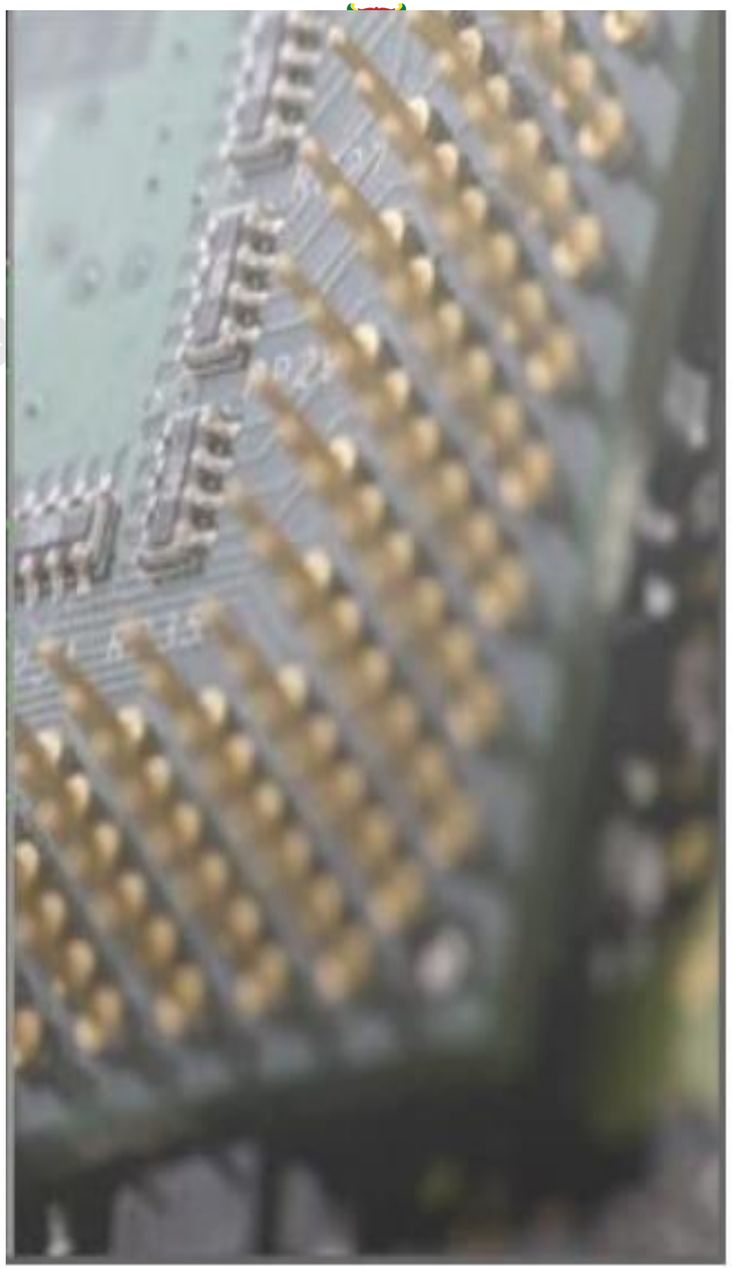
= B(1) T5′: Complements

= B T1: Identity

#	Theorem	Dual	Name
T1	$B \bullet 1 = B$	$B + 0 = B$	Identity
T2	$B \bullet 0 = 0$	$B + 1 = 1$	Null Element
T3	$B \bullet B = B$	$B + B = B$	Idempotency
T4	$\bar{\bar{B}} = B$		Involution
T5	$B \bullet \bar{B} = 0$	$B + \bar{B} = 1$	Complements
T6	$B \bullet C = C \bullet B$	$B + C = C + B$	Commutativity
T7	$(B \bullet C) \bullet D = B \bullet (C \bullet D)$	$(B + C) + D = B + (C + D)$	Associativity
T8	$B \bullet (C + D) = (B \bullet C) + (B \bullet D)$	$B + (C \bullet D) = (B + C)\,(B + D)$	Distributivity
T9	$B \bullet (B + C) = B$	$B + (B \bullet C) = B$	Covering
T10	$(B \bullet C) + (B \bullet \bar{C}) = B$	$(B + C) \bullet (B + \bar{C}) = B$	Combining
T11	$(B \bullet C) + (\bar{B} \bullet D) + (C \bullet D) = (B \bullet C) + (\bar{B} \bullet D)$	$(B + C) \bullet (\bar{B} + D) \bullet (C + D) = (B + C) \bullet (\bar{B} + D)$	Consensus
T12	$\overline{B \bullet C \bullet D \ldots} = \bar{B} + \bar{C} + \bar{D} \ldots$	$\overline{B + C + D \ldots} = \bar{B} \bullet \bar{C} \bullet \bar{D} \ldots$	De Morgan's Theorem



Karnaugh maps



Karnaugh maps

Karnaugh maps (k-maps) are a graphical method for simplifying Boolean equations. This method is useful when inputs are 3 or more. Figure show a 3 inputs k-map.

A	B	C	Y
0	0	0	**0**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**0**

C \ AB	00	01	11	10
0	0	1	0	0
1	0	1	0	0

C \ AB	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{\boldsymbol{A}}\boldsymbol{B}\bar{\boldsymbol{C}}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{\boldsymbol{A}}\boldsymbol{BC}$	ABC	$A\bar{B}C$

Karnaugh maps

We can extract mintems in true table and reduce the equation using Boolean algebra. But K-maps help us do this graphically circling 1's in adjacent squares, we conserve terms in circles such that values doesn't change.

A	B	C	Y
0	0	0	**0**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**0**

C \ AB	00	01	11	10
0	0	1	0	0
1	0	1	0	0

C \ AB	00	01	11	10
0	$\bar{A}\bar{B}\bar{C}$	$\bar{\boldsymbol{A}}\boldsymbol{B}\bar{\boldsymbol{C}}$	$AB\bar{C}$	$A\bar{B}\bar{C}$
1	$\bar{A}\bar{B}C$	$\bar{\boldsymbol{A}}\boldsymbol{BC}$	ABC	$A\bar{B}C$

$$Y = \bar{A}B\bar{C} + \bar{A}BC = \bar{A}B$$

C change

K-maps rules (for minterms)

- Every 1 must be circled at least once
- Each circle must span a power of 2 (i.e. 1, 2, 4) squares in each direction
- Each circle must be as large as possible
- A circle may wrap around the edges
- A 1 may be circled multiple times if doing so allows fewer circles to be used.

Exercise

- Get a simplified expression given the truth table

CD \ AB	00	01	11	10
00				
01				
11				
10				

A	B	C	D	Y
0	0	0	0	**1**
0	0	0	1	**0**
0	0	1	0	**1**
0	0	1	1	**1**
0	1	0	0	**0**
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**1**
1	0	0	0	**1**
1	0	0	1	**1**
1	0	1	0	**1**
1	0	1	1	**0**
1	1	0	0	**0**
1	1	0	1	**0**
1	1	1	0	**0**
1	1	1	1	**0**

Exercise

- Get a simplified expression given the truth table

CD \ AB	00	01	11	10
00	1	0	0	1
01	0	1	0	1
11	1	1	0	0
10	1	1	0	1

$$Y = \bar{A}C + \bar{B}\bar{D} + \bar{A}BD + A\bar{B}\bar{C}$$

A	B	C	D	Y
0	0	0	0	**1**
0	0	0	1	**0**
0	0	1	0	**1**
0	0	1	1	**1**
0	1	0	0	**0**
0	1	0	1	**1**
0	1	1	0	**1**
0	1	1	1	**1**
1	0	0	0	**1**
1	0	0	1	**1**
1	0	1	0	**1**
1	0	1	1	**0**
1	1	0	0	**0**
1	1	0	1	**0**
1	1	1	0	**0**
1	1	1	1	**0**

Don’t care values

Don’t care values, as we saw, are input values that help us to simplify truth tables. In k-maps don’t care values should be circled only if it helps to minimize the equation (make bigger circles).

Exercise

- Get a simplified expression given the truth table

CD \ AB	00	01	11	10
00				
01				
11				
10				

A	B	C	D	Y
0	0	0	0	**1**
0	0	0	1	**0**
0	0	1	0	**1**
0	0	1	1	**1**
0	1	0	0	**0**
0	1	0	1	**X**
0	1	1	0	**1**
0	1	1	1	**1**
1	0	0	0	**1**
1	0	0	1	**1**
1	0	1	0	**X**
1	0	1	1	**X**
1	1	0	0	**X**
1	1	0	1	**X**
1	1	1	0	**X**
1	1	1	1	**X**

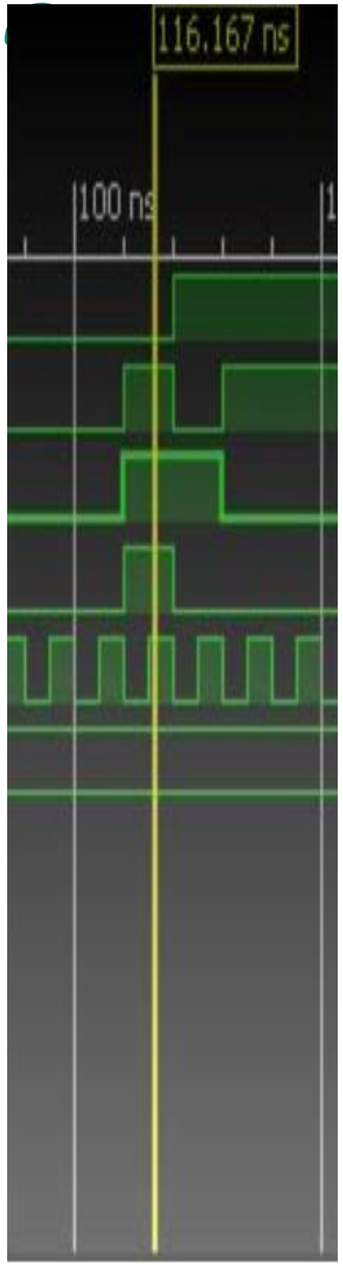
Exercise

- Get a simplified expression given the truth table

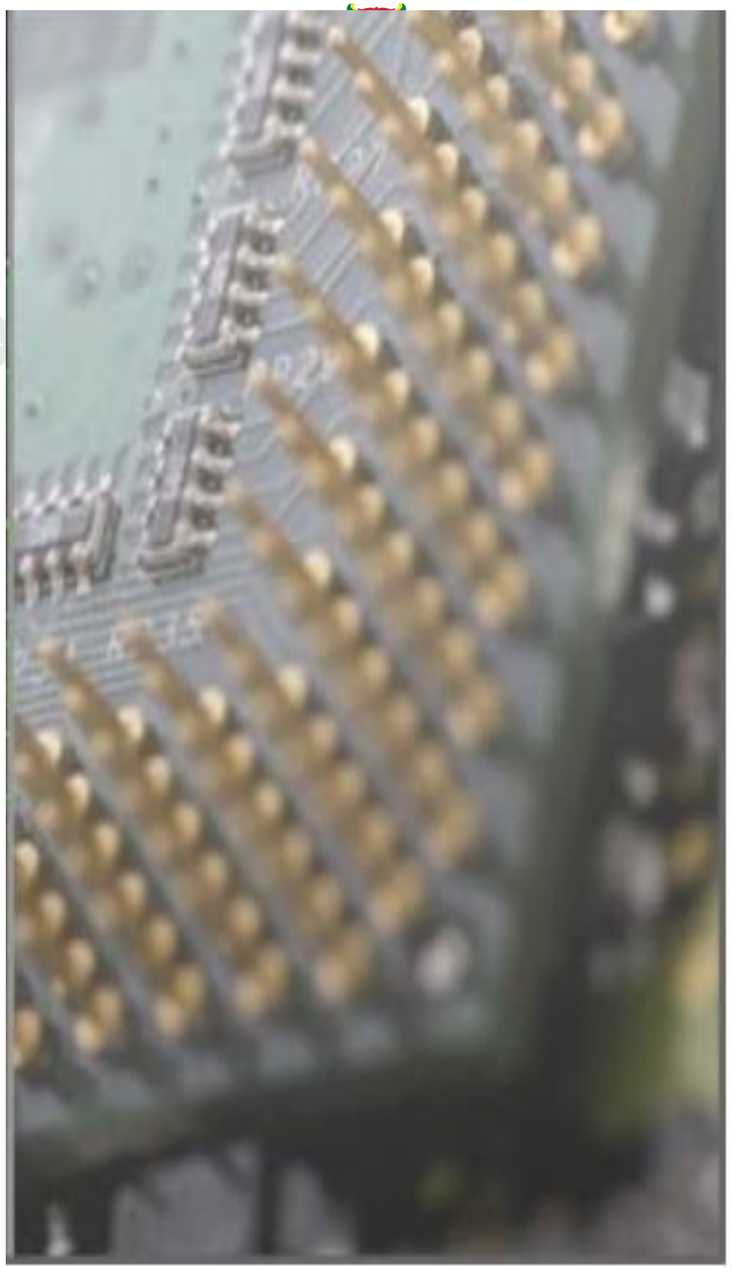
CD \ AB	00	01	11	10
00	1	0	X	1
01	0	X	X	1
11	1	1	X	X
10	1	1	X	X

$$Y = \bar{B}\bar{D} + C + A$$

A	B	C	D	Y
0	0	0	0	**1**
0	0	0	1	**0**
0	0	1	0	**1**
0	0	1	1	**1**
0	1	0	0	**0**
0	1	0	1	**X**
0	1	1	0	**1**
0	1	1	1	**1**
1	0	0	0	**1**
1	0	0	1	**1**
1	0	1	0	**X**
1	0	1	1	**X**
1	1	0	0	**X**
1	1	0	1	**X**
1	1	1	0	**X**
1	1	1	1	**X**



From logic to gates

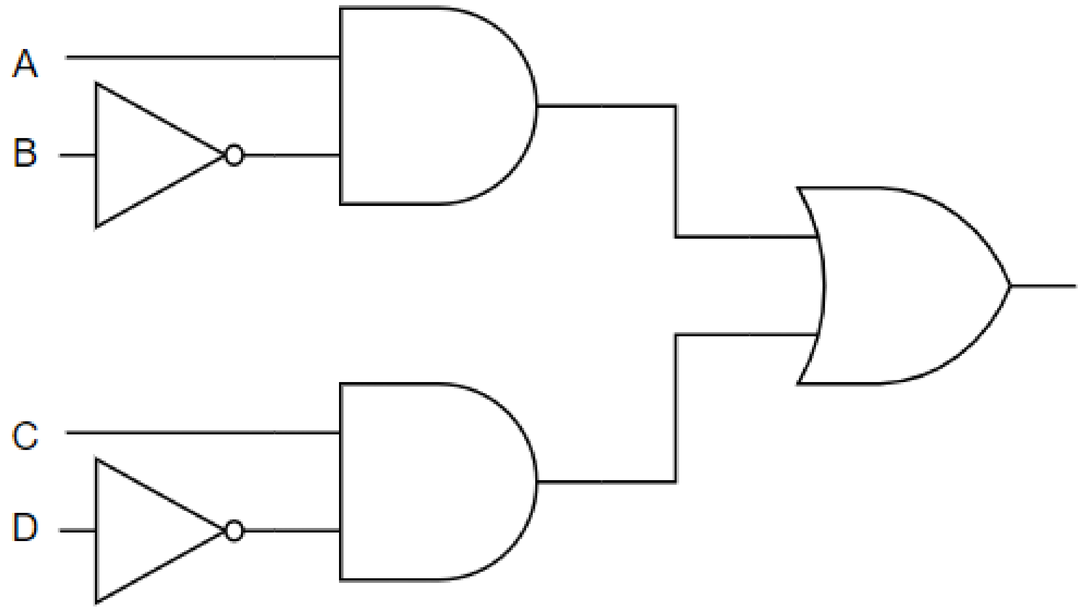


Schematics

A ***schematic*** is a diagram of a digital circuit showing the elements and wires that connect them.

Build the following equation using logic gates.

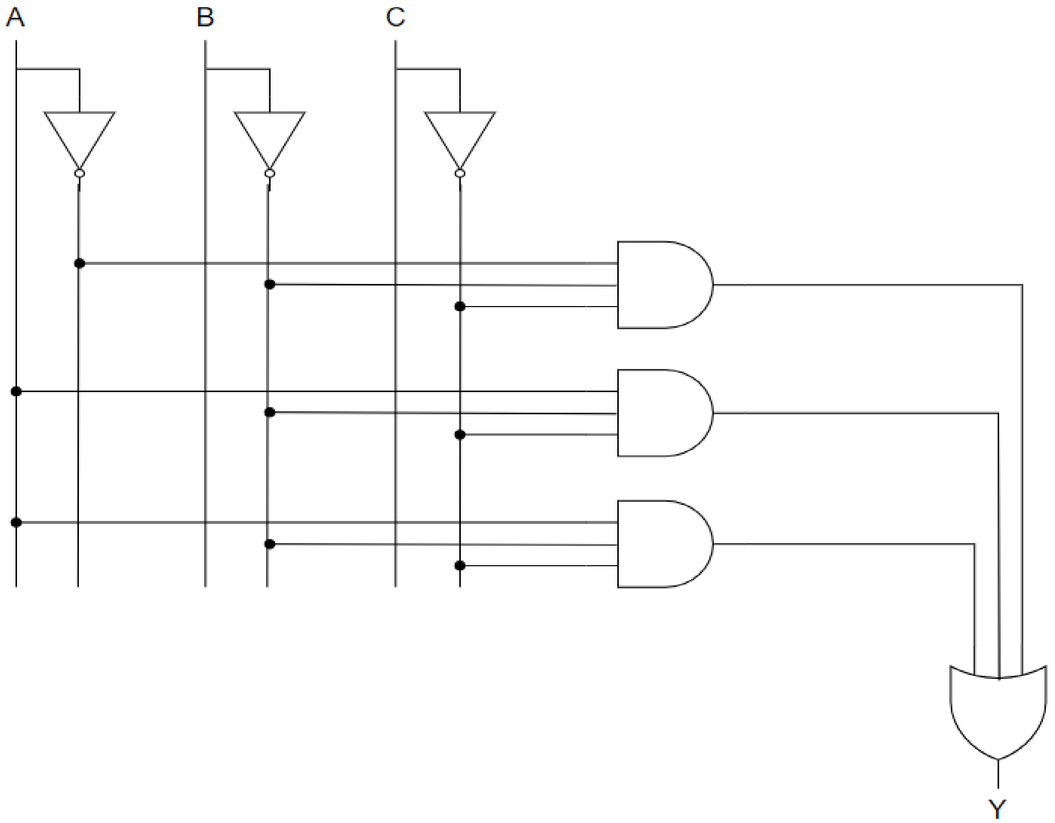
$$Y = A\bar{B} + C\bar{D}$$



Schematics

Build the following equation using logic gates.

$$Y = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C} + A\bar{B}C$$



Multiple output circuits

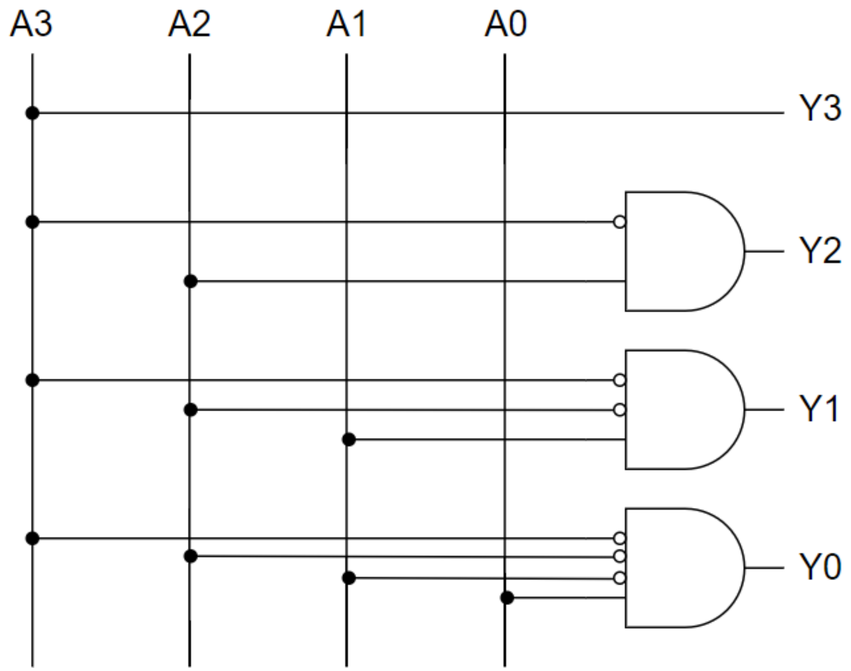
Combinational circuits often have multiple outputs. The next truth table represent a priority circuit.

$$Y_3 = A_3$$

$$Y_2 = \overline{A_3} A_2$$

$$Y_1 = \overline{A_3}\ \overline{A_2}\ A_1$$

$$Y_0 = \overline{A_3}\ \overline{A_2}\ \overline{A_1}\ A_0$$



A3	A2	A1	A0	Y3	Y2	Y1	Y0
0	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	0	0	**1**	0
0	1	0	0	0	**1**	0	0
0	1	0	1	0	**1**	0	0
0	1	1	0	0	**1**	0	0
0	1	1	1	0	**1**	0	0
1	0	0	0	**1**	0	0	0
1	0	0	1	**1**	0	0	0
1	0	1	0	**1**	0	0	0
1	0	1	1	**1**	0	0	0
1	1	0	0	**1**	0	0	0
1	1	0	1	**1**	0	0	0
1	1	1	0	**1**	0	0	0
1	1	1	1	**1**	0	0	0

Don’t care inputs.

Note that if A3 is HIGH, don’t care about other inputs, the outputs will be t Y3=1, Y2=Y1=Y0=0.

$$Y_3 = A_3$$

$$Y_2 = \overline{A_3} A_2$$

$$Y_1 = \overline{A_3}\ \overline{A_2}\ A_1$$

$$Y_0 = \overline{A_3}\ \overline{A_2}\ \overline{A_1}\ A_0$$

A3	A2	A1	A0	Y3	Y2	Y1	Y0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	**1**
0	0	1	x	0	0	**1**	0
0	1	x	x	0	**1**	0	0
1	x	x	x	**1**	0	0	0

We use the symbol ***x*** to describe inputs that the output doesn’t care about.

A3	A2	A1	A0	Y3	Y2	Y1	Y0
0	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	0	0	**1**	0
0	1	0	0	0	**1**	0	0
0	1	0	1	0	**1**	0	0
0	1	1	0	0	**1**	0	0
0	1	1	1	0	**1**	0	0
1	0	0	0	**1**	0	0	0
1	0	0	1	**1**	0	0	0
1	0	1	0	**1**	0	0	0
1	0	1	1	**1**	0	0	0
1	1	0	0	**1**	0	0	0
1	1	0	1	**1**	0	0	0
1	1	1	0	**1**	0	0	0
1	1	1	1	**1**	0	0	0