FIT 1047

Introduction to computer systems, networks and security

Space	Forward
Right, Down, Page Down	Next slide
Left, Up, Page Up	Previous slide
P	Open presenter console
H	Toggle this help

From logic to algebra

- Boolean logic
- Boolean algebra rules
- Karnaugh maps

This slide should only give some idea of what this unit is about. No details and not supposed to be complete.

Space	Forward
Right, Down, Page Down	Next slide
Left, Up, Page Up	Previous slide
P	Open presenter console
H	Toggle this help

Boolean logic is probably the simplest possible (useful) logic.
Basic concepts:

- TRUE, FALSE
- AND, OR
- NOT

TRUE and FALSE are values for statements.

Note that not all statements qualify:

- Today, the temperature is over 15 degrees Celsius.
- Today, the weather is good.
- Haggis tastes great.

Usually, TRUE is represented by 1 and FALSE is represented by 0.

Do not confuse binary and Boolean:

	Binary	Boolean
0	Zero	FALSE
1	One	TRUE

A AND B can be represented as
A ▯ B, A × B, AB

A OR B can be represented as
A ▯ B, A+B

NOT A can be represented as
$\overline{A}$, ¬A

Do not confuse binary and Boolean:

	Binary	Boolean
0+0	0	0 OR 0 = 0
1+1	10	1 OR 1 = 1

AND

Statement A AND Statement B = TRUE only when both statements are TRUE.

Examples

- Donald Duck wears a blue sailor suit AND Donald Duck does not wear trousers.

Is obviously TRUE

- At the moment we have 15 degrees AND at the moment we have below 5 degrees.

Is obviously FALSE

Using 1 and 0, we can get a very compact representation as a truth table:

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

OR

A OR B means that either A or B or both are TRUE

OR different from our usual understanding of or.

Example: On my toast I like bacon or jam.

In Boolean logic, this would mean I also enjoy having both, bacon and jam on my toast, which is actually not true.

Examples

- Today it will be warm OR it will be raining.

Means that it can be warm and dry, cold and raining, or warm and raining.

- Today it is more than 15 degrees OR below 16 degrees.

This is obviously always TRUE.

Truth table for OR

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

Now, we only need to add negation and we can construct rather complex expressions.

NOT

If something is TRUE, then, the negation NOT TRUE is obviously FALSE.

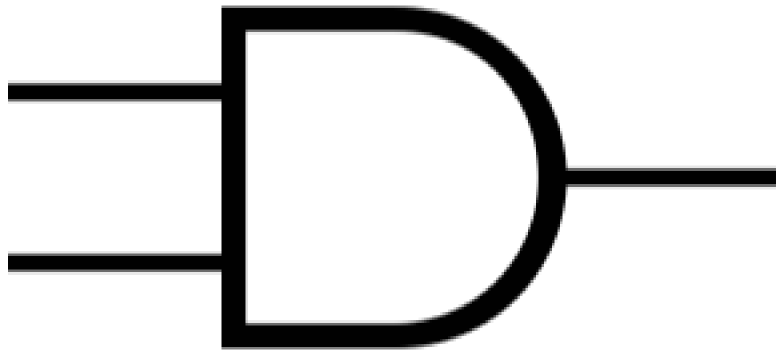
A	$\overline{A}$
0	1
1	0

In electrical circuits, AND, OR, NOT and additional operators (e.g. XOR, NAND, NOR) are realised as so-called logic gates.

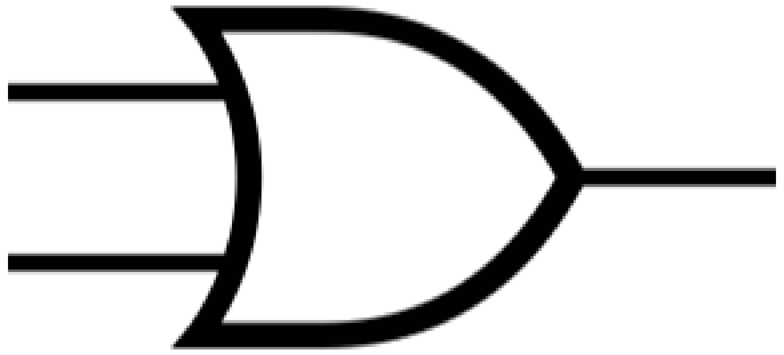
A gate performs one (or several) logical operations on some logical input (i.e. bits) and produces a single logical output.

Examples for gates

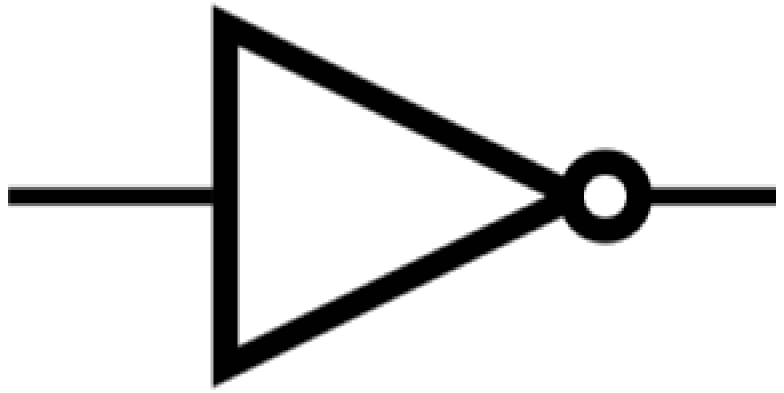
AND



OR

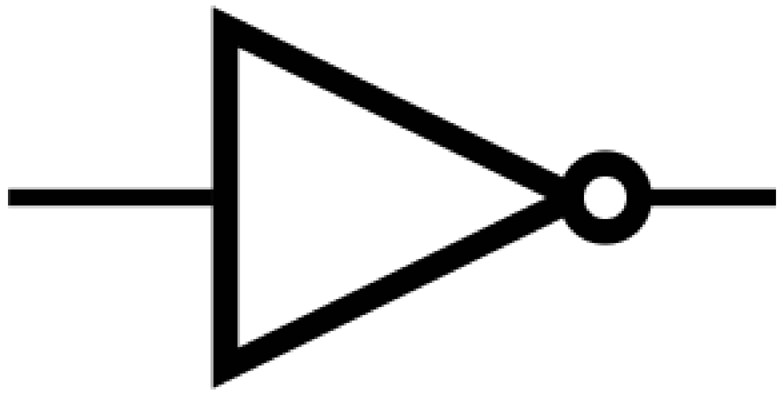


NOT

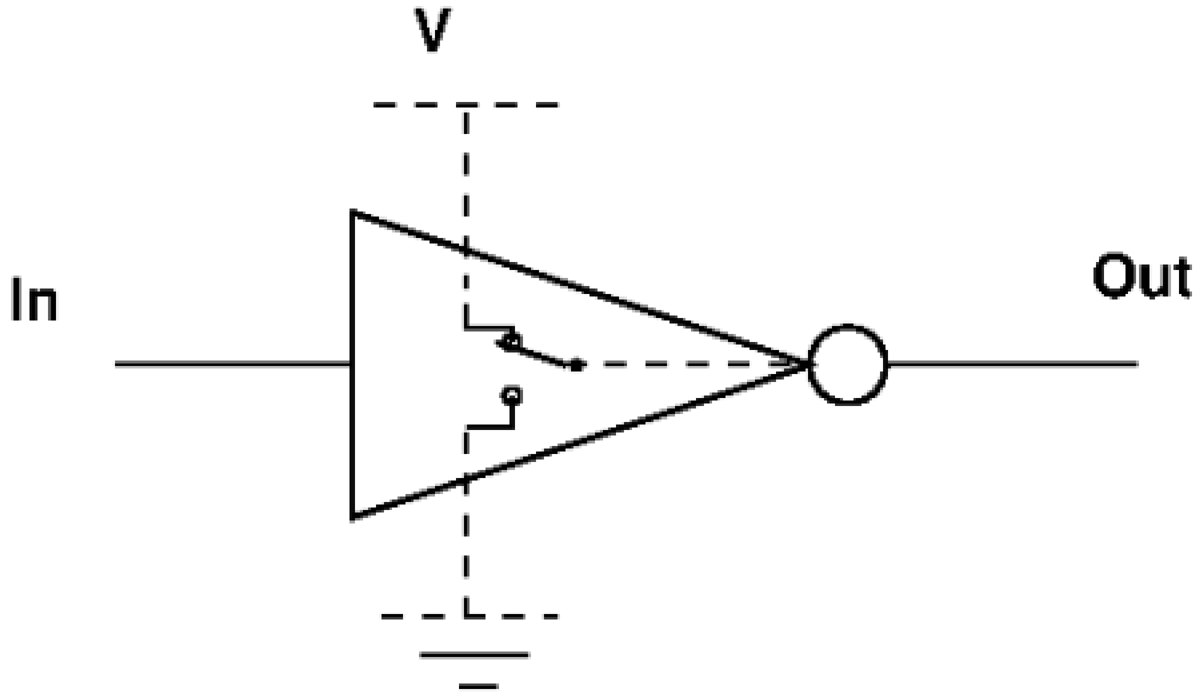


Logical circuits are not electrical circuits!

Low input can result in high output:



The reason is, that power supply is not shown in the schematic symbols.



- In the lab we will use a tool (Logisim) to simulate logical circuits.

So far, you have mainly seen Boolean logic

The term Boolean algebra implies that we might be able to do arithmetic on symbols.

Algebra

(Merriam Webster)

Laws of Boolean Algebra

- Identity Law
- Null Law (or Dominance Law)
- Idempotent Law
- Complement Law
- Commutative Law
- Associative Law
- Distributive Law
- Absorption Law
- DeMorgans Law
- Double Complement Law

Identity Law

AND Form	OR Form
$1A = A$	$0+A = A$

Null Law (Dominance Law)

AND Form	OR Form
0A = 0	1+A = 1

Idempotent Law

AND Form	OR Form
AA = A	A+A = A

Complement Law

AND Form	OR Form
$A\overline{A} = 0$	$A+\overline{A} = 1$

Commutative Law

AND Form	OR Form
AB = BA	A+B = B+A

Associative Law

AND Form	OR Form
$(AB)C = A(BC)$	$A+(B+C) = (A+B)+C$

Distributive Law

AND Form	OR Form
A+(BC) = (A+B)(A+C)	A(B+C) = AB+AC

Absorption Law

AND Form	OR Form
$A(A+B) = A$	$A+AB = A$

DeMorgans Law

AND Form	OR Form
$\overline{AB} = \overline{A}+\overline{B}$	$\overline{(A+B)} = \overline{AB}$

Double Complement Law

$$\overline{\overline{A}} = A$$

Optimization of Boolean functions

When realizing a function as circuit, one would like to minimize gates. Boolean functions can be minimized using the different laws.

However, determining the correct order of applying the laws is sometimes difficult.

In addition, one would like to minimize the use of different types of gates. Therefore, normalized forms can be useful.

One generic approach for minimizing (smaller) Boolean functions are Karnaugh maps or K-maps.

One of the most common forms of simplification in Boolean algebra is the following:

$A\overline{B} + AB = A(\overline{B} + B) = A$

One of the most common forms of simplification in Boolean algebra is the following:

$A\overline{B} + AB = A(\overline{B} + B) = A$

The same with three variables:

$A\overline{B}C + ABC = AC$

Both functions are independent from the value of B.

K-maps provide an easy graphical way to find minimal AND terms that are then combined with OR to get the complete function.

Truth table for A AND B

A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

K-map for A AND B

		B	
		0	1
A	0	0	0
	1	0	1

Let's look at a K-map with 3 variables:

$\overline{B}A\overline{C} + AB\overline{C} + ABC + \overline{A}BC + \overline{A}B\overline{C}$

		BC			
		00	01	11	10
A	0	0	0	1	1
	1	1	0	1	1

- Find groups of 1s.
- The group of four 1s represents B.

Let's look at a K-map with 3 variables:

$\overline{B}A\overline{C} + AB\overline{C} + ABC + \overline{A}BC + \overline{A}B\overline{C}$

		BC			
		00	01	11	10
A	0	0	0	1	1
	1	1	0	1	1

- Find groups of 1s.
- The group of four 1s represents B=1.
- The wrapped group represents A=1 AND C=0.

Let's look at a K-map with 3 variables:

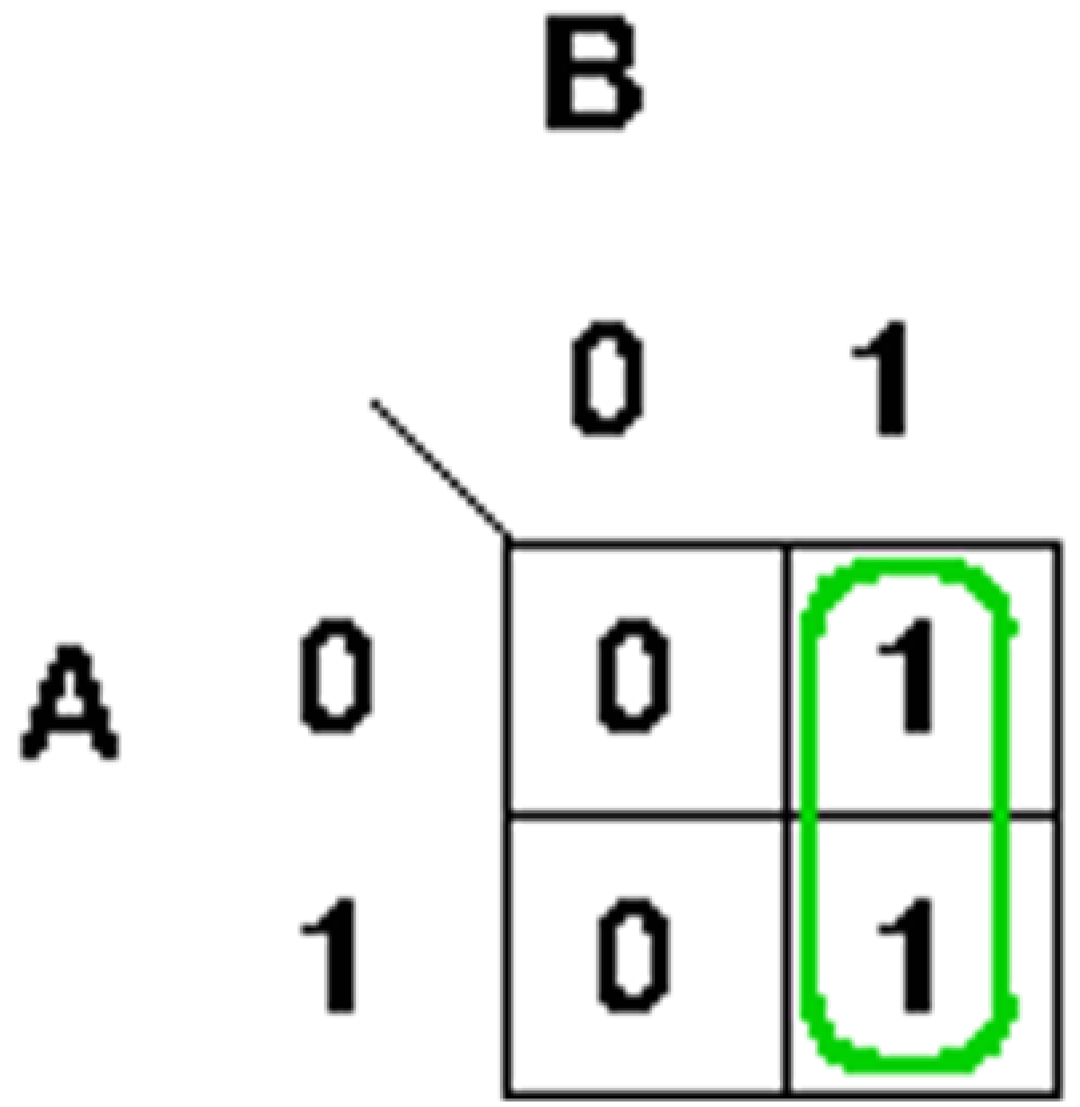
$\overline{B}A\overline{C} + AB\overline{C} + ABC + \overline{A}BC + \overline{A}B\overline{C}$

		BC			
		00	01	11	10
A	0	0	0	1	1
	1	1	0	1	1

- Find groups of 1s.
- The group of four 1s represents B=1.
- The wrapped group represents A=1 AND C=0.

Simplified version: $B + A\overline{C}$

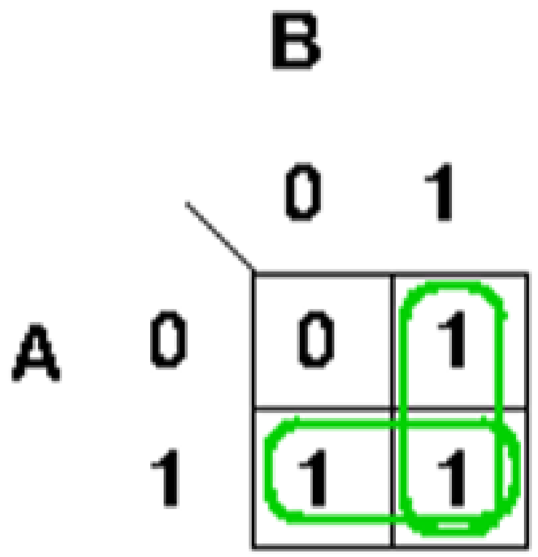
7 rules for working with K-maps



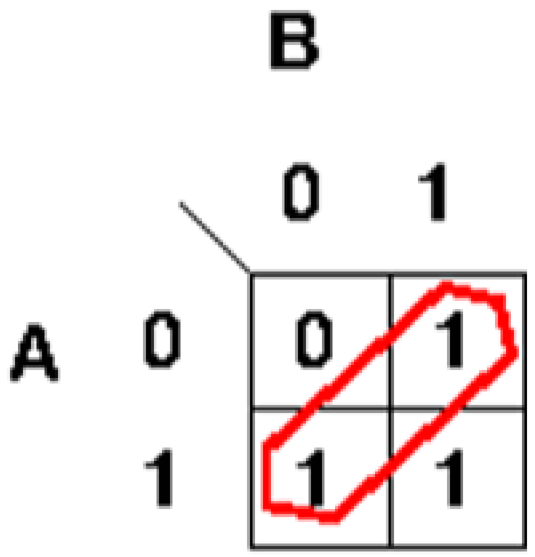
		B	
		0	1
A	0	0	1
	1	0	1

Rule 1: No group can contain a zero.

Rule 2: Groups may be horizontal/vertical/square, but never diagonal.

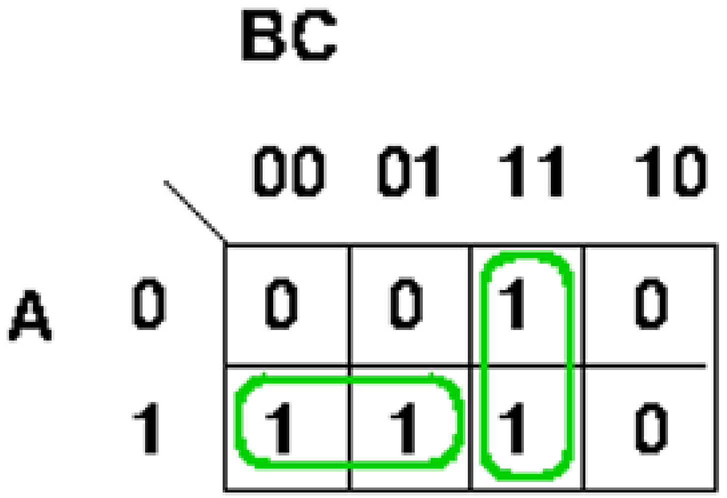


Correct



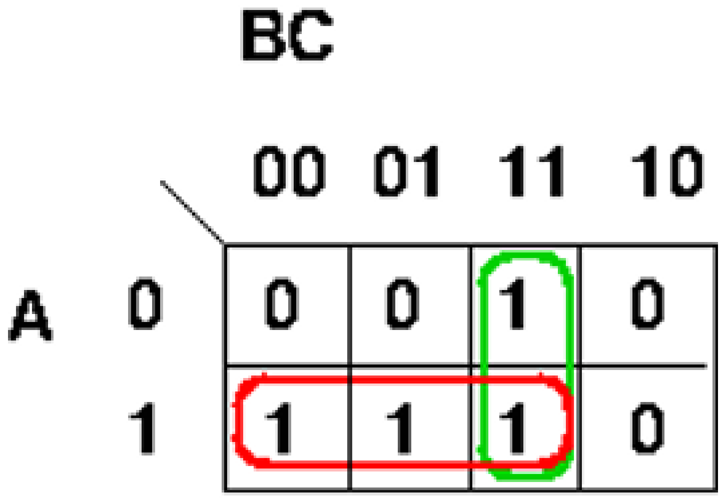
Wrong

Rule 3: Groups must contain 1,2,4,8,16,32,... (powers of 2) cells.



		BC			
		00	01	11	10
A	0	0	0	1	0
	1	1	1	1	0

Correct



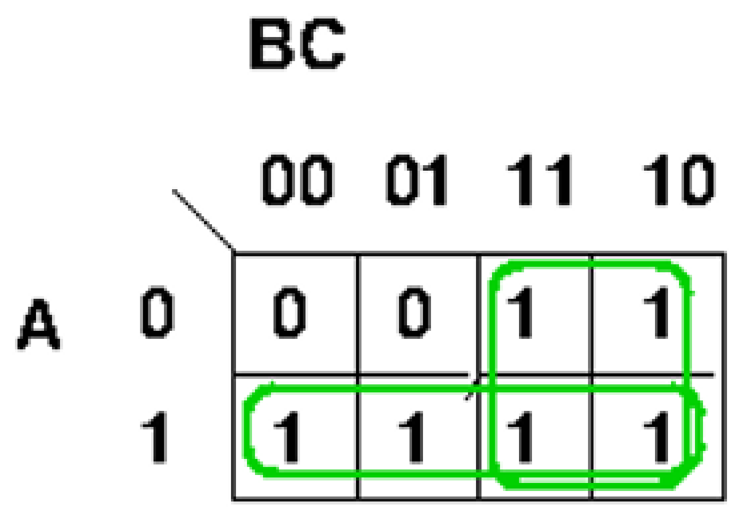
		BC			
		00	01	11	10
A	0	0	0	1	0
	1	1	1	1	0

Wrong

Rule 4: Each group must be as large as possible.

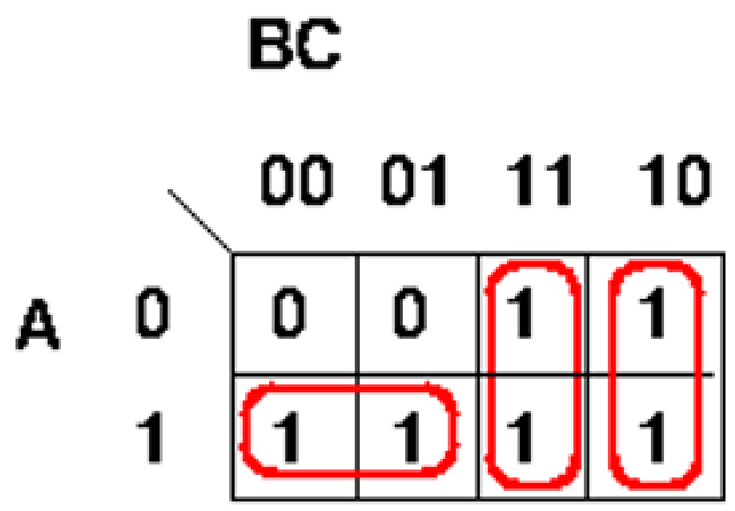
Rule 5: Groups can overlap.

Rule 6: Each 1 must be part of at least one group.



A \ BC	00	01	11	10
0	0	0	1	1
1	1	1	1	1

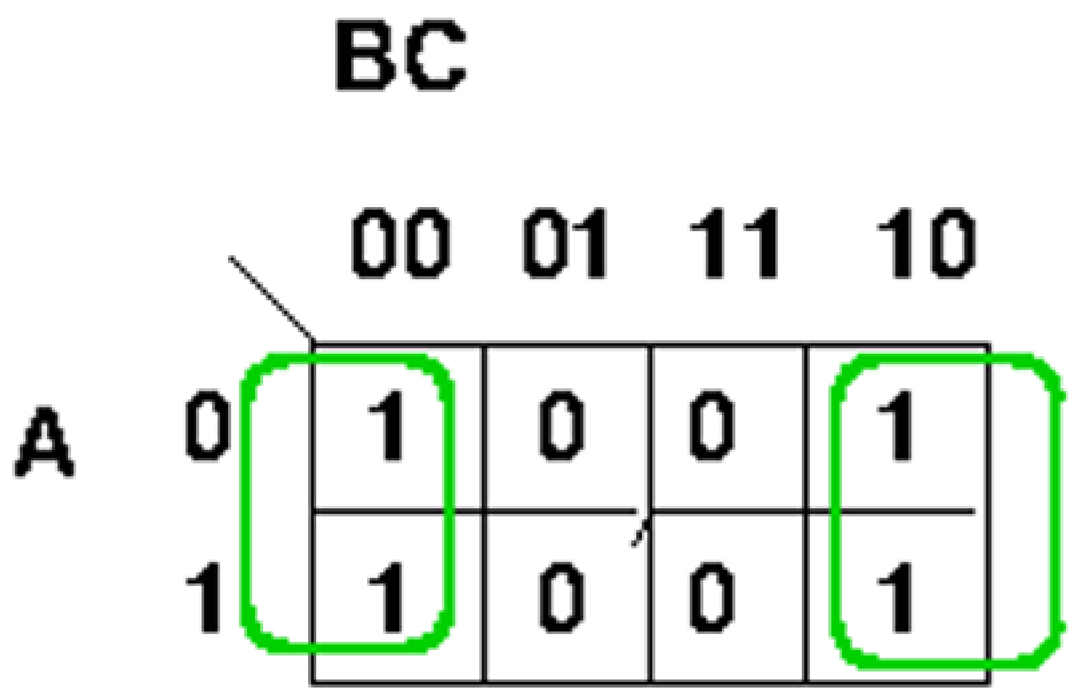
Correct



A \ BC	00	01	11	10
0	0	0	1	1
1	1	1	1	1

Wrong

Rule 7: Groups may wrap around the map.



	BC			
A	00	01	11	10
0	1	0	0	1
1	1	0	0	1

Wrap around

Rule 1: No group can contain a zero.

Rule 2: Groups may be horizontal/vertical/square, but never diagonal.

Rule 3: Groups must contain 1,2,4,8,16,32,... (powers of 2).

Rule 4: Each group must be as large as possible.

Rule 5: Groups can overlap.

Rule 6: Each 1 must be part of at least one group.

Rule 7: Groups may wrap around the map.

K-maps are useful for smaller Boolean functions.

Automated algorithms for optimization are used for bigger functions.

Universal gates

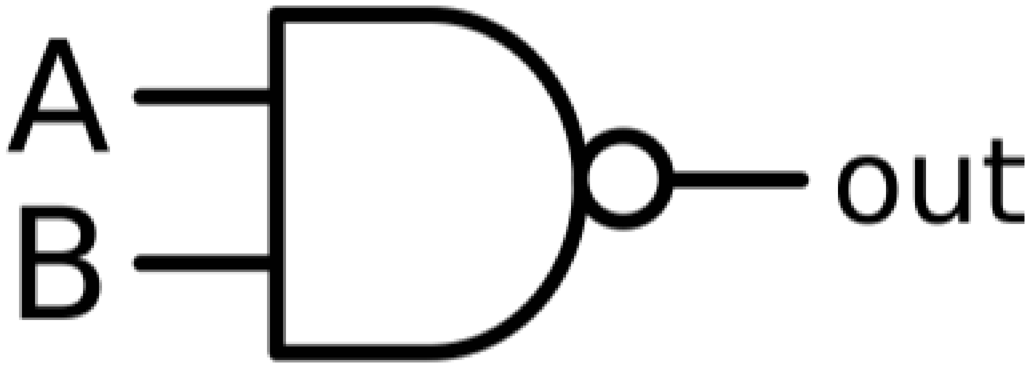
NAND and NOR gates have special properties:

1. NAND can be realised very efficiently.
2. All other gates can be build only using NAND gates.

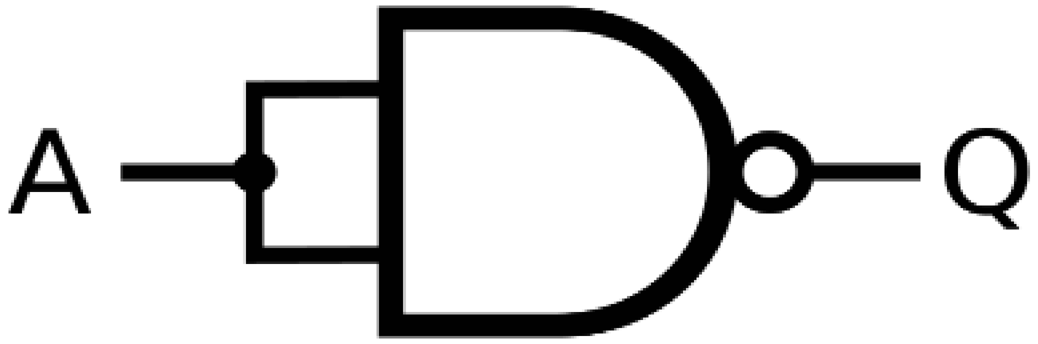
NAND is $\overline{AB}$

Truth table and symbol for NAND

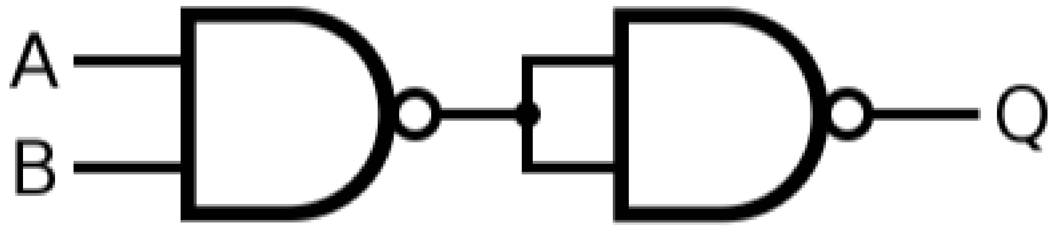
A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0



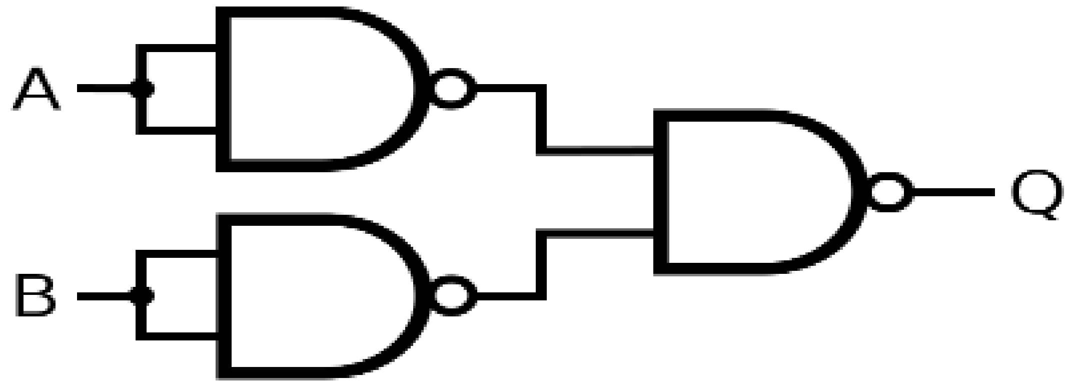
Building NOT with NAND gates



Building AND with NAND gates



Building OR with NAND gates



Summarize topics of first 2 weeks

- A little bit of history
- Vacuum tubes and transistors
- bit, byte, word (8bit/16bit/32bit/64bit)
- Numbering systems (base 2, base 10, base 16)
- Conversion between numbering systems

- Signed integer representations (sign/magnitude, 1's complement, 2's complement)
- Properties of 2's complement, adding 2's complement numbers
- Floating point representation
- Properties of floating point, precision, rounding
- Characters: ASCII and Unicode

- Error detection: Parity bits, Checksum, CRC
- Boolean Logic AND, OR, NOT, XOR, NAND, NOR
- Logic gates
- Simple logic circuits
- Boolean Algebra, Laws
- Karnaugh Maps/K-maps