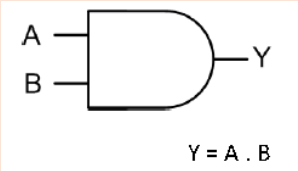
Logic Gates and truth tables

These are devices that implement a Boolean function, that is they perform logical operations on one or more logical inputs to produce a single logical output. Every terminal has one of the two binary conditions: low (0) and high (1) represented by different voltage levels.

AND Gates:

When at all inputs are high (1) the output will be high (1).

Input X	Input Y	Output
1	1	1
1	0	0
0	1	0
0	0	0

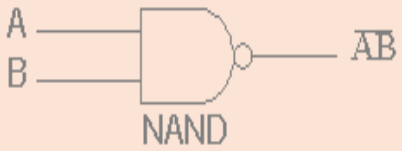


A dot (.) is used to show the AND operation i.e. A.B - Bear in mind that this dot is sometimes omitted i.e. AB

NAND Gates:

"NOT AND", hence when at least one input is high (1) the output is high(1). If both inputs are high (1) the the output is low (0).

Input X	Input Y	Output
1	1	0
1	0	1
0	1	1
0	0	1



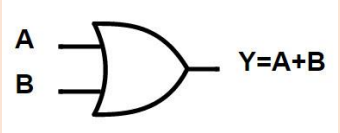
It is represented as A.B (or $\overline{AB}$) with a bar over the top. In the exam we put ¬ with the object of interest in brackets AFTER the ¬ instead of the bar. NOT is applied after AND.

This is a NOT-AND gate which is equal to an AND gate followed by a NOT gate. Or two NOT gates followed by an OR gate.

OR Gates:

When one or more of the inputs is high (1) the output will be high (1).

Input X	Input Y	Output
1	1	1
1	0	1
0	1	1
0	0	0



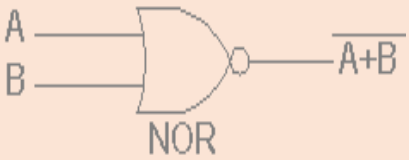
It is represented as A + B.

Be careful + means OR.

NOR Gates:

When any one of the inputs is high (1), the output will be low (0). If both inputs are low (0), the output is high (1).

Input X	Input Y	Output
1	1	0
1	0	0
0	1	0
0	0	1



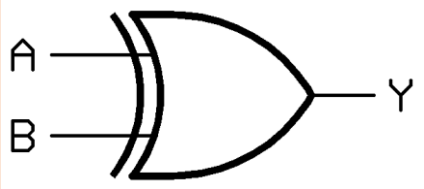
It is represented as NOT(A or B), hence ¬(A + B), or $\overline{A + B}$

Same as an OR gate with a NOT gate

XOR Gates:

`Exclusive Or gates'. These will only ever give an output that is high (1) when either, not both of the inputs is h gh (1).

Input X	Input Y	Output
1	1	0
1	0	1
0	1	1
0	0	0

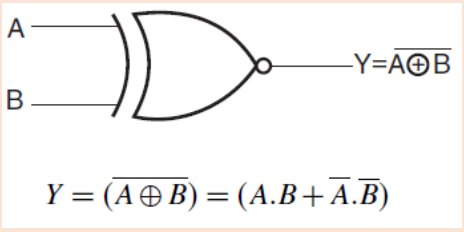


It is represented as A ⊕ B. Where the encircled plus `⊕' is sued to show the XOR operation.

XNOR:

`Exclusive NOT OR', does the opposite to an XOR gate. It will give a low (0) output if either, but not both, of th inputs is high (1). Only when the inputs are the same state (both 1 or both 0) will the output be high (1). If only one input is high then the output will be low.

Input X	Input Y	Output
1	1	1
1	0	0
0	1	0
0	0	1



It is represented as ¬(A ⊕ B). Where the XOR function is applied before the NOT operation.

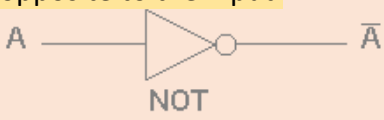
Sometimes = A.B + (¬A.¬B)

Same as an AND gate paralleled with an AND gate that has both inputs inverted by 2 NOT gates. This is then fed into an OR gate.

NOT Gates:

Sometimes called an inverter. The output is the opposite to the input.

Input X	Output
1	0
0	1



It is represented as ¬ followed by item(s) of interest in brackets. Or by a bar drawn over items being inverted.

A NOT gate can be created with NAND gate where the inputs are linked so identical. Therefore when the single input is low (0), creates two identical conditions - 2 low inputs (0). The output is high Since at least one low input is required for a high output (1).

When the single input is high (1), two identical high inputs are created (1). The output is low since at least one input needs to b low (0) for a high (1) output.

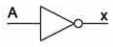
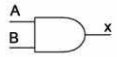
Boolean algebra = The branch of algebra where the values of the variables are the truth values of true (1) and false (0). The main operations are addition and multiplication and the multiplicative inverse function.

+ or V means add (OR)

. or ^ Means multiply (AND)

¬ Means invert (raise by the power of -1) (multiplicative inverse function.)

Logic Gates

Name	NOT	AND	NAND	OR	NOR	XOR	XNOR
Alg. Expr.	$\overline{A}$	AB	$\overline{AB}$	$A+B$	$\overline{A+B}$	$A \oplus B$	$\overline{A \oplus B}$
Symbol							
Truth Table	A X 0 1 1 0	B A X 0 0 0 0 1 0 1 0 0 1 1 1	B A X 0 0 1 0 1 1 1 0 1 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 1	B A X 0 0 1 0 1 0 1 0 0 1 1 0	B A X 0 0 0 0 1 1 1 0 1 1 1 0	B A X 0 0 1 0 1 0 1 0 0 1 1 1

The exam board only ever uses:

- **^ = AND**
- **V = OR**
- **¬ = NOT**
- **() = brackets**
- **≡ means can be written as (identity)**

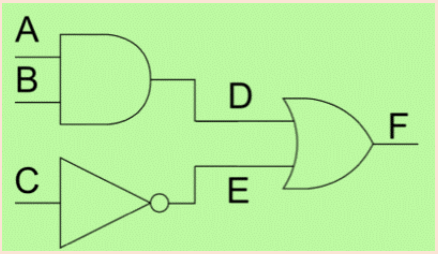
Logic gate diagrams

Logic gates may be combined to form logic gate diagrams that perform more complicated logical operations. Truth tables are used to show the states of each terminal and hence the logical operations.

e.g.

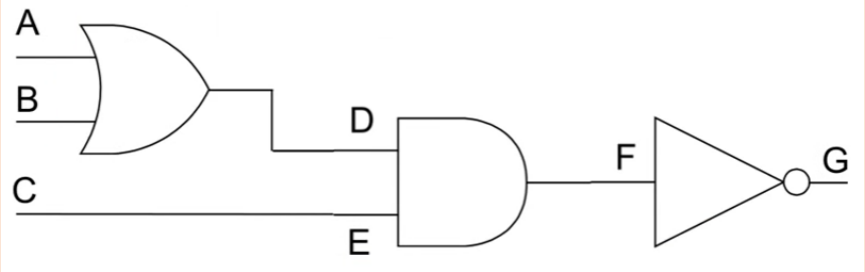
Create a logic gate for the following. Then create the truth table

$$R=(A \wedge B) \vee \neg C$$



Inputs			Intermediate outputs		Output
A	B	C	D	E	R
1	1	1	1	0	1
1	1	0	1	1	1
0	0	0	0	1	1
0	0	1	0	0	0
0	1	1	0	0	0
1	0	1	0	0	0
1	0	0	0	1	1
0	1	0	0	1	1

e.g. 2) What is the algebraic expression, where variables are denoted with Boolean logic for the following logic gate diagram? Give the truth table for this.



Expression = ¬ [(A V B) ^ C]

Inputs			Intermediate outputs			Output
A	B	C	D	E	F	G
1	1	1	1	0	0	1
1	1	0	1	1	1	0
0	0	0	0	1	0	1
0	0	1	0	0	0	1
0	1	1	1	0	0	1
1	0	1	1	0	0	1
1	0	0	1	1	1	0
0	1	0	1	1	1	0

Rules for simplifying Boolean Algebra

De Morgan's Law:

Rule 1) Either logical function AND or OR may be replaced by the other, given certain changes to the equatio .

- NOT (A OR B) is the same as (NOT A) AND (NOT B)
  i.e. ¬(A V B) ≡ (¬A) ^ (¬B)
- Likewise, NOT (A AND B) is the same as (NOT A) OR (NOT B)
  i.e. ¬(A^B) ≡ (¬A) V (¬B)

An analogy in English is: It cannot be winter AND summer at any point in time
which is the same as: At any point in time, It is NOT winter OR it is NOT summer.

The Law of distribution:

Rule 2) This law allows for the multiplying or factoring out the common terms of an expression.

- The **OR distributive** law: A AND (B OR C) is the same as (A AND B) OR (A AND C)
  i.e. A^(B V C) ≡ (A^B) V (A ^ C)
- The **AND distributive** law: A OR (B AND C) is the same as (A OR B) AND (A OR C)
  i.e. A V (B^C) ≡ (A V B) ^ (A V C)

An analogy in English is:
You can choose 1 main course AND either a starter OR dessert. This is the same as you can choose 1 main AND 1 starter R 1 main AND 1 desert

You can choose a cake OR a biscuit AND a milkshake.
This is the same as you can choose a cake or a biscuit AND a cake or a milkshake.

The Law of association:

Rule 3) This law allows for the removal of brackets from an expression and regrouping of the variables.

- The **OR association** law: A OR (B OR C) is the same as A OR B OR C
  i.e. AV (B V C) ≡ A V B V C

- The **AND association** law: A AND (B AND C) is the same as A AND B AND C
  i.e. A ^ (B^C) ≡ A ^ B ^ C

The Law of commutation:

Rule 4) The order of application of two separate terms is not important so does not affect end result.

- A OR B is the same as B OR A i.e. A V B ≡ B V A
- A AND B is the same as B AND A i.e. A ^ B ≡ B ^ A

The rule of double negation:

Rule 5) If a variable is reversed twice then it remains the same.

- NOT NOT A is the same as A i.e. ¬(¬A) ≡ A

The rule of absorption:

Rule 6) The second term inside a bracket can always be eliminated (absorbed) by the term outside the brack t if given results are met.

- A OR (A AND B) is the same as A i.e. A V (A ^ B) ≡ A
- A AND(A OR B) is the same as A i.e. A ^ (A V B) ≡ A

- The operators inside and outside the brackets must be different.

- The term outside the brackets must also be included inside the brackets.

In addition to these 6 rules. There are also 8 general rules (for AND and OR gates) that can be applied very quickly.

Remember that with **AND** both terms have to be **1** or **TRUE** for the result to be **TRUE**

1	$X \wedge 0 = 0$	X **AND** 0 is the same as 0 *Or to put it another way... X **AND FALSE** has to equal **FALSE** (See truth table on reverse side for proof)*
2	$X \wedge 1 = X$	X **AND** 1 is the same as X *Or to put it another way... X **AND TRUE** has to equal **TRUE** (See truth table on reverse side for proof)*
3	$X \wedge X = X$	X **AND** X is the same as X *Or to put it another way... X **AND** X has to equal X (See truth table on reverse side for proof)*
4	$X \wedge \neg X = 0$	X **AND not** X is the same as 0 *Or to put it another way... X **AND NOT(X)** has to equal **FALSE** (See truth table on reverse side for proof)*

Remember that with **OR** only **1** term has to be **1** or **TRUE** for the result to be **TRUE**

5	$X \vee 0 = X$	X **OR** 0 is the same as X *Or to put it another way... X **OR FALSE** has to equal **TRUE** (See truth table on reverse side for proof)*
6	$X \vee 1 = X$	X **OR** 1 is the same as X *Or to put it another way... X **OR TRUE** has to equal **TRUE** (See truth table on reverse side for proof)*
7	$X \vee X = X$	X **OR** X is the same as X *Or to put it another way... X **OR** X has to equal X (See truth table on reverse side for proof)*
8	$X \vee \neg X = 1$	X **OR not** X is the same as 1 *Or to put it another way... X **OR NOT(X)** has to equal **TRUE** (See truth table on reverse side for proof)*

Simplifying Boolean algebra

Two important rules

1. **Order of precedence (this is a rule)**
   If we have: (A^B) V ¬C
   It is written in word form as: (A AND B) OR NOT C
   The order of precedence is **not** followed by **and** followed by **or**.
2. **Rules of Boolean algebra**
   A ^ ¬A = 0 (Because when A =1 Output = 0, A = 0 output = 0)
   A V ¬A = 1 (Because one of the terms will always be a 1)

When a Boolean expression is not in the simplest form it can make it difficult to understand and the logical statement may require many logic gate components so it is not an efficient circuit.

Simplifying Boolean algebra

i) A ^ B V A ^ (BVC) V B ^ (B V C)

A AND B OR A AND (B OR C) OR B AND (B OR C) (Writing in letters and adding brackets)

A AND B OR (A AND B) OR (A AND C) OR B (Rule of absorption)

A AND B OR A AND C OR B (Removing repeated term)

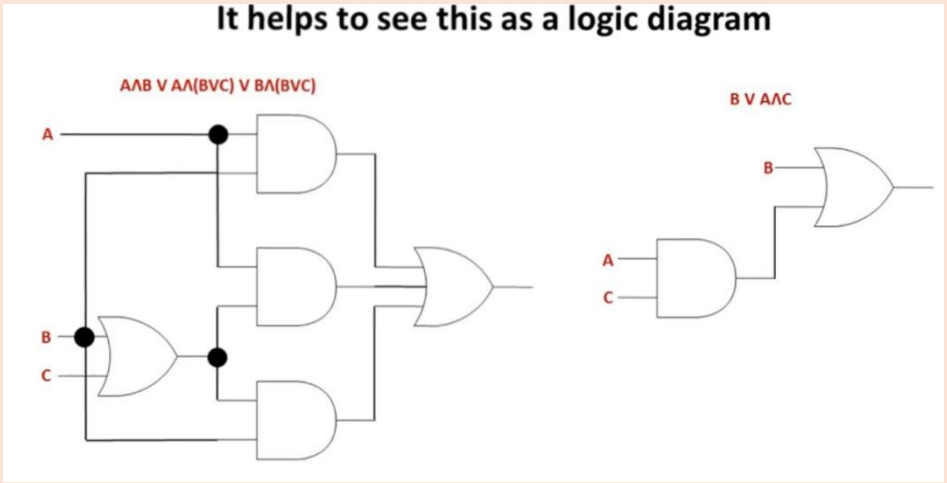
B OR (A AND B) OR A AND C (Reordering and adding brackets)

B OR A AND C (Rule of absorption)

= B V A^C **(This is the simplest form of the same original expression)**

We can see how this would save on logic gate components making a circuit more efficient.

It helps to see this as a logic diagram



This saves money on components, makes circuits smaller, reduces energy consumption, reduces stock levels for manufacturers.

Simplifying Boolean algebra

1. Simplify the expression (A^¬A)VB

= (A AND NOT A) OR B → 0 OR B (we can never get true from A) → **B**

2. Simplify the expression (A V B) V (A^C)

= (A OR B) OR (A AND C) → B OR A OR (A AND C) → B OR A → **A V B**

3. Simplify the expression ¬(A^¬B) V (¬A^B)

= NOT(A AND NOT B) OR (NOT A AND B) →NOT A AND B OR NOT A AND B → NOT A AND B → **¬A ^ B**

4. Simplify the expression (A^B) V (A^(B^C)) V (B^(BVC))

= A AND B OR A AND (B OR C) OR B AND B OR C → (A^B) V (A^(B^C)) V B → B V (A^B) V (A^(B^C)) →
B V (A^(B^C)) → B V (A ^ B ^ C)→ **B**

Simplify the expression ¬R = ¬(¬A^(BVC))

= ¬¬A V ¬(BVC) The and became or as it was inversed by the ¬, however, the sign within brackets has not changed yet only the brackets have been inversed "¬(BVC)" **Technically part of De Morgan's Law**

= A V ¬B ^ ¬C Once again the NOT will inverse the OR sign to become an AND when expanding **De Morgan's Law**

= This can't be simplified further: **A V ¬B ^ ¬C**

Class examples

2. (AVB) V (A^C)
   = A V B V (A^C) → B V A V (A^C) → **A V B**

1. (A^B) V A^(BVC)
   = (A^B) V (A^B) V (A^C) = **(A^B) V (A^C)**

3. ¬A V ¬B V ¬(AVB)
   = ¬A V ¬B V ¬A V ¬B = **¬A V ¬B**