**Introduction to Boolean Algebra and Logic Gates**

Boolean algebra is named after the mathematician, called George Boole, who first developed it. It is a form of algebra that deals with logic.

Consider a single light switch, it can be either ON or OFF - TRUE or FALSE - 1 or 0

|  |  |
| --- | --- |
| http://staffweb.cms.gre.ac.uk/~sp02/logic/images/andswitch.gif | http://staffweb.cms.gre.ac.uk/~sp02/logic/images/orswitch.gif |
| Switches in Series (AND) | Switches in Parallel (OR) |
| **A.B** means **A AND B** | **C + D** means **C OR D** |

**An example of OR**

University regulations state that before a student can be accepted for a Creative Arts degree they require:-

Art "A" Level   
OR  
12 cornflake packet tops

Fill in the table to decide which groups of students are accepted on the course.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Condition A "A" level | Condition B Packet tops | Accept for course |
| Student 1 | No | No |  |
| Student 2 | No | Yes |  |
| Student 3 | Yes | No |  |
| Student 4 | Yes | Yes |  |

**An example of AND**

To successfully rob a bank you require BOTH a get-away car AND dynamite. Fill in the table to decide which crooks get away with the loot.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Condition A Get Away Car | Condition B Dynamite | Successful Robbery |
| Crook 1 | No | No |  |
| Crook 2 | No | Yes |  |
| Crook 3 | Yes | No |  |
| Crook 4 | Yes | Yes |  |

**Truth Tables**

Most Boolean algebra expressions use "1" for YES and "0" for NO and a table like these above can be used to show all valid possibilities. This is called a TRUTH table.

**OR**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **A + B** |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

**AND**

|  |  |  |
| --- | --- | --- |
| **A** | **B** | **A.B** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**NOT**

|  |  |
| --- | --- |
| **A** | **A** |
| 0 | 1 |
| 1 | 0 |

 A + B means (NOT A) or (NOT B)

A + B means NOT (A OR B)

**Operator Precedence**

The order is as follows:-

1. Brackets
2. NOT
3. AND, NAND
4. OR, XOR, NOR

Any expression on the same line has the same precedence, e.g. AND has the same precedence as NAND.

If the operators have equal precedence then precedence travels from right to left. ALWAYS put in the brackets to show precedence, when writing Boolean expressions.

**Rules of Boolean Algebra**

1. Commutative laws
   * A + B = B + A
   * A.B = B.A
2. Associative laws
   * A + (B + C)  =  (A + B) + C
   * A.(B.C)  =  (A.B).C
3. Distributive laws
   * A.(B + C)  =  (A.B) + (A.C)
   * A + (B.C)  =  (A + B).(A + C)
4. Tautology laws
   * A.A = A
   * A + A = A
   * A + A = 1
   * A . A =  0
5. Absorption laws
   * A + (A.B) = A
   * A.(A + B) = A
6. Common Sense laws
   * 0.A = 0
   * 1 + A = 1
   * 1.A = A
   * 0 + A = A

* + 0 = 1

* + 1 = 0

1. Double Complement law

          \_

* + A = A

1. De Morgan's law

* + A + B = A .B
  + A + B = A . B

All these laws can all be proved using truth tables.

Consider Rule De Morgan's below

A + B = A .B

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | B | A | B | A + B | A.B | A.B |
| 0 | 0 | 1 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 |

* Prove

A + B = A . B

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| A | B | A | B | A . B | A + B | A + B |
| 0 | 0 |  |  |  |  |  |
| 0 | 1 |  |  |  |  |  |
| 1 | 0 |  |  |  |  |  |
| 1 | 1 |  |  |  |  |  |

**Example**

Show that A + A . B = A using Boolean algebra and a truth table

From the Common Sense laws A .1 = A the expression A + A . B = A can be written as

(A . 1) + (A . B) = A

A.(1 + B) = A

A.(1) = A

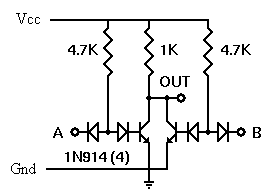
A = A

Verify with a truth table

|  |  |  |  |
| --- | --- | --- | --- |
| **A** | **B** | **A . B** | **A + A .B** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 |

**Electronic Logic**

A logic gate is an electronic circuit constructed of transistors designed to perform a specific function, such as AND, OR and NOT.



Two Input AND Gate

Each logic gate has a special symbol.

AND Gate        Z = A . B

AND Gate

OR Gate             Z = A + B

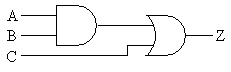
OR Gate

NOT Gate          Z =  A

NOT Gate

Any logic circuit, no matter how complex, can be described using the Boolean expressions of AND, OR and NOT.

**Example** the circuit given below has three inputs A, B and C and a single output F. Utilising the Boolean expression for each gate it is easy to arrive at an expression for the output.



Z = A.B + C

**Circuits containing Inverters**

Whenever an inverter is present in a logic-circuit diagram, its output expression is simply equal to the input expression with the bar over it. i.e.

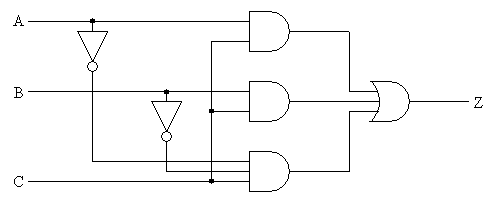
http://staffweb.cms.gre.ac.uk/~sp02/logic/images/logic2a.gif

Z = A.B + C

http://staffweb.cms.gre.ac.uk/~sp02/logic/images/logic2b.gif

Z = A.B

**Example** Construct A.C + B.C  + A.B.C



**Binary Half Adder**

Consider a simple circuit to add together two binary digits, A0 and B0 and produce the correct answer, which is called sum. The circuit will also produce a carry (if necessary), so it is not possible to use a simple OR gate.



Truth table for a binary adder

|  |  |  |  |
| --- | --- | --- | --- |
| **A0** | **B0** | **S0 (Sum)** | **C0 (Carry)** |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |

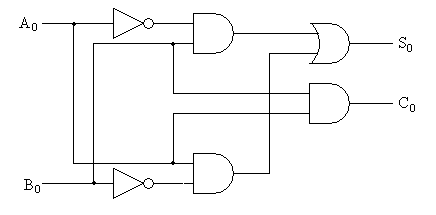
From the truth table it can be seen that:-  S0 is 1 if (A0 is 0 AND B0 is 1) OR (A0 is 1 AND B0 is 0) i.e.

S0 = A0 . B0  +  A0 . B0

Next consider the carry (C0). By looking at the truth table we can see that:- C0 is only 1 when both A0 AND B0 are 1 i.e.

C0 = A . B

The final circuit is shown below:-



This is not the most efficient way to add binary digits, but it does show how logic gates such as AND, OR and NOT can be usefully combined to perform functions inside computers.

**Karnaugh Maps**

Karnaugh maps are a useful aid to simplifying certain types of Boolean expressions and are often quicker and easier than applying the rules of Boolean algebra, with greater chance of getting the right answer. A Karnaugh map is simply a convenient tabular arrangement of a Venn diagram.

Consider the arrangement of 4 cells shown below:-

|  |  |  |
| --- | --- | --- |
| B\A | 0 | 1 |
| 0 | A . B | A . B |
| 1 | A . B | A . B |

Consider 4 variables. Note the ordering of the logic values.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CD\AB | 00 | 01 | 11 | 10 |
| 00 | \_   \_   \_   \_  A . B. C . D | \_        \_   \_  A . B. C . D | \_   \_ A . B. C . D | \_   \_   \_  A . B. C . D |
| 01 | \_   \_   \_       A . B. C . D | \_        \_         A . B. C . D | \_       A . B. C . D | \_   \_         A . B. C . D |
| 11 | \_   \_             A . B. C . D | \_                   A . B. C . D | A . B. C . D | \_              A . B. C . D |
| 10 | \_   \_         \_ A . B. C . D | \_              \_ A . B. C . D | \_ A . B. C . D | \_         \_ A . B. C . D |

**Example** Map the following expression to a Karnaugh map

A.B.C.D  +  A.B.C.D  +  A.B.C.D  +  A.B.C.D

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CD\AB | 00 | 01 | 11 | 10 |
| 00 |  |  |  |  |
| 01 |  |  |  |  |
| 11 |  |  |  |  |
| 10 |  |  |  |  |

Having mapped the given expression it can now be simplified. Draw an outline around it the part of the graph with the 1's in it. This is the only part to be considered. Only horizontal or vertical shapes can be linked Look carefully at the part of the map which contains the 1's. If a variable covers both a 0 and a 1 state then it can ignored. For example - consider C in the above map - it covers both 0 and 1 states and so the final expression will not contain the variable C. Similarly, D covers both 0 and 1 states, so the final expression will be?

**Example** Simply the following expression

A.B.C.D  +   A.B.C.D  +  A.B.C.D  +  A.B.C.D

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CD\AB | 00 | 01 | 11 | 10 |
| 00 |  |  |  |  |
| 01 |  |  |  |  |
| 11 |  |  |  |  |
| 10 |  |  |  |  |

**Rules of Symmetry for Karnaugh Maps**

In addition to the binary powers, the map can be folded in such a way as to form a cylinder, or so that all four corners touch one another. This can aid simplification, and some of the possible ways are shown below.

Vertical cylinder (Ans B)

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

Horizontal cylinder (Ans A.B.D)

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

All Four Corners (Ans B.D)

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

The examples shown use four variable maps, but two or three variables can be used if required.