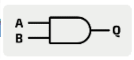
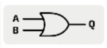
Boolean Algebra

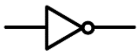
Circuit diagram: A logic diagram created using a selection of AND, OR and NOT gates. Constructing a Boolean expression from a logic diagram: Start with the output, and work backwards (post-order traversal). Why use binary: Easy to build electronic circuits and store data with only two states. We can also use brackets to group various parts of expressions. Equivalence: the same as. Converting Expression to circuit diagram: Start with the output and work backwards.

AND ∧ I 

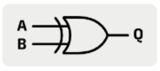
If both inputs are True, the output is True. Otherwise the output is False.

OR ∨ 

If both inputs are False, the output is False, otherwise the output is True.

NOT ¬ 

The NOT operation flips a value to its opposite.

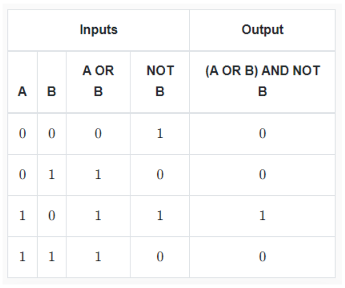
XOR ⊻ r 

If both inputs are the same, the output is False. Otherwise, the output is True.

Making Truth Tables

For example, (A OR B) AND (NOT B).

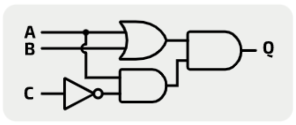
1. Create a table with n columns and 2^n rows (where n is the number of inputs)
2. Add all possible combinations of inputs for A and B
3. It is useful to count upwards to do this, in binary, from 0 to (2^n) - 1 inclusive
4. Add the intermediate and output Boolean expressions
5. Calculate the results



Inputs				Output
A	B	A OR B	NOT B	(A OR B) AND NOT B
0	0	0	1	0
0	1	1	0	0
1	0	1	1	1
1	1	1	0	0

Order of Operations:
1. Brackets
2. NOT
3. AND, NAND
4. OR, NOR, XOR

If multiple gates receive the same input, create a branch in the line using a dot as a connector. When you have overlapping lines, use a curve to show that the lines are not connected.



De Morgan's Law 1: A NOR gate with inputs A, B is equivalent to inverting A, B then passing them through an AND gate.

Boolean Algebra

De Morgan's Law 2: A NAND gate with inputs A, B is equivalent to inverting A, B then passing them through an OR gate
Distribution: Allows you to multiply or factor out an expression
Association: Allows you to group variables into brackets
Commutation: The order of variables in an expression doesn't matter.
Absorption: Allows you to reduce a slightly complicated expression into an equivalent simpler one
Double Negation: A is equal to NOT(NOT A)
General Rules: X AND 0 = 0, X AND 1 = X, X OR 1 = 1

Applying De Morgan's Laws:

1. Change the operator (AND to OR, and vice versa)
2. Negate the variables on either side of the operator
3. Negate the whole statement and simplify

Boolean expressions describe electric circuits and selection statements. Using as few expressions as possible will reduce the circuit's size, manufacturing cost and power consumption.

Making a Karnaugh Map:

1. Create a grid
2. The number of rows is equal to 2 ^ (n), where n is the number of variables on the top row (maximum two)
3. The number of columns is equal to 2 ^ (n), where n is the number of variables on the left column (maximum two)
4. Count up from 0 to (2 ^ (n) - 1) inclusive
5. Order the values so that only one bit changes from one value to the next
6. Break up the equation at the ORs
7. Consider each part separately: write a 1 wherever each part is True
8. Write a 0 in the empty cells
9. Box any groups, where:
    a. Boxes are rectangles or squares
    b. Boxes can only contain 1s
    c. Boxes must be as large as possible
    d. Boxes of 2^n 1s
    e. Boxes can overlap
    f. Smallest possible number of boxes
    g. Boxes can wrap around the map
10. Consider each box separately
11. Create an expression for each box by discarding the variables that change and keeping the ones that don't
12. Join the expressions together using ORs

A flip-flop: Stores 1 bit and flips it between 0 and 1. A 1-bit memory device that's enabled and disabled by a clock signal.

- Single-bit data Input (D)
- A clock signal Input (C)
- A single-bit data Output (Q)
- The inverse of the data Output (NOT(Q))

The clock signal is provided by another circuit that changes state at timed pulses synchronised by the computer's internal clock. When the clock pulse is at a rising or positive edge, the new input value is stored. Otherwise, it stores the same value (Q), due to an internal loop. They are used in SRAM cells to store data.

Half Adder: Adds two binary bits together. Full Adder: Adds two binary bits and a previous carry bit together. Consists of two half adders joined with an OR gate. *N* full adders connected together can add two binary numbers of *N* bits together.