Chapter 4 | Logic

● Introduction ● Boolean Algebra ● Relay ● Building Logic Gates ● Combinational Circuits ● Combining Local Gates ● Conversion Between Boolean Expression, Logical Gates and Truth Table ● Simplification of Combinational Circuits (Gate Reduction) Recapitulation

CS102 Computer Environment

Introduction

Boolean Algebra

Georges Boole

The Mathematician George Boole (1815 – 1864) was born in England. In 1847 Boole wrote "The Mathematical Analysis of Logic, Being an Essay towards a Calculus of Deductive Reasoning". In 1854 he wrote "An Investigation of the Law of Thought on which are Founded the Mathematical Theories of Logic and Probabilities", more known as the "The Laws of Thought".

Boole was primarily interested in developing a mathematical-style "algebra" to replace Aristotelian syllogistic logic. So, he came with a notational system to permit an algebraic manipulation of logical statements. Such manipulation can demonstrate whether a statement is true or false.

He also showed how a complicated statement can be rephrased into a simpler statement without changing its meaning

When Boole formulated his new algebra he did not foresee that someone will use it to build computers. In 1938 and while at MIT, Claude Elwood Shannon first described in his Master's thesis a relationship between the algebra of Boole and relay circuits.

The thesis was entitled "A Symbolic Analysis of Relay and Switching Circuits". Relays were invented earlier and used in the invention of the telegraph by Samuel Morse in 1836.

Digital systems are composed of combinations of logical gates that can be described by a truth table, a Boolean expression, or a logic symbol diagram.

Truth Table

They are used to define or test a logical circuit.

Input		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Boolean expression

Is an algebraic Boolean equation composed of operands and logical operators?

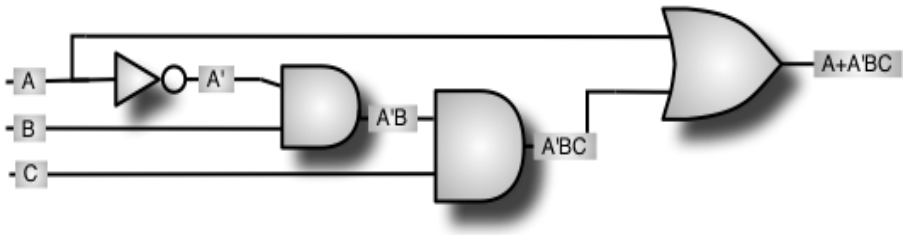
Output = function (Input)

It can be used to define or analyze a logical circuit

Example: X = AB'(C+D)

Logic symbol diagram

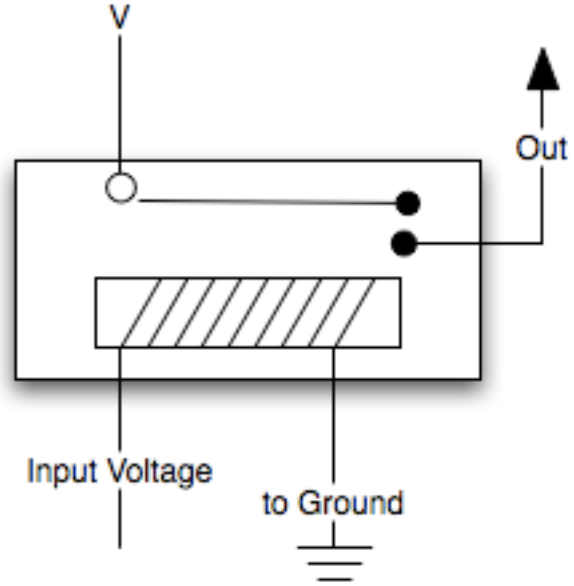
It is the implementation with interconnected gates. It has several inputs at the leftmost end, and 1 output at the rightmost end.



Relay

In 1836, Samuel F. Morse (born 1791 in Massachusetts) invented the telegraph (which means "far writing"). The idea is fairly simple: you do something at one end of a wire (apply a current) that causes something to happen (flashing a light) at the other end of the wire. At the core of a telegraph is a relay; its function relies on electromagnetism.

A relay is an electrically controlled switch. Current flowing through the coil of the relay creates a magnetic field which attracts a lever and changes the switch contacts.



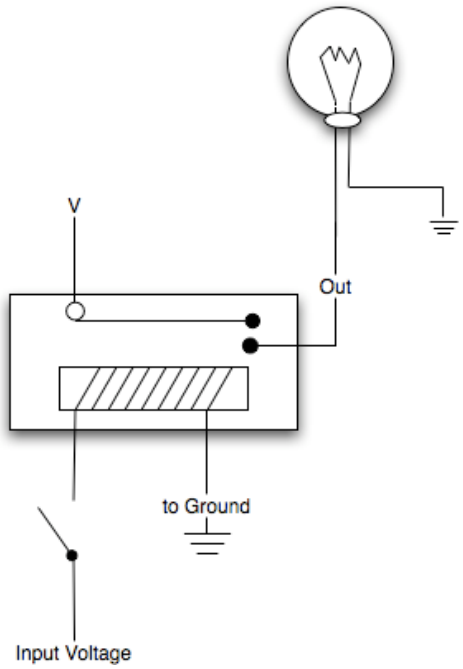
The coil current can be on or off, so a relay has two switch positions.

Relays allow one circuit to switch a second circuit, which can be completely and remotely separate from the first.

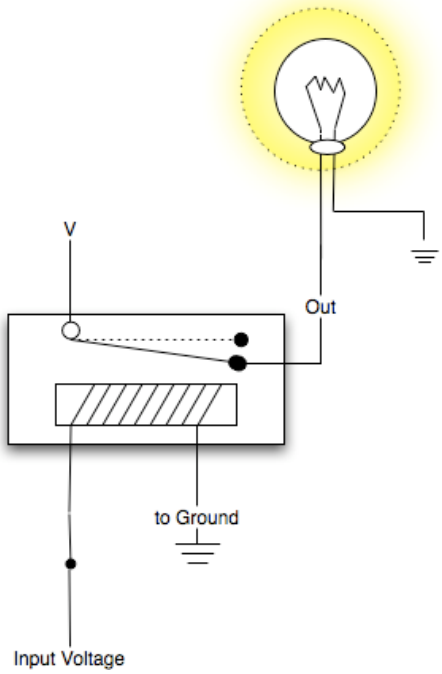
For example a low voltage battery circuit can use a relay to switch an 110V circuit. There is no electrical connection inside the relay between the two circuits; the link is both magnetic and mechanical.

Relay circuits

Switch open



Switch closed



Building logic gates

Using relays the following simple "machines" (called gates) could be manufactured:

- AND gate
- OR gate
- NOT gate or Inverter

More complex gates can be manufactured by combining simple gates. The following gates could be manufactured:

- NAND gate
- NOR gate
- XOR gate
- XNOR gate

The relationship between gates functionality and Boolean algebra has been demonstrated by Claude Shannon using relay circuits.

NOT gate (Inverter)

The Inverter is the simplest logic gate.
The logical circuit has 1 input and 1 output.
The output is the opposite of the input.

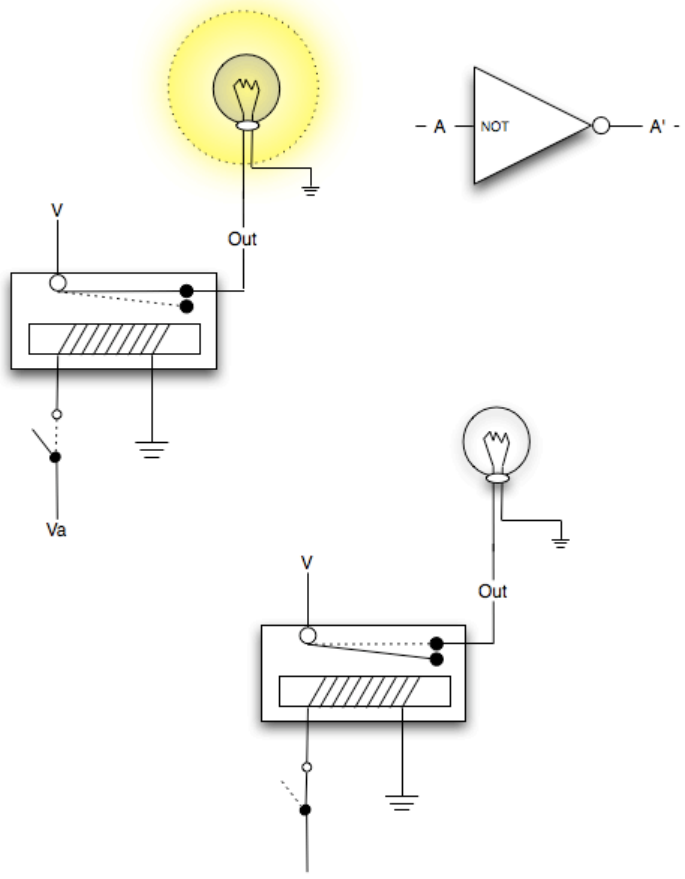
Truth Table

A	Not
0	1
1	0

Boolean Expression

Out = A'

Relay Circuit and Logic Symbol



AND gate

The basic logical AND gate has has 2 inputs and 1 output. Whenever at least one of the inputs is low, the output voltage will be low. If both inputs are high, then the output voltage will be high.

Some AND gates have 3 or more inputs.

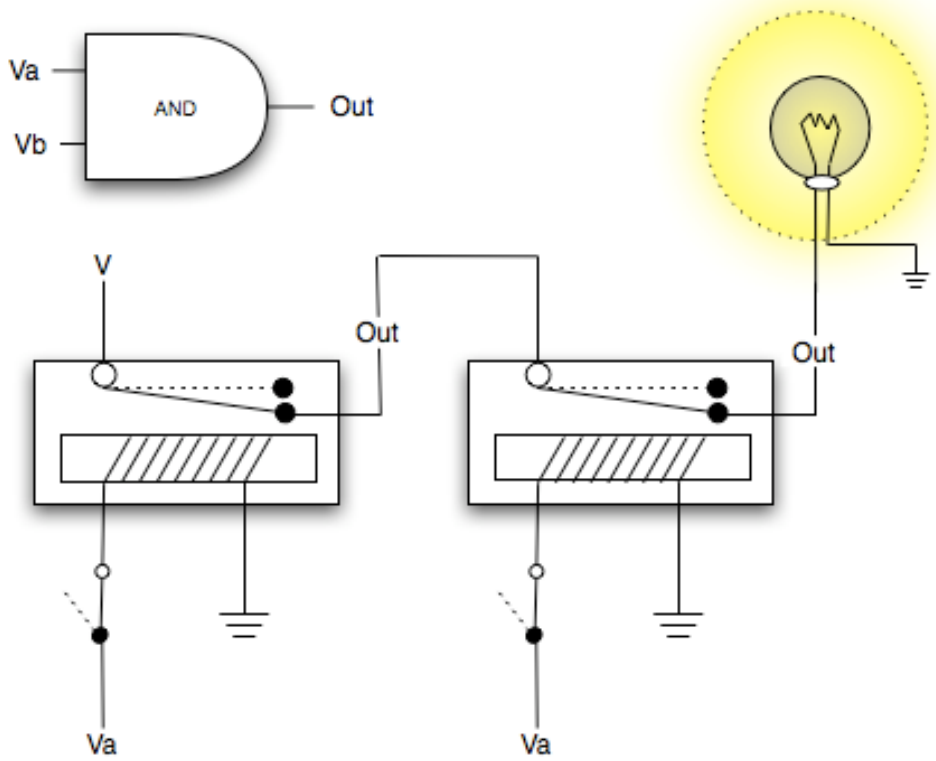
Truth Table

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

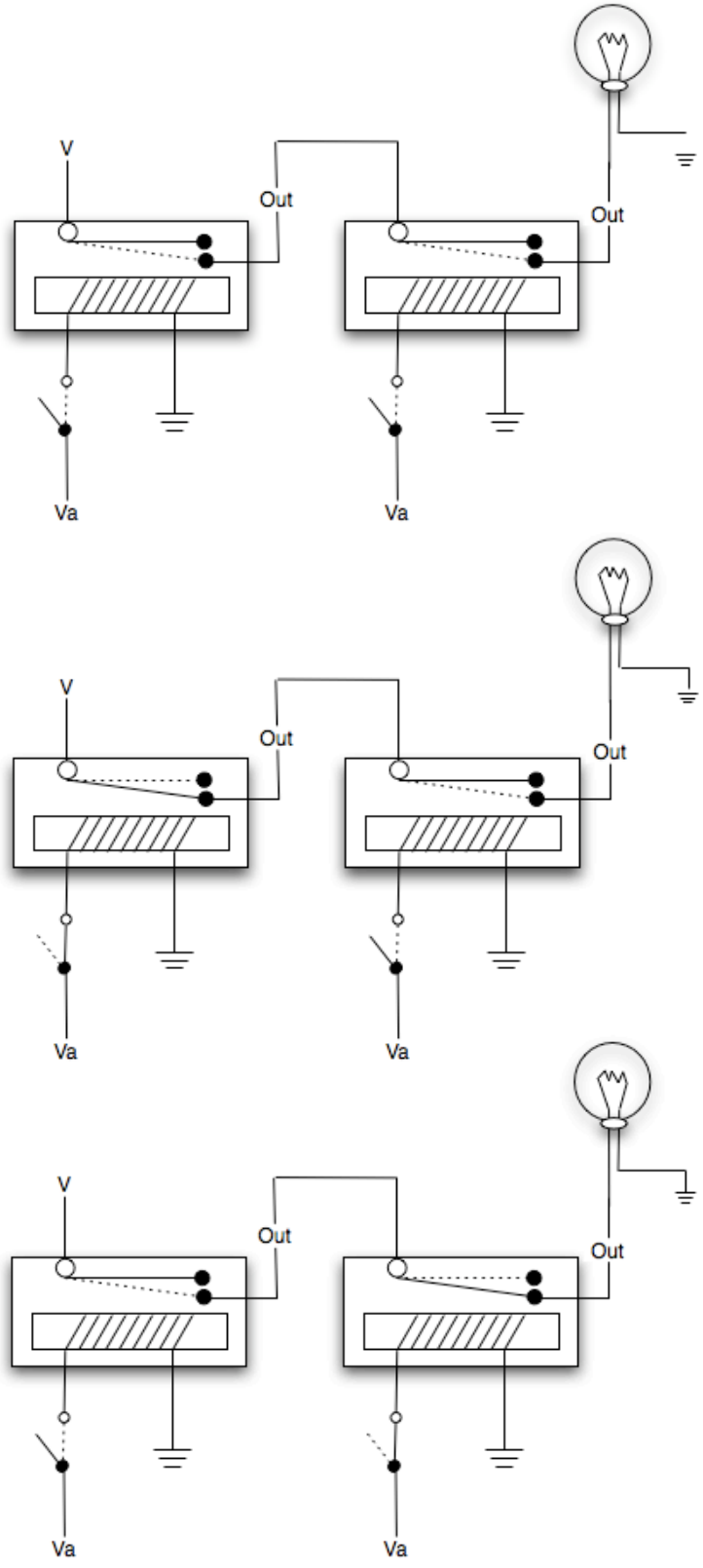
Boolean expression

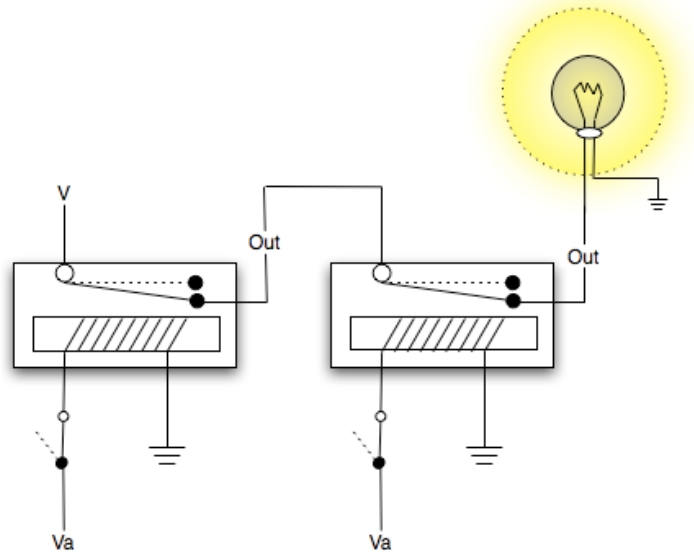
Out = A * B

Relay Circuit and Logic Symbol



Constructing the AND Relay circuit





OR gate

The basic logical OR gate has 2 inputs and 1 output.
Whenever at least one of the inputs is high, the output voltage will be high.
If both inputs are low, then the output voltage will be low.
Some OR gates have 3 or more inputs.
Note that this is an inclusive OR function.

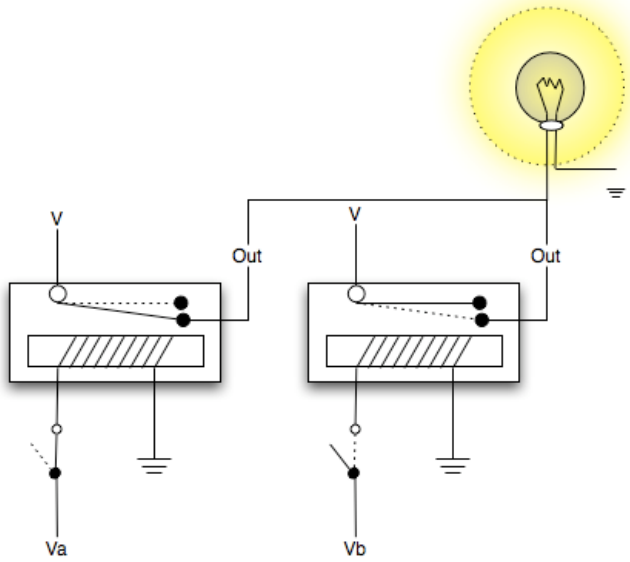
Truth Table

A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1

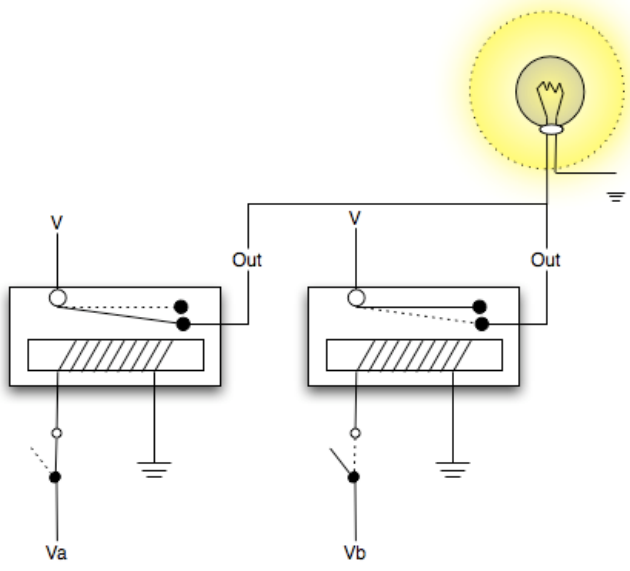
Boolean Expression

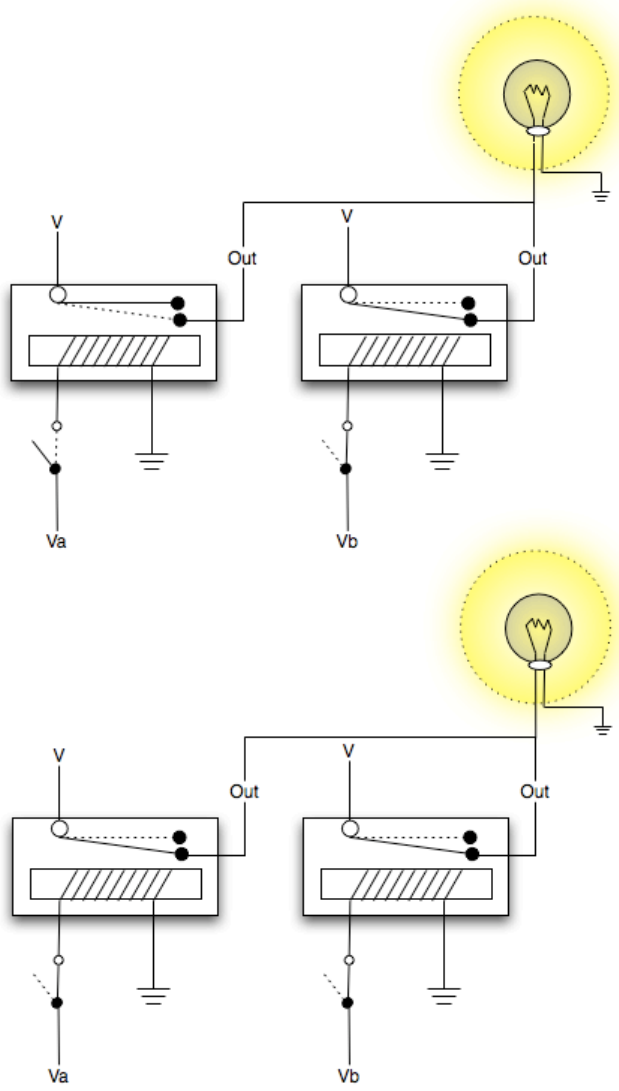
Out = A + B

Relay Circuit and Logic Symbol



Constructing the OR Relay Circuit

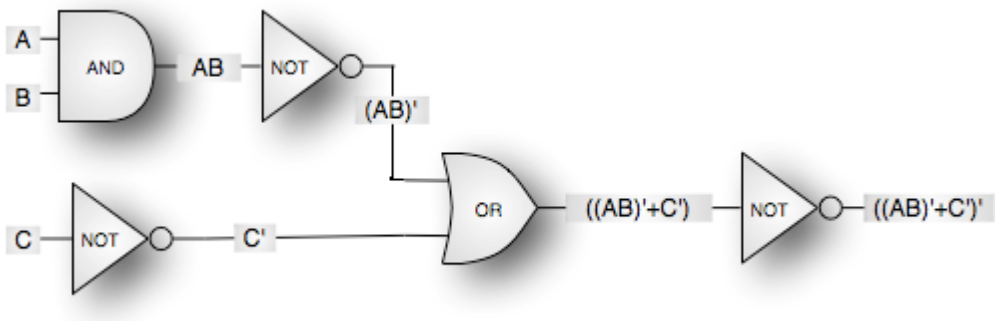




Combinational circuits

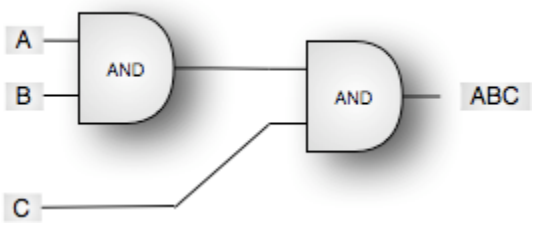
Logic Gates implement basic Boolean operations (AND, OR, NOT, NAND...). The output voltage is on or off. It is based on input voltage being on or off. Logic gates are elementary building blocks. They can be connected to build more complex circuits (adder or subtractor)

Combinational circuits are hardware circuits providing the logical function for electronic systems. These circuits are interconnected set of various gates where outputs connected to other inputs. Combinational circuits perform a logical function: if you change the inputs, you get a new output. The output is a Boolean function of a combination of inputs that can be described as a Boolean expression, a truth table, or a logical diagram.



A	B	C	AB	C’	AB’	((AB’)+c’)	((AB’)+c’)’
0	0	0	0	1	1	1	0
0	0	1	0	0	1	1	0
0	1	0	0	1	1	1	0
0	1	1	0	0	1	1	0
1	0	0	0	1	1	1	0
1	0	1	0	0	1	1	0
1	1	0	1	1	0	1	0
1	1	1	1	0	0	0	1

Compare this circuit to the following one, you will find that they are equivalent, they produce similar outputs:



A	B	C	AB	ABC
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	0	0
1	1	0	1	0
1	1	1	1	1

Combining logical gates

NAND gate

The logical NAND gate has 2 inputs and 1 output. It is equivalent to an AND gate followed by a NOT gate. Whenever at least one of the inputs is low, the output voltage will be high.

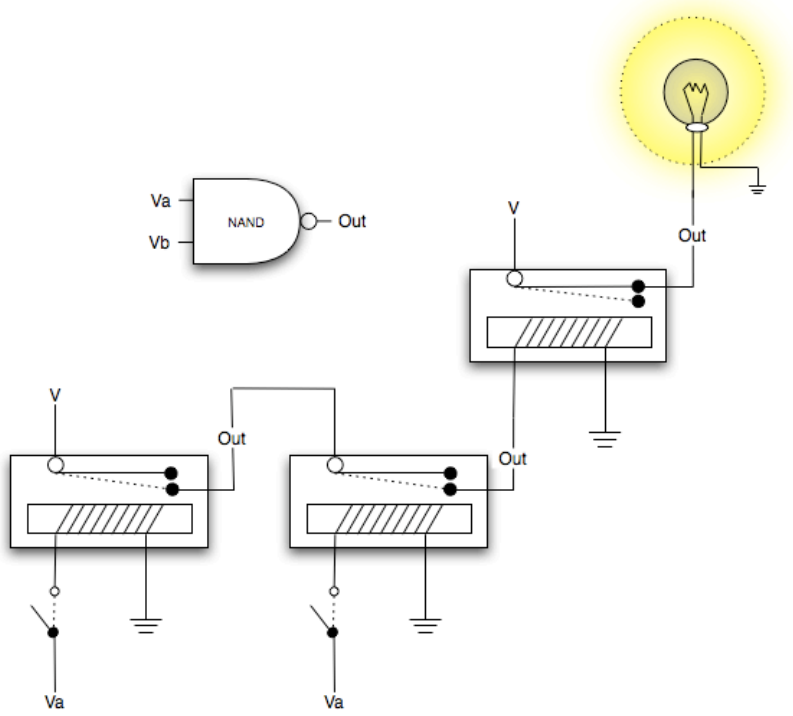
If both inputs are high, then the output voltage will be low.

Truth Table

A	B	NAND
0	0	1
0	1	1
1	0	1
1	1	0

Boolean Expression
Out = (A * B)'

Relay Circuit and Logic Symbol



NOR gate

The logical NOR gate has 2 inputs and 1 output. It is equivalent to an OR gate followed by a NOT gate. Whenever at least one of the inputs is high, the output voltage will be low.

If both inputs are low, then the output voltage will be high.

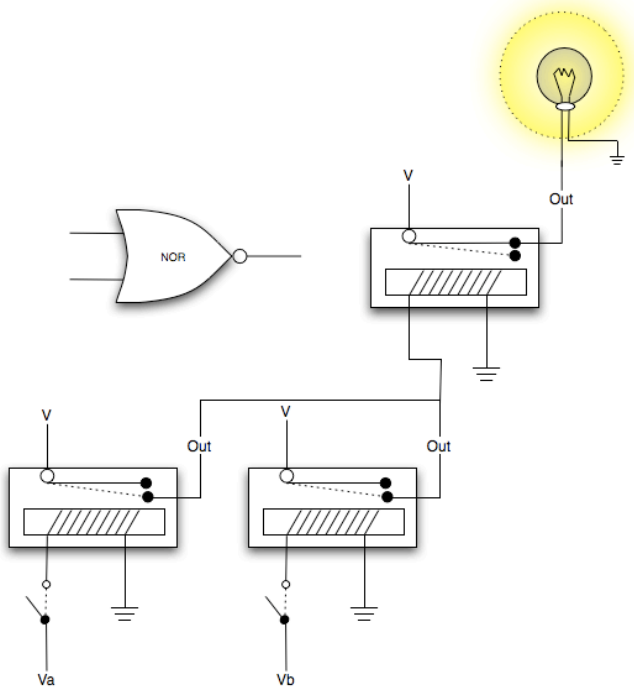
Truth Table

A	B	NOR
0	0	1
0	1	0
1	0	0
1	1	0

Boolean Expression

Out = (A + B)'

Relay Circuit and Logic Symbol



XOR gate

Called exclusive OR gate. The logical XOR gate has 2 inputs and 1 output. Whenever only one of the inputs is high, the output voltage will be high.

If both inputs are low, then the output voltage will be low.
If both inputs are high, then the output voltage will be low.

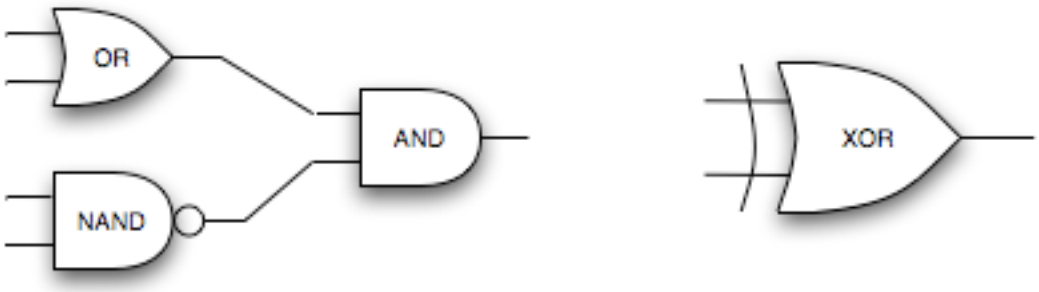
Truth Table

A	*B*	*XOR*
0	0	0
0	1	1
1	0	1
1	1	0

Boolean expression

Out = (A + B) * (A * B)'

Relay Circuit and Logic Symbol



XNOR gate

The exclusive NOR gate. The logical XNOR gate has 2 inputs and 1 output. It is equivalent to an XOR gate followed by a NOT gate. Whenever only one of the inputs is high, the output voltage will be low.

If both inputs are low, then the output voltage will be high.
If both inputs are high, then the output voltage will be high.

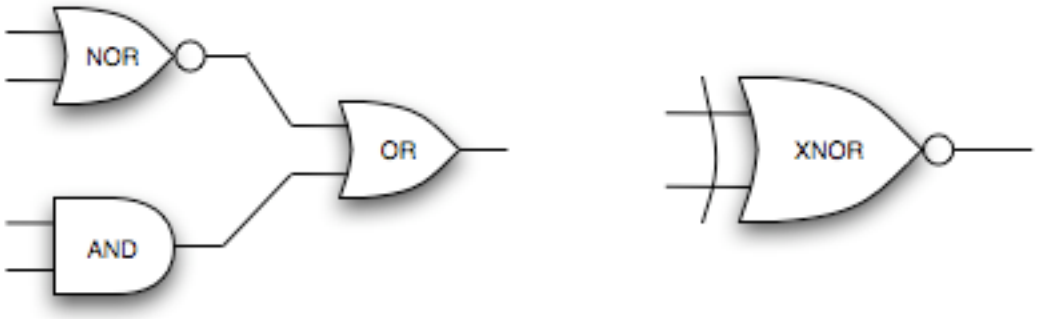
Truth Table

A	*B*	*XNOR*
0	0	1
0	1	0
1	0	0
1	1	1

Boolean Expression

Out = (A + B)' + (A * B)

Relay Circuit and Logic Symbol



Conversion between Boolean expression, Logical Gates, and Truth table

From Boolean Expression to a Truth Table

Analyze the Boolean expression and decide how many columns will be in the truth table: Each variable gets an input column.

When all the variables have been allocated an input column in the table, fill in all the possible input (combinations of 0s and 1s). Note that 3 variables = 2^3 = 8 combinations.

Add columns for sub-expressions as desired (inside-out).

Fill in the table usually by column.

The number of the output columns or their order does not matter, but try to make the output columns easy to evaluate.

Example1: X = A'B + ABC'

A	*B*	*C*	*(A'B)*	*(ABC')*	*X*
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	1	0	1
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	1	1
1	1	1	0	0	0

Example2: X = AB'(A+BC')

A	*B*	*C*	*BC'*	*(A+BC')*	*(AB')*	*X*
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	1	1	0	0
0	1	1	0	0	0	0
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

From Truth Table to a Boolean Expression

Example 1:

A	*B*	*C*	*X*
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

X = A'BC + AB'C'

Notice: when is the output a 1? For each combo (row) of inputs that leads to a 1!

Example: A is 0 AND B is 1 AND C is 1 => A'BC is 1
OR A is 1 AND B is 0 AND C is 0 => AB'C' is 1

Inspect the truth table and start from the first row.

For all the input variables in a given row whose output is 1:
If the value of variable P is 1, then write P
If the value of variable P is 0, then write P'

Connect all the input variables in the row with the '*' operator.

Repeat for all the rows in the truth table where the output is 1.

When all rows (with output =1) have been translated to Boolean expressions, connect these expressions with the '+' operator.

What you end up with is a Sum of Products (SOP): OR-ing of several product terms for which the output X = 1; where each term is the product (AND-ing) of all input variables and/or their complements.
Example2:

A	***B***	***C***	***X***
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

X = A'BC + AB'C + ABC'

From Truth Table to a Logical Circuit

Determine the Boolean expression as described earlier.

Then, convert to a logical circuit.

Example1:

A	*B*	*C*	*X*
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

From Logical Circuit to a Boolean Expression

Work out the circuit from the leftmost gate (input) and work forward towards the right (output).

Build the corresponding Boolean expression from the innermost to the outermost level:

- The leftmost gates (inputs) are the innermost operators of the Boolean expression.
- The rightmost gate (the output) is the outermost operator of the Boolean expression.

The inputs of the circuit are assigned to variables of the Boolean expression.

Write the Boolean operator that correspond to each gate:

- OR gate translates to a +
- AND gate translates to a *
- NOT gate translates to a '

Add parenthesis as desired, especially around the innermost expressions.

The order of the expressions does not matter but try to make the expression easy to read.

From Logical Circuit to a Truth Table

Determine the Boolean expression for the logic circuit as described earlier.

Then fill in the truth table as described earlier.

Simplification of combinational circuits (Gate reduction)

Combinational circuits can become too large to handle.
Optimizing the circuit is to eliminate gate redundancy.
Gates are manufactured and integrated into chips.
Fewer gates means smaller, less expensive and faster circuit

Simplification can be:

- Achieved manually using Boolean theorems & axioms
- Automated using tabular methods (K-map and Quine-McKluskey methods)

Ex1: A (A+A'B) = A
Ex2: A'BC' + A'BC + ABC' = B (A'+C')
Ex3: (A+B'+C) (A+B'C) = A+B'C
Ex4: A'B'C + A'BC + AB'C + ABC = C
Ex5: AB + (AC)' + AB'C (AB + C) = 1 (DM)

Recapitulation

So far we covered binary numbering system, and the electricity leading to logical gates implementing the Boolean algebra. Since the binary numbering system can be represented using electricity, in the next chapter we will build a combinational logic circuit that will take as input two numbers and produce their sum as an output. The adder is the core unit of the CPU (Arithmetic unit). Similarly, logical gates are used to build the logical unit. Usually, they are called the ALU, the Arithmetic and Logical unit.