Circuits

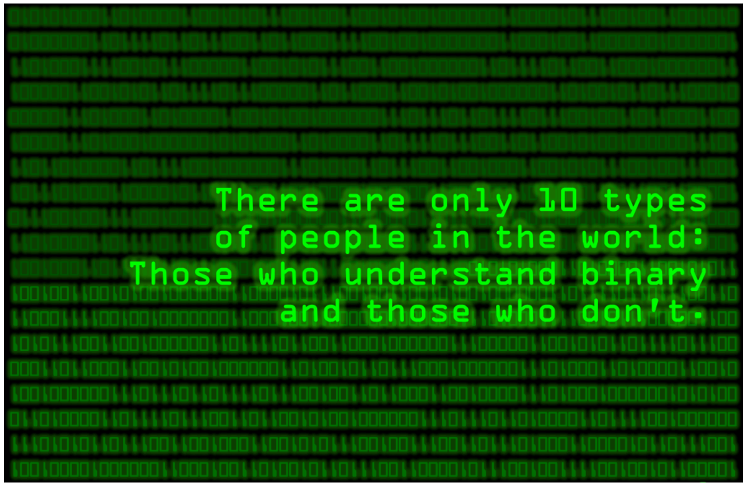
The Language of Computers

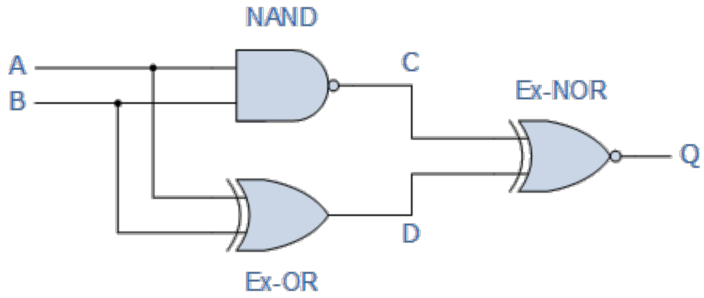
Binary

- 1's and 0's
- Binary Digit = Bit

Boolean Logic

- George Boole
- Logic for binary variables
- True/False or High/Low, 1/0, On/Off
- 1 bit does not a computer make!





Circuit

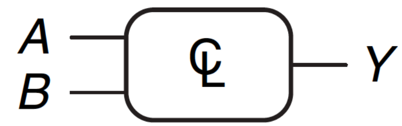
Black-box representation

- One or more inputs
- One or more outputs
- Performs some operation

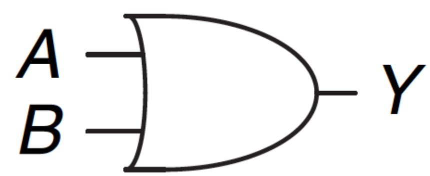
Combinational Circuit

Outputs depend only on the current value of the inputs

How could this not be the case?

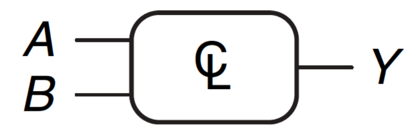


$$Y = F(A, B) = A + B$$



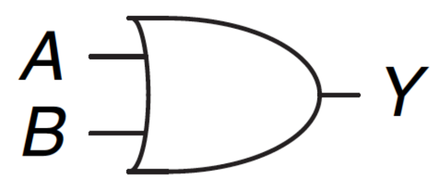
Combinational Circuit

Outputs depend only on the current value of the inputs



$$Y = F(A, B) = A + B$$

How could this not be the case?



Answer: if the circuit has *memory* somewhere inside.

Combinational Circuit

We will see later what rules can be enforced to ensure a circuit is combinational

Why bother?

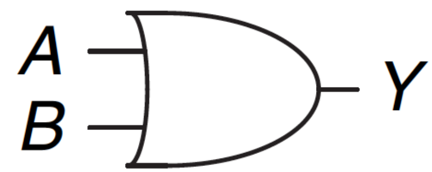
By restricting ourselves to combinational circuits, we make the behavior of circuits very easy to describe using **truth tables**

Since outputs depend only on inputs, simply list all input-output pairs. We are working with discrete systems, so there will be a finite number

Truth Tables

Characterize the behavior of a group of inputs and outputs

- 0's and 1's
- 1 column per variable
- N variables 2^N rows



A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

Lists all possible values of the variables

- x possible values of 0 or 1
- xy possible values of 0 0, 0 1, 1 0, 1 1
- xyz 0 0 0, 0 0 1, 0 1 0, 0 1 1, 1 0 0, 1 0 1, 1 1 0, 111

Truth Tables

Truth tables, however, are not the only way to represent a combinational circuit

When limiting ourselves to combinational circuits, all three of these are equally expressive

- Truth tables
- Boolean algebra
- Logic gates

We will need to understand all three, including their pros and cons and how to switch between them

Boolean Expressions

Expression that produces a Boolean value when evaluated

Examples:

X > 1

X

X > Y || Y == Z

(X && Y) + (Z + X)

ABC + D

Logic Gates

Simple digital circuits that take one or more binary inputs and produce a binary output

Drawn with a symbol showing inputs and outputs

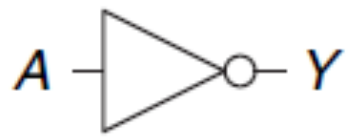
- Inputs at left or top
- Outputs at right or bottom

NOT

Inverter

- Changes 1 to 0, vice versa

NOT



$$Y = \overline{A}$$

A	Y
0	1
1	0

AND

Output true when both inputs are true

Other Forms:

Y = A * B,

y = AB,

y = A (intersection) B

AND

$Y = AB$

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

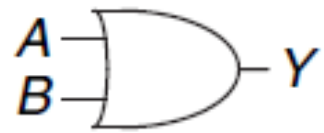
OR

Output true when A, B, or both are 1

Other Forms: Y = A + B,

Y = A (union) B

OR



$Y = A + B$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

XOR

Output true only when A is 1 and B is 0, or A is 0 and B is 1

Multiple inputs make it a parity checker, true when odd number of inputs are 1, 0 if even

XOR

$$Y = A \oplus B$$

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

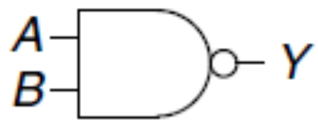
Other Gates

NOR

$Y = \overline{A + B}$

A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

NAND

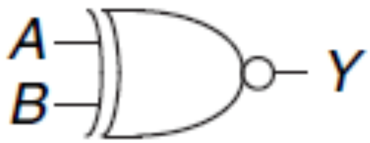


$Y = \overline{AB}$

A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

XNOR

XNOR



$$Y = \overline{A \oplus B}$$

A	B	Y
0	0	
0	1	
1	0	
1	1	

Multiple Input Gates

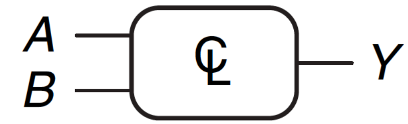
Adding more than two inputs possible for most gates.

For example, a 5-input AND gate would output 1 only if all 5 inputs were 1

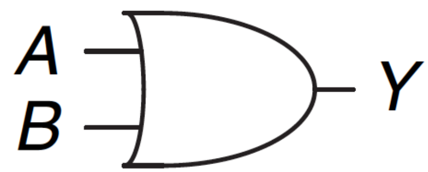
Similarly, a 5-input OR gate would output 1 if *any* of the 5 inputs were 1

Combinational Circuit Rules

1. Every subcircuit is combinational
2. A wire cannot be connected to the output of two different subcircuits
3. No cyclic paths!



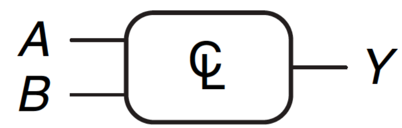
$$Y = F(A, B) = A + B$$



Combinational Circuit

Restricting ourselves to combinational circuits is an example of *discipline*

We limit our own options in order to make things easier to work with and understand



$$Y = F(A, B) = A + B$$