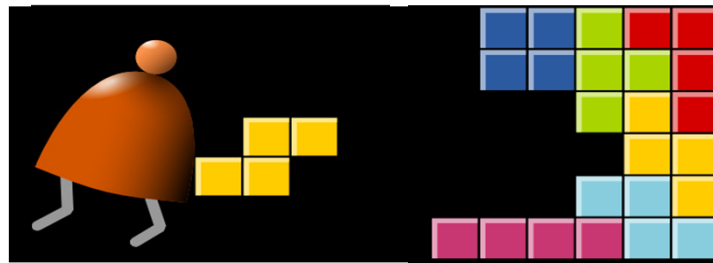


# Boolean Logic



*Building a Modern Computer From First Principles*

[www.nand2tetris.org](http://www.nand2tetris.org)

### **Usage and Copyright Notice:**

Copyright © Noam Nisan and Shimon Schocken

This presentation accompanies the textbook “The Elements of Computing Systems” by Noam Nisan & Shimon Schocken, MIT Press, 2005.

We provide 13 such presentations.

Each presentation is designed to support about 3 hours of classroom or self-study instruction.

You are welcome to use or edit this presentation as you see fit for instructional and non-commercial purposes.

If you use our book and course materials, please include a reference to [www.nand2tetris.org](http://www.nand2tetris.org)

If you have any questions or comments, please write us at [nand2tetris@gmail.com](mailto:nand2tetris@gmail.com)



This work is licensed under a  
[Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](https://creativecommons.org/licenses/by-nc-sa/3.0/).

# Boolean algebra

## Some elementary Boolean functions:

- Not(x)
- And(x,y)
- Or(x,y)
- Nand(x,y)

x	Not (x)
0	1
1	0

x	y	And (x, y)
0	0	0
0	1	0
1	0	0
1	1	1

x	y	Or (x, y)
0	0	0
0	1	1
1	0	1
1	1	1

x	y	Nand (x, y)
0	0	1
0	1	1
1	0	1
1	1	0

## Boolean functions:

x	y	z	$f(x, y, z) = (x + y)\bar{z}$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

- A Boolean function can be expressed using a functional expression or a truth table expression
- Important observation:  
Every Boolean function can be expressed using And, Or, Not.

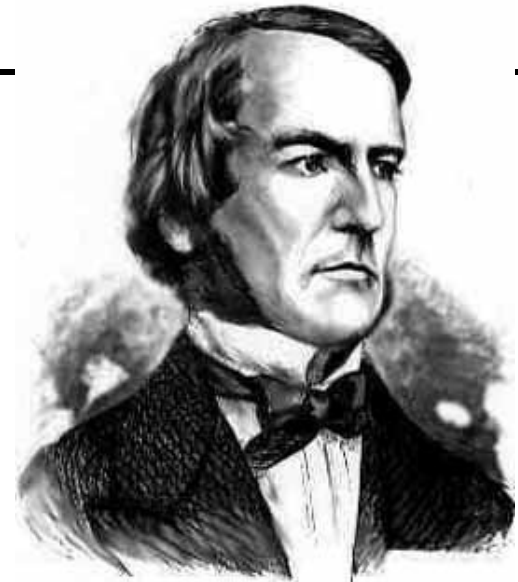
# Boolean algebra

---

Given:  $\text{Nand}(a, b)$ ,  $\text{false}$

We can build:

- $\text{Not}(a) = \text{Nand}(a, a)$
- $\text{true} = \text{Not}(\text{false})$
- $\text{And}(a, b) = \text{Not}(\text{Nand}(a, b))$
- $\text{Or}(a, b) = \text{Not}(\text{And}(\text{Not}(a), \text{Not}(b)))$
- $\text{Xor}(a, b) = \text{Or}(\text{And}(a, \text{Not}(b)), \text{And}(\text{Not}(a), b))$
- Etc.

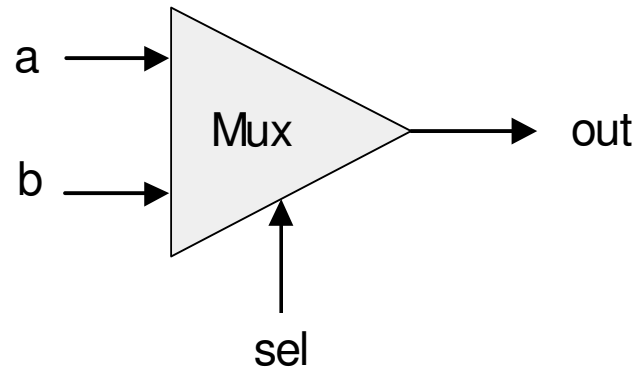


George Boole, 1815-1864  
*("A Calculus of Logic")*

# Multiplexer

---

a	b	sel	out
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	1
1	1	1	1



sel	out
0	a
1	b

Proposed Implementation: based on Not, And, Or gates.

# Example: Building an **And** gate



And.cmp

a	b	out
0	0	0
0	1	0
1	0	0
1	1	1

## Contract:

When running your .hdl on our .tst, your .out should be the same as our .cmp.

And.hdl

```
CHIP And
{
  IN  a, b;
  OUT out;
  // implementation missing
}
```

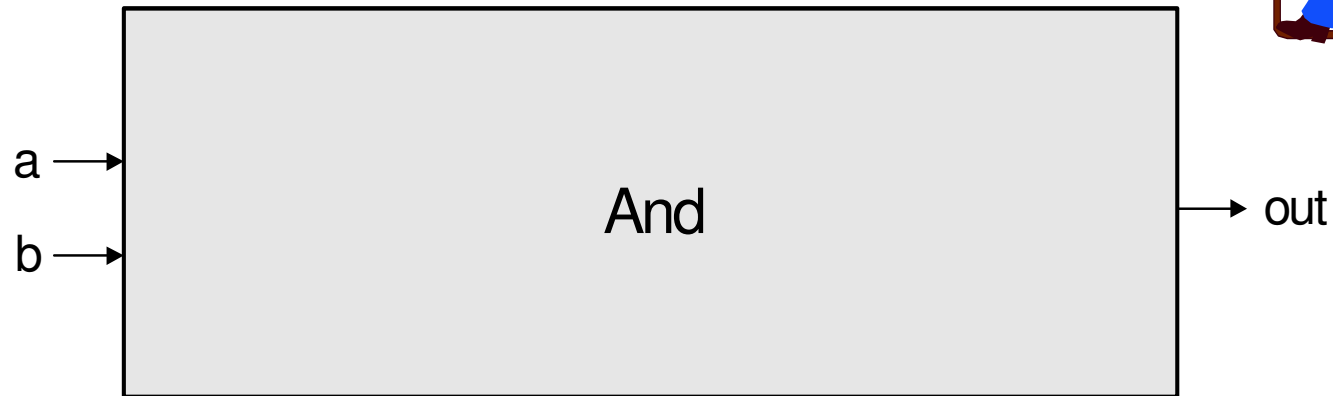
And.tst

```
load And.hdl,
output-file And.out,
compare-to And.cmp,
output-list a b out;
set a 0, set b 0, eval, output;
set a 0, set b 1, eval, output;
set a 1, set b 0, eval, output;
set a 1, set b 1, eval, output;
```

# Building an **And** gate



**Interface:**  $\text{And}(a,b) = 1$  exactly when  $a=b=1$



And.hdl

```
CHIP And
{
  IN  a, b;
  OUT out;
  // implementation missing
}
```

# Building an **And** gate



Implementation:  $\text{And}(a,b) = \text{Not}(\text{Nand}(a,b))$



And.hdl

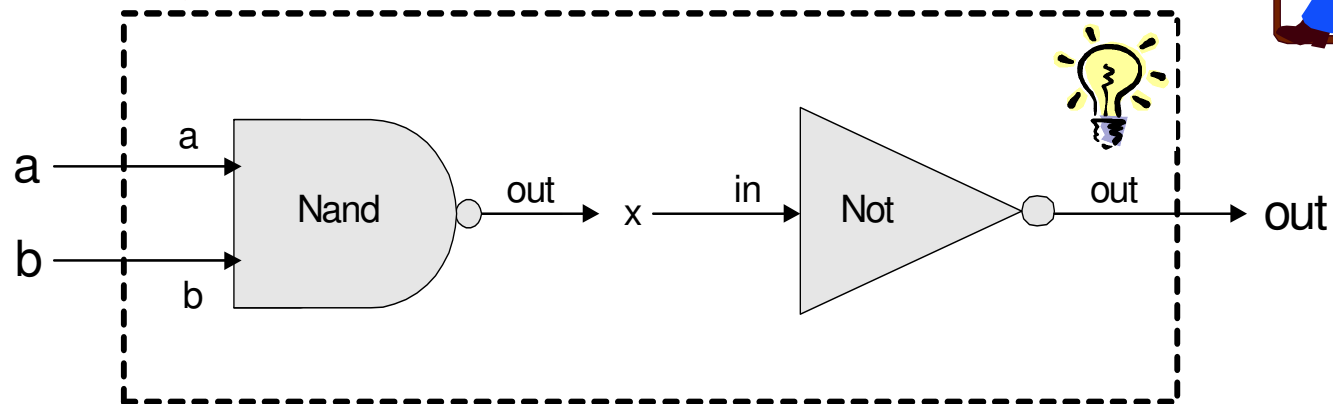
```
CHIP And
{
  IN  a, b;
  OUT out;
  // implementation missing
}
```



# Building an And gate



Implementation:  $\text{And}(a,b) = \text{Not}(\text{Nand}(a,b))$



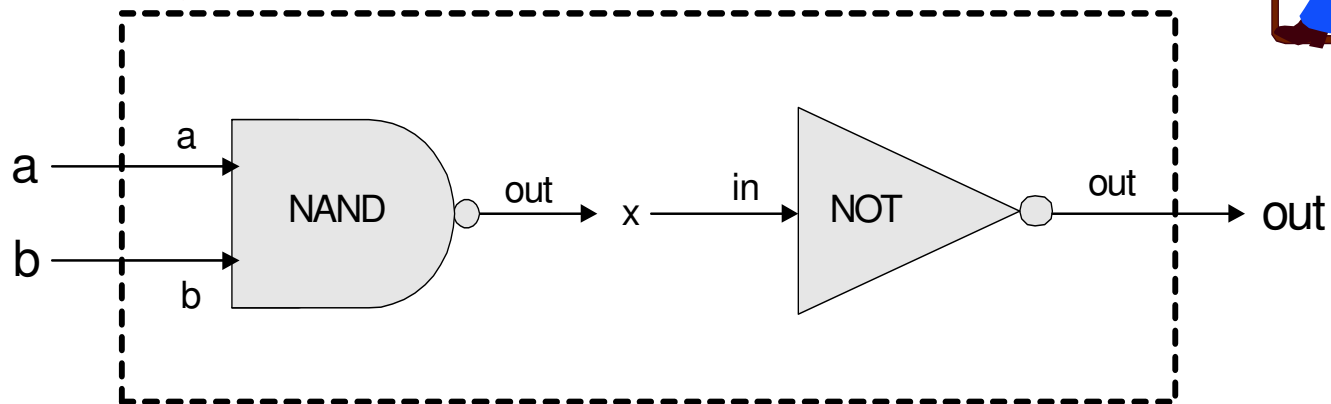
And.hdl

```
CHIP And
{
  IN  a, b;
  OUT out;
  // implementation missing
}
```

# Building an And gate



Implementation:  $\text{And}(a,b) = \text{Not}(\text{Nand}(a,b))$



And.hdl

```
CHIP And
{
  IN  a, b;
  OUT out;
  Nand(a = a,
        b = b,
        out = x);
  Not(in = x, out = out)
}
```



# Hardware simulator (demonstrating Xor gate construction)

Hardware Simulator - D:\hack\Chips\Project 1\Xor.hdl

File View Run Help

Chip Name:  Time: 0

Input pins		Output pins	
Name	Value	Name	Value
a	0	out	0
b	0		

**HDL program**

```
// Xor (exclusive or) gate
// if a<>b out=1 else out=0
CHIP Xor {
  IN a,b;
  OUT out;
  PARTS:
    Not (in=a,out=nota);
    Not (in=b,out=noth);
    And (a=a,b=noth,out=x);
    And (a=nota,b=b,out=y);
    Or (a=x,b=y,out=out);
}
```

Internal pins	
Name	Value
nota	1
noth	1
x	0
y	0

**test script**

```
load Xor,
output-file Xor.out,
compare-to Xor.cmp,
output-list a%B3.1.3 b%B3.1.3 out%B3.1.3;

set a 0,
set b 0,
eval,
output;

set a 0,
set b 1,
eval,
output;

set a 1,
set b 0,
eval,
output;

set a 1,
set b 1,
eval,
output;
```

Script restarted

# Hardware simulator

The screenshot shows a hardware simulator window titled "Hardware Simulator - D:\hack\Chips\Project 1\Xor.hdl". The interface includes a menu bar (File, View, Run, Help), a toolbar with navigation buttons (a red circle highlights the "Run" button), and a status bar at the bottom that reads "Script restarted".

The main workspace is divided into several sections:

- Input pins:** A table with columns "Name" and "Value".

Name	Value
a	0
b	0
- Output pins:** A table with columns "Name" and "Value".

Name	Value
out	0
- HDL:** A text area containing Verilog code for an XOR gate:

```
// Xor (exclusive or) gate
// if a<>b out=1 else out=0
CHIP Xor {
  IN a,b;
  OUT out;
  PARTS:
  Not (in=a,out=nota);
  Not (in=b,out=noth);
  And (a=a,b=noth,out=x);
  And (a=nota,b=b,out=y);
  Or (a=x,b=y,out=out);
}
```
- Internal pins:** A table with columns "Name" and "Value".

Name	Value
nota	1
noth	1
x	0
y	0
- Script Execution Log:** A text area on the right showing the execution of a script. The first line, "load Xor,", is highlighted in yellow. Subsequent lines show the simulation of the circuit with different input combinations (a, b) and the resulting output (out).

```
load Xor,
output-file Xor.out,
compare-to Xor.cmp,
output-list a%B3.1.3 b%B3.1.3 out%B3.1.3;

set a 0,
set b 0,
eval,
output;

set a 0,
set b 1,
eval,
output;

set a 1,
set b 0,
eval,
output;

set a 1,
set b 1,
eval,
output;
```

# Hardware simulator

The screenshot shows a hardware simulator window titled "Hardware Simulator - D:\hack\Chips\Project 1\Xor.hdl". The interface includes a menu bar (File, View, Run, Help), a toolbar with simulation controls (Play, Stop, Step, Slow, Fast), and a status bar at the bottom.

**Chip Name:** Xor **Time:** 0

**Input pins:**

Name	Value
a	1
b	1

**Output pins:**

Name	Value
out	0

**HDL:**

```
// Xor (exclusive or) gate
// if a<>b out=1 else out=0
CHIP Xor {
  IN a,b;
  OUT out;
  PARTS:
    Not (in=a,out=nota);
    Not (in=b,out=notb);
    And (a=a,b=notb,out=x);
    And (a=nota,b=b,out=y);
    Or (a=x,b=y,out=out);
}
```

**Internal pins:**

Name	Value
nota	0
notb	0
x	0
y	0

**Script:**

```
load Xor,
output-file Xor.out,
compare-to Xor.cmp,
output-list a%B3.1.3 b%B3.1.3 out%B3.1.3;

set a 0,
set b 0,
eval,
output;

set a 0,
set b 1,
eval,
output;

set a 1,
set b 0,
eval,
output;

set a 1,
set b 1,
eval,
output;
```

**Output file:**

a	b	out
0	0	0
0	1	1
1	0	1
1	1	0

**Status:** End of script - Comparison ended successfully