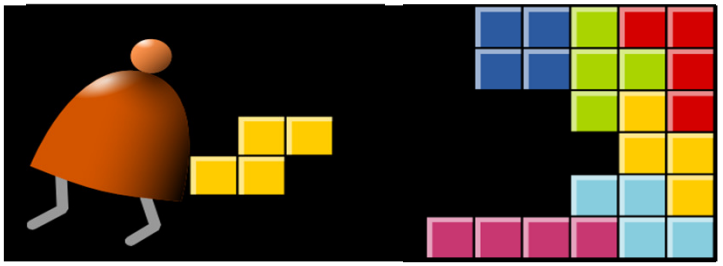
Boolean Logic



Building a Modern Computer From First Principles

www.nand2tetris.org

Usage and Copyright Notice:

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

If you have any questions or comments, please write us at nand2tetris@gmail.com

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

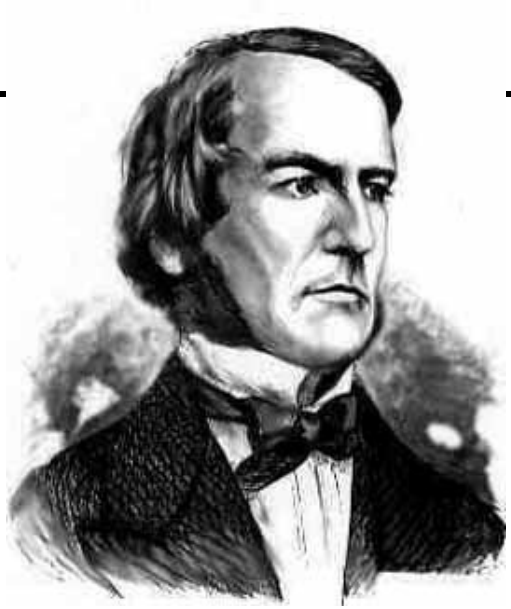
- A Boolean function can be expressed using a functional expression or a truth table expression
- Important observation:
  Every Boolean function can be expressed using And, Or, Not.

Boolean algebra

Given: **Nand(a,b), false**

We can build:

- **Not(a) = Nand(a,a)**
- **true = Not(false)**
- **And(a,b) = Not(Nand(a,b))**
- **Or(a,b) = Not(And(Not(a),Not(b)))**
- **Xor(a,b) = Or(And(a,Not(b)),And(Not(a),b))**
- Etc.

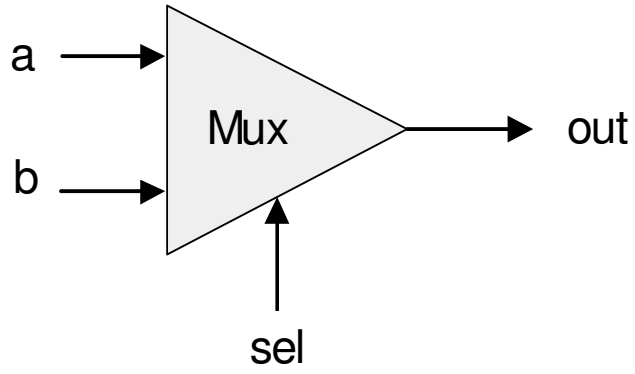


George Boole, 1815-1864

("*A Calculus of Logic*")

Multiplexer

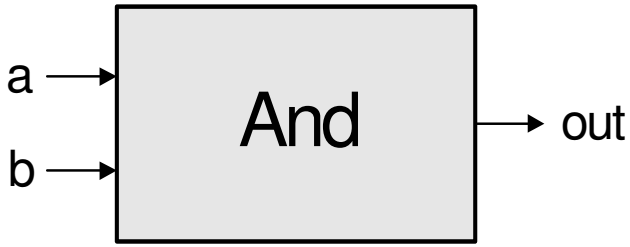
a	b	sel	out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1



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

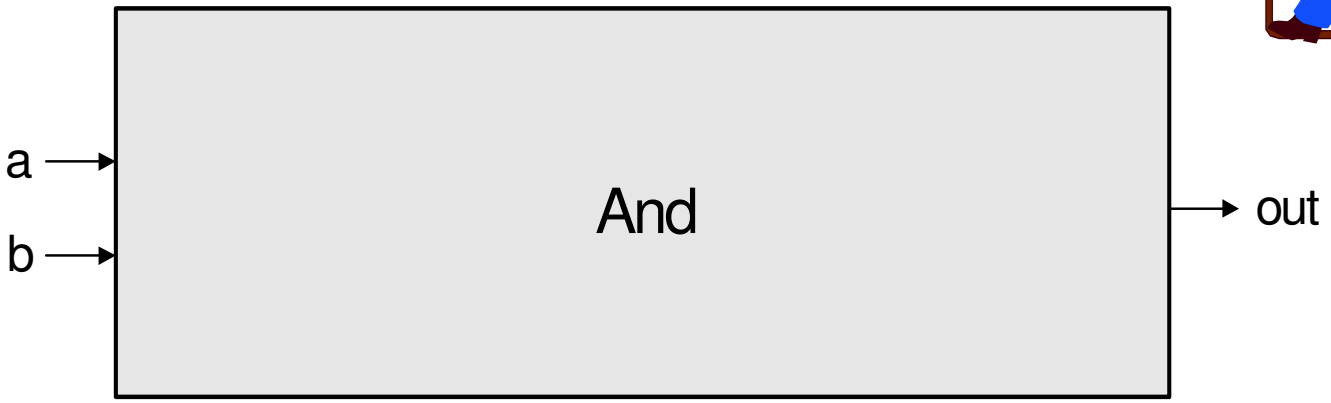
```
CHIP And
{    IN  a, b;
     OUT out;
     // implementation missing
}
```

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: And(a,b) = 1 exactly when a=b=1



And.hdl

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

Building an **And** gate

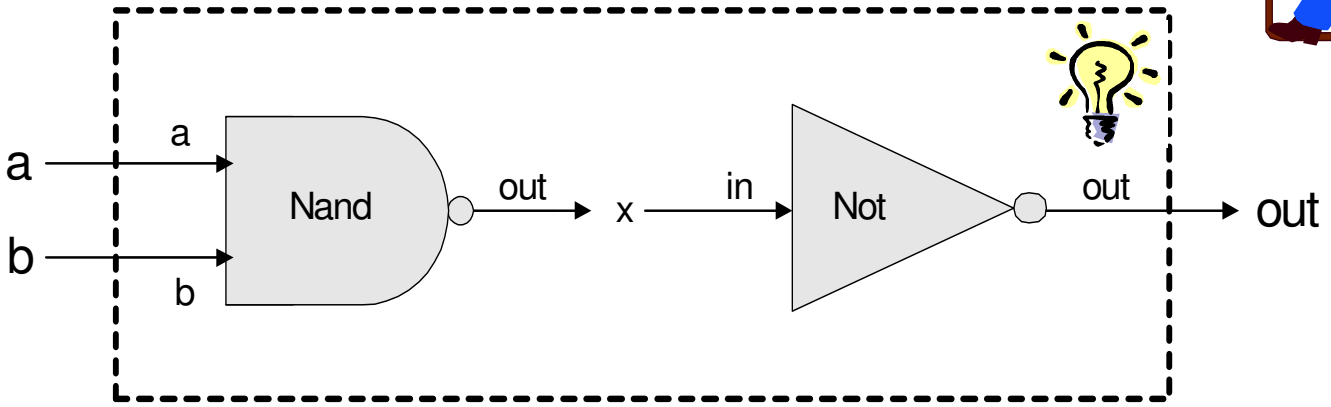
Implementation: And(a,b) = Not(Nand(a,b))

And.hdl

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

Building an **And** gate

Implementation: And(a,b) = Not(Nand(a,b))

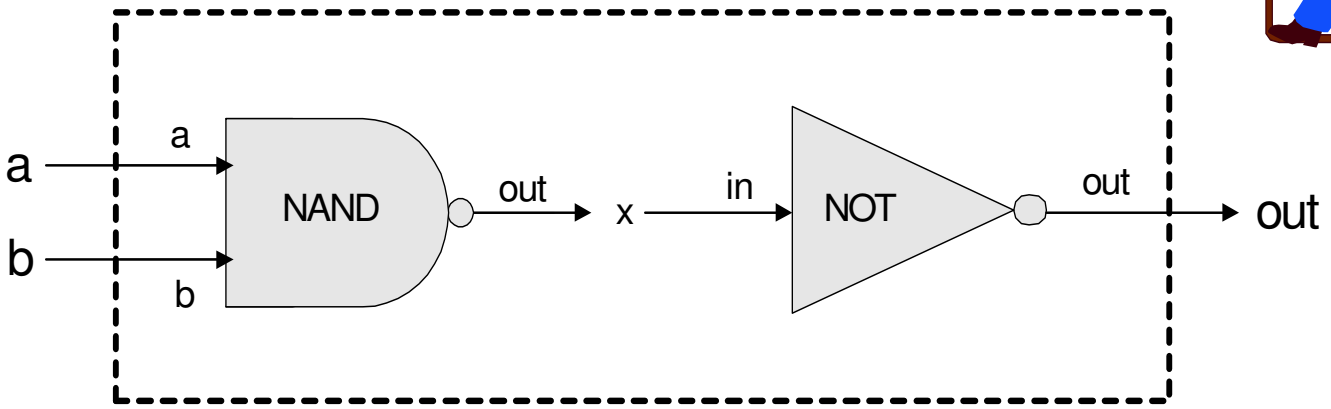


And.hdl

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

Building an **And** gate

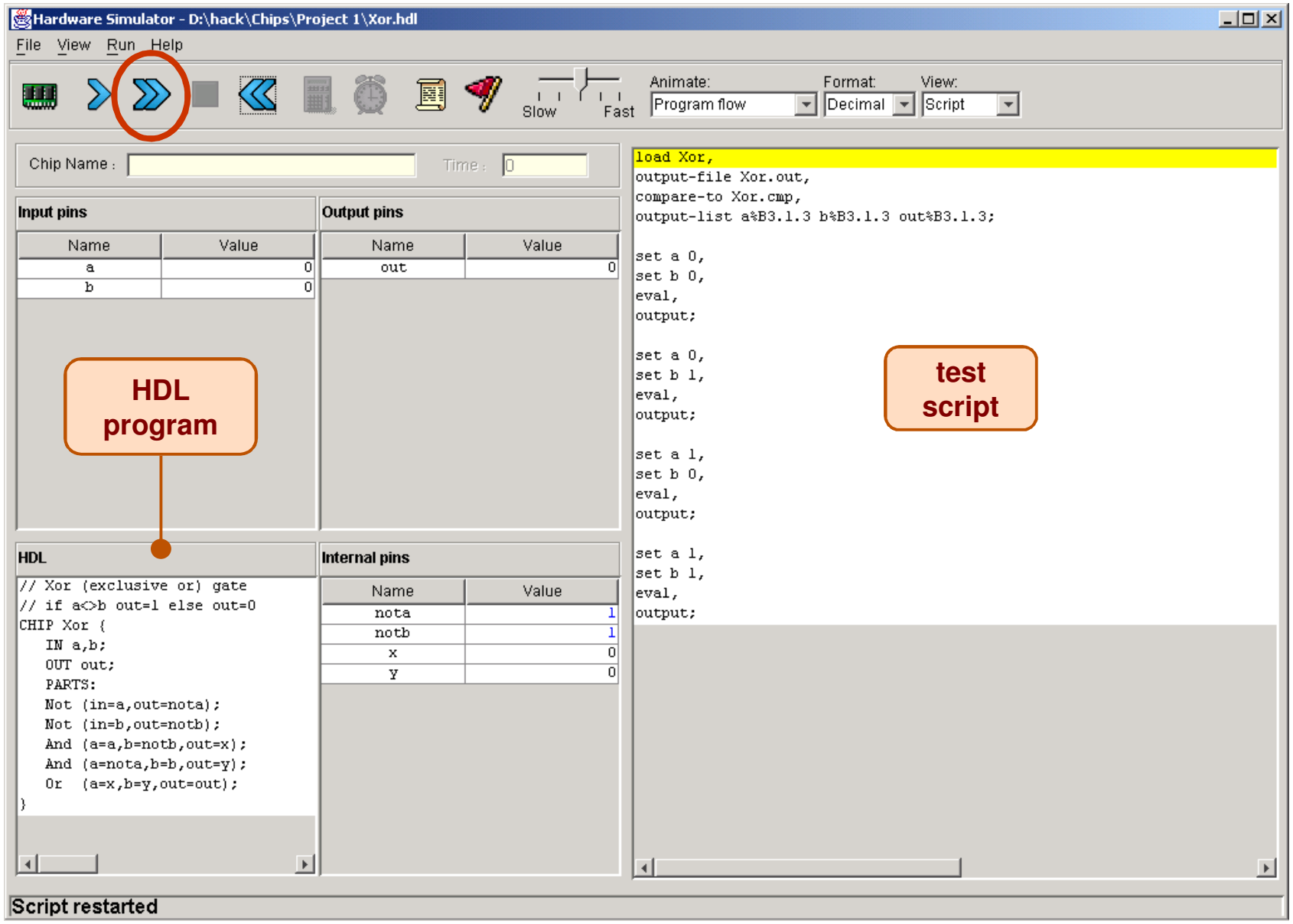
Implementation: And(a,b) = Not(Nand(a,b))



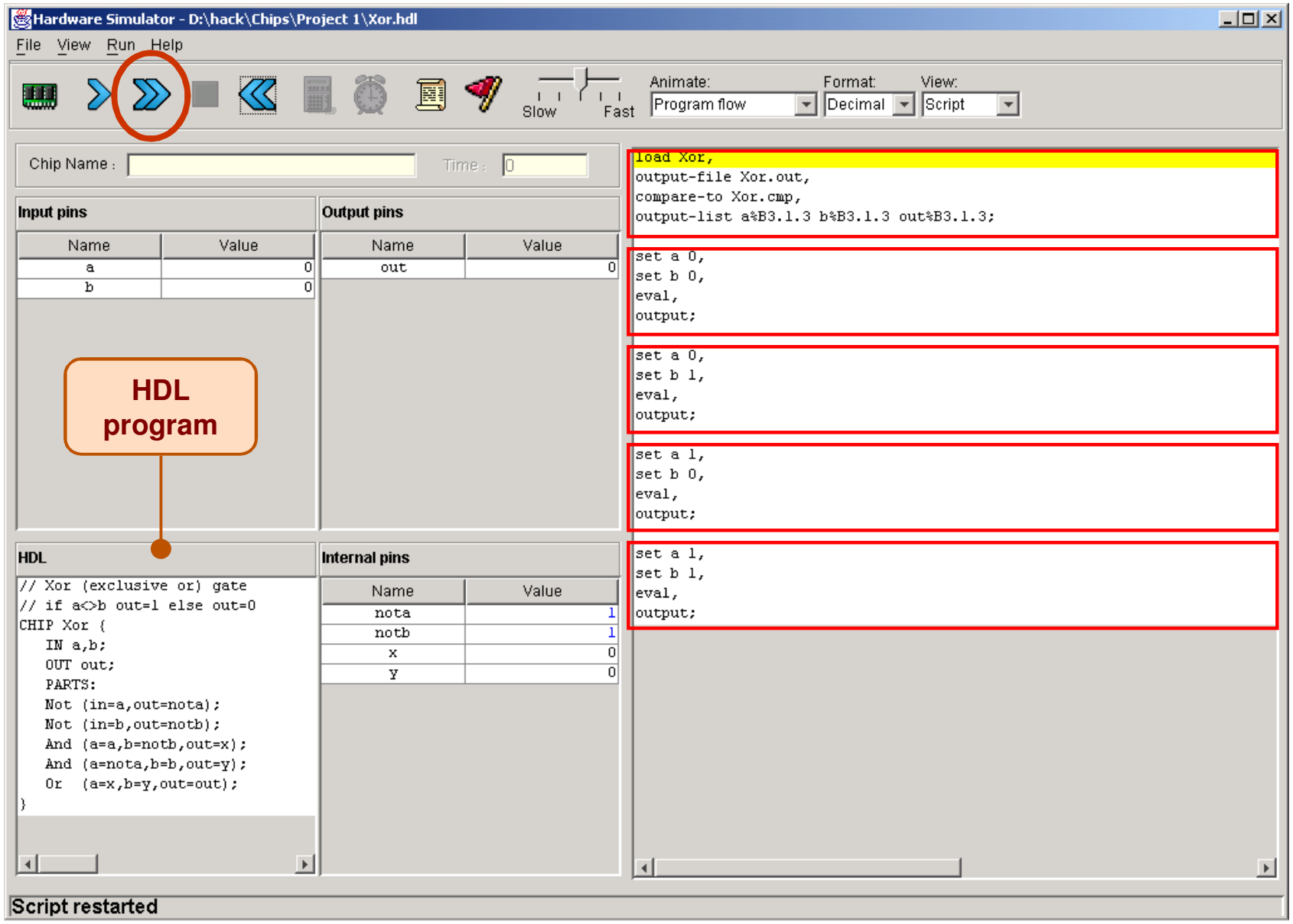
And.hdl

```
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



Hardware simulator