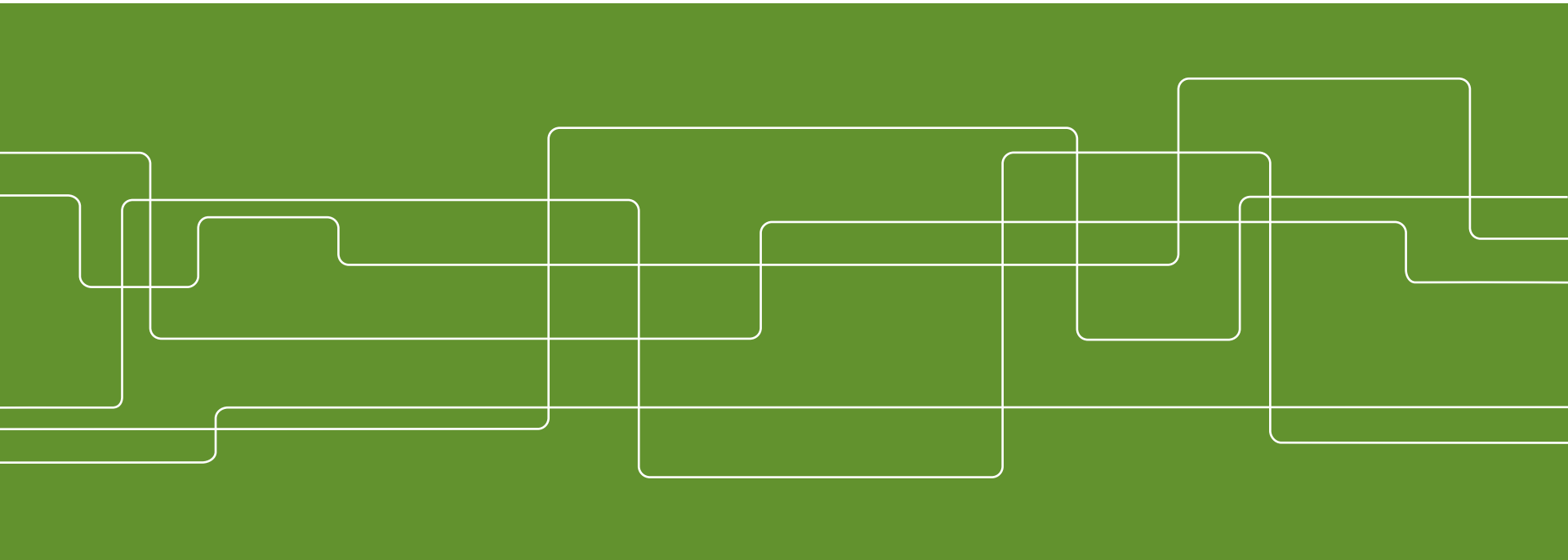
EP1200 Introduction to Computing Systems Engineering

Boolean Logic



Usage and Copyright Notice:

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

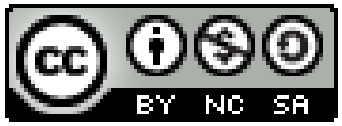
We provide 13 such presentations.

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 LOGIC

From NAND to any logic gate

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

Two forms of expressions

- truth table expression (input-output relations)
- functional expression
- can be expressed using And, Or, Not

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

All Boolean functions of two variables

Function	x	0	0	1	1
	y	0	1	0	1
Constant 0	0	0	0	0	0
And	$x \cdot y$	0	0	0	1
x And Not y	$x \cdot \overline{y}$	0	0	1	0
x	x	0	0	1	1
Not x And y	$\overline{x} \cdot y$	0	1	0	0
y	y	0	1	0	1
Xor	$x \cdot \overline{y} + \overline{x} \cdot y$	0	1	1	0
Or	$x + y$	0	1	1	1
Nor	$\overline{x + y}$	1	0	0	0
Equivalence	$x \cdot y + \overline{x} \cdot \overline{y}$	1	0	0	1
Not y	$\overline{y}$	1	0	1	0
If y then x	$x + \overline{y}$	1	0	1	1
Not x	$\overline{x}$	1	1	0	0
If x then y	$\overline{x} + y$	1	1	0	1
Nand	$\overline{x \cdot y}$	1	1	1	0
Constant 1	1	1	1	1	1

Boolean algebra

Given

`Nand(a,b), false`

We can build

- **`Not(a) = Nand(a,a)`**
- **`true = Not(false)`**
- **…**

George Boole, 1815-1864

(“*A Calculus of Logic*”)

Project 1: First elementary logic gates

Given

```
Nand(a,b), false
```

Build

```
Not(a) = ...

true = ...

And(a,b) = ...

Or(a,b) = ...

Mux(a,b,sel) = ...
```

a	b	Nand(a,b)
0	0	1
0	1	1
1	0	1
1	1	0

Ten chips to build altogether

- They provide all basic building blocks needed to build a computer

N.b. You may use all chips that you have built in the design of new chips!

HARDWARE SIMULATOR

Tutorial in Scalable Learning

BOOLEAN ARITHMETIC

Adders and ALU

Counting systems

quantity	decimal	binary	3-bit register
	0	0	000
*	1	1	001
**	2	10	010
***	3	11	011
****	4	100	100
*****	5	101	101
******	6	110	110
*******	7	111	111
********	8	1000	overflow
*********	9	1001	overflow
**********	10	1010	overflow

Representing negative numbers

2's complement

- The codes of all positive numbers begin with a "0"
- The codes of all negative numbers begin with a "1"

To convert a number

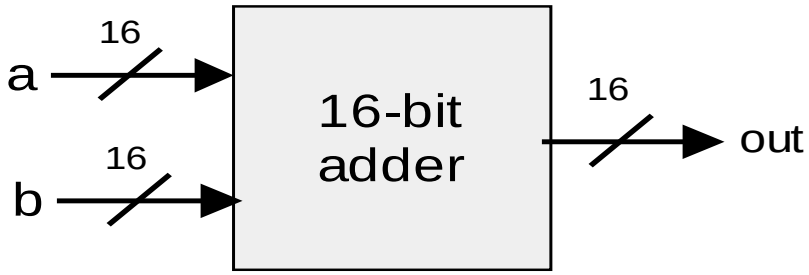
- leave all trailing 0's and first 1 intact
- flip all the remaining bits

Example: 2 – 5 = 2 + (–5)

```
  0 0 1 0
+ 1 0 1 1
---------
  1 1 0 1    =  -3
```

0	0000		
1	0001	1111	-1
2	0010	1110	-2
3	0011	1101	-3
4	0100	1100	-4
5	0101	1011	-5
6	0110	1010	-6
7	0111	1001	-7
		1000	-8

Building an Adder chip



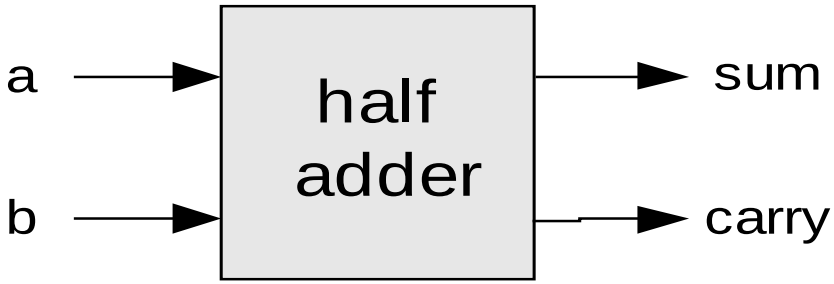
Adder: a chip designed to add two integers

Proposed implementation

- Half adder: adds 2 bits
- Full adder: adds 3 bits
- Adder: adds two *n*-bit numbers

Half adder

a	b	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

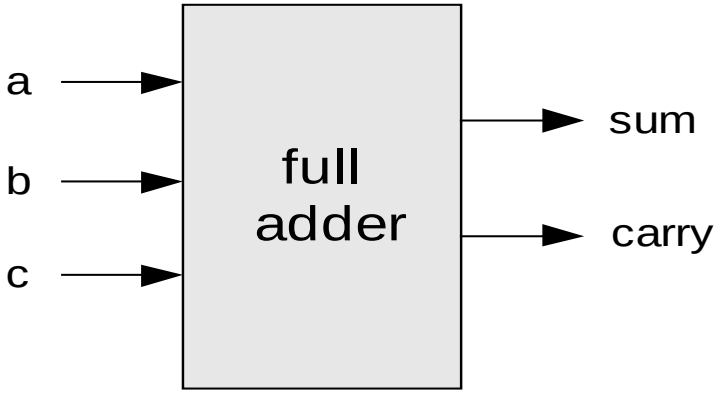


Implementation:

based on two gates that you've seen before.

Full adder

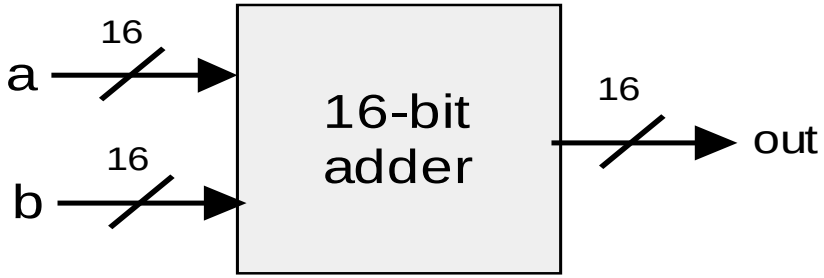
a	b	c	sum	carry
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



Implementation:

based on half-adder gates

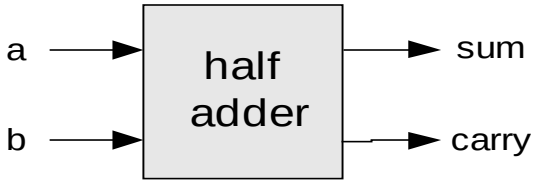
n-bit Adder

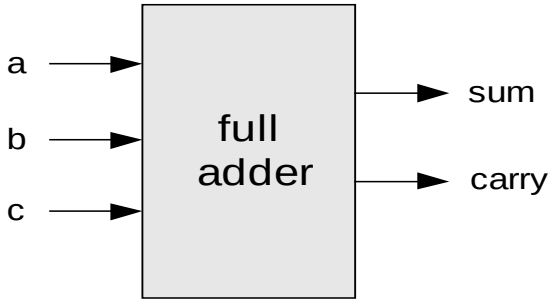


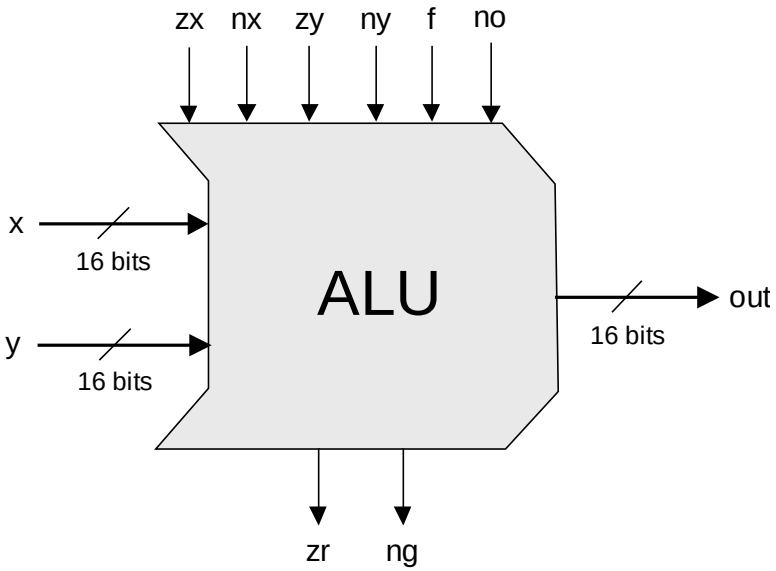
Implementation: array of full-adder gates

...	1	0	1	1	a
					+
…	0	0	1	0	b
…	1	1	0	1	out

The ALU of the Hack platform







out(x, y, control bits**) =**

x+y, x-y, y–x,

0, 1, -1,

x, y, -x, -y,

x!, y!,

x+1, y+1, x-1, y-1,

x&y, x|y

ALU logic of the Hack platform

These bits instruct how to pre-set the x input		These bits instruct how to pre-set the y input		This bit selects between + / And	This bit inst. how to post-set out	Resulting ALU output
zx	nx	zy	ny	f	no	out=
if zx then x=0	if nx then x=!x	if zy then y=0	if ny then y=!y	if f then out=x+y else out=x And y	if no then out=!out	f(X,y)=
1	0	1	0	1	0	0
1	1	1	1	1	1	1
1	1	1	0	1	0	-1
0	1	0	0	1	1	x-y
0	0	0	1	1	1	y-x
0	0	0	0	0	0	x&y
0	1	0	1	0	1	x|y

Implementation: build a logic gate architecture that "executes" the control bit "instructions"

- if zx==1 then set x to 0 (bit-wise),
- et cetera

For next class

1. View the video *Getting started with logic design*
2. Download the book's software suite and project templates
3. Read the book and a guide
   - Introduction, Chapter 1 and 2; Appendix A up to page 288
   - Hardware Survival Guide by Mark Armbrust
     - Link under *Resources* on the course web
     - Tip: read about sub-bussing
4. Project descriptions on course web in KTH Social
   - Much work: start working immediately!

For next class

5. Do Project 1 on Boolean logic and arithmetic
   - Chips
     - Not, And, Or, Mux, Dmux, And16, Or8Way, Mux16, Mux4Way16, DMux4Way
       - Project folder: projects/01
     - HalfAdder, FullAdder, Add16, Inc16, ALU
       - Project folder: projects/02
   - Note
     - Office hours
       - Thursday March 23 at 10 to 12 in Q15

Submission instructions

- You submit your own solutions
  - Copying and other forms of plagiarism and cheating will be reported for disciplinary action!
- You declare what chips you have done
  - May be asked to present those parts
  - Zero points if a declared part cannot be presented
- The solutions and the declaration are put in a zip-folder
- The folder is mailed to the group leader before the deadline
  - Group 1: Gunnar Karlsson (gk@kth.se)
  - Group 2: György Dán (gyuri@kth.se)
  - Group 3: Viktoria Fodor (vjfodor@kth.se)
  - Subject line: *EP1200*
- **Email Project 1 by March 24 at 8.00**