IxD Theory 2: Telecomunicazioni

IUAV University of Venice

Visual and Multimedia Communication graduate programme

Computation

Why do we need to know?

It’s our ‘medium’ as designers

It’s shaping our world

It’s amazing

Key concepts

1 Digital

2 Code

DIGITAL

To do with discrete amounts (from digit, digitus—counting on fingers—discrete counting).

Contrast with analog (a continuous spectrum).

Computers work by controlling the **flow** of electricity.

Computers use electrical voltages to represent binary numbers (base 2 numbers)

no voltage = 0 voltage = 1

Binary numbers

decimal	*binary*
1 =	1
2 =	1 0
3 =	1 1
4 =	1 0 0
5 =	1 0 1
6 =	1 1 0
7 =	1 1 1
8 =	1 0 0 0
9 =	1 0 0 1
10 =	1 0 1 0
? =	1 1 1 1

Binary numbers

decimal	*binary*	*4-bit code*
1 =	1	
2 =	1 0	
3 =	1 1	
4 =	1 0 0	
5 =	1 0 1	
6 =	1 1 0	
7 =	1 1 1	
8 =	1 0 0 0	
9 =	1 0 0 1	
10 =	1 0 1 0	
? =	1 1 1 1	

Binary numbers

decimal	*binary*	*4-bit code*
1 =	1	0001
2 =	1 0	0010
3 =	1 1	0011
4 =	1 0 0	0100
5 =	1 0 1	0101
6 =	1 1 0	0110
7 =	1 1 1	0111
8 =	1 0 0 0	1000
9 =	1 0 0 1	1001
10 =	1 0 1 0	1010
? =	1 1 1 1	1111

Binary numbers

1 bit of information is the smallest possible unit of information. It has two states: on or off.

Computers use 1 bit of information to represent each binary digit (measuring a voltage on or off).

Binary numbers

1 bit of information is the smallest possible unit of information. It has two states: on or off.

Computers use 1 bit of information to represent each binary digit (measuring a voltage on or off).

A **byte** is 8 bits.

It can represent numbers from 0 to 255.

Binary numbers

decimal		*binary*	*8-bit code*
15	=	1 1 1 1	0000 1111
255	=		1111 1111

Binary numbers: a byte

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	**1**	**1**	**1**	**1**	**1**	**1**	**1**

128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255

Binary numbers: a byte

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
128	64	32	16	8	4	2	1	
1	**0**	**1**	**0**	**1**	**0**	**1**	**0**	= ?

Binary numbers: a byte

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1
1	**0**	**1**	**0**	**1**	**0**	**1**	**0**

$128 + 0 + 32 + 0 + 8 + 0 + 2 + 0 = 170$

Binary numbers: a byte

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
128	64	32	16	8	4	2	1	
0	**0**	**0**	**0**	**0**	**1**	**1**	**1**	= ?

Binary numbers: a byte

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
128	64	32	16	8	4	2	1	
0	**0**	**0**	**0**	**0**	**1**	**1**	**1**	
0	0	0	0	0	4	+ 2 + 1		= 7

Adding decimal numbers

1 3

1 9

———

2 carry 1

Adding decimal numbers

1 3

1 9

———

2 carry **1**

Adding decimal numbers

1 3

1 9

---

2 2

1

3 2

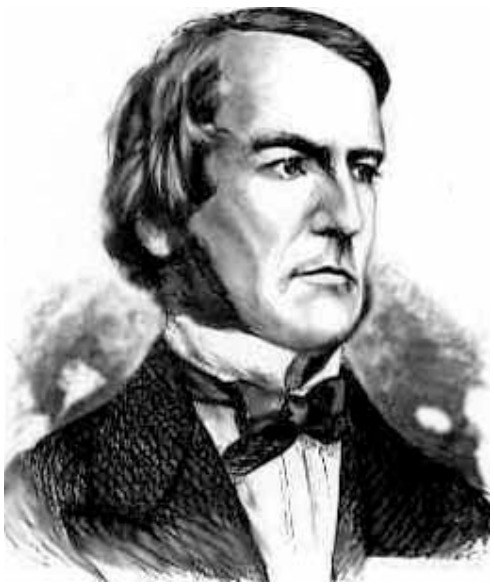
Adding binary numbers

A		*B*		*carry*	
0	+	0 = 0		0	
0	+	1 = 1		0	
1	+	0 = 1		0	
1	+	1 = 0		1	= 10 (binary)

Boolean logic

George Boole (1815–64)

His aim: to make an algebra of logical thought



Boolean logic

Truth table for **AND**

AND	**F**	**T**
F	F	F
T	F	T

B

A

Take two inputs: proposition A **AND** proposition B.

The table shows the different permutations according to the truth of A and B.

Boolean logic

Is this true?:

(A) Phil is a woman **AND** (B) Phil is bald?

Boolean logic

Is it true that both

(A) Phil is a woman **AND** (B) Phil is bald?

False **AND** False = ?

Boolean logic

Is this true?:

(A) Phil is a woman **AND** (B) Phil is bald?

False **AND** False = False

Boolean logic

Is it true that both

AND	F	T
F	F	F
T	F	T

B

A

(A) Phil is a woman **AND** (B) Phil is bald ?

False **AND** False = False

Boolean logic

Is it true that both

AND	F	T
F	F	F
T	F	T

B

A

(A) Phil is a woman **AND** (B) Phil is bald ?

False **AND** False = False

Boolean logic

Is this true?:

(A) Phil is a man **AND** (B) Phil has hair?

Boolean logic

Is this true?:

(A) Phil is a man **AND** (B) Phil has hair?

True **AND** True = ?

Boolean logic

Is this true?:

(A) Phil is a man **AND** (B) Phil has hair?

True **AND** True = True

Boolean logic

Is this true?:

AND	F	T
F	F	F
T	F	T

B

A

(A) Phil is a man **AND** (B) Phil has hair?

True **AND** True = True

Boolean logic

(A) Phil is a woman **AND** (B) Phil is bald

False **AND** False = False

(A) Phil is a woman **AND** (B) Phil has hair

False **AND** True = False

(A) Phil is a man **AND** (B) Phil is bald

True **AND** False = False

(A) Phil is a man **AND** (B) Phil has hair

True **AND** True = True

Boolean logic

AND	F	T
F	F	F
T	F	T

B

A

(A) Phil is a woman **AND** (B) Phil is bald

False **AND** False = False

(A) Phil is a woman **AND** (B) Phil has hair

False **AND** True = False

(A) Phil is a man **AND** (B) Phil is bald

True **AND** False = False

(A) Phil is a man **AND** (B) Phil has hair

True **AND** True = True

Boolean logic: OR

Is it true that

AND	F	T
F	F	F
T	F	T

B

A

(A) Phil is a man **AND** (B) Phil is bald

True **AND** False = False

OR	F	T
F	F	T
T	T	T

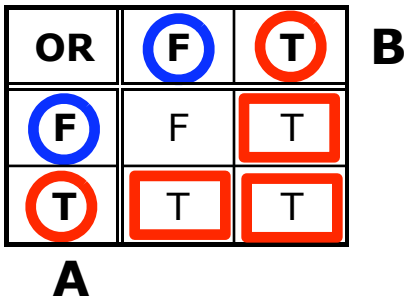
B

A

(A) Phil is a man **OR** (B) Phil is bald

True **OR** False = True

Boolean logic: OR



(A) Phil is a woman **OR** (B) Phil is bald

False **OR** False = False

(A) Phil is a woman **OR** (B) Phil has hair

False **OR** True = True

(A) Phil is a man **OR** (B) Phil is bald

True **OR** False = True

(A) Phil is a man **OR** (B) Phil has hair

True **OR** True = True

Boolean logic: OR is inverse of AND

(A) Phil is a woman **OR** (B) Phil is bald = False

(A) Phil is a man **OR** (B) Phil has hair = True

(A) Phil is a woman **OR** (B) Phil has hair = True

(A) Phil is a man **OR** (B) Phil is bald = True

(A) Phil is a woman **AND** (B) Phil is bald = False

(A) Phil is a man **AND** (B) Phil has hair = True

(A) Phil is a woman **AND** (B) Phil has hair = False

(A) Phil is a man **AND** (B) Phil is bald = False

Boolean logic

Truth tables

AND	F	T
F	F	F
T	F	T

OR	F	T
F	F	T
T	T	T

XOR	F	T
F	F	T
T	T	F

NOT	
F	T
T	F

Boolean logic

Truth tables

AND	F	T
F	F	F
T	F	T

OR	F	T
F	F	T
T	T	T

XOR	F	T
F	F	T
T	T	F

NOT	
F	T
T	F

XOR: Is (A) true or (B) true, but not both?

(A) Phil is a man **XOR** (B) Phil has hair

True **XOR** True = False

Boolean logic

What has Boolean logic to do with computers?

Boolean logic

Truth tables: false = 0, true = 1

AND	F	T
F	F	F
T	F	T

OR	F	T
F	F	T
T	T	T

XOR	F	T
F	F	T
T	T	F

NOT	
F	T
T	F

Boolean logic

Truth tables: false = 0, true = 1

AND	F	T
F	F	F
T	F	T

OR	F	T
F	F	T
T	T	T

XOR	F	T
F	F	T
T	T	F

NOT	
F	T
T	F

AND	0	1
0	0	0
1	0	1

OR	0	1
0	0	1
1	1	1

XOR	0	1
0	0	1
1	1	0

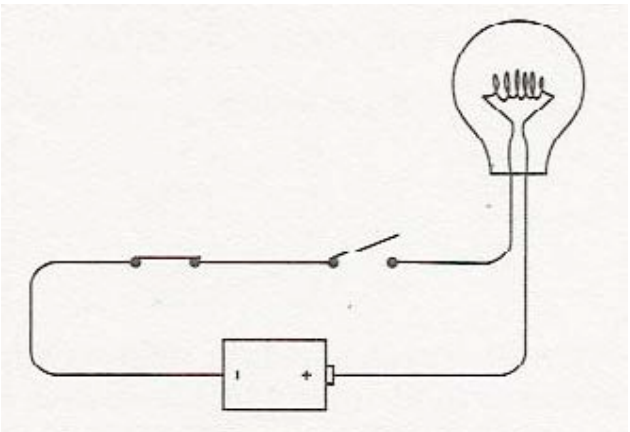
NOT	
0	1
1	0

Boolean logic

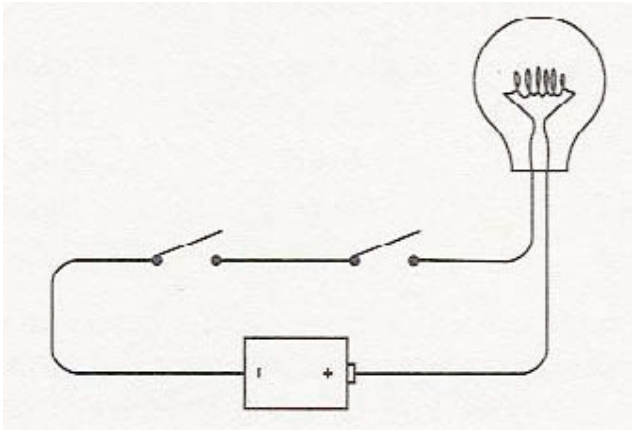
The next step is to do this physically, with electricity.

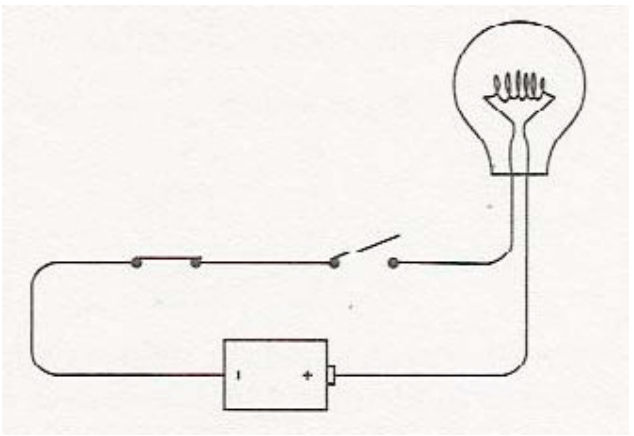
Boolean logic with electricity

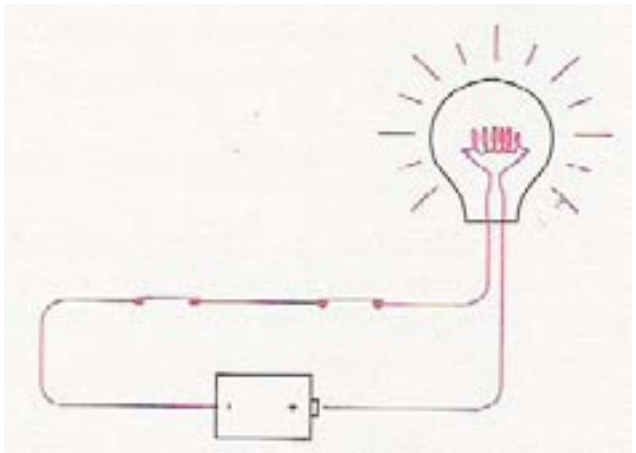
The principle is this (in a computer there are microscopic switches in the silicon of a chip):



Boolean logic with electricity







AND 0 0 = **0**
1 0 = **0**
0 1 = **0**
1 1 = **1**

(from Charles Petzold: *Codice: il linguaggio segreto dei computer*)

Adding binary numbers—remember?

A		*B*		*carry*	
0	+	0	= 0	0	
0	+	1	= 1	0	
1	+	0	= 1	0	
1	+	1	= 0	1	= 10 (binary)

Truth tables meet binary arithmetic

A	B	sum	carry
0 + 0 = 0		**0**	**0**
0 + 1 = 1		**1**	**0**
0 + 1 = 1		**1**	**0**
1 + 1= 10		**0**	**1**

Truth tables meet binary arithmetic

A	B	sum	carry
0	0 = 0	**0**	**0**
0	1 = 1	**1**	**0**
0	1 = 1	**1**	**0**
1	1= 10	**0**	**1**

XOR

0 0 = **0**
1 0 = **1**
0 1 = **1**
1 1 = **0**

AND

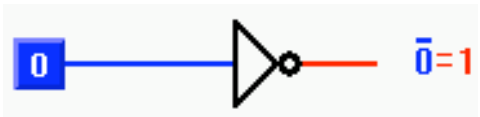
0 0 = **0**
1 0 = **0**
0 1 = **0**
1 1 = **1**

Electronic notation

AND gate

OR gate

NOT (inverter)



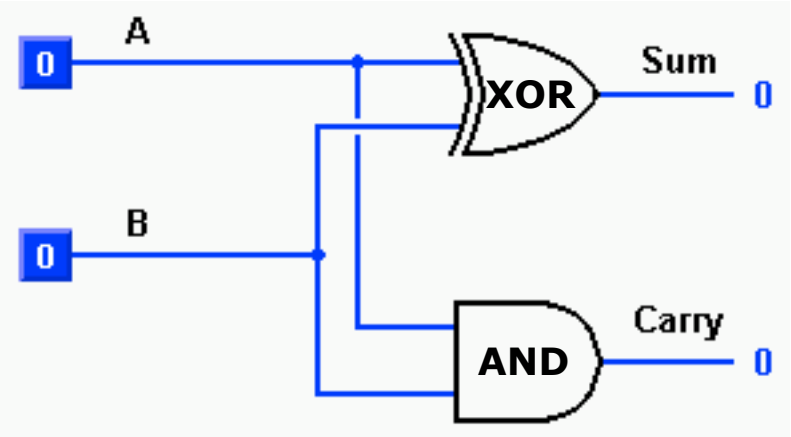
XOR gate

Binary logic: adder

XOR gate

+

AND gate



A *B* *carry*

0 + 0 = 0 **0**

Boolean logic

Truth tables: false = 0, true = 1

AND	F	T
F	F	F
T	F	T

XOR	F	T
F	F	T
T	T	F

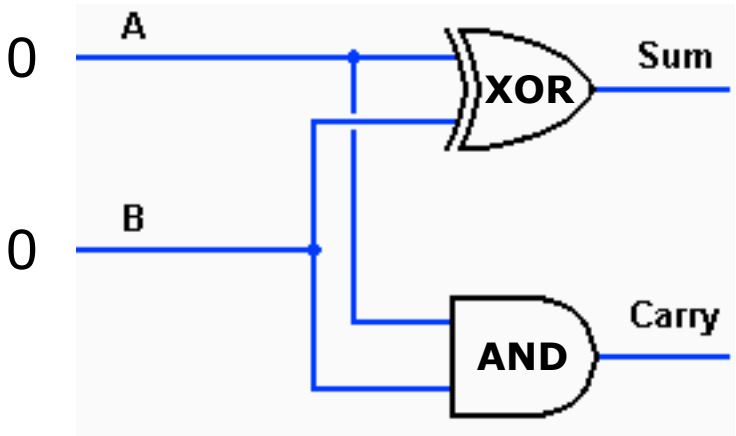
AND	0	1
0	0	0
1	0	1

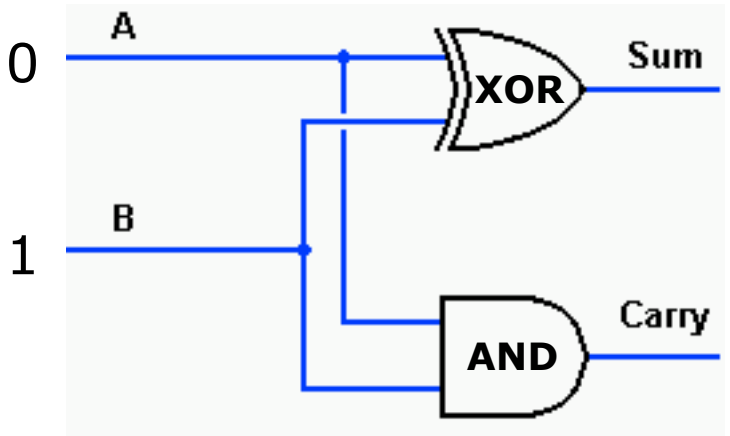
XOR	0	1
0	0	1
1	1	0

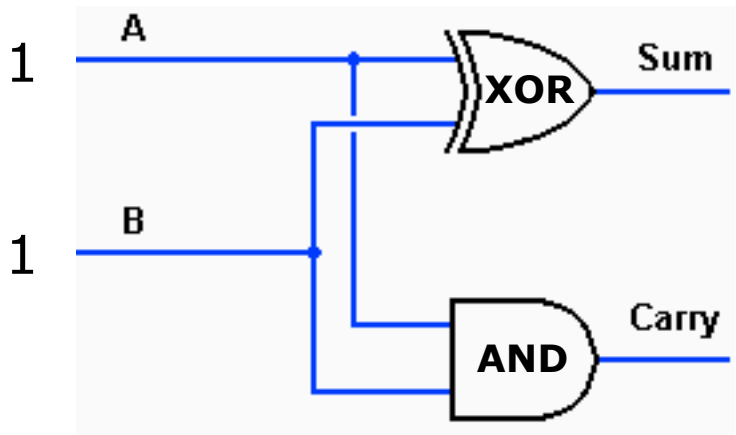
EXERCISE!

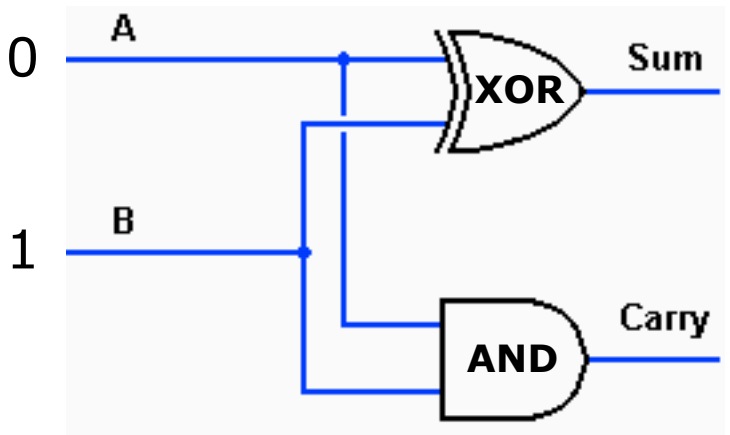
Write down the AND and the XOR truth tables

Binary logic: adder









Adding a byte

When we add two bytes we need to be able to carry over a digit to the next column:

128	64	32	16	8	4	2	1	
0	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**+**
1	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**=**
1	**0**	**1**	**1**	**0**	**1**	**0**	**0**	
			1			**1**	**1**	

Adding a byte

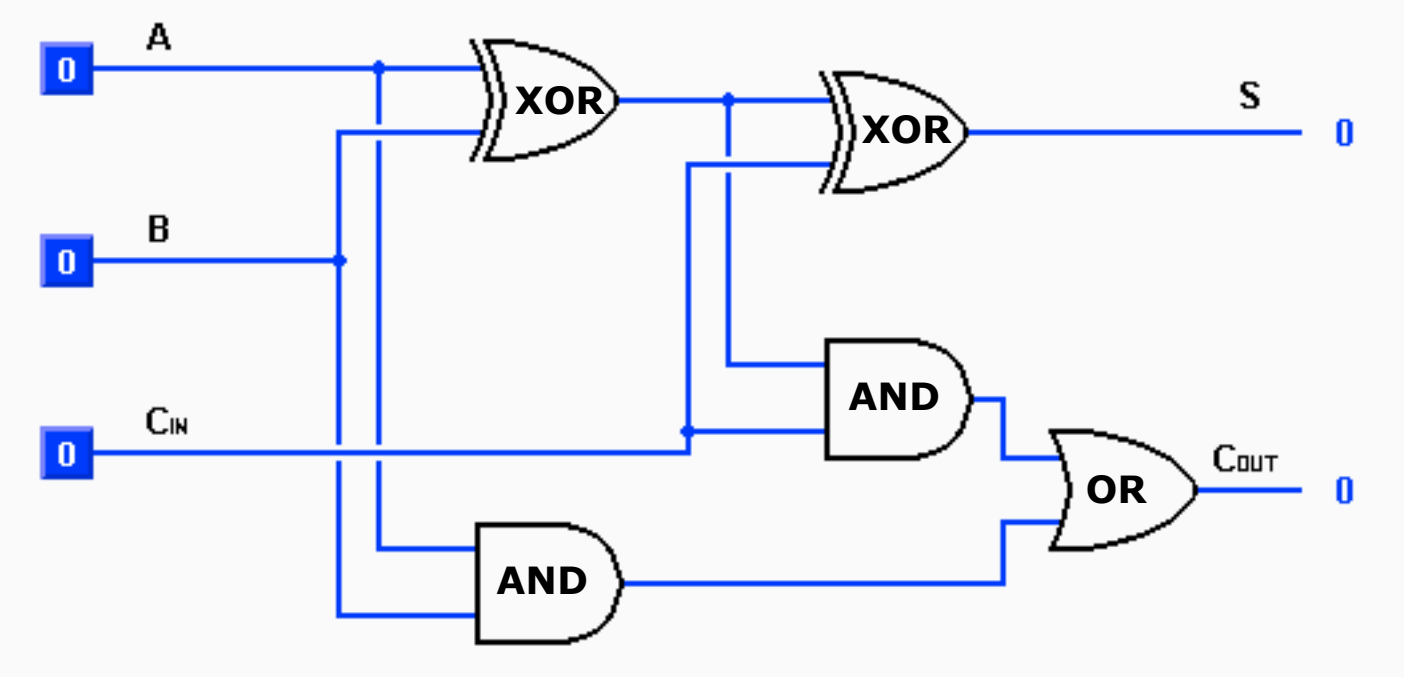
When we add a byte we need to be able to carry over a digit to the next column:

128	64	32	16	8	4	2	1		
0	**0**	**0**	**0**	**1**	**0**	**1**	**1**	**+**	**A**
1	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**=**	**B**
1	**0**	**1**	**1**	**0**	**1**	**0**	**0**		
			1			**1**	**1**		**C**

In order to add numbers more than 0 or 1 we need a circuit that will take in three inputs: (A) (B) and (C), the carry from the previous column.

Electronic notation: full adder

Three inputs: number (A), number (B) and a carry from the previous addition



http://www.play-hookey.com/digital/basic_gates.html

DIGITAL: summary

- Computers use binary numbers
- To add numbers they use electronic circuits to represent Boolean logic
- More complicated circuits allow addition of 8-, 32- and 64-bit numbers as well as multiplication, subtraction and division
- These tiny elements are combined into the very complex systems we use today