Information Technology for Digital Humanities

Lecture 7

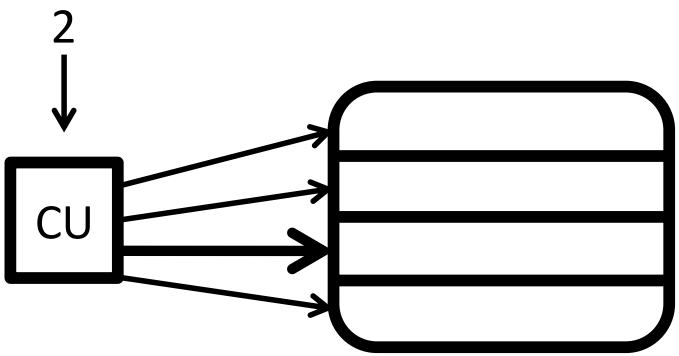
Mario Verdicchio

Università degli Studi di Bergamo

Academic Year 2024-2025

Lecture 7 (October 15 2024)

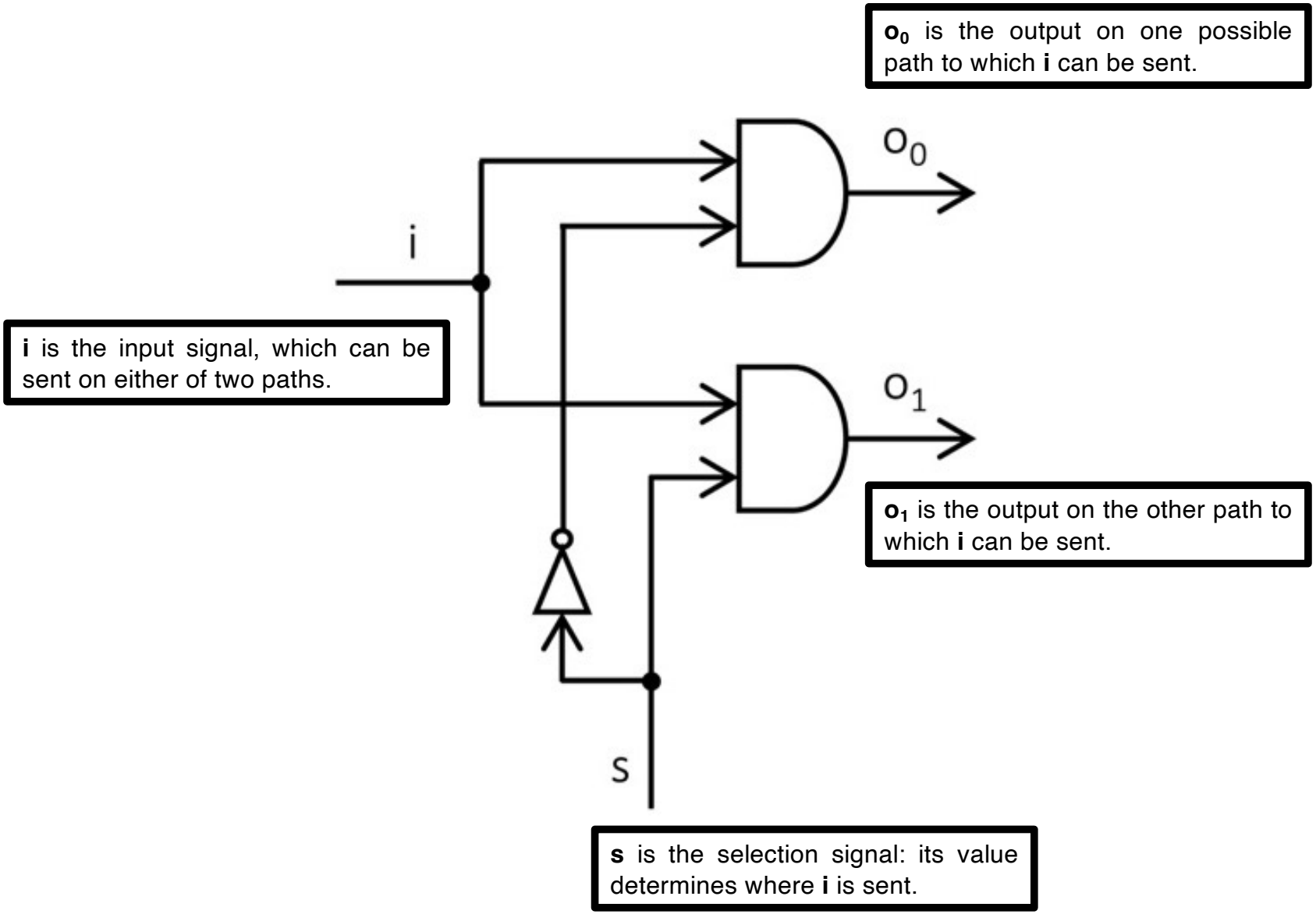
- More gates
- Automated Reasoning



Let’s now see how these gates can be combined to operate the selection of a specific path.

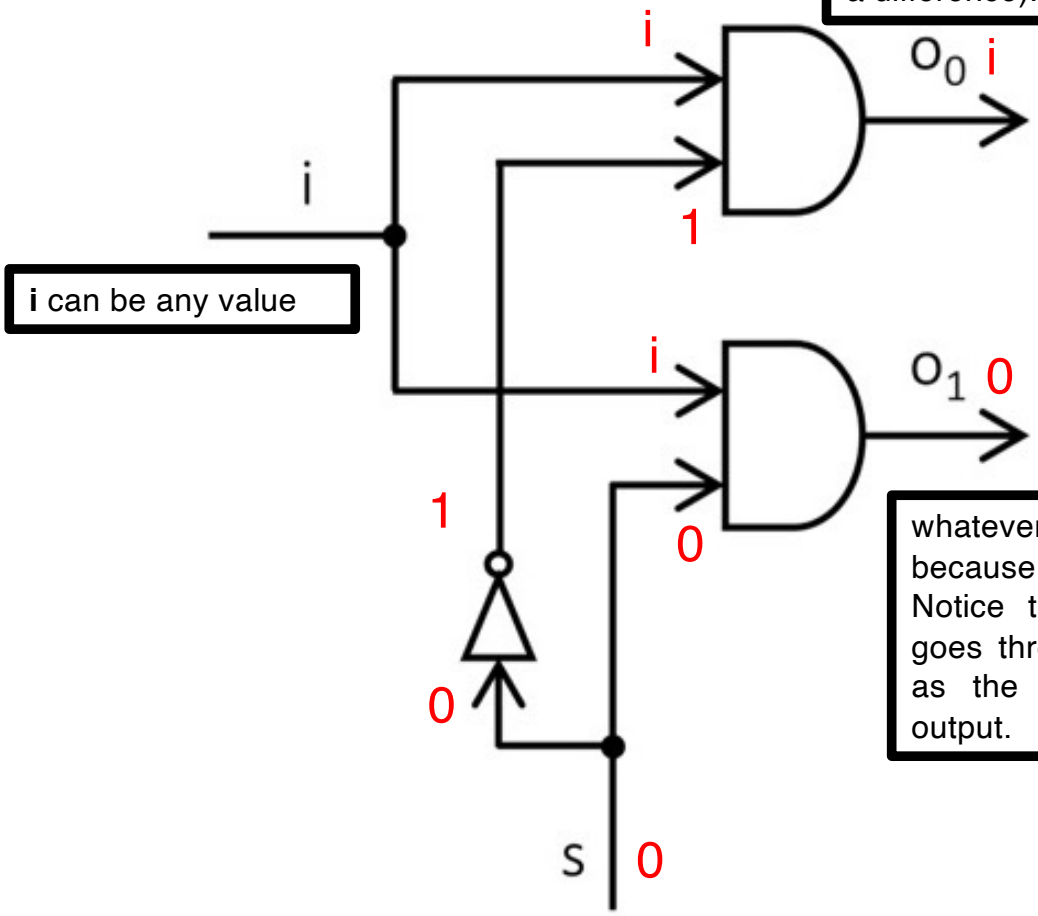
Let’s focus on a simpler case than the one depicted in the figure, where the CU chooses among 4 different paths.

In what follows, we see how to build a circuit to choose between 2 paths.


o_0 is the output on one possible path to which **i** can be sent.
O_0
i
i is the input signal, which can be sent on either of two paths.
O_1
o_1 is the output on the other path to which **i** can be sent.
S
s is the selection signal: its value determines where **i** is sent.

if s is 0…

o_0 is the same as **i** because the other input of the AND gate is 1. Notice that **i** AND 1 is the same as **i** (whether **i** is 0 or 1 does not make a difference).



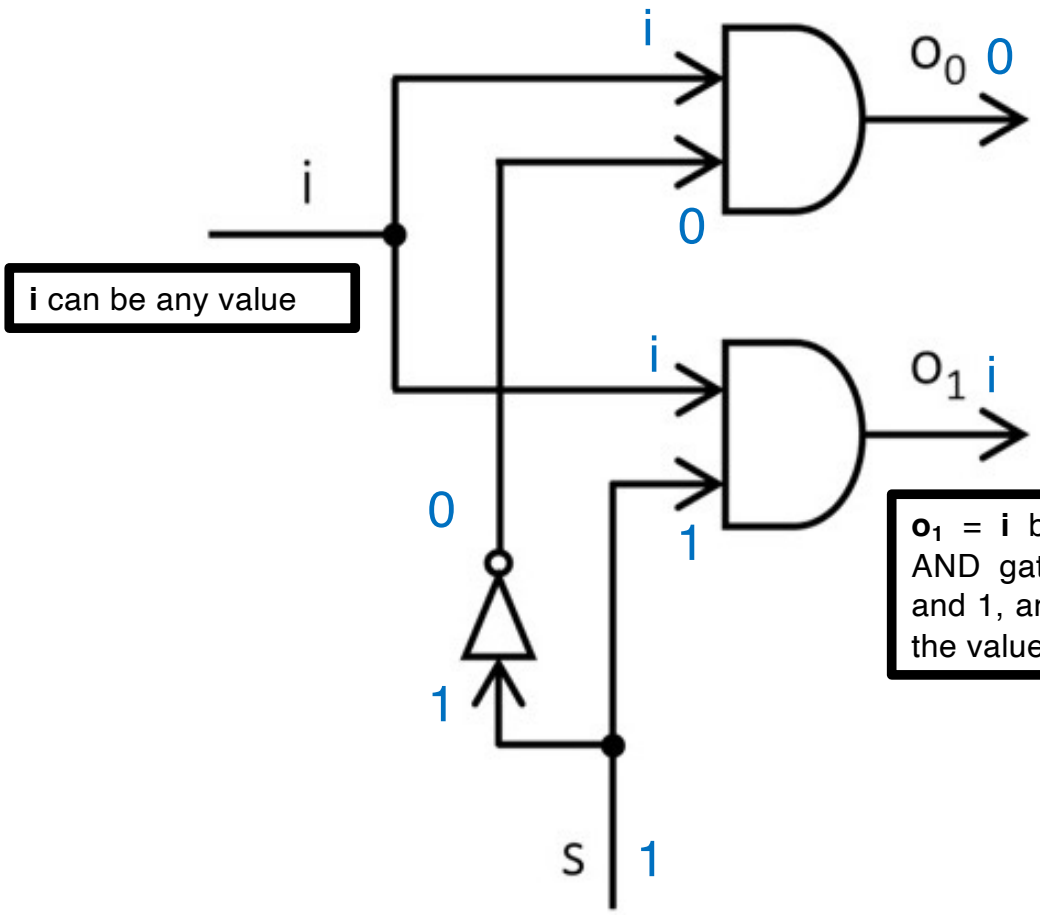
whatever the value of **i**, **o_1** is 0 because one of its inputs (**s**) is 0. Notice that any input value that goes through an AND gate with 0 as the other input yields 0 as output.

…then o_0 = i.

In other words, i is sent to the first path.

if s is 1…

o_0 is 0 no matter the value of **i**, because the other input of the AND gate is 0.

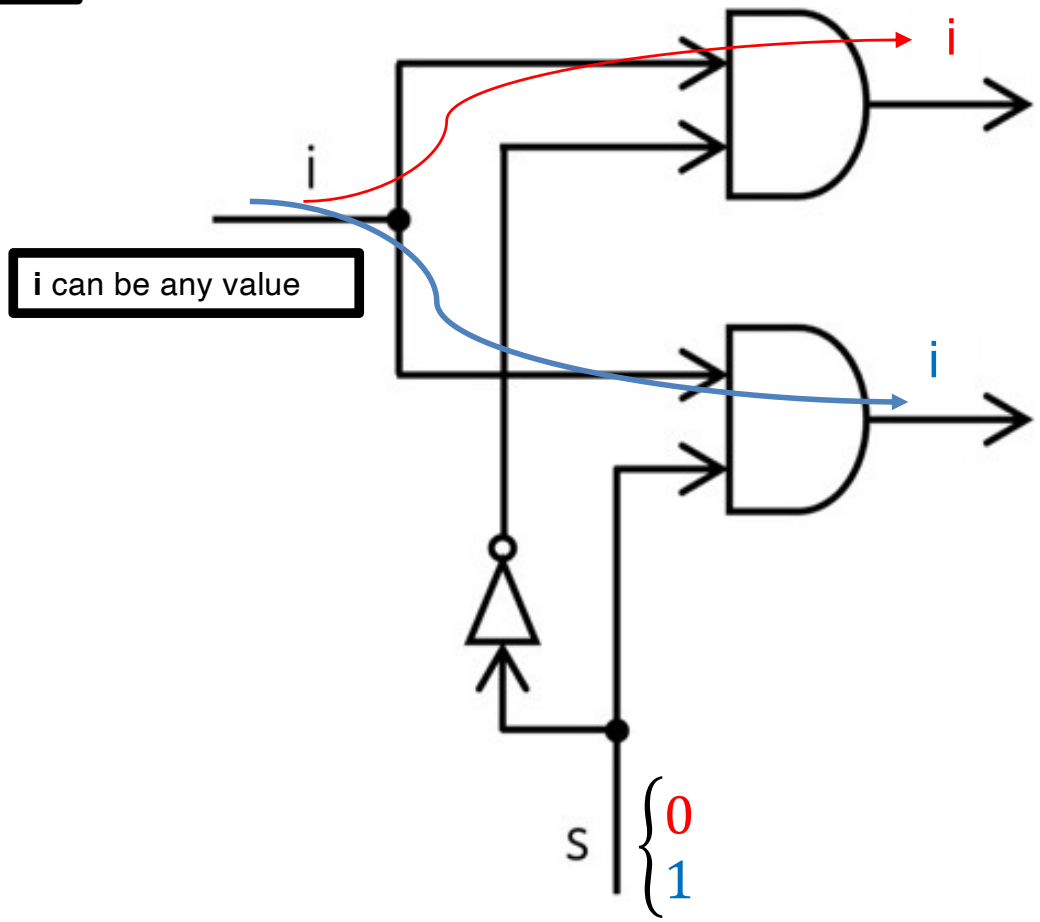


o_1 = **i** because the inputs of the AND gate that produces **o_1** are **i** and 1, and (**i** AND 1) = 1, whatever the value of i.

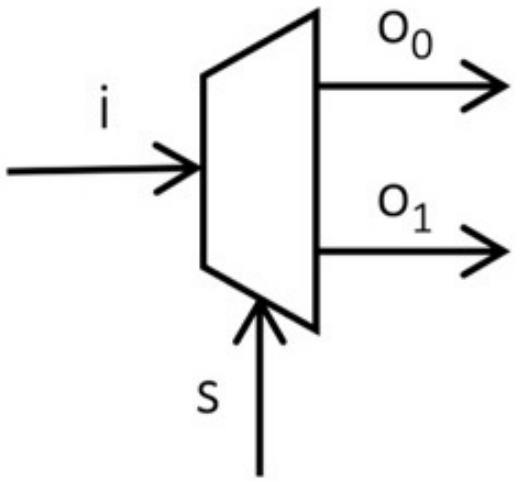
…then o_1 = i.

In other words, i is sent to the second path.

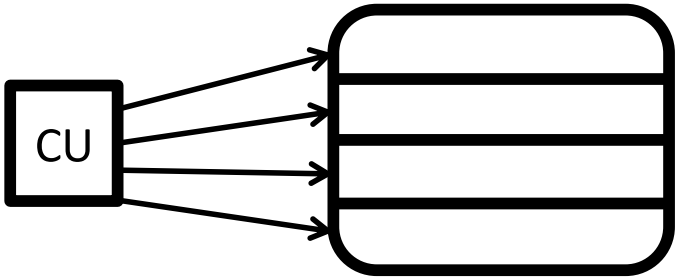
This means…



…that it is as if **s** selects where **i** goes.



This is the circuit design style for representing such a system, which is called “demultiplexer” (also known as “DEMUX”).
Since the choice is between 2 paths, this one is called a 1:2 DEMUX. In a 1:2 DEMUX, the selection signal is just 1 bit.



In our original example, the choice is between 4 paths, so we actually need a 1:4 DEMUX, with 2 bits for the selection signal (s_0 and s_1, enabling the selection among 4 paths: o_{00}, o_{01}, o_{10}, and o_{11}). The electronics is a bit more complicated, but the principles are the same as the ones guiding the design of the 1:2 DEMUX.

The story so far

- With all these considerations around logic, and the construction of electronic circuits that embody logical operators like NOT, AND, and OR, we have focused on the L in ALU
- Moreover, we have combined NOT and AND gates to create a DEMUX device, to enable the selection of a path among many, by which the CU can send signals where they are needed
- But what about the A in ALU?
- Aren’t computer built to compute, that is, do arithmetics after all?

Back to basics of arithmetic

- Numbers work as input for arithmetic operations no matter the numerical system they are expressed in
- We can do 3 + 4 and obtain 7 (base 10)
- In the same way, we can do 011 + 100 and obtain 111 (base 2)

Simple arithmetics in binary

- Given two bits in input, the rules for adding them are very simple:

  0 + 0 = 0

  0 + 1 = 1

  1 + 0 = 1

  1 + 1 = 10, that is 0 with carry 1

- Since + applies to 2 bits and yields 1 bit in output (possibly with a carry) just like AND and OR, we can imagine to build a system with logical gates that manipulate bits in a way that coincides with what + does

Addition in electronics

- Let's ignore the carry in the last case for now, and focus on what happens with the bits:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0

The output of + is almost ideantical to the output of OR

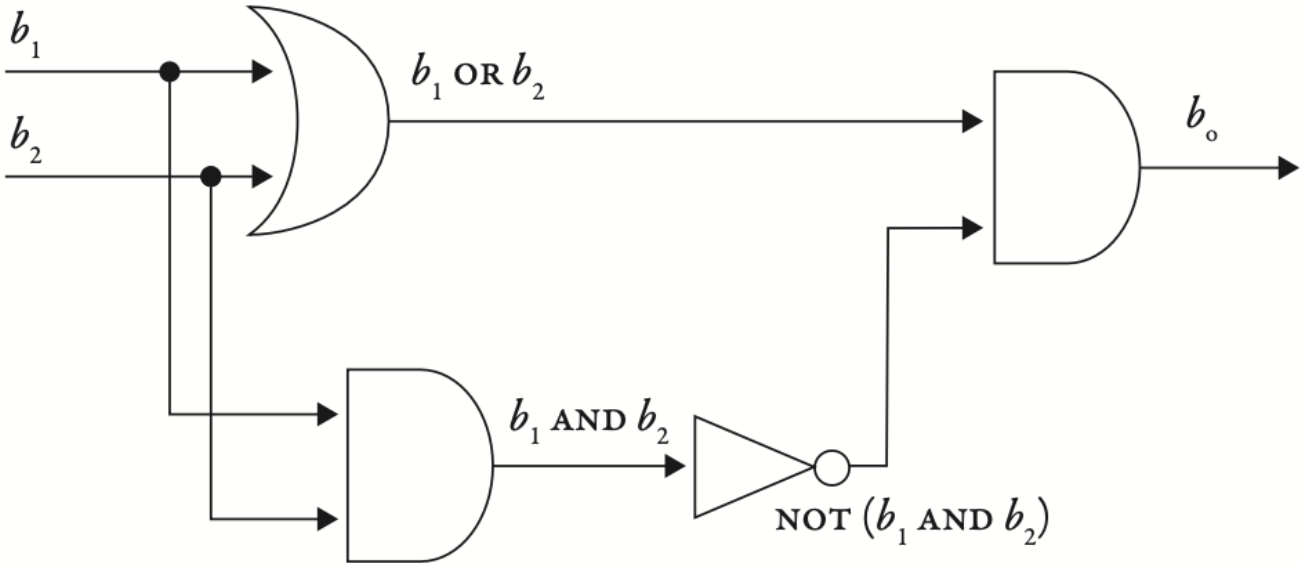
$input_1$	$input_2$	output
0	0	0
0	1	1
1	0	1
1	1	1

Except for the last case, where both input bits are 1 and the output is 0 instead of 1.

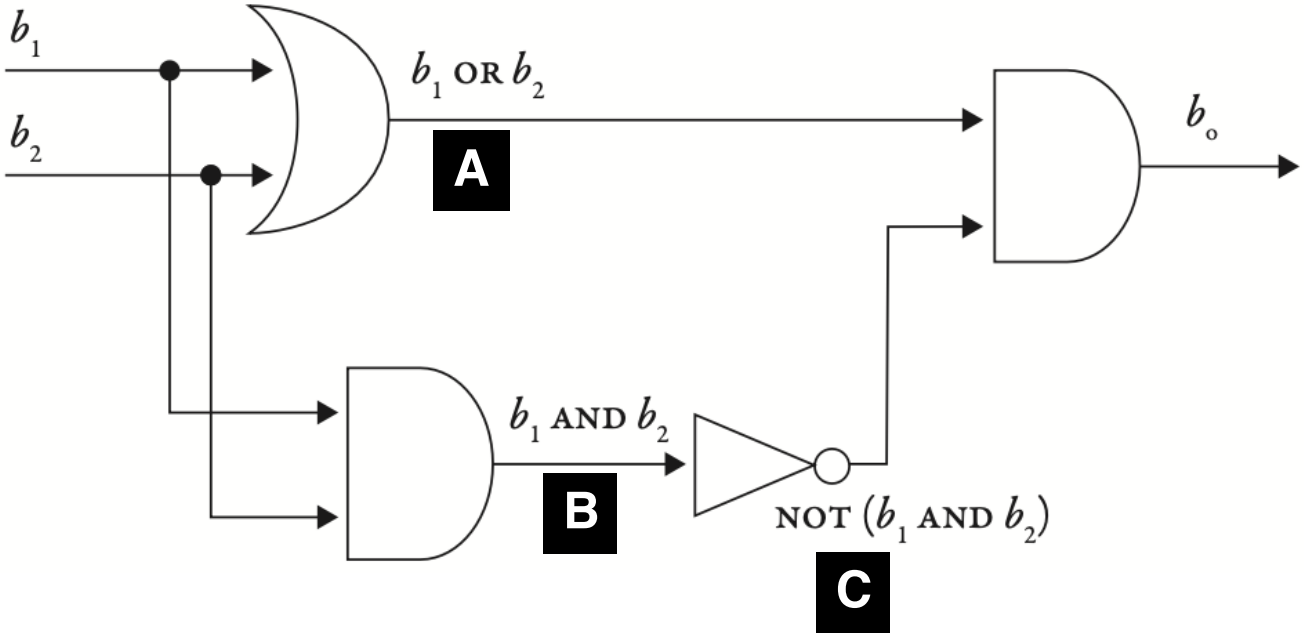
So, given two bits in input, b_1 and b_2, $b_1 + b_2$ is like b_1 OR b_2 but not if they are both 1.

Given two bits in input, b_1 and b_2, b_1 + b_2 is like b_1 OR b_2 but not if they are both 1.

To design a system that electronically realizes this behavior, we can combine the outputs of:
b_1 OR b_2
AND
NOT (b_1 AND b_2),
as shown in the system in the figure (taken from "Che cos'è un computer" by Mario Verdicchio, published by Carocci, 2023).



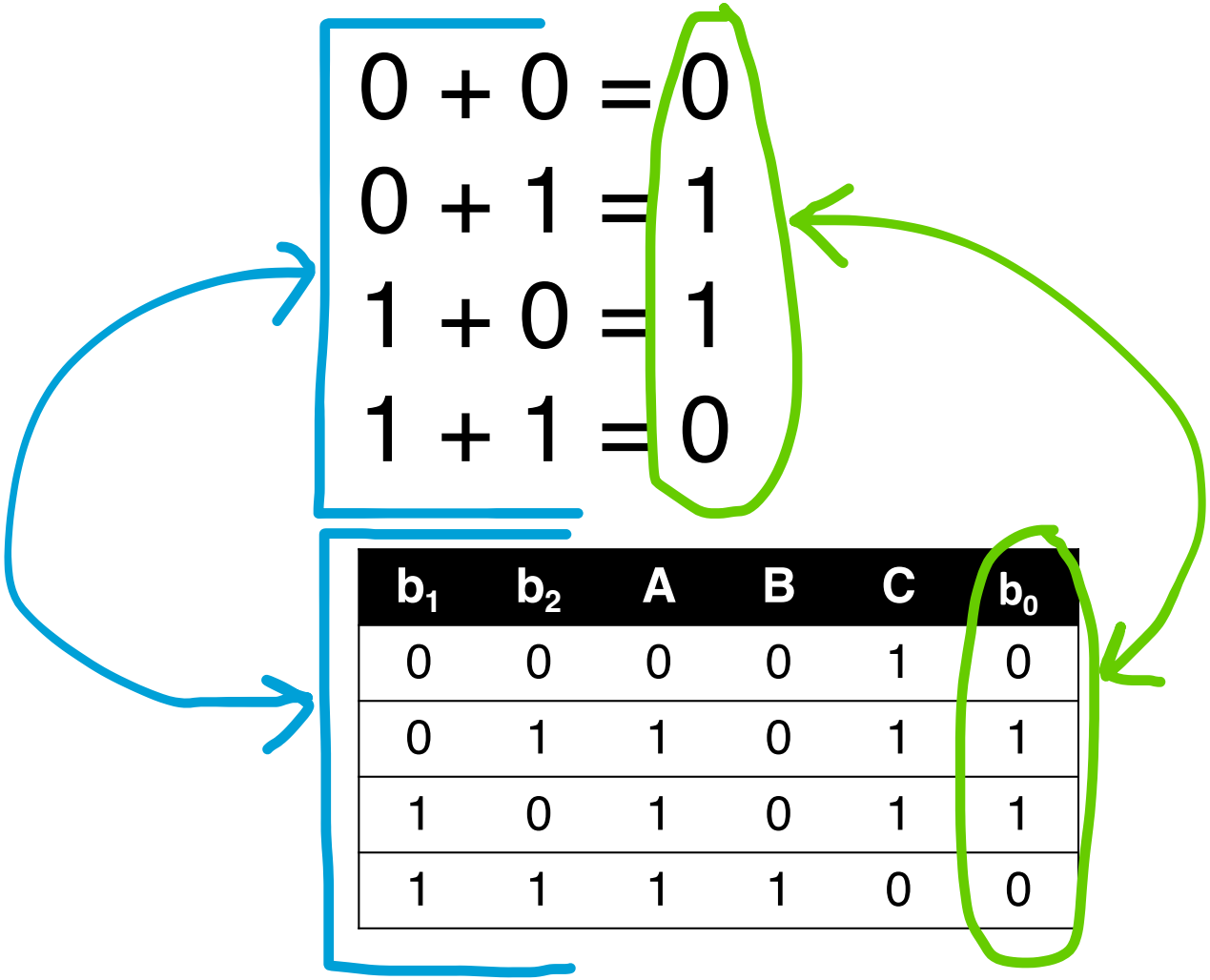
Let's see who this system works in the following slide.



b_1	b_2	A	B	C	b_0
0	0	0	0	1	0
0	1	1	0	1	1
1	0	1	0	1	1
1	1	1	1	0	0

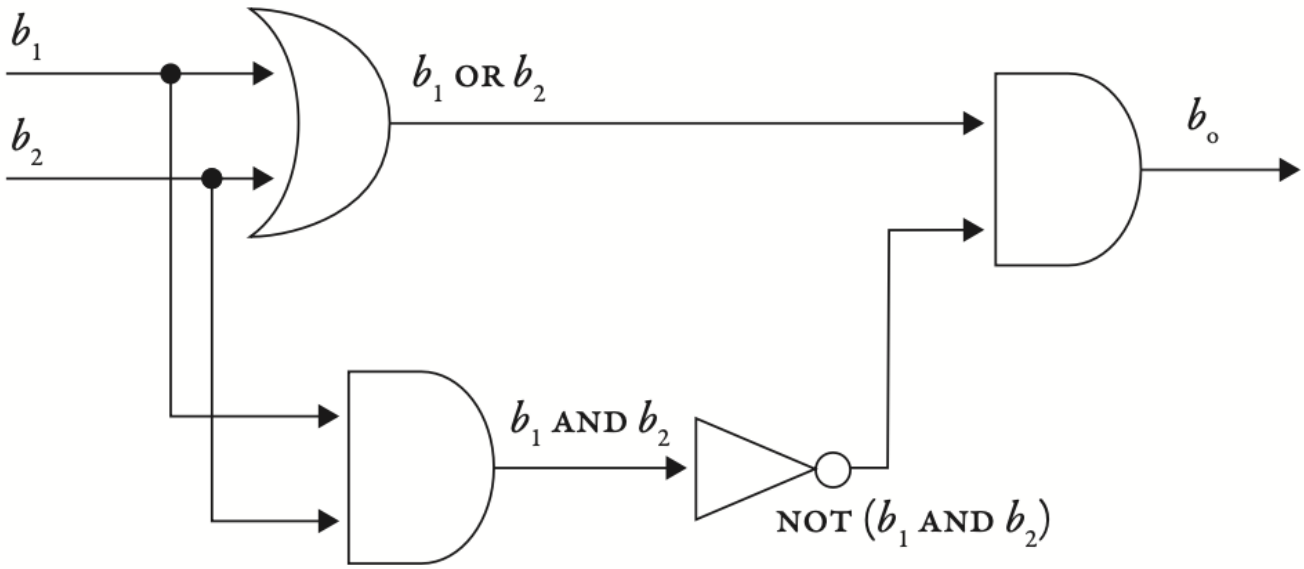
The system uses the inputs b_1 and b_2 to compute (b_1 OR b_2) (which we called A) on the one side, and the negation of (b_1 AND b_2) on the other. The negation of (b_1 AND b_2) is computed by means of an AND gate (obtaining B) and then a NOT gate (obtaining C).

The final computation, which gives the output, combines A and C with an AND, which corresponds to setting the final result to 1 in all cases when (b_1 OR b_2) is 1, except for when (b_1 AND b_2) is 1: in that case C is 0 and, through the last AND gate, sets the final result to 0.



b_1	b_2	A	B	C	b_0
0	0	0	0	1	0
0	1	1	0	1	1
1	0	1	0	1	1
1	1	1	1	0	0

That electronic circuit, comprised of one OR gate, one NOT gate, and two AND gates combined as previously shown, yields the same output as the addition applied to two single-bit inputs. Thus, it can work as an ADDER.
From a physical perspective, it is just a circuit that manipulates electric tensions, but with the fundamental encoding in mind, we can see its (electric) operation as an (arithmetic) operation of addition.



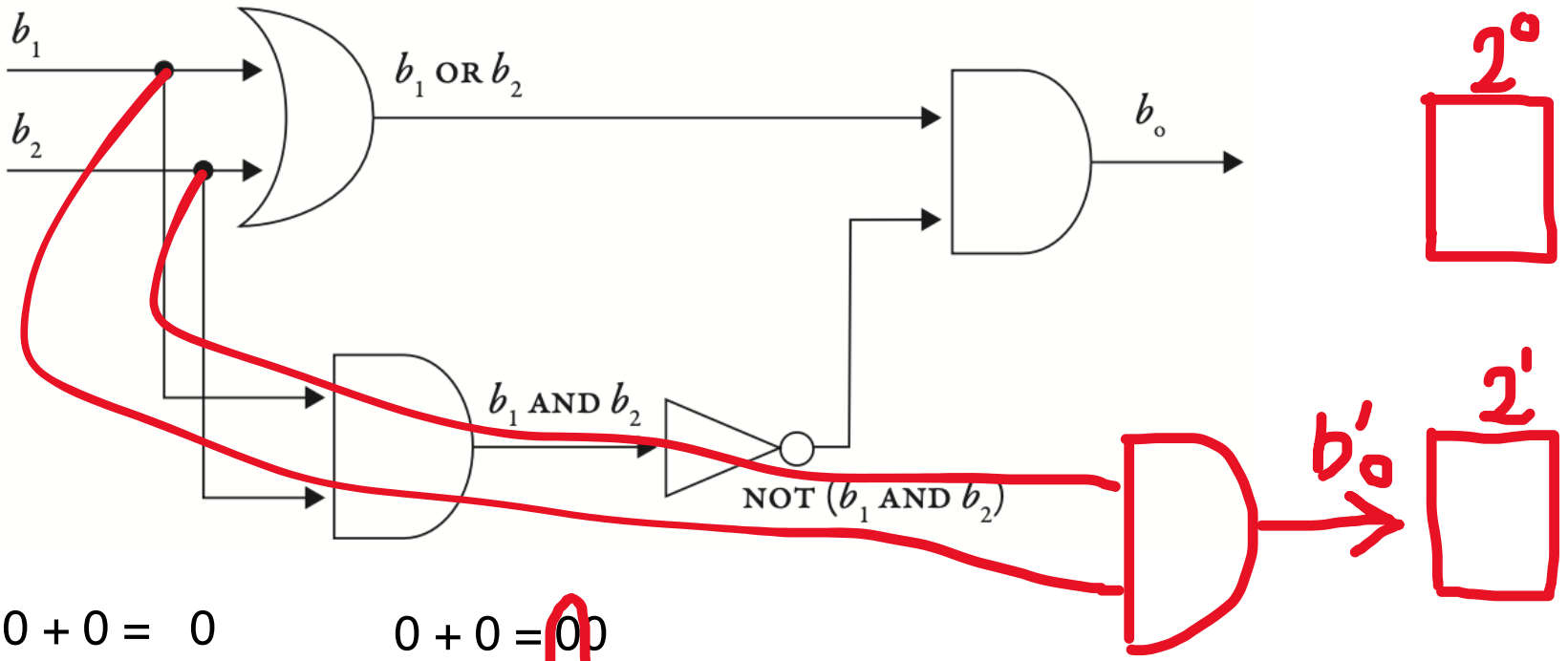
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

0 + 0 = 00
0 + 1 = 01
1 + 0 = 01
1 + 1 = 10

So to build the circuit that computes the carry, we just need to put the two inputs through an AND gate.

We must not forget about the carry.

The carry is another bit that is part of the final result. It is 0 in the first 3 cases, and 1 in the last. The output bits are exactly like the output of (b_1 AND b_2).



$0 + 0 = 0$
$0 + 1 = 1$
$1 + 0 = 1$
$1 + 1 = 10$

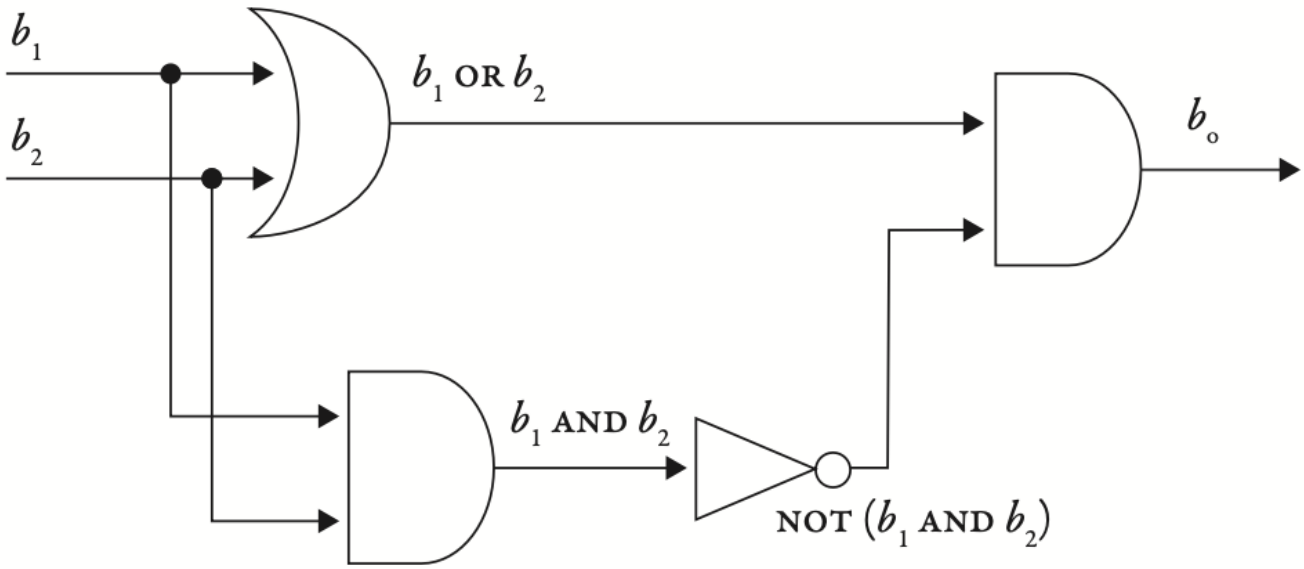
$0 + 0 = 00$
$0 + 1 = 01$
$1 + 0 = 01$
$1 + 1 = 10$

We must not forget about the carry.

The carry is another bit that is part of the final result. It is 0 in the first 3 cases, and 1 in the last. The output bits are exactly like the output of (b_1 AND b_2).

So to build the circuit that computes the carry, we just need to put the two inputs through an AND gate.

The bit in output from the first system (in black, printed) is saved in a 1-bit memory space that represents how many units (2^0) are in the final result.
The bit in output from the second system (in red, handwritten) is saved in another memory unit, indicating how many twos (2^1) are in the final result.



Since we have built an arithmetic system by combining a logical system of AND, OR, and NOT, does this mean that arithmetic is based on logic?

Absolutely not. In the real world, arithmetic and logic have a very distinct origin. Actually, since counting with fingers came before formalizing reasoning, if anything, arithmetic should be considered the basis of a way of thinking that lead humanity to conceive logic at a later stage, but these are questions for historians and philosophers of science.

It is just that, in the material world of electronic circuits built my means of transistors, logical operators are much simpler to implement than arithmetic operations. You can see it by noticing that we need only one transistor to build a NOT gate, whereas an ADDER has a much more complex structure.

Computer Science gives us extremely useful devices, but those devices are the result of discoveries in science (e.g. semiconductors) and socio-political agreements (e.g. encodings like RGB, standards like JPG). They should not be taken as an indication of some fundamental principles regarding how the universe works or how our thinking and reasoning work.

And yet, there are some subfields of this discipline whose experts forget about this. You must not.

Reasoning

$$\frac{A, \quad A \Rightarrow B}{B}$$

Logic is the discipline that aims at the formalization of reasoning processes.

A	B	$A \Rightarrow B$
t	t	t
t	f	f
f	t	t
f	f	t

Some logic definitions: conditional and its equivalence to a combination of negation and disjunction.

A	B	$\neg A$	$\neg A \vee B$	$A \Rightarrow B$
t	t	f	t	t
t	f	f	f	f
f	t	t	t	t
f	f	t	t	t

Automated Reasoning

$$\frac{A, \quad A \Rightarrow B}{B}$$

$$\frac{A, \quad \neg A \vee B}{B}$$

Automation, in this context, means to transform the task into something that can be executed by the computer (as usual, it is symbol processing).

Davis, Martin; Putnam, Hilary (1960).
"A Computing Procedure for Quantification Theory"
J. ACM. 7 (3): 201–215.

Robinson, J. Alan (1965).
"A Machine-Oriented Logic Based on the Resolution Principle".
J. ACM. 12 (1): 23–41