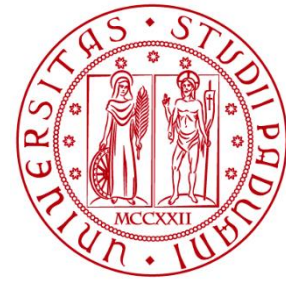




OF THE
DEPARTMENT OF
INFORMATION ENGINEERING



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Digital Systems

Introduction to Digital Systems

Marta Bagatin, marta.bagatin@unipd.it

Degree Course in Information Engineering
Academic Year 2023-2024

Introduction to Digital Systems

The Information Age

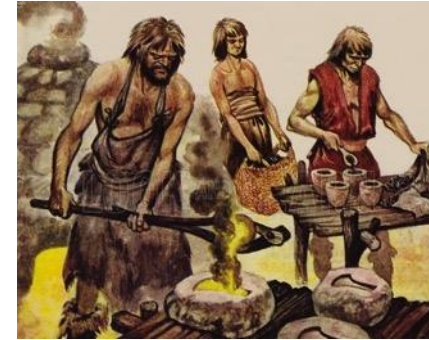
Stone Age



Bronze Age



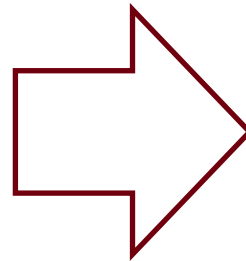
Iron Age



Machine age (1880 -19xx)



Information age (20xx)

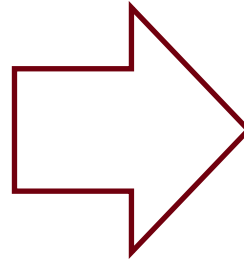


The Digital Revolution

Machine age (1880 - 19xx)



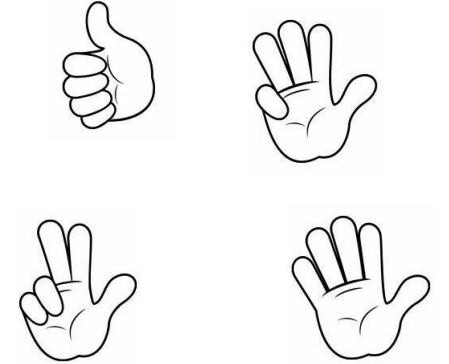
Information age (20xx)



- At the end of the '80's, less than 1% of the information was in digital format. In 2007, 94% was in digital format!

What is a Digital System?

- Digital: «made with digits»
- «Digit» comes from the Latin «digitus» (= fingers): the oldest method of counting numbers is to use the fingers
- A digital system **stores, moves, and processes digital information**, that is **operates on discrete elements**
 - A **set with a finite number of elements** contains discrete elements (there is a neighborhood of each point of the set in which there is no other element of the set)
 - Examples: the 10 fingers of the hands, the 26 letters of the alphabet, the 54 playing cards in a deck, the number of people in a room, ...



Digital Signals vs. Analog Signals

- A **digital system** operates on **digital signals**
- The **real world quantities are continuous** (the speed of a body, the intensity of the sound, the atmospheric pressure, ...)
- Continuous quantities are described by **analog signals**, that is, they can assume **all values within a finite range**
- To represent the **quantities in a digital way** we need to **make them discreet**
 - Example: In a digital thermostat controlling a heating system, the temperature of a room (continuous signal) is represented by discrete values, such as integers between 0 and +40

Why Studying Digital Systems?

- In the 1960s, most of the electronic systems were **analog**
- Today, practically all electronic systems are **digital**, except those "at the edge" (i.e. interfacing with the real world)
- Advantages of digital systems
 - No noise accumulation
 - Virtually arbitrary precision
- **Scaling** of the integrated circuits (Moore's law) has allowed a continuous evolution and miniaturization of digital systems over the last decades

Digital Systems and Electrical Signals

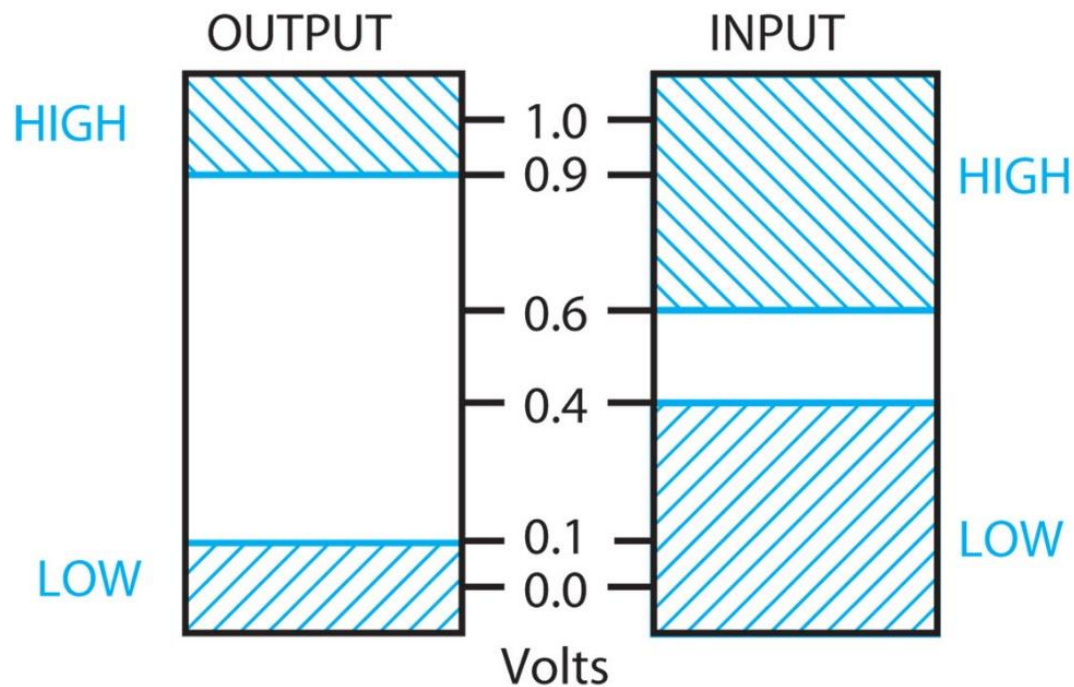
- Digital systems are made of electronic systems that operate on **electrical voltages and currents**, by means of electronic devices called **transistors**
 - Intuitively: transistors are switches that let a signal pass or not, depending on their configuration (ON or OFF)

Binary Digital Systems

- Binary digital systems use signals that can assume only **two discrete values**
 - These discrete values are represented with '**1**' and '**0**' (digits of the binary number system), **ON** and **OFF**, or **TRUE** and **FALSE**
 - A **binary digit** is called **bit** (Binary digIT) of information
- These two values correspond to physical quantities (that is to two ranges of electrical voltage), called **HIGH logical value** and **LOW logic value**, which are the inputs and outputs of the digital system

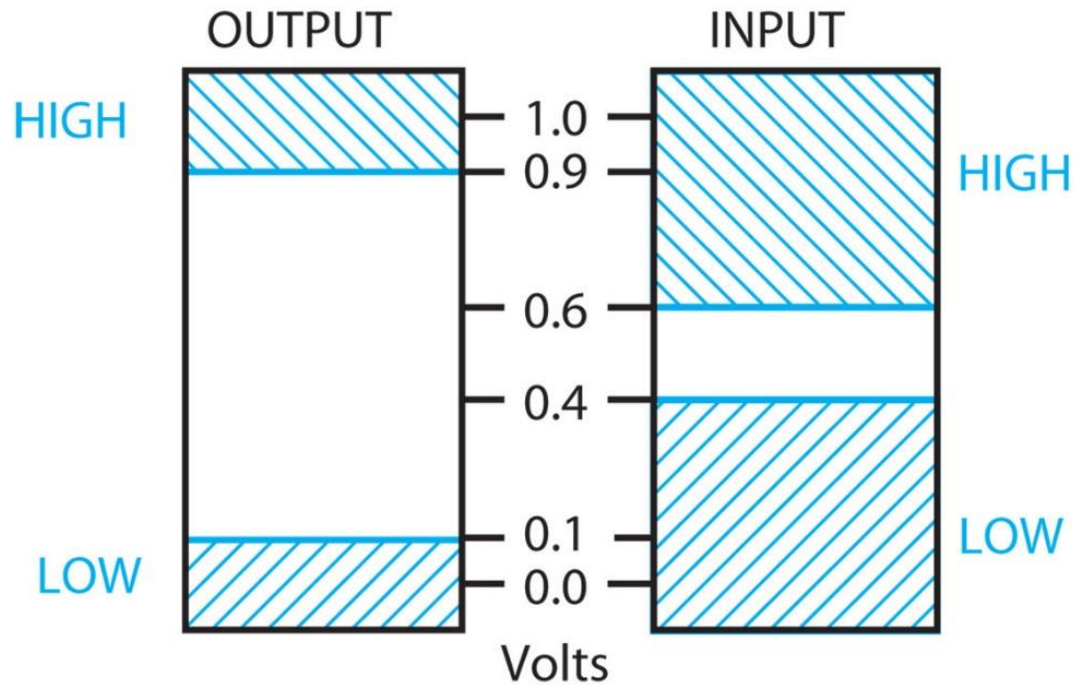
Inputs and Outputs

Voltage values that the inputs and outputs of a binary system can assume are binary: HIGH or LOW



- The **input** is recognized as
 - **HIGH** if it is between 0.6 V and 1.1 V,
 - **LOW** if it is between -0.1 V and 0.4 V
- The **output** is recognized as
 - **HIGH** if it is between 0.9 V and 1.1V
 - **LOW** if is between -0.1 V and 0.1 V
- The range of **values allowed at the input** is **wider** than the range of values allowed at the output

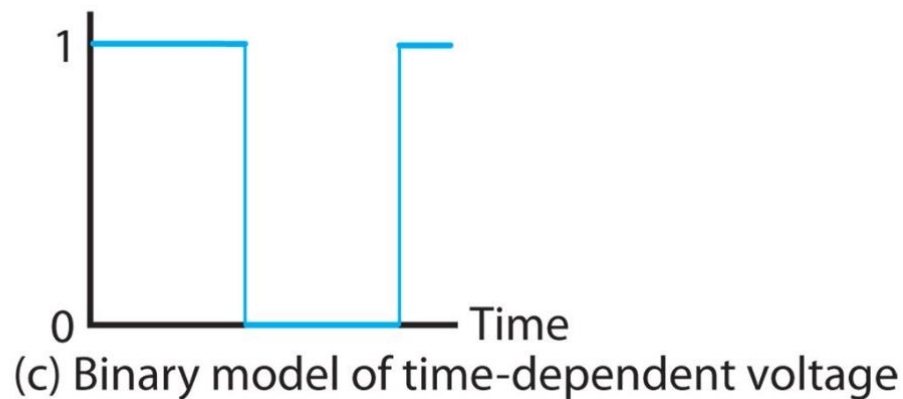
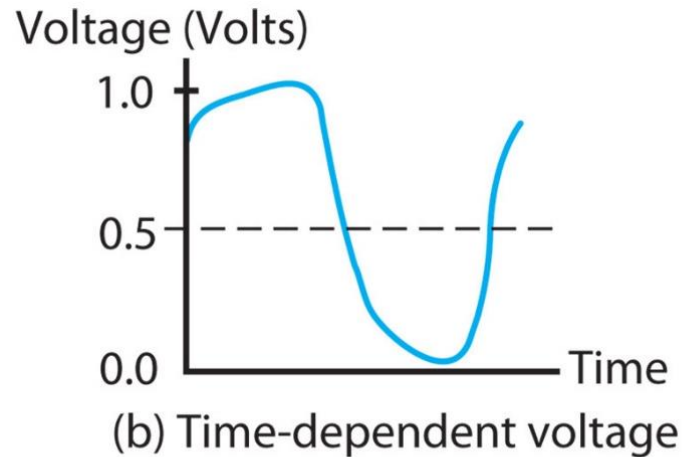
Inputs and Outputs



(a) Example voltage ranges

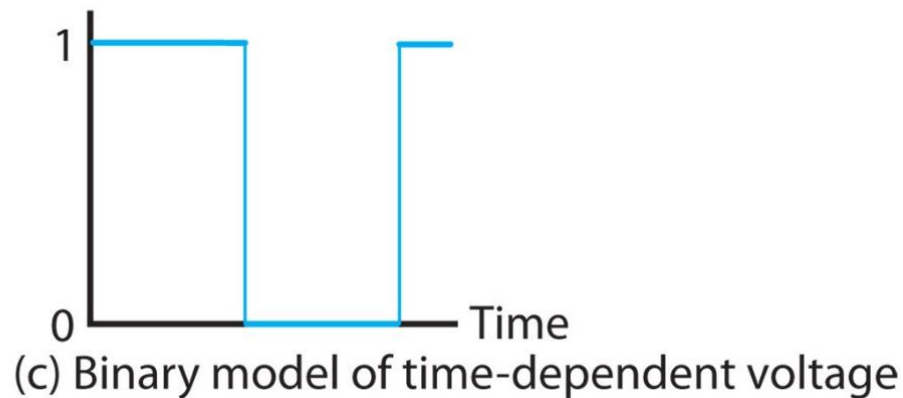
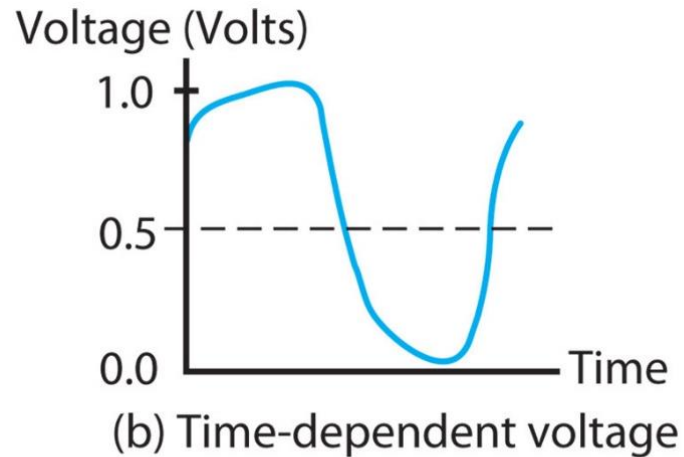
- The system is able to operate correctly, even in the presence of noise superimposed to the input signal: **regenerative property**
- Digital circuits are more **noise tolerant** compared to analog circuits. In presence of undesired variations on the inputs, they are able to transmit **cleaner outputs, less distorted by noise**

Binary Representation of Signals



- A continuous signal (e.g. voltage) can be **represented with a binary waveform**: all the **values above 0.5 V** are read as '**1**' (**HIGH logic value**), those **below 0.5 V** are read as '**0**' (**LOW logic value**)
- At a logic level, we idealize the behavior of inputs and outputs of a digital system with that of binary digits '**0**' and '**1**'

Binary Representation of Signals



- This idealization, in which the electrical aspects are neglected and only the logic ones are highlighted, is an example of **abstraction**
- We say that the **logic description has a higher level of abstraction** (= less detailed) **than the electrical description**

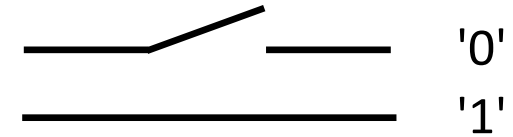
Binary Digital Systems

- In binary digital systems, information is represented in the form of **groups of bits**
- We can use **different coding techniques**. Groups of bits can represent **not only numbers, but also other groups of discrete symbols**
 - Example: ASCII code → 7-bit codes used to encode the characters of a keyboard

Ex: Representation of Quantities with Digital Signals

(a) Binary information: single-bit binary signals

- Light on or off, door open or closed



(b) Discrete elements of a set: multi-bit binary signals, i.e. groups of bits

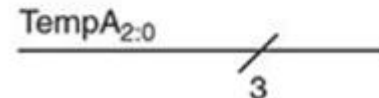
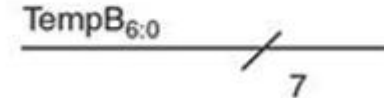
- 8 colors can be represented with 3 bits

000	white	011	purple
001	red	101	orange
010	blue	110	green
100	yellow	111	black

(c) Continuous quantities: can be quantized and represented with multi-bit binary signals

- 8 temperatures represented with different codes (representation is chosen depending on the type of application)

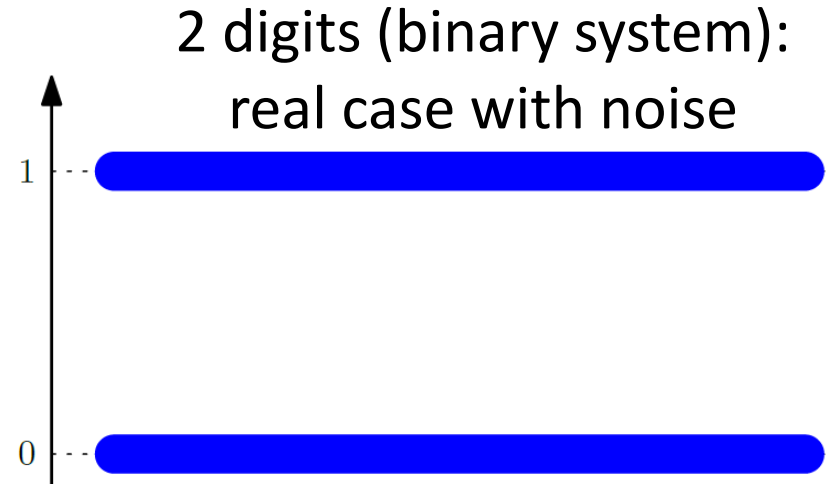
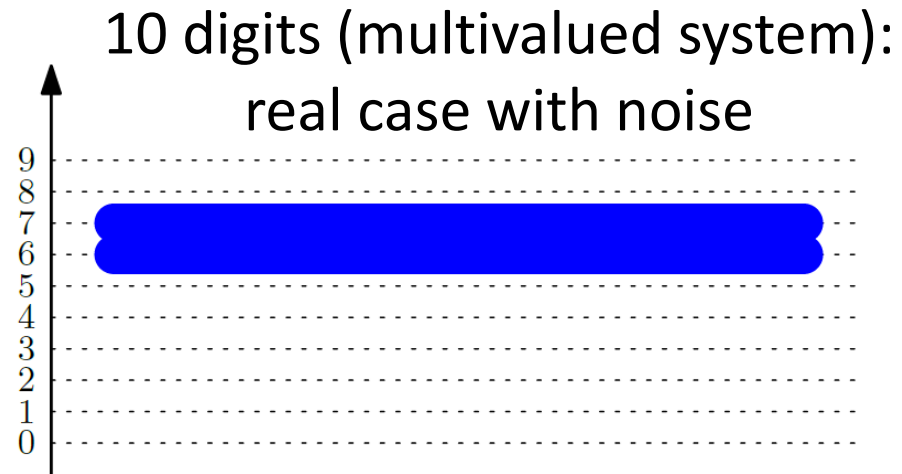
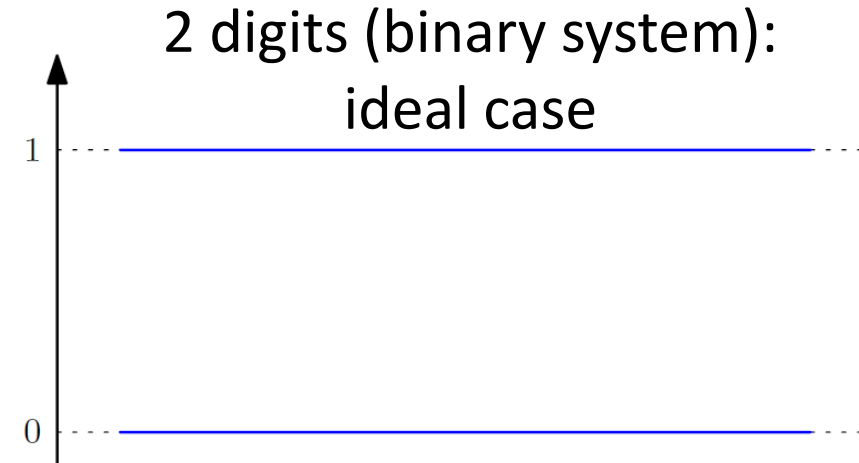
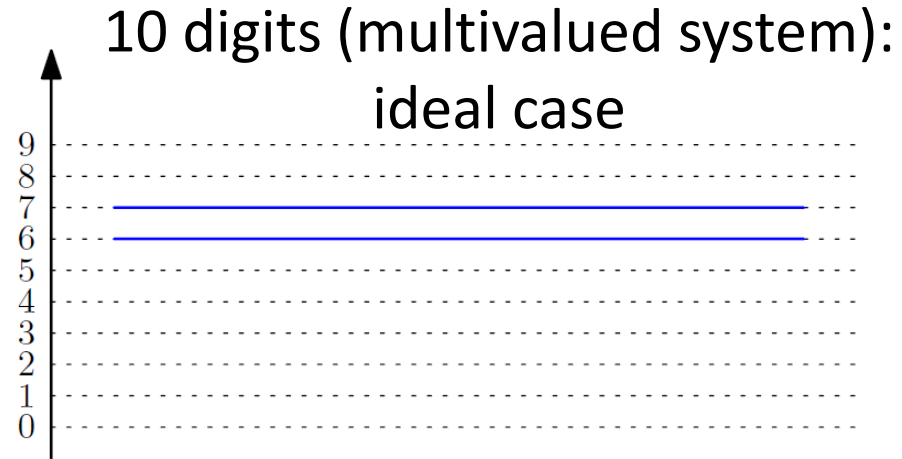
000	68	100	76	0000000	68	0001111	76
001	70	101	78	0000001	70	0011111	78
010	72	110	80	0000011	72	0111111	80
011	74	111	82	0000111	74	1111111	82

TempA_{2:0}  TempB_{6:0} 

Why Using Binary Systems?

- A binary system has a **higher noise tolerance than a multivalued digital system**
 - If we divide the voltage range into 10 intervals instead of 2 (we will have 10 possible input values and 10 output values): with the same input noise, it will be easier for the value to be interpreted incorrectly due to the noise (i.e., read in an adjacent slot)
- So **binary digital systems** are **more reliable** compared to multivalued digital systems, ensuring better operation in presence of noise at the input and more repeatable results (same set of inputs gives the same set of outputs)

Why Using Binary Systems?

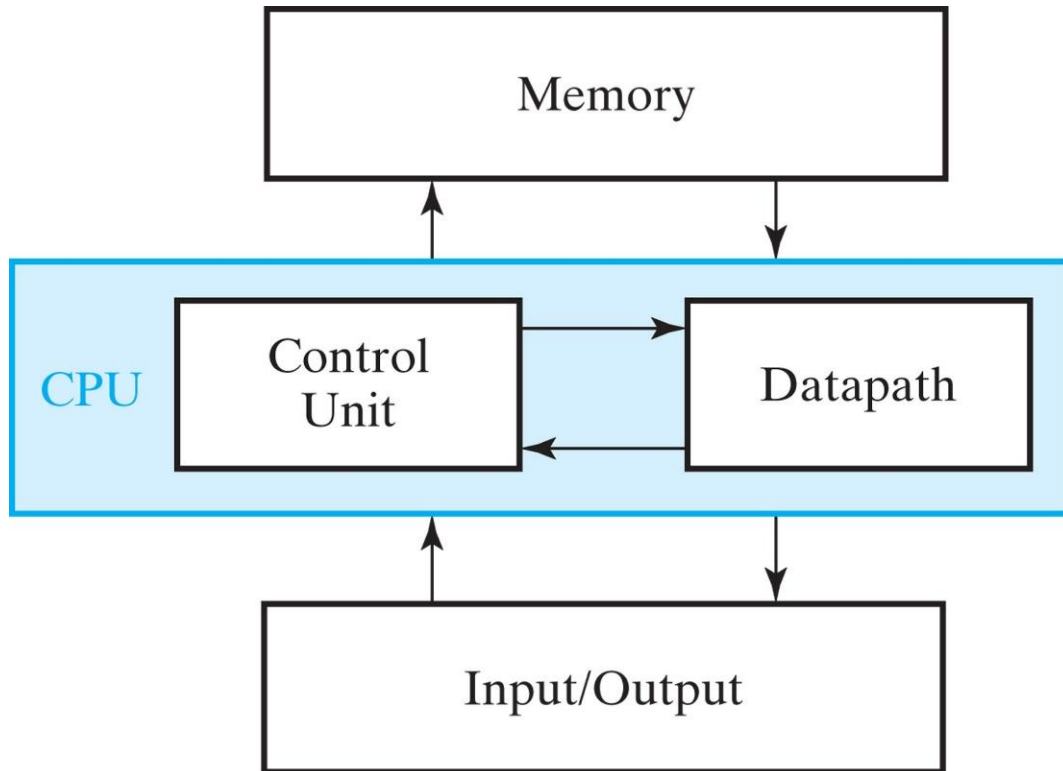


Digital Computer and Beyond

A Look at the Computer

- The computer is the most popular **digital system**
- The term "**digital computer**" derives from the fact that the first computers were mainly used for numerical computations (the discrete elements on which they operated were the digits)
- Today computers are used for data processing virtually in **all fields**
 - Ex: industrial applications, banking and insurance, medical, transportation, meteorology, communication, entertainment, and engineering design (CAD tools - Computer Aided Design, used in electronics, construction, mechanics, ...)

Block Diagram of a Computer



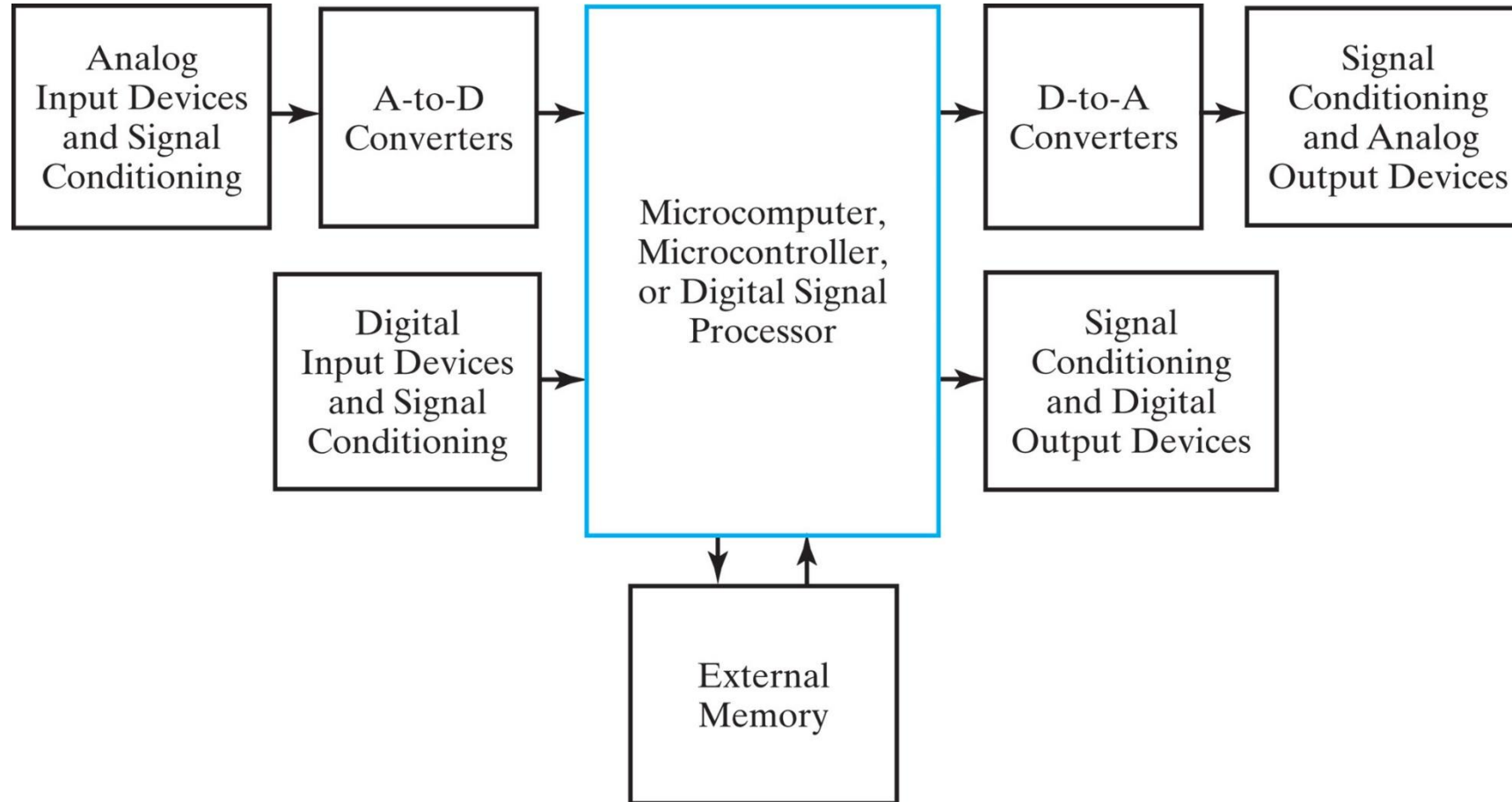
Copyright ©2016 Pearson Education, All Rights Reserved

- **Memory**: stores inputs, outputs, intermediate data, instructions
- **CPU: Central Processing Unit**
 - **Control Unit**: supervises the information flow between the various blocks
 - **Datapath**: performs operations on data, as specified by the program
- **Input devices** (keyboard, scanner, microphone), **output devices** (screen, printer, speakers) or **I/O devices** (USB stick, DVD)

Beyond the Computer

- The definition of a computer can be extended to other components that operate in a similar way
- **Microcomputer** or **microcontroller** or **DSP (Digital Signal Processor)** are less powerful and more compact computers, often integrated in a single silicon chip, typically incorporated into complex systems (car, mobile phone, washing machine, air conditioner)
 - Referred to as **embedded** microcomputers (the opposite of embedded is stand-alone)

Blocks of an Embedded System



Copyright ©2016 Pearson Education, All Rights Reserved

Digital and Analog I/O Devices: Examples

- **Digital input devices**

- Limit switch: device with binary digital input, may or may not have a force applied to the input
- Remote control: device with digital multivalued input (ex. pressing one of its 10 keys)

- **Analog input devices**

- Thermocouple: sensor that receives a temperature at the input and produces a voltage proportional to the temperature at the output
- Quartz: composed of crystals which, in response to an applied pressure, deform and polarize, producing an electrical voltage at the output

- **Analog output devices**

- Speaker: produces (continuous) sound waves by converting an electrical signal

- **Digital output devices**

- LED (Light-Emitting Diode) display: can be ON or OFF, depending on the applied voltage
- Electromechanical relay: can be OPEN or CLOSED, depending on the applied voltage

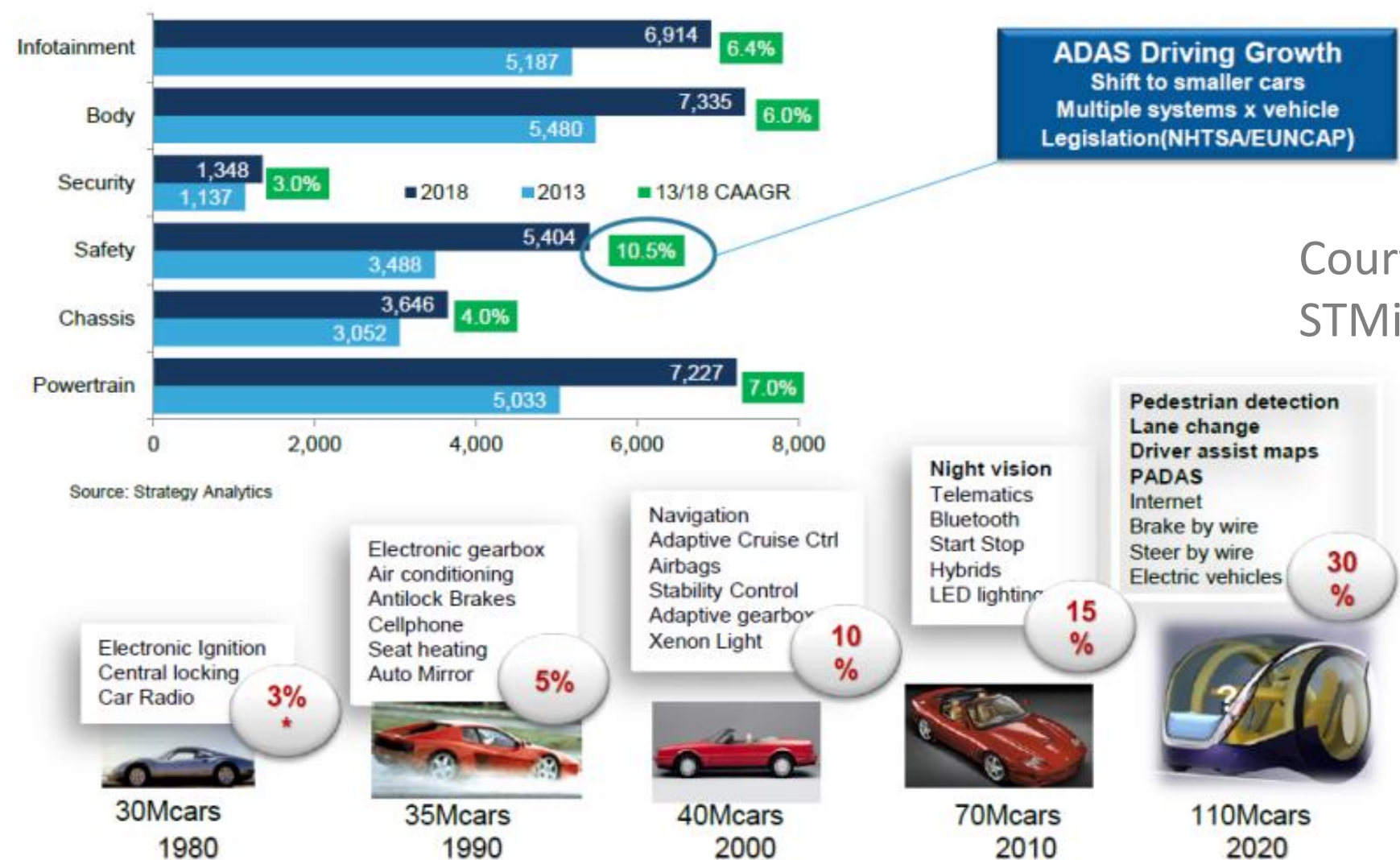
Examples of Embedded Systems

Application Area	Product
Banking, commerce and manufacturing	Copiers, FAX machines, UPC scanners, vending machines, automatic teller machines, automated warehouses, industrial robots, 3D printers
Communication	Wireless access points, network routers, satellites
Games and toys	Video games, handheld games, talking stuffed toys
Home appliances	Digital alarm clocks, conventional and microwave ovens, dishwashers
Media	CD players, DVD players, flat panel TVs, digital cameras, digital video cameras
Medical equipment	Pacemakers, incubators, magnetic resonance imaging
Personal	Digital watches, MP3 players, smart phones, wearable fitness trackers
Transportation and navigation	Electronic engine controls, traffic light controllers, aircraft flight controls, global positioning systems

Examples of Embedded Systems

- A modern car can hold several **dozens of microcontrollers/ microcomputers**
- Each of these deals with a functional block in the car
 - ABS control
 - stability control
 - engine control
 - ...

Example: the Modern Car



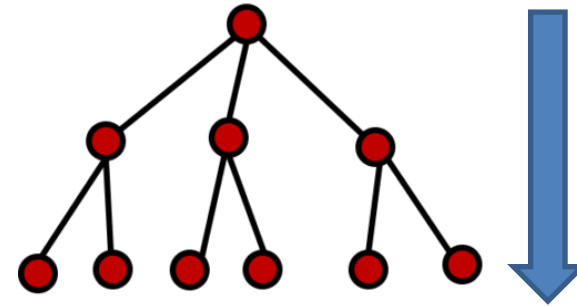
Courtesy of M. Duncan,
STMicroelectronics

* Percentage of the cost of electronics on the total cost of a car

Levels of Abstraction in the Design of Digital Systems

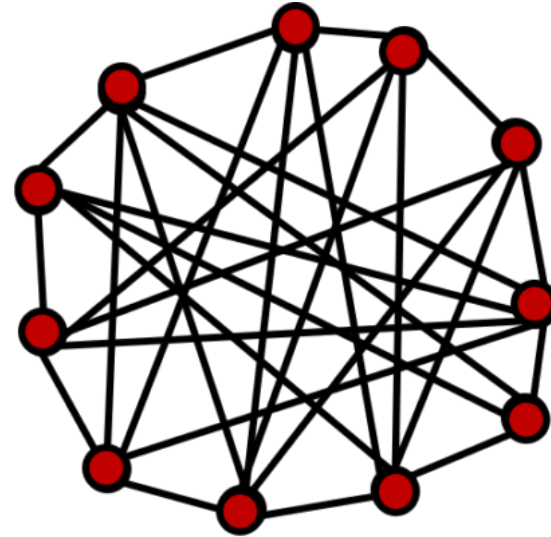
Design of Digital Systems and Levels of Abstraction

- The design of a **digital systems** typically occurs with a **top-down approach**
 - It starts from system **specifications** at a high level of abstraction (without dealing with the details of the individual sub-blocks)
 - Subsequently breaks down the system into **smaller and smaller blocks**, until simple blocks are reached (simple enough to be implemented)
 - Finally, the overall system is built from the **union of these blocks**, suitably connected to each other



Design of Digital Systems and Levels of Abstraction

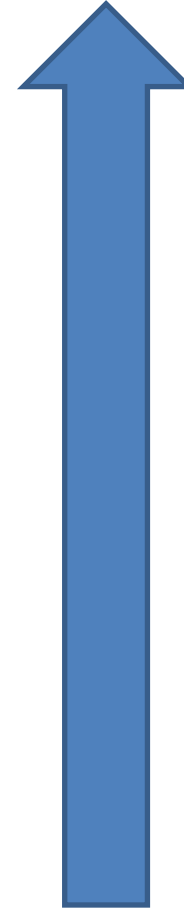
- In this course we will use a **bottom-up** approach
- **We will start from the study and design of the simplest blocks, and then connect them to form a complex system**



Levels of Abstraction

Algorithms
Programming Languages
Operating Systems
Instruction Set Architecture
Microarchitecture
Register Transfers
Logic Gates
Transistor Circuits

Copyright ©2016 Pearson Education, All Rights Reserved



The level of abstraction increases going from the bottom to the top: the details on the implementation are lost going up, a more simplified and general view is acquired.

Each level hides the details and complexity of the level below

Levels of Abstraction

Algorithms
Programming Languages
Operating Systems
Instruction Set Architecture
Microarchitecture
Register Transfers
Logic Gates
Transistor Circuits

Copyright ©2016 Pearson Education, All Rights Reserved

Algorithm: description of **sequences of steps** to perform a certain operation. The algorithm is **generic**, as it is independent from a particular programming language

Levels of Abstraction

Algorithms
Programming Languages
Operating Systems
Instruction Set Architecture
Microarchitecture
Register Transfers
Logic Gates
Transistor Circuits

Copyright ©2016 Pearson Education, All Rights Reserved

Programming language (ex. Java, C ++): includes a series of **instructions to implement the algorithm** and produce given output data. It can work on different operating systems

Levels of Abstraction

Algorithms
Programming Languages
Operating Systems
Instruction Set Architecture
Microarchitecture
Register Transfers
Logic Gates
Transistor Circuits

Copyright ©2016 Pearson Education, All Rights Reserved

Operating system: controls the flow of the program execution and it manages the allocation of resources between the various programs.

The operating system can operate on several processors

Levels of Abstraction

Algorithms
Programming Languages
Operating Systems
Instruction Set Architecture
Microarchitecture
Register Transfers
Logic Gates
Transistor Circuits

Copyright ©2016 Pearson Education, All Rights Reserved

ISA: set of elementary instructions in machine language (processor language) into which the program must be translated to run on a specific processor

Levels of Abstraction

Algorithms
Programming Languages
Operating Systems
Instruction Set Architecture
Microarchitecture
Register Transfers
Logic Gates
Transistor Circuits

Copyright ©2016 Pearson Education, All Rights Reserved

Microarchitecture: specific realization (implementation) of the instruction set (ISA) in a particular processor

Levels of Abstraction

Algorithms
Programming Languages
Operating Systems
Instruction Set Architecture
Microarchitecture
Register Transfers
Logic Gates
Transistor Circuits

Copyright ©2016 Pearson Education, All Rights Reserved

Data transfer between registers: Description (e.g. via HDL) of the microoperation as sequences of data transfers, i.e. groups of bits

Levels of Abstraction

Algorithms
Programming Languages
Operating Systems
Instruction Set Architecture
Microarchitecture
Register Transfers
Logic Gates
Transistor Circuits

Copyright ©2016 Pearson Education, All Rights Reserved

Logic gates: bricks that describe the elementary operations from a logical (not electrical!) point of view. Logic gates do a series of **logic operations on bits**

Levels of Abstraction

Algorithms
Programming Languages
Operating Systems
Instruction Set Architecture
Microarchitecture
Register Transfers
Logic Gates
Transistor Circuits

Copyright ©2016 Pearson Education, All Rights Reserved

Transistor: electronic devices which operate as switches. Transistors are the bricks that, suitably connected to each other, build up the logic gates. A transistor-level circuit describes the system from an electrical point of view

Levels of Abstraction

- The system can be changed at a given level of abstraction by **leaving unchanged all the levels above**
 - This allows us to continue to use solutions at high levels of abstraction also when the underlying implementations have been changed
- It is important to **choose the right level of abstraction** to work on
 - Neglecting unnecessary details
 - Optimize the aspects relevant to a certain type of design

Algorithms
Programming Languages
Operating Systems
Instruction Set Architecture
Microarchitecture
Register Transfers
Logic Gates
Transistor Circuits

Copyright ©2010 Pearson Education, All Rights Reserved

Sum: Example of Abstraction Levels

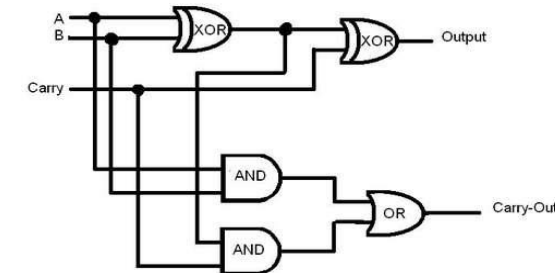
1)	Algorithms
2)	Programming Languages
3)	Operating Systems
4)	Instruction Set Architecture
5)	Microarchitecture
6)	Register Transfers
7)	Logic Gates
8)	Transistor Circuits

Copyright © 2012 Pearson Education, Inc. All rights reserved.

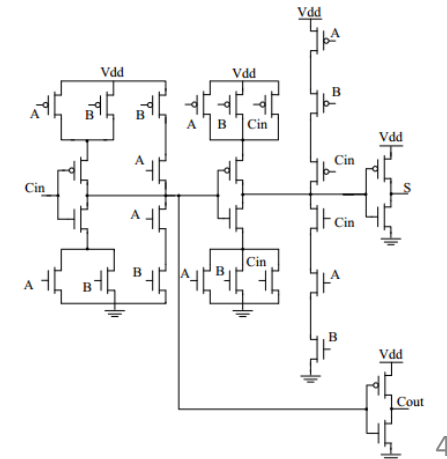
2) Programming language: tells the system to add the values contained in two variables (a, b) and store the content of the sum in a third variable (S)

$$S = a + b$$

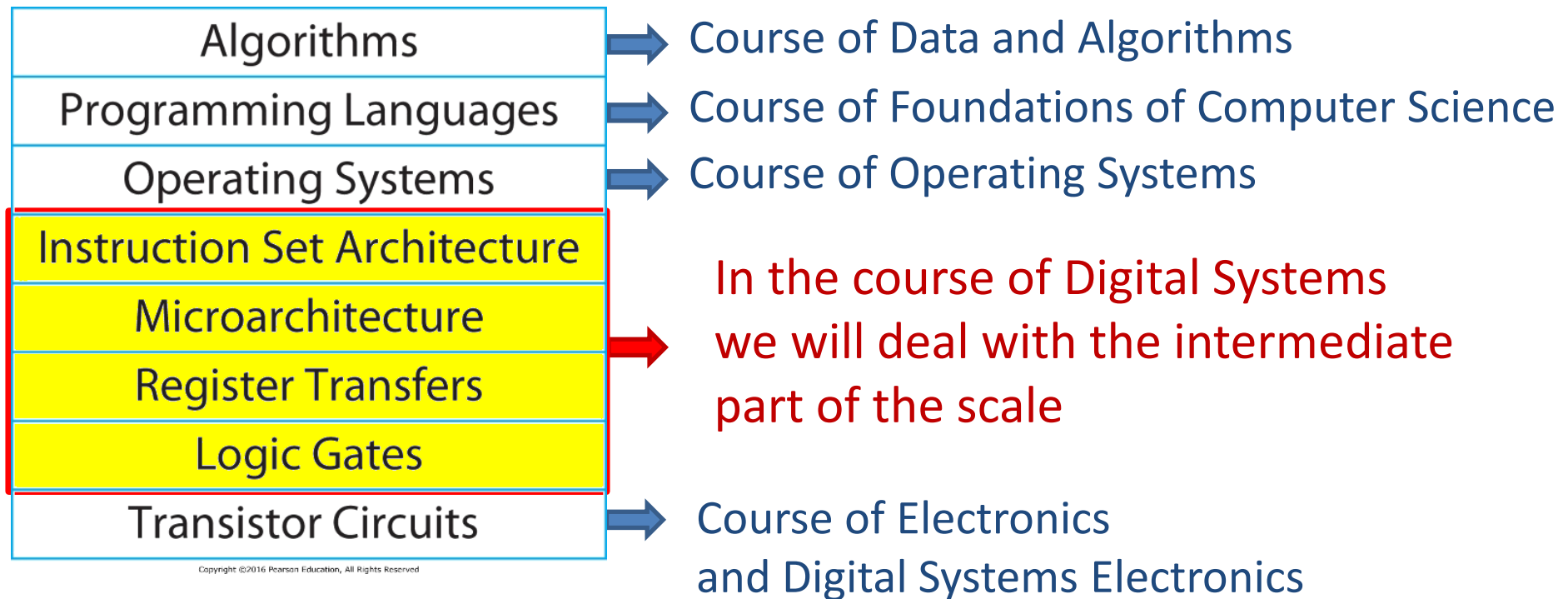
7) Logic gates: detail of the logic gates (their inputs, outputs, connections)



8) Circuit at transistor-level: the sum is performed through a movement of electric charge between the nodes of a circuit, the result of which will be to represent the bits of the sum at the output node of the circuit



Levels of Abstraction



Overview of the Design Process of Digital Systems and VHDL

Design Flow for a Digital System

- **Specification definition:** description of the circuit behavior (in the form of text or HDL code). Specifications abstract from the implementation: they represent what the system does and not how the system does it
- **Formulation:** truth table or boolean equations to define the logic relation between the inputs and outputs of the system
- **Optimization:** minimization of the number of logic gates (transistors, i.e. silicon area). Constraints such as the maximum number of inputs and outputs of a logic gate, delays, etc. must also be considered, which depend on technology. Optimization is an iterative process, which often needs to be repeated after the mapping phase
- **Technological mapping:** translation of the logic diagram/netlist into a diagram that contains the components available in the chosen technology
- **Verify:** check of the correctness of the design

VHDL

- **VHDL** (Very-high-speed integrated circuits **H**ardware **D**escription Language) is one of the most used languages for the design of digital systems
- *VHDL is NOT a programming language!*
 - In a programming language, instructions are executed sequentially by an infrastructure (ex: CPU) => only one instruction at a time is active (in a single-core system)
- In VHDL all **statements** are active at the same time, that is, they are **performed in parallel**
 - The statements of the language define hardware blocks
 - There is no underlying infrastructure, there is no sequential execution

VHDL: Purposes

- **Documentation** of a design
- **Simulation** of the behavior of a design: study of the time evolution of signals to evaluate the correct operation of the circuit
 - Apply a series of input signals to the circuit and evaluate the outputs
- **Automatic synthesis** of a design: transition from a behavioral description to a logic-gate level description, i.e. translation of VHDL code into a network (netlist) of cells (logic cells) physically achievable, which can in turn be used to produce a real circuit
 - Special programs that rely on libraries where the available logic gates are described (supplied by the seller)
- In this course we will NOT deal with circuit synthesis

VHDL: Levels of Abstraction

- Powerful and versatile, VHDL allows to describe a logic circuit at **different levels of abstraction**
 - **Structural**: describes the system in the form of interconnected components, equivalent to a circuit schematic → low level of abstraction
 - **Behavioral**: describes the behavior and functionality of the system, regardless of the actual implementation at the component level → high level of abstraction
 - **Dataflow**: intermediate level of abstraction between structural and behavioral, describes the system in the form of a flow of data

Disclaimer

Figures from *Logic and Computer Design Fundamentals*,
Fifth Edition, GE Mano | Kime| Martin

© 2016 Pearson Education, Ltd