Hardware Systems Definitions & Taxonomies (Part II)

Paolo PRINETTO

Director
CINI Cybersecurity
National Laboratory

Paolo.Prinetto@polito.it

Mob. +39 335 227529

CYBER CHALLENGE
CyberChallenge.IT

cini
Cybersecurity
National Lab

https://cybersecnatlab.it

License & Disclaimer

License Information

Disclaimer

- We disclaim any warranties or representations as to the accuracy or completeness of this material.
- Materials are provided “as is” without warranty of any kind, either express or implied, including without limitation, warranties of merchantability, fitness for a particular purpose, and non-infringement.
- Under no circumstances shall we be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this material.

Prerequisites

- This is the second part of a single lecture, split of usability reason, the 1st part being:
  - *HW_S_1 .1 – Hardware Systems Definitions & Taxonomies - Part I*

Outline

- The concept of System
- System Taxonomies
  - Information coding
  - Memory capability
  - Functionality
  - Implementation
- Data vs Control vs Timing
- Synchronous Sequential Circuits

Implementation

- System-On-Rack
- System-On-Board
- System-In-Package (SIP)
- System-On-Chip (SoC)
- MPSoC
- System-On-FPGA

Implementation

- System-On-Rack
- System-On-Board
- System-In-Package (SIP)
- System-On-Chip (SoC)
- MPSoC
- System-On-FPGA

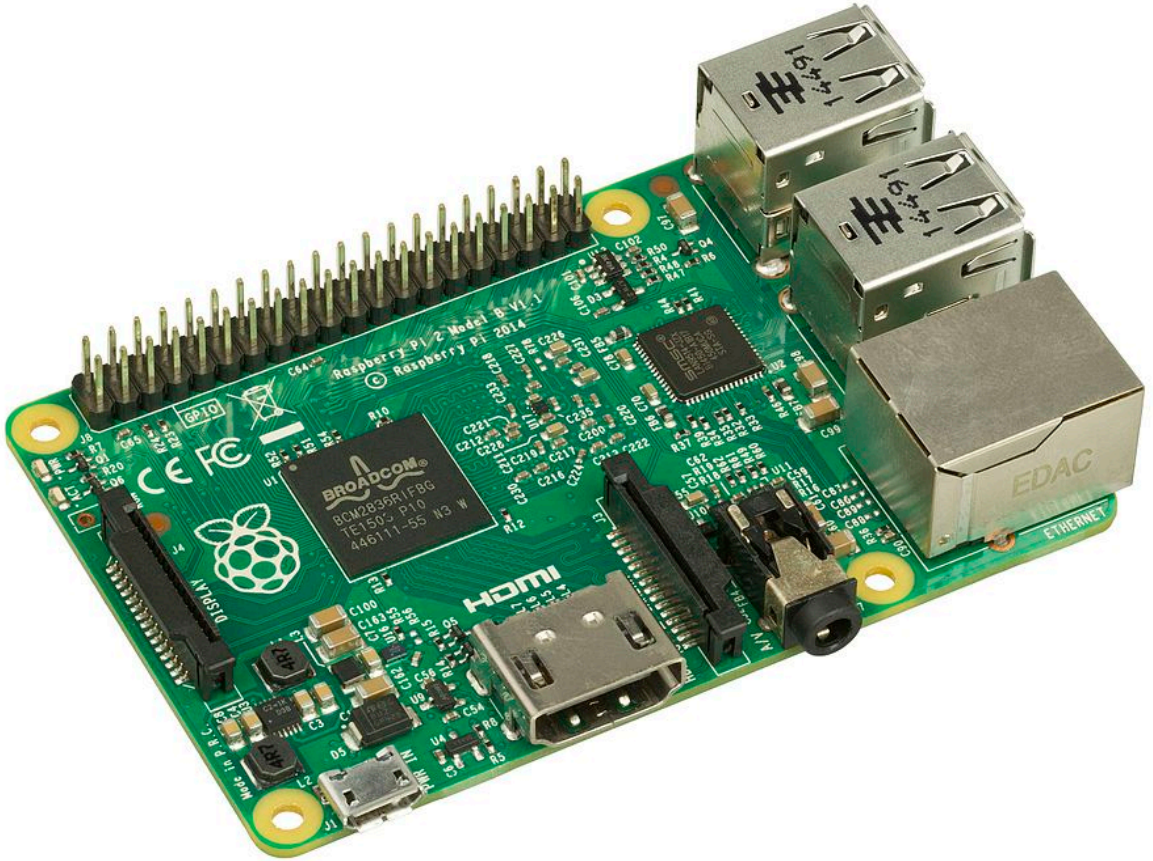
Implementation

- System-On-Rack
- System-On-Board
- System-In-Package (SII
- System-On-Chip (SoC)
- MPSoC
- System-On-FPGA

System-on-Board – An example

The Raspberry Pi 2 (model B), showing:

- an HDMI port
- audio/video port
- micro USB power input
- two ribbon connectors



System-on-Board – An example

- The Raspberry Pi 2 is an inexpensive ($35) and self-contained micro-computer that features a 900 MHz, 32-bit quad-core ARM processor and 1 GB of RAM.
- Built by the Raspberry Pi Foundation, it is designed to be an easy and accessible platform to promote computer science.
- The Pi has also become a popular computer for hobbyists, who use it for personal projects, video game emulators and digital media playback.

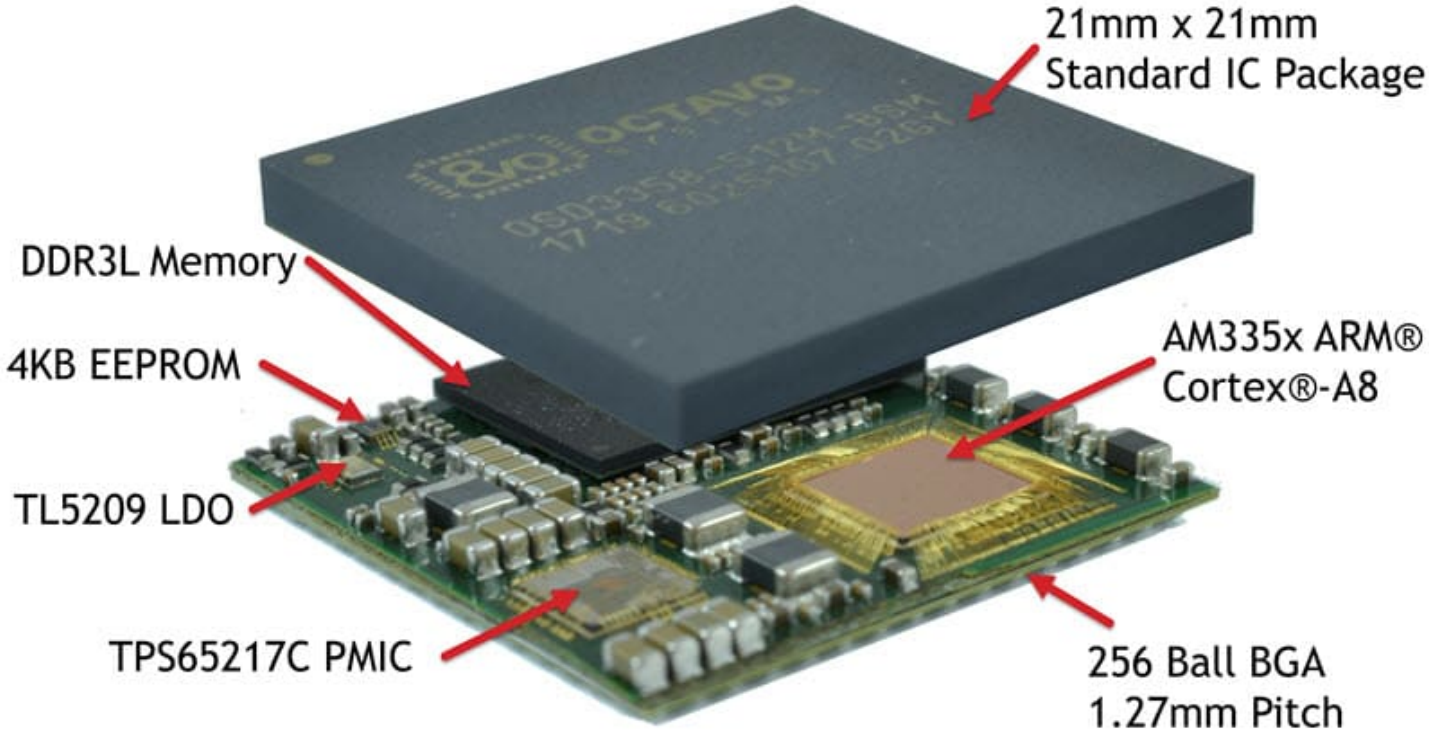
Implementation

- System-On-Rack
- System-On-Board
- System-In-Package (SIP)
- System-On-Chip (SoC)
- MPSoC
- System-On-FPGA

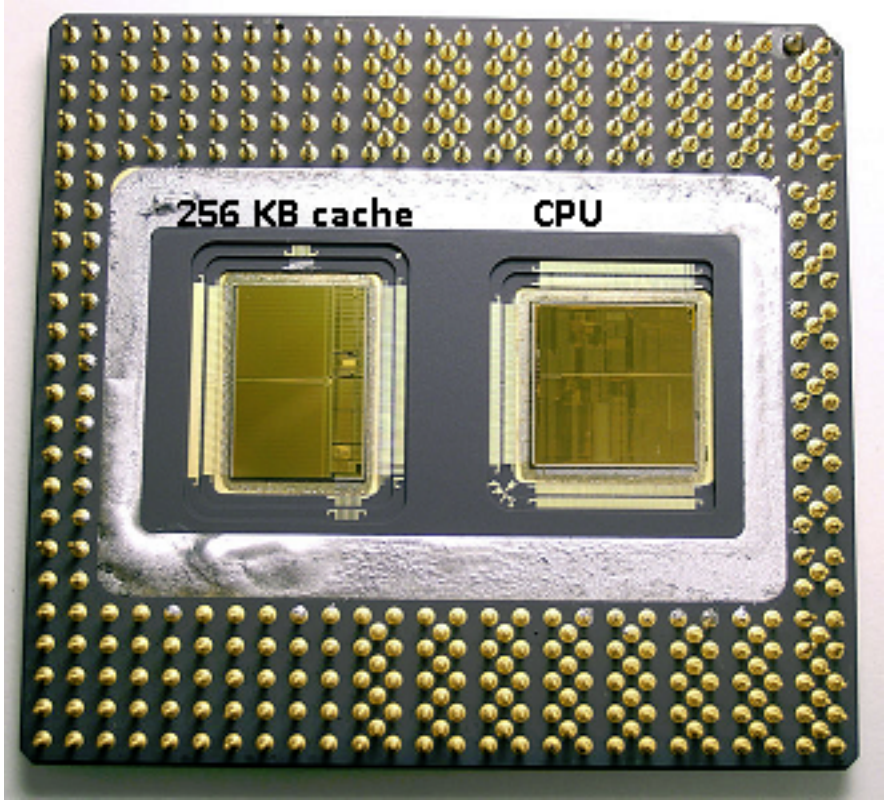
System-in-package (SiP)

- A combination of multiple active electronic components of different functionality, assembled in a single unit that provides multiple functions associated with a system or sub-system

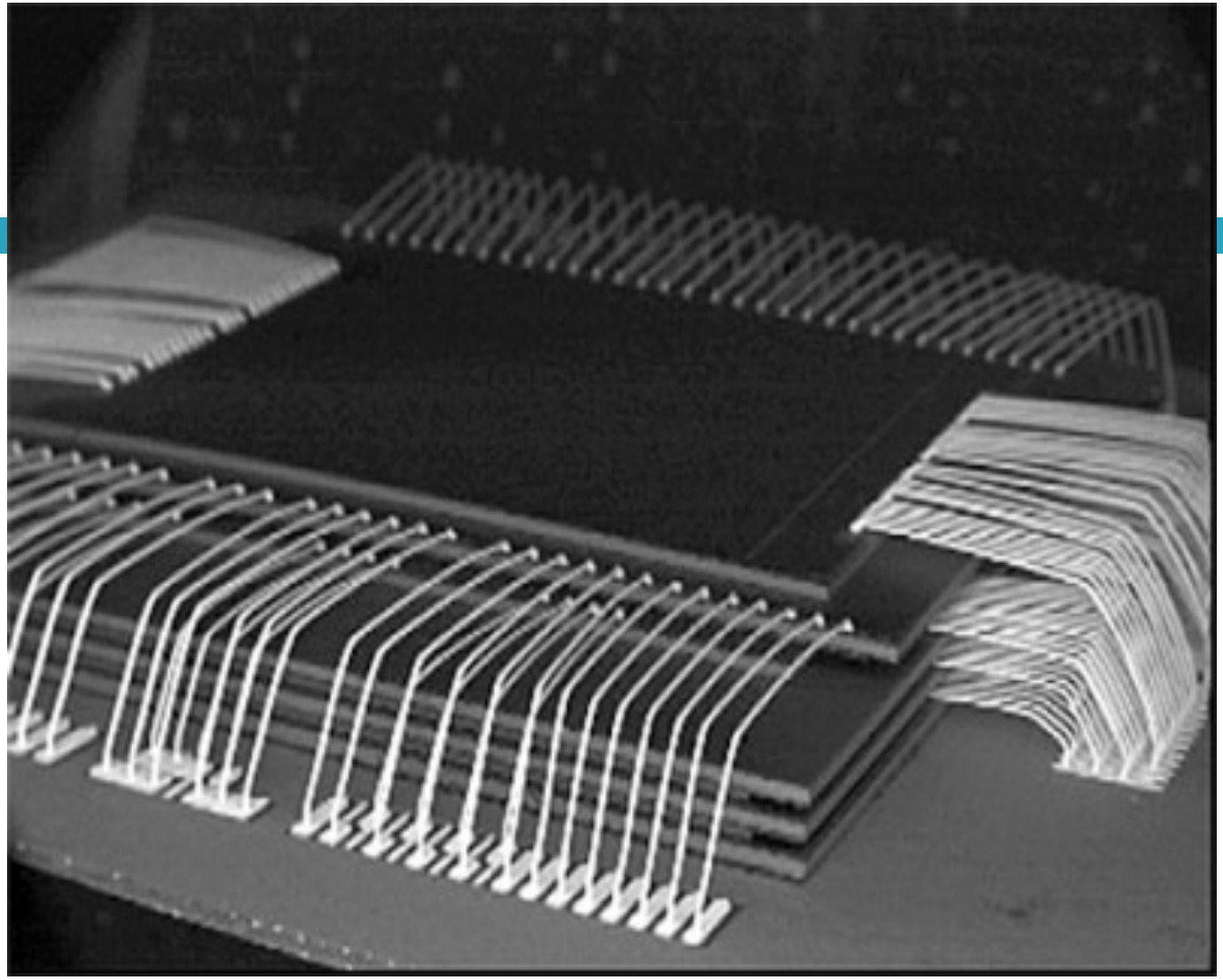
System-in-package (SiP)



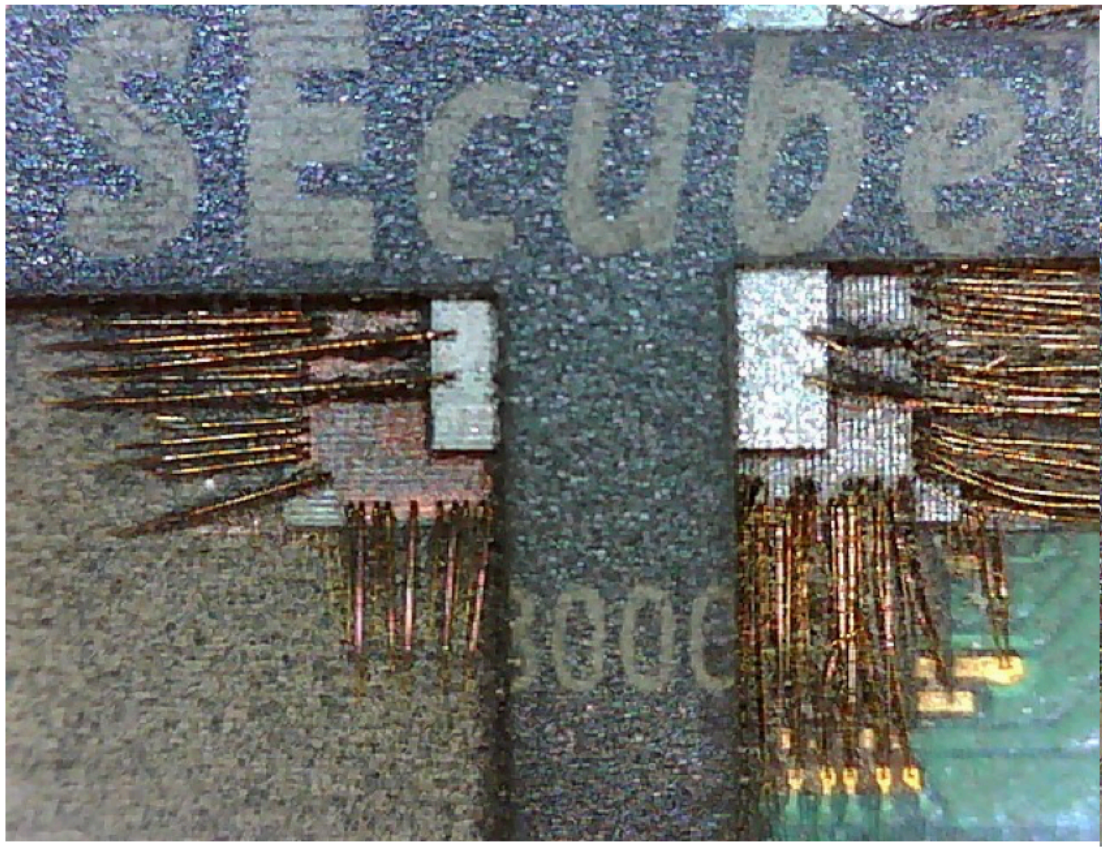
System-in-package (SiP)



3D SiP

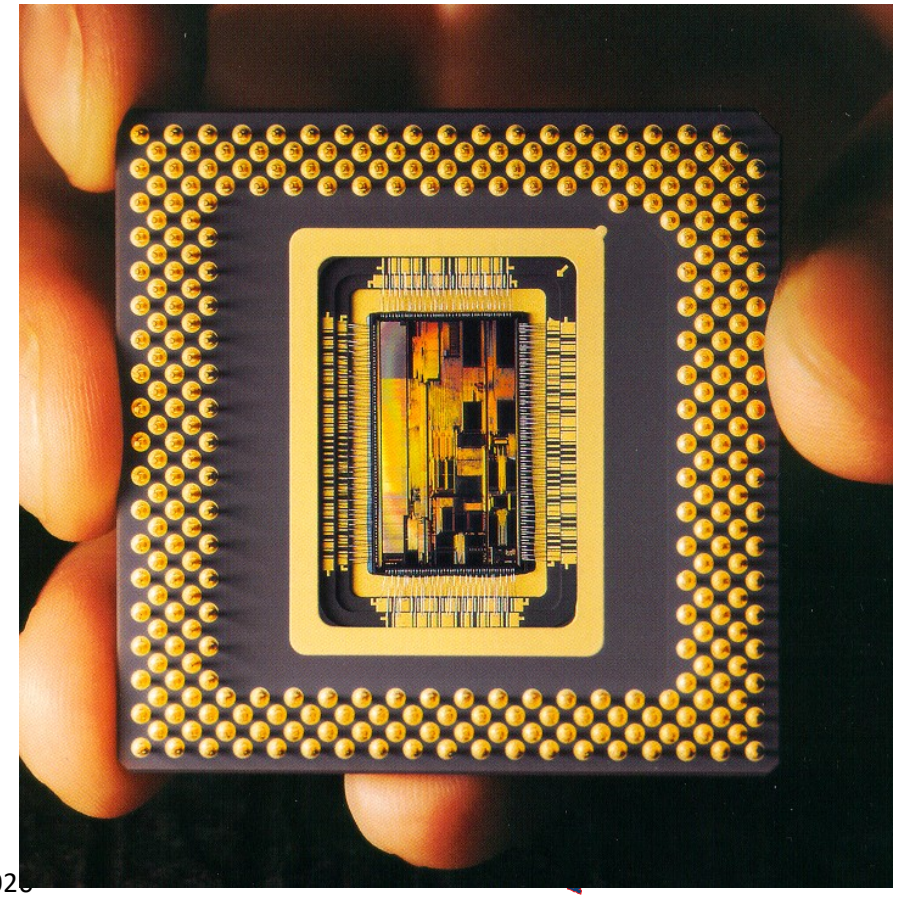


3D SiP – An Example: SEcube™



Implementation

- System-On-Rack
- System-On-Board
- System-In-Package (SIP)
- System-On-Chip (SoC)
- MPSoC
- System-On-FPGA



System-On-Chip (SoC)

- The whole system is implemented on a same chip
- Often referred to as *ASICs* (Application Specific Integrated Circuits)
- When the system is processor-based, one usually distinguishes between:
  - *Microprocessor*
  - *Microcontroller*

Microprocessor

Microprocessor

- The SoC includes just the CPU

Microcontroller

- The SoC includes not only the CPU, but memories (RAM and flashes) and peripheral devices

Implementation

- System-On-Rack
- System-On-Board
- System-In-Package (SIP)
- System-On-Chip (SoC)
- MPSoC
- System-On-FPGA

Pollack's Rule

- In a given process technology, performance increase is roughly proportional to square root of increase in complexity

Pollack's Rule

- In a given process technology, performance increase is roughly proportional to square root of increase in complexity
- In other words, if you double the logic in a processor core, then it delivers only 40% more performance

Consequences: MPSoC MultiProcessor SoC Architecture

- They provide near linear performance improvement with complexity and power

Consequences: MPSoC MultiProcessor SoC Architecture

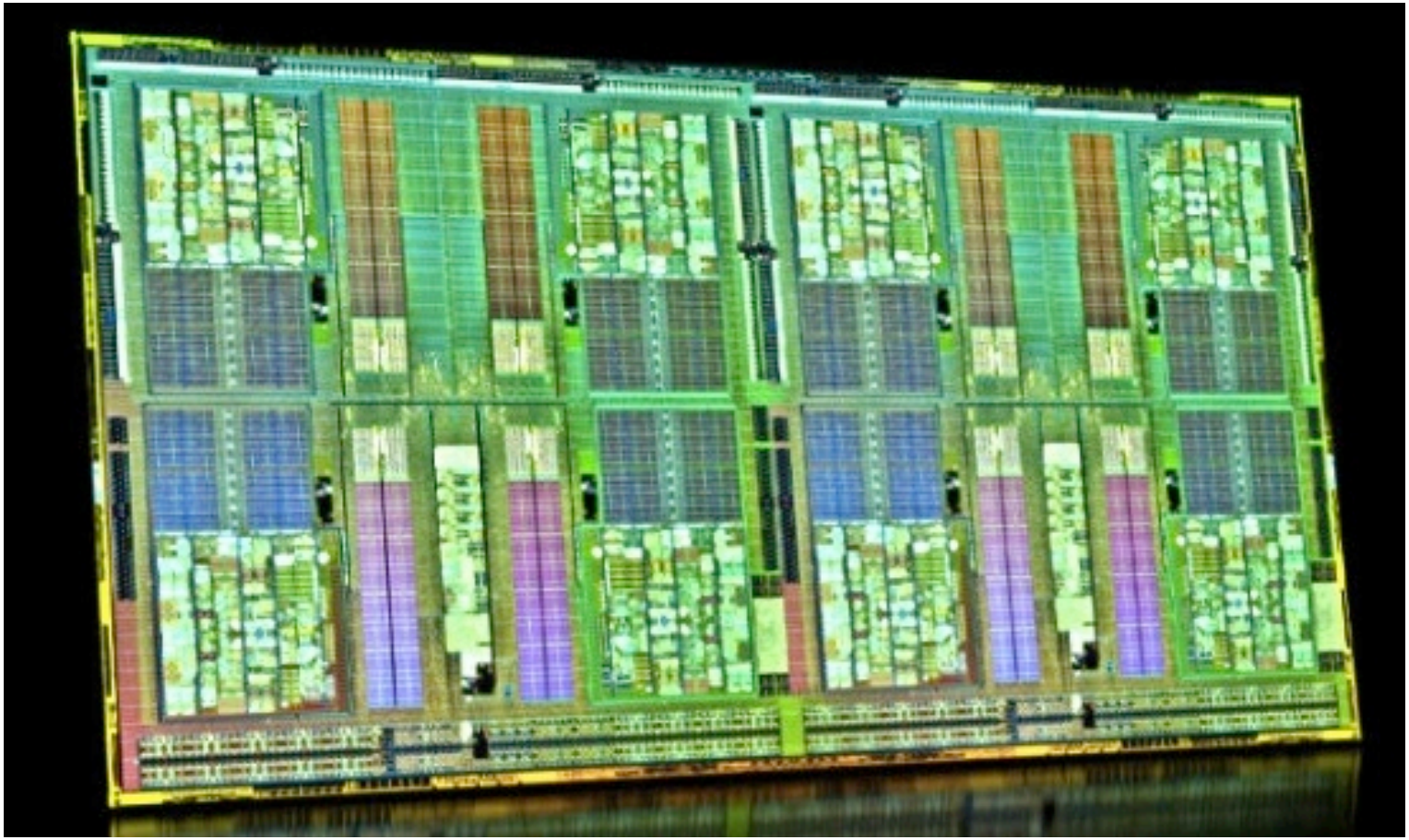
- They provide near linear performance improvement with complexity and power
- Two smaller processor cores, instead of a large monolithic processor core, can potentially provide 70-80% more performance, as compared to only 40% from a large monolithic core

Today Multiprocessors

- Multiprocessor SoC Architecture (MPSoC):
  - *Multi-core*: on a same chip, more instances of a same processor (e.g., Dual-core, Quad-core, etc)
  - *Many-core*: on a same chip, more instances of different processors

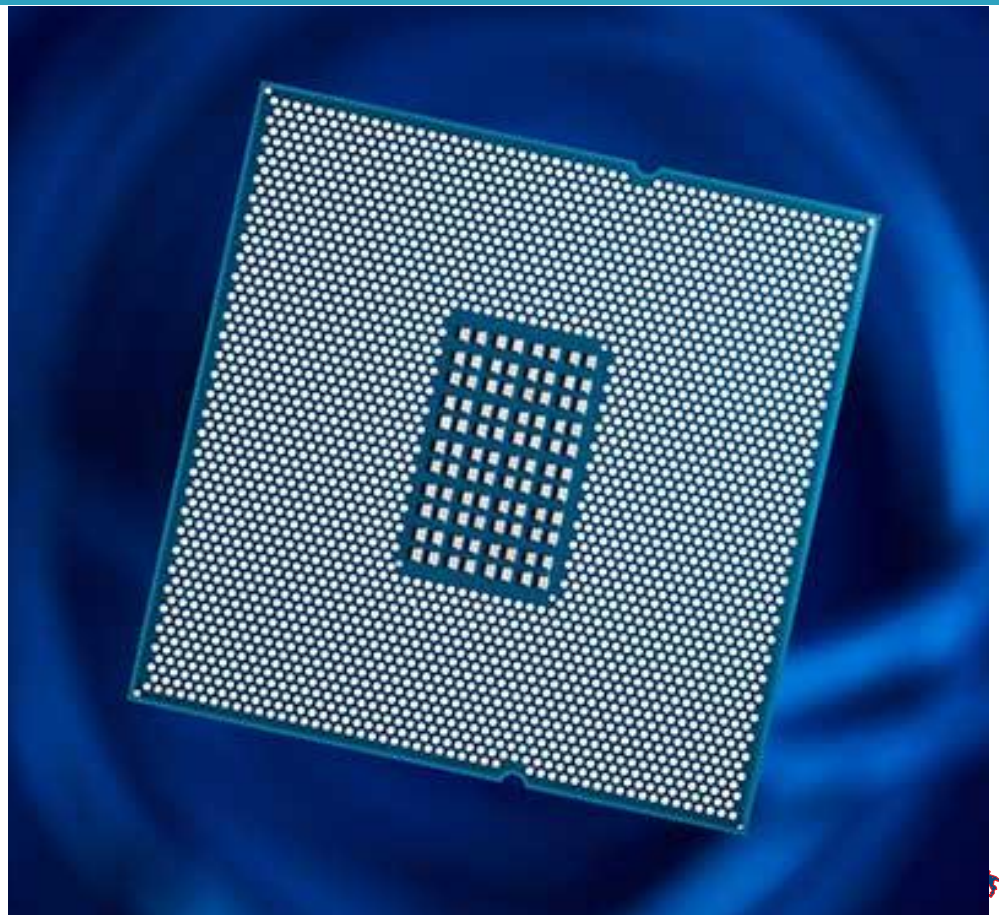
Examples of MPSoC: CPU AMD Opteron

16 cores x86
@ 2.6 GHz



Examples of MPSoC: Qualcomm Centriq 2400

48-cores built on
10nm FinFET
process technology



Implementation

- System-On-Rack
- System-On-Board
- System-In-Package (SIP)
- System-On-Chip (SoC)
- MPSoC
- System-On-FPGA

FPGA

- An FPGA (Field-Programmable Gate Array) is an integrated circuit designed to be configured or programmed by the customer after manufacturing once it is in a circuit (in-the-field) to become almost any kind of digital circuit or system, thus enabling the possibility of updating or changing system's functionalities

FPGA

FPGA

- FPGA can have its internal configuration (i.e., the performed operations) set by software or, as it is termed, by *firmware*

FPGA today

- Several today FPGAs:
  - include complete blocks of memory elements, DSPs and embedded processors
  - support *dynamic partial reconfiguration*

Dynamic Partial Reconfiguration

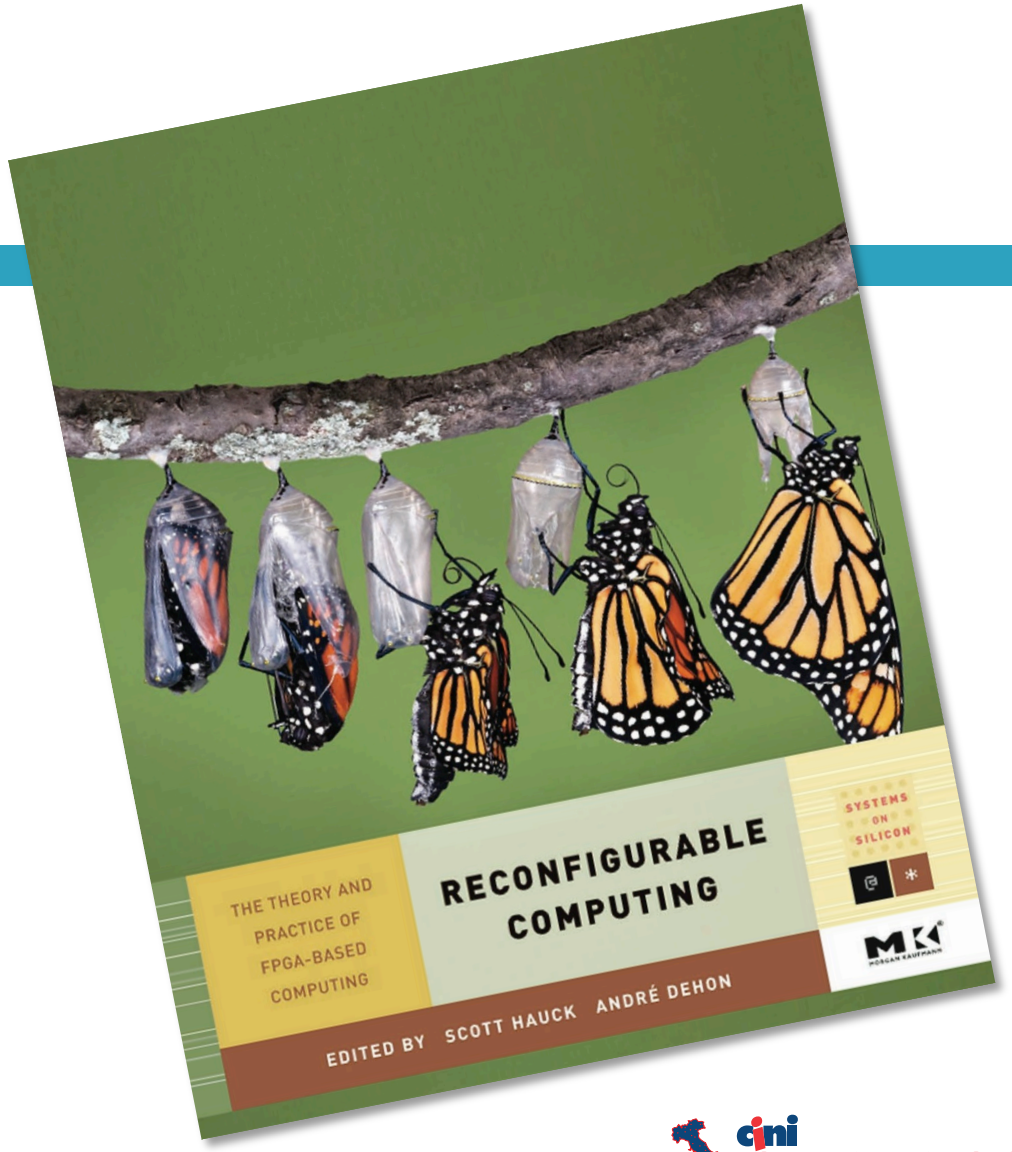
- Dynamic partial reconfiguration (DPR) extends the inherent flexibility of the FPGA by allowing specific regions of the FPGA to be reprogrammed with new functionality while applications continue to run in the remainder of the device.

ASIC vs. FPGA

- ASIC and FPGAs have different value propositions, and they must be carefully evaluated before choosing any one over the other.
- FPGAs are a compelling proposition for almost any type of design.

Further readings

- Students interested in making a reference to a textbook on the arguments covered in this lecture can refer, for instance, to:



Outline

- The concept of System
- System Taxonomies
  - Information coding
  - Memory capability
  - Functionality
  - Implementation
- Data vs Control vs Timing
- Synchronous Sequential Circuits

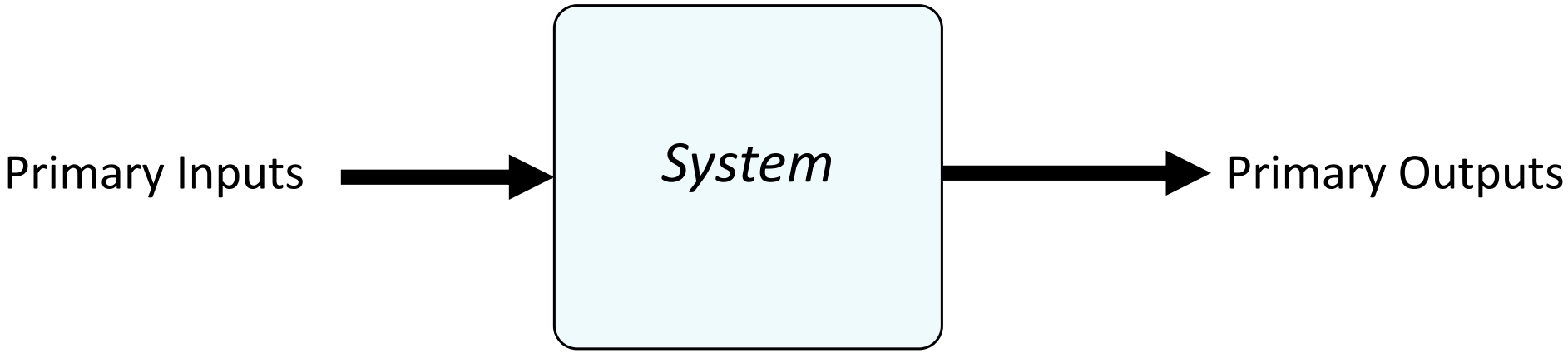
Data vs Control vs Timing

- When dealing with hardware systems, regardless their complexity, a clear distinction must always be done among *Data*, *Control*, and *Timing.*
- The distinction must concern:
  - Input/Output Signals
  - Functional units

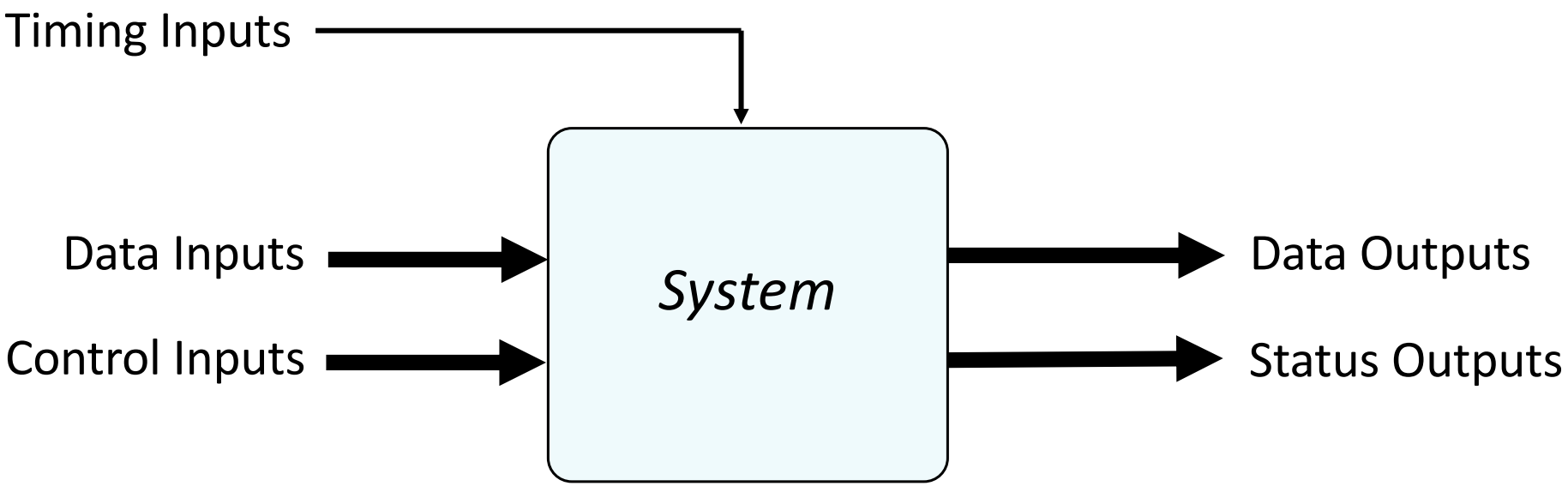
Data vs Control vs Timing

- When dealing with hardware systems, regardless their complexity, a clear distinction must always be done among *Data*, *Control*, and *Timing.*
- The distinction must concern:
  - Input/Output Signals
  - Functional units

System I/O signals



System I/O signals



Timing signals

- Are present in Stateful (Sequential) systems, only.
- Are typically period signals, usually referred to as *clock signals*, and used to force the system to change its *state*

Timing signals

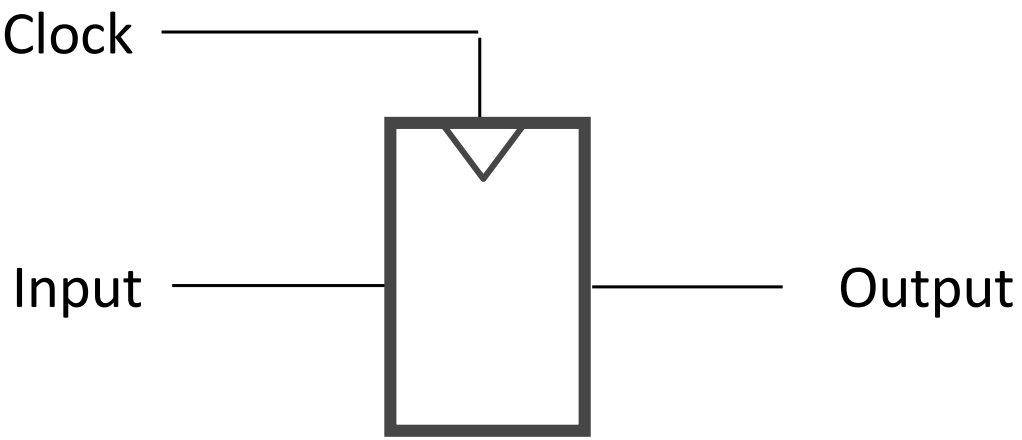
- They synchronize the overall behavior of the network by:
  - *sampling* Data and Control inputs
  - *updating* Data and Control outputs

Timing signals

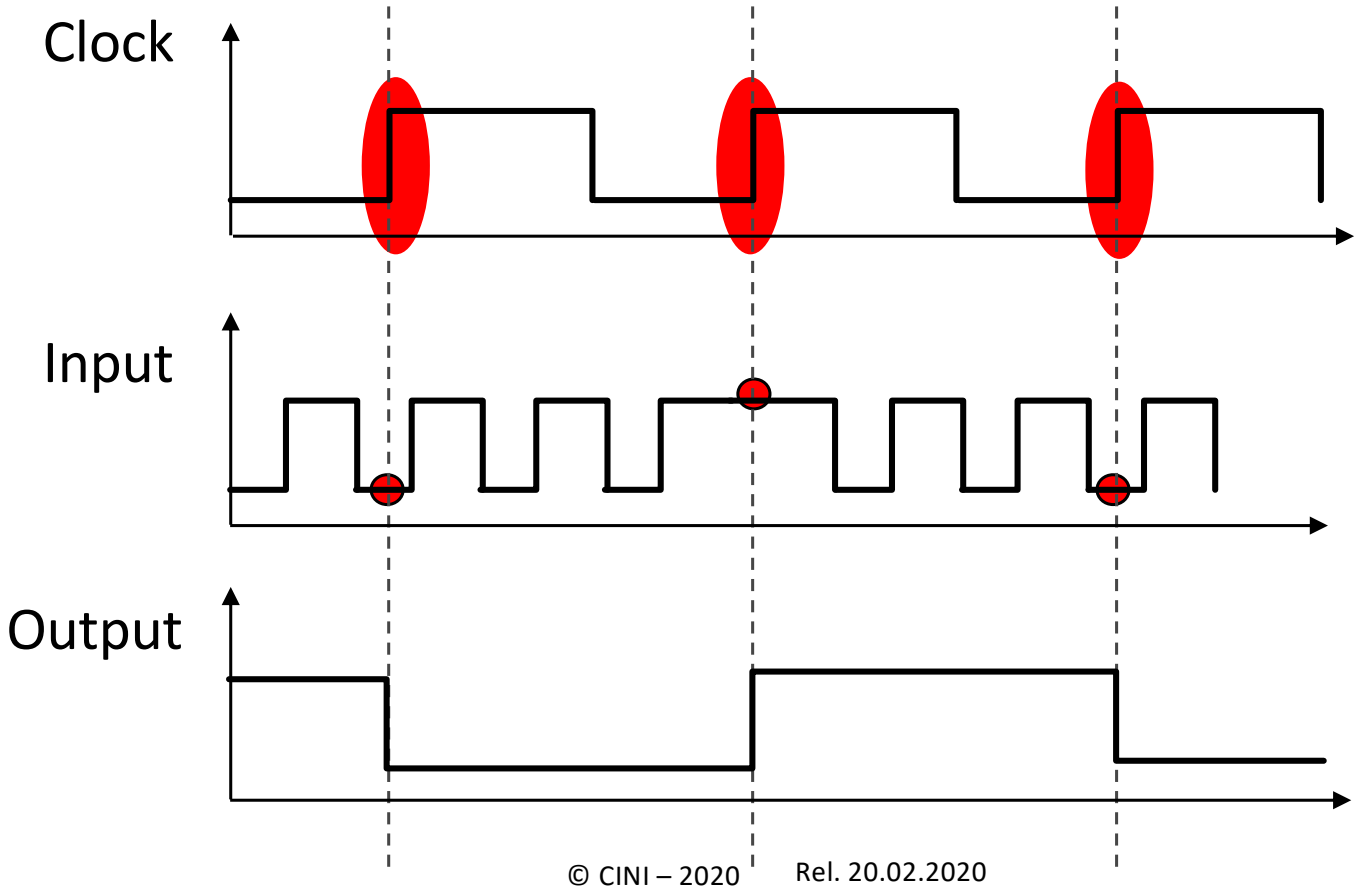
- Sampling and updating are usually triggered by the clock signal's edge:
  - raising edge → *positive edge triggered*
  - falling edge → *negative edge triggered*

Example

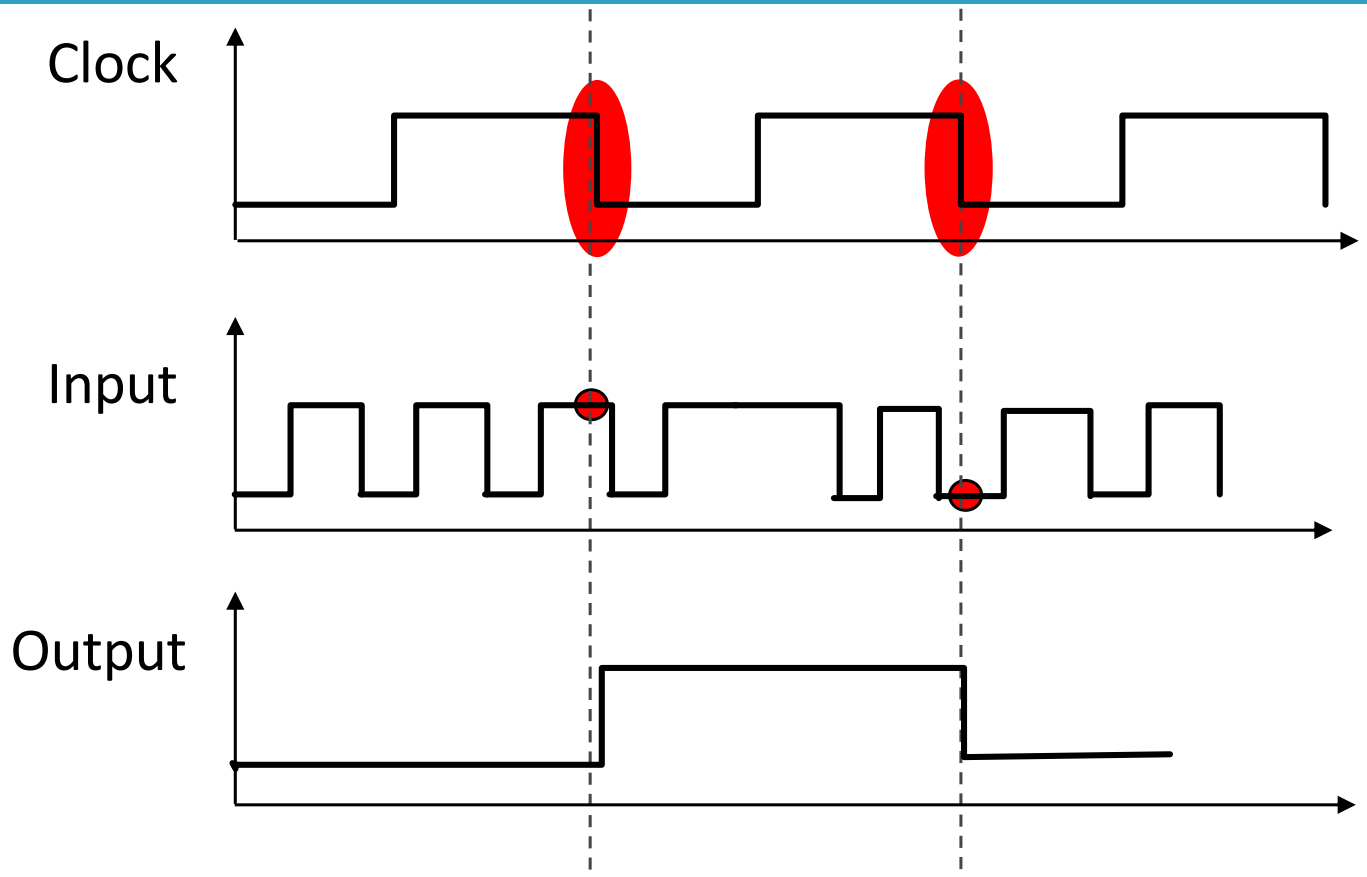
- Assuming a very simple device that stores just 1 bit of information (by the way, usually referred to as a *Flip-Flop*):



Positive edge triggered



Negative edge triggered



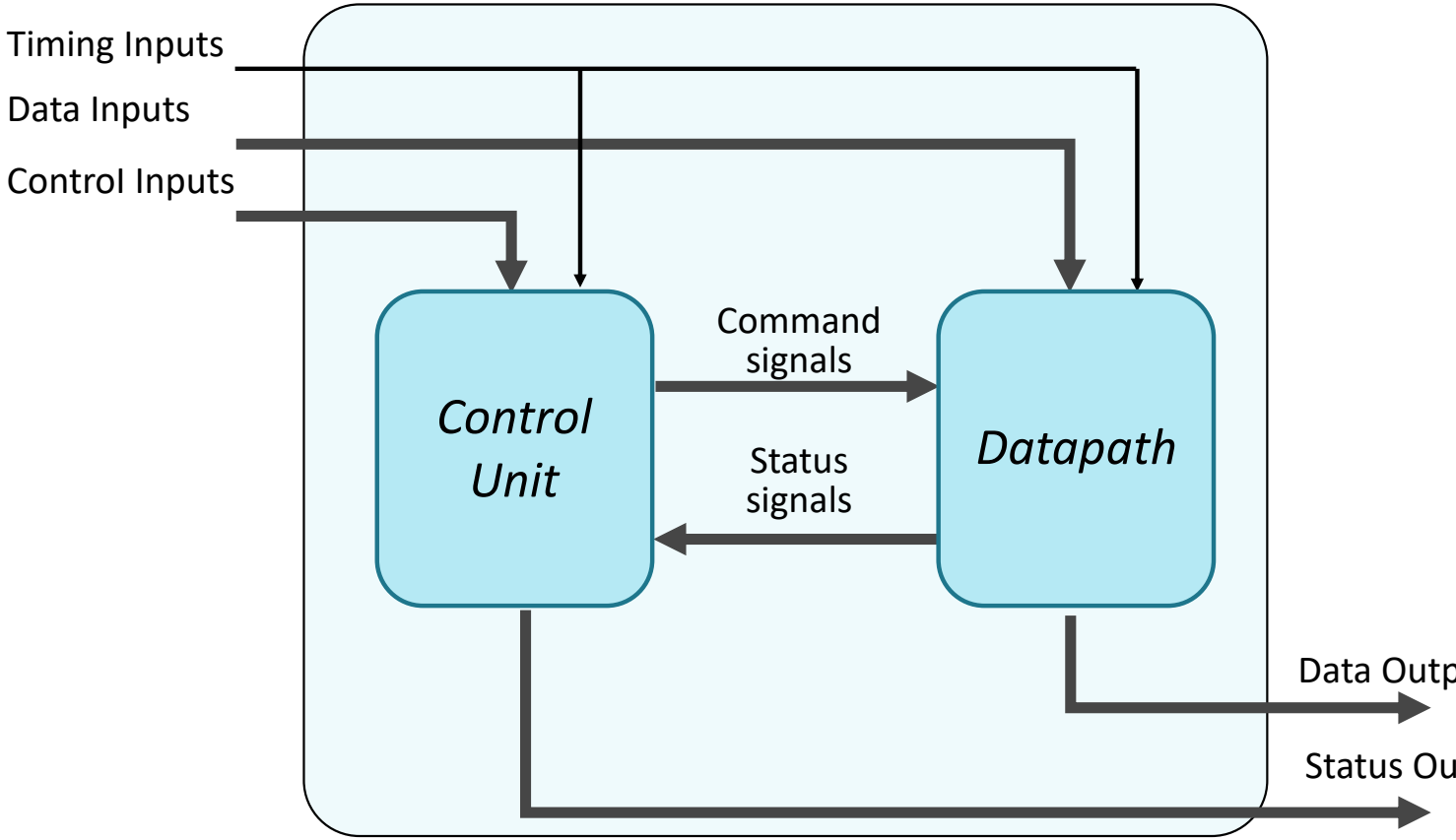
Data vs Control vs Timing

- When dealing with hardware systems, regardless their complexity, a clear distinction must always be done among *Data*, *Control*, and *Timing.*
- The distinction must concern:
  - Input/Output Signals
  - Functional units

Datapath & Control Unit

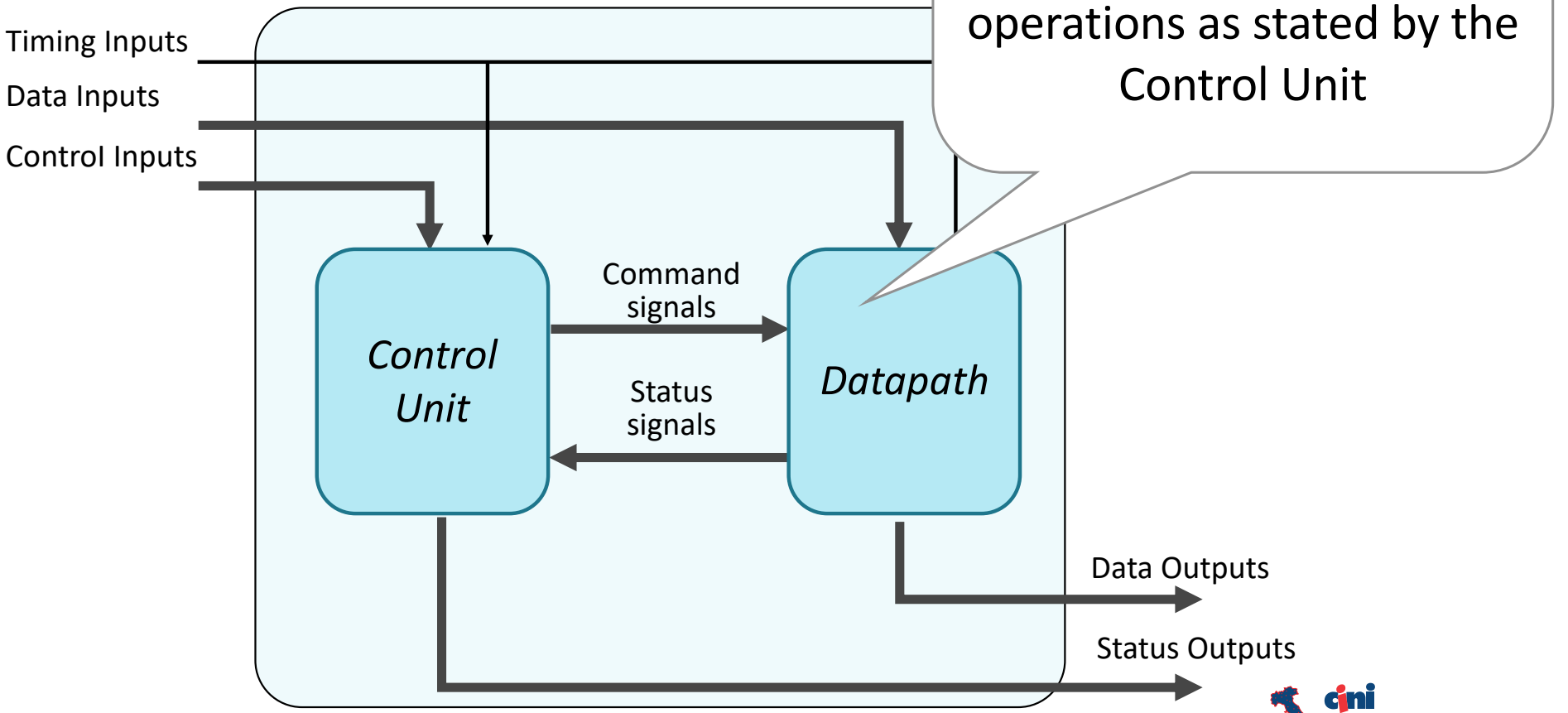
- Each system, regardless its complexity, can always be internally modeled as follows:

Datapath & Control Unit



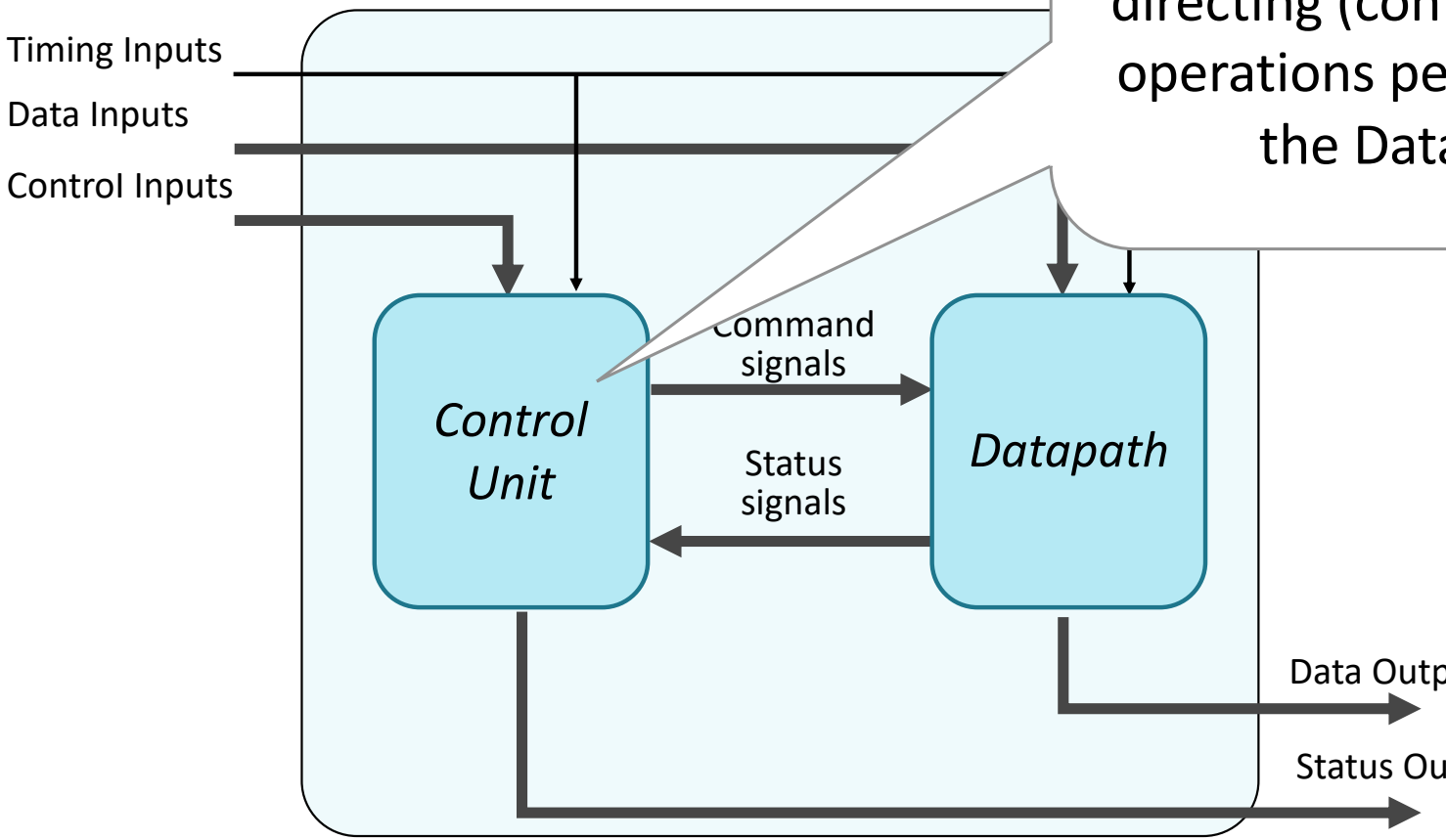
Datapath

A set of functional units charged of carrying out data processing operations as stated by the Control Unit



Control Unit

A collection of components charged of directing (controlling) the operations performed by the Datapath



Outline

- The concept of System
- System Taxonomies
  - Information coding
  - Memory capability
  - Functionality
  - Implementation
- Data vs Control vs Timing
- Synchronous Sequential Circuits

The concept of state

- A sequential circuit must be able to “remember”, or “store”, some information items related to the values the PIs have got.
- Such a storing capability is accomplished in terms of *internal states*: in any instant, the circuits is in a well defined “state”, univocally represented by the values got by the set of its *internal state variables*.

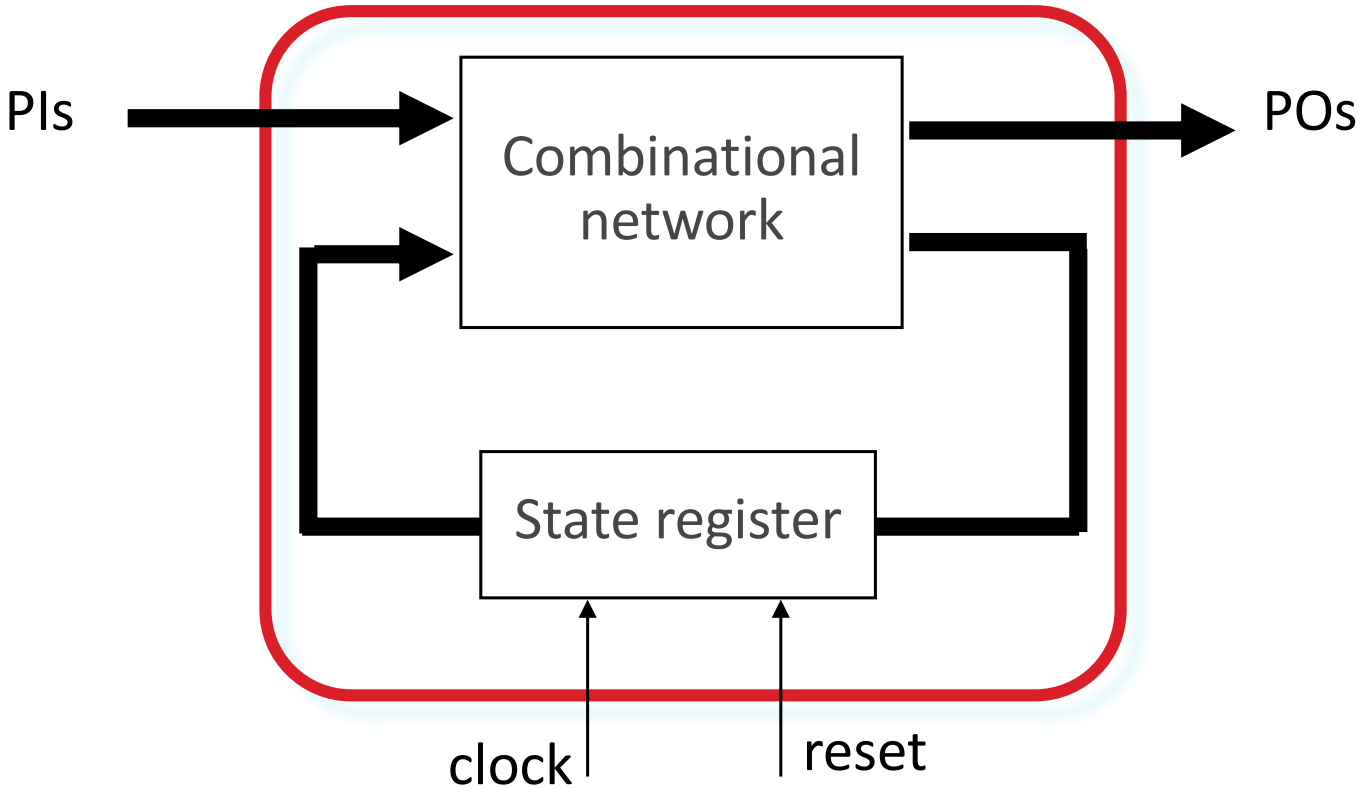
State variables

- Each state variable has to be “stored”
- Or, reversely, each Flip-Flops stores a state variable
- A sequential network has as many state variables as Flip-Flops
- Each Flip-Flop can get 2 possible values
- A network with n Flip-Flops is characterized by 2^n possible states.

State register

- The set of Flip-Flops is often referred to as *State Register*, since it stores the network state variables.

Synchronous Sequential Circuits



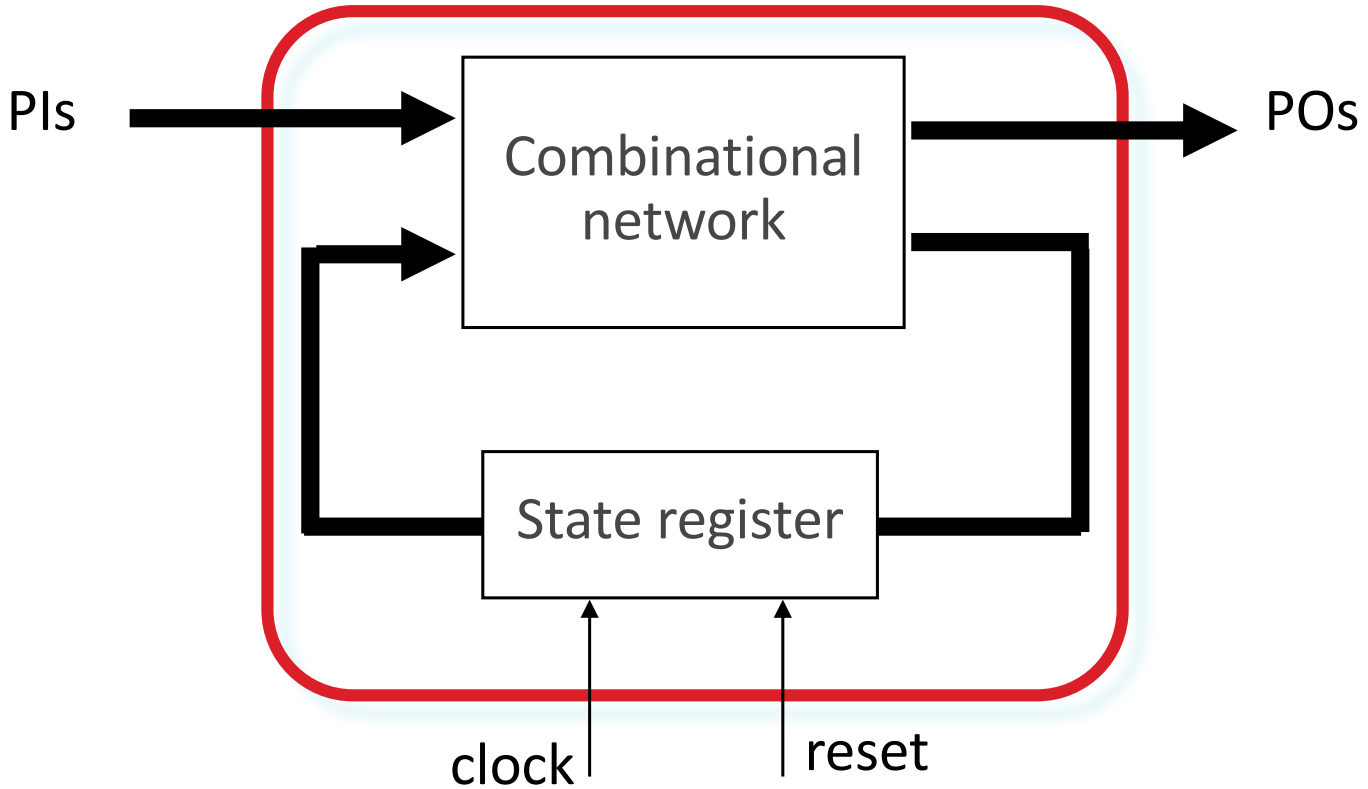
Finite State Machines

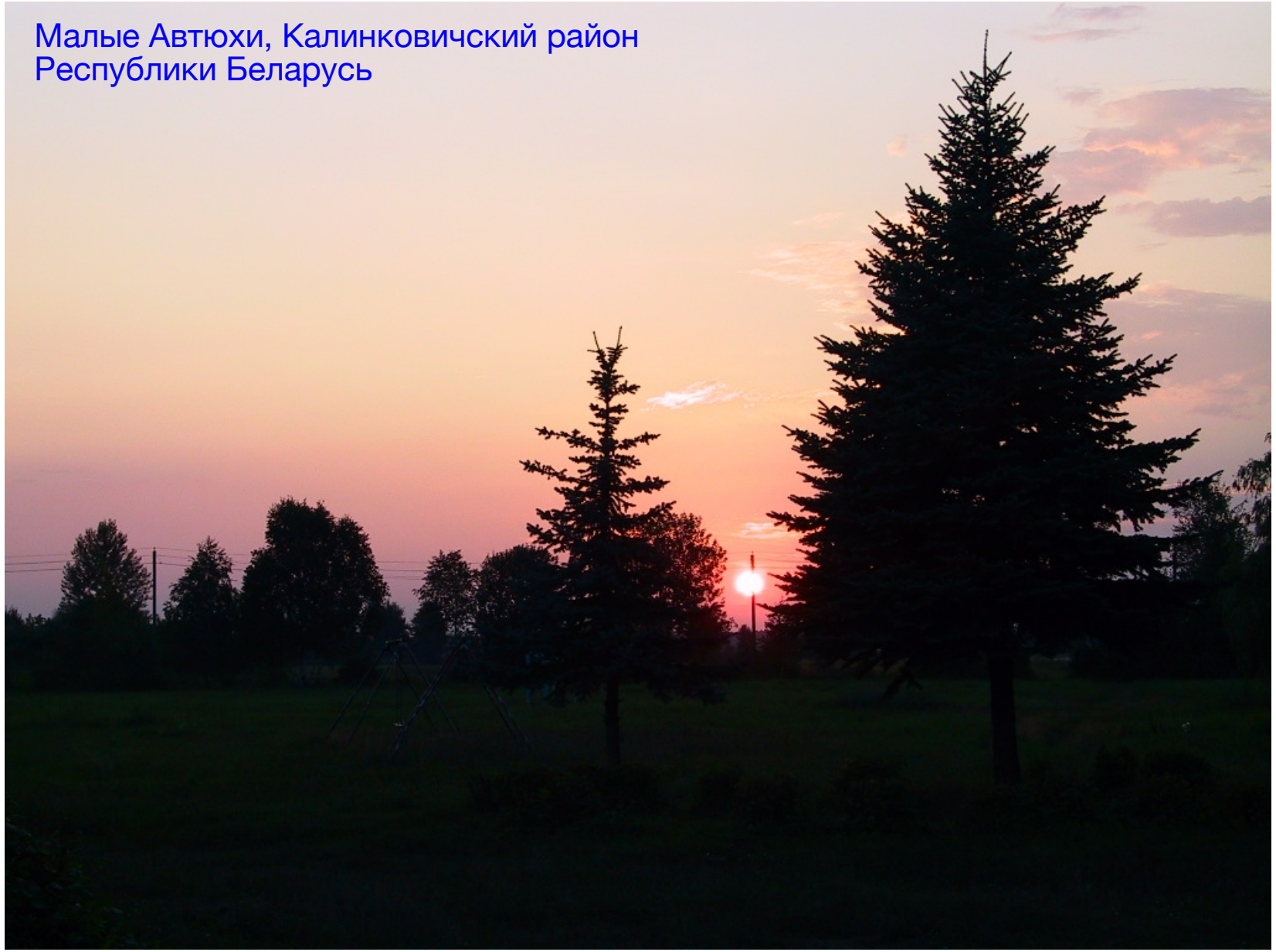
- Synchronous networks, being characterized by a finite # of flip-flops, and thus of states, are very often referred to as *Finite State Machines* (FSMs).

Caveat

- Any FSM, regardless its complexity, MUST have:
  - a particular control input signal, named *reset signal* (or simply *reset*) characterized by the highest priority
  - a particular state, named *reset state*, in which the network moves whenever the reset signal is asserted.

Sequential circuit





Малые Автюхи, Калинковичский район
Республики Беларусь

Paolo PRINETTO

Director
CINI Cybersecurity
National Laboratory

Paolo.Prinetto@polito.it

Mob. +39 335 227529

CYBER CHALLENGE
CyberChallenge.IT

cini Cybersecurity National Lab

https://cybersecnatlab.it