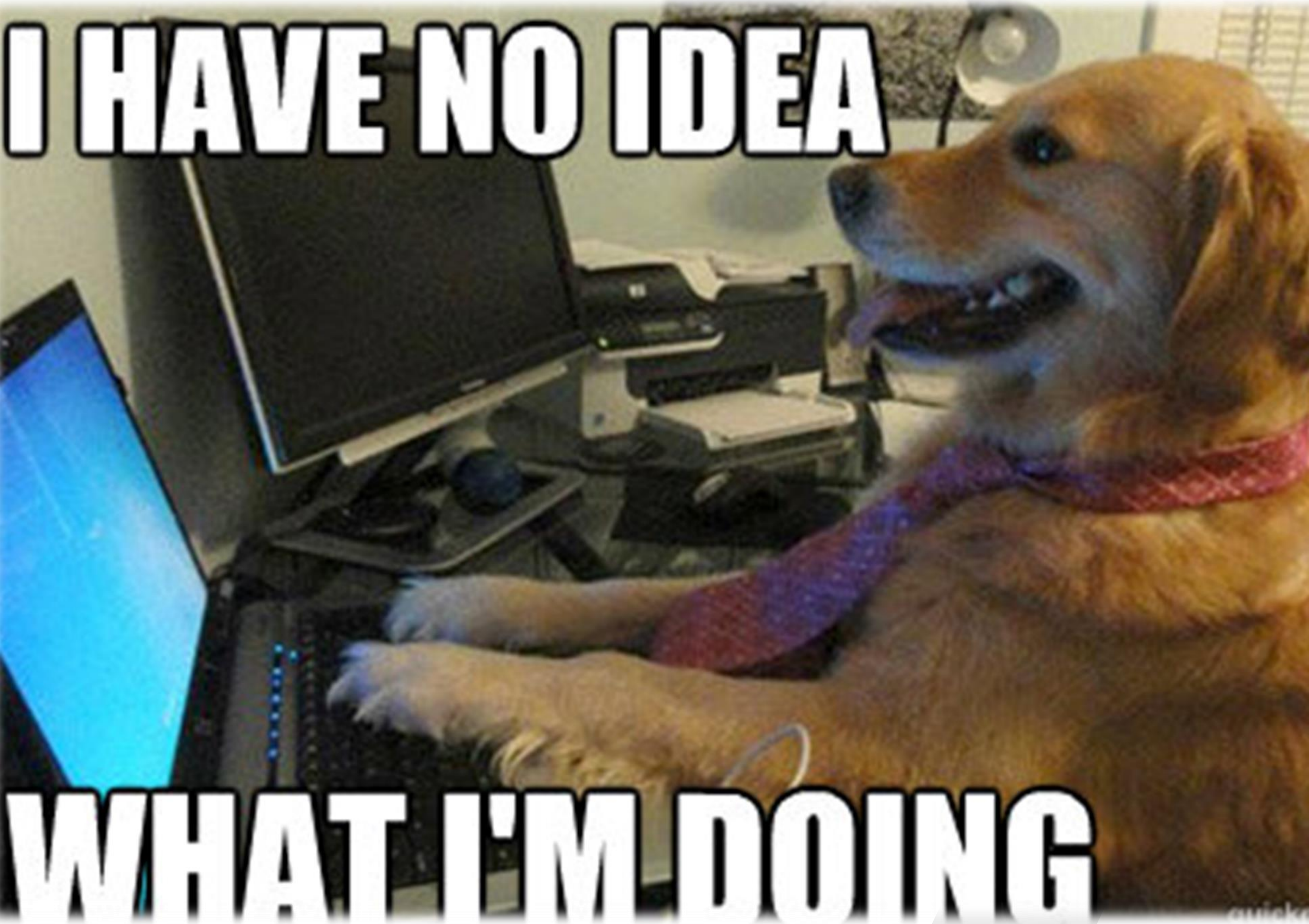
Industrial processors

Paolo Burgio
paolo.burgio@unimore.it

UNIVERSITAS STUDIORUM MUTINENSIS ET REGIENSIS 1175
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA
High Performance Real Time Lab


I HAVE NO IDEA
WHAT I'M DOING

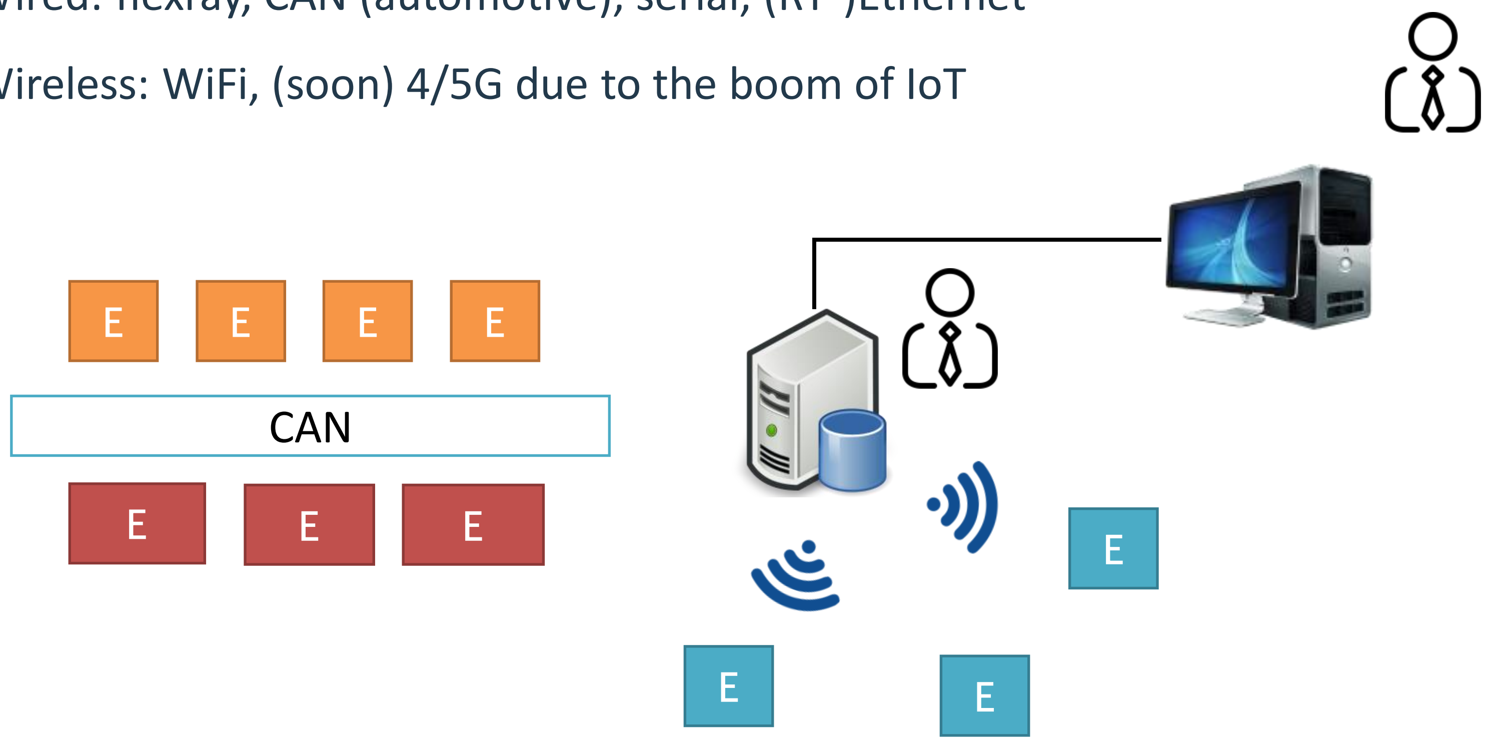
Industrial computing continuum

A modern design: in the past might have been different!

- Embedded (*edge*) devices for plant control
- Centralized aggregator

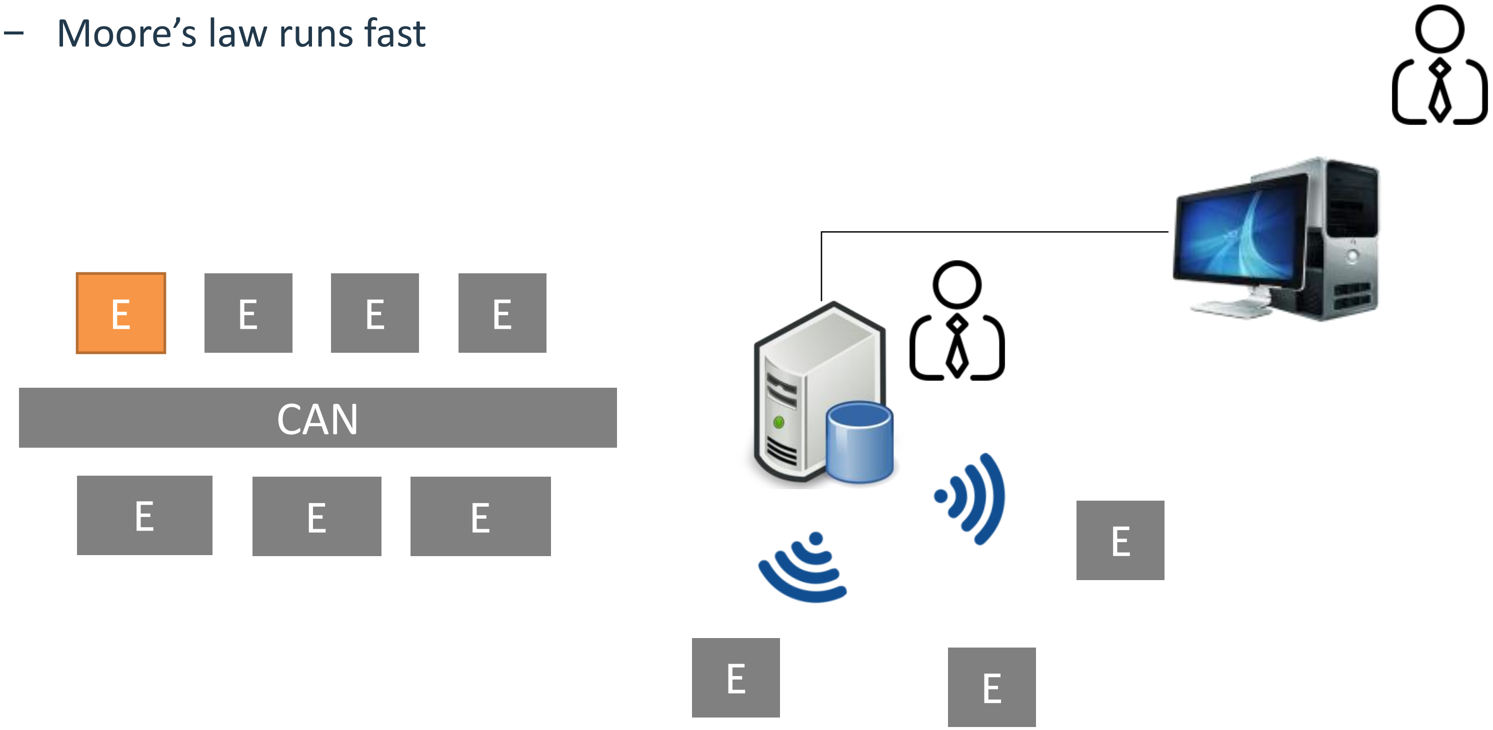
Connectivity *via* industrial standards

- Wired: flexray, CAN (automotive), serial, (RT-)Ethernet
- Wireless: WiFi, (soon) 4/5G due to the boom of IoT



Requirements for industrial edge devices

- Low cost and form factor might be key feature
  - …we can trade performance for that
  - Relibility, dependability, safety, certifications….
- Might be costly to update your plant to a new generation of processors
  - Several companies rely to old technologies (also, software tools!!)
  - Electrical/electronic/informatic engineers in 80s/90s
  - Moore's law runs fast



Families of edge processors

As opposite to GP/desktop systems, where more or less we know “who won”…

- Micro Controller Units – MCUs
- Digital Signal Processors - DSPs
- Micro Processor Units – MPUs
- Programmable Logic Controllers / PLCs

Now

More recently, heterogeneous architectures

- Multi-core host + accelerator
- Many-core processor, such as GPGPUs (but not only….)
- Field-Programmable Gate Arrays – FPGAs

….soon

Micro Controller Units

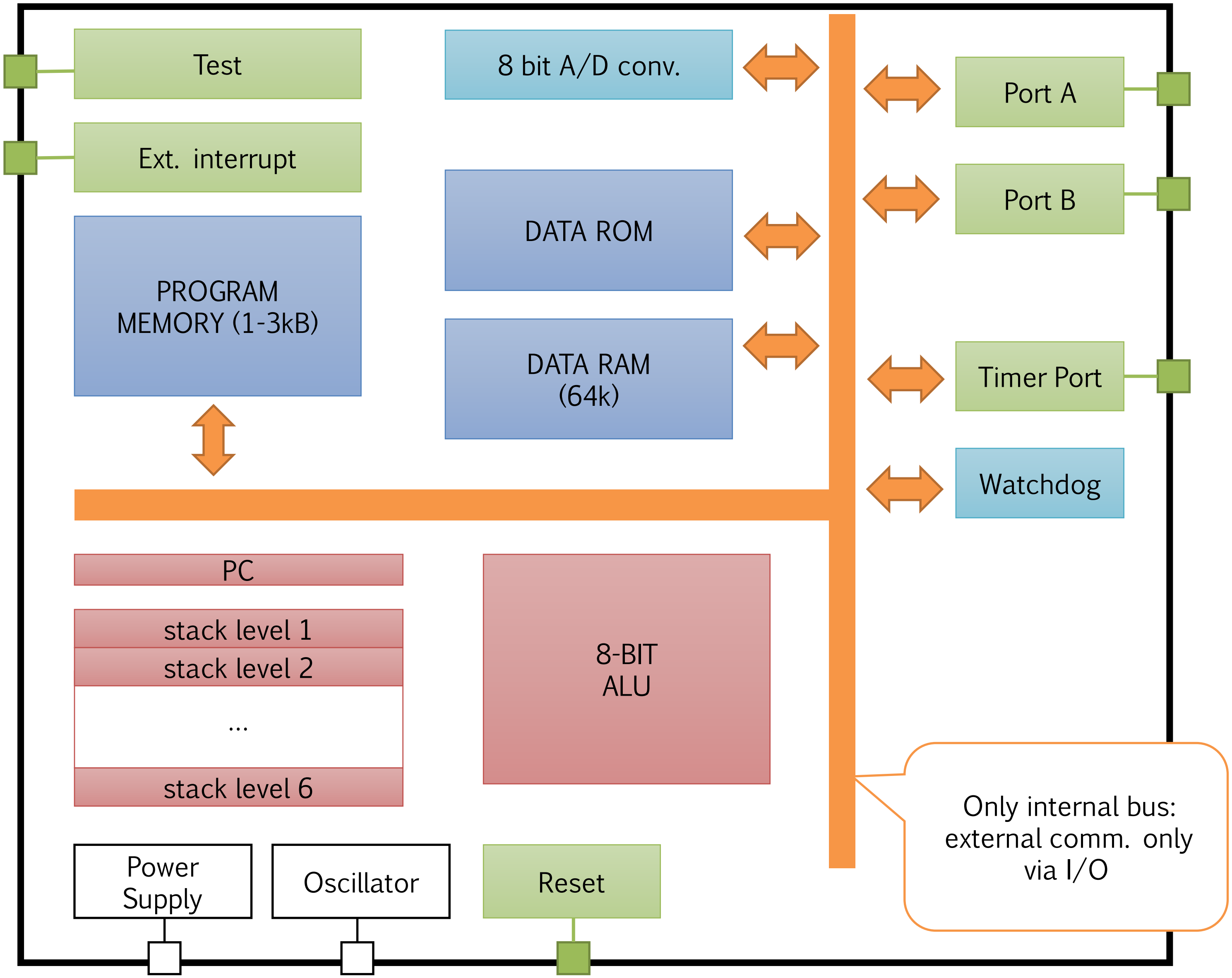
Lowest end that we will see

- 8, 16, 32 bit processors, limited memory
- Designed for I/O interaction, can poorly do more
- Also, programmability might suffer (ASM)

We will see (and play with)

- ST Microelectronic’s SC6
- Arduino’s first family
- Expressif’s ESP8266
- Raspberry PI

STM’s ST6


Test
Ext. interrupt
PROGRAM MEMORY (1-3kB)
8 bit A/D conv.
DATA ROM
DATA RAM (64k)
Port A
Port B
Timer Port
Watchdog
PC
stack level 1
stack level 2
...
stack level 6
8-BIT ALU
Power Supply
Oscillator
Reset
Only internal bus: external comm. only via I/O

Watchdog

Watchdog

Checks whether a ~~program~~ ~~process~~ processor is stuck, e.g., in deadlock, or infinite loop

In MCUs/single core, single thread machines it is a problem

- A watchdog circuit is basically a *x*-bit counter
- It **must** be **manually** reset every $\frac{2^x}{y \cdot 10^6}$ seconds (*y* = clk in Mhz) by software
- Else, it "takes care of the situation"
  - Typically => full reset

Modern system does not actually block the full machine (SW managed) nor resets it

- Multi-processing
- Preemption (we'll see..)
- OS-level controls on processes

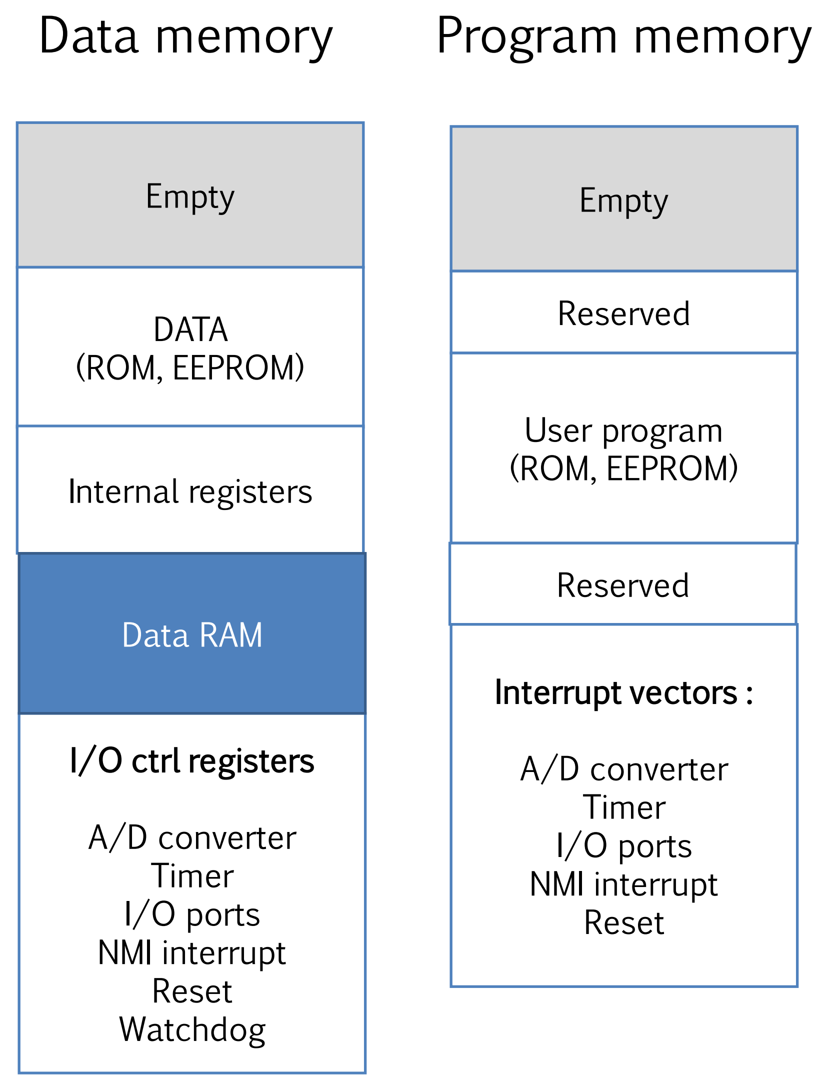
Memory space

This is what programmers "see"

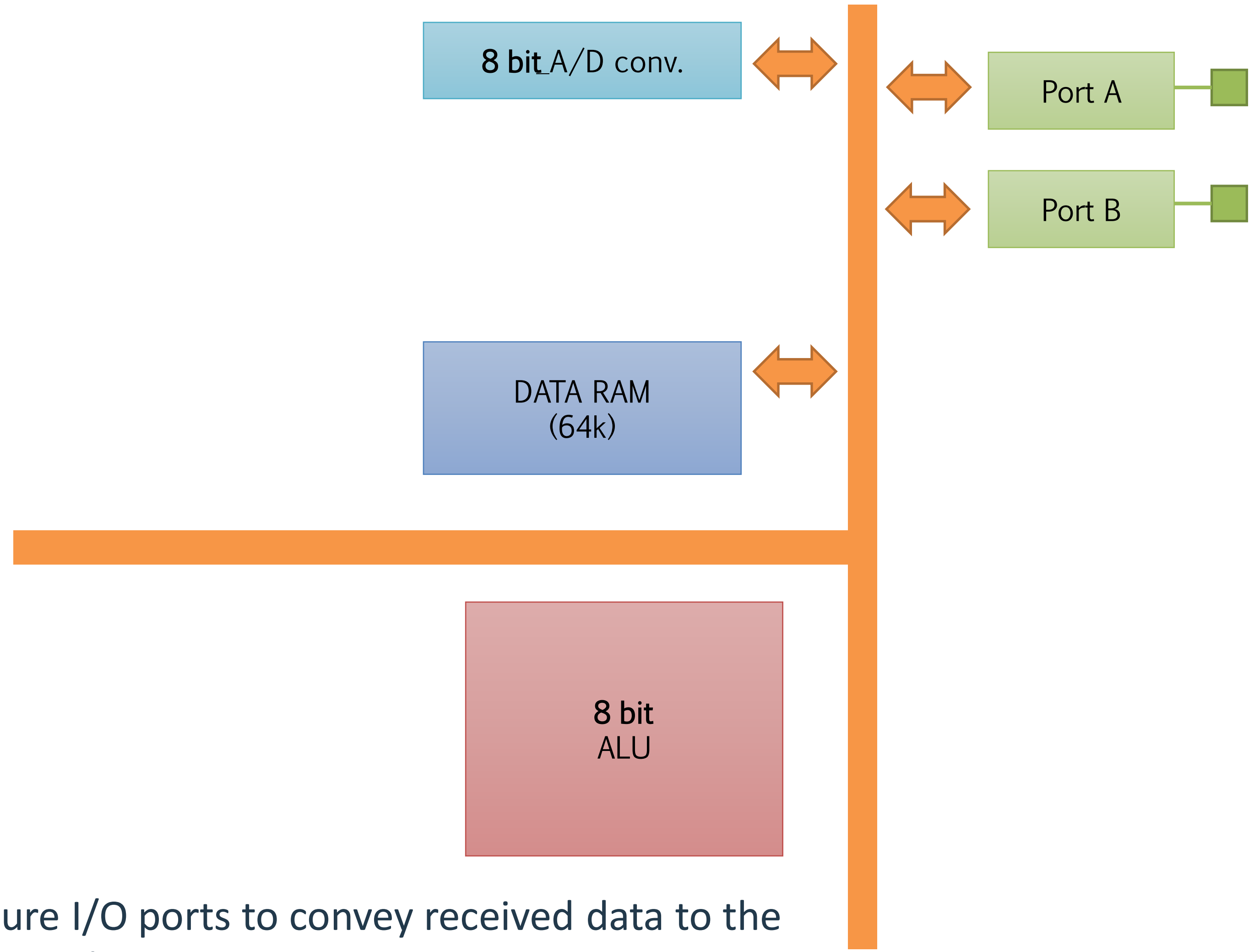
- Non-physical, here, abstracts the HW blocks
- != virtualized ☺

Program data (variables, ..) are in data RAM

- Stack, heap (..?)
- ...if we could use high-level languages! (e.g., C)
- Might be necessary to work in ASM



A/D converter



Configure I/O ports to convey received data to the A/D converter

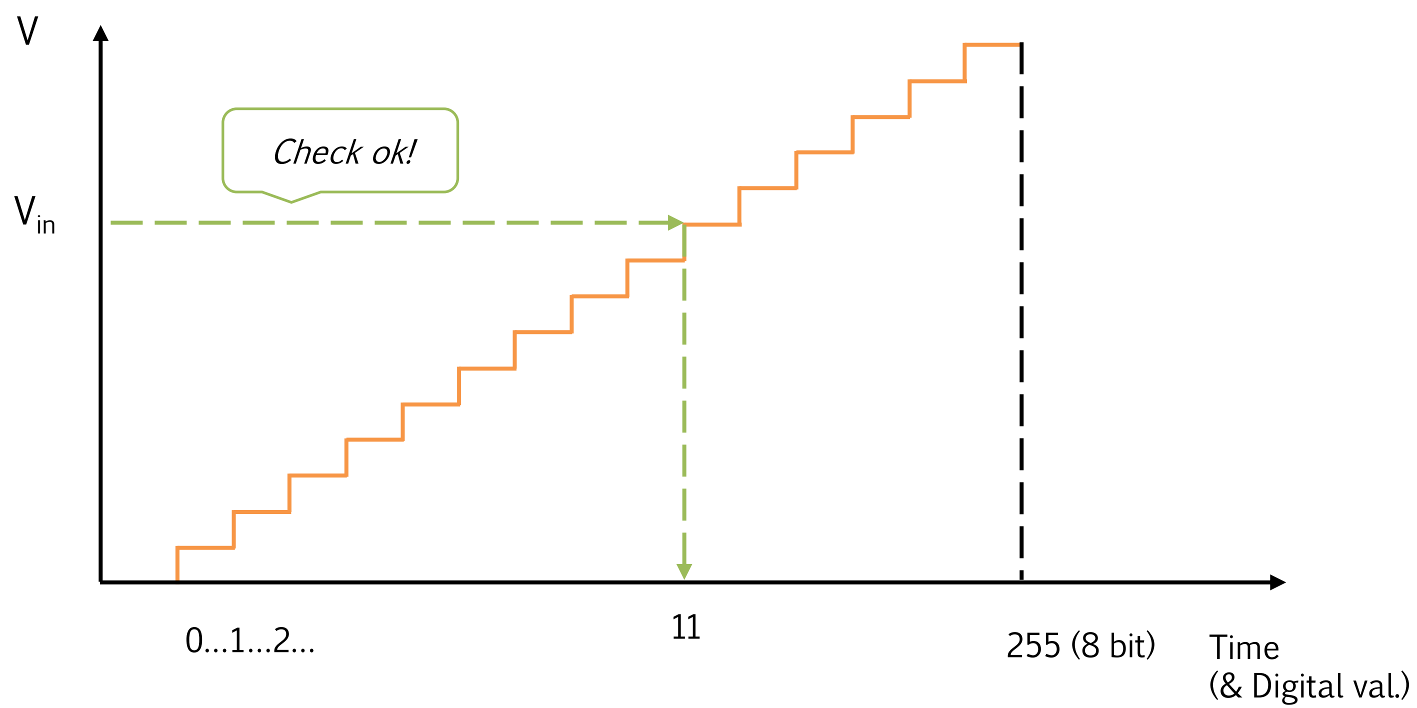
- Use memory-mapped registers

A/D conversion

8 bit A/D conv.

Using the internal wave generator of A/D module, our program

- Generate a signal with increasing V_{in} (y-axis)
- Compare (in HW) it with the V received by the analog I/O port
- When equals, assign the corresponding digital value (x-axis)



D/A conversion

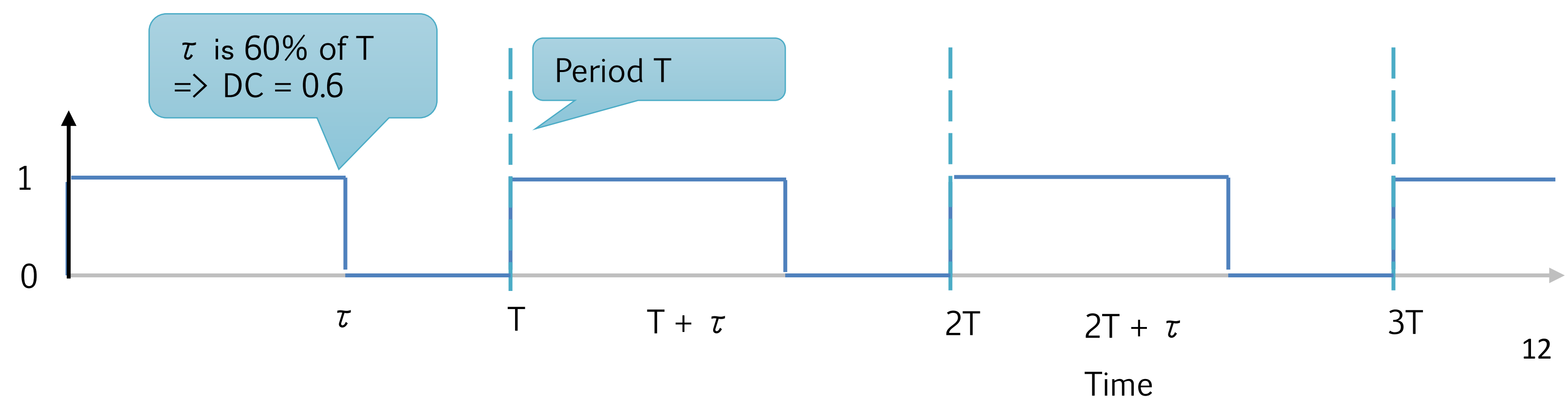
Generate a tension corresponding to a digital value stored in a register

- Not easy! Use Pulse-Width Modulation (PWM)
  - (Almost) fully implementable in SW!

DC = τ / T

How it works

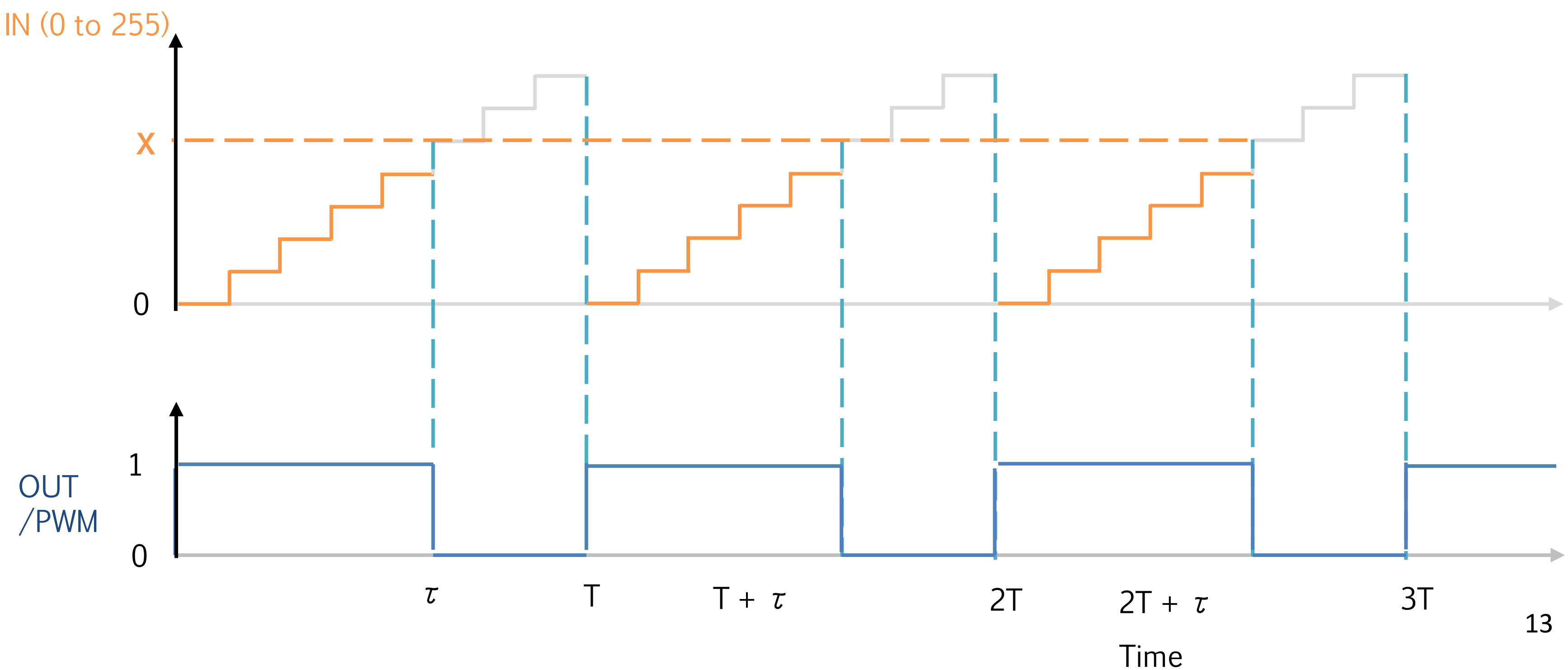
1. Generate a periodic signal of amplitude 1 whose **duty cycle** is proportional to the digital value we want to convert
2. Give it to a low-pass filter, to average (such as Resistor-Capacitor RC) circuit)
3. ..and enjoy your analog signal! ☺



PWM – step 1

We need

- A register that counts from 0 to 255 (8-bit)
- An output port (bit) set to '1' and becoming '0' when input value matches the one of the register

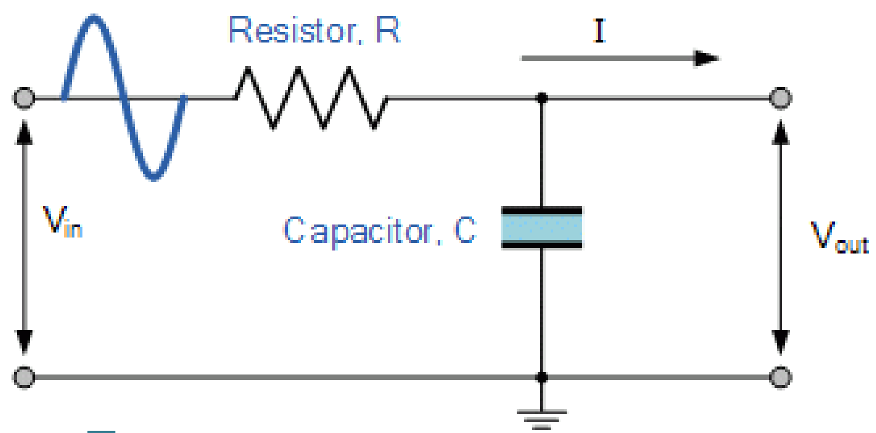


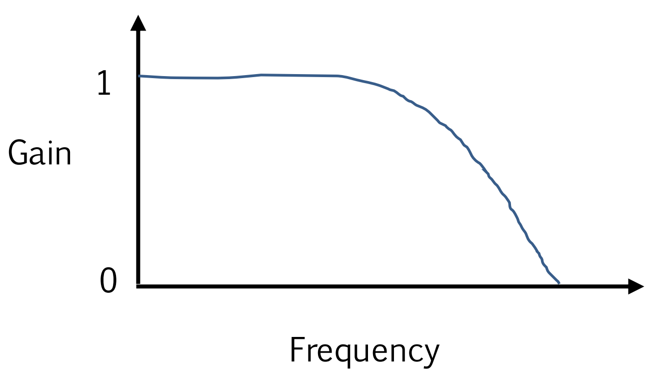
RC low-pass filter

Simple electric circuit with a Capacitor and a Resistance

- “Averages” the IN value

$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2}$$



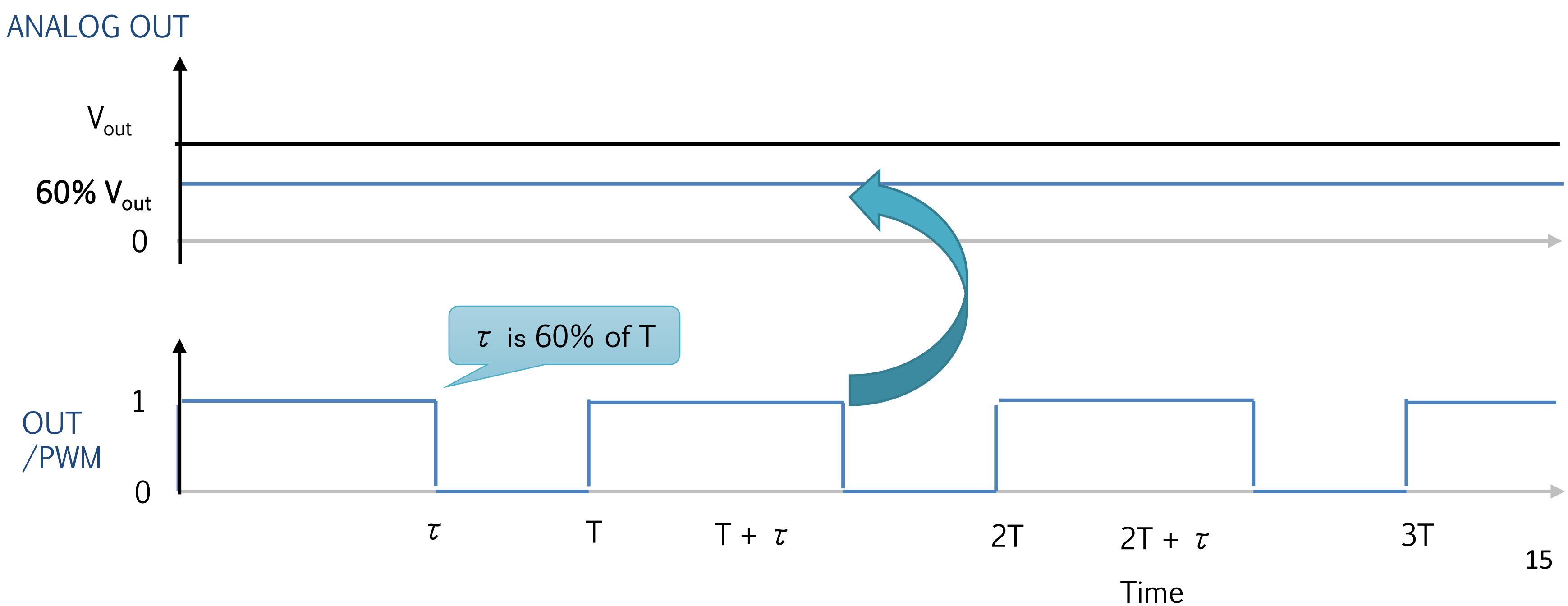


PWM – step 2

Now, compute the average for every T

- Using a low-pass filter
- Plug it to output port
- *Et voilà*

Extremely useful in engine controls



Digital Signal Processors

A family of MCUs explicitly designed for **digital** signal processing

- We have A/D converters..

Example: **Digital** Finite Impulse Response filter (FIR)

- Computes the weighted sum of *N* timing samples of a discrete signal *x*

$$y_n = a_0 \cdot x_n + a_1 \cdot x_{n-1} \ldots a_N \cdot x_{n-N}$$

$$y_n = \sum_{i=0}^{N-1} x_{n-i} \cdot a_i$$

- Typically, discrete time series
- N is also called *order* and specifies “how much in the past” we go

How to implement it?

$$y_n = a_0 \cdot x_n + a_1 \cdot x_{n-1} \dots a_N \cdot x_{n-N}$$

- (Assume time series…)
- We can't accumulate multiplied values, as every sample is multiplied with a different coefficient a
- We can keep a buffer in memory of the last N-1 samples, and slide on it (aka: sliding window)

Compute multiplications in parallel, then sum

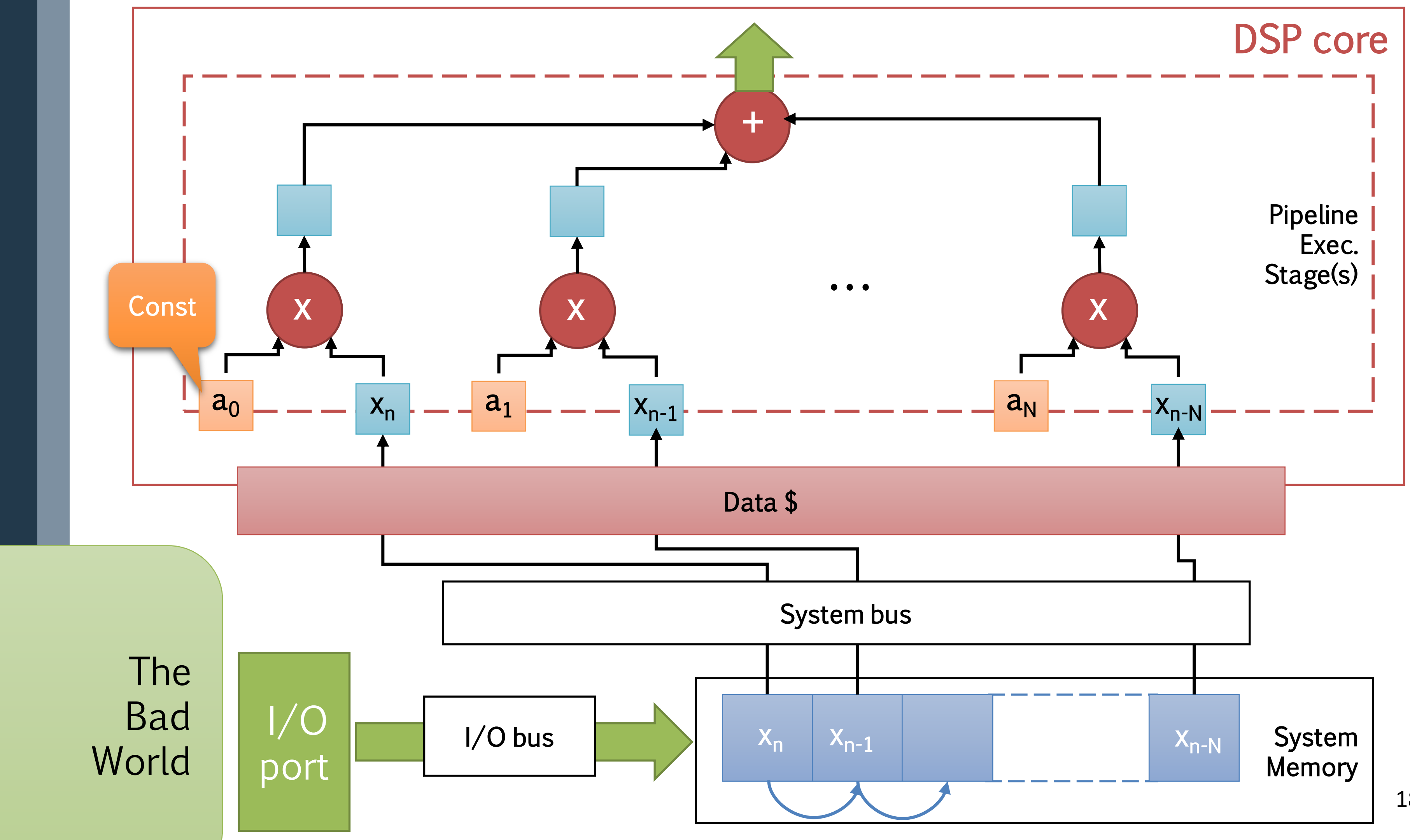
- Parallel programming patterns: reduction, data parallelism
- The principle behind GPUs!

Assuming that

- We have a N-wide execution pipeline
- Our memory bandwidth can read N samples in parallel
- Else, we create stages

Data-parallel FIR

$$y_n = a_0 \cdot x_n + a_1 \cdot x_{n-1} \dots a_N \cdot x_{n-N}$$



DSP typical ingredients

Similar to standard MCUs..but..

- Wide and simple pipeline for data parallelism
  - Multipliers, accumulators
- Wide data bus to avoid staging
  - Similar problem also for exec pipeline, but it’s it’s much more exacerbated in busses!
- Fast I/O and A/D, D/A conversion
  - Typically, ~~interacts with the world~~ Cyber-Physical System CPS

Often, problem-dependent architecture

- Clock frequency depends on how fast is input data sampling
- Register size (16, 32-bit) and data type (fixed, double) depends on the signal we want to process

A recent example: Texas Instrument’s Keystone II

Limits of MCUs

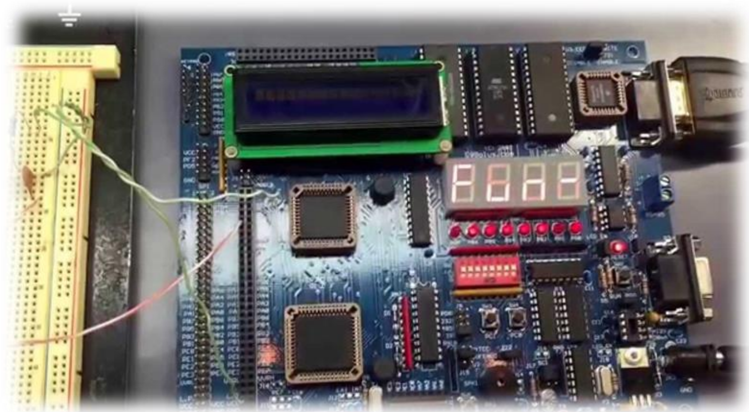
As complexity of applications grows….

- 8-bits might not be enough to capture wide data dynamics
- Limited memory size
- Poor programmability
  - In the worst case, assembly
- Poor extendability
- Industry can stand more consuming, costly and bulky edge computers
  - Traded for performance

Micro Processor Units - MPUs

- Embed "standard" processors
  - 16… 32… 64 bit
  - Also, Intel!
  - Advanced RISC Machines (ARM) is **the big guy** here
- Desktop-like memories and programmability
  - C, C++..
- Rich I/O and connectivity
  - A/D, D/A, watchdog..
  - Support for industrial-grade fieldbusses such as CAN
  - Traditional connectivity (ETH, Wireles..)
- Typically, built to be mounted in racks
- Easily extendable
  - Arduino, Raspberry Pi are very simple, low-cost samples
  - ExpressIF's ESP8266 (NodeMCU) and ESP32

Motorola 68HC11



Programmable Logic Controllers - PLCs

References

Course website

- http://hipert.unimore.it/people/paolob/pub/Industrial_Informatics/index.html

My contacts

- paolo.burgio@unimore.it
- http://hipert.mat.unimore.it/people/paolob/

Resources

- Alessandro Fantechi, «Informatica Industriale», Città Studi Edizioni
- Giacomo Bucci, «Calcolatori elettronici: architettura e organizzazione», McGraw-Hill Education
- A "small blog"
  - http://www.google.com