



# Introduction

Digital Systems M  
Stefano Mattocchia, Matteo Poggi  
Università di Bologna

## Goals

- ✓ Introduction to *embedded systems*
- ✓ Design methodologies based on **high-level languages** (C/C++, Python, more)
- ✓ Applications: **computer vision** and **deep-learning**
- ✓ Several slots for hand-on experimentation
- ✓ Hardware description languages, e.g. VHDL and Verilog, **won't be studied**

# Exams

## ✓ Project

- ✓ designed and deployed on a **real** device
- ✓ grade: up to 20, depending on project **difficulty level** and **quality** of proposed solution
- ✓ examples will be shown during the course
- ✓ groups of **max. 2-3**
- ✓ individual evaluation (see below)

## ✓ Oral exam

- ✓ discussion of the project + questions covering **all of the topics** of the course
- ✓ individual exam
- ✓ grade: up to +10 on the **project grade**

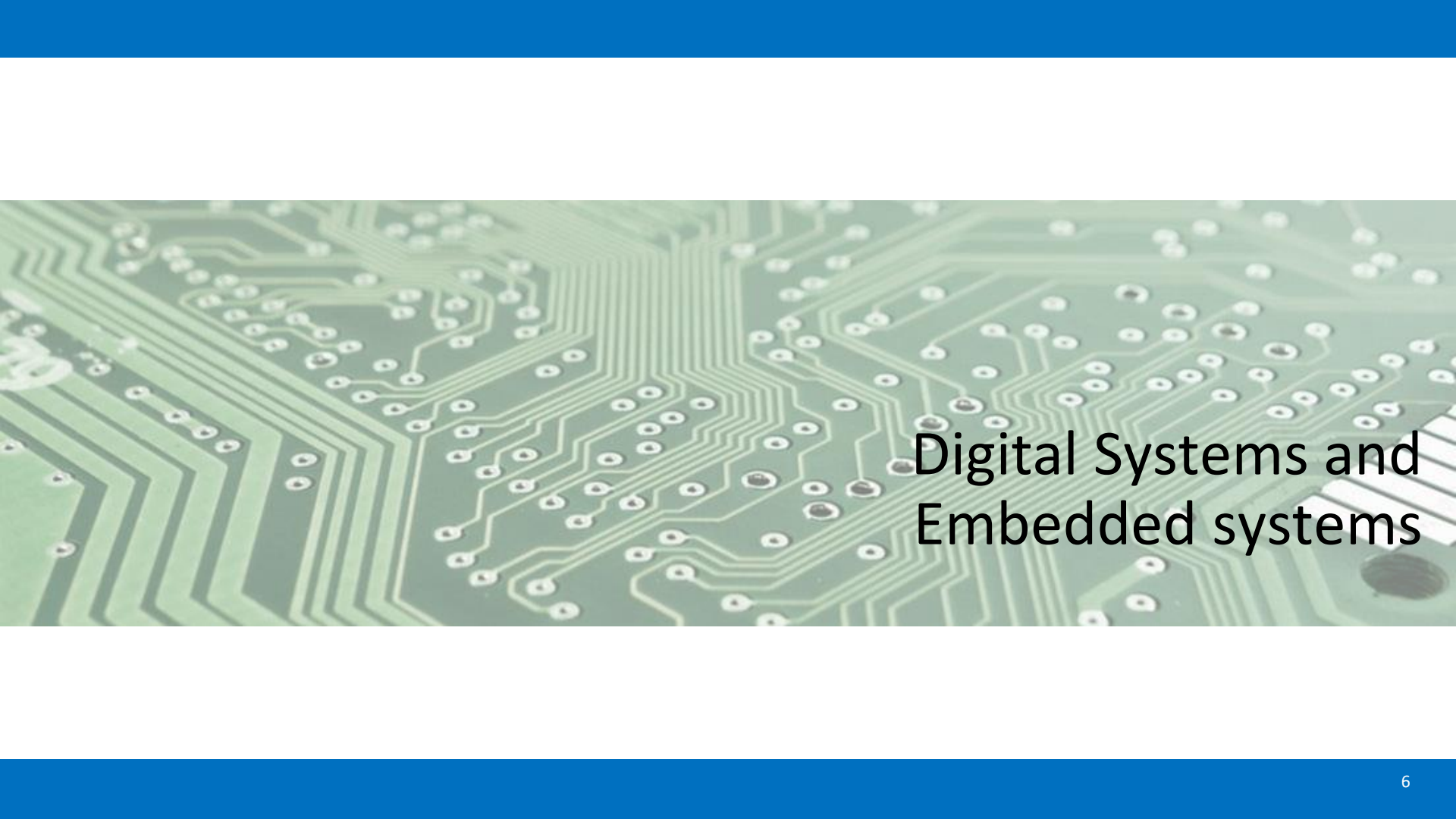


## Project Work (+4 CFU)

- ✓ Project works are available and can be done by:
  - ✓ Extending the project developed for the exam (recommended)
  - ✓ Taking a brand-new project
- ✓ Same evaluation protocol as for the exam

## Material

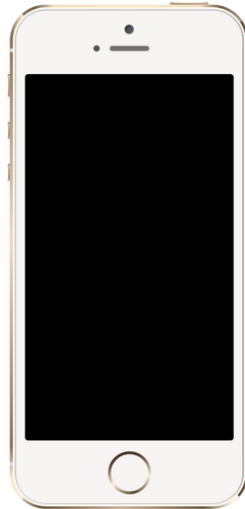
- ✓ Slides and code
- ✓ Additional documentation reported in the slides



# Digital Systems and Embedded systems

## What is an embedded system (ES)

- ✓ There is no real definition for ESs
- ✓ Processing system with low power consumption, that is lightweight and easy to carry along?
- ✓ A popular example:  
*smartphone*



# Introduction

## Examples of applications



[www.irobot.com](http://www.irobot.com)



[www.mercedes-benz.com](http://www.mercedes-benz.com)



[www.piaggiofastforward.com/gita](http://www.piaggiofastforward.com/gita)



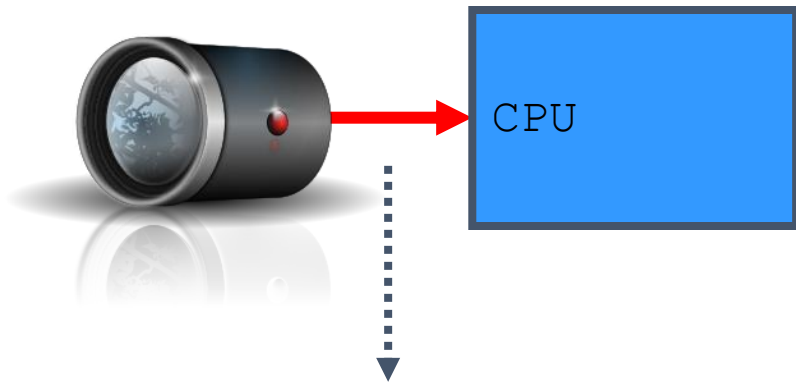
[www.orcam.com](http://www.orcam.com)



## An application: Computer Vision (CV)

- ✓ Raising in popularity
- ✓ All the systems from previous slide process images
- ✓ Extracting data from images is expensive
- ✓ Recently (2014+), **paradigm shift**
- ✓ Nowadays, mostly based on machine-learning/deep-learning
- ✓ Most popular framework: Convolutional Neural Network (CNN)
- ✓ Further increase of computational requirements
- ✓ Dedicated hardware (examples, Google TPU, Intel Stick, Apple Neural Engine, etc)
- ✓ Developers frameworks (eg, Tensorflow) and code optimization (eg, Apple, Google, Intel, etc)

## Video Stream processing 1/2



### Monochrome

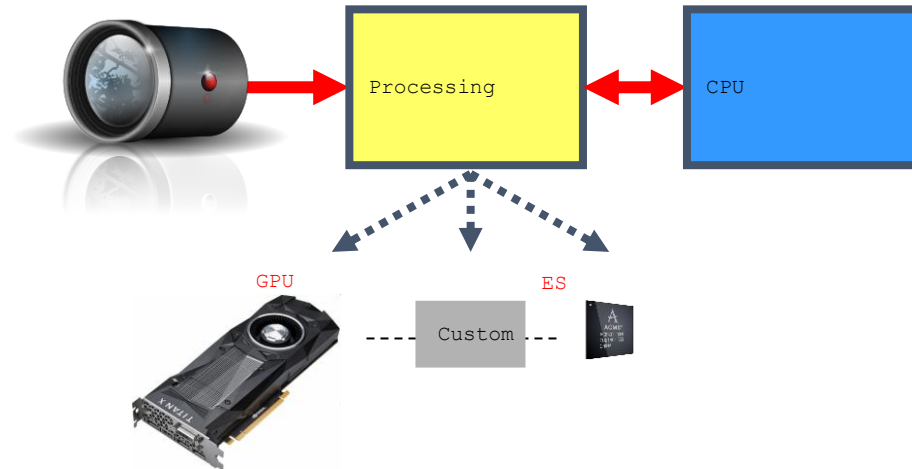
- VGA (0.3MP) 640x480@30fps : 9+ MP/s
- VGA (0.3 MP) 640x480@60fps : 18+ MP/s
- 720p (0.9 MP) 1280x720@60fps : 55+ MP/s
- 1080p (2 MP) 1920x1080@60fps : 124+ MP/s
- UHD (8 MP) 3840x2160@60fps : 497+ MP/s
- .....

# Introduction

## Video Stream processing 2/2

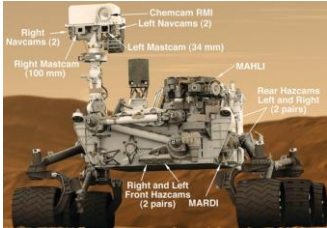
In this course, we will see two main strategies:

- ARM + FPGA ([low-level](#))
- Custom embedded systems ([high-level](#))



# Introduction

## Applications



NASA Mars rover



Google car



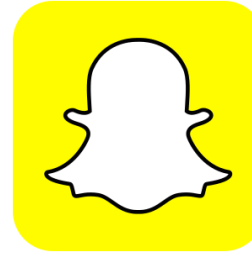
Apple iPhone X



DJI drones



Microsoft HoloLens



Snapchat



<http://f1tenth.org/>  
Università Modena e Reggio Emilia



# Introduction

## Data processing from images: setup

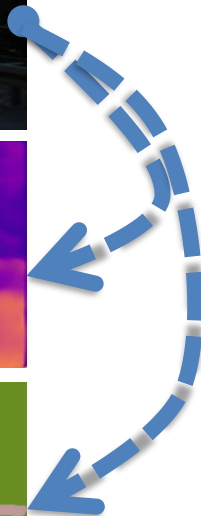
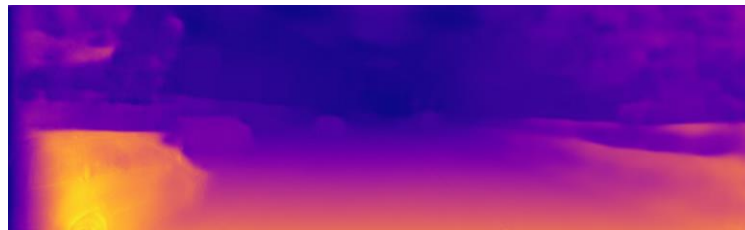
- ✓ Stereo (2+ sensors)  
Increasing availability
- ✓ Monocular  
Available everywhere (eg, surveillance camera, smartphones)



In both cases, deep-learning solutions are state-of-the-art

# Introduction

## Examples of data extracted from images



- ✓ Depth and other cues used for high-level tasks
- ✓ Examples:  
autonomous driving, Augmented Reality, etc

# Introduction



# Introduction





# Introduction



# Introduction



# Introduction



# Introduction





# Introduction



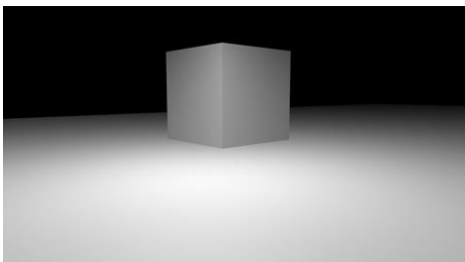
# Introduction



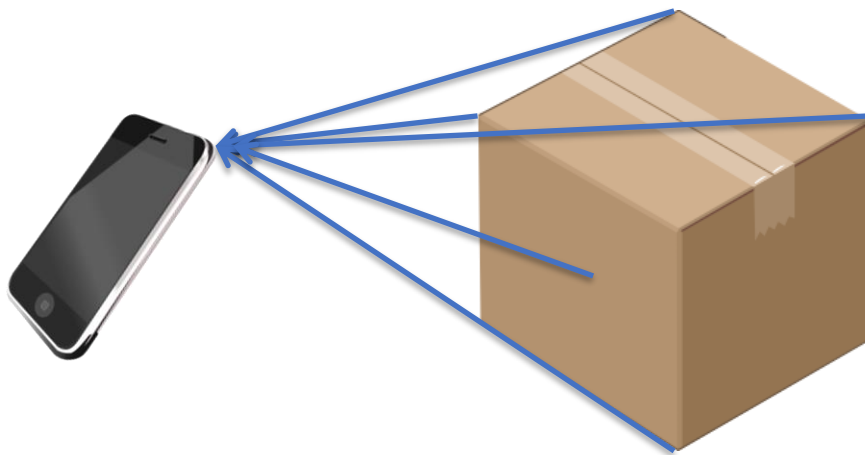
# Introduction

## Depth inference

- ✓ Unconstrained scene processing
- ✓ Allows for interaction with the observed scene
- ✓ E.g., occlusion handling



<https://ascendstudios.com>







# Introduction



# Introduction



# Introduction



# Introduction





# Introduction





# Introduction



# Introduction



# Introduction



# Introduction

## Depth from monocular camera



<https://www.youtube.com/watch?v=h6Wo5MqbCY0&t=10s>



# Introduction

## Depth and semantics



<https://www.youtube.com/watch?v=IJLToD9XPHQ>

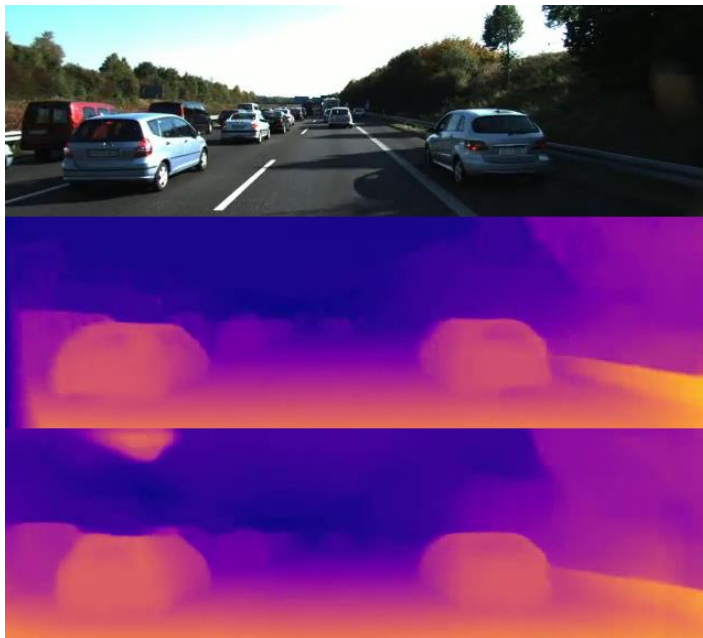


## First optimization

- ✓ In the previous examples, high frame rates (10+ FPS) are possible only on GPUs
- ✓ Anyway, proper network design choices can ease deployment on ES
- ✓ The following examples goes in this direction
- ✓ Starting from this, further optimizations are possible thanks to custom optimization frameworks made available by hardware producers (eg, Apple e Intel)

# Introduction

## Real-time depth from monocular camera



<https://www.youtube.com/watch?v=PCLmr8V456o&t=44s>

# Introduction

## Real-time self-adaptation

- ✓ These systems often suffers when moving to unknown environments
- ✓ Deployment of systems that can adapt (in *unsupervised* manner) to the new environment



<https://www.youtube.com/watch?v=7SjyzDxmCY4>



# Introduction

## Standard processing platforms

- ✓ Deep learning systems require high amounts of memory and power for training and inference
- ✓ Typically, high-end hardware (eg, NVIDIA Titan)
- ✓ About 250 W
- ✓ Together with a PC (additional 100+ W)
- ✓ Not suited for many applicative scenarios



# Introduction

## Non-programmable ES

✓ *Low-power embedded platforms exist for this purpose*



<https://developer.nvidia.com/embedded/jetson-tx2>

<https://cloud.google.com/tpu/>



[www.jevoisinc.com](http://www.jevoisinc.com)



[www.intel.com](http://www.intel.com)

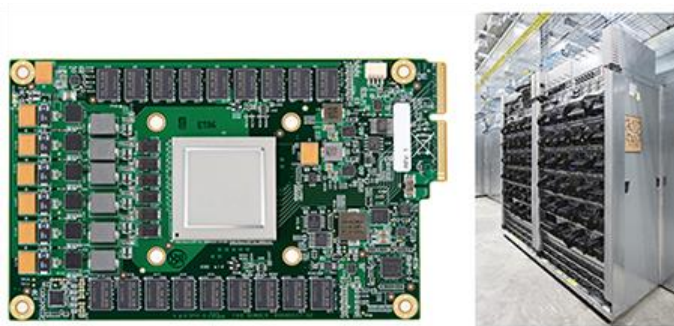


[www.google.com](http://www.google.com)  
[www.apple.com](http://www.apple.com)



# Introduction

## Tensor Processing Units (TPUs)



<https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu>

- Custom architecture (ASIC) designed to reduce costs and maximize performance in datacenters
- Nowadays ASICs are used for data processing concerning deep learning (Google)
- A standard architecture (CPU+GPU) can not compete in terms of Performance/Watt

# Introduction

## Brainwave (Microsoft)



<https://www.microsoft.com/en-us/research/blog/microsoft-unveils-project-brainwave/>

- FPGA + ARM architecture, used for the same purpose of Google TPUs
- Again, consumer solutions (CPU+GPU) can not compete on Performance/Watt
- As for TPUs, it is used for inference
- Training on GPU

## Programmable ES



[www.xilinx.com](http://www.xilinx.com)

- ✓ Previous examples are not programmable
- ✓ Some platforms allow for creating our own architecture, tailored for our task
- ✓ Platforms combining CPU and FPGA are particularly appealing for this purpose

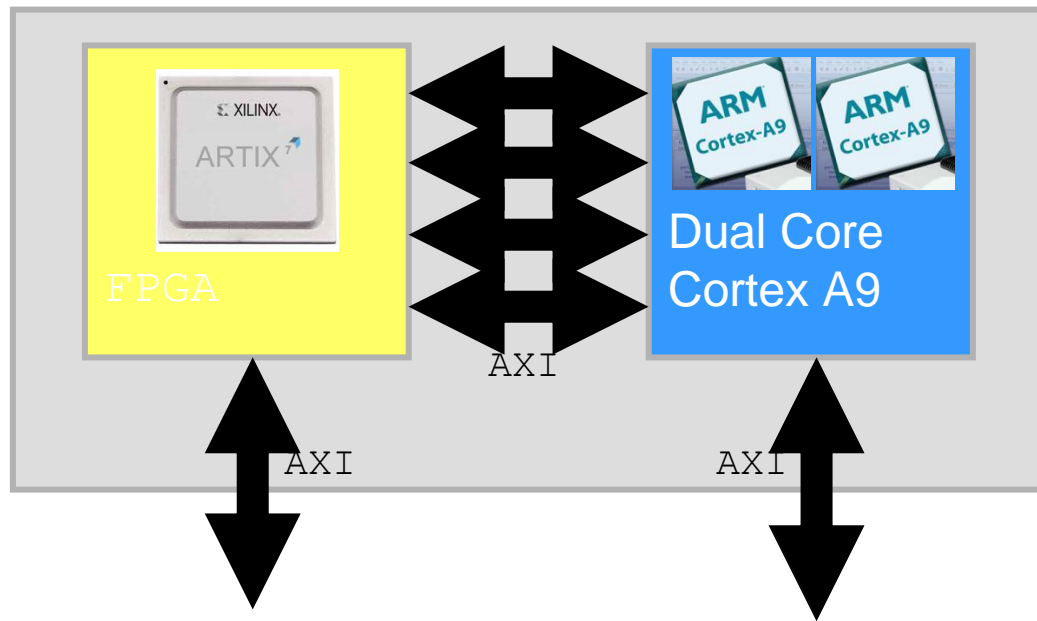
## Non-Programmable ES (high-level)

- Smartphone
  - Raspberry
  - Jetson TX2
  - ...
- 
- The architecture is fixed
  - Several optimization tools for network deployment
  - Programming languages: Python (+ Tensorflow)

## Programmable ES (low-level) 1/3

Among programmable or reconfigurable devices (FPGA) and hard-core systems(ARM):

- ZYNQ 7000 (Xilinx)



[http://www.xilinx.com/support/documentation/data\\_sheets/ds190-Zynq-7000-Overview.pdf](http://www.xilinx.com/support/documentation/data_sheets/ds190-Zynq-7000-Overview.pdf)



## Programmable ES (low-level) 2/3

ARM specifics (PS):

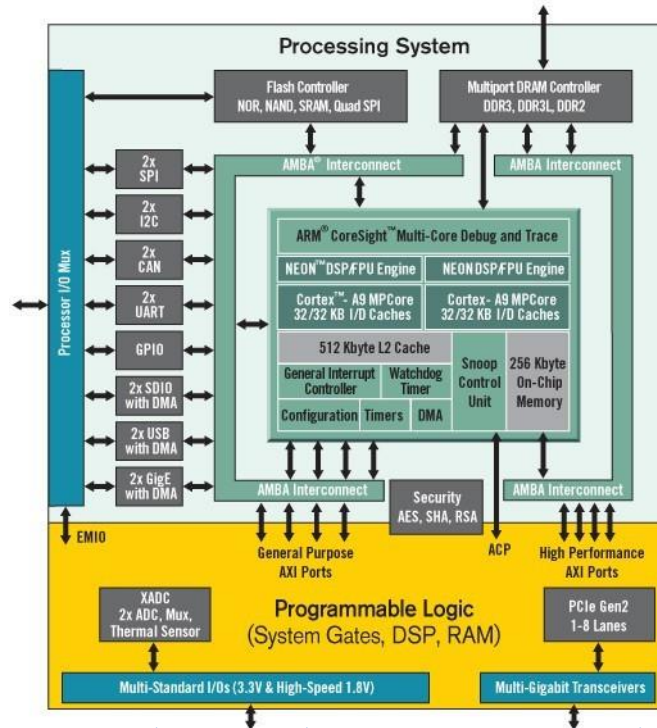
- 1 or 2 CPU ARM Cortex A9 (ARM v7-A architecture)
- Cache L1, for data and code, 32 KB
- Cache L2, unified among the 2 cores, 512 KB
- ROM for bootstrap
- 256 KB on-chip RAM
- Interface to external memories (DDR2,DDR3,etc)
- DMA Controller, 8 channels
- Bus AXI and others
- Ethernet 10/100/1000 Gb Ethernet
- USB controller
- etc ...

## Programmable ES (low-level) 2/3

- Concerning the FPGA, 7 series Xilinx FPGA settings are kept
- ZYNQ adopts Linux as OS (the architecture is fully supported by Linux kernel).
- It is possible to design our own hardware units using the programmable gates (yet, we need to write our own driver)
- At bootstraps, the ZYNG board starts one of the ARM core, allowing for a safe configuration of both Linux and FPGA.
- ZYNG devices can be emulated by means of software (QEmu for Linux).

# Introduction

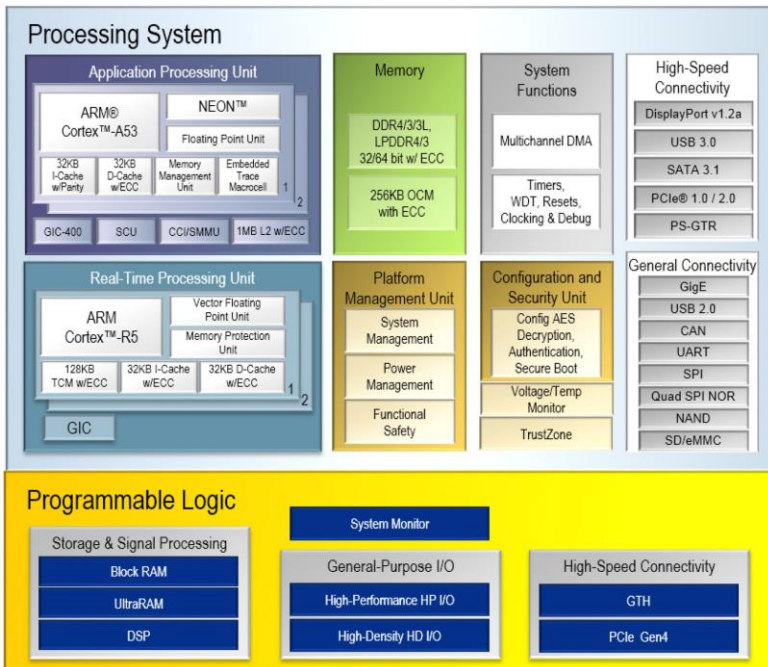
## Zynq 7000 architecture



<http://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html>

## New generation: Ultrascale+ MPSoC

- 2 or 4 core Cortex A53 (ARM 64 bit architecture)
- 2 core Cortex R5 (real-time)
- Encoding/Decoding H264/H265
- .....

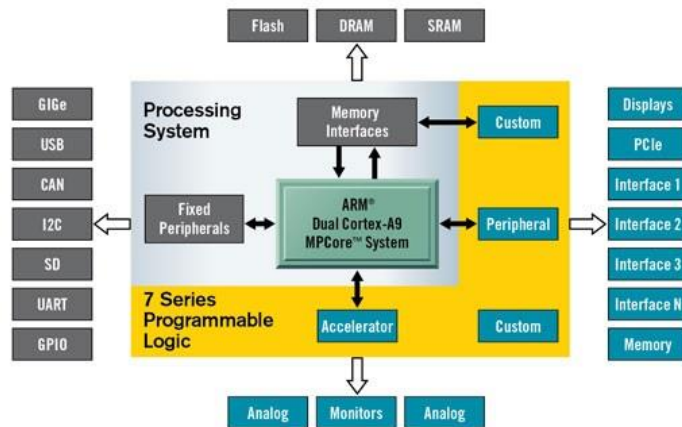


<https://www.xilinx.com/products/silicon-devices/soc/zyng-ultrascale-mpsoc.html>

# Introduction

## PYNQ architecture

- A high-level productivity language (Python in this case)
- FPGA overlays with extensive APIs exposed as Python libraries
- A web-based architecture served from the embedded processors, and
- The Jupyter Notebook framework deployed in an embedded context



[https://pynq.readthedocs.io/en/v2.5/pynq\\_overlays.html](https://pynq.readthedocs.io/en/v2.5/pynq_overlays.html)



# Introduction

## Computer vision (CV) con ZYNQ (low-level)

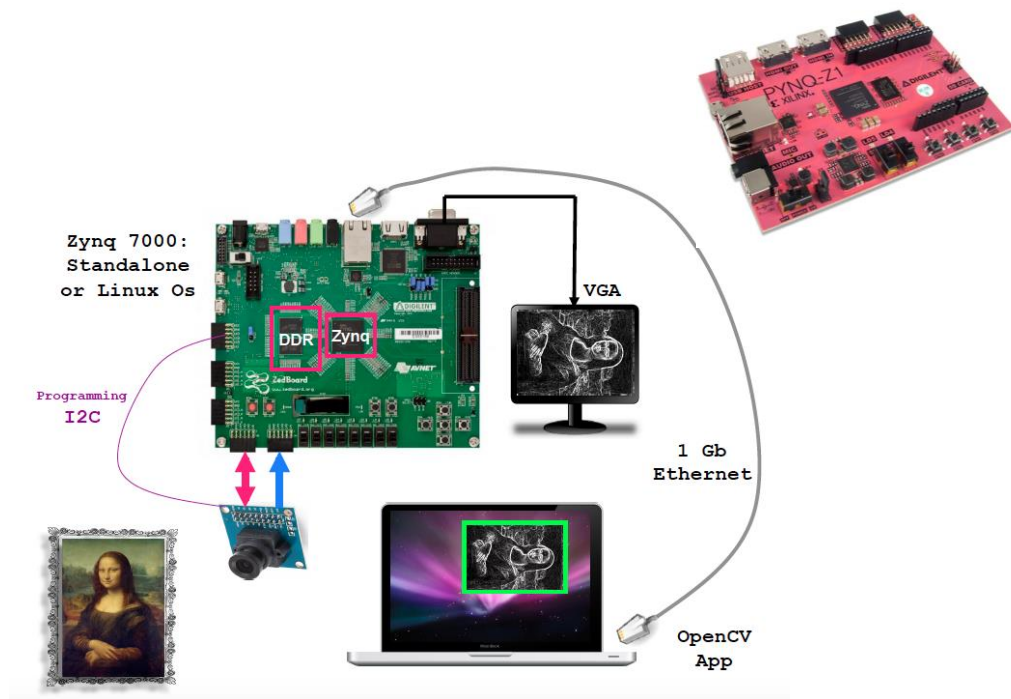
- ✓ Programming language: C/C++
- ✓ More recent solutions: Python (PYNQ boards)
- ✓ Bootstrap project available at:  
<https://github.com/smatt-github/SmartCamera>

- ✓ Examples:

<https://www.youtube.com/watch?v=EG3NYqMJvZI>

<https://www.youtube.com/watch?v=QFxiXWgBBcc>

<https://www.youtube.com/watch?v=6HEgdZEHpsM>



## In this course:

- ES design (low-level) with high-level synthesis (HLS) frameworks in C/C++
- CNN mapping on custom ES (high-level)
- Main applications (but not limited to): computer vision and deep learning systems