Embedded Systems

Achitectures and Design Styles

Prof. William Fornaciari
Politecnico di Milano, DEIB
William.fornaciari@polimi.it

Outline

- Architectures and approaches to the design
  - Printed Circuit Board
    - Components
    - Support
    - Mounting
    - Design
  - **System-on-Chip**
    - **Design**
      - **Design for testability**
  - **Distributed Embedded Systems**
    - **Application fields**
    - **Wireless sensor networks**
    - **Design**
  - **Platforms for prototyping**
    - **Types of systems**
    - **Design**

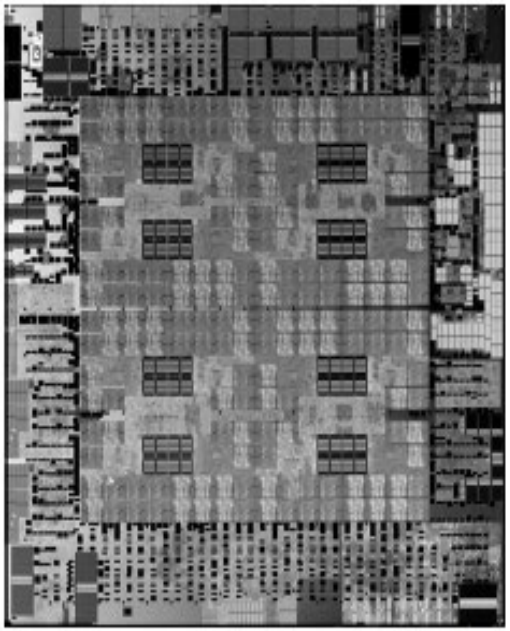
System-on-Chip

SoC

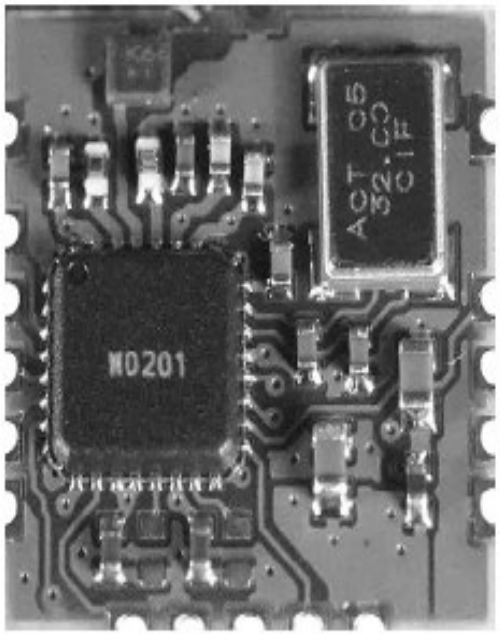
System-on-Chip

- In general, a SoC is a complete system where all the parts are integrated in a single silicon die
  - Reduction of the number of external/discrete components
  - Small form factor
- However, some parts of the overall system cannot be integrated for electrical and/or mechanical reasons
  - High currents
  - Not available in the integrated silicon process
  - Robustness of the connection
  - In any case, presumably, a small board will be necessary

System-on-Chip



a. Die di un system on chip (IBM)



b. Piccolo system on chip montato board

System-on-Chip

- Why implementing an application as a SoC?
  - Unit cost
  - Performance
  - Energy/Power
  - Size
  - Number of requested I/O
  - Security
    - Protection of Intellectual Properties (IPs)
    - Sensitive information/data

System-on-Chip

- SoC is a generic term with several flavors, depending on the architecture itself
  - One or multiple processor, possible heterogeneous
  - Several types of memories
  - Digital blocks dedicated to accelerate applications
  - Standard digital cores
    - Encoder/decoder, filters, FFT
  - Time-related functions
    - Timer, watchdog, oscillators, phase locked loop (PLLs)
  - Power supply distribution
  - Interfaces
    - ADC, DAC, standard serial and parallel, Network interfaces

System-on-Chip

- Approach to the design
  - Similarly to the PCB-based systems, the first step is the system partitioning
    - Since the adopted technology will be the same, in this case, the driving criteria are mainly related to the functionality/performance
      - Connectivity is less critical w.r.t. PCB-based systems
    - Once defined the partitioning in terms of Hw and Sw functionalities, the following step is the allocation of the functionalities on the different computing resources

System-on-Chip

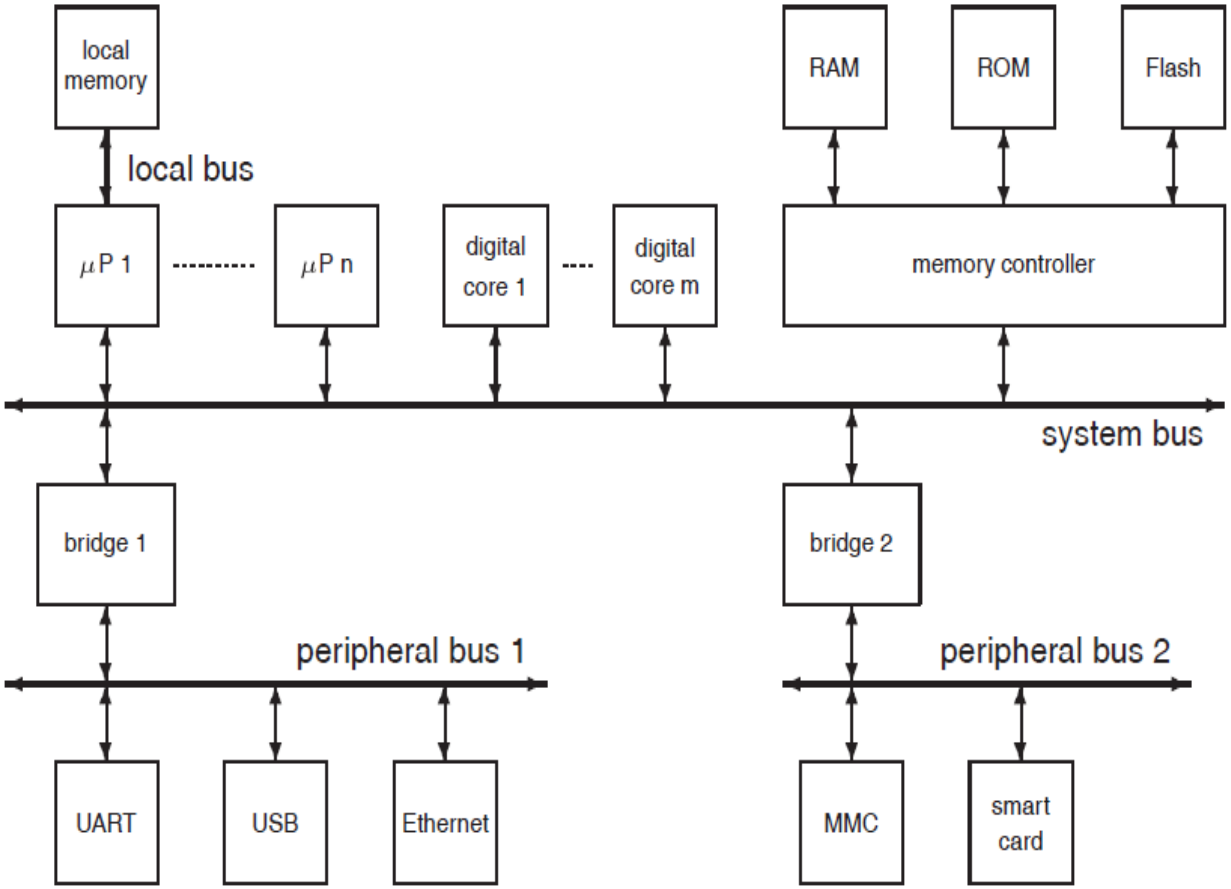
- Approach to the design
  - Based on the chosen allocation, the next step is the identification of the memory hierarchy and of the access policies
    - Centralized vs distributed solution
  - The definition of the memory architecture is depending mainly on performance aspects and the locality of data
    - In general high locality increases the performance of the sub-systems, but makes more complex the communication, synchronization and the data consistency
    - More centralized solutions relax the data consistency problems, simplify communication but, due to data conflicts, the performance are worse

System-on-Chip

- Approach to the design
  - The communication infrastructure is strongly linked to the memory structure and the partitioning of the functionality
  - Classical (hierarchical bus) solutions are losing popularity in favor of NoC architectures, where a small net, encompassing addressing and on-chip routing, is in charge of the information delivery
  - NoC, despite the need of relevant hw resources (20-30%), solves several complex problems, the main being the synchronization of sub-systems with different clocks
    - NoC is suitable to highly scalable solutions
    - Mesh structure focus more on the use of switches

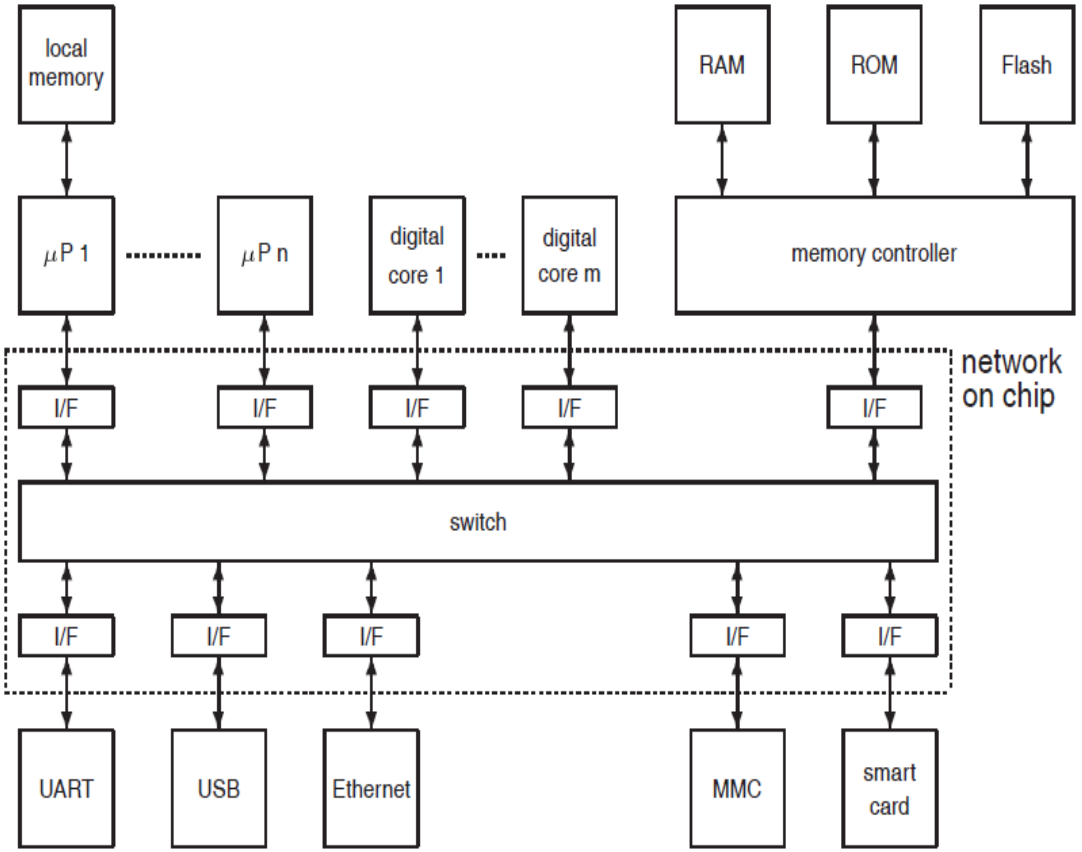
System-on-Chip

- Structure with hierarchical buses



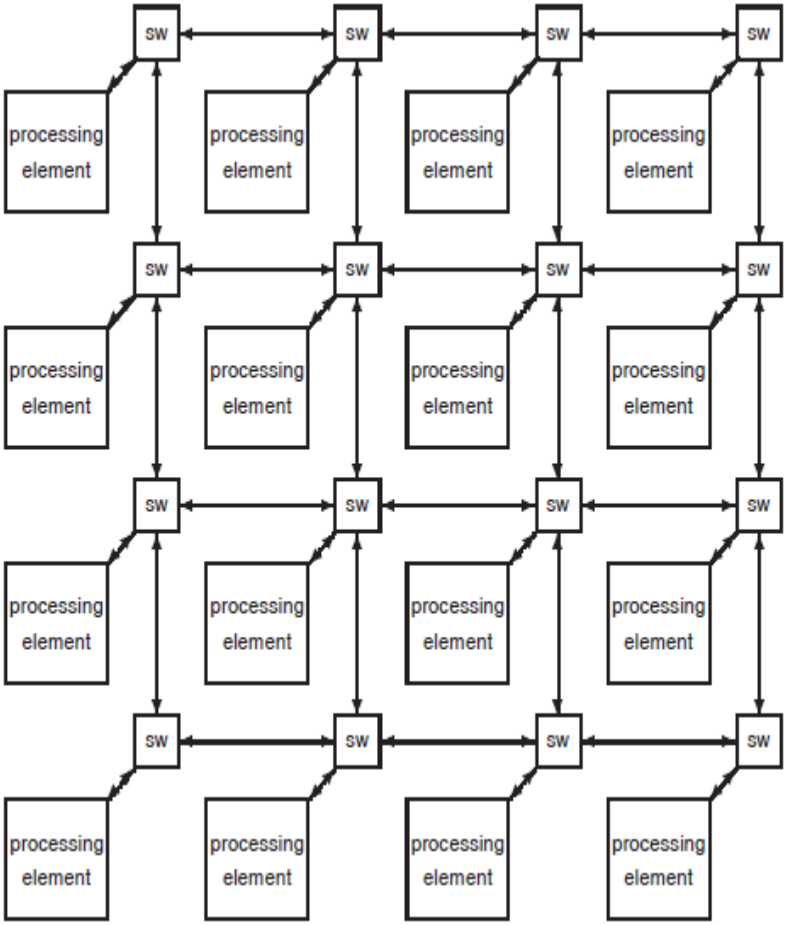
System-on-Chip

- Centralized NoC



System-on-Chip

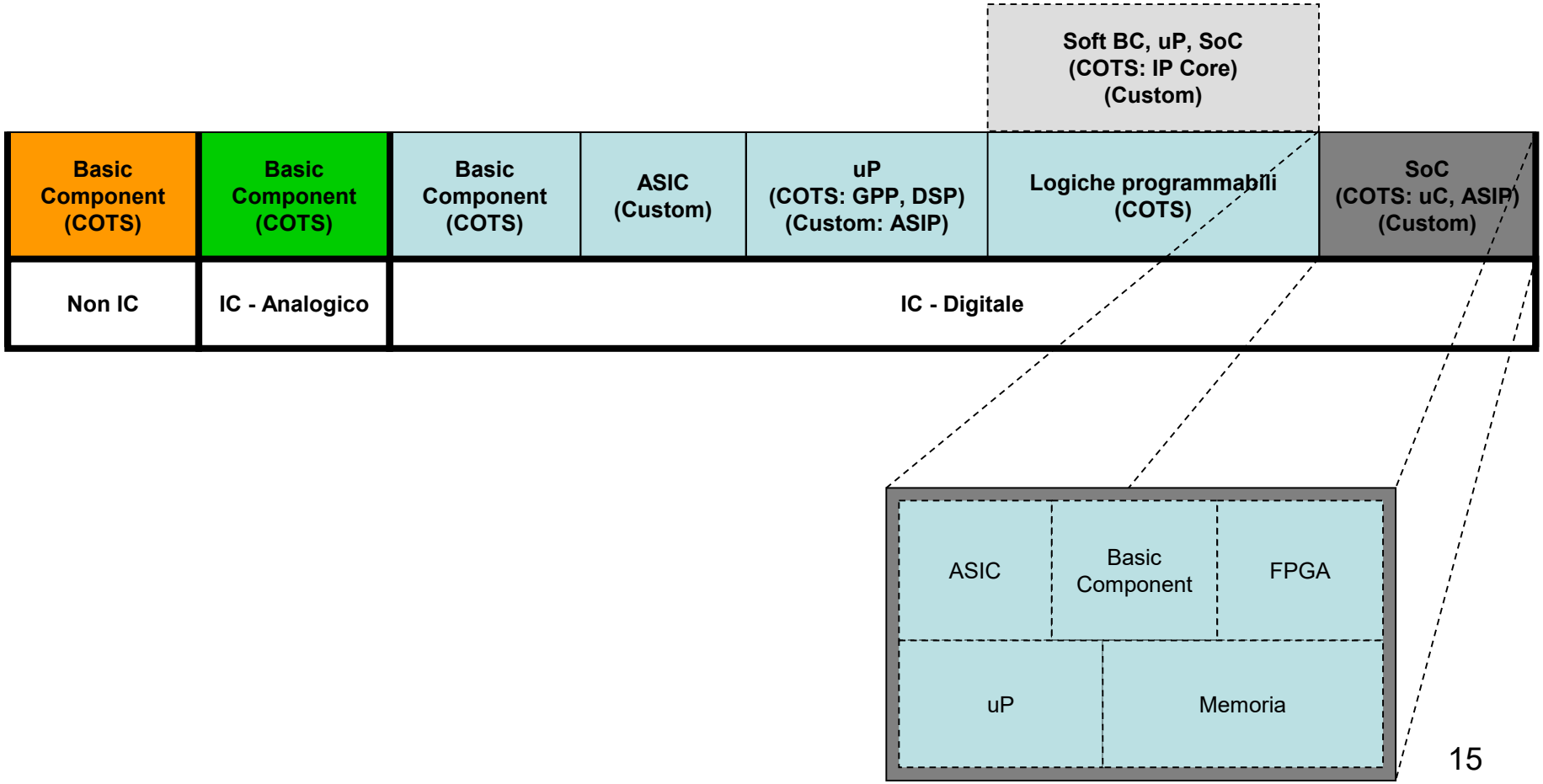
- NoC Meshed



Racap…

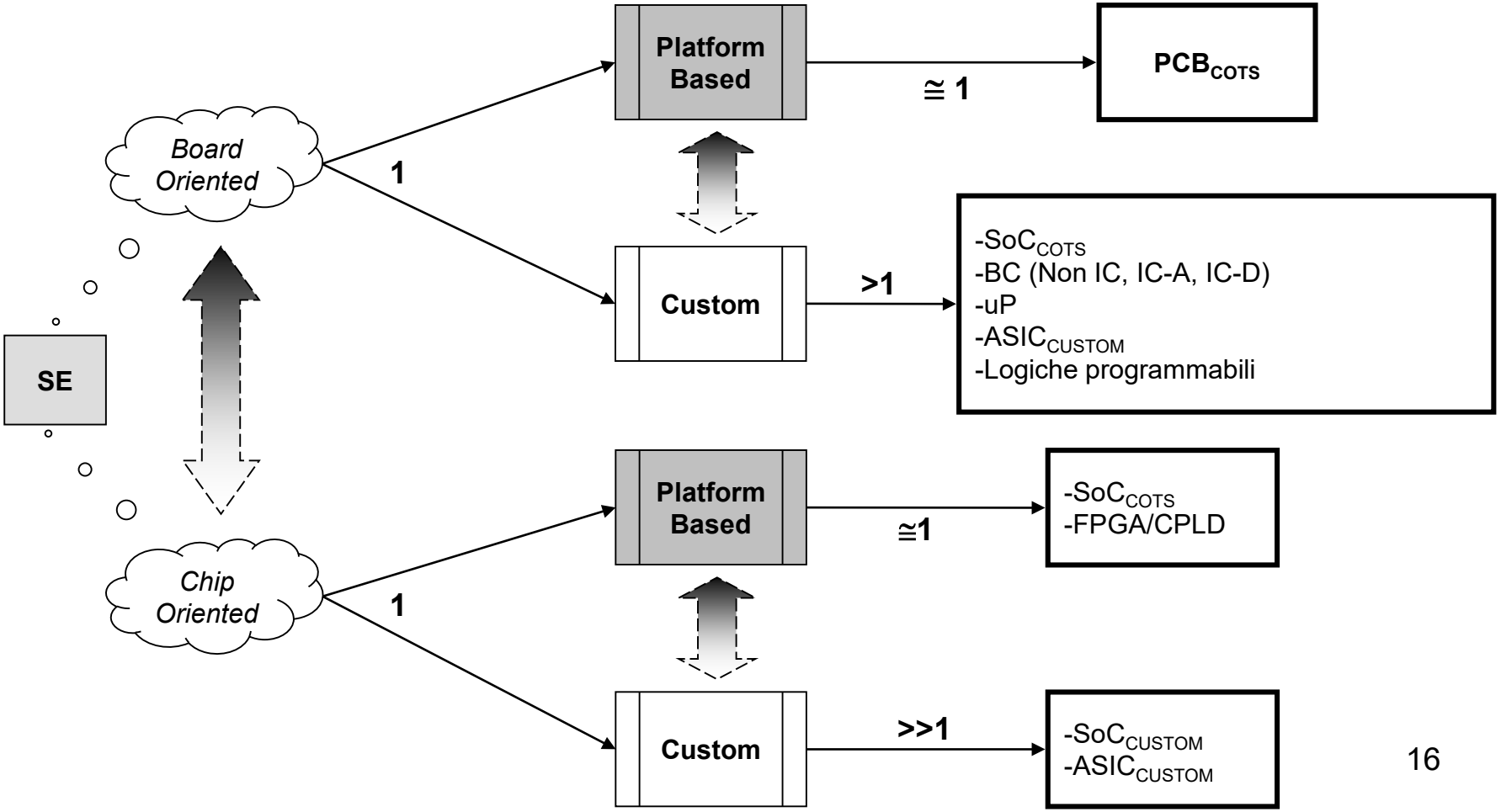
Recap…

- Hw Components for SE



Recap…

- Solution space for the HW of an ES



System-on-Chip

- Design for testability
  - Singolo chip architecture: verification problem
    - Off the shelf components can be tested before integration, and then proceed with the system testing
      - A similar strategy cannot be adopted for SoCs, since it is impossible to produce and to test the different modules, separately
  - Prototype systems offer a possible escape path, but usually the problem is afforded at design time
    - The design methodology aims at simplifying the testing, possibly by introducing additional modules with the unique goal to support the testing phases

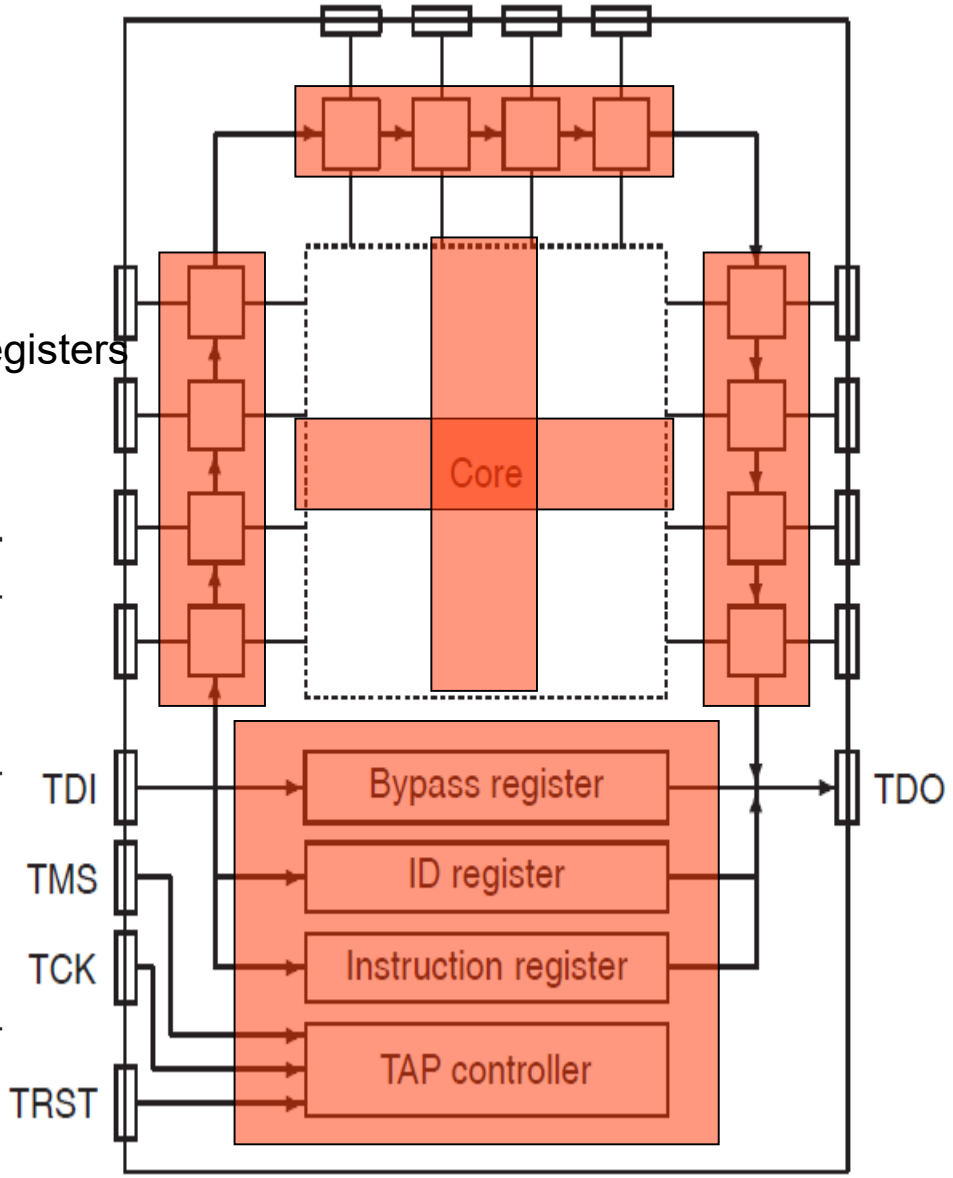
System-on-Chip

- Design for testability
  - Boundary, partial and full scan (same concept seen for PCBs)
    - Beyond boundary scan, there are structure (partial or full scan), depending if they include all or part of the chip registers
      - Full scan is expensive in terms of resources and complex to manage
      - Partial scan, if well organized, is a valuable support
    - To carry out complex testing operation, it is necessary to configure and to use the scan chain in several ways
      - TAP controller (*Test Access Port controller*): implements a simple protocols with some configuration commands
      - The architecture of the TAP controllers, the communication protocols, and the configuration commands are standardized: IEEE 1149.1 JTAG (Joint Test Action Group)

System-on-Chip

- Design for testability
  - Boundary, partial e full scan
    - JTAG Architetcures
      - TAP controller and scan registers

Istruzione	Registro	Descrizione
EXTEST	Scan	Test delle linee di connessione tra i dispositivi su board
BYPASS	Bypass	Connette TDI a TDO, saltando la catena di scan
SAMPLE	Scan	Carica nel registro di scan i valori alle uscite del core
PRELOAD	Scan	Carica nel registro di scan i valori impostati tramite TDI
INTEST	Scan	Applica un pattern agli ingressi del core
IDCODE	ID	Legge l'identificativo memorizzato nel registro ID
USERCODE	ID	Legge un identificativo opzionale
RUNBIST	—	Attiva il BIST (vedi oltre)
CLAMP	Bypass	Fissa i valori sulla scan chain e seleziona bypass
HIGHZ	Bypass	Porta i pin di uscita in alta impedenza e seleziona bypass

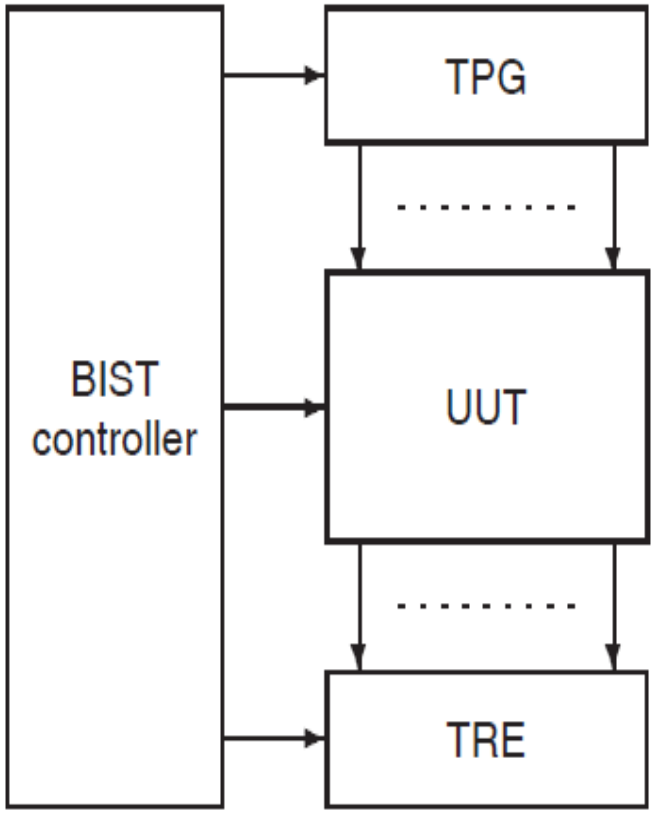


System-on-Chip

- Design for testability
  - BIST
    - Tests realized using scan are not sufficient or not effective for complex systems
      - The main problem is that they operates at a frequency that is lower w.r.t. the normal operation and the systems need to be controlled externally
    - To overcome such limitations, families of verification strategies have been developed to operate internally and automatically, named *Built-In Self-Test*
      - A generator of test vectors (TPG) produces in output sequences of test vectors that are sent to the UUT (Unit Under Test), under the control of a control unit (BIST Controller)
      - The output produced by the UUT are analyzed by a dedicated hw, named ORA (Output Response Analyzer) or TRE (Test Response Evaluator)

System-on-Chip

- Design for testability
  - BIST
    - General architecture
      - TPG
        - » Exhaustive test
        - » Statistical Test
        - » Segmented Test
      - TRE (ORA)
        - » Signature based



Distributed Embedded Systems

Distributed/Networked ES

- Processing and storing of data split over a number of interconnected devices
  - Equally distributed or just connected
    - *Distributed vs. Networked*
- When the interconnection and the communication is not trivial, the design approach need to be ad-hoc selected
  - Initial focus on the criticalities
  - In a later stage the classical methodologies for PCB-based or SoC based will be followed

Distributed ES

- Two classes depending on communication infrastructures
  - wired
    - Infrastructure fixed and well-known and characterized
  - wireless
    - Infrastructure not fixed and more complex to characterize
  - Typical wired applications: domotic, automotive, industrial, …
  - A relevant interest is towards *Sensor Networks*, *wired* or *wireless*
    - *Wireless Sensors Networks*, are gaining attention in sectors like telemedicine, monitoring of the environment and cultural heritage, etc

Distributed ES

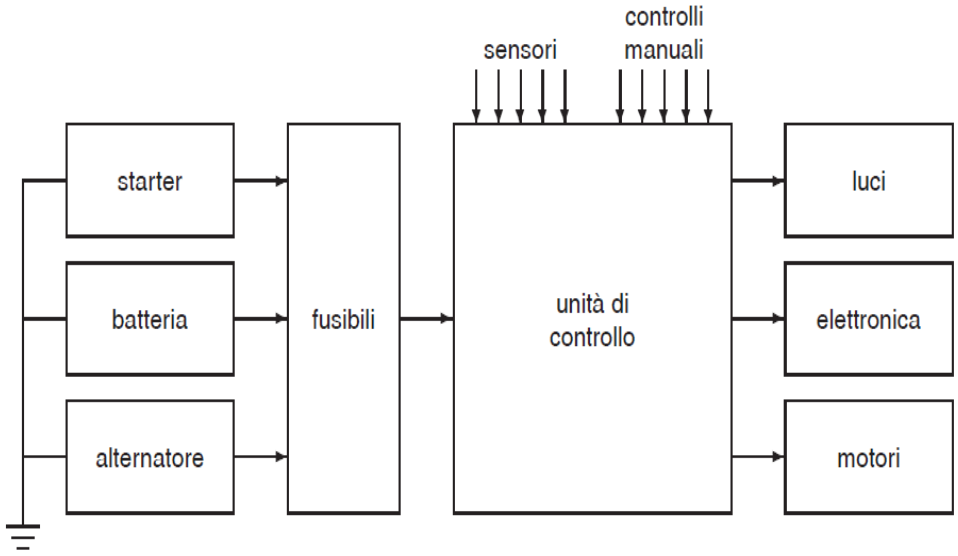
- Application domains
  - Domotic
    - Many apparatus need to be connected and controlled, distributed nature of the systems
      - From simple electrical plugs to alarm systems, white goods, audio/video
      - An obstacle to the design is the need to interface devices with different complexity ad computing capabilities
    - Some commercial solutions exploit a centralized control encompassing most of the control logic and some nodes to detect and to collect specific information

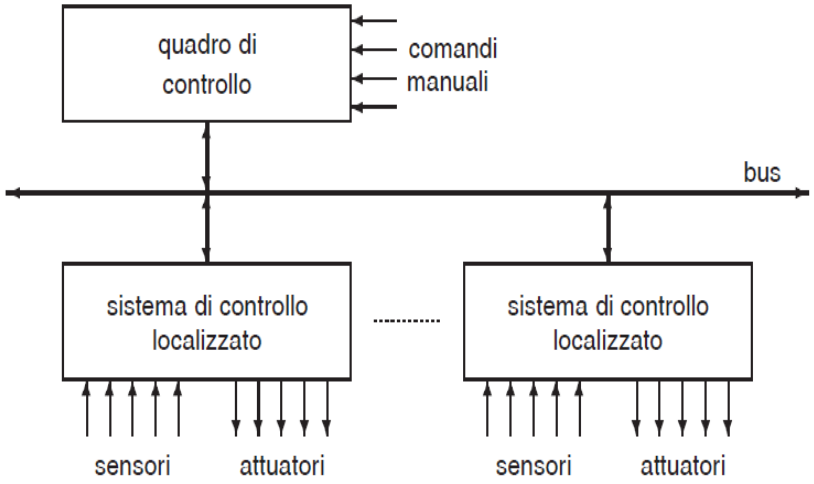
Distributed ES

- Application domains
  - Automotive
    - Automation and control of sub-systems installed and operating on a generic vehicle
    - The classical approach consists of a central unit driving a number of actuators
      - Dedicated wireless connections are used (e.g., CAN field bus)
    - There is a push towards a distributed solution, impacting on the budget of the system
      - Safety: the cabling could be a weak point producing malfunctioning, and can be a load and cost not negligible
      - Different OEM suppliers need to be integrated

Distributed ES

- Application domains
  - Automotive
    - Distributed vs centralized architecture



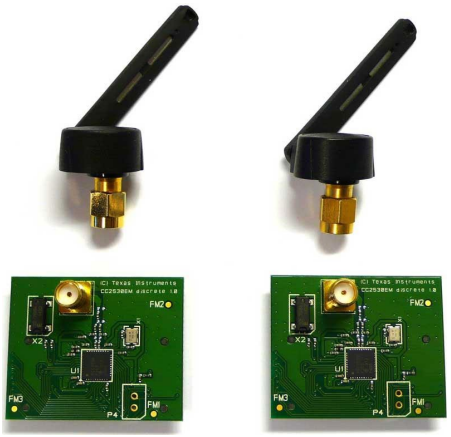


Distributed ES

- Wireless Sensor Networks
  - WSN is a net of sensor nodes, rather autonomous, distributed in the space and equipped with several sensors. They cooperate to provide a global monitoring function for the controlled environment
  - Beside the focus on the sensors, actually each node is integrating active components like a microcontroller, a small amount of memory, transceiver, antennas, a energy source (battery, solar panel, …) and, sometimes, also actuators
    - The size and the computing power of each node can vary significantly
    - From complex nodes similar to small computers down to simple nodes, possibly disposable
    - Emerging problem: what about the reliability/security of the collected information? A new entry point for attacks?

Distributed ES

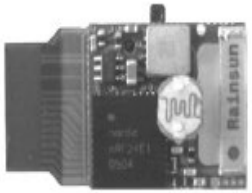
- Wireless Sensor Networks
  - Examples



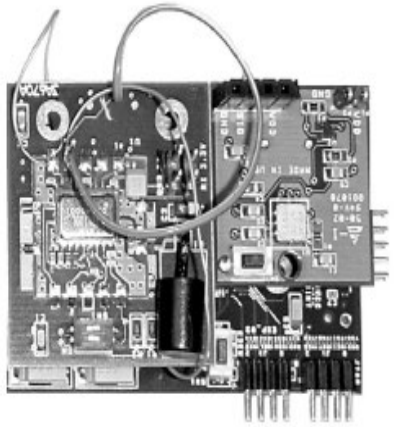
CC2530 802.15.4 LR-WPAN Evaluation Kit
By Texas Instruments

a. Nodo Mica di Crossbow.

b. Nodo Mica2dot di Crossbow.



c. Nodo EcoMote di UC Irvine.



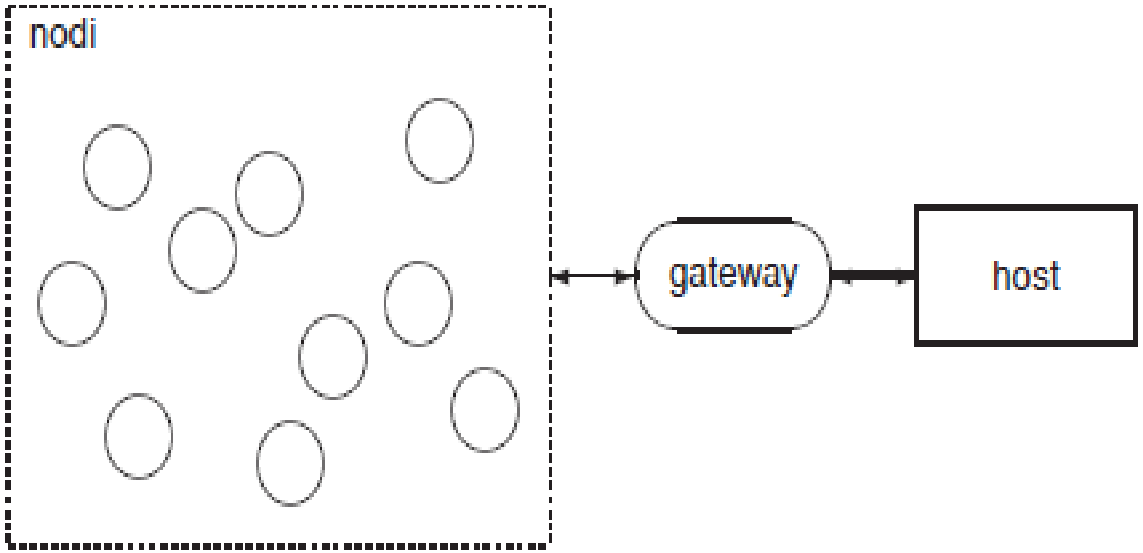
d. Nodo di Gnome di Rice University.

Distributed ES

- Wireless Sensor Networks
  - There are specific commonalities among the different WSNs
    - Limited possibility to collect and to store energy
    - Operating in harsh environments
    - Heterogeneity of nodes
    - Mobility of nodes
    - Failures and possible breakdown of single nodes
    - Malfunctioning of the communication
    - Variable topology of the network infrastructure
    - Possibility to operate for long periods of time without maintenance

Distributed ES

- Wireless Sensor Networks
  - General System Architecture



Distributed ES

- Wireless Sensor Networks
  - Approach to the design
    - System models
      - Physical nodes vs. functional components
      - Local computation vs communication
    - Hardware architecture of the nodes
      - Microprocessor/Microcontroller
        - » IBM 8051, Atmel ATmega 128L, XScale PXA271, Philips CoolFlux, Cambridge Consultant 16-bit XAP2b e Texas Instruments MSP430
      - Chipset for communication + antenna
        - » ChipCom CC1000 e CC2420
      - Local communication buses
        - » SPI, I2C or proprietary

Distributed ES

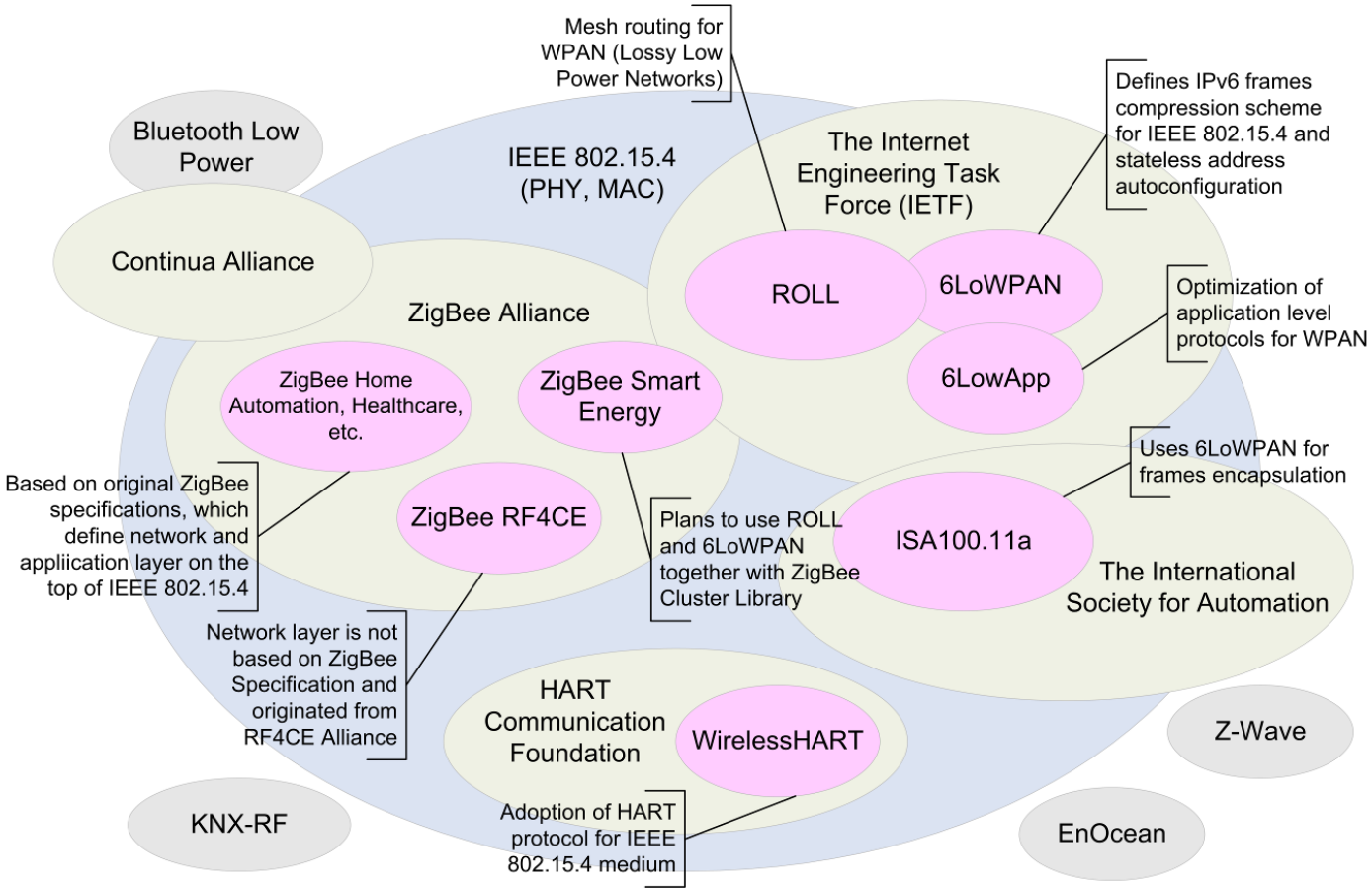
- Wireless Sensor Networks
  - Approach to the Design
    - Software Architecture of the nodes
      - The need of protocol stacks, timers, alarms, I/O, interrupts, …frequently leads to the use of simple operating systems
        - » TinyOS, MANTIS, FreeRTOS, SOS, Contiki,..
    - Programming model
      - The programing models provides support to data access and to the computing resources of the network
        - » Still today, the common model is the same of the distributed databases
      - Once defined the way used by the sensor nodes to expose the data, it is necessary to define how to use them at the application level
        - » Query, Events, Thread, Publish/Subscribe

Distributed ES

- Wireless Sensor Networks
  - Approach to the design
    - Protocols
      - Different and quite heterogeneous
        - » At the lowest levels, standard protocols are used, such as IEEE 802.15.4, SP-100.11a, IETF 6lowpan, Bluetooth, etc…
      - A crucial point is the dynamic re-configurability of the network topology that, in many cases, it is subject to modifications
        - » Specific routing algorithms: SPIN (Sensor Protocols for Information via Negotiation), Directed Diffusion, Rumor Routing, Q-RC (Q-learning Routing and Compression), etc…
        - » Every day someone is reinventing the wheel ☺

Distributed ES

- Wireless Sensor Networks
  - Approach to the design: protocols (research in progress…)



Platforms for prototyping

Platforms and Kits for prototyping

- The definition of the architecture and the choice of a technology has an impact on the overall system design
- After a coarse-grain identification of the architecture, the next step is the development and verification of the macro-functionalities
  - Co-Simulation
    - The tool-flow for Hw-Sw co-simulation are useful during the initial phases but they are not sufficient for a complete verification
      - In many cases the simulation models for COTS are not available
      - Simulation time can be an obstacle for systems of realistic size

Platforms and Kits for prototyping

- The solution is to start by using components for which it exists a platform for prototyping
  - Configurable computing platform, highly configurable, useful to realize a prototype
    - The verification of the systems can be much faster, complete and at a fine grain
- Such prototypes consist of dedicated Hw sections, standard components, possibly an OS and some application/demo software
  - It is used in a realistic application context for direct verification
  - This approach is named **emulation**
  - Given the complexity of nowadays systems, is it almost a mandatory step

Platforms and Kits for prototyping

- Types of development systems
  - Simple emulation systems
    - Boards with maybe a programmable device (FPGA or CPLD) and with a few standard interfaces (RS-232, USB, JTAG)
      - Useful for verification of not critical functionalities
  - Complex emulation systems
    - Boards with one or more programmable components, external memories (RAM, E2PROM, Flash), several interfaces (PCI, PCI Express, Serial ATA, Ethernet, FireWire, GPIO, USB), sections for the management of clocks and power supply, sections for interfacing and A/D conversions
      - Suitable for systems realized in the final version, on PCB or as SoCs

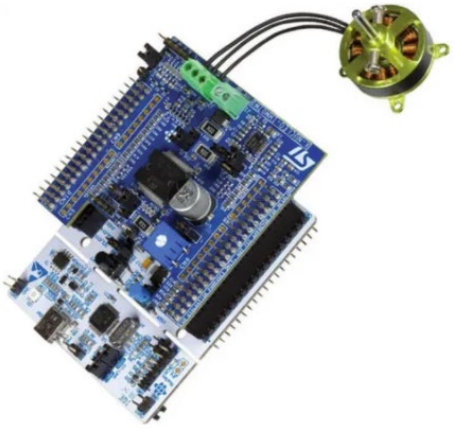
Platforms and Kits for prototyping

- Types of development environments
  - Software emulation systems
    - Many embedded systems are built around a main computing core with few interfaces
      - There exists simple solutions, with a single processor, a reasonable amount of memory, some standard interfaces and flexible and powerful solutions
  - Hardware/software emulation systems
    - This class encompasses systems of variable complexity, integrating one or more programmable devices and one or more processors
      - They offer a basic firmware (HAL/BSP), an OS and libraries for the configuration and use of the integrated peripherals

Platforms and Kits for prototyping

- Types of development environments
  - SoC Emulation systems
    - Similar to the previous platforms, they are the integrated version for single chip
      - Typically are high-end FPGAs, integrating one or more *fused* processors, memories, dedicated functional blocks like fast multipliers, standard interfaces, PLL/DPLL, etc…
  - Application-Specific Emulation systems
    - The most common are those for the development of network applications, providing a range of interfaces, or kit for sensor networks including a gateway plus nodes, or platforms for the development of audio/video applications, equipped with encoders/decoders, and many more

STM32 Nucleo Pack



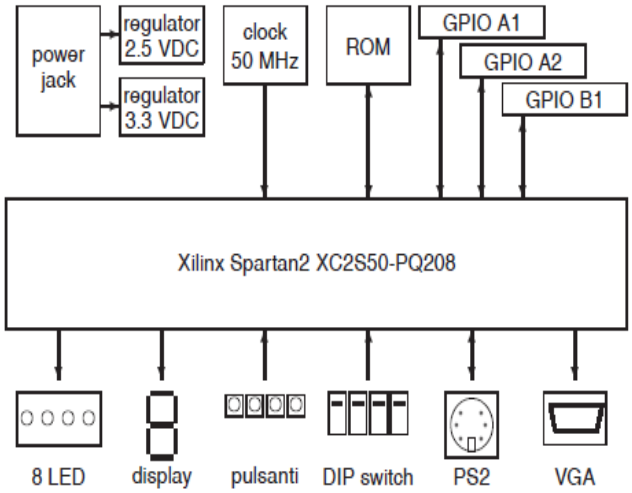
- The platform provides a motor control solution for three-phase, low-voltage and low-current DC brushless motor
- It is based on L6230 driver and STM32F302R8 MCU
  - L6230 driver is a DMOS fully integrated device for three-phase brushless PMSM motor, with overcurrent and thermal protection
  - STM32F302R8 is a 32-bit MCU based on a high-performance ARM® Cortex®-M4 32-bit RISC core, with floating point unit (FPU), operating at a frequency of up to 72 MHz and embedding an advanced analog peripheral set
  - The X-NUCLEO-IHM07M1 board is fully configurable and ready to support different closed loop control, FOC or 6-step, based on sensorless or sensor mode. It is compatible with three shunts or single shunt for current sense measuring.
  - It does not require any separate probe as it integrates the ST-LINK/V2-1 debugger and programmer

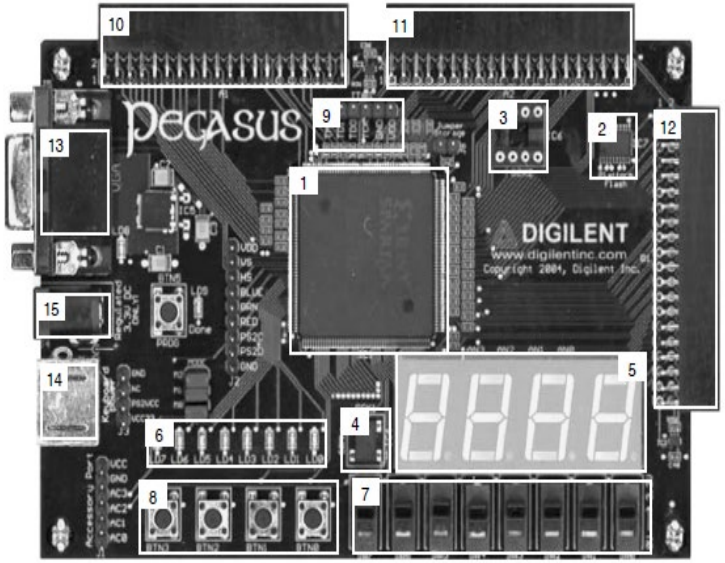
Platforms for prototyping

- Example: Digilent Pegasus

#	Descrizione
1	Spartan2 XC2S50-PQ208 FPGA
2	XCF01S Platform Flash ROM
3	Connettori per clock esterno
4	Generatore di clock on-board
5	Display LED a 7 segmenti
6	Banco di 8 LED
7	Banco di 8 DIP switch
8	Banco di 4 pulsanti

#	Descrizione
9	Connettore JTAG
10	Banco A1 di connettori GPIO
11	Banco A2 di connettori GPIO
12	Banco B1 di connettori GPIO
13	Porta VGA
14	Porta seriale PS2
15	Alimentazione





Platforms for prototyping

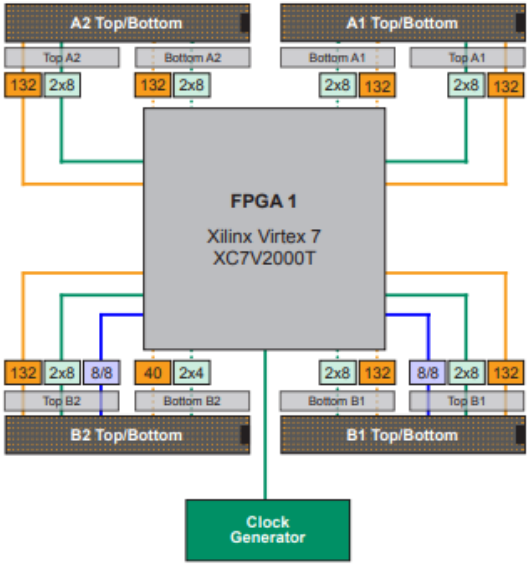
- Types of development systems
  - One of the first examples: XUP Virtex-II Pro (Xilinx)
    - Board based on FPGA Xilinx, Virtex-II Pro
      - On board there are 256MB of RAM DDR, on AC97, LED, buttons, switches for configuration, and several interfaces
    - The FPGAs integrates two PowerPCs, and the development environment provides several OSs
      - VxWorks, Linux, etc
    - For the HW design it is available *Embedded Development Kit* (EDK) by Xilinx (now Vivado replaces these tools), that allows to develop independent modules and to interface them to the CPUs
  - Similar products exploits Altera, Lattice or other Silicon providers

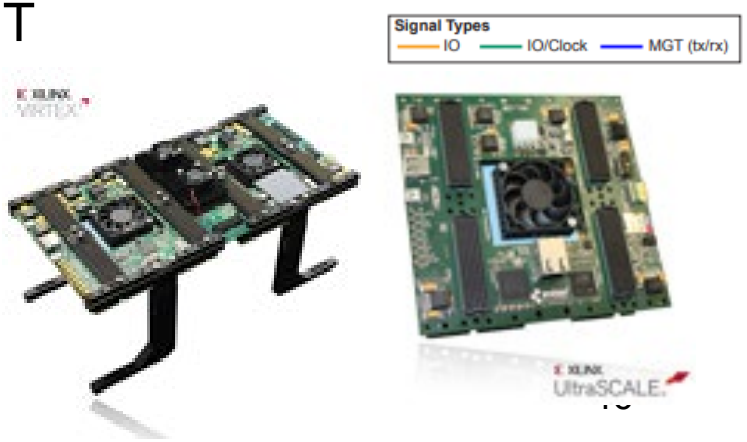
Example for development of apps on DSP (TMDXEVM6455)

- The core of the systems is a couple of processors TMS320C6455, working in floating point with a throughput of 8G inst/sec
  - Systems valid to prototype SoC solutions, since each process is a SoC itself, with many peripheral and interfaces
    - 1 decoder for turbocodes, viterbi decoder, etc
  - Beside the high computing power and the richness of interfaces offered by the two processors, the board integrates also other devices
    - 2 Ethernet interfaces
    - One A/D 24 bit converter
    - One D/A converted with audio quality
- To manage such amount of resource it is necessary a powerful development environment like Code Composer Studio
  - Integrated environment for compiling and linking of DSP applications, with tools for analysis and code optimization, an *instruction set simulator*, a *profiler*, a cache analyzer, etc
    - The systems provides also the BSP (board support package) of the board and a complete library to support the functionality of the chip (CSL)
    - There was also a simple proprietary OS named DSP-Bios, by Texas

proFPGA uno V7 Prototyping System

- ASIC prototyping solution
  - Up to 12 M ASIC gates capacity
  - Up to 1084 available user I/O
  - Up to 36 dedicated high speed serial I/O transceivers
  - Up to 8 individually adjustable voltage regions
  - Up to 1.8 Gbps/12.5 Gbps point to point speed
  - proFPGA FPGA Mixing Technology (FMT
  - Smart Stacking Technology (SST)
  - High performance host interface (DMBI)
  - proFPGA Builder Software
- See 4 boards for 16 Virtex-/ cores @ Polimi HeapLab





proFPGA uno V7 family

profpga uno V7 Specification	
Available FPGA types	- Xilinx Virtex XC7V2000T, XC7VX690T, XC7V585T or XC7VX330
Capacity	- Up to 12 M ASIC gates (XC7V2000T FPGA)
FPGA-internal memory	- Up to 52,920 kbits on one board (XC7VX690T FPGA)
Signaling rate	- Up to 1.8 Gbps (standard I/O)/ 12.5 Gbps (MGT)
Extension sites	- Up to 8 Extension sites with High Performance (up to 21 Gbps) connectors
I/O resources	- Up to 1084 signals for I/O (XC7V2000T FPGA) - Up to 738 signals for I/O (XC7VX690T FPGA) - Up to 738 signals for I/O (XC7V585T FPGA) - Up to 540 signals for I/O (XC7VX330T FPGA)
High speed I/O transceivers	- Up to 16 MGTs (XC7V2000T FPGA) - Up to 32 MGTs (XC7VX690T FPGA) - Up to 36 MGTs (XC7V585T FPGA) - Up to 28 MGTs (XC7VX330T FPGA)
Available interface boards	- USB 3.0, PCIe Gen2/Gen3, MIPI, DVI, DDR3 memory, Gb Ethernet, etc.
Voltage regions	- Up to 8 individually adjustable voltage regions per FPGA Module - Stepless from 0.6V up 3.3V depending on used FPGA type - Automated detection of daughter card and adjustment of right voltage
Clocking	- 8 fixed clocks - 2 quartz as clock references
Configuration	- With host software via Ethernet, USB 2.0 or standalone over USB stick

Data exchange	- On board DMBI (Device Message Box Interface)
	- Data exchange rate:
	  - Ethernet
	  - USB

Power	External (optional) ATX Power Supply (12 V, 24 - 35 A output)
Dimensions	- 5.91" x 0.95" x 5.91" / 150 mm x 24 mm x 150 mm (width x height x depth) - 0.5 kg weight

Platforms for prototyping

- Approach to the design
  - Two main phases
    - From concept to prototype
    - From prototype to product

    - Approach for the prototyping
      - Requirement analysis
      - Identification of the platform and procurement
      - Design
      - Realization of the prototype
      - Verification
    - At the end of the verification, the following step can be the realization of the final system
      - In any case there is the need to adapt e/o redesign some functionalities and to develop the entire system (PCB or SoC)

Conclusions

- Definition and selection of the Hw architecture of an ES is a crucial step
- Several alternative technologies exist
  - PCB-based is the common one
    - Presented the flow, the material, tools and timing
  - SoC – based
    - Presented the process and the critical aspects such as testing and design time
  - Distributed networked ES
    - Low power end possibility to customize the range of sensors
  - Prototyping platforms
    - Enabling element to catch leading edge technologies, to reduce integration risks and shrink time to market