Embedded Systems

2015/2016

Architectures & Design

From:

“Sistemi Embedded: Sviluppo HW e SW per sistemi dedicati (Cap. 2)”

W. Fornaciari, C. Brandolese - Edizioni Pearson – Prentice Hall 2007

and other different sources.

Overview

- Architectures and Design
  - Printed Circuit Board
    - Physical (Discrete) Components
    - Support
    - Components mounting and soldering
    - Design issues
  - System-on-Chip
    - Design issues
      - Design for testability
  - Distributed Embedded Systems
  - Development kits
    - Classifications
    - Design Issues

Architectures and Design

Architectures and Design

- Due to ES heterogeneity and different constraints it is difficult to define a general design methodology
  - No mature “embedded systems engineering” discipline
    - The designer experience, together with empirical criteria and qualitative analysis, plays a very relevant role
      - Modern approaches: *Platform Based Design*
- The definition of the embedded system architecture is one of the most critical design task
  - Errors at this stage could severely impact quality and cost
    - Later their discovering, greater the costs to fix them

Architectures and Design

- Main general design (development) steps
  - Design Flow
    - Requirement analysis
    - Requirement specifications
    - Architecture definition (HW or HW/SW)
    - HW/SW and communication partitioning
    - Implementation
    - Verification and validation
    - Production
      - Post-production test

Architectures and Design

- Main reference architetcures
  - System-on-Board
    - Based on *Printed Circuit Board* (PCB)
      - System-on-Module (SoM)
  - System-on-chip (SoC)
    - (COTS) ASIC and/or FPGA
  - Distributed (Networked) Embedded Systems
- Main support to design and prototyping
  - Development Boards/Kits (Platform)
    - Based on different processor technologies and integrating several devices and peripherals, they allow the designer to (partially) avoid integration and interfacing issues while focusing only on application development

Printed Circuit Board

Printed Circuit Board

- System-on-Board (or System-or-PCB) exploits COTS and/or ad-hoc physical components (or modules) assembled on a plastic board that provides all the needed connections
  - Virtually, all the embedded systems, in their final form, are PCB but a real System-on-Board has its functionality realized by means of several interacting physical components (i.e. distributed among them)
    - Main PCB elements
      - Physical (Discrete) Components
      - Support

Printed Circuit Board

- Physical (Discrete) Components
  - Passive
    - Sockets, switches, LED, resistors, capacitors, inductors, etc.
  - Power systems
  - Converters, filters, amplifiers, oscillators, etc.
  - Electro-optical
    - Decouplers, fotodiodes, etc.
  - RF
  - Displays
  - Sensors
  - Digital components
    - ASIC/PLD (for GPP/ASP/SPP)
  - …

Printed Circuit Board

- Physical (Discrete) Components
  - Given a component often it is possible/needed to select the package properties
    - Mounting technique
      - Through-Hole, Surface Mounted, etc.
    - Pin positioning
      - *In-Line*: SIP, DIP, PDIP, etc.
      - *Small-Outline*: SOJ, SOP, TSOP, etc.
      - *Quad Surface Mount*: CC, QFP, etc.
      - *Grid Array*: PGA, BGA, etc.
    - Material
      - Plastic, ceramic, etc.

Printed Circuit Board

- Support
  - The physical support for the components is the real PCB

    - It is composed of 3 different materials
      - Conductor
      - Insulating
      - Adhesive Material (insulating)

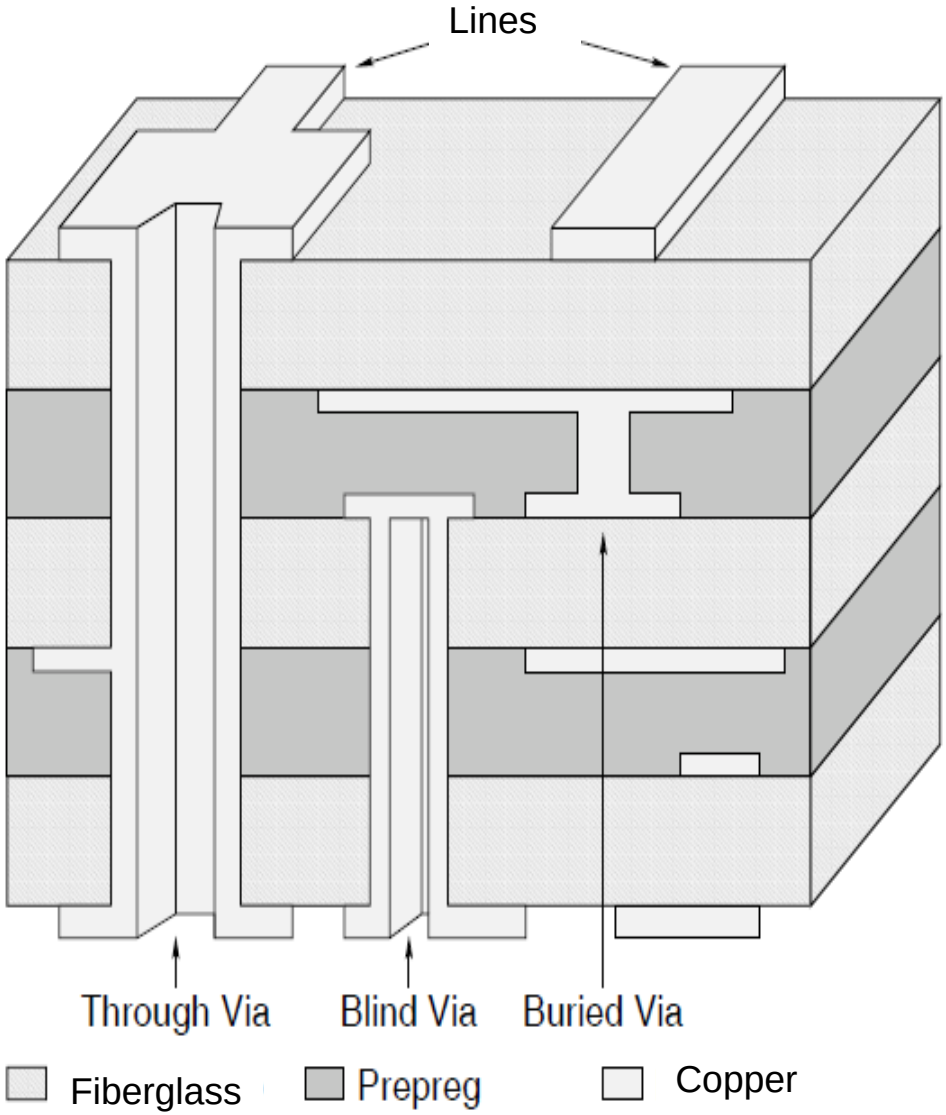
Printed Circuit Board

- Support
  - Conductor
    - It is normally copper used to realize interconnections, power supply and the contact points for the component pins
      - Complex multy-layers PCBs are normally composed of 4-8 layers for power and 6-10 layers for interconnections
  - Insulating
    - Used to insulate interconnections and to provide mechanical support to mount components
      - Fiberglass Epoxy Resin (FR-2, FR-4, etc.), Kevlar, Kapton (flexible)
  - Adhesive
    - Different layers are glued by means of *pre-preg* sheets
      - Solid sheets that becames adhesive at specific pressure/temperature conditions
        - » They go back to solid state keppiong required electrical/chemical/mechanical properties

Printed Circuit Board

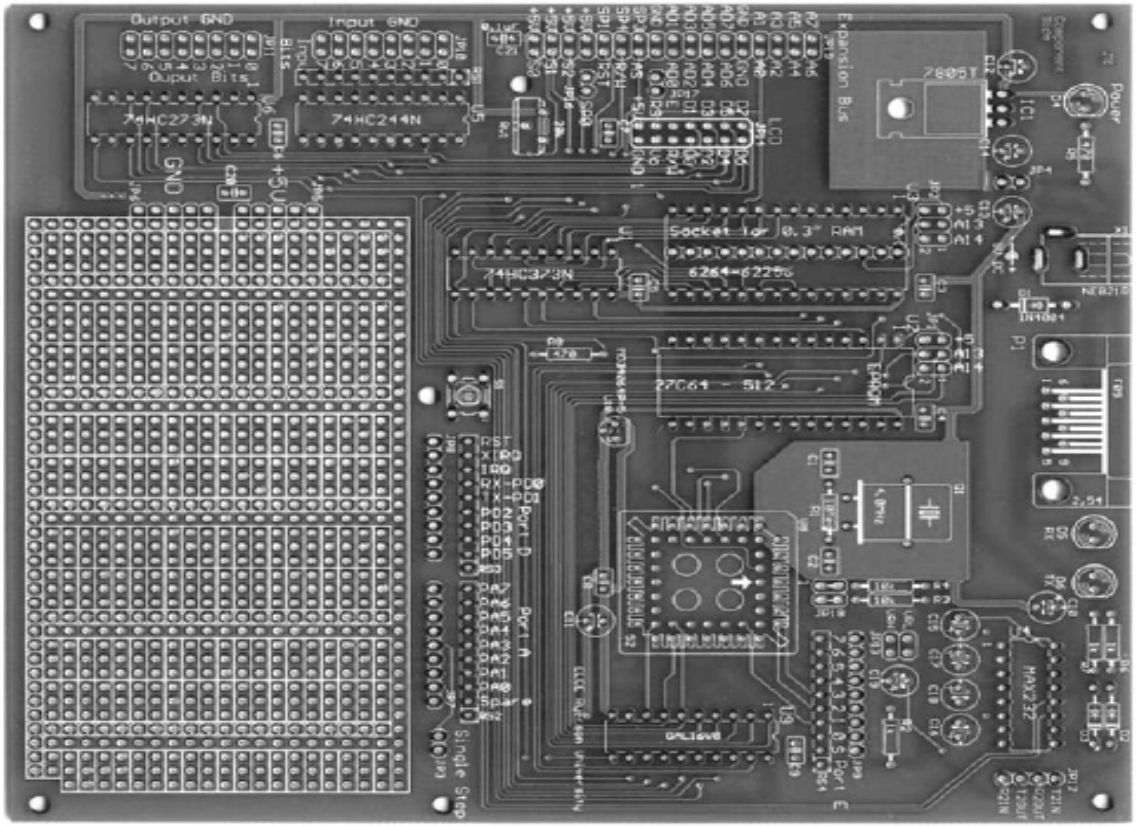
- Support manufacturing is a complex activity
  - Normally performed by specialized companies
    - The starting point is a file representing the output of the design
      - Gerber, Excellon, etc.
    - Main steps
      - Manufacturing of layers
      - Realization of interconnections
        - » By means of a proper resin (*photoresist*) it is possible to create acid-resistant lines
      - Assembling
        - » Different layers are stacked interleaved with pre-preg and properly pressed
      - Drilling (*via*)
      - Manufacturing of external layers
      - Refinement

Printed Circuit Board



Printed Circuit Board

- Support manufacturing result



Printed Circuit Board

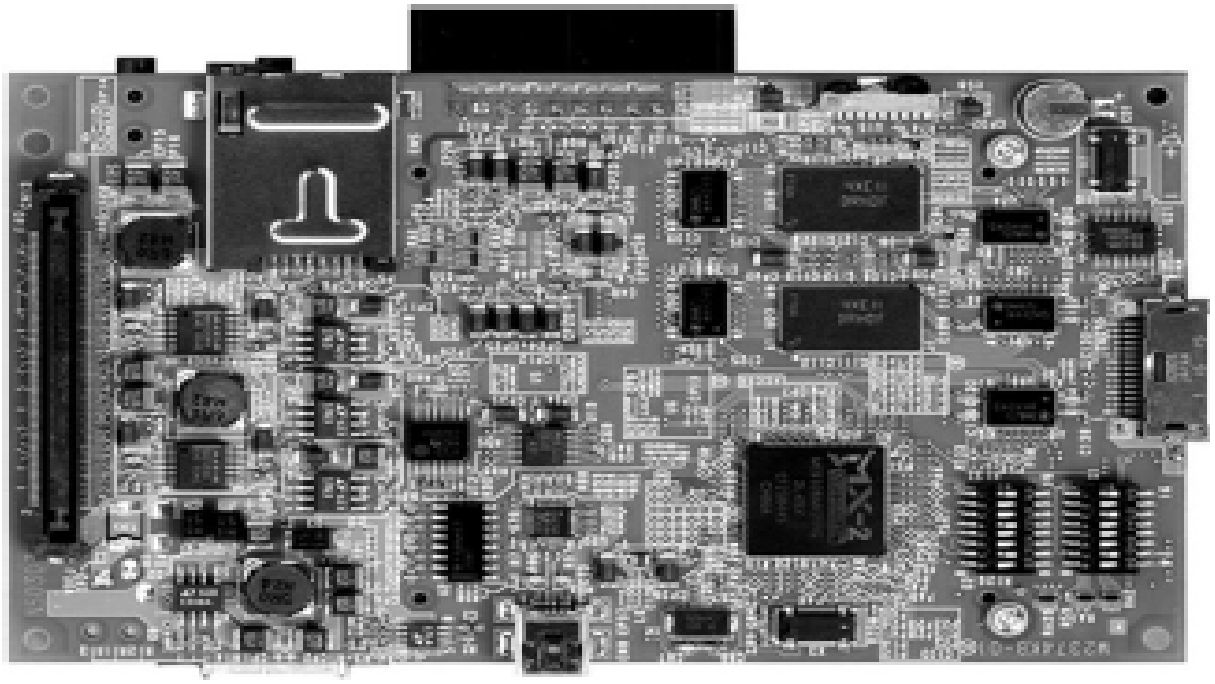
- After support manufacturing it is needed to perform the electrical test (*bare-board testing*) to check if lines and contact points are all connected
  - Bed-of-nails tester
  - Flying-probe tester
  - X-ray analysis
    - *Image recognition*
- Board-testing is a slow and expensive process
  - As it is explained later, there is the need for design techniques that integrates in the PCB itself a non-functional section to support testing

Printed Circuit Board

- Components mounting and soldering
  - When the support is ready and tested it’s time to mount and solder physical (discrete) components
    - For small, simple, and not many PCBs (e.g. protoypes) M&S could be performed manually by specialized technicians
    - Otherwise, there is the need for specialized and expensive equipment
      - For such a reason, a PCB designer limits his work to the “only” design and exploit a specialized companies for manufacturing

Printed Circuit Board

- Final result



Printed Circuit Board

- Design Issues
  - A PCB is a complex and heterogenous system
    - It is important to plan activities to allow concurrent work until final system integration and verification
      - The first step is the partitioning of funcionality with respect to HW and SW and with respect to the available components (allocation)
        - » This allows to define I/O logical signals for each single component and to map them to component physical pins (pin-out)
      - Partitioning criteria are several and related to F/NF issues
        - » Number of signal needed to connect diffferent components
        - » Bandwidth of involved signals
        - » Delay of signal that are routed on the board
        - » …
  - Once fixed partitioning and pin-out, several design teams can work concurrently on different parts

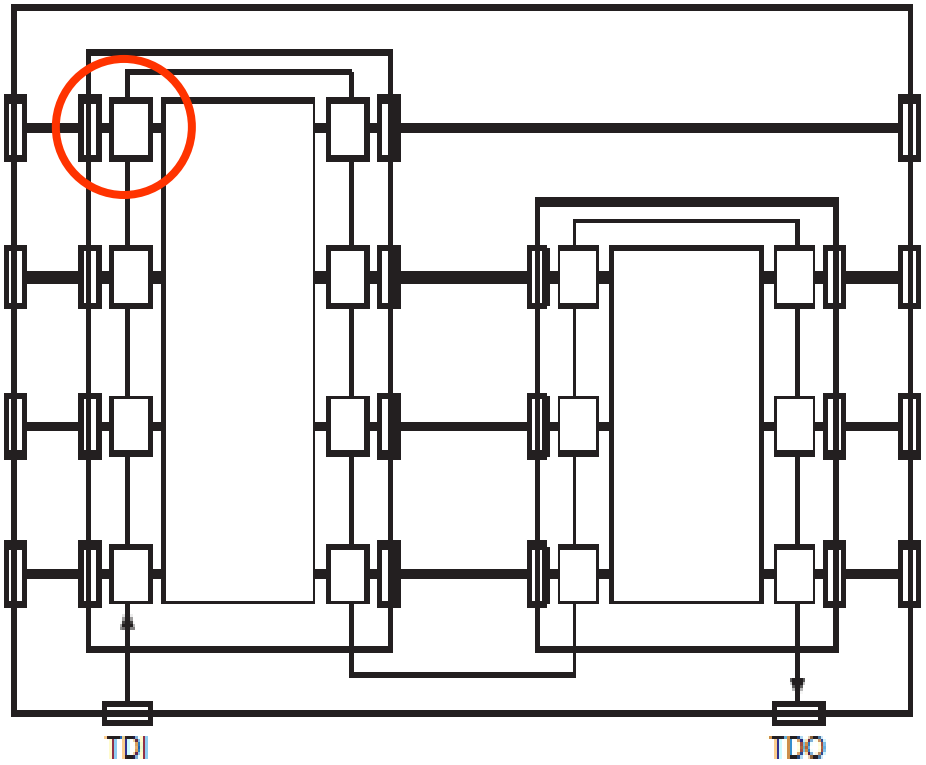
Printed Circuit Board

- Design Issues
  - As anticipated before, board testing is a critical process

    - *Boundary scan testing*
      - Methodology that require the realization on the board a chain of flip-flop and multiplexers (*scan chain*) connecting the pins of the components
        - » Normal mode: the scan chain is transparent and the involved pins are connected to real system I/O
        - » Test mode: the scan chain behaves as a “big shift register” that allows to write/read specific logic values in specific pins from the external
      - As it is shown layer, the same approach is used for integrated components on System-On-Chip testing
        - » Chip-level boundary scan testing

Printed Circuit Board

- Design Issues
  - *Board-level boundary scan testing*



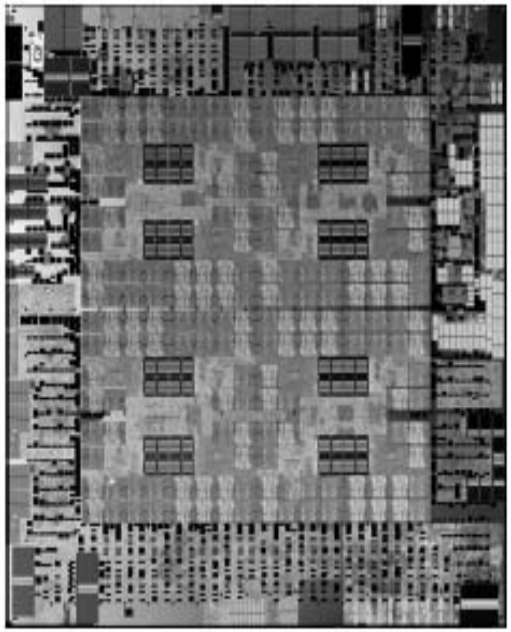
System-on-Chip

System-on-Chip

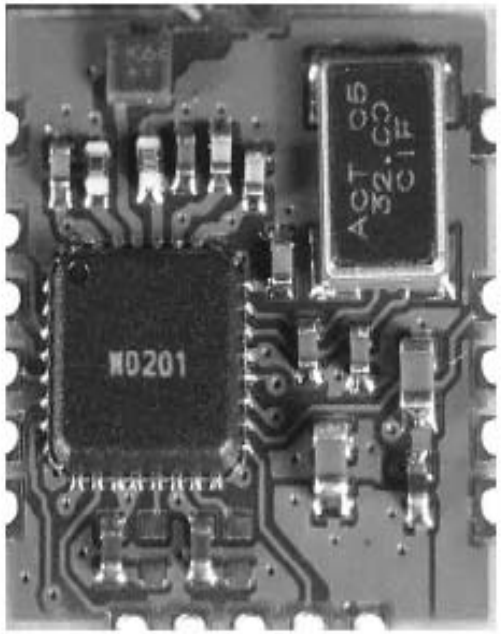
(System-on-Programmable-Chip)

- SoC are architectures based on a single chip (i.e. a single IC) that implements the whole system
  - Practically (mainly for electrical/mechanical issues), some portions of the system are outside the main chip
    - Connectors, antennas, buttons, leds, slots for external cards, etc.)
      - They are built on a simple PCB
        - » PCB also acts as a mechanical support for the SoC
  - The real difference with a System-On-Board is the design approach that shall be able to exploit all the features of integrated technologies
    - For this, the design is mainly based on logical components that become physical only inside the final chip

System-on-Chip



SoC die



SoC mounted on a board

System-on-Chip

- Main reasons for a SoC approach
  - Unit cost, Performance, Power/Energy, I/O, Security,…
- Main elements on SoC architectures
  - One or more heterogeneous GPP/ASP/SPP processors
  - One or more different memory types
  - One or mere power and clocking units

System-on-Chip

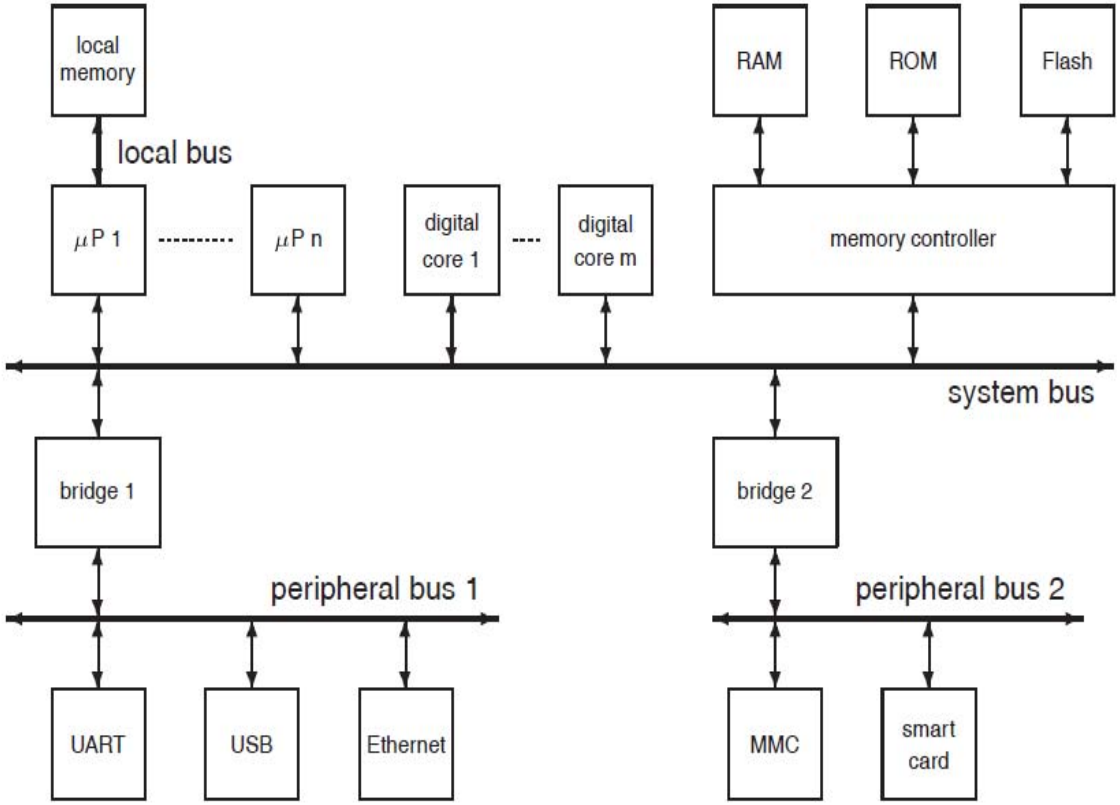
- Design Issues
  - As for SoB the firts step is the HW/SW partitiong
    - Since all the functionalities are realized inside the same IC technology, partitioning is mainly driven by functional/performance criteria
      - Connectivity is a less critical issue
    - Once defined the HW/SW partitioning, there is the need to perform allocation/mapping between functionalities and specific processors
    - Then, it is needed to define memory and communication architecture

System-on-Chip

- Design Issues
  - Memory architecture
    - Depending on the selected allocation/mapping there is the need to define the memory hierarchy and the access policies
      - Centralized vs Distributed
        - » D: good local performance, needs to manage comm, synch and coherency
        - » C: less coherency problems, simple comms, medium access problem
  - Communication architecture
    - Traditional hierarchical buses approaches vs Network-on-Chip
      - NoC
        - » A little network with N/I and packet routing
        - » More complex and more resource consuming but they simplify synchronization among different clock domains and they are more scalable

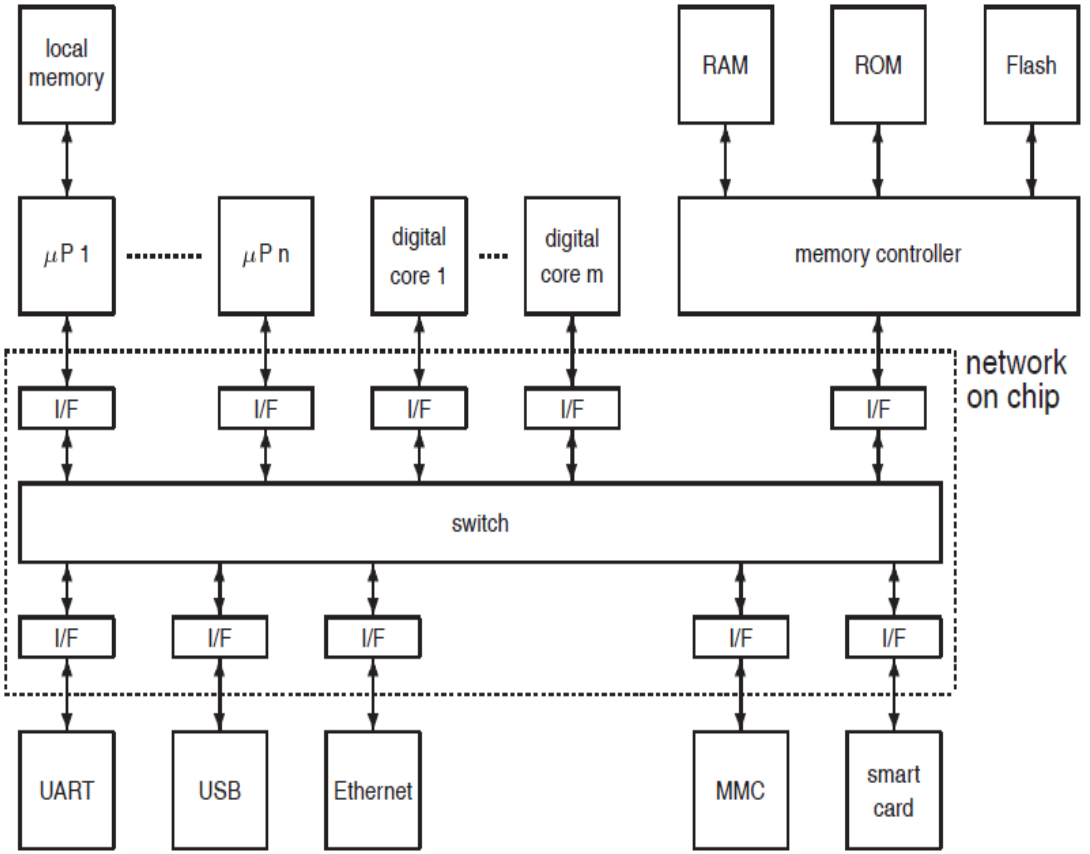
System-on-Chip

- Design Issues
  - Architecture based on Hierarchical buses



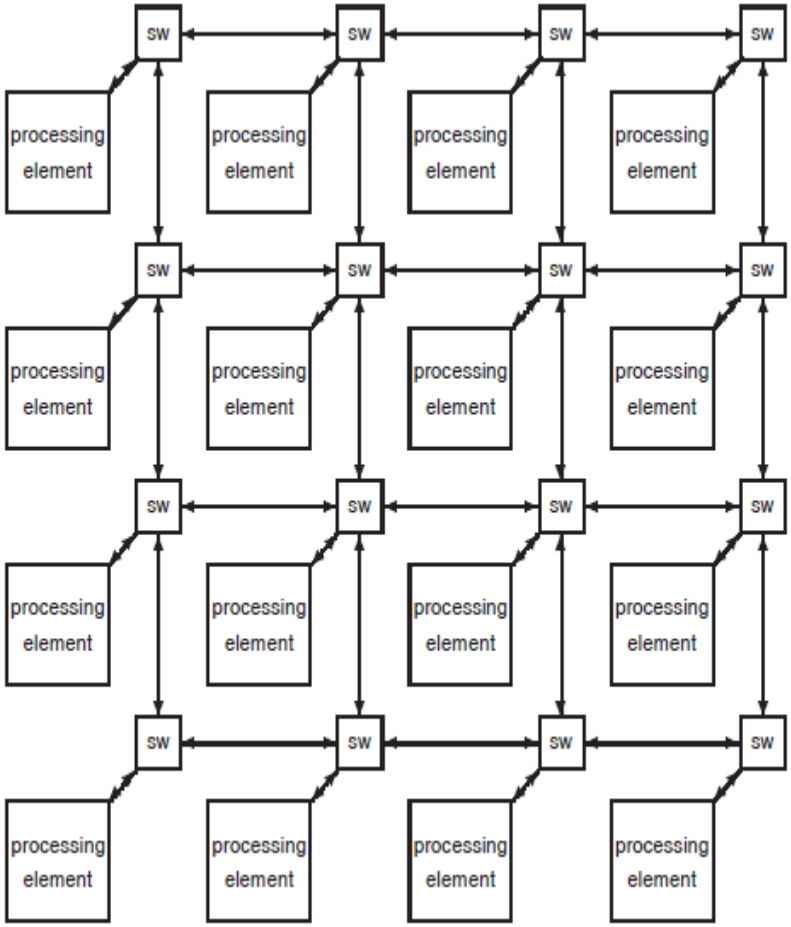
System-on-Chip

- Design Issues
  - Centralized NoC



System-on-Chip

- Design Issues
  - Meshed NoC



System-on-Chip

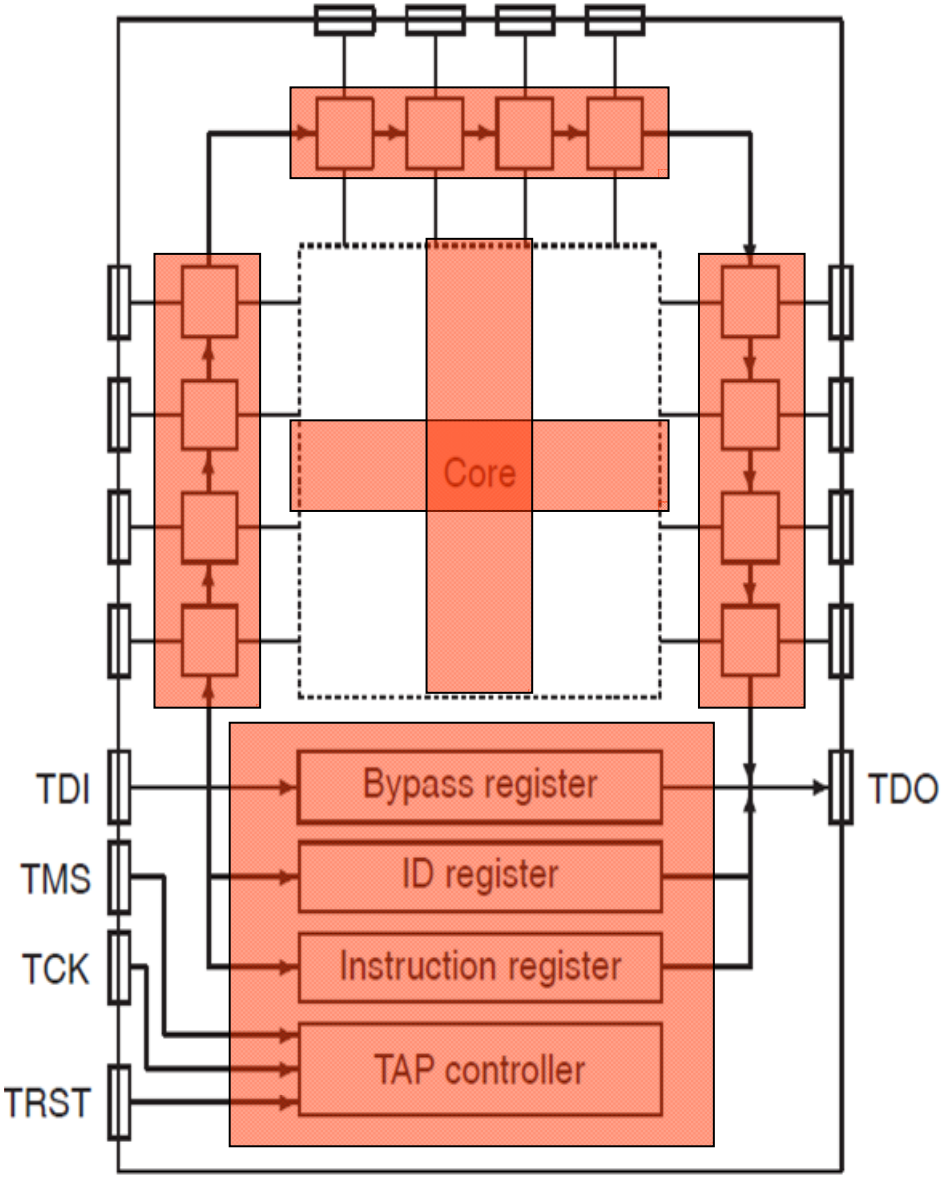
- Design for testability
  - SoC: hard verification problem
    - Components don’t physically exist independently and so cannot be tested separately before integration!
      - As we will se later, one solution is to exploit development kits, but the problem is also tackled during the design phase
  - DFT: design methodology that aims at simplifying testing activities by introducing additional components
    - Boundary, partial e full scan
      - For complex testing procedures it is needed a configurable scan chain
      - A solution is provided by the standard IEEE 1149.1 JTAG (Joint Test Action Group) standard
        - » TAP controller (*Test Access Port controller*): a component interprets a simple protocol constituted by some configuration commands

System-on-Chip

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

e.g.

Instruction	IR Value	Registers	Function
EXTEST	000000	Boundary	PCB Interconnect test
SAMPLE	000001	Boundary	Sample and data preload
BYPASS	111111	Bypass	Bypass mode
SEL_INT_SCAN	010000	Internal Scan Reg.	Scan IIep internal register
SEL_DBG_SCAN	011111	Internal Scan Reg.	Scan IIep internal register
IDCODE	100000	JTAG ID Reg.	Scan the ID register
SEL_CCR	011110	Clock Control Reg.	Set up clock control
CLK_RST	100000	Bypass	Reset Clock Control

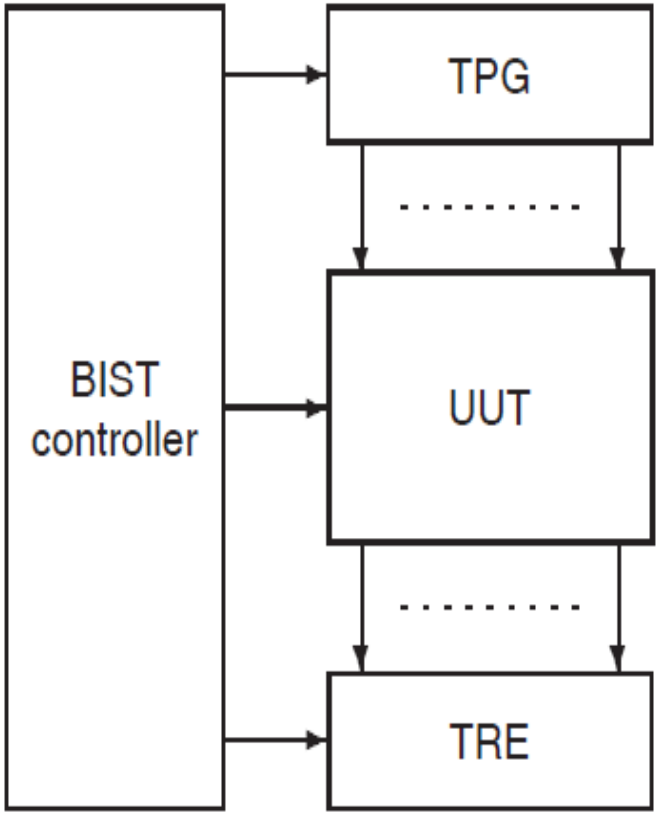


System-on-Chip

- Design for testability
  - Bult-in Self-Test (BIST)
    - Test based on scan chains could be not efficient in very complex systems since they have to be controlled form the outside and often they are performed at a frequency lower than the operative one
    - To overcome such a limitation they have been developed a technique based on automatic internal verification (BIST)
      - Integrated architecture composed of a *Test Pattern Generator* (TPG) that provides *test vectors* then applied to the Unit Under Test (UUT) under the control of the *BIST Controller*
      - Outputs from UUT are analyzed by an *Output Response Analyzer* (ORA) also called *Test Response Evaluator* (TRE)
    - Normally, the BIST mode is executed at the same (or near) operative frequency used for the nominal device

System-on-Chip

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



Distributed Embedded Systems

Distributed Embedded Systems

- They are systems that provide a complex functionality by means of several separate sub-systems that interacts by wired and/or wireless networks
  - Processing and storing tasks are distributed
    - *Distributed vs. Networked*
- When the network infrastructure is not trivial, traditional design approaches are not sufficient and should be integrated with specific techniques
  - Once faced the whole problem, it is then possible to follow traditional design methodologies for SoB/SoC

Distributed Embedded Systems

- Two main categories (often mixed)
  - Wired Systems
    - Fixed and well-known network infrastructure
  - Wireless Systems
    - More flexible and complex network infrastructure
- Typical distributed applications
  - Home Automation and Automotive
    - A lot of emerging interest for the so called *Sensor Networks*
      - In particular, *Wireless Sensors Networks*, are more and more exploited in several application domains
        - » Healthcare, Environmental Monitoring, Cultural Heritage Monitoring, etc.

Development Kits

Development Kits

- Architecture definition and technologies selection are of critical importance on system design
- The next step is to proceed with development and verification of system functionalities
  - ISS/HDL Co-Simulation
    - HW/SW Co-Simulation environments provide support in the early steps but they are not suitable for a complete verification
      - For some COTS components are not available proper models
      - Very long time is needed to verify and validate realistic complex systems

Development Kits

- For this, it is needed to exploit development environments that provides proper **development kits**
  - Configurable platforms useful to rapidly develop first prototypal versions of the system under development
    - They allow also to perform meaningful verification/validation
- DK are normally composed of GPP/ASP, SPP, memories, OS and also applicative SW
  - They can be inserted in a realistic environments for meaningful verification/validation
    - This approach is called **emulation** and, given the complexity of modern systems, is quite always mandatory in an efficient and effective design methodology

Development Kits

- Classification
  - Emulators for simple digital systems
    - DK that integrate a configurable device (FPGA) and some basic standard digital interfaces (RS-232, USB, JTAG)
      - Used for verification/validation of basic functionalities
  - Emulators for complex digital systems
    - DK with one or more configurable digital devices, external memories (RAM, E2PROM, Flash), several interfaces (PCI, PCI Express, Serial ATA, Ethernet, FireWire, GPIO), clock and power units, A/D and D/A converters
      - Used for verification/validation of complex SoB/SoC

Development Kits

- Classification
  - Emulators for software systems
    - DK with one ore more microprocessors (or microcontrollers) and some interfaces
      - Often provided with a basic layer of firmware and software (HAL/BSP), an OS and some libraries to exploit integrated peripherals
  - Emulators for hardware/software systems
    - DK with different complexities that integrate one or more configurable/programmable digital devices
      - Often provided with a basic layer of firmware and software (HAL/BSP), an OS and some libraries to exploit integrated peripherals

Development Kits

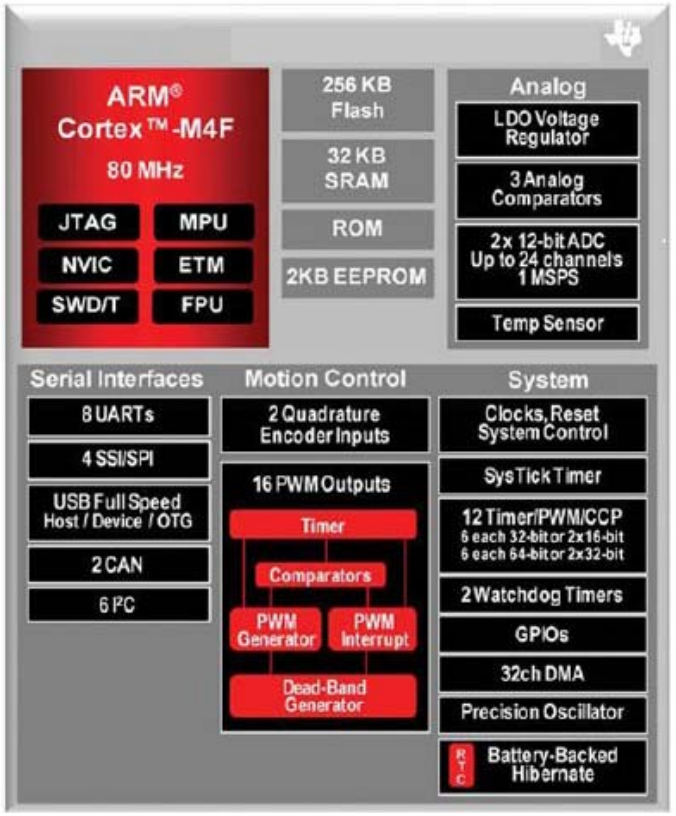
- Classification
  - Emulators for SoC
    - Similar to the previous one but with the components integrated on a single-chip
      - Often composed of high-end FPGA with some fuse (hardwired) GPP/ASP, SPP, memories and interfaces
  - Emulators for Application-Specific systems
    - DK for network applications provided of several interfaces and NP
    - DK for Wireless Sensor Networks provided of nodes and gateways
    - DK for multimedia applications with audio/video encoder/decoder
    - DK for *Software Defined Radio* (SDR) or True Software Radio (TSR) systems

Development Kits

- Classification
  - Example
    - Digilent Xilinx Spartan 3AN Starter Kit

Development Kits

- Classification
  - Example
    - *Texas Instruments TIVA TM4C123G*



Development Kits

- Classification
  - Example
    - Xilinx ZedBoard (Zynq-7000)

Development Kits

- Classificaton
  - Example
    - *Texas Instruments* DSP DK (TMDXEVM6455)

Development Kits

- Classification
  - Other examples
    - Software Defined Radio DK (Sundance)
      - SDR_3.0/WiMax_2.0
        - » http://www.kanecomputing.co.uk/sundance_SDR-3.htm
        - » http://www.ti.com.cn/devnet/docs/catalog/thirdpartydevtoolfolder.tsp?actionPerformed=productFolder&productId=7921
    - WSN DK
      - http://www.memsic.com/wireless-sensor-networks/
    - Sierra Wireless Q26 Extreme DK
      - http://developer.sierrawireless.com/
    - OMAP L138 DK
      - http://www.logicpd.com/products/system-on-modules/zoom-omap-l138-evm/

Development Kits

- Design Issues
  - DK-based design methodologies are composed of two main phases
    - from the concept to the prototype
    - from the prototype to the product

    - Main steps
      - Requirements analysis and specification
      - DK identification and retrieval (or development)
      - Design
      - Prototype realization
      - Verification/Validation
      - Final system development
        - » DK adaptation or re-design of SoB/SoC