System-on-Chip: Architectures and Design

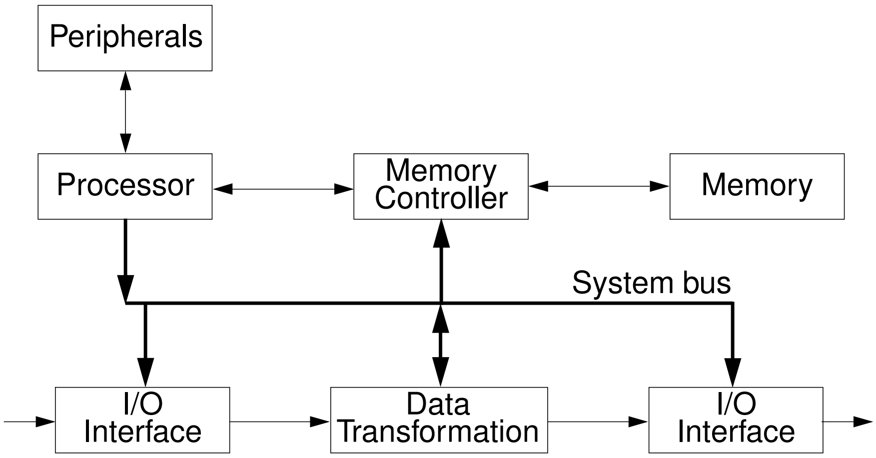
1. What is System-on-Chip?
2. Advantages and Problems
3. Core-Based Design
4. System Design Flow
5. Examples

System-on-Chip: What is it?

- Something with a lot of transistors in it (how many?)
- Something which solves an application and contains everything needed for it (what about human interface?)
- Something which is heterogeneous (hardware/software, controller/data-flow, digital/analog, ...)
- Something which contains more than a “simple” processor (processor + memory, I/O, ASIC, A/D converter, ...)

Typical System-on-Chip

A “generic” design:



Typical System-on-Chip (cont’d)

Components of the generic design:

- A microprocessor and its memory subsystem
  - could be an 8-bit microcontroller core or a 64-bit RISC;
  - the memory system can be single or multi-levelled.
- A datapath with interfaces to the external system
  - the external interfaces can be bus drivers, Ethernet interfaces, A/D - D/A, electro-mechanical or electro-optical converters.
- Blocks performing transformations on data received from the external system
  - can be implemented as ASICS and/or DSP cores.
- Interface logic to peripherals

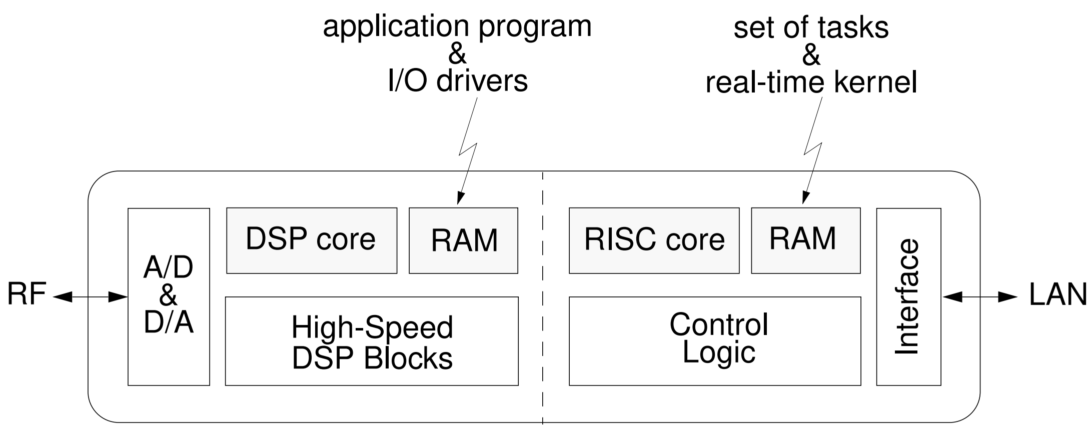
Typical System-on-Chip (cont'd)

☞ More and more programmable cores:
- Algorithms and protocols become increasingly complex (which makes implementation in hardware difficult).
- Processors are fast, as result of sophisticated design.
- Flexibility: upgrading is easy, even "on field".

☞ ASICS are still there (and will be):
- Power consumption
- Performance

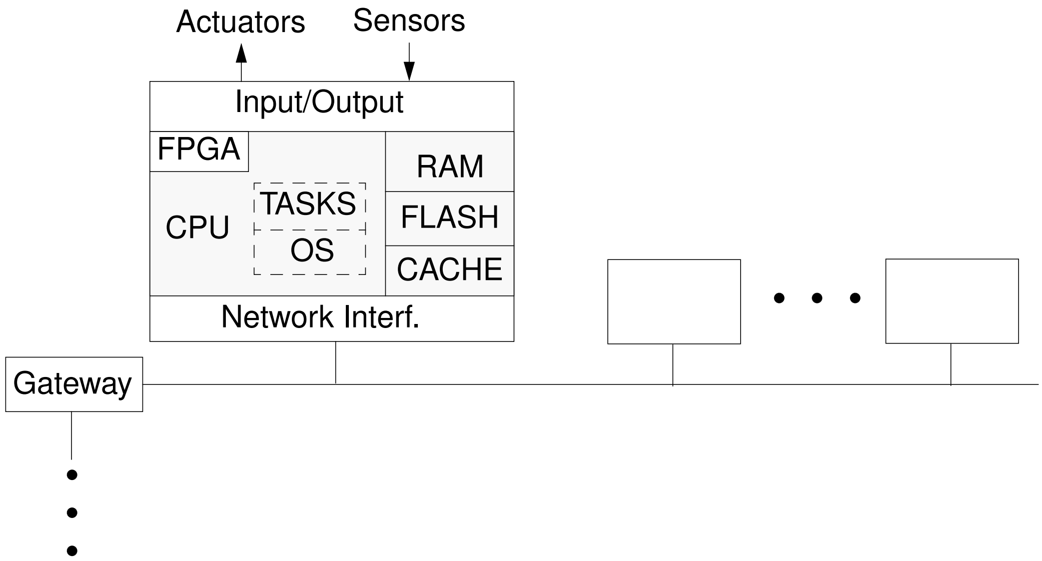
Typical System-on-Chip (cont'd)

A typical structure for a telecommunication (wireless) application:



Typical System-on-Chip (cont'd)

A node (ECU) in an automotive application can also be seen as a SoC:



Why Do We Need Systems-on-Chip

- Now SoC became technologically possible.
- Higher performance (fast data transfer compared to multi-chip design).
- Potential of innovative architectures.
- Low power (multi-chip designs need additional drivers and bus interface for inter-chip and inter-board connection).
- Reduced size.
- Low cost.
- Increased reliability.

Problems with System-on-Chip Design

☞ Heterogeneity
- Analog - digital
- Processors - ASICS - Programmable logic - Memories
- Different memory types (EEPROM, ROM, RAM, DRAM)
- ....

This creates problems with:
- Design: expertise in very different design styles and areas.
- Technology: several processes have to be incorporated onto a single die.

Problems with System-on-Chip Design (cont'd)

☞ Design productivity vs. time-to-market pressure

- Design productivity is much behind of what the semiconductor industry can deliver.
- Yearly increase:
  - in semiconductor technology - 58%
  - in design productivity - 28%.

The Solution: Core-Based Design

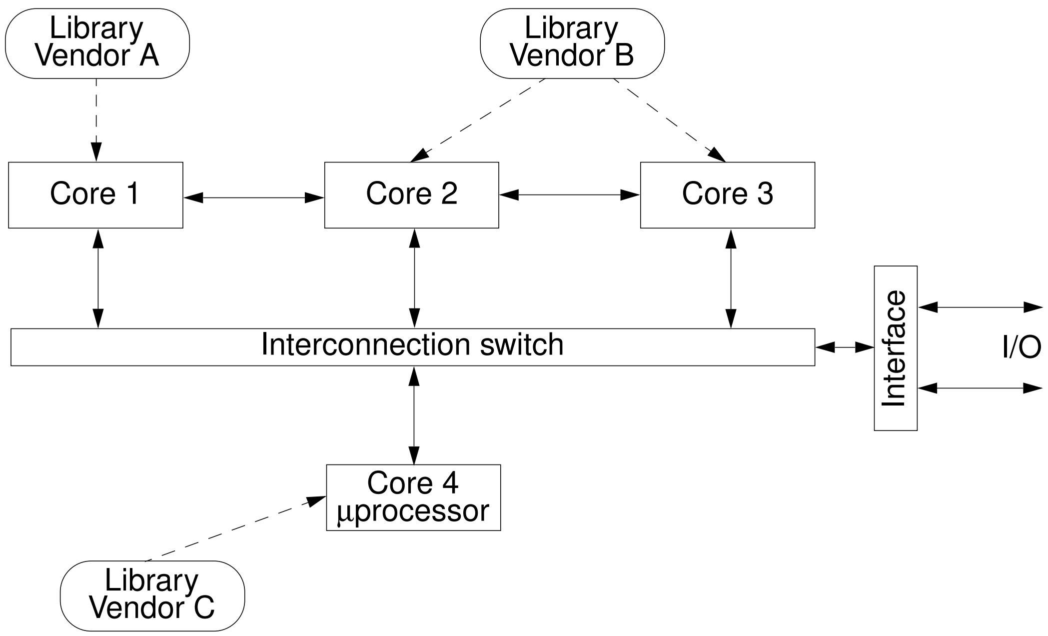
Core-based design ≡ IP-based design ≅ SoC-design

Cores:filters, sorters, timers, memories, interfaces, encoders/ decoders, microcontrollers, DSPs, microprocessors, ...

Possible definition

- A core is a design block which is larger than a typical RTL component.

Core-Based Design



Core-Based Design (cont'd)

☞ Core-based design is the process of composing a new system design by **reusing** existing components.

Problem:

How to specify an existing core for the reuse library?

- functionality
- timing
- interface properties
- power consumption, ...

Some essential design issues in this context:

- specification, simulation, estimation, exploration
- integration (interfaces)
- verification, testing

Types of Cores

- Hard cores: are fully designed, placed, and routed by the supplier.

A completely validated layout with definite timing

rapid integration | low flexibility

- Firm cores: technology-mapped gate-level netlists.

less predictability | flexibility during place and route

Types of Cores (cont'd)

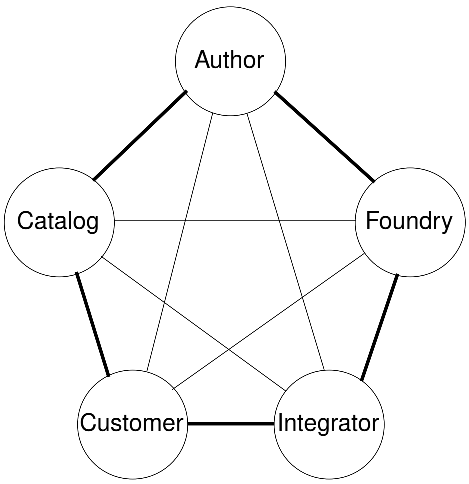
- Soft cores: synthesizable RTL or behavioral descriptions.

⇩ much work with integration and verification.

⇩ maximal flexibility

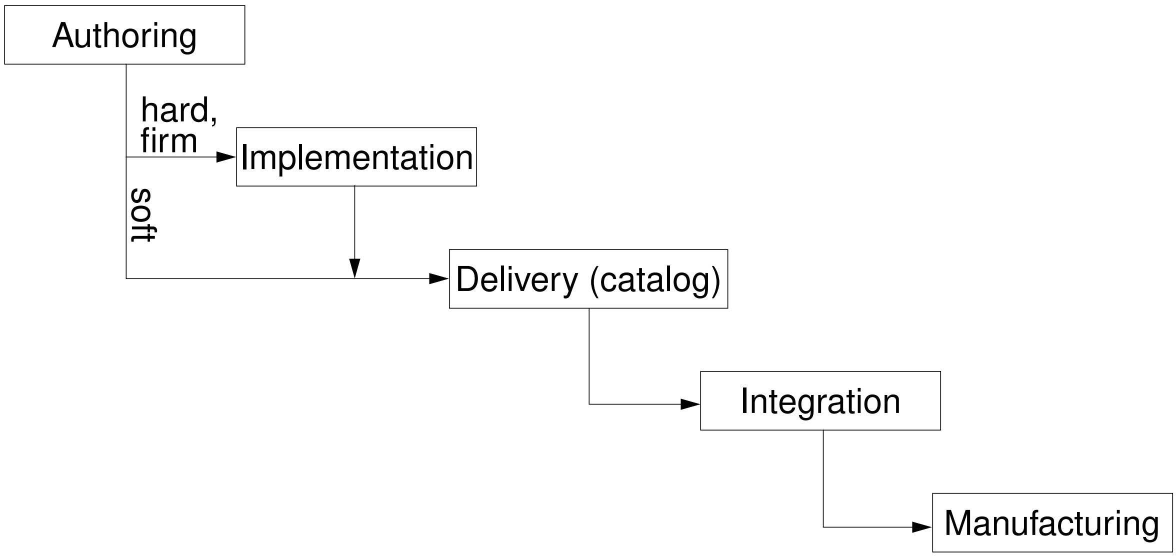
Flexibility can provide opportunities like e.g. adding application specific instructions to a processor core by modifying the behavioral description (see lecture on ASIPs).

Core-Based Design - The Players



- Author: creator of the core.
- Foundry: manufacturer
- Catalog: enables customers to find and select cores.
- Integrator: designs the chip for the customer.

The Core Lifecycle



The Core Lifecycle (cont'd)

☞ Authoring

- Find idea.
- Create models and verify (formal verification & simulation).
- Create synthesizable specification of the validated model.
- Create simulation models to be delivered with the core (protection!).
- Customizable (parameterized) cores

The Core Lifecycle (cont'd)

☞ Implementation
- Create hard or firm core

☞ Delivery
- Extract design attributes and characteristics (functionality, interfacing, timing, power consumption, technology related aspects, compatibilities, etc.).
- Create a catalog and adequate search engines.

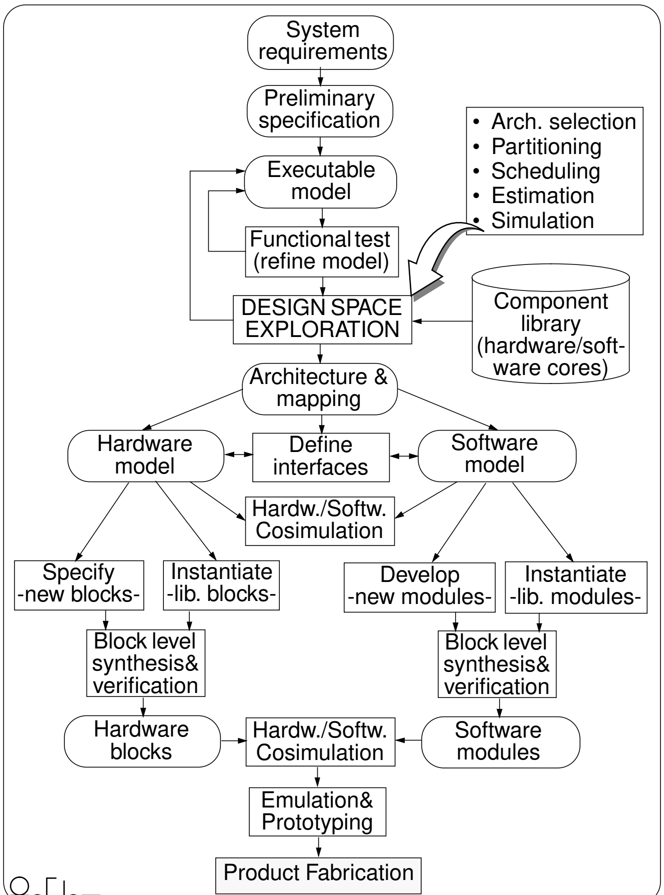
The Core Lifecycle (cont'd)

☞ Integration (*design by composition*)
- Customize cores and instantiate
- Build interfaces between cores
- Use interface synthesis to generate glue logic and interconnection software.
- Verify (formal verification & simulation).
- Design for test (create a core-based design that is testable *after* production)

☞ Manufacturing
- Fabrication of the IC and production testing

System Design Flow

- Core-based design is also a solution for improving software design productivity.
- Reuse techniques and libraries of predefined modules (classes, functions) are not new in software.
  However, several problems are unsolved in the context of a system level hardware/software design environment (portability, estimation, compilation, new standards, ...).
- When commercially available, software cores are called COTS (commercial of the shelf).

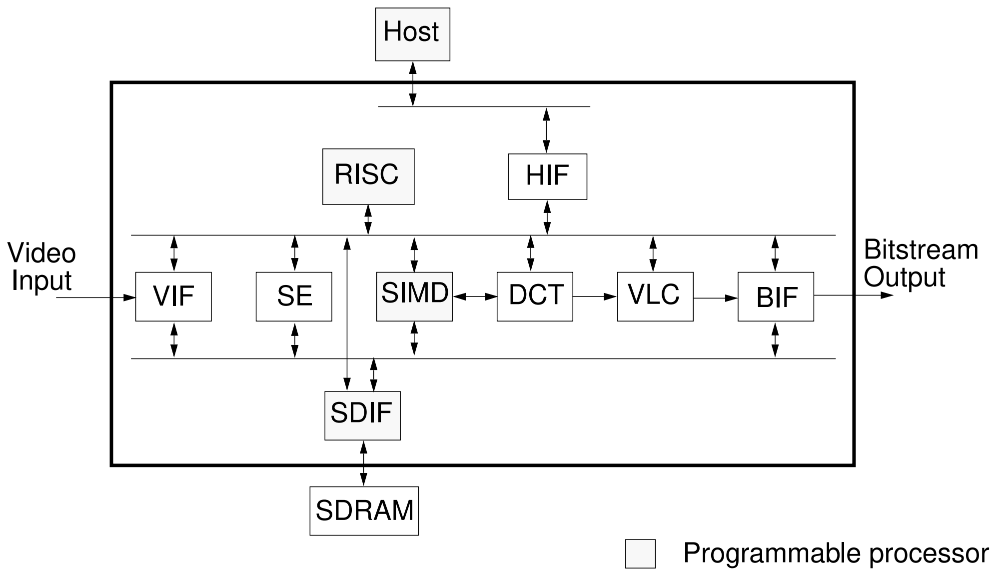


System-on-Chip for Video Encoding (by NTT)

MPEG-2 Encoding

- The encoder receives video input and outputs a bitstream.
- ME (motion estimation) + MC (motion compensation): find out where a macroblock has moved to, when comparing the current picture with a reference; remove temporal redundancy.
- DCT: discrete cosine transform.
- VLC: variable length coding.
- BIF: bitstream interface.
- VIF: video interface
- HIF: host interface
- SDIF: SDRAM interface

System-on-Chip for Video Encoding (cont'd)



System-on-Chip for Video Encoding (cont'd)

☞ Functions which require flexibility and programmability have been mapped on programmable processors:
- A RISC processor controls the overall encoding process.
- ME + MC are partitioned between a dedicated hardware (search engine - SE) and a programmable SIMD machine.
- Pictures are stored on an external SDRAM; buffering and data-flow to/from SDRAM are controlled by a programmable device.

☞ RISC, HIF, VIF, DCT, VLC are reused cores.

☞ Some features:
- Size: 9.8 x 9.8 mm^2 (die size); 0.25 CMOS technology
- Transistors: 5 M
- Clock Frequency: 81 MHz

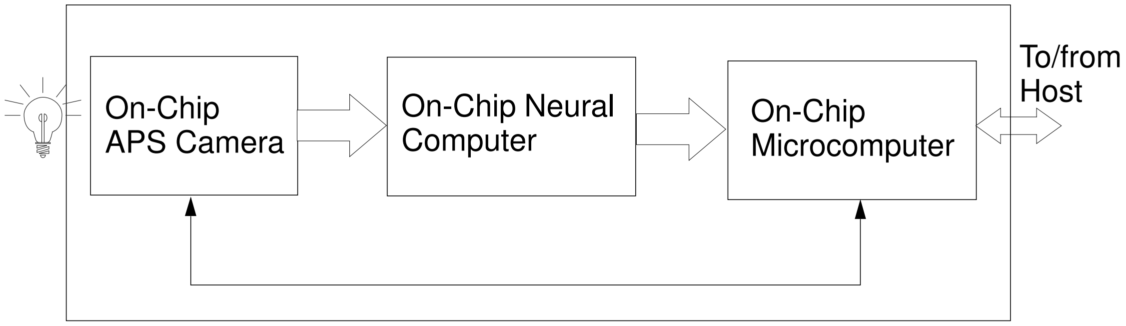
System-on-Chip for Smart Vision (by NASA)

☞ An ultra-fast (real-time) fast vision system on-chip.

☞ The system is programmable and capable of image acquisition, image fusion, image analysis, scene interpretation.

☞ Extremely high performance is due to massively parallel computing structures and high data throughput rates.

System-on-Chip for Smart Vision (cont'd)



System-on-Chip for Smart Vision (cont'd)

☞ 1024 x 1024 APS (Active Pixel Sensor) camera
- Optical sensing array (CMOS technology, fotodetection and read-out amplifier for each pixel).
- captures 30 frames per second.

☞ Neural computer
- Programmable neural processor + programmable synaptic weight memory + learning co-processor.
- Performs image generation, image fusion, and rough image analysis at very high speed.

System-on-Chip for Smart Vision (cont'd)

☞ On chip microcomputer

- PowerPC microprocessor + data memory + program memory + I/O modules + bus interface.
- Performs scene interpretation + control of the system operation.

☞ Some features:

- 3 x 3 cm^2 chip size
- 0.18 CMOS technology

Summary

- With the current technology and in the context of increasingly complex applications and strong market pressure, System-on-Chip is a natural approach for several application areas.
- Programmable components provide the necessary flexibility; ASICS are needed for performance and low power.
- SoC's are highly heterogeneous.
- The real bottleneck is design productivity; core-based design (reuse) is one part of the solution.
- With core-based design the focus is on a system-level view. main issues: specification, exploration, integration, verification, testing.