Computer Architectures

Overview of SOC Architecture design

Tien-Fu Chen

National Chung Cheng Univ.

SOC design Issues

- **SOC architecture**
- **Reconfigurable**
  - System-level
  - Programmable processors
  - Low-level reconfiguration
- **On-chip bus**
- **Embedded Software Issues**

Embedded Systems vs. General Purpose Computing - 1

❑Embedded System

❑ Runs a few applications often known at design time

❑ Not end-user programmable

❑ Operates in fixed run-time constraints, additional performance may not be useful/valuable

❑General purpose computing

❑Intended to run a fully general set of applications

❑ End-user programmable

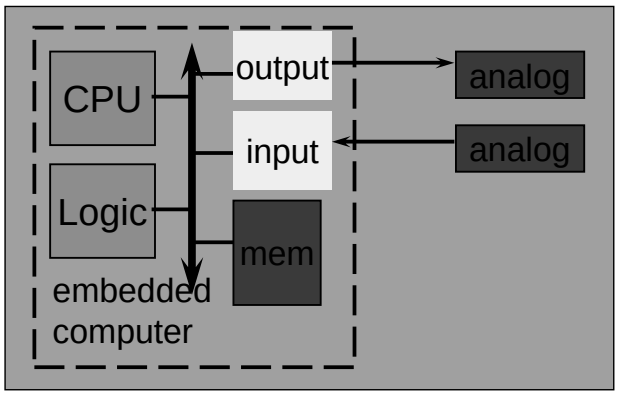
❑ Faster is always better

Embedded Systems vs. General Purpose Computing - 2

❑Embedded System

❑Differentiating features:

- power
- cost
- speed (must be predictable)

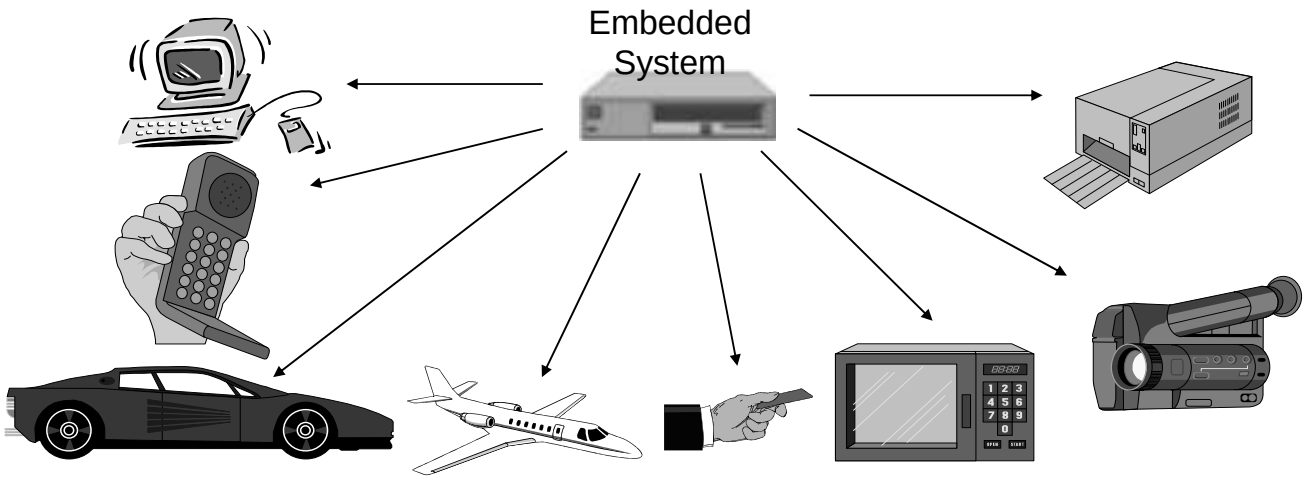


❑General purpose computing

❑Differentiating features

- speed (need not be fully predictable)
- speed
- did we mention speed?
- cost (largest component power)

Embedded System: Examples

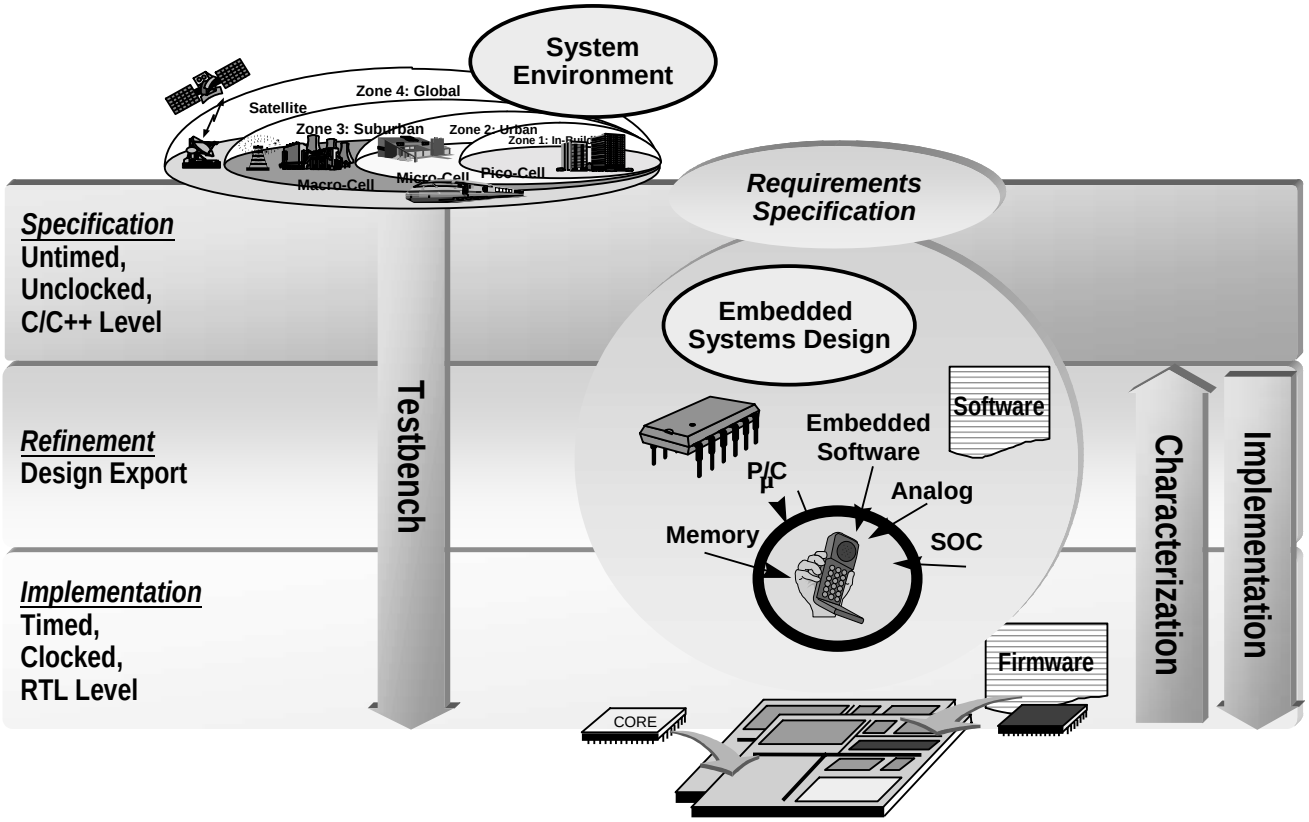


Design Complexity Increase

SoC Characteristics	Current Design	Next Design
Average Gate Count	**616K**	**1,000K**
Designs with Gate Count > 1M	**18%**	**31%**
Units Shipped >5M	**20%**	**37%**
# of lines of DSP SW	**54K**	**295K**
# of lines of uP SW	**75K**	**266K**
Clock Speed >133mhz	**44%**	**61%**
>400mhz	**15%**	**22%**
>5 Clock Domains	**25%**	**35%**
>9 Clock Domains	**10%**	**12%**

Source: Collett International Research- December 2000 Research on 360 IC/ASIC Design Teams in North America

Embedded System on Chip (SoC) Design



Architectures

- **Supplements *Models* by specifying how the system will actually be implemented**
- **Goal of each architecture is to describe**
  - Number of components
  - Type of each component
  - Type of each connection among above components
- **General classification**
  - Application-specific architectures: DSP
  - General-purpose architectures: CISC, RISC
  - Parallel processors: VLIW, SIMD, MIMD

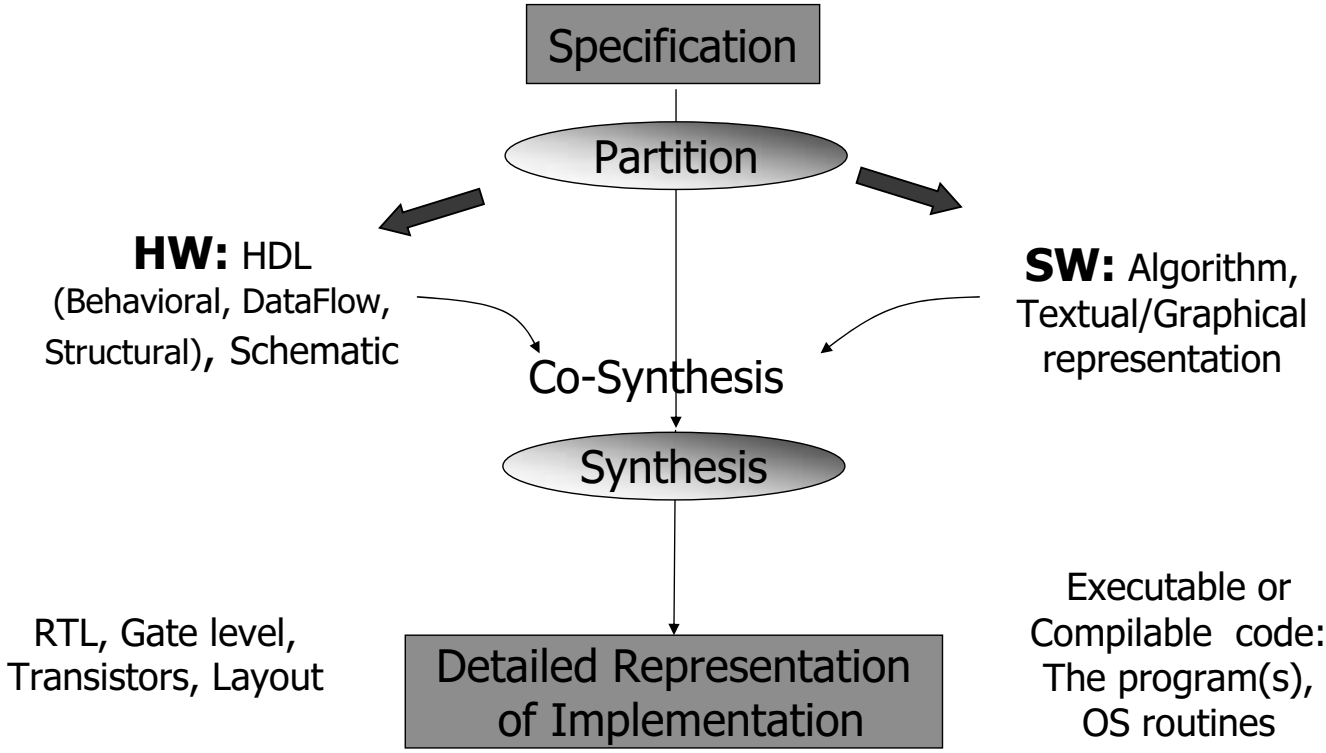
System Architecture Design

- **System Architecture & Exploration**
- **What**
  - Hardware/Software partitioning; processor, and memory architecture choices; system timing budget, power management strategy, system verification strategy…
  - Partitioning into HW block hierarchy, cycle time budgeting, block interfaces, block verification, clock architecture and test strategy
  - Fixed point architecture exploration and design
- **How - Quickly assemble architecture(s) for exploration to measure system timing/performance. Need to accurately (enough) model the bottlenecks**

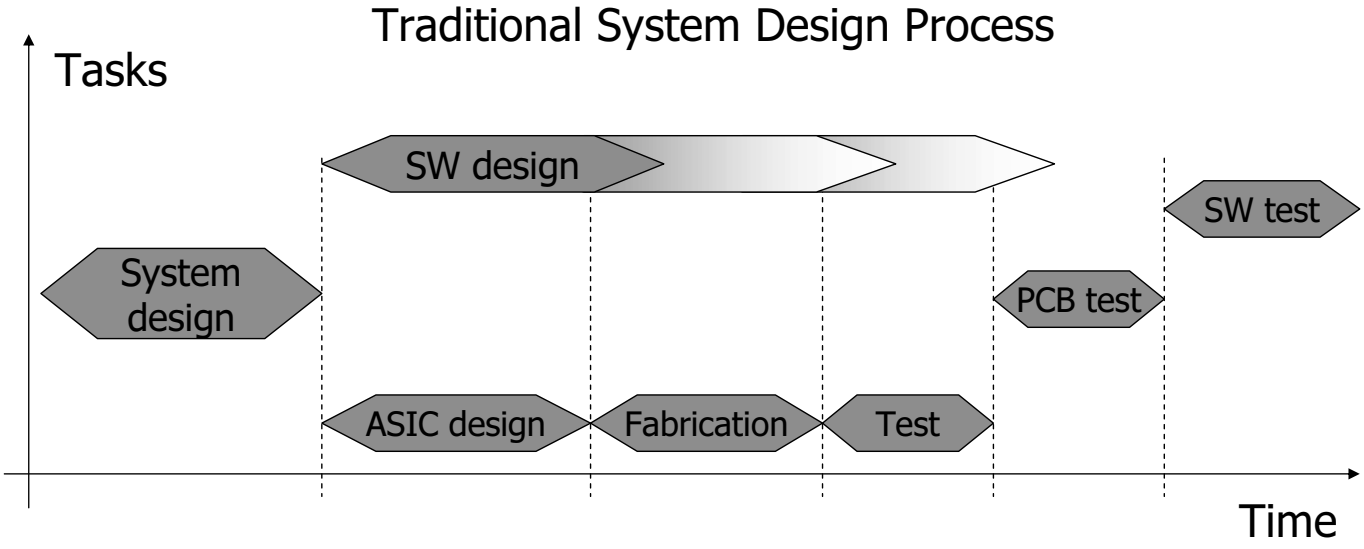
System Integration & Verification

- System Design Environment for HW/SW Refinement, Verification and Integration
- What
  - Enables hierarchical (manual or automatic) refinement of individual blocks of design in context of system. Maintain system and hierarchical test benches
  - Verification of refined hardware/software with entire system design
  - Define next level of clock architecture (derived) and test strategy
- How - Build a system verification hierarchy that allows integration of HW blocks, system software (HAL), embedded application SW and eventually verifying the entire design at cycle accurate (or RTL) level

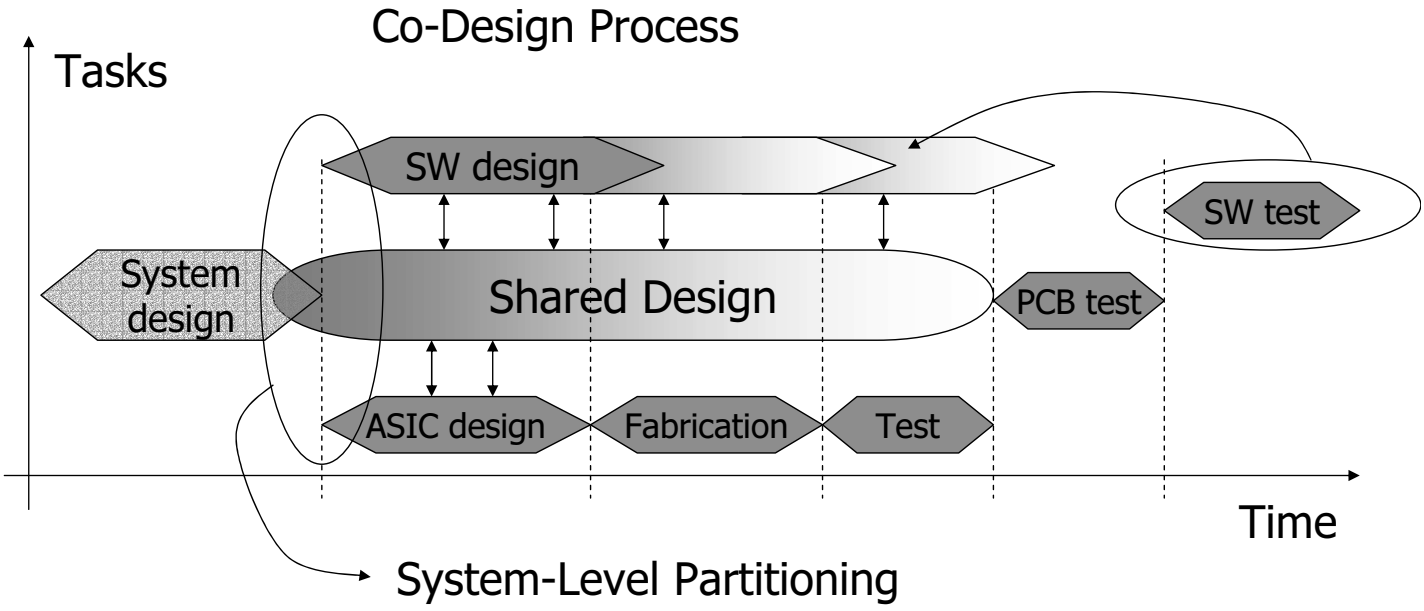
CoDesign and Co-Synthesis



Traditional design



System-level Co-design



Configurabilty and Embedded Systems

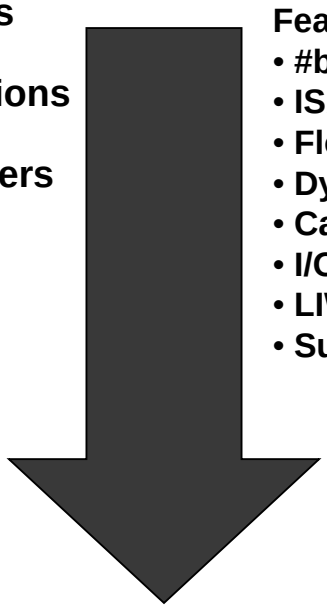
- **Advantages of configuration:**
- **Pay (in power, design time, area) only for what you use**
- **Gain additional performance by adding features tailored to your application:**
- **Particularly for embedded systems:**
  - Principally in embedded controller microprocessor applications
  - Some us in DSP

What to Configure?

- **What parts of the microcontroller/microprocessor system to configure?**
- **Easy answers:**
  - Memory and Cache Sizes - get precisely the sizes your applications needs
  - Register file sizes
  - Interrupt handling and addresses
- **Harder answers:**
  - Peripherals
  - Instructions
- ***But first we need more context***

Trickle Down Theory of Embedded Architectures

- **Mainframe/supercomputers**
- **High-end servers/workstations**
- **High-end personal computers**
- **Personal computers**
- **Lap tops/palm tops**
- **Gadgets**
- **Watches**
- **...**

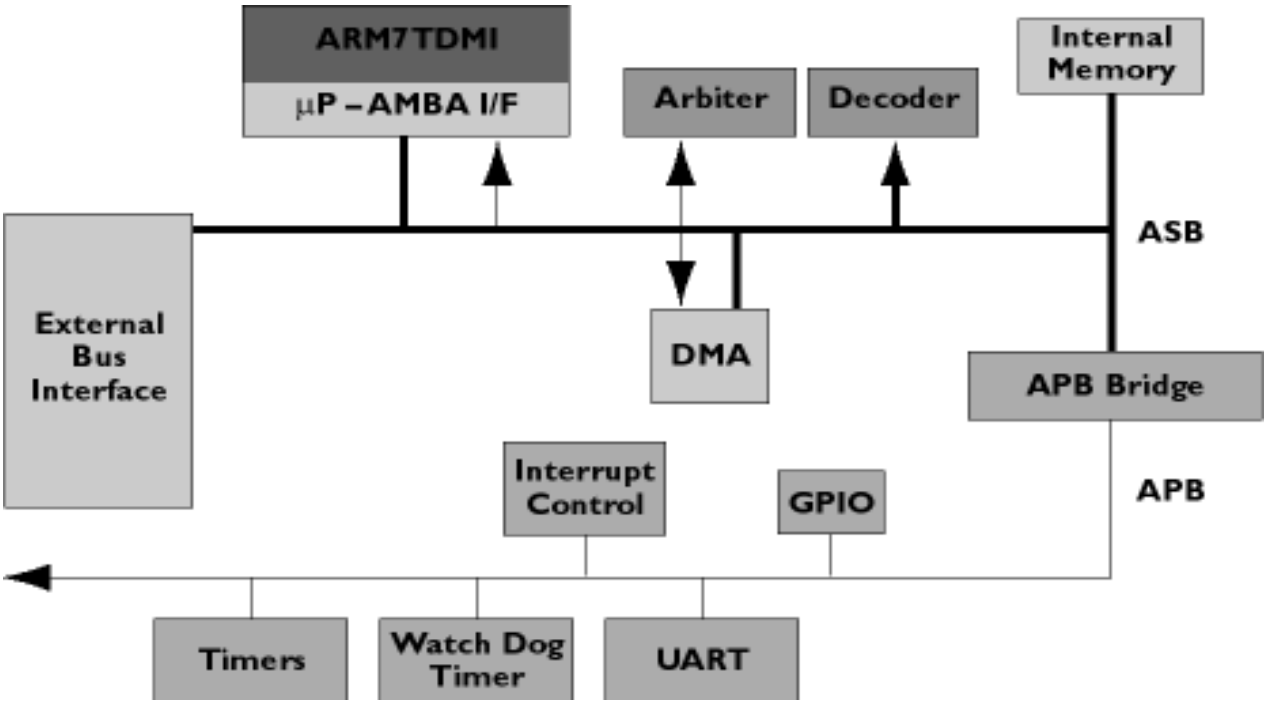


Features tend to trickle down:
- **#bits: 4->8->16->32->64**
- **ISA's**
- **Floating point support**
- **Dynamic scheduling**
- **Caches**
- **I/O controllers/processors**
- **LIW/VLIW**
- **Superscalar**

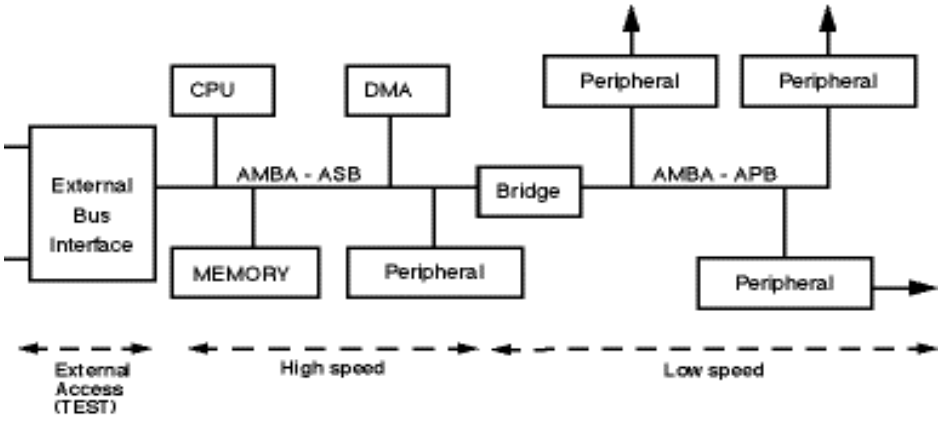
Configurability in ARM Processor

- ARM allows for configurability via AMBA bus
- Offers ``prime cell'' peripherals which hook into AMBA Peripheral Bus (APB)
- UART
- Real Time Clock
- Audio Codec Interface
- Keyboard and mouse interface
- General purpose I/O
- Smart card interface
- Generic IR interface

ARM7 core



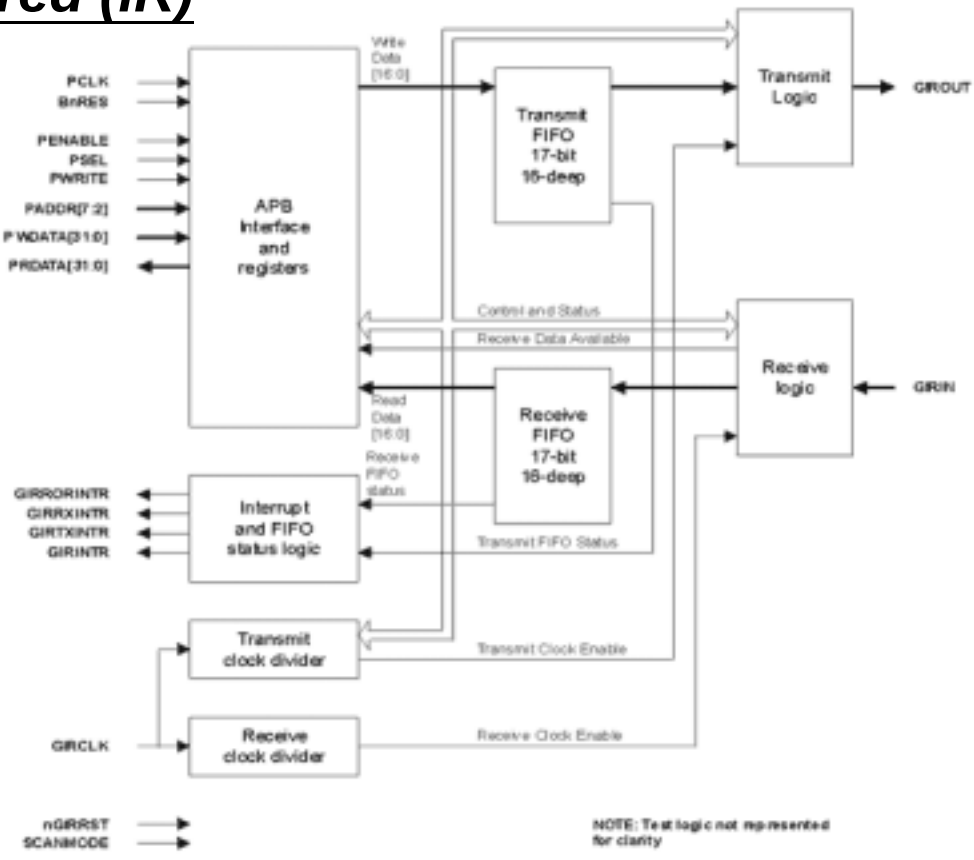
ARM's Amba open standard



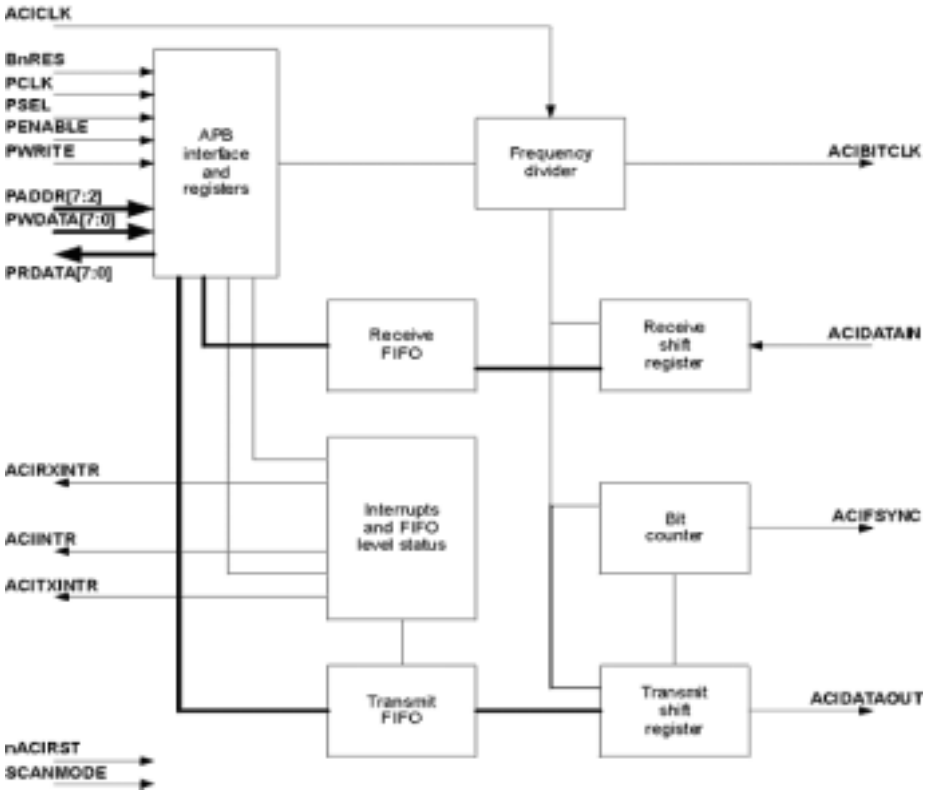
- **Advanced System Bus, (ASB) - high performance, CPU, DMA, external**
- **Advanced Peripheral Bus, (APB) - low speed, low power, parallel I/O, UART's**
- **External interface**

http://www.arm.com/Documentation/Overviews/AMBA_Intro/#intro

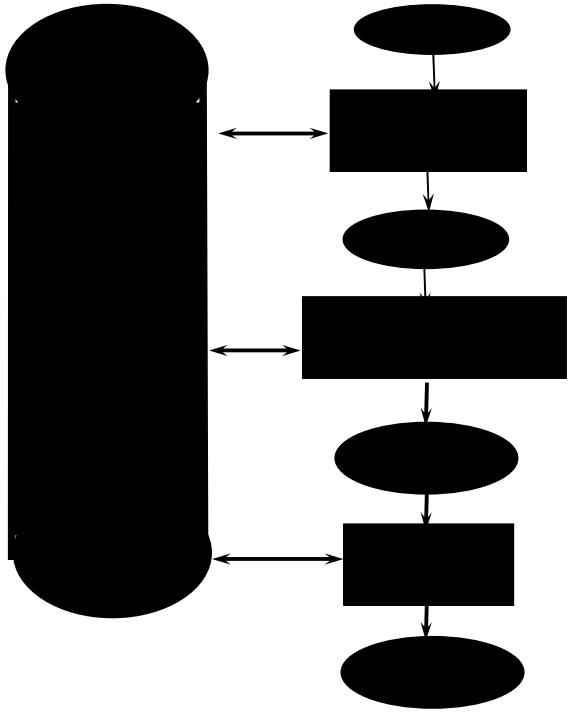
Ex: ARM Infrared (IR) Interface



Ex : Audio Codec



Another Kind of Configurability



Synthesis of a processor core from an RTL description allows for:

- **full range of other types of configurability**
- **additional degrees of freedom in quality of implementation**

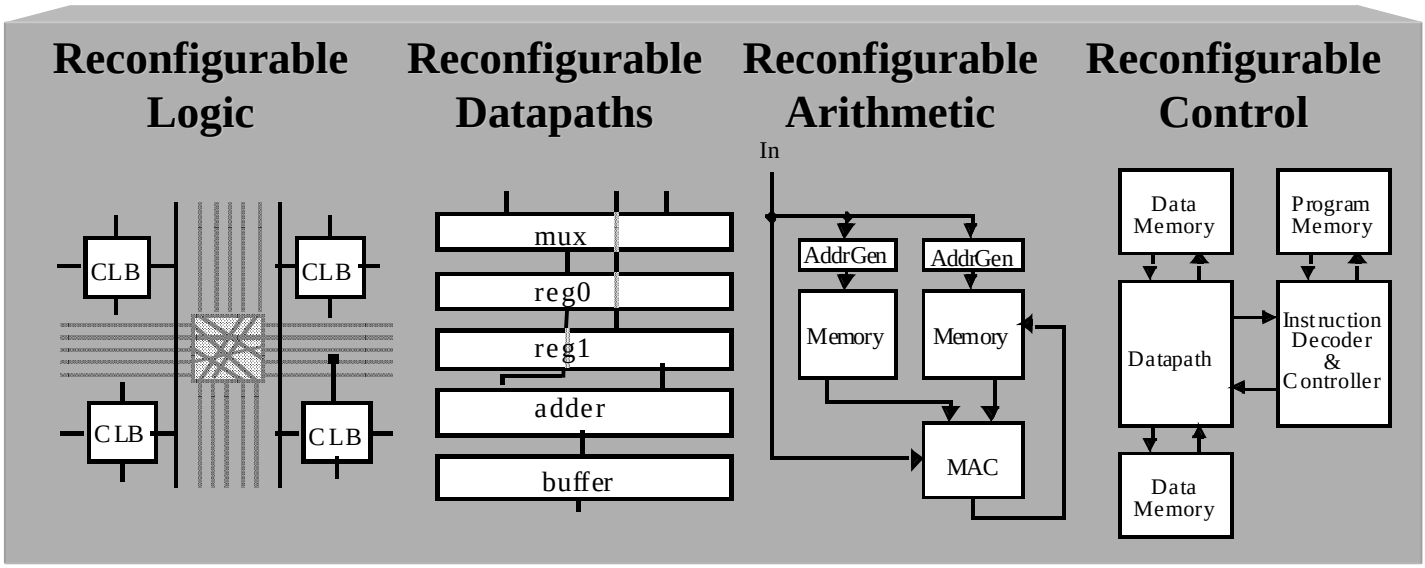
Examples:

- **ARM7**
- **Motorola Coldfire**
- **Tensilica Xtensa**

Issues in low-level Configurable Design

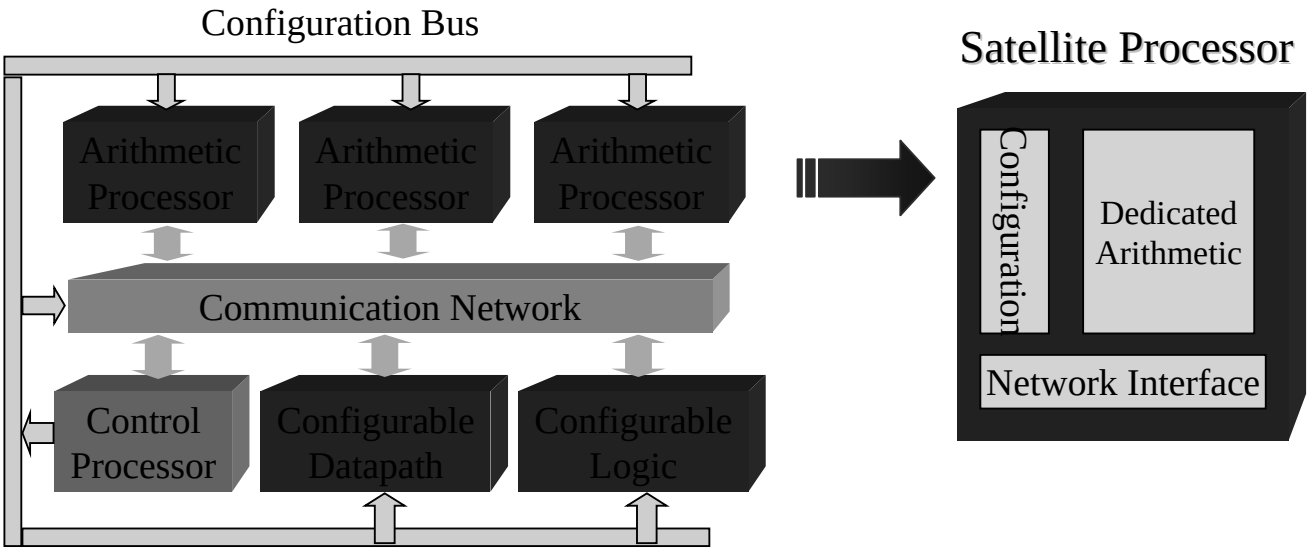
- **Choice and Granularity of Computational Elements**
- **Choice and Granularity of Interconnect Network**
- **(Re)configuration Time and Rate**
  - Fabrication time --> Fixed function devices
  - Beginning of product use --> Actel/Quicklogic FPGAs
  - Beginning of usage epoch --> (Re)configurable FPGAs
  - Every cycle --> traditional Instruction Set Processors

The Choice of the Computational Elements



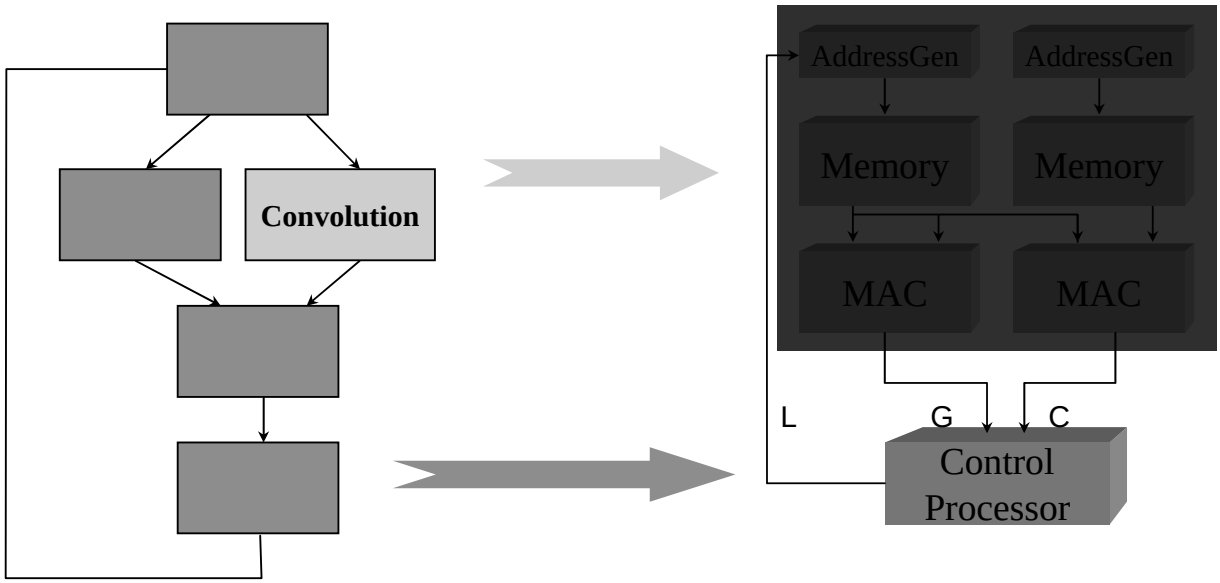
Bit-Level Operations	Dedicated data paths	Arithmetic kernels	RTOS
e.g. encoding	e.g. Filters, AGU	e.g. Convolution	Process management

Multi-granularity Reconfigurable Architecture: The Berkeley Pleiades Architecture



- Computational kernels are “spawned” to satellite processors
- Control processor supports RTOS and reconfiguration
- Order(s) of magnitude energy-reduction over traditional programmable architectures

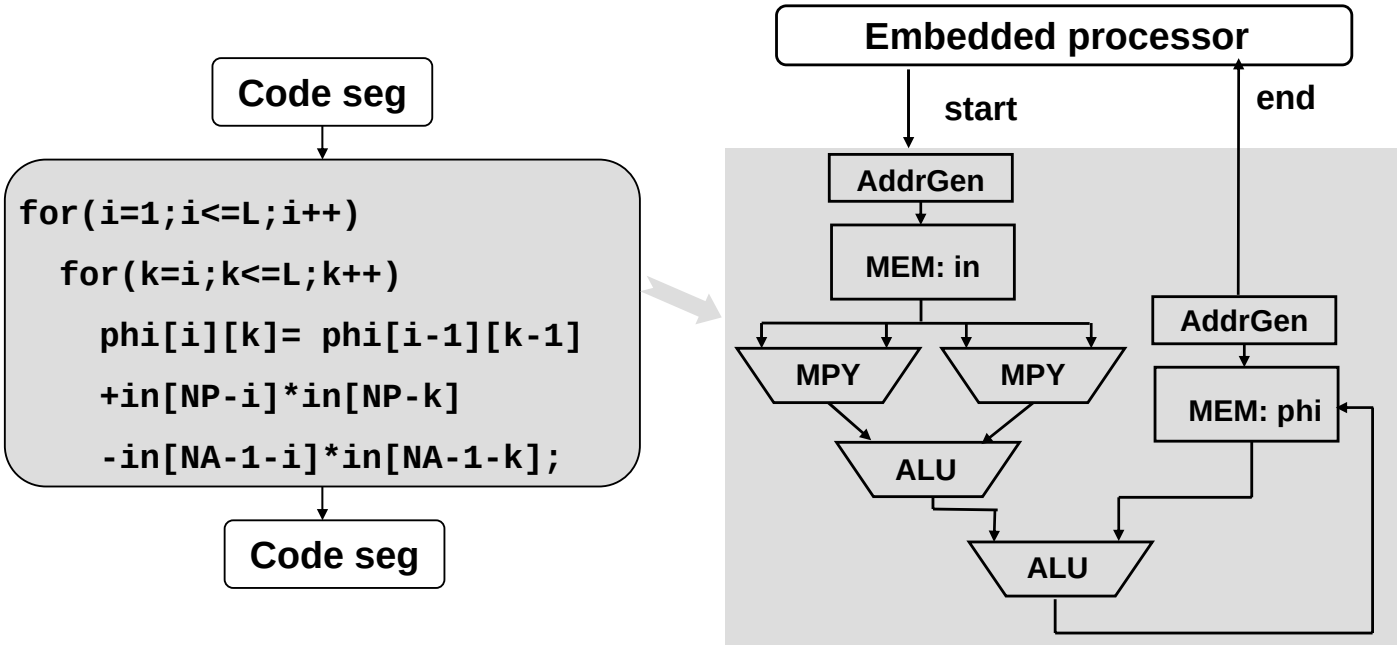
Matching Computation and Architecture



Two models of computation:
communicating processes + data-flow

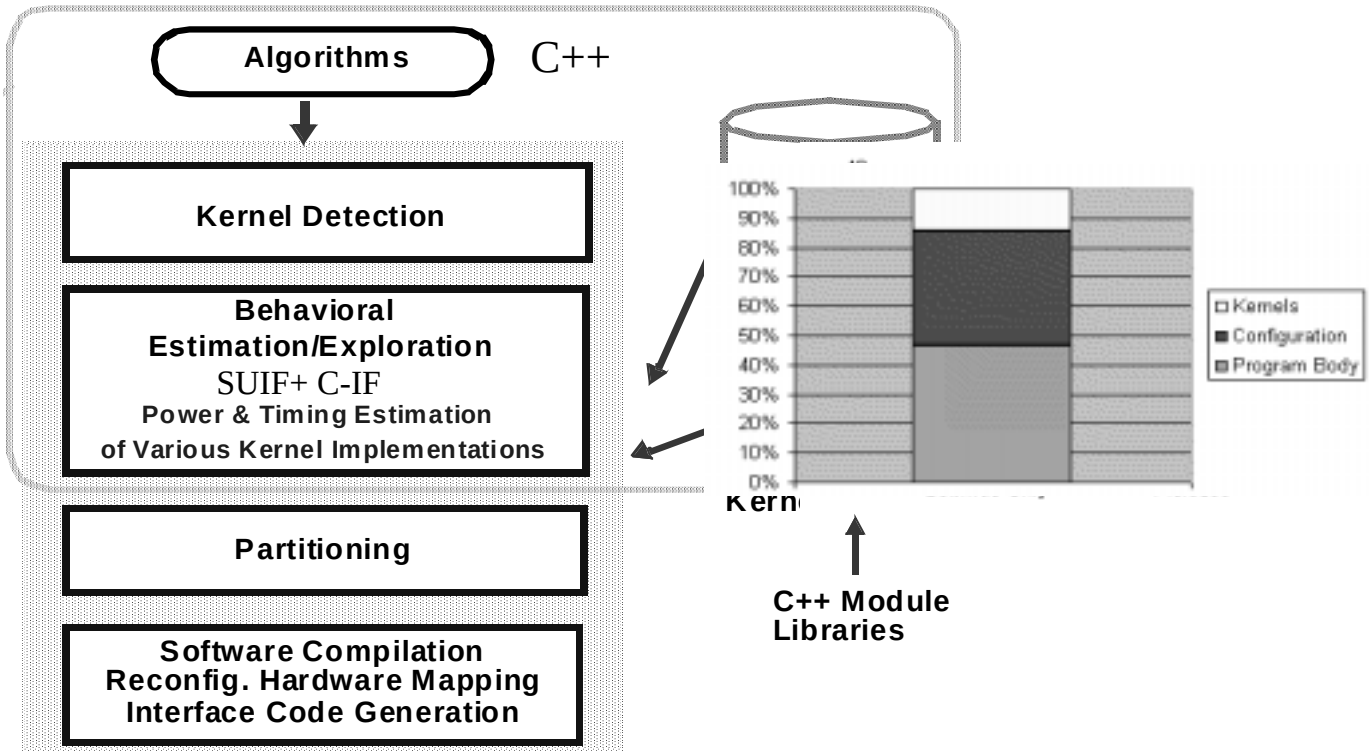
Two architectural models:
sequential control+ data-driven

Execution Model of a Data-Flow Kernel

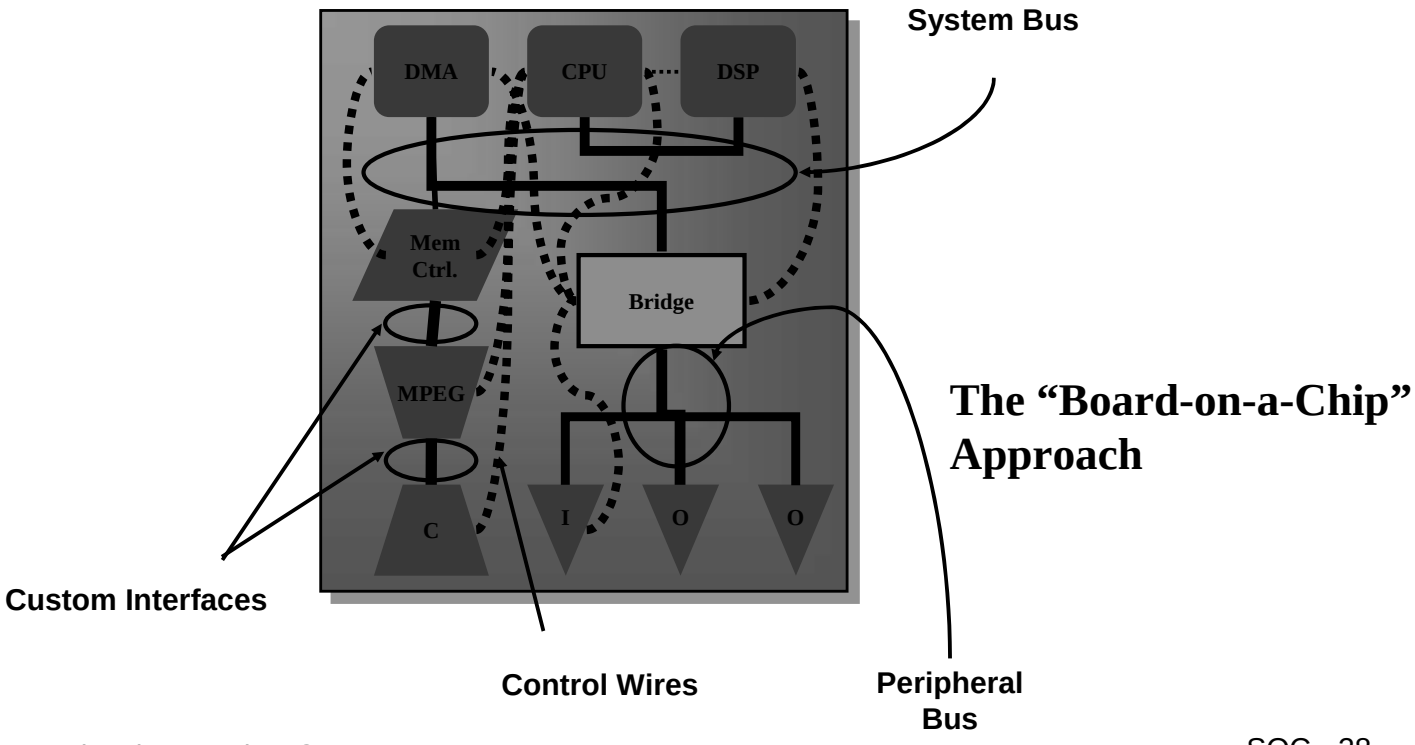


- **Distributed control and memory**

Software Methodology Flow

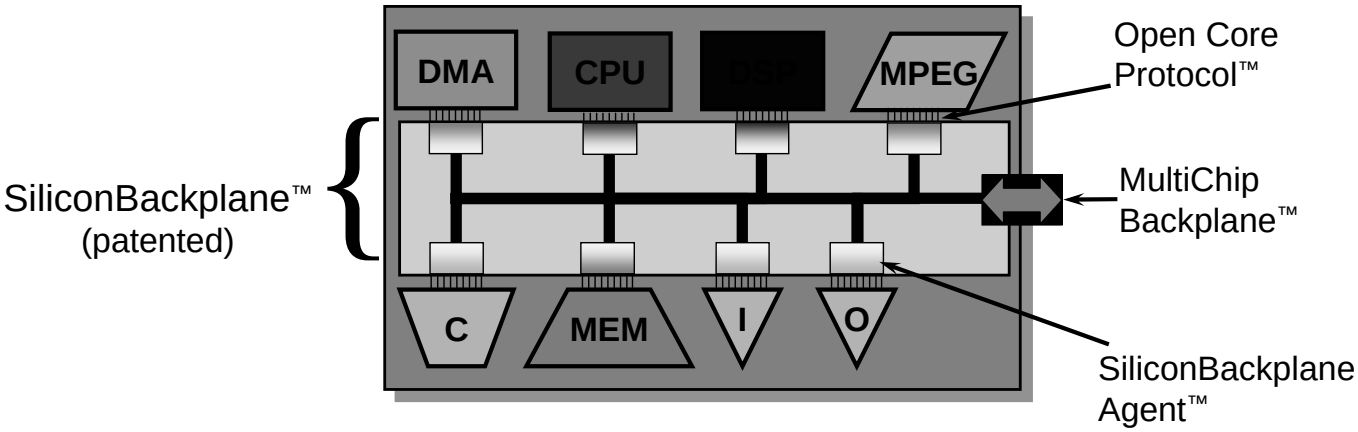


The System-on-a-Chip Nightmare

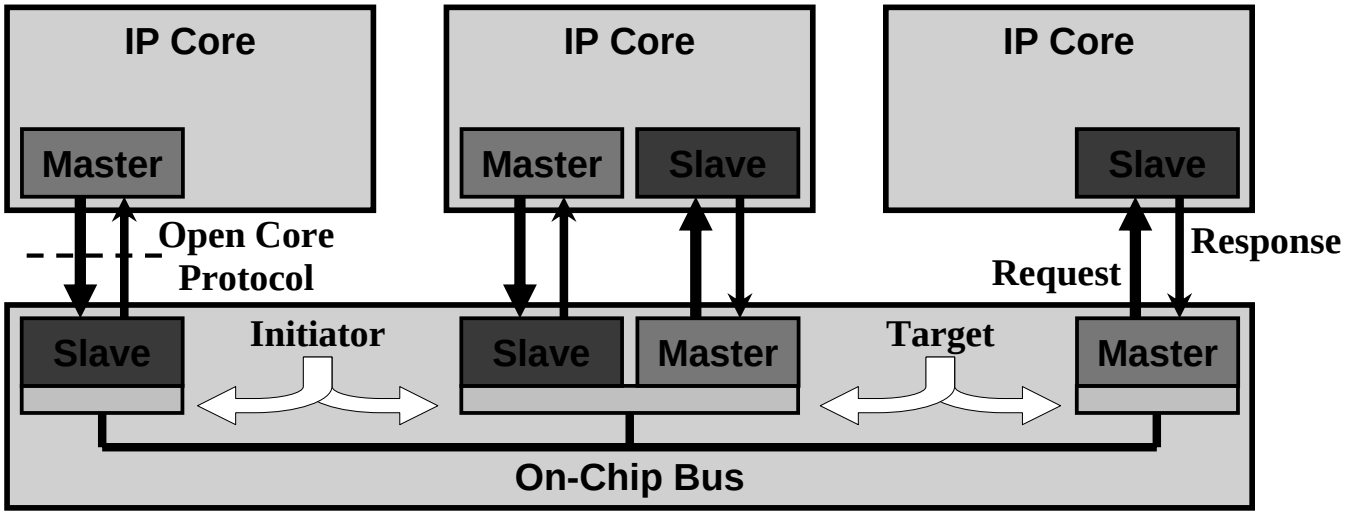


The "Board-on-a-Chip" Approach

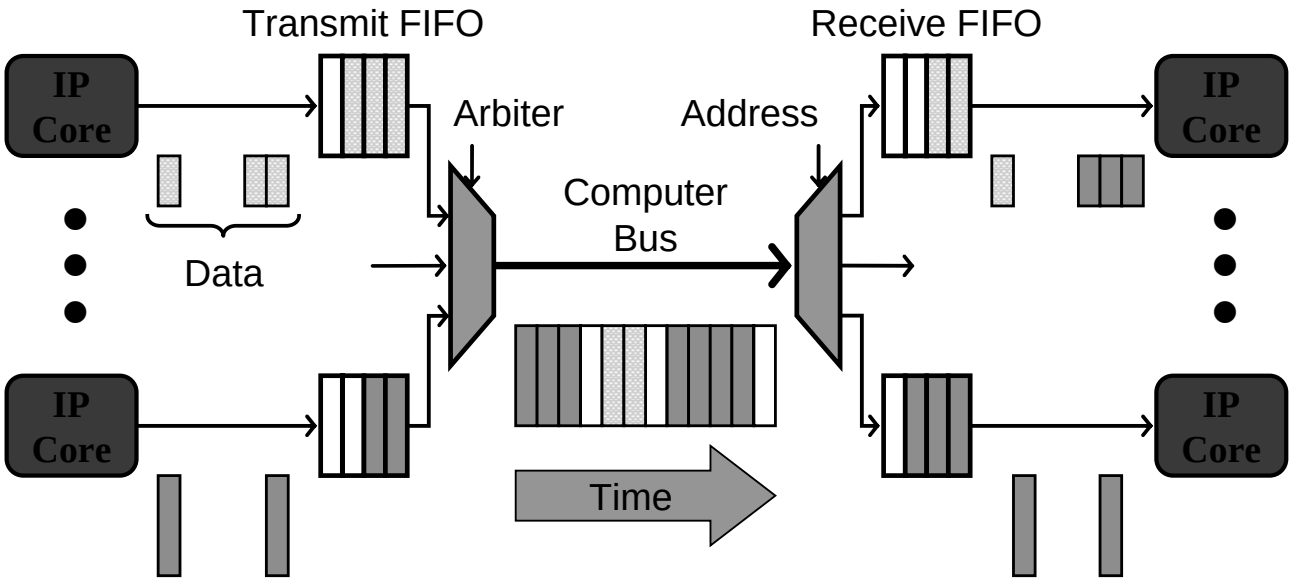
Sonics SOC Integration Architecture



Master vs. Slave

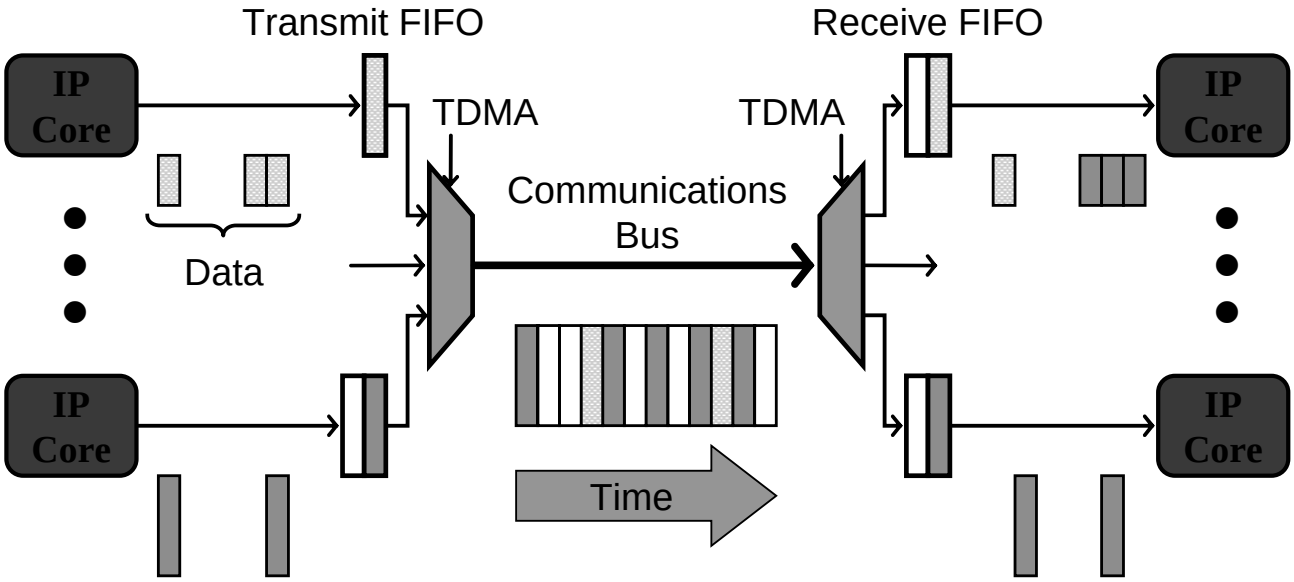


The Backplane: Why Not Use a Computer Bus?



- **Expensive to decouple**
- **Not designed for real-time**

Communication Buses Decouple and Guarantee Real Time



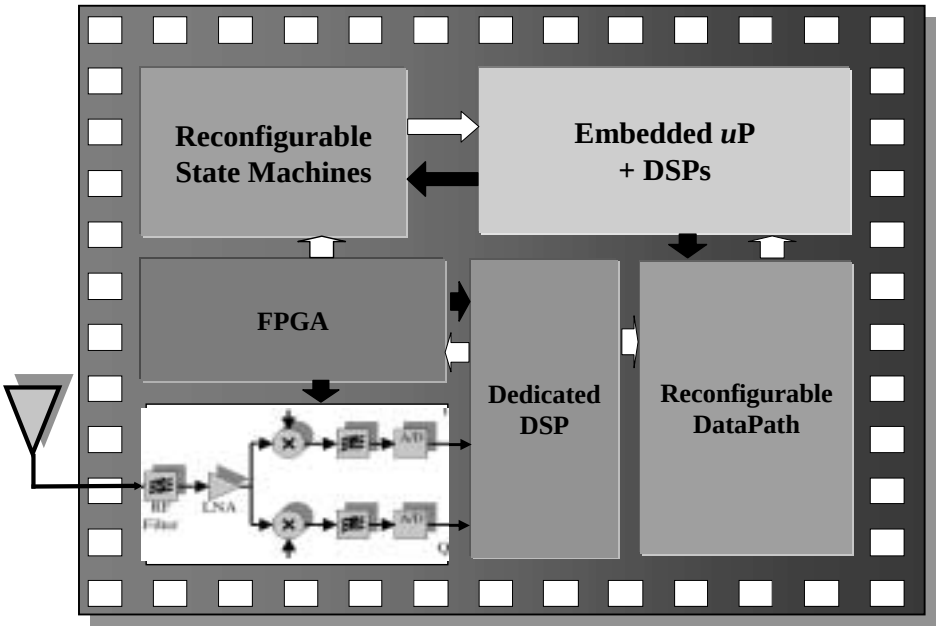
- **Connections are expensive**
- **Poor read latency**

On-Chip Bus for SOC

- **Example on- chip bus interconnects**
  - ARM's AMBA bus
  - IBM's Core Connect
  - Virtual Socket Interface Alliance group
  - Open Connect Protocol group
- **Example processor cores**
  - ARM
  - MIPS
  - PowerPC

Design Example:

The Radio-on-a-Chip



- ❑DSP and control intensive
- ❑Mixed-mode
- ❑Combines programmable, flexible, and application-specific modules
- ❑Cost and energy are the key metrics

System Level Design Science

❑Design Methodology:

- Top Down Aspect:
  - Orthogonalization of Concerns:
    - ***Separate Implementation from Conceptual Aspects***
    - ***Separate computation from communication***
  - Formalization: precise unambiguous semantics
  - Abstraction: capture the desired system details (do not overspecify)
  - Decomposition: partitioning the system behavior into simpler behaviors
  - Successive Refinements: refine the abstraction level down to the implementation by filling in details and passing constraints
- Bottom Up Aspect:
  - IP Re-use (even at the algorithmic and functional level)
  - Components of architecture from pre-existing library

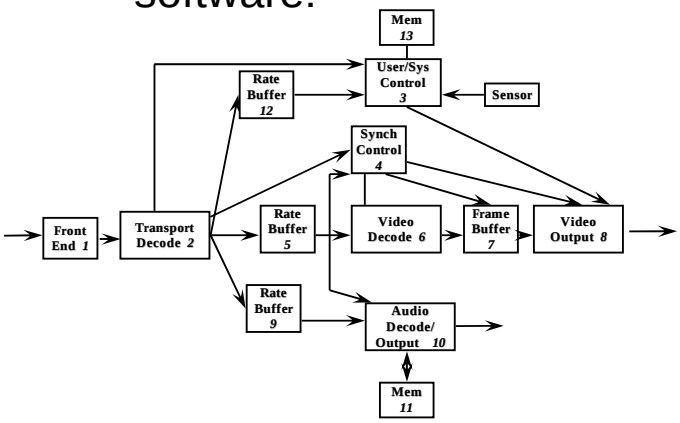
Separate Behavior from Micro-architecture

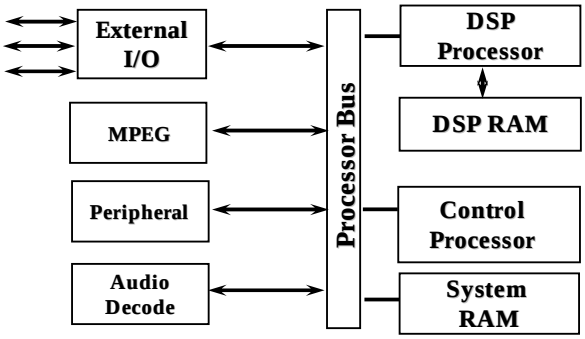
❑System Behavior

- Functional Specification of System.
- No notion of hardware or software!

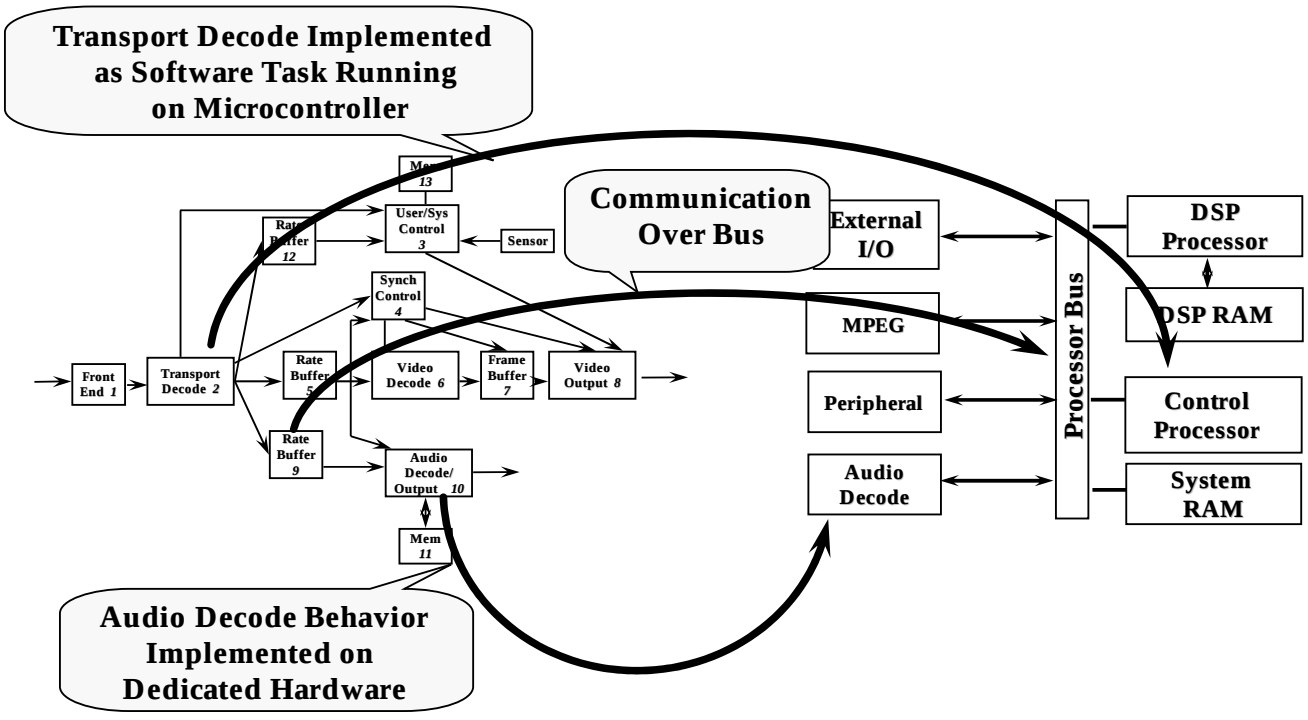
❑Implementation Architecture

- Hardware and Software
- Optimized Computer



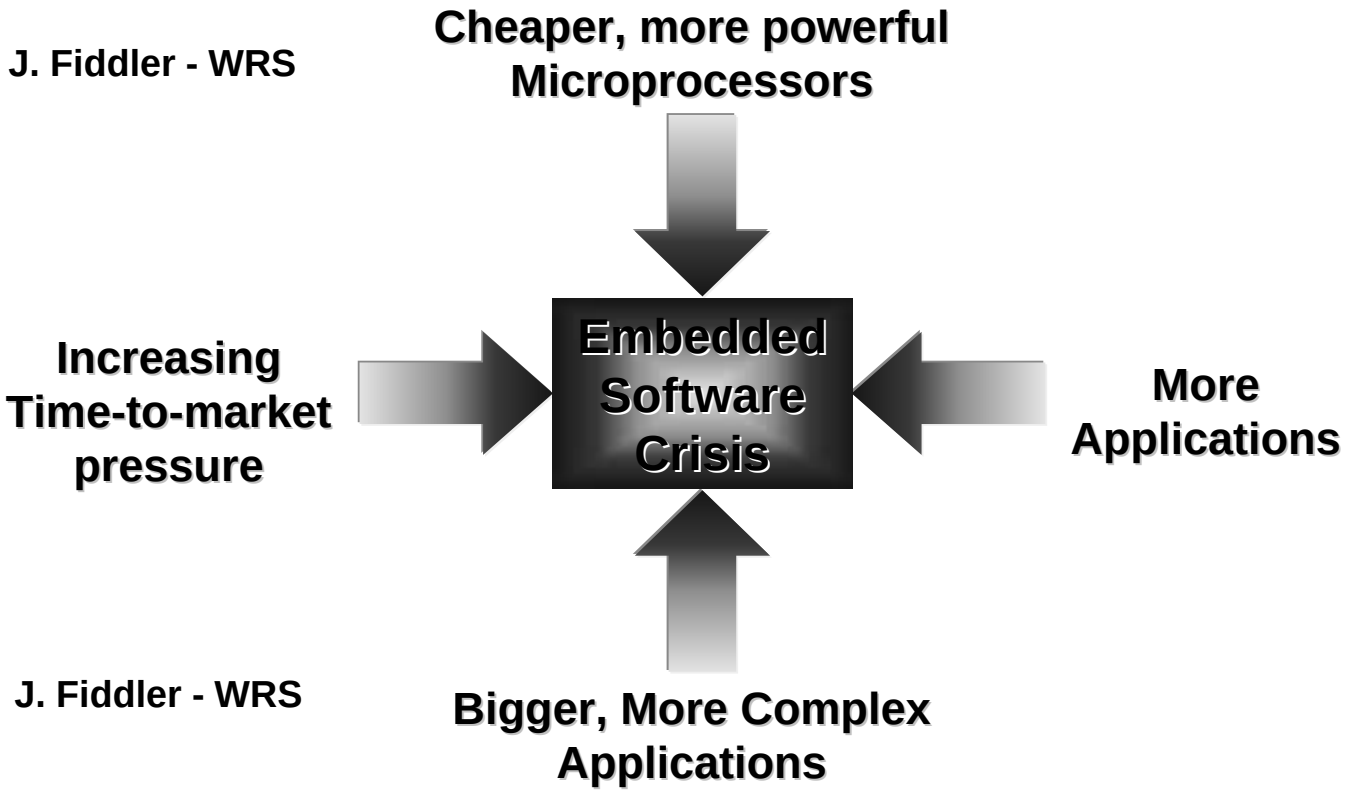


Map Between Behavior from Architecture



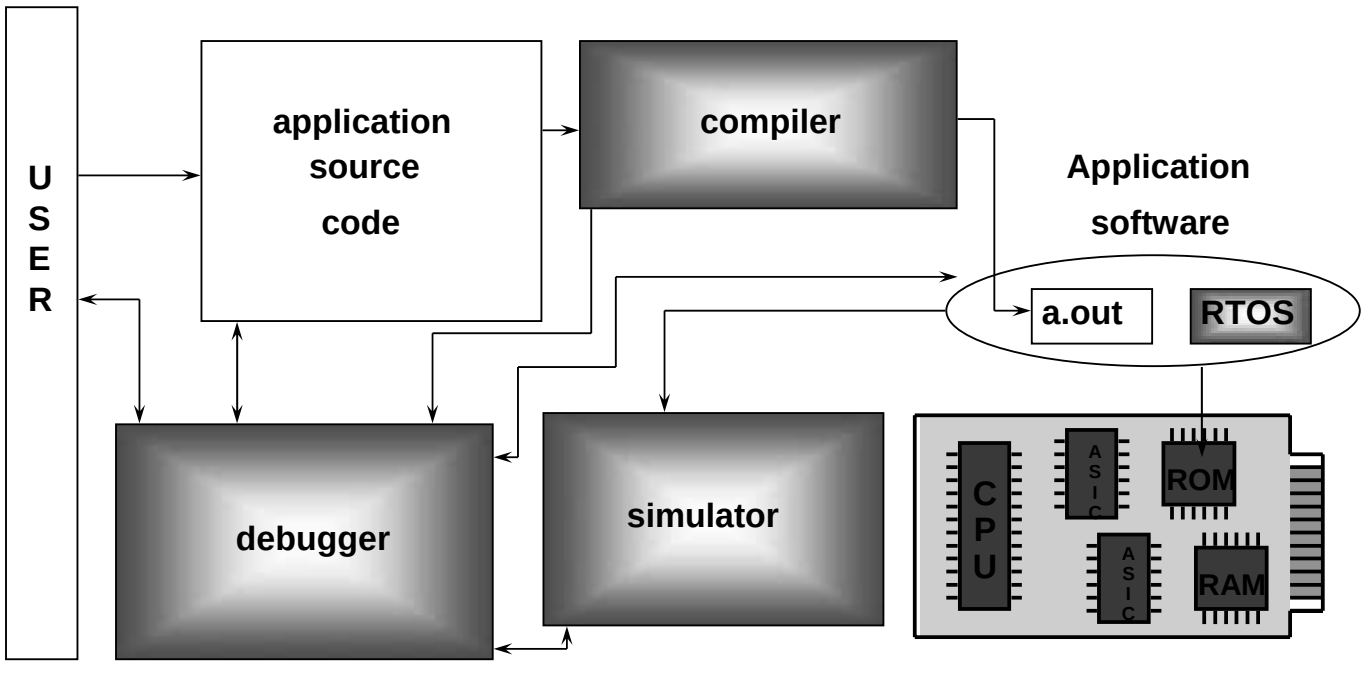
Embedded Software Crisis

J. Fiddler - WRS



J. Fiddler - WRS

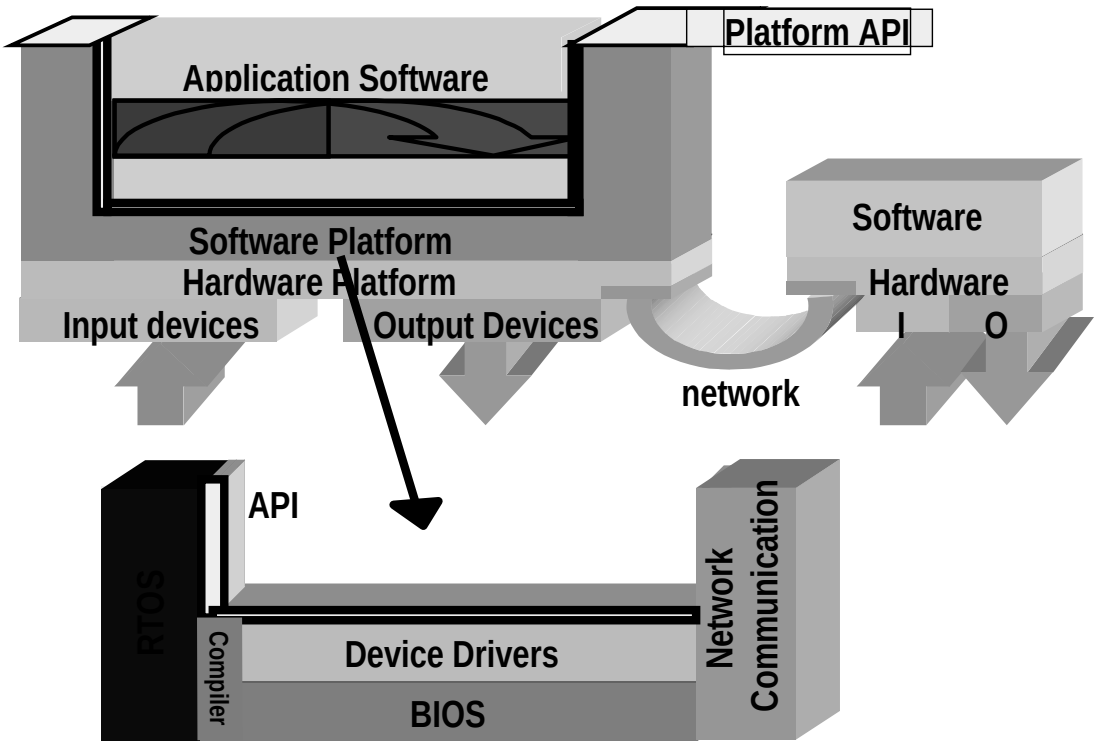
SW: Embedded Software Tools



Hardware Platforms Not Enough!

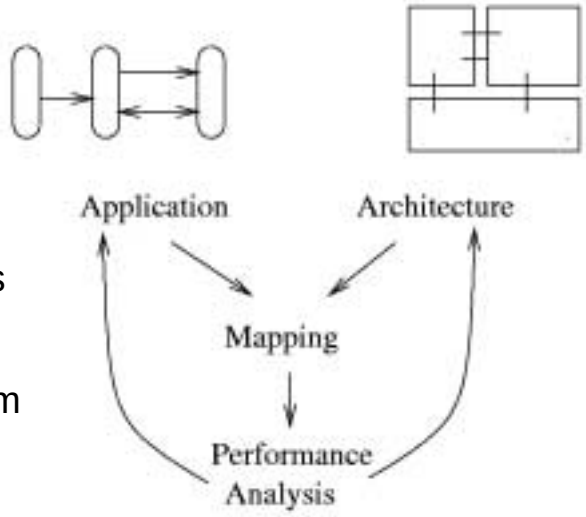
- **Hardware platform has to be abstracted**
- **Interface to the application software is the "API"**
- **Software layer performs abstraction:**
  - Programmable cores and memory subsystem "hidden" by RTOS and compilers
  - I/O subsystem with Device Drivers
  - Network with Network Communication Software

Software Platforms



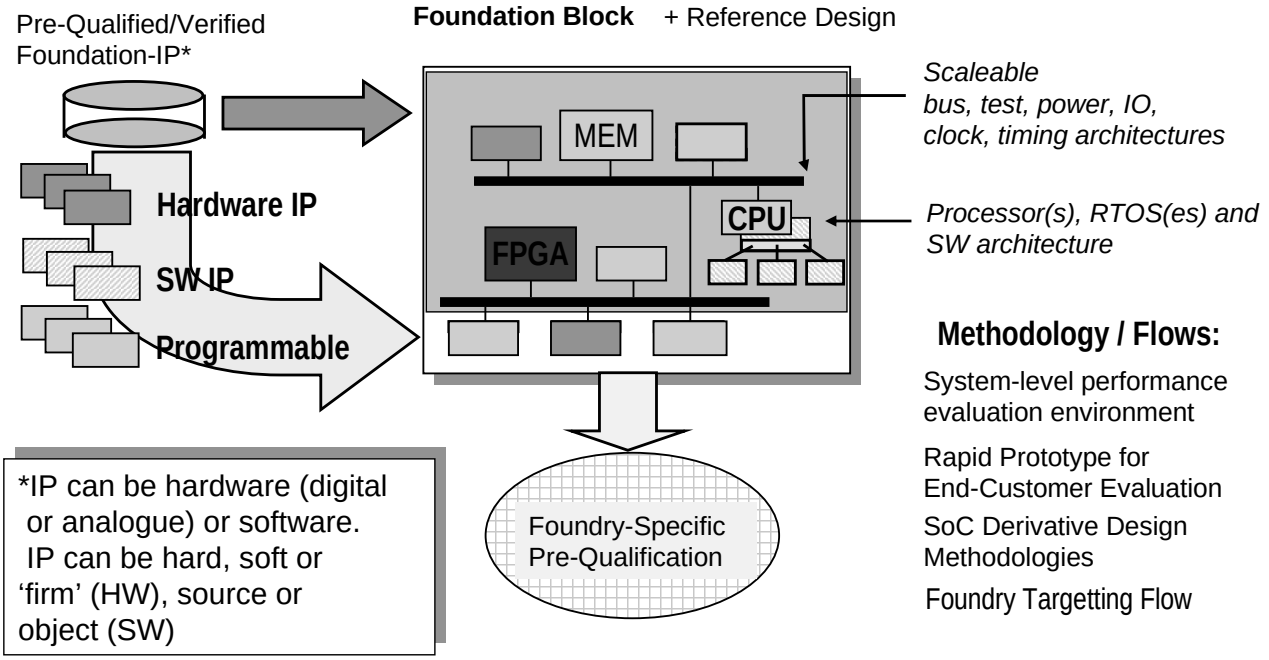
Platform-based methodology

- **Platform based design:**
  - Application mapped on architecture
  - Performance evaluation and iterative refinement
- **Challenges:**
  - complete system simulation
  - complexity management
  - composability and reuse
- **Key elements for composability**
  - Identification and use of useful models of computation
    - FSMD, DE, DF, CSP, ...
  - A flexible, extensible language platform to capture the functionality.
- **Composability can be achieved using Object-oriented mechanisms:**



Final goal for SOC: Platform-Based Design

Taking Design Block Reuse to the Next Level



Summary of SOC Design