System Design and Methodology / Embedded Systems Design

VIII. Architectures and Platforms

TDTS07/TDDI08
VT 2026

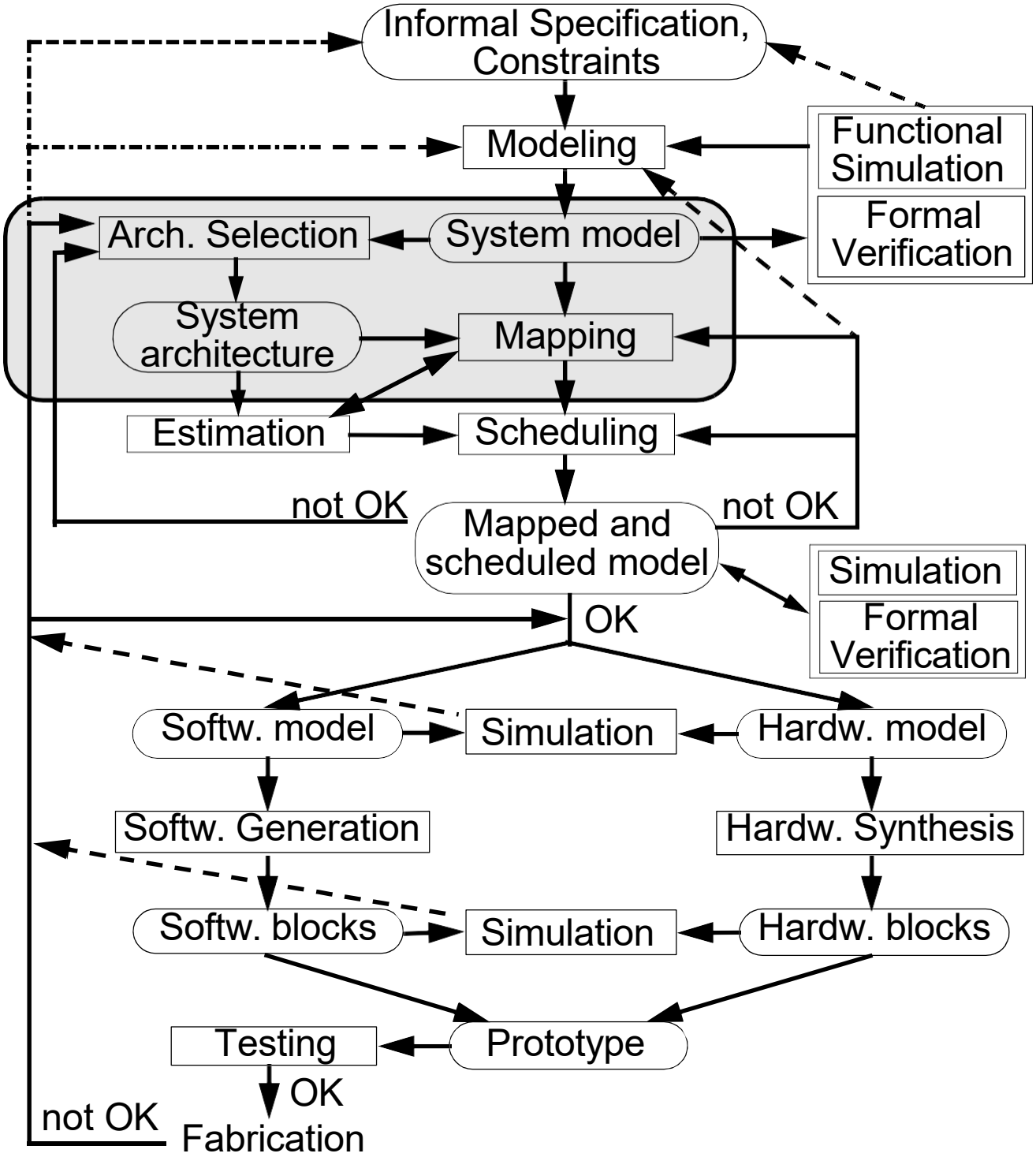
Ahmed Rezine

(Based on material by Petru Eles and Soheil Samii)

Institutionen för datavetenskap (IDA)
Linköpings universitet

ARCHITECTURES AND PLATFORMS

1. Architecture Selection: The Basic Trade-offs
2. General Purpose vs. Application-Specific Processors
3. Processor Specialisation
4. ASIP Design Flow
5. Tool Support for Processor Specialisation
6. Application Specific Platforms
7. IP-Based Design (Design Reuse)
8. Reconfigurable Systems


Informal Specification, Constraints
Modeling
Functional Simulation
Formal Verification
Arch. Selection
System model
System architecture
Mapping
Estimation
Scheduling
not OK
not OK
Mapped and scheduled model
Simulation
Formal Verification
OK
Softw. model
Simulation
Hardw. model
Softw. Generation
Hardw. Synthesis
Softw. blocks
Simulation
Hardw. blocks
Testing
Prototype
OK
not OK
Fabrication

Architecture Selection and Mapping

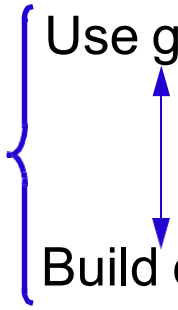
- Select underlying hardware structure on which to run the modelled system.

- Map the functionality captured by the system over the components of the selected architecture.

  Functionality includes processing and communication.

Architecture Selection

General Purpose vs. Application Specific



Use general purpose, existing platform and map the application on it.

or something in-between

Build customised architecture strictly optimized for the application.

Architecture Selection

General Purpose vs. Application Specific

- Use general purpose, existing platform and map the application on it.
- or something in-between
- Build customised architecture strictly optimized for the application.

Software vs. Hardware

- Use programmable processors running software.
- or both
- Use dedicated electronics
  - fixed
  - reconfigurable

Architecture Selection

General Purpose vs. Application Specific

- Use general purpose, existing platform and map the application on it.
- or something in-between
- Build customised architecture strictly optimized for the application.

Software vs. Hardware

- Use programmable processors running software.
- or both
- Use dedicated electronics
  - fixed
  - reconfigurable

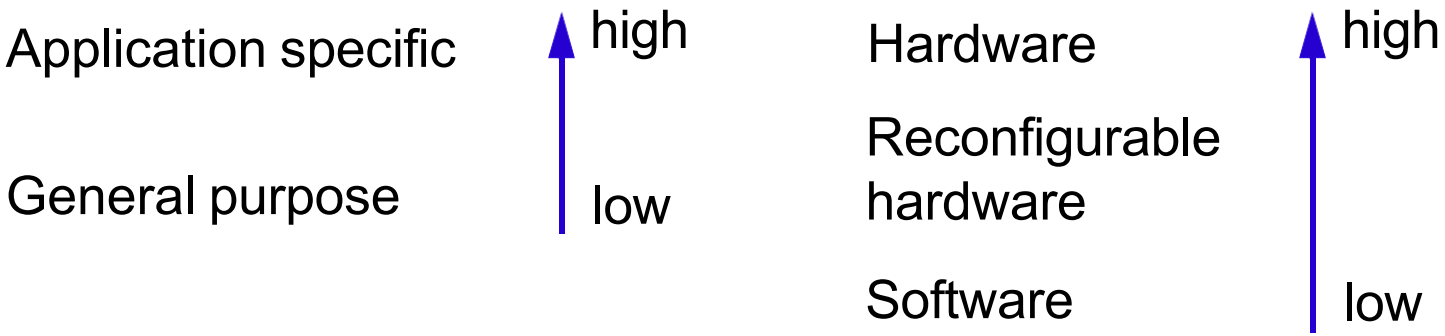
Mono vs. Multipr. Single vs. Multichip

- Monoprocessor
- Multiprocessor
  - single chip
  - multi chip

Architecture Selection

The trade-offs:

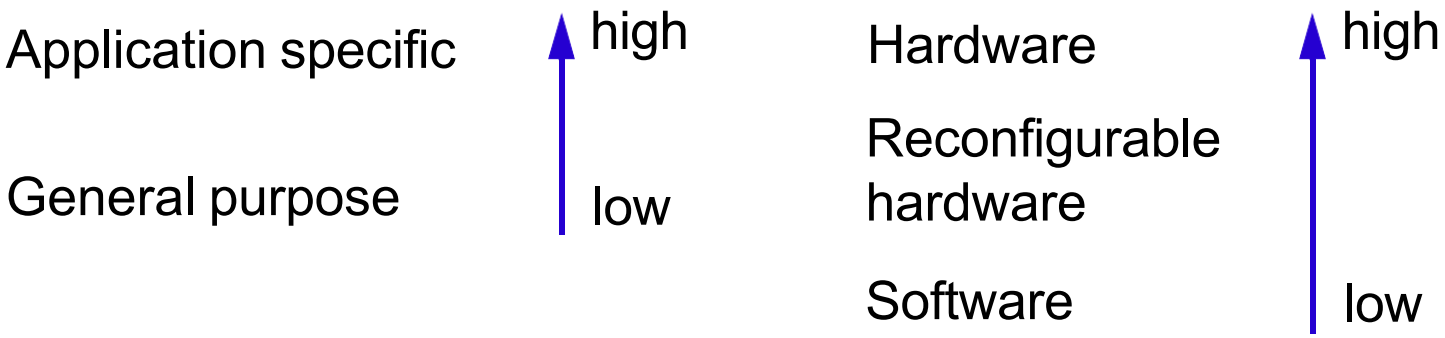
- Performance (high speed, low power consumption)



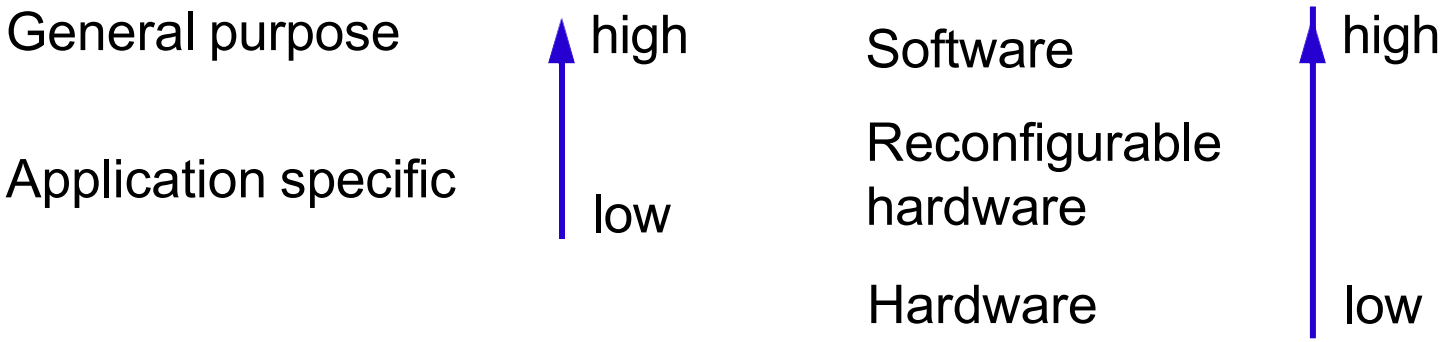
Architecture Selection

The trade-offs:

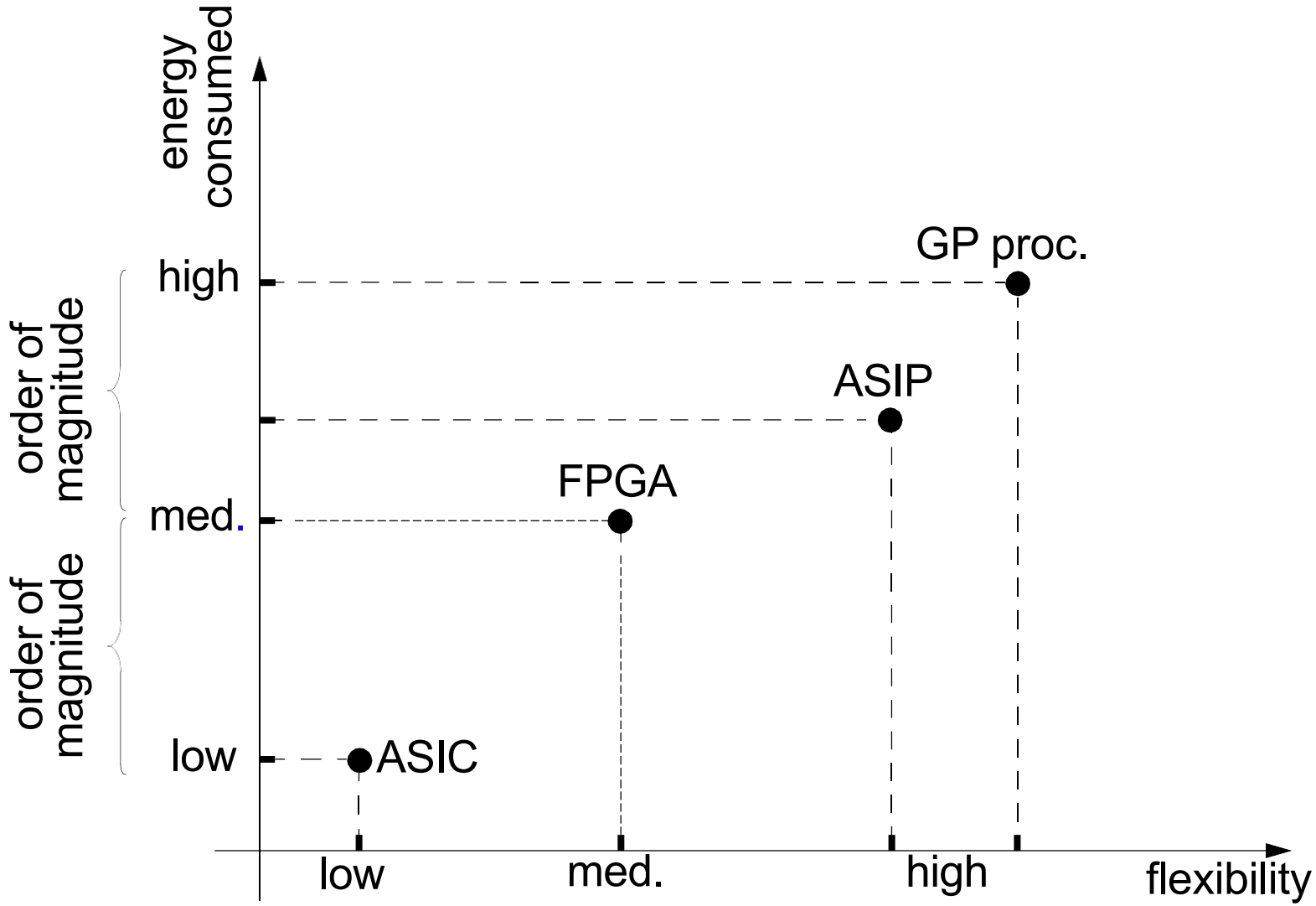
- Performance (high speed, low power consumption)



- Flexibility (how easy it is to upgrade or modify)



Architecture Selection



General Purpose vs. Application Specific Processors

- Both GP processors and ASIPs (application specific instruction set processors) can be RISCs, CISCs, DSPs, microcontrollers, etc.
  - GP processors
    - Neither instruction set nor microarchitecture or memory system are customised for a particular application or family of applications
  - ASIPs
    - Instruction set, microarchitecture and/or memory system are customised for an application or family of applications.

Results in better performance and reduced power consumption.

What Makes an ASIP “Specific”?

- Instruction set (IS) specialisation
  - Exclude instructions which are not used
    - reduces instruction word length (fewer bits needed for encoding);
    - keeps controller and data path simple.
  - Introduce instructions, even “exotic” ones, which are specific to the application: combinations of arithmetic instructions (multiply- accumulate), small algorithms (encoding/decoding, filter), vector operations, string manipulation or string matching, pixel operations, etc.
    - reduces code size $\Rightarrow$ reduced memory size, memory bandwidth, power consumption, execution time.
    - increases speed.

What Makes an ASIP “Specific”?

- Function unit and data path specialisation

  - Once an application specific IS is defined, this IS can be implemented using a specific data path and specific function units.

    - Adaptation of word length.
    - Adaptation of register number.
    - Adaptation of functional units

      Highly specialised functional units can for string matching and manipulation, pixel operation, arithmetics, and even complex units to perform certain sequences of computations (co-processors).

What Makes an ASIP “Specific”?

- Memory specialisation
  - Number and size of memory banks.
  - Number and size of access ports.
    - They both influence the degree of parallelism in memory access.
    - Having several smaller memory blocks (not one big) increases parallelism and speed, and reduces power consumption.
    - Sophisticated memory structures can increase cost and bandwidth requirement.
  - Cache configuration:
    - separate instruction/data?
    - associativity
    - cache size
    - line size

What Makes an ASIP “Specific”?

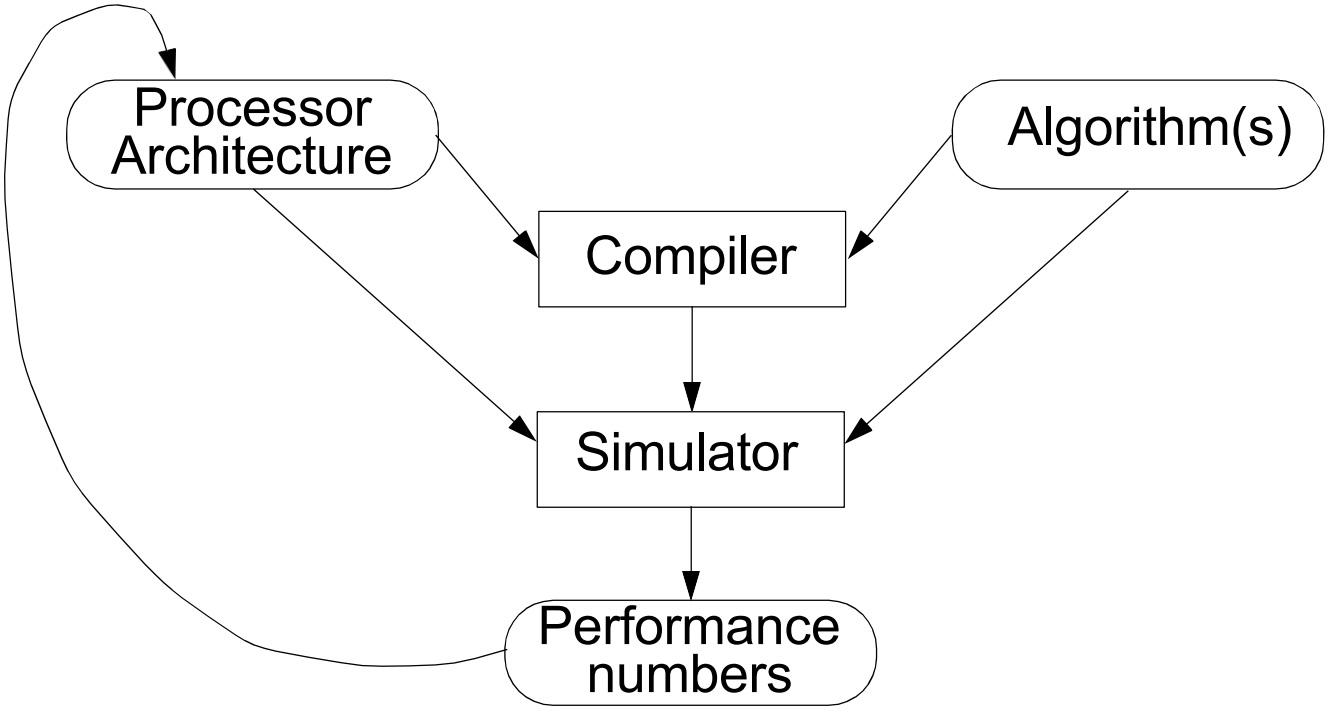
- Interconnect specialization
  - Interconnect of functional modules and registers.
  - Interconnect to memory and cache.
    - How many internal buses?
    - What kind of protocol?
    - Additional connections increase the potential of parallelism.

What Makes an ASIP “Specific”?

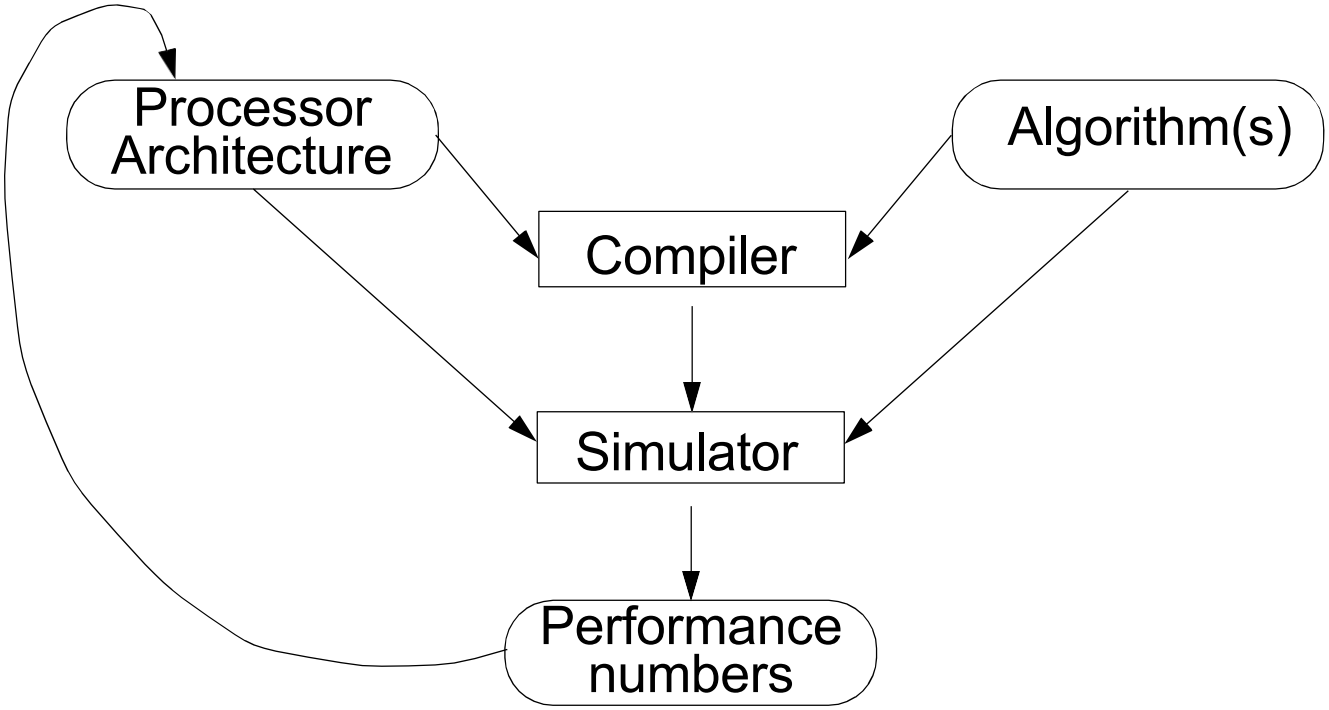
- Interconnect specialization
  - Interconnect of functional modules and registers.
  - Interconnect to memory and cache.
    - How many internal buses?
    - What kind of protocol?
    - Additional connections increase the potential of parallelism.

- Control specialisation
  - Centralised control or distributed (globally asynchronous)?
  - Pipelining?
  - Out of order execution?
  - Hardwired or microprogrammed?

ASIP Design Flow

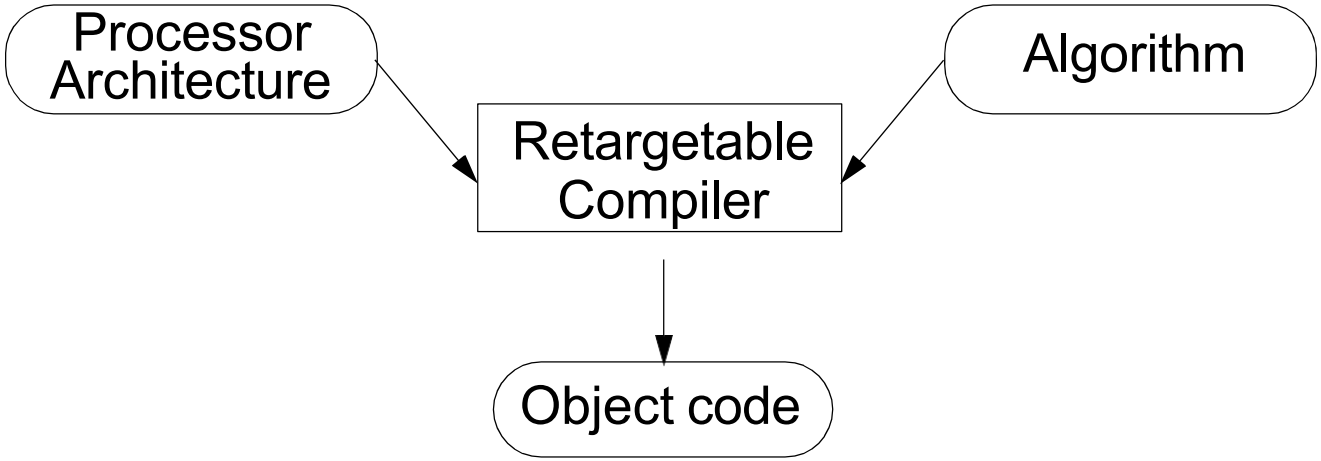


ASIP Design Flow



- In order to be able to generate a specialised architecture you need:
  - Retargetable compiler
  - Configurable simulator

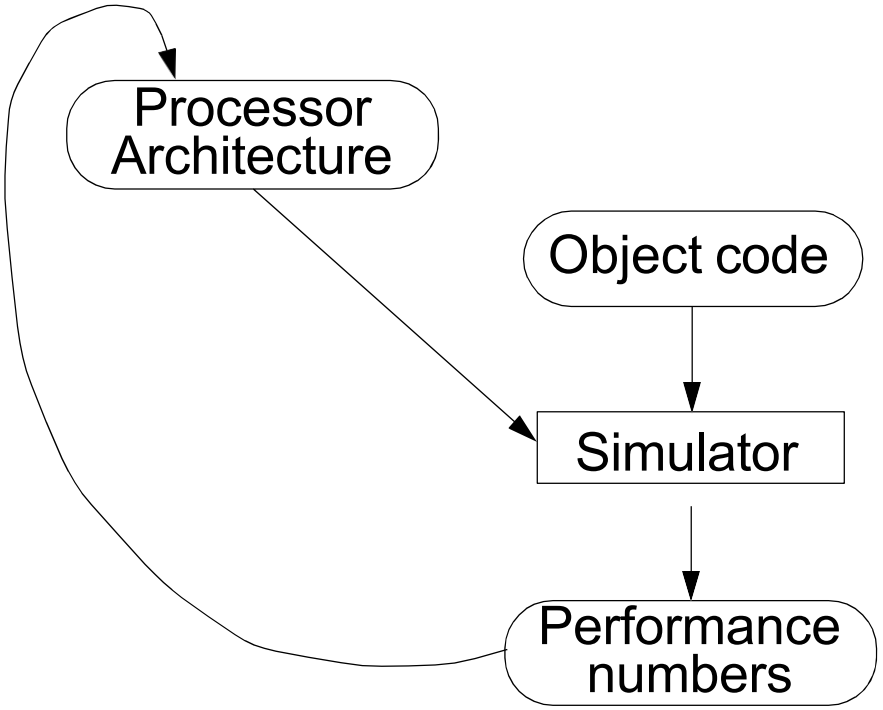
Retargetable Compiler



- An *automatically retargetable compiler* can be used for a range of different target architectures.

  The actual code optimization and code generation is done by the compiler, based on a description of the target processor architecture. This description is formulated in a, so called, “architecture description language”.

Configurable Simulator



- Such a simulator can be configured for a particular architecture (based on an architecture description)
- The most important output produced by the simulator is performance numbers:
  - throughput
  - delay
  - power/energy consumption

Application Specific Platforms

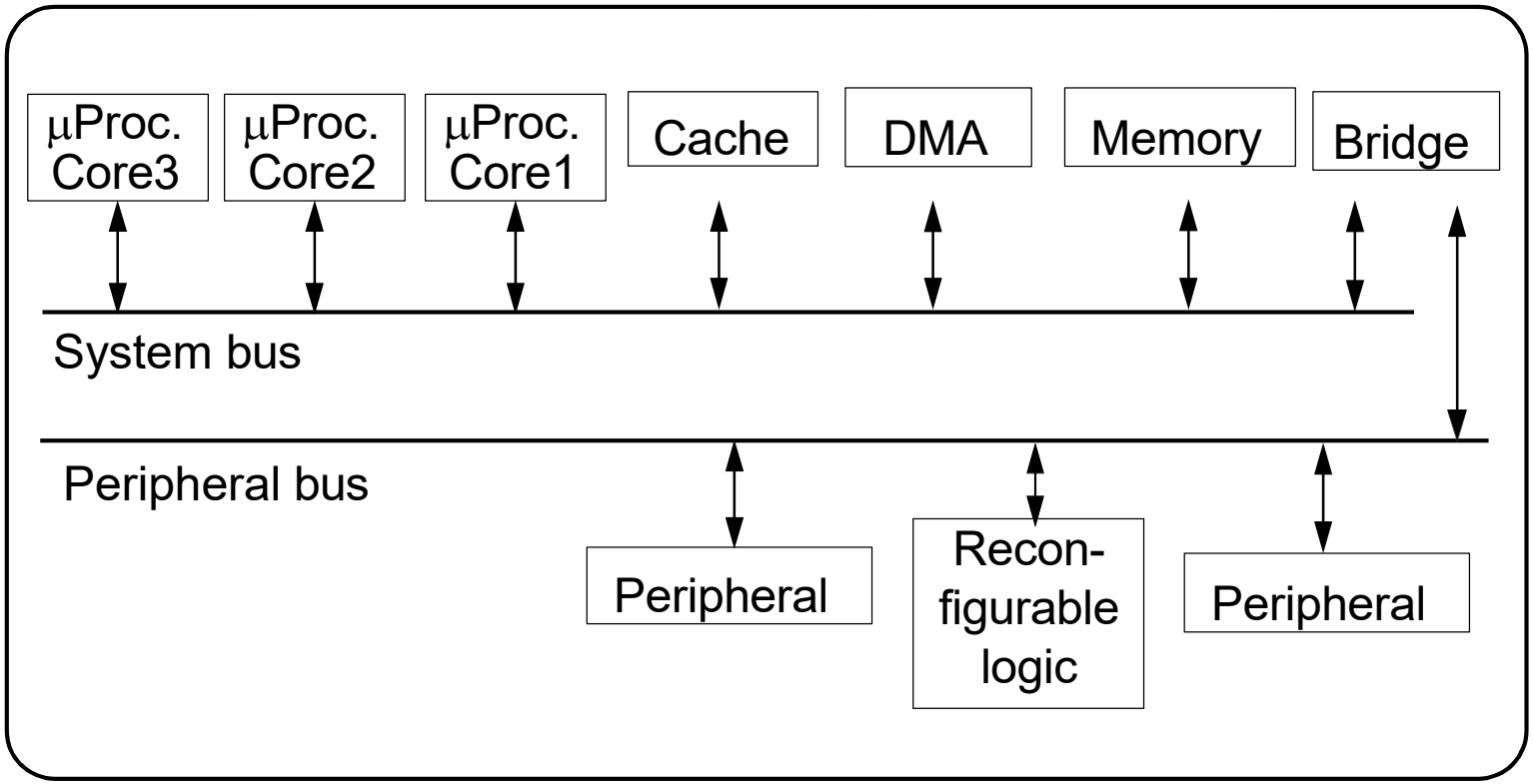
- Not only processors but also *hardware platforms* can be specialised for classes of applications.

  The platform will define a certain communication infrastructure (buses and protocols), certain processor cores, peripherals, accelerators commonly used in the particular application area, and basic memory structure.

Application Specific Platforms

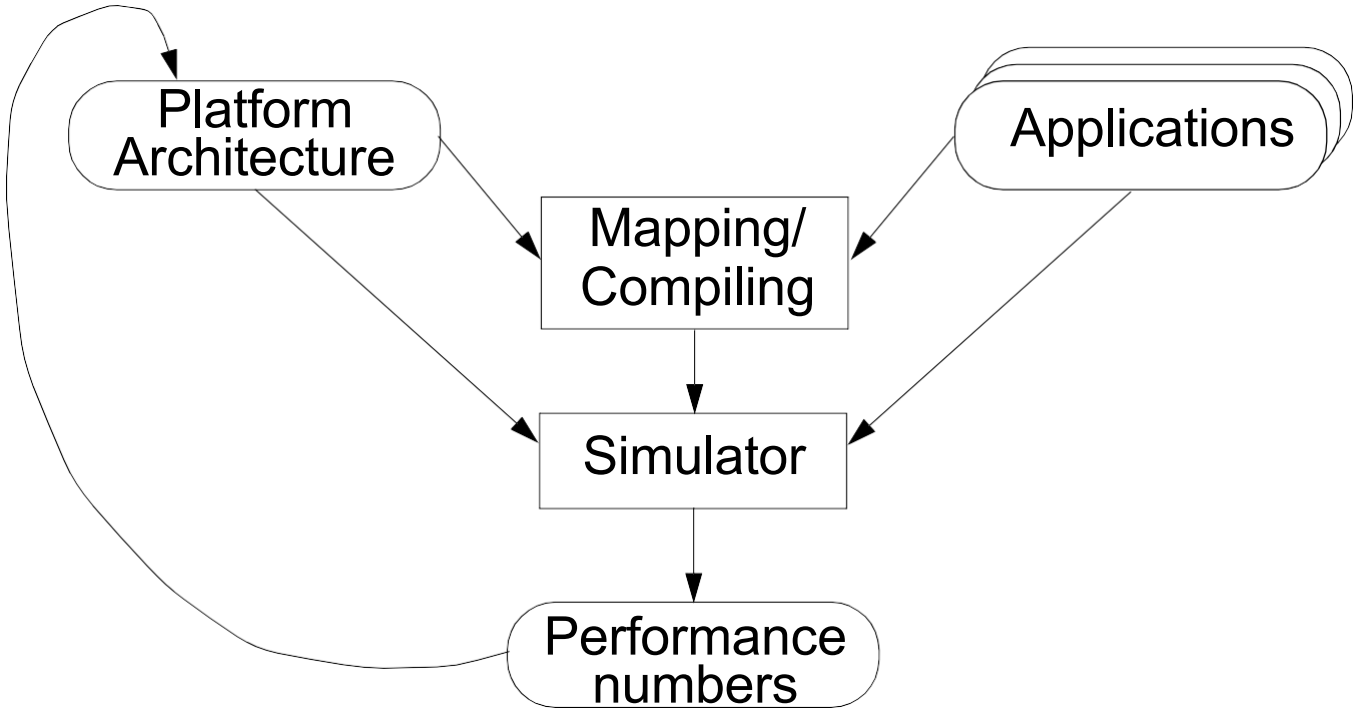
- Not only processors but also *hardware platforms* can be specialised for classes of applications.

  The platform will define a certain communication infrastructure (buses and protocols), certain processor cores, peripherals, accelerators commonly used in the particular application area, and basic memory structure.



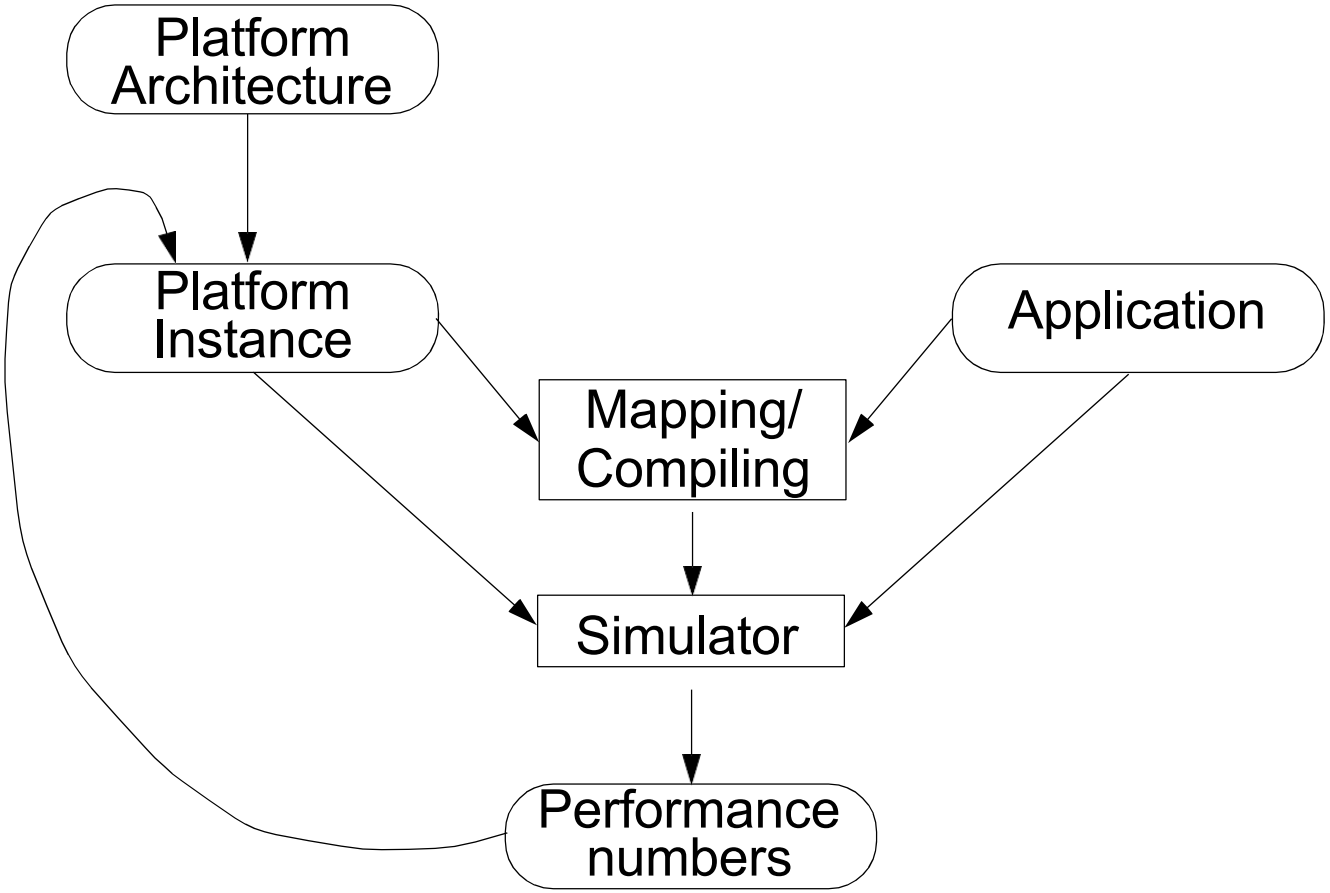
Application Specific Platforms

Design space exploration for platform definition:



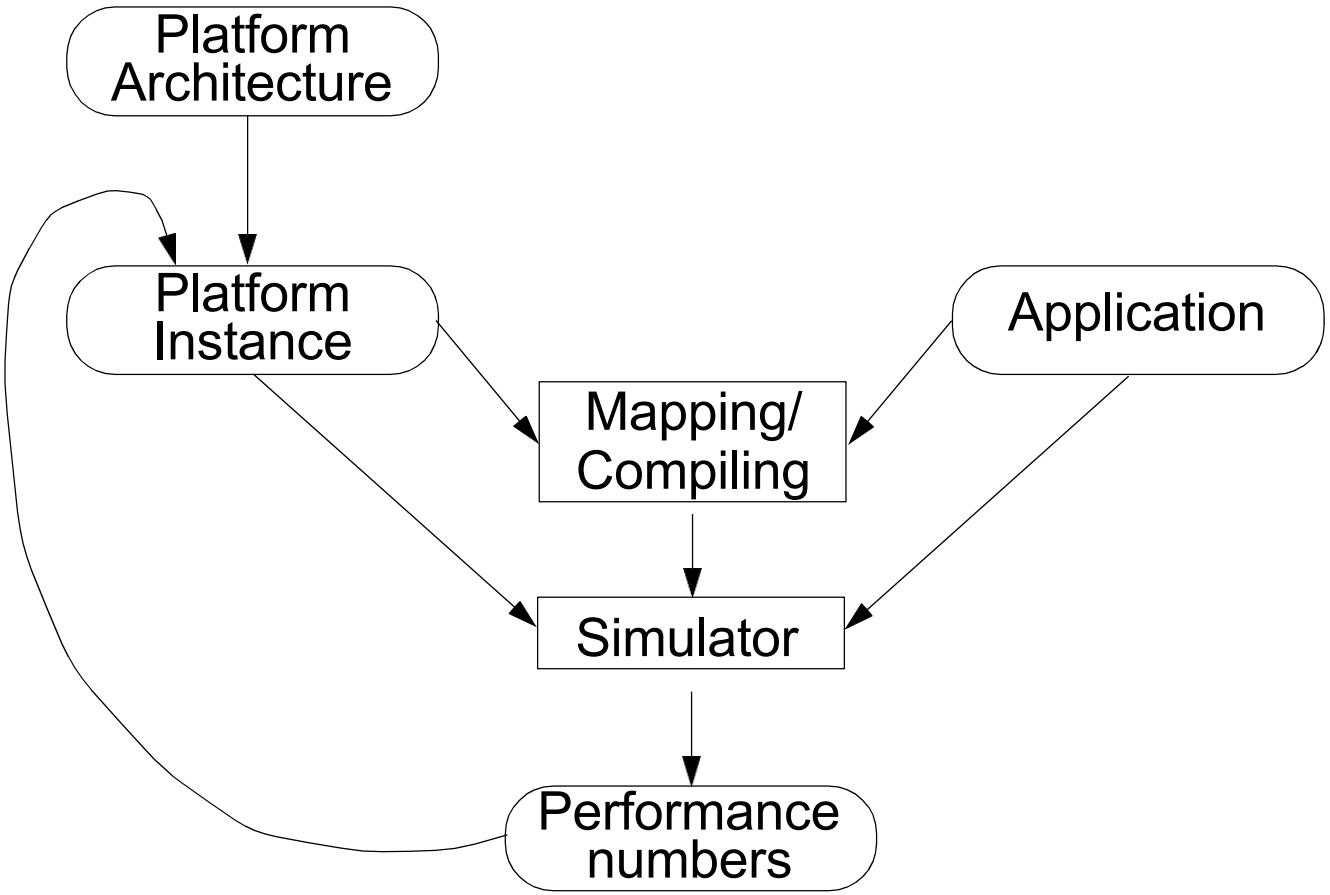
Instantiating a Platform

- Once we have an application, the chip to implement on will not be designed as a collection of independently developed blocks, but will be *an instance of an application specific platform*.



Instantiating a Platform

- Once we have an application, the chip to implement on will not be designed as a collection of independently developed blocks, but will be *an instance of an application specific platform*.



- The hardware platform will be refined by
  - determining memory and cache size;
  - identifying the particular cores, peripherals;
  - adding specific ASICs, accelerators;
  - determining the amount of reconfigurable logic.

System Platforms

- What we discussed about are hardware platforms.

- The hardware platform is delivered together with a software layer:
  hardware platform + software layer = *system platform*.
  - Software layer:
    - real-time operating system
    - device drivers
    - network protocol stack
    - compilers
  - The software layer creates an abstraction of the hardware platform (an application program interface) to be seen by the application programs.

IP-Based Design (Design Reuse)

- The key concept in order to increase designers' productivity is *reuse*.

  In order to manage the complexity of current large designs we do not start from scratch but reuse as much as possible from previous designs, or use commercially available pre-designed *IP blocks*.

  IP: intellectual property.

- Some people call this *IP-based design*, *core-based design*, reuse techniques: The process of producing a system design by reusing existing components.

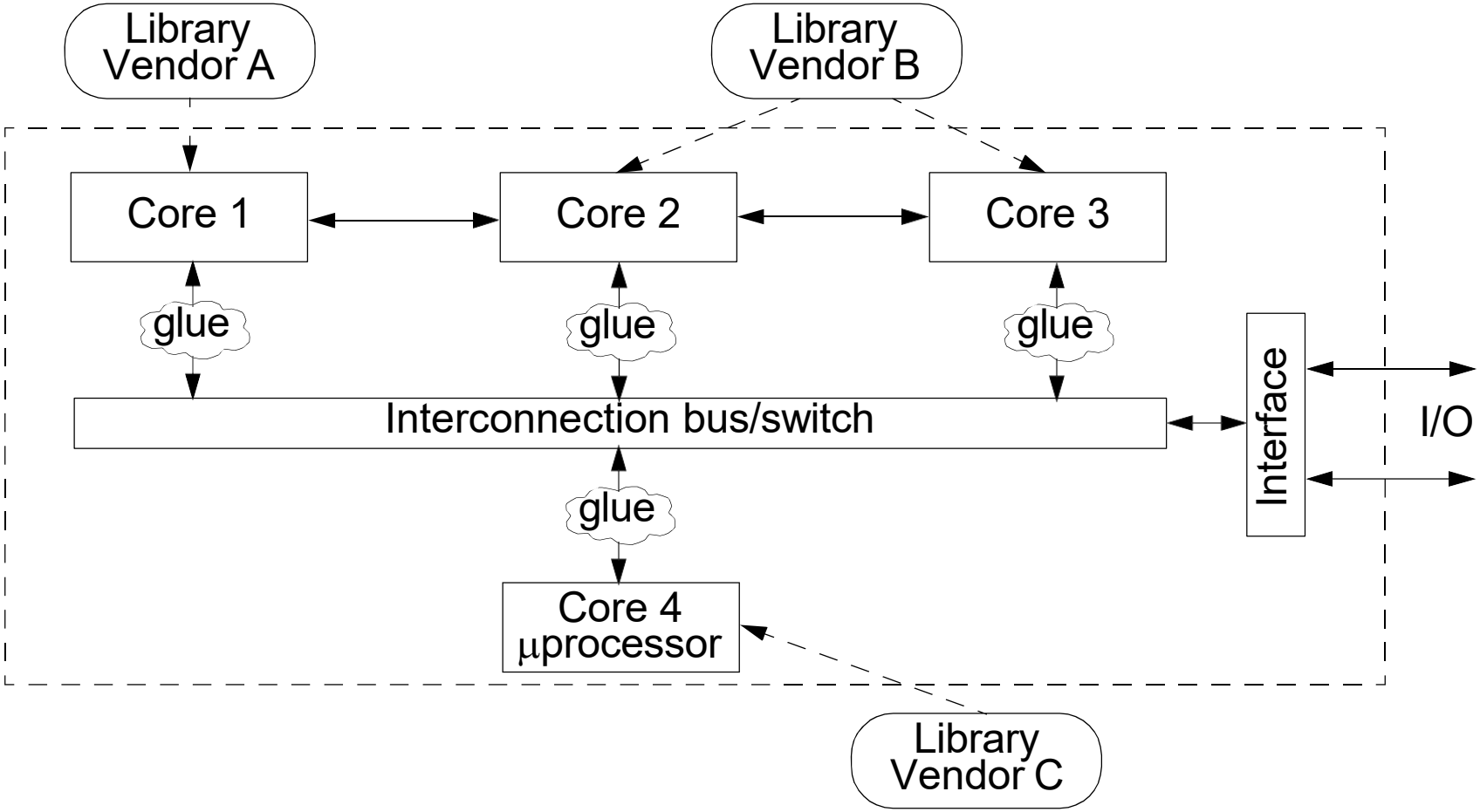
IP-Based Design

What are the blocks (cores) we reuse?

- interfaces, encoders/decoders, filters, memories, timers, microcontroller-cores, DSP-cores, RISC-cores, GP processor-cores.

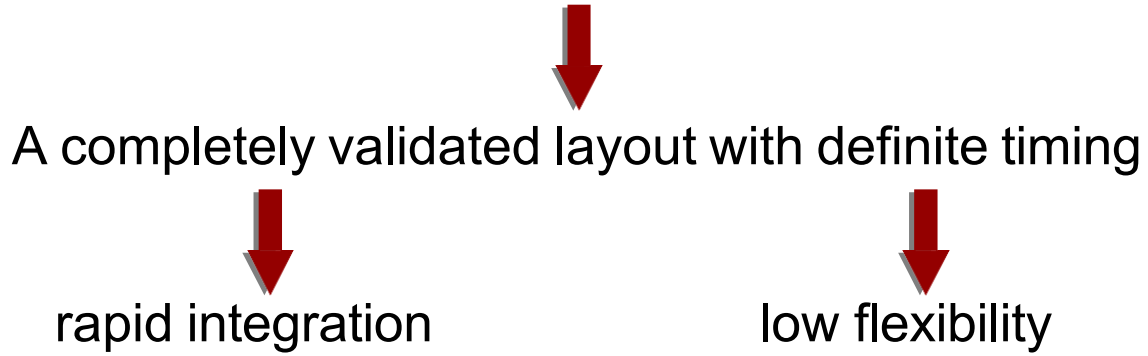
- A *core* is a design block which is larger than a typical RTL component.

IP-Based Design



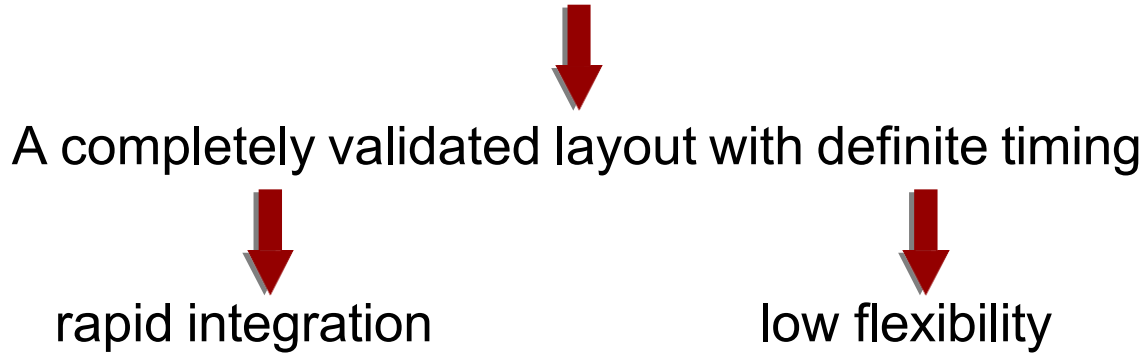
Types of Cores

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



Types of Cores

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



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

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

- Soft cores: synthesizable RTL or behavioral descriptions.

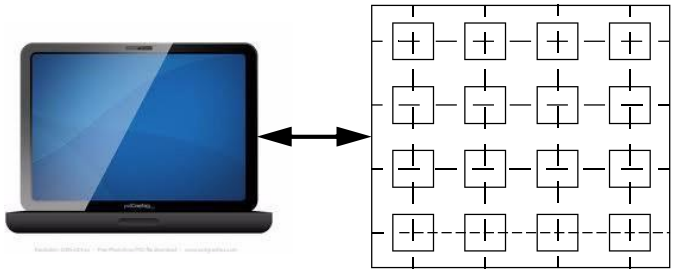
much work with integration and verification.

maximal flexibility

Reconfigurable Systems

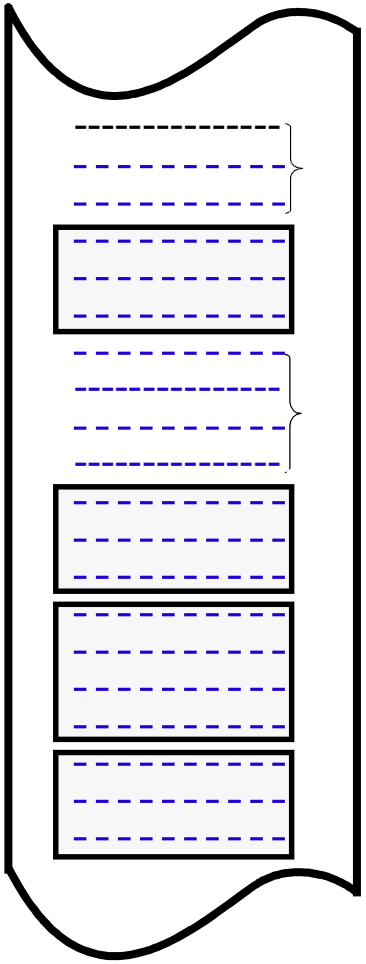
- Programmable Hardware Circuits:
  - They implement arbitrary combinational or sequential circuits and can be configured by loading a local memory that determines the interconnection among logic blocks.
  - Reconfiguration can be applied an unlimited number of times.

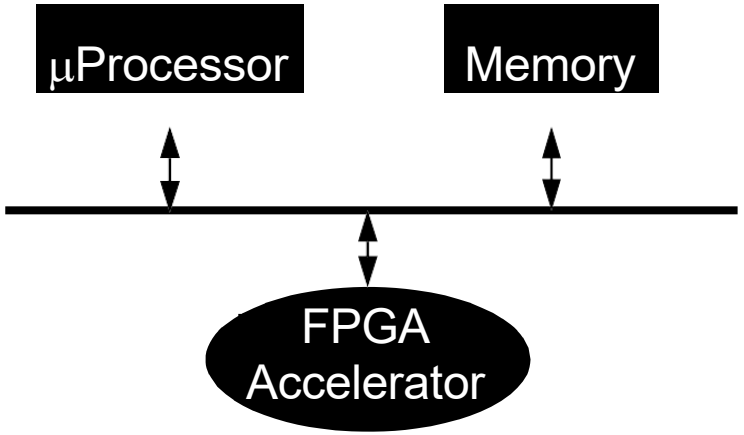
- Main applications:
  - Software acceleration
  - Prototyping



Reconfigurable Systems

Dynamic reconfiguration:



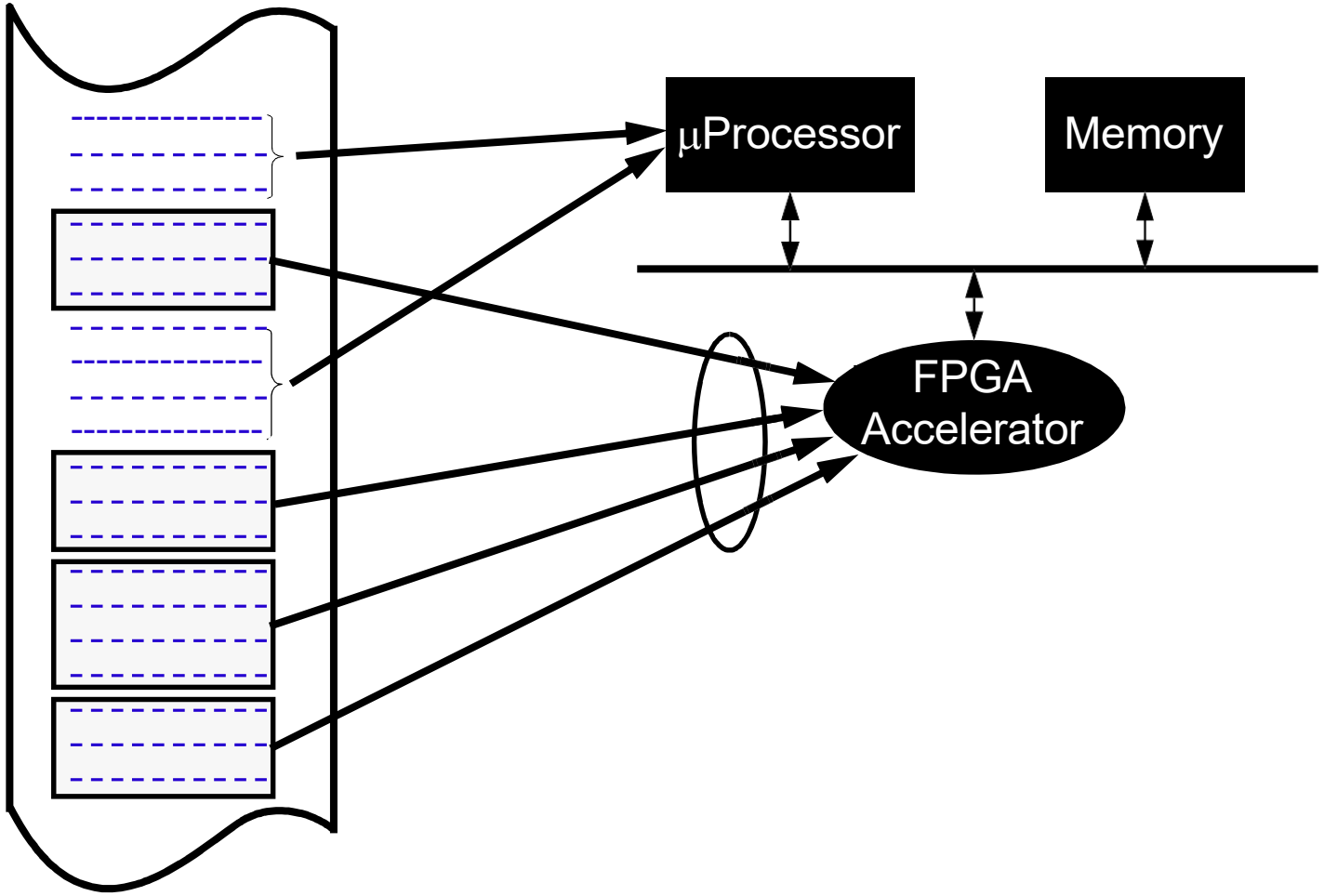


Reconfigurable Systems

Dynamic reconfiguration:

- Spacial partitioning:

  What goes into software(μProcessor) and what into hardware (FPGA)?

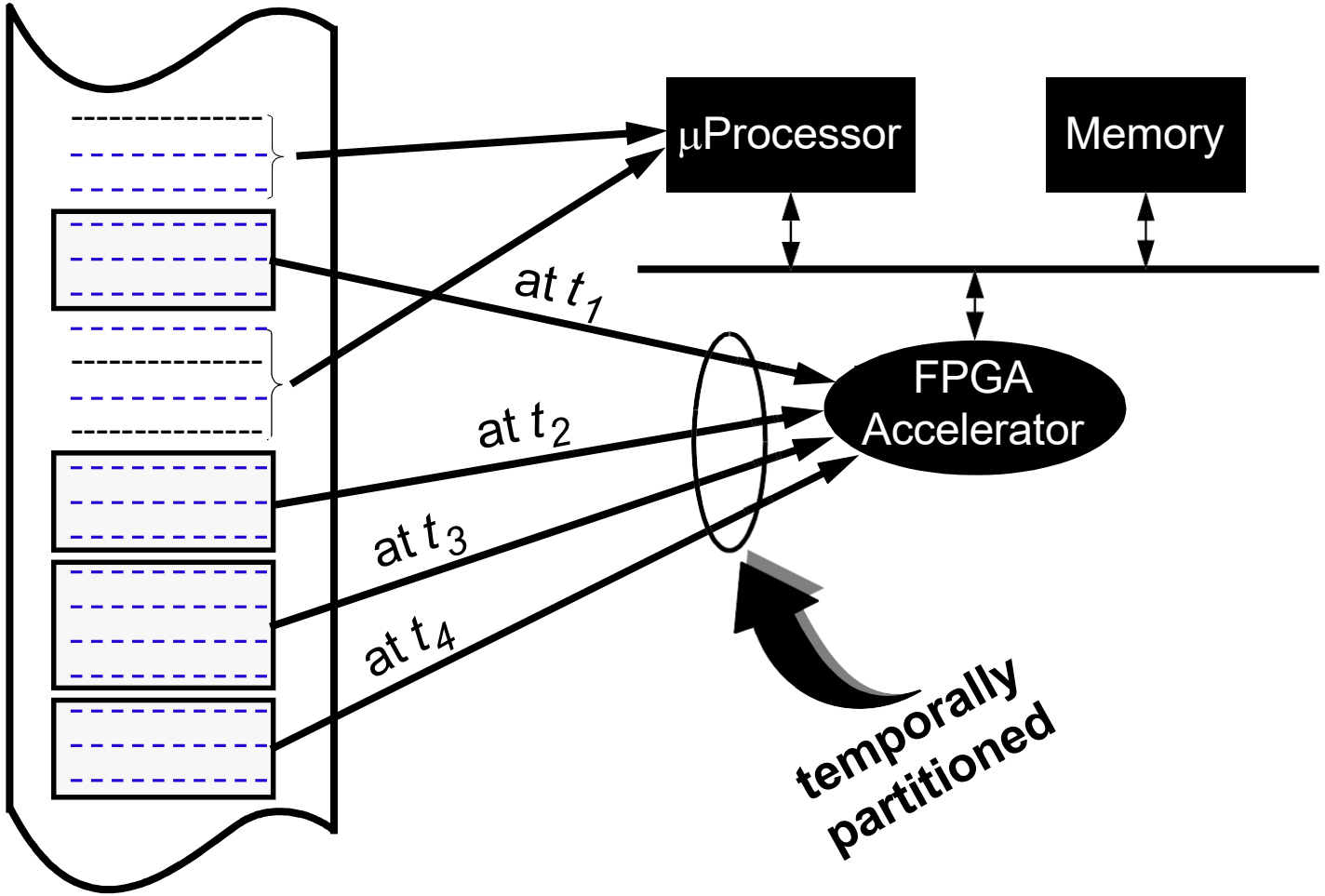


Reconfigurable Systems

Dynamic reconfiguration:

- Temporal partitioning:

  At which moment to download which module into the FPGA?



Reconfigurable Systems

System on Chip with reconfigurable datapath: