DATE 2010 – Dresden, Germany
Friday Workshops - WP3

Design Space Exploration for Run-Time Resources Management The MULTICUBE View

William FORNACIARI
fornacia@elet.polimi.it

Contributors: Simone Corbetta, Patrick Bellasi

Politecnico di Milano – Dipartimento di Elettronica e Informazione

Introduction

Outline

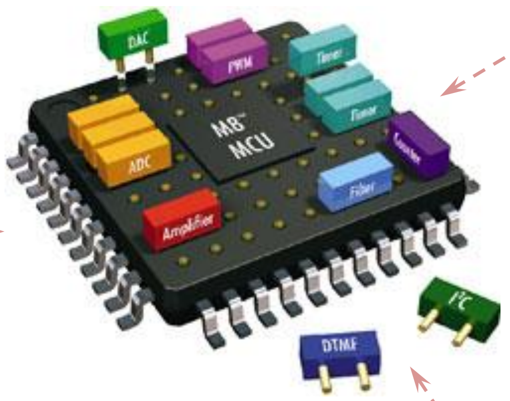
- **Introduction**
  - Application scenarios & modern requirements
- **Many-core architectures**
  - Technology perspective
  - Benefits
  - MPSoCs
- **Run-Time Resource Management**
  - Needs and value addeed of a run-time resource manager
  - Dynamic approaches
  - Power budgeting
- **Research directions**
- **The MULTICUBE View**

Introduction

Application scenarios

MOBILE

Programmable MPSoC platform



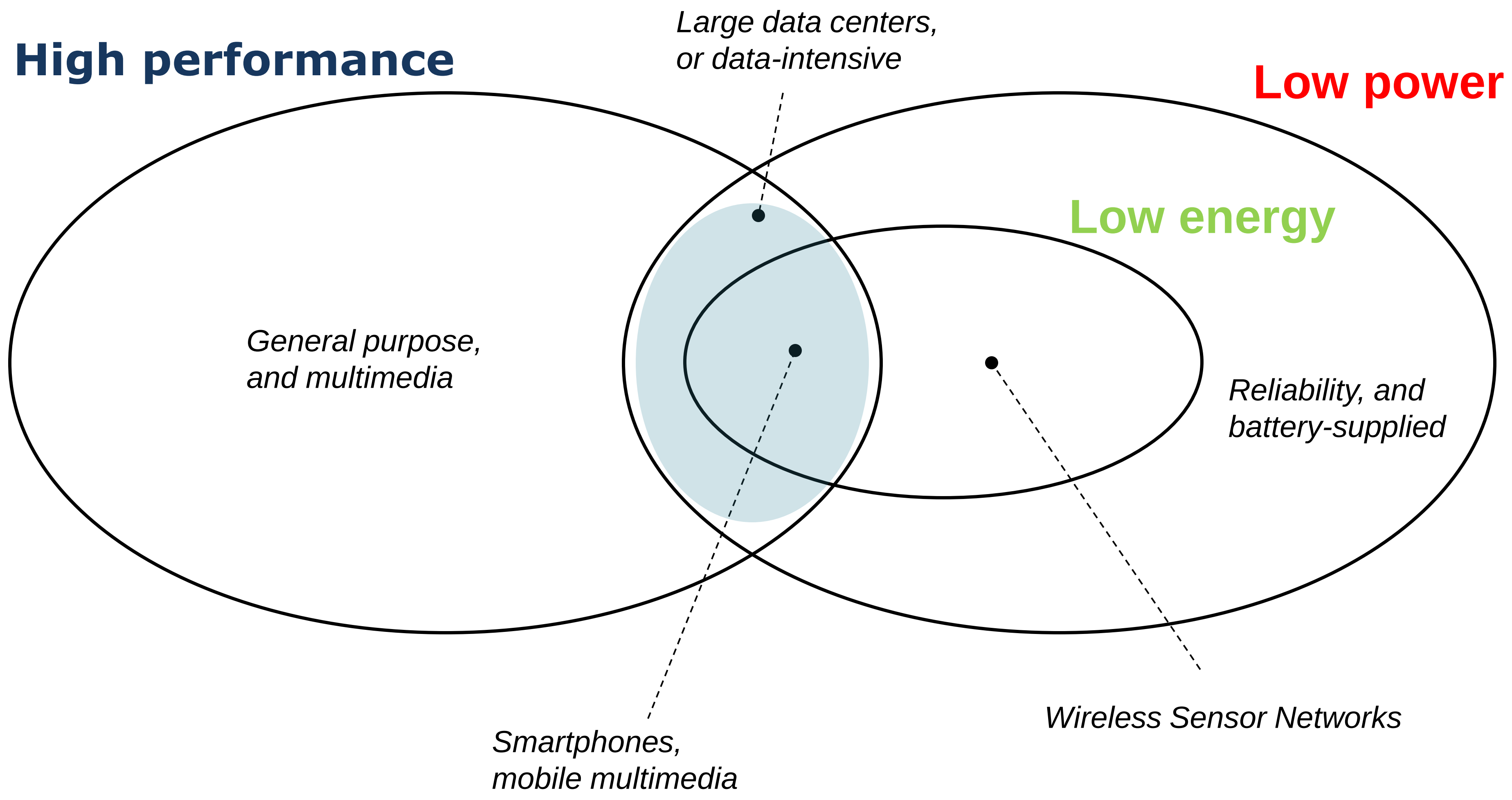
LIFE-CRITICAL

PROCESSING POWER

Introduction

Modern power/energy/performance requirements

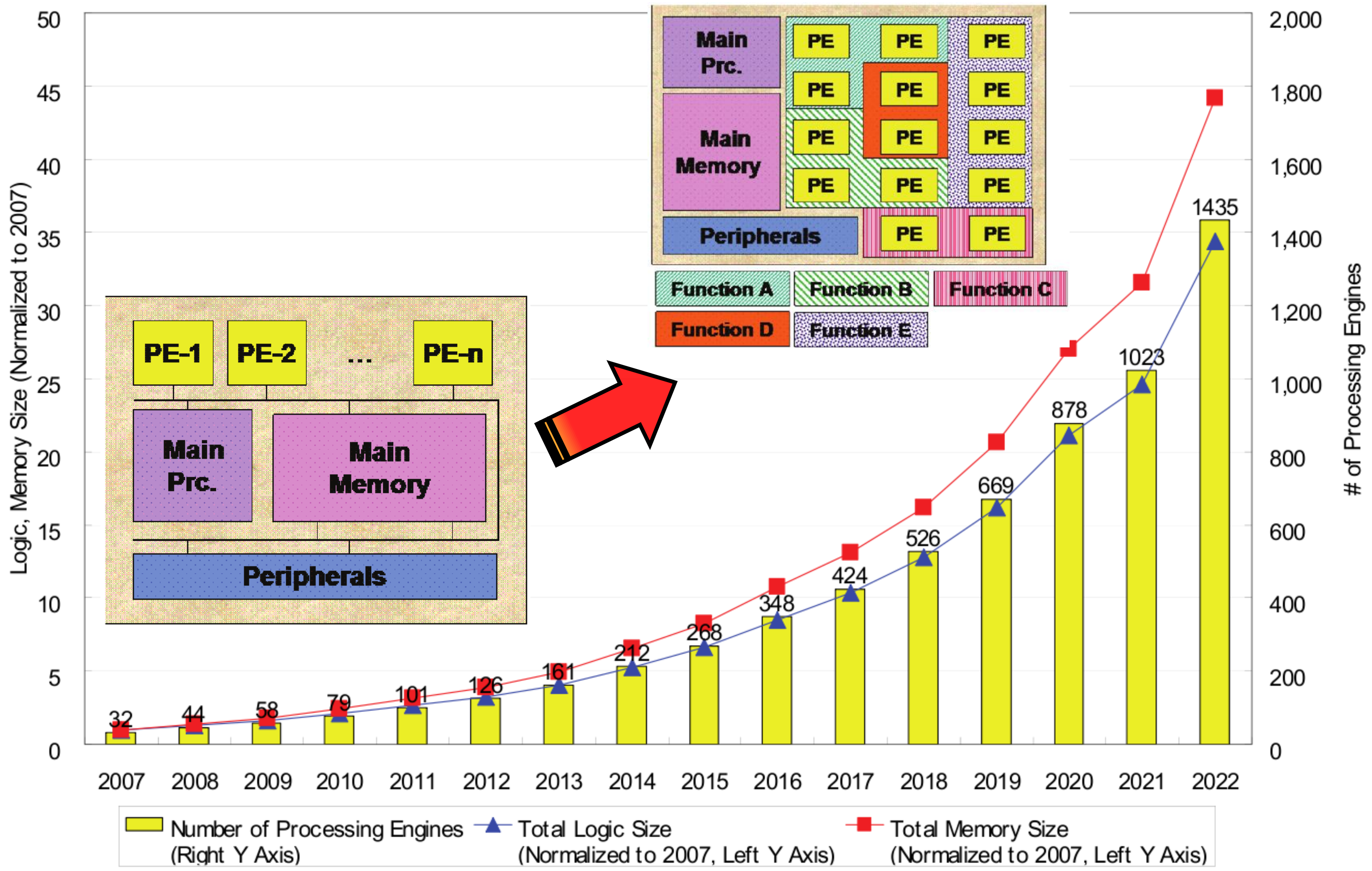
- **Wide spectrum of application requirements**


High performance
Large data centers, or data-intensive
Low power
Low energy
General purpose, and multimedia
Reliability, and battery-supplied
Wireless Sensor Networks
Smartphones, mobile multimedia

Many-core Architectures

A technology perspective

- **SoC architectures evolution**



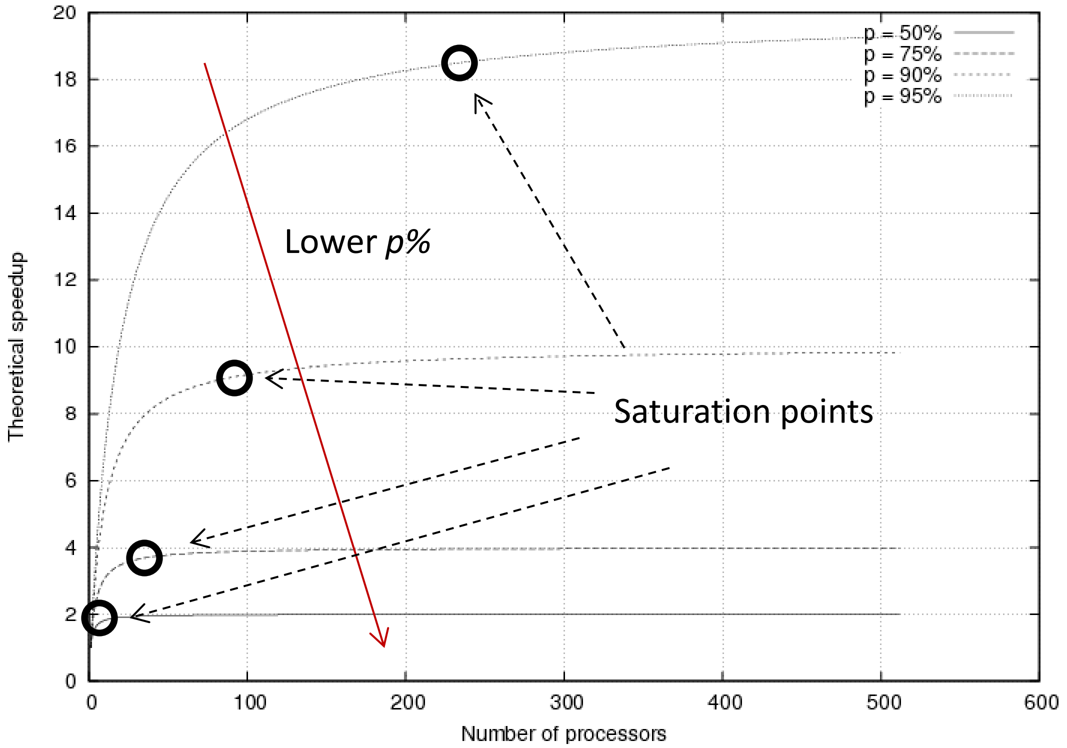
Source: Internation Technology Roadmap on Semiconductors. 2008 Edition, Design chapter.

Many-core Architectures

A technology perspective (cont'd)

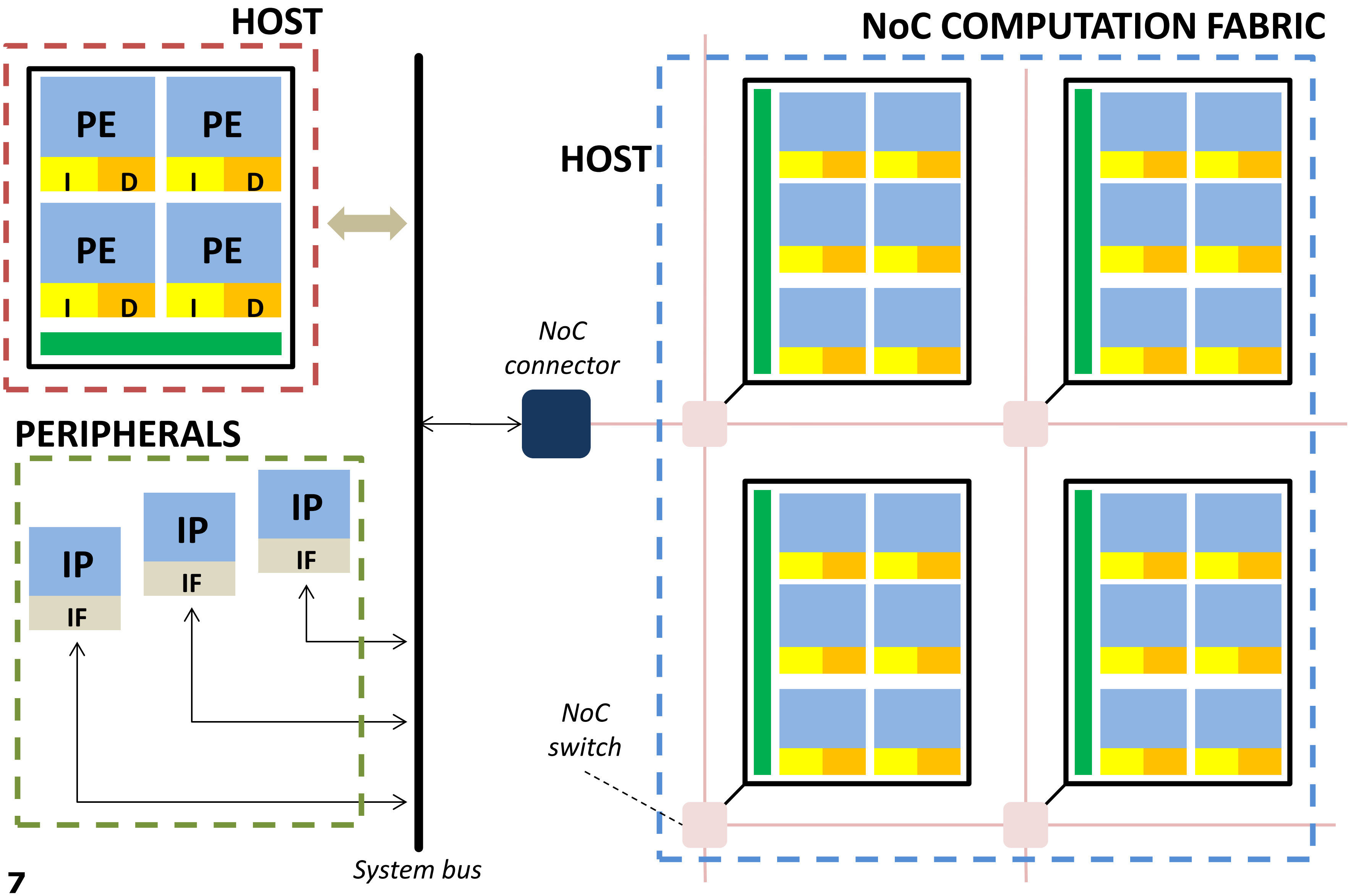
- **Amdahl's Law limits the speedup due to parallelization**
  - Bounded to percentage (*1-p)* of serial code, and number *N* of available processors
  - Gain saturate soon as *p* approaches low values

$$Speedup = \frac{1}{(1-p)+\frac{p}{N}}$$



Many-core Architectures

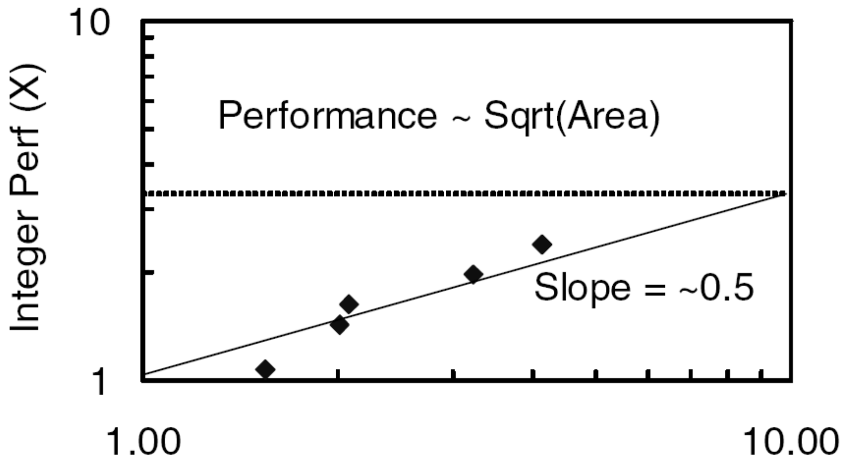
MPSoCs: heterogeneous and Network-on-Chip

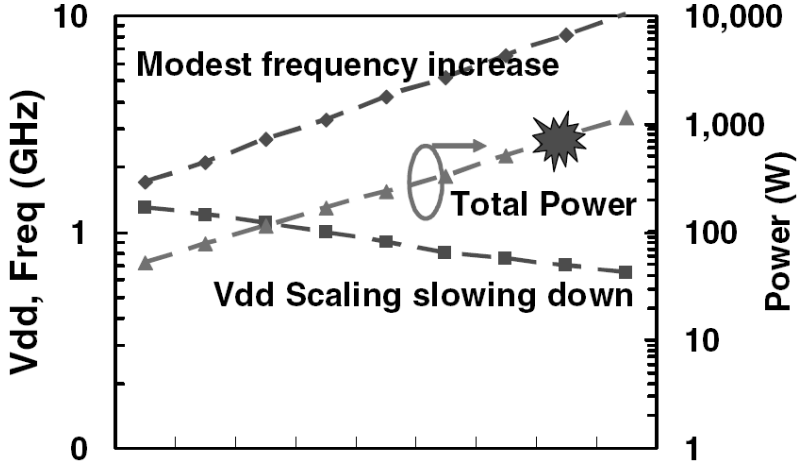


Many-core Architectures

A technology perspective (cont'd)

- **Pollack's Rule**
  - (Integer) performance goes with square root
  - Doubling complexity leads to 40% of performance gain
- **Power versus integration trend**
  - Linear: as soon as systems are more complex, power consumption gets worse





Source: Shekhar Borkar. **Thousand core chips: a technology perspective**. *Inproceedings of the 44th annual Design Automation Conference*, New York, NY, USA. 746-749. 2007.

Many-core Architectures

A technology perspective (cont'd)

- **Needs**
  - Need to find suitable architecture design for MPSoCs
    - Extend Amdahl's Law by smart architectural choices
  - Need to find suitable approaches to manage on-chip resources
    - Meeting power budget
    - Meeting user expectations and application constraints
    - Exploiting very high-performance

- **MPSoC architectures**
  - Resources management is very complex
  - Static approaches do not ensure flexibility
    - Dynamic approaches

Many-core Architectures

Typical Goals

- **Power savings**
  - By tuning each core voltage and frequency individually
- **Load balancing**
  - Use major portion of Silicon resources to exploit (not only) parallel operations
- **Lower on-die temperature**
  - Avoid hot-spots, decrease cooling costs and leakage power consumption
- **Reliability**
  - From previous reasons, and for persistence
- **Scalability**
  - Retarget MPSoC platform to different application scenarios

Many-core Architectures

Main issues

- **On-chip resources**
  - Heterogeneous (DSPs, CPUs, GPUs, generic PEs…)
  - Asymmetric w.r.t. performance and power consumption
- **On-chip applications**
  - Multiple scenarios for the same application
  - Multiple applications for the same device (true physical parallelism)
  - Cooperation or Competition
- **On-chip communication is a relevant bottleneck**
  - Collaborative application model must ensure performance, delays and low-power operation

Resources Management

The need for a run-time approach

- **Static approaches**
  - Optimal solution can be found for any application without impact on system performance
  - Design and compile-time lack of flexibility
- **Modern applications exploit dynamic behavior**
  - Static approach no more suitable for run-time requirements
  - Need to manage resources at run-time
    - Upon new application arrival
    - Upon new constraint asserted
    - Upon system fault
    - Upon resources conjestion (e.g., NoC links)
    - …

Resources Management

Benefits of run-time management

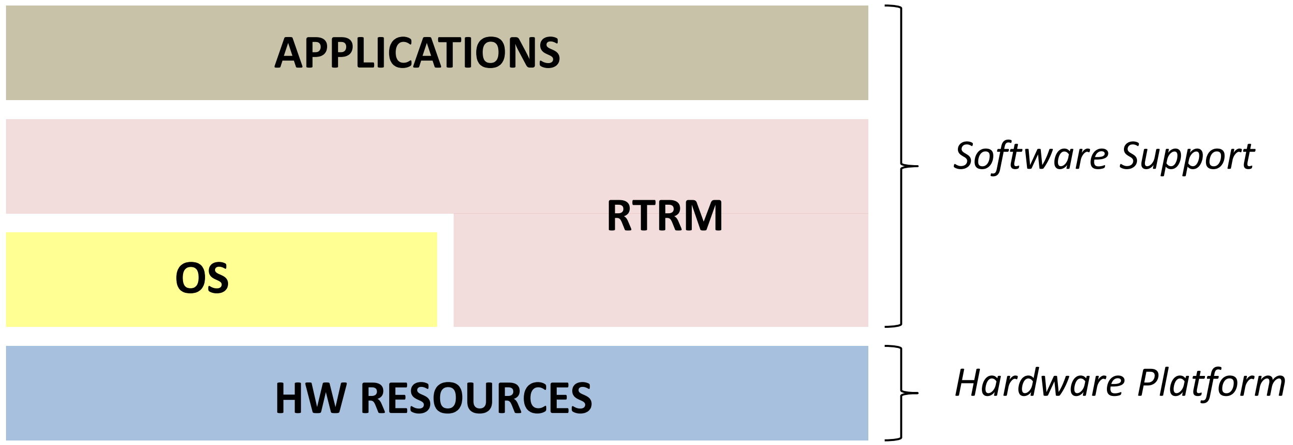
- **Flexibility**
  - Changing requirements and new applications
- **Adaptability**
  - Changing application scenario and requirements

- **Energy efficiency**
  - Power and thermal balancing for energy efficiency
- **Reliability**
  - Provide run-time solutions to application faults
- **Resource utilization**
  - Increase chip area utilization

Resources Management

Run-time manager

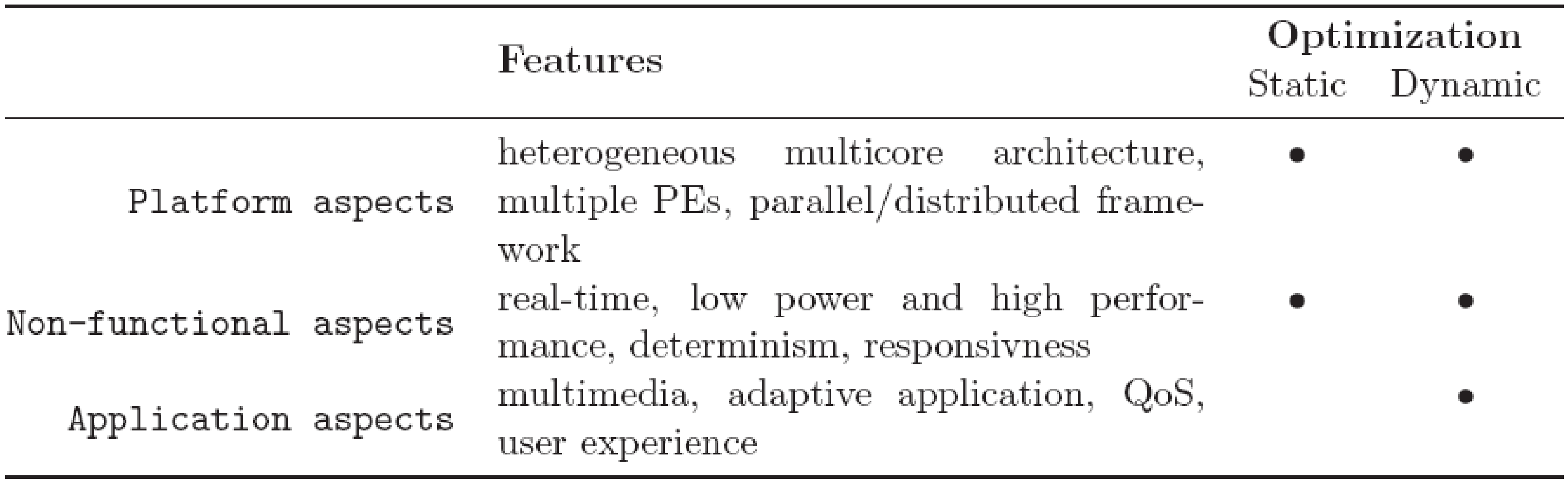
- **A run-time manager is required to take decisions and apply choices**
  - A software layer atop the hardware and below the applications
  - Provide applications with *flexible* and *transparent* resource management



Resources Management

Run-time manager components optimization

- **Different aspects of the same system have to be considered while managing resources**
  – Optimization can be performed statically, dynamically or through a smart mix of the two approaches



	Features	Optimization	
		Static	Dynamic
Platform aspects	heterogeneous multicore architecture, multiple PEs, parallel/distributed framework	●	●
Non-functional aspects	real-time, low power and high performance, determinism, responsivness	●	●
Application aspects	multimedia, adaptive application, QoS, user experience		●

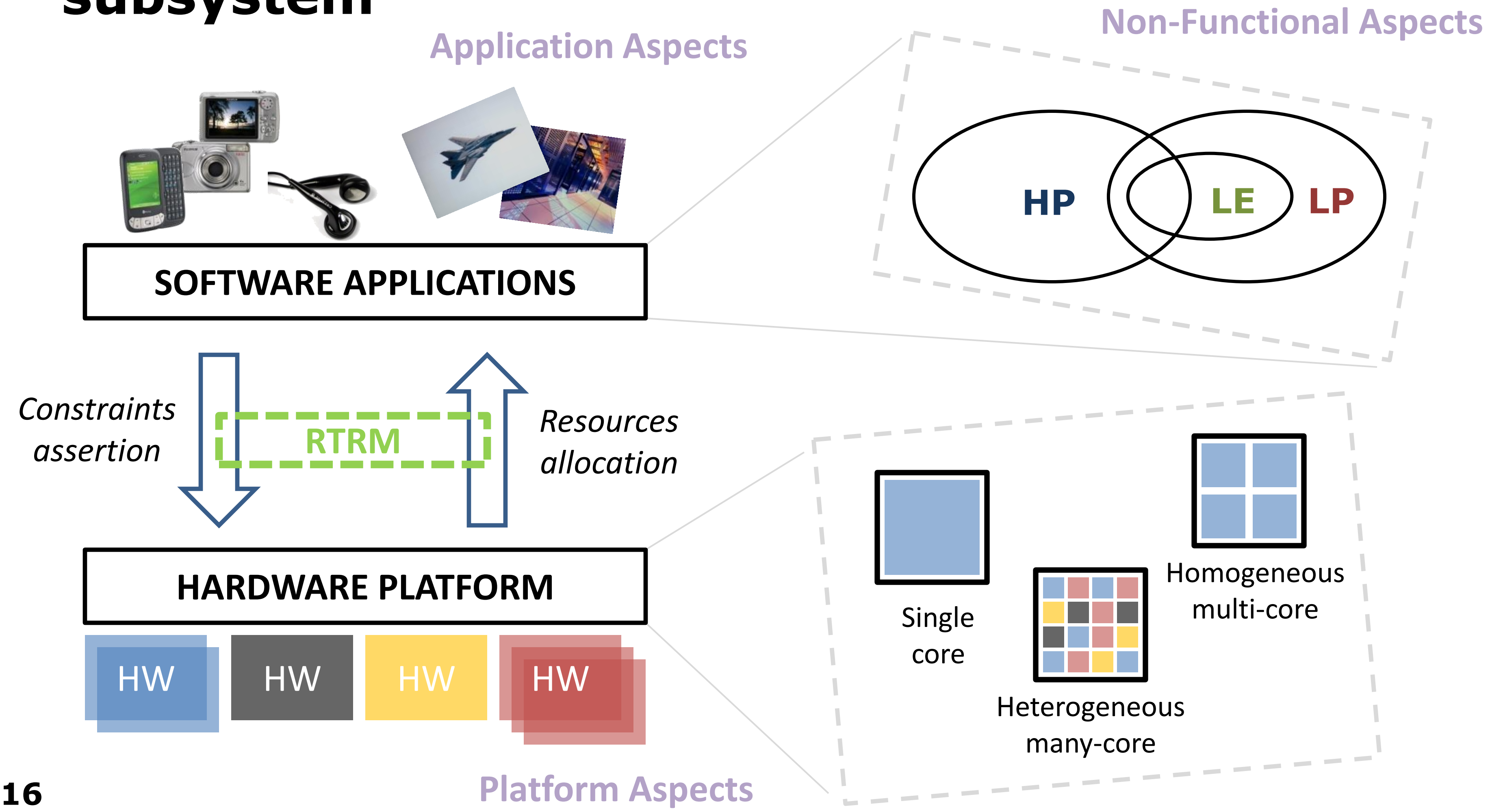
Many problems are interconnected and they can never be solved by looking at each in insulation [….] where do you start? By changing the way we are seeing the world…

[quotes from the Sangiovanni-Vincentelli Keynote @ DATE 2010]

Resources Management

Run-time manager components mapping

- **Each aspect can be mapped to a specific SoC subsystem**



Resources Management

Design and implementation space

- **Design and implementation space defines the features of the resources manager**
  - *Distribution*
    - Distributed manager
    - Centralized manager
  - *Flexibility*
    - Adaptive
    - Non-adaptive
  - *Implementation aspects*
    - Hardware acceleration
    - Software

Resources Management

Run-time approaches

- **There exist different techniques to manage on-chip resources dynamically**
  – *Task migration*
    - Directly manage executing tasks

    → Load balancing
  – Memory hierarchy management
    - Manage caches

    → Data locality and memory switching power
  – Dynamic thermal (distribution) management
    - Manage tasks, by moving them from overloaded processors

    → Power distribution (focus on static power), avoid hot-spots
  – Dynamic power management
    - Manage power resource directly

    → Dynamic and static power reduction

Task Migration

Main issues

- **Task migration requires communication between source and destination tile**
  - Can be expensive if PEs are physically far each other
  - Executing PE (source and destination) have to communicate with the host
  - Memory hierarchy traversal
  - NoC communication inserts non-null latency

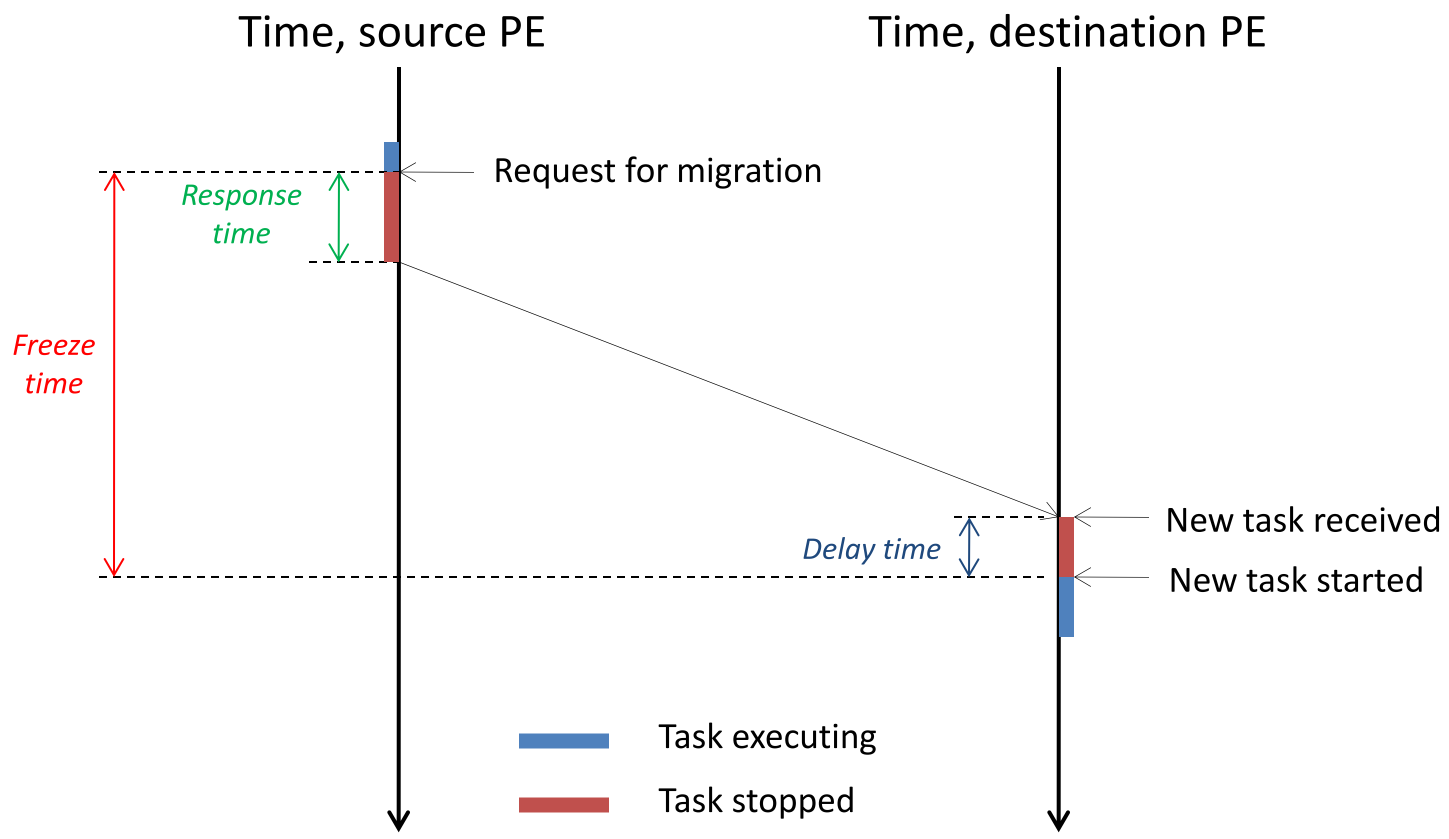
- **Freeze time interval may be too wide**
  - Real-time tasks are not suitable candidates for migration

- **Revenue estimation**
  - How to select *when* and *where* to move a task?

Task Migration

Basic principles

- **A task is moved from the source node (a PE) to the destination node (another PE)**



Task Migration

Performance measures

- **Response time**
  - Time required to initiate migration once the request has been received
- **Freeze time**
  - Time interval in which the task is not executing
- **Delay time**
  - Delay required at the destination node to insert task in processor queues
  - Accounts for destination node internal status upon task arrival and new task insertion

Task Migration

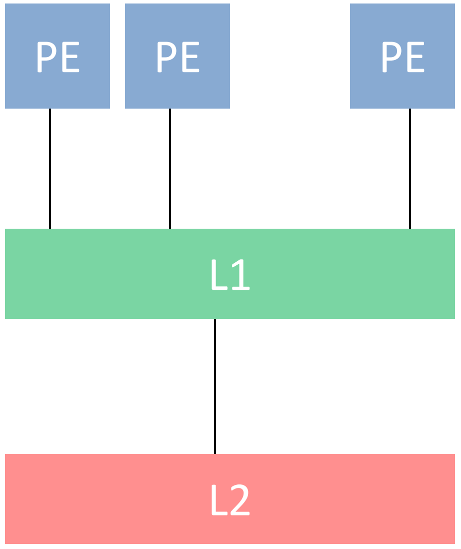
Benefits

- **Why should we take care about task migration?**
  - Load balancing
    - Move tasks from overloaded processor to available processor
    - Access to distributed processing power
  - Power and thermal distribution
    - Higher performance and reliability
      - → Meet application requirements
    - Lower power requirements
      - → Compensate battery discharging rate on mobile systems
  - Resource locality
    - Remote data access vs migration toward data
  - Fault resilience
    - Detect a fault, and move to an appropriate safer place

Memory Management

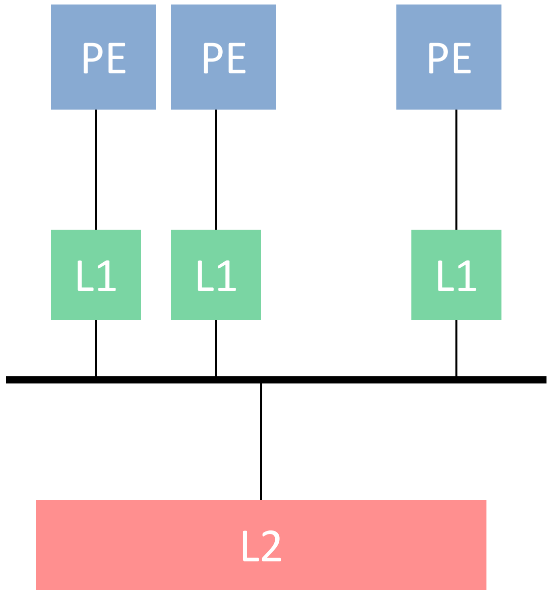
Cache architectures templates

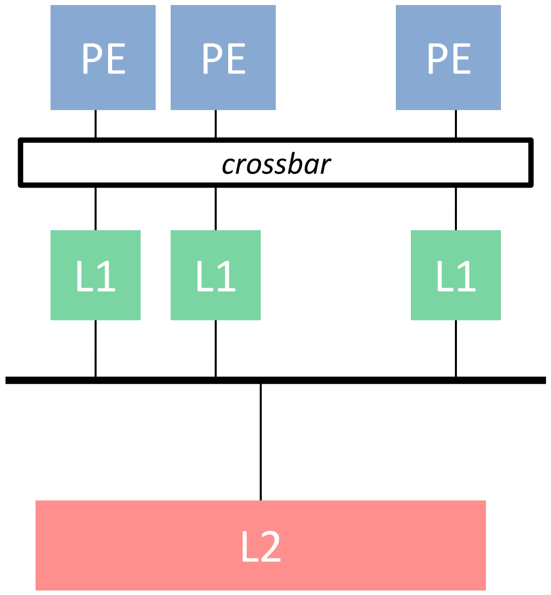
- **Different memory organizations for different purposes**
  - Trade-off with respect to power consumption and memory access performance



L1 and L2 shared

L1 private, L2 shared through bus



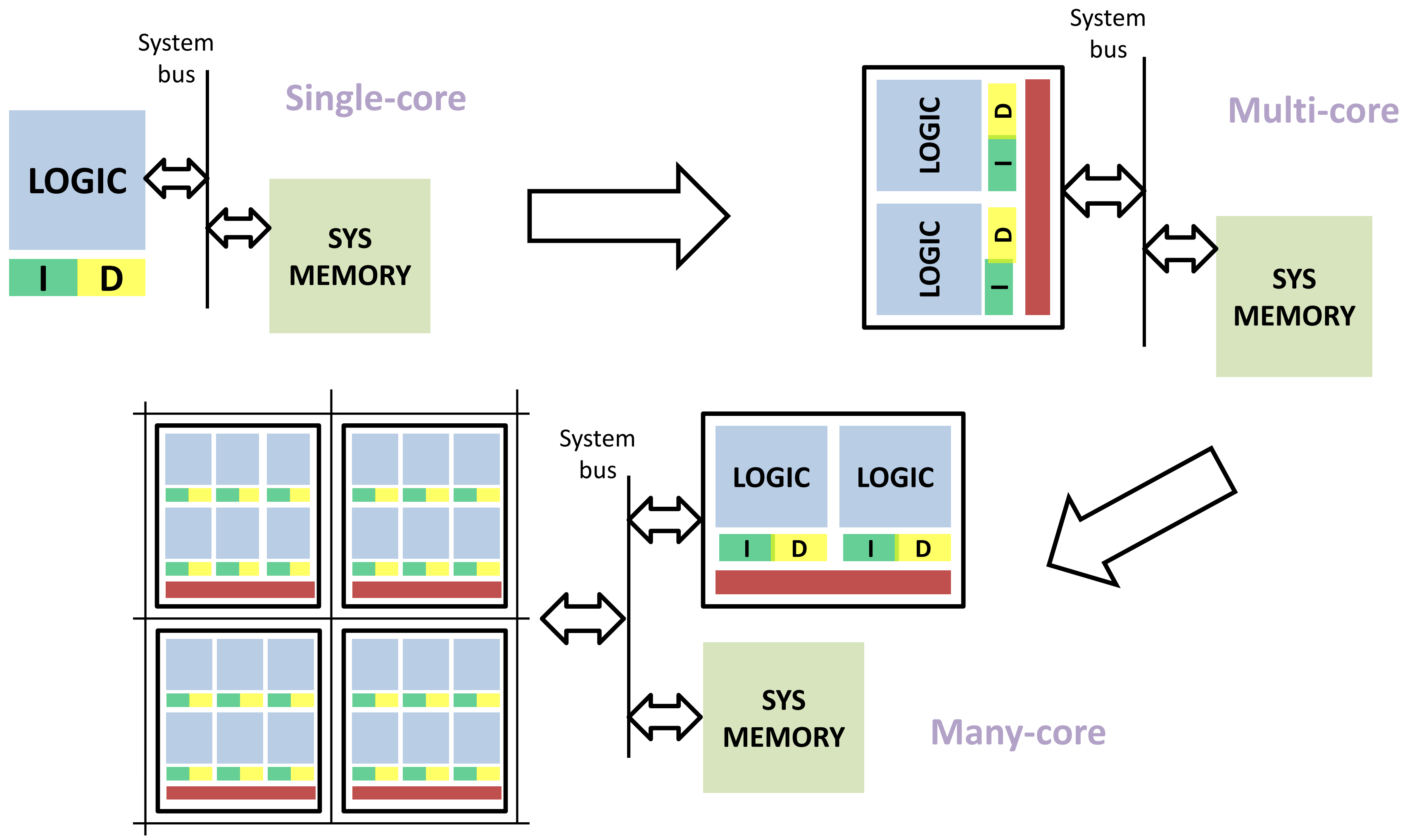


L1 shared thorugh crossbar, L2 shared through bus

Memory Management

Memory hierarchy evolution

- **The evolution of SoC memory architectures**



Memory Management

Cache architectures

- **Memory hierarchy constitutes a significant portion of the system resources**
  - Energy is spent during memory accesses
- **Active energy is consumed during data accesses**
  - Access bit lines for storing data
- **Standby power is consumed during no-operations**
  - Unwanted bit lines activation

- **Optimization can be done from software implementation guidelines or hierarchy architecture optimization**

Memory Management

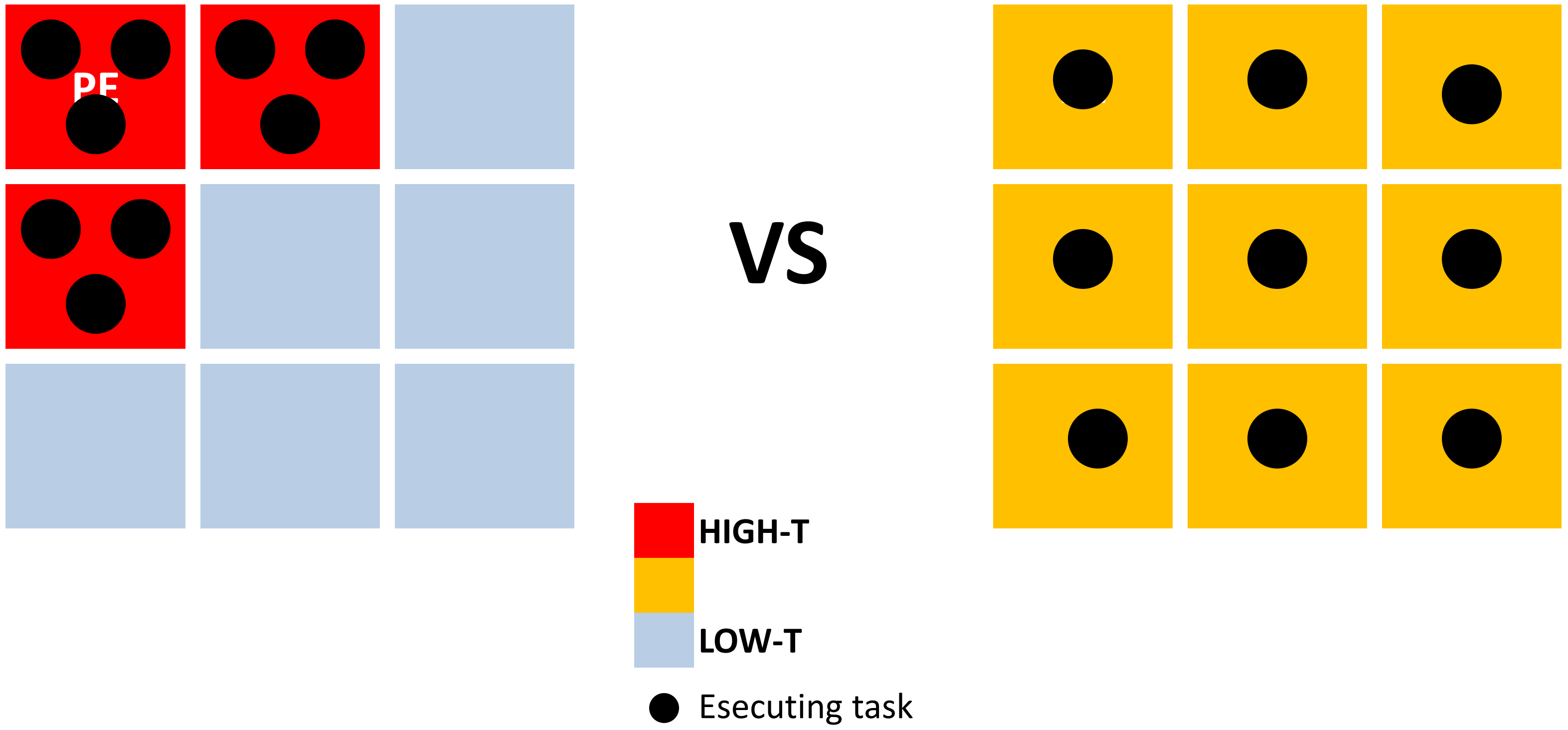
Major issues

- **Hierarchy management**
  - Hierarchy depth and complexity
  - Memory consistency
    - Cache coherence
- **Dynamic power management**
  - Operate on data locality
  - Avoid switching memory when unnecessary
- **Static power management**
  - Operate on low-power states
  - Retention voltage, voltage scaling

Thermal Distribution

Dynamic thermal management

- **Thermal distribution**
  - Distribute resources so that thermal profile of the integrated circuit is balanced



Thermal Distribution

Benefits

- **Thermal distribution spread power sources chip-wide**
  - Leakage phenomena are reduced locally (at each PE)
  - Each core has improved reliability
    - High temperature reduces stability, reliability and performance
  - Lower cooling costs
    - Cooling circuit is much more efficient

- **Thermal distribution is provided at chip-level, not at core-level**

Thermal Distribution

Dynamic management

- **Thermal distribution management can not be addressed a priori (i.e., statically)**
  - There is no knowledge on work load
  - Statistical approaches lack of flexibility
    - What about if new applications arrive?
    - What about deviation from predicted behavior?

- **Need to manage it at run-time**
  - Monitor per-core thermal (and power) profiles
  - Make choices according to
    - Thermal profile
    - Communication between tasks

Power Management

Dynamic power management

- **Basic idea**
  - Manage power resource during normal system execution
  - Hardware and software integrated approach
    - Hardware provide basic mechanisms, software provides system-wide optimization
  - Require both hardware blocks knowledge and software support exploitation
    - OS level, kernel code

- **Directly address the problem from the power view-point**
  - RTRM is a more general problem

Source: Patrick Bellasi, Simone Corbetta, and William Fornaciari. **Hardware/Software Codesign for Dynamic Power Management**.

Open research directions

Run time resource management - Summary

- **MPSoCs and application complexity**
  - Energy-efficient resources management becomes challenging
  - Need to provide both static and dynamic support
  - Flexibility and system-wide power/performance optimization
- **Different techniques exist**
  - Not so appropriate for heterogeneous highly distributed and parallel architectures
  - Need to define novel approaches for system-wide power management
- **Future works**
  - Find suitable RTRM solution that is scalable toward future manycore architectures
    - Hierarchical power management
    - Mobile embedded systems application scenarios
    - Industrial feedback and support with real circuitry

MULTICUBE Project

Partners

MULTI-OBJECTIVE DESIGN SPACE EXPLORATION OF MULTI-PROCESSOR SOC ARCHITECTURES FOR EMBEDDED MULTIMEDIA APPLICATIONS

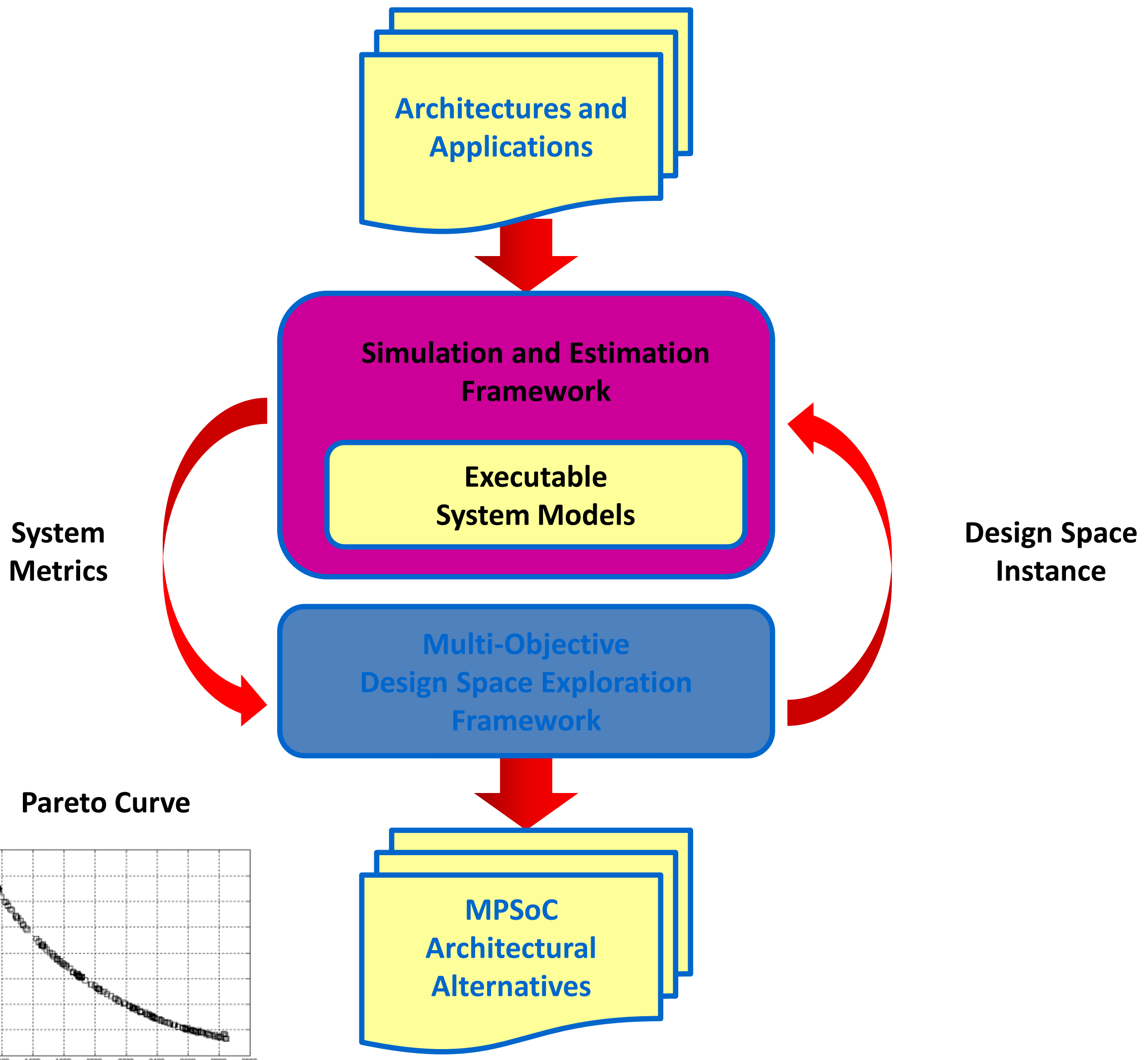
www.multicube.eu

Project Duration: from January 2008 to June 2010

Politecnico di Milano (POLIMI) – Italy (Project Coordinator)

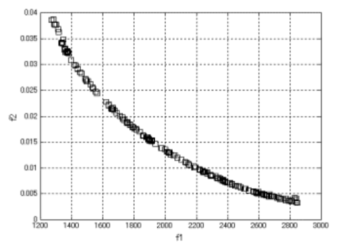
DS2 – Spain	**Università della Svizzera Italiana (ALaRI) - CH**
IMEC - Belgium	**University of Cantabria - Spain**
STMicroelectronics - Italy	**STMicroelectronics - China**
ESTECO - Italy	**Institute of Computing Technology (ICT) China**

MULTICUBE Design Flow

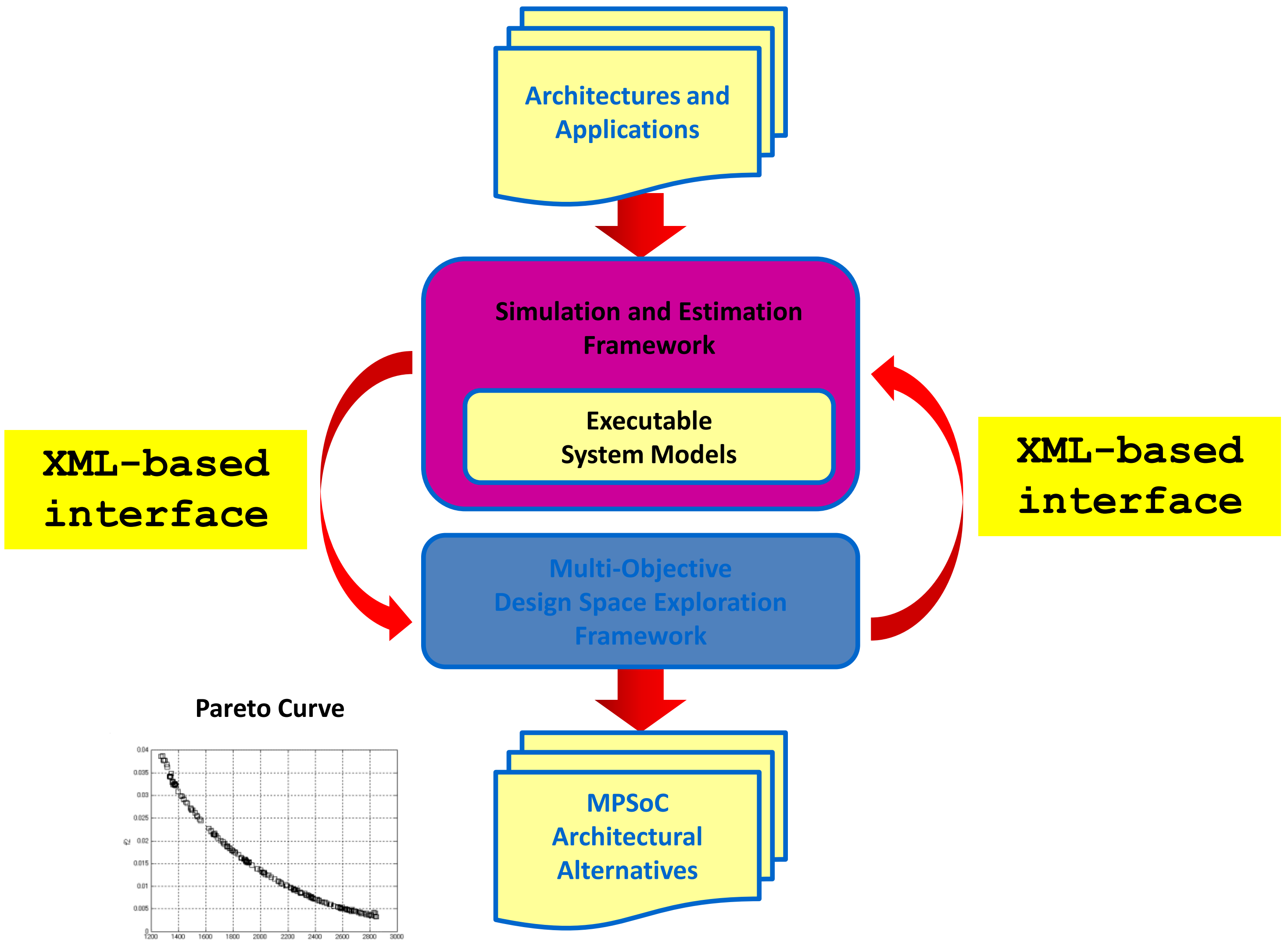


Architectures and Applications
Simulation and Estimation Framework
Executable System Models
System Metrics
Design Space Instance
Multi-Objective Design Space Exploration Framework
MPSoC Architectural Alternatives

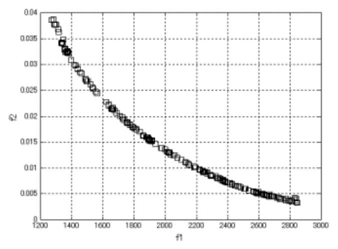
Pareto Curve



MULTICUBE Design Flow


Architectures and Applications
Simulation and Estimation Framework
Executable System Models
Multi-Objective Design Space Exploration Framework
MPSoC Architectural Alternatives
XML-based interface
XML-based interface

Pareto Curve



Automatic DSE

Main benefits

- Efficiency of DSE process is improved

  1. Minimizing the numbers of simulations to be executed by using exploration heuristics such as state-of-art evolutionary algorithms

  2. Speeding up simulations

  3. Simulating at higher abstraction levels

  4. Defining an analytical response model of the system behavior based on a subset of simulations to predict the unknown system response

MULTICUBE Explorer

What is it?

- MULTICUBE Explorer is **not** *"yet another simulator"*, there already many simulators….

- MULTICUBE Explorer is an automation and **acceleration infrastructure** to minimize the numbers of simulations to be executed during the DSE process

- MULTICUBE Explorer is a DSE infrastructure where you can easily **plug-in your own simulator**

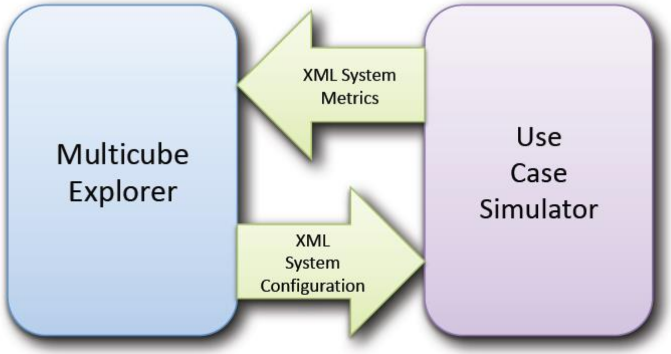
- MULTICUBE Explorer can **interface** several simulators at **multiple abstraction levels**

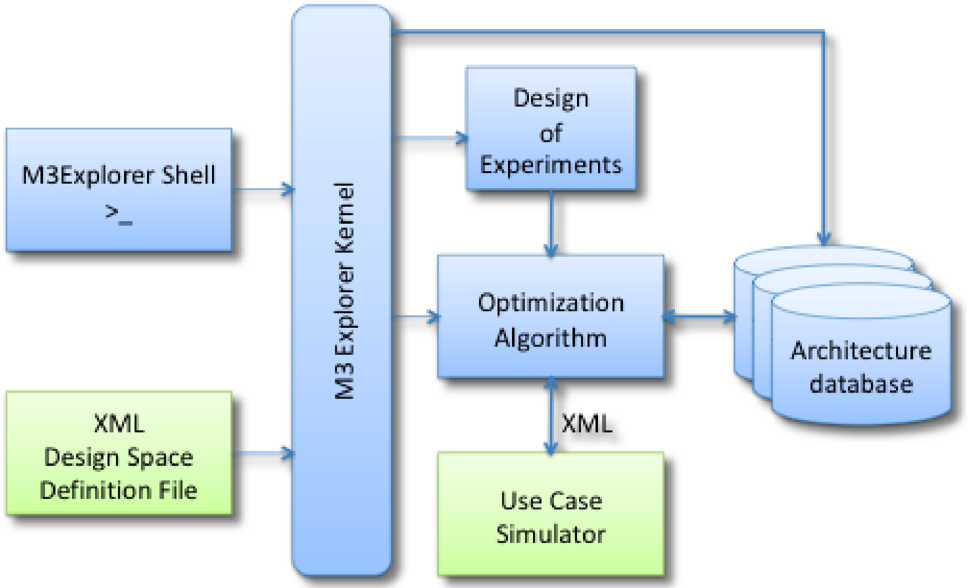
MULTICUBE Explorer

Main Goals

- **Open-source prototype exploration framework (MULTICUBE Explorer):**
  The tool enables a fast automatic optimization of parameterized system architectures towards a set of multiple objectives

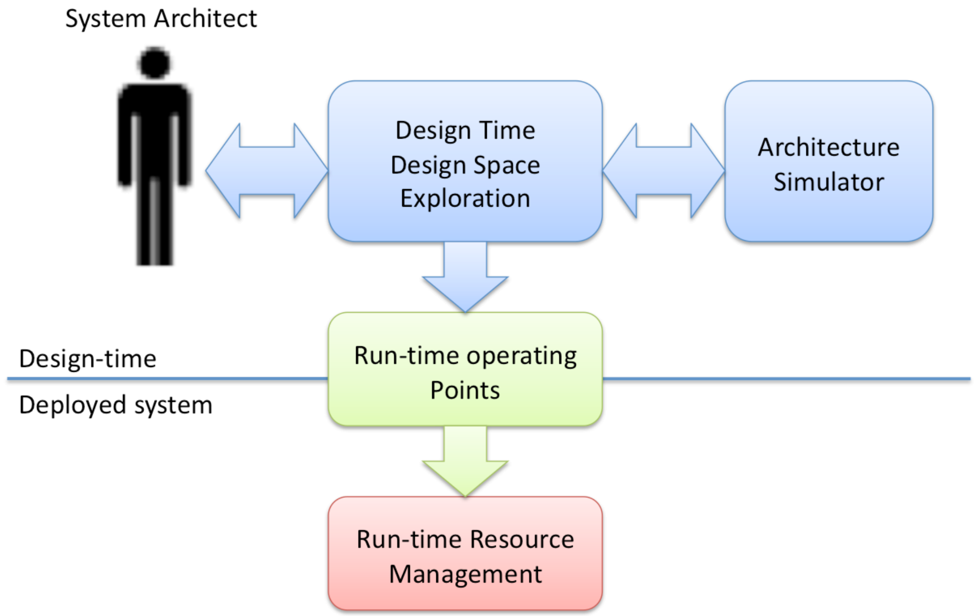
www.multicube.eu





MULTICUBE Explorer

Identification of run-time operating points

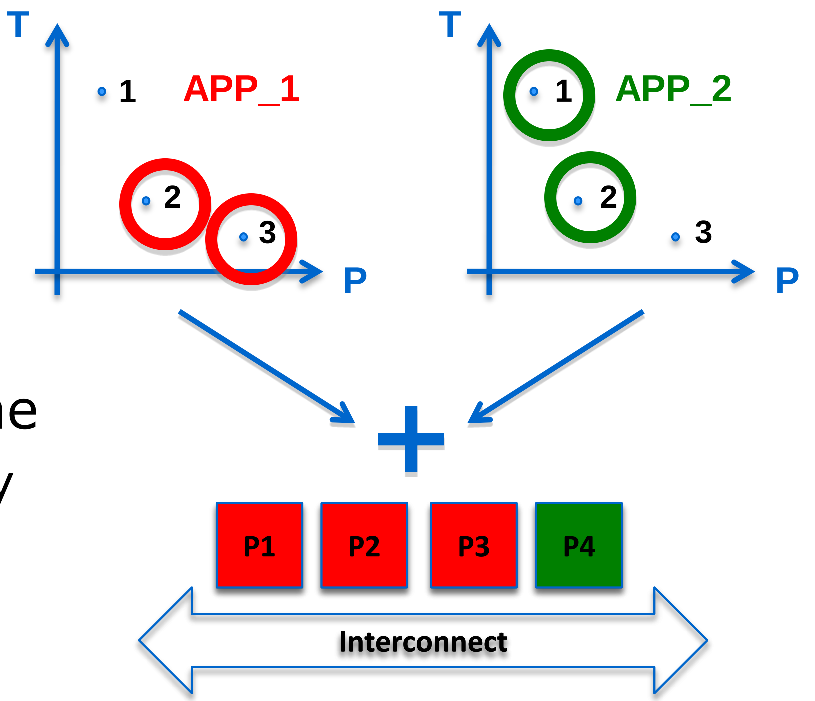

System Architect
Design Time Design Space Exploration
Architecture Simulator
Design-time
Deployed system
Run-time operating Points
Run-time Resource Management

- **Design Time DSE** of run-time configurable parameters
  - Pareto optimal Run-time operating Points
- **Run-time Resource Management**
  - Selects dynamically the pre-identified operating points
- **Value added of MULTICUBE Explorer**
  - Fast design space exploration of usage scenarios
  - Pre-computation of run-time manager behavior

A running example

Paper presented @ DATE 2010

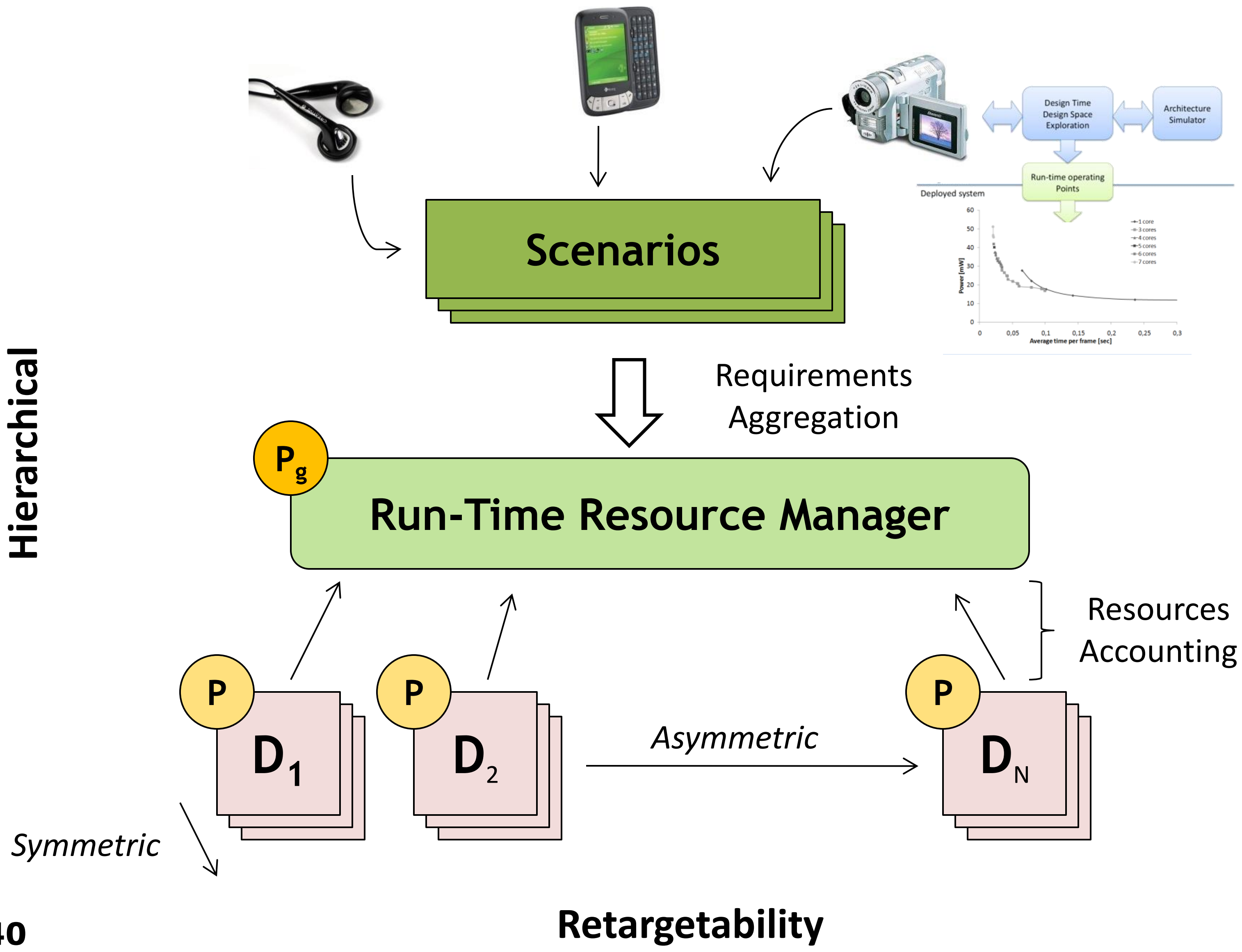
- **An example of usage of DSE tool for Run-Time Resouce Management**
  - An application is profiled off-line
    - Operating points are identified on a Pareto frontier
  - Two instances of the same application are run
    - Each application has its own set of requirements
  - The number of processors allocated for each application instance is determined at run-time
    - According to resources availability and constraints



From "AN INDUSTRIAL DESIGN SPACE EXPLORATION FRAMEWORK FOR SUPPORTING RUN-TIME RESOURCE MANAGEMENT ON MULTI-CORE SYSTEMS"

DSE integration with RTRM

Evolution of the MULTICUBE approach → 2PARMA



The End

- **Links to FP7 projects**
  - www.multicube.eu end June 2010
  - www.2parma.eu FP7 Strep
  - complex.offis.de FP7 IP
- **MULTICUBE Explorer download**
  - Follow links reported in www.multicube.eu
- **Info regarding MULTICUBE**
  - A book from Springer will appear in early 2011

THANKS FOR YOUR ATTENTION

William Fornaciari

fornacia@elet.polimi.it