Architecture and Design Methodology for Autonomic System on Chip

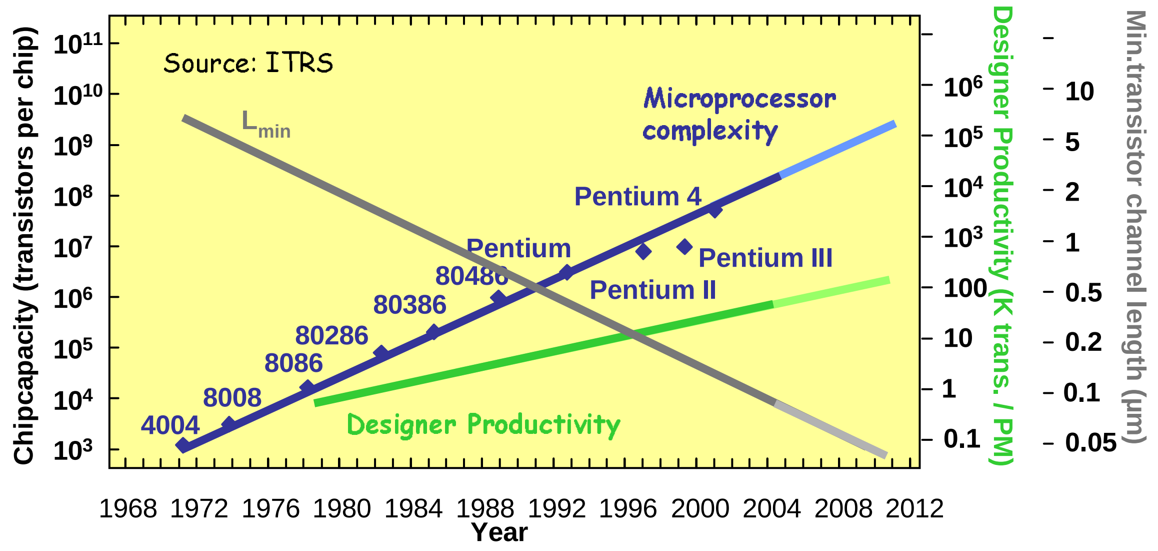
A. Herkersdorf, G. Lipsa,
O. Bringmann, W. Rosenstiel, W. Stechele

Technische Universität München ◆ Universität Tübingen
FZI Forschungszentrum Informatik

Outline

- Integrated Systems (R)Evolution
  - Moore's Law continues and enables SoCs of ever more increasing complexity, heterogeneity
  - Conservative worst-case design approach no longer feasible when reaching physical CMOS limits
- Autonomic SoC Design Concept – Conquer Complexity with Complexity
  - ASoC Architecture Framework
  - ASoC Design Methodology
- Autonomic SoC Research Challenges

Integrated Systems (R)Evolution



Proposition:

- New conceptual approach in IC design method necessary
- Incorporating autonomic principles into SoC architecture and design tools

Challenges originating from CMOS scaling:

- Coping with billion transistor design complexities at all abstraction levels
- "Productivity gap" increases engineering costs and development time cycles
- Increasing fabrication defects and sensitivity for stochastic events from reaching CMOS physical limits
- Increasing dynamic and static power dissipation densities

Autonomic Computing – IBM Manifesto

Server Trends:

- Inter-networked computer systems witness ever increasing complexity and diversity
  - Millions of lines of code
  - Heterogeneous OS and application SW stacks

#1 I/T Challenge:

- Dealing with the complexity of I/T systems originating as a consequence of technological progress

How Does Nature Deal with Complexity?

- Example: human organism and its autonomic nervous system

Conquer Complexity with Complexity…

- Sub-conscious self-control, -management, -organisation, -healing capabilities

and learn to work around imperfections

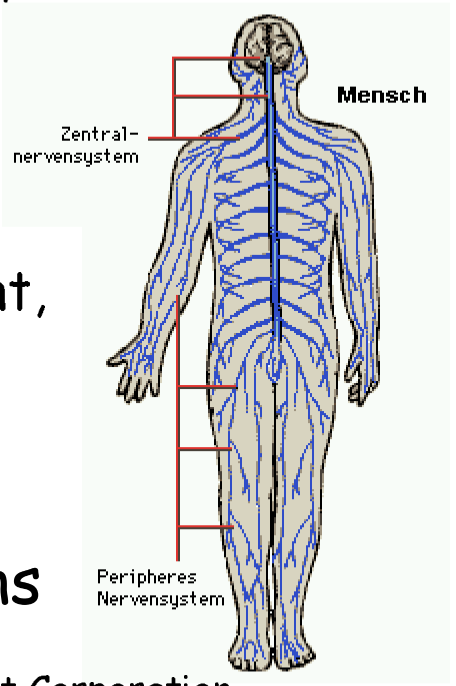
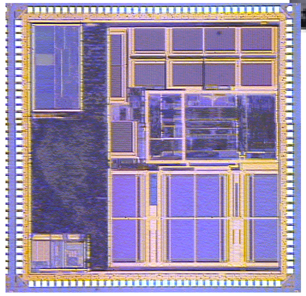


Figure source: Encarta Encyclopedia © Microsoft Corporation

Autonomic System on Chip (ASoC)

Systems are defined at different levels:

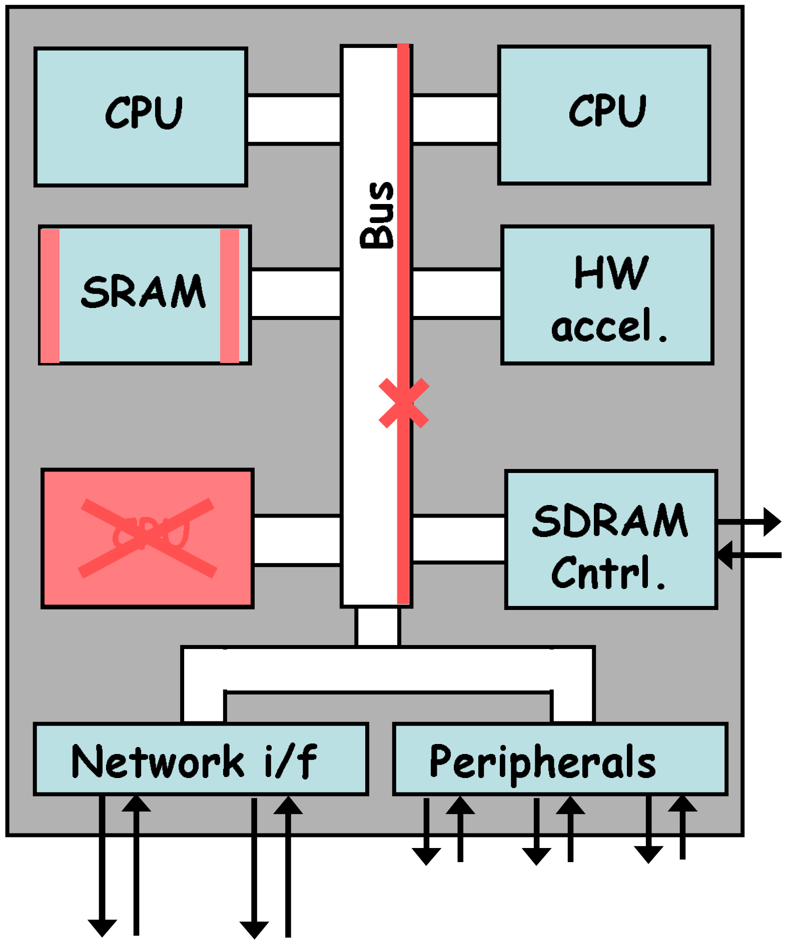
- Compute Servers, Internet Routers "Box level"
- Chip-level multi-processors, platform-based mixed-signal SoCs "Chip level"



Why to embed "autonomic" concepts into chip HW layer?

- Holistic approach:
  - SoC CMOS technology is basis of higher-order IT systems
  - Investigate and establish HW "hooks" for higher layer autonomic software
  - Nanosecond control loop cycle periods don't allow for SW-in-the-loop
  - HW defects may prevent SW diagnosis activation

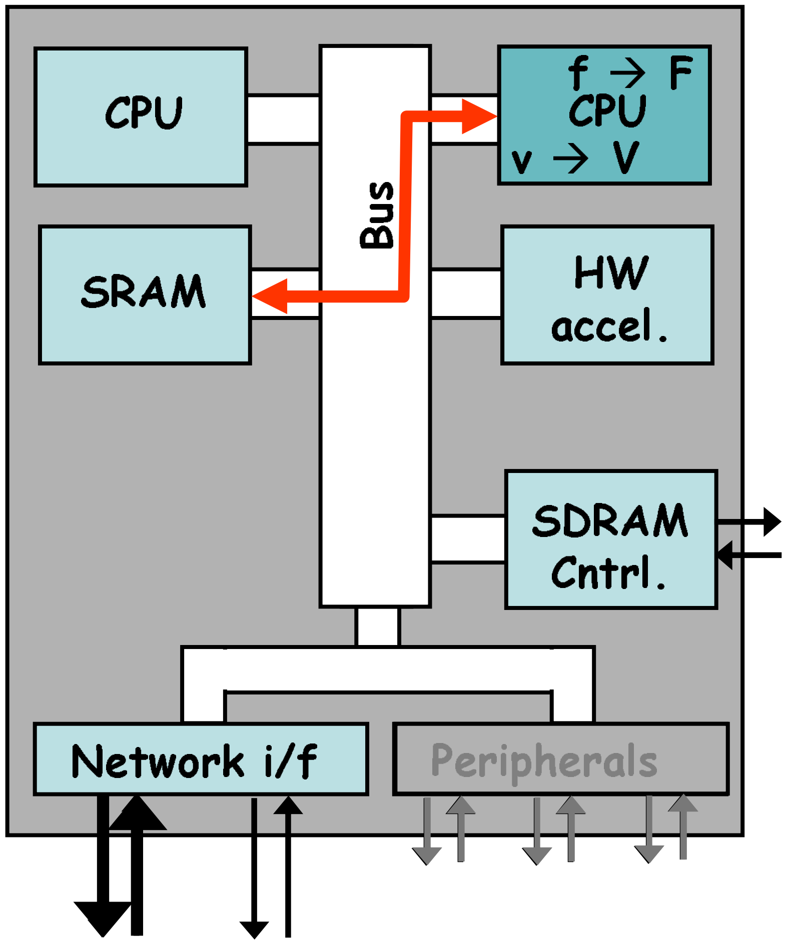
Autonomic SoC – Future Perspectives



Dealing with defects:

- Continuous, SoC component-specific diagnosis determines and disables faulty:
  - (sub-) building blocks,
  - interconnect resources,
  - memory regions
- Active/Stand-by selection among redundant IP functions
- Determines minimum feature set for current system operation

Autonomic SoC – Future Perspective



Performance/Power Optimization:

- When work load increases,
- raw bus capacity is increased,
- critical transactions among bus entities are prioritized,
- voltage and frequency of CPU are raised,
- secondary interfaces are temp. deactivated,
- redundant resources may be activated for load balancing,
- reconfigurable HW accelerators are activated on demand to facilitate CPU off-load

Existing Techniques Reusable for ASoC

- DVFS: dynamic voltage and frequency scaling
- BIST: built-in-self test
- On-Chip debugging aids: ChipScope, RISCWatch
- Reconfigurable computing platforms
- General area of fault tolerant systems

However,

- ... either not yet widely used, or mostly in isolation
- Have to be made syntactically compatible and semantically coupled
  - Standardization in industry forum required

Related Prior Art (… incomplete list)

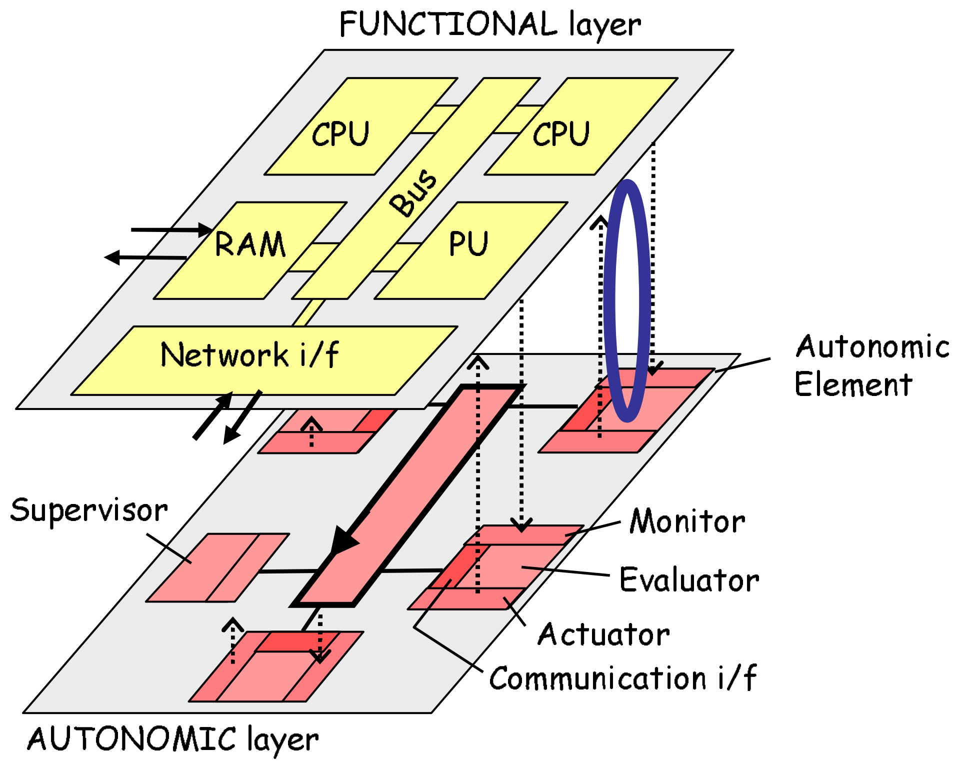
- Brinkschulte, Becker, Ungerer: "CARUSO – Low Power Autonomic Systems on Chip for Embedded RT-Applications"
  - Self-x functions at the SoC level through SW "helper threads"
- DeMicheli: "Designing Robust Systems with Uncertain Information"
  - Unsafe and faulty functions on the lowest process levels demand new design processes and tools
- Mitra, Huang, Saxena, et al.: "Reconfigurable Architectures for Autonomous Self-Repair"
  - Dual FPGAs with embedded soft CPUs facilitating self-healing system behavior

ASoC Architecture Framework

Evolutionary approach with compatibility to SoC design method

Rededicate part of SoC capacity to self-x surveillance and control functions in Autonomic SoC layer

- Autonomic Element (AE) library
- Closed HW control loop between functional PU and AE element
  - monitor, evaluate, memorize, (communicate), execute
- Autonomic SW functions have access to AE parameters



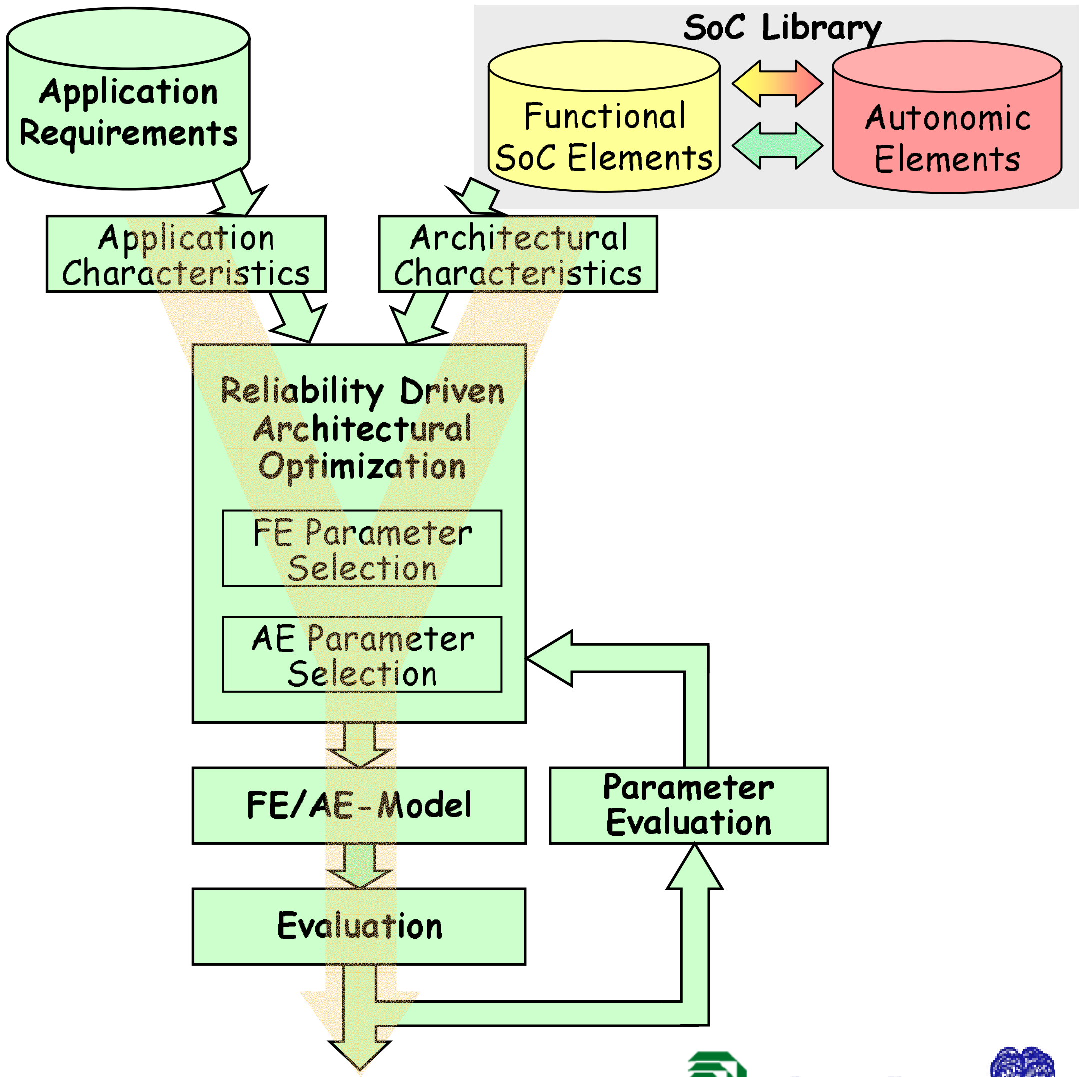
Maintaining state info on PU history in AE elements can be exploited for emergent system behaviour

ASoC Design Methodology

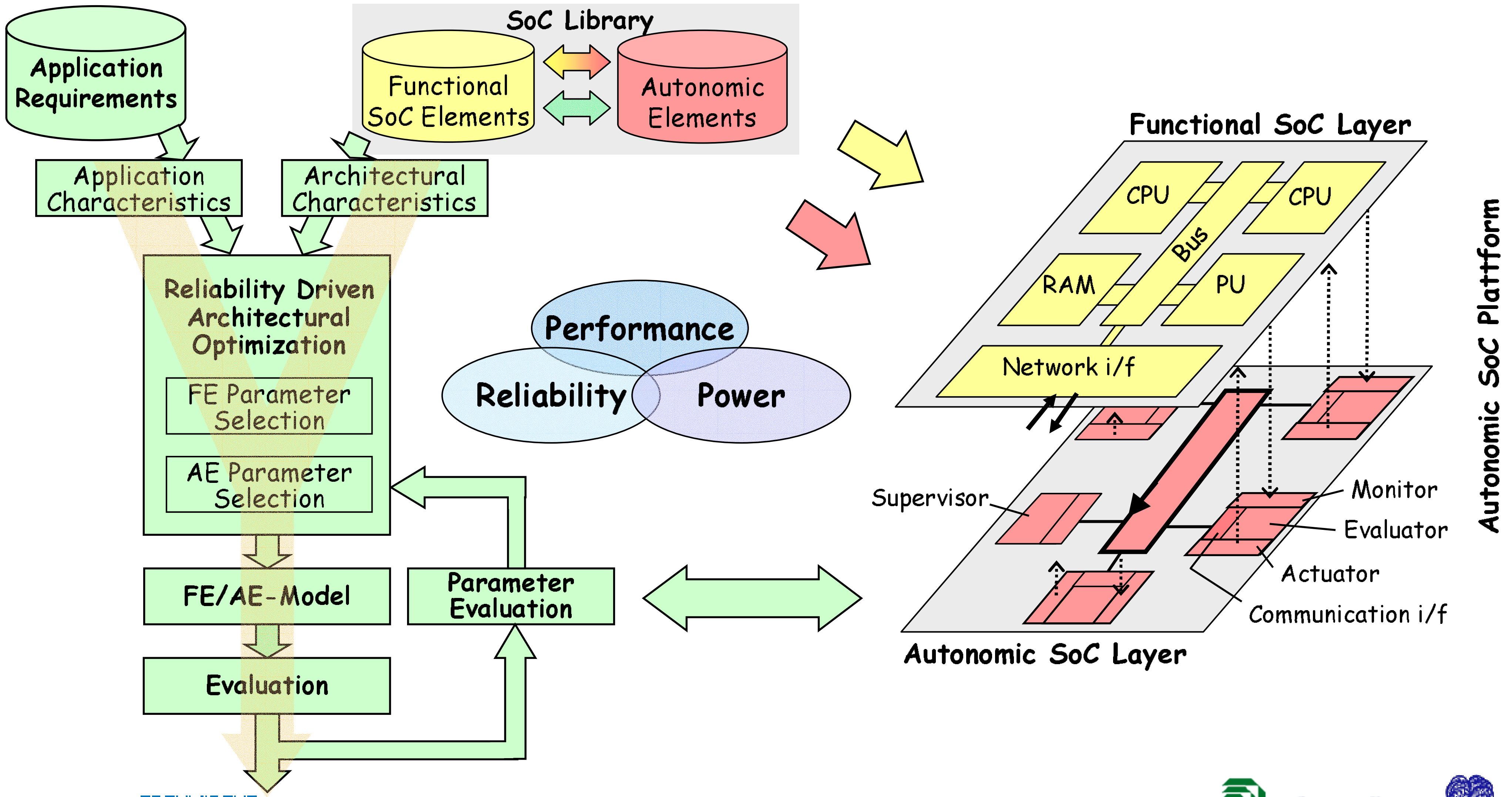
Design Flow for ASoC Architecture

ASoC design methodology to explore during design time

- self-organization capabilities and characteristics
- fault tolerance
- reliability (required MTTF)
- performance vs. power optimizations
- emergence properties



ASoC Architecture and Design Methodology



ASoC Architecture: Planned Activities

CPU:

- Extension of DIVA/RAZOR concepts
    - Functional and runtime errors
- Dynamic switch-over between parallel EXE pipelines
- Demonstrate with public domain soft core (e.g. LEON2)

Memories:

- On the fly interleaved rd/wr accesses (w/o parity, ECC) to monitor memory locations
- Detect / repair sporadic and persistent bit defects
- Exclude defective locations from future accesses by addr. translation

AE Interconnect:

- Shared Buffer-Insertion Ring
    - Native broadcast support
    - Simultaneous p2p connections
    - Simple comm. i/f

AE Control:

- Distributed: Syntactic and semantic coupling of different FE/AE events at proper rate
- Centralized: Autonomic Supervisor "watchdog" for AE elements

TECHNISCHE UNIVERSITÄT MÜNCHEN

FZI

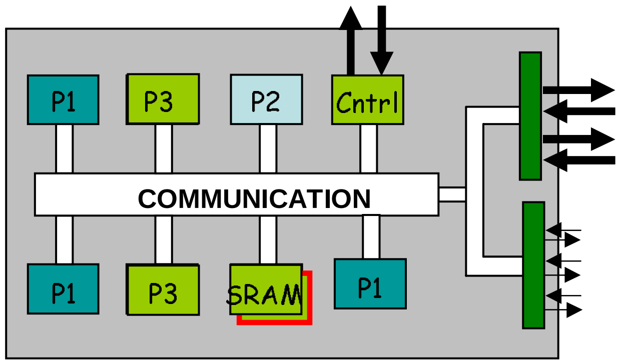
Eberhard Karls Universität Tübingen

ASoC Design Method: Planned Activities

Templates

- FE Templates
- AE Templates
- FE-AE Interdependence

Application Requirements

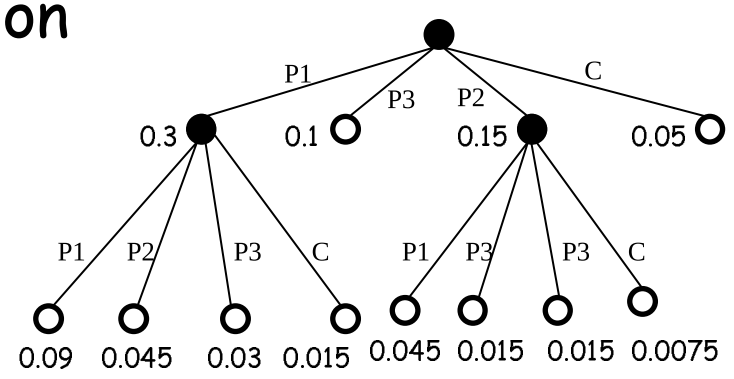


Reliability Driven Architecture Optimization

- Initial Configuration
- Refinement

FE/AE-Model Evaluation

- Dynamic fault tree – Decision tree
  - Exploration of the architecture
  - Changes in the states of the system
  - State evaluation



ASoC Research Challenges

- New Architectures, Design Methods and Tools
  - Dealing with **graceful degradation** and **redundancy** in distributed SoCs with **changing structure**
  - Interleaving and/or co-existence of functional and autonomic layers
  - Validation techniques for partially verified interdependent macros
  - Concepts for dynamic and coupled power/performance mgmt.

- Methods for flexible and dynamic HW/SW repartitioning
  - Defective HW is replaced on demand by equivalent SW process
  - Dynamically loaded HW configuration replaces lower-performing SW process

Conclusions

- Reliable design of complex SoCs is the key challenge for nanoelectronics when CMOS technology approaches its physical limits
- Application of autonomic or organic computing principles has been proposed to tackle this challenge
  - Requires conceptual change in SoC design process, architecture enhancements, and tools support

Thanks!