

# Design Methodology and Framework for Autonomic System-on Chip

Gabriel Lipsa

A. Herkersdorf W. Rosenstiel

O. Bringmann W. Stechele



Forschungszentrum  
Informatik  
SiM





# Motivation

- Coping with billion transistor design complexities at all level of abstraction
- Productivity gap
- Increasing fabrication defects from reaching physical limits
- Increasing dynamic and static power dissipation densities

Proposition:

- New concepts for Integrated Circuits design method
- Incorporating autonomic (organic) computing principles into SoC architecture and design tools.



# State-of-the-Art

- IBM Perspective on the State of Information Technology
- Principles of Self-Organization
  - Maturana & Varela – Theory of Autopoiesis (Self-Organization)
  - G. Jetschke – Mathematik der Selbstorganisation
- Caruso (Brinkschulte, Becker, Ungerer)
  - Autonomic self-x functions provided by the middleware layer of a multi-threaded CPU system
- Diva (Austin)
  - Dynamic verification
- Razor
  - Tunes supply voltage
  - Monitors bit error rate in the execution pipelines during operation
- Reconfigurable Hardware
  - Dynamic Voltage and Frequency Scaling
  - Built-in Self Test Concepts
  - Mechanisms and strategies to support fault tolerant behaviors of complex systems

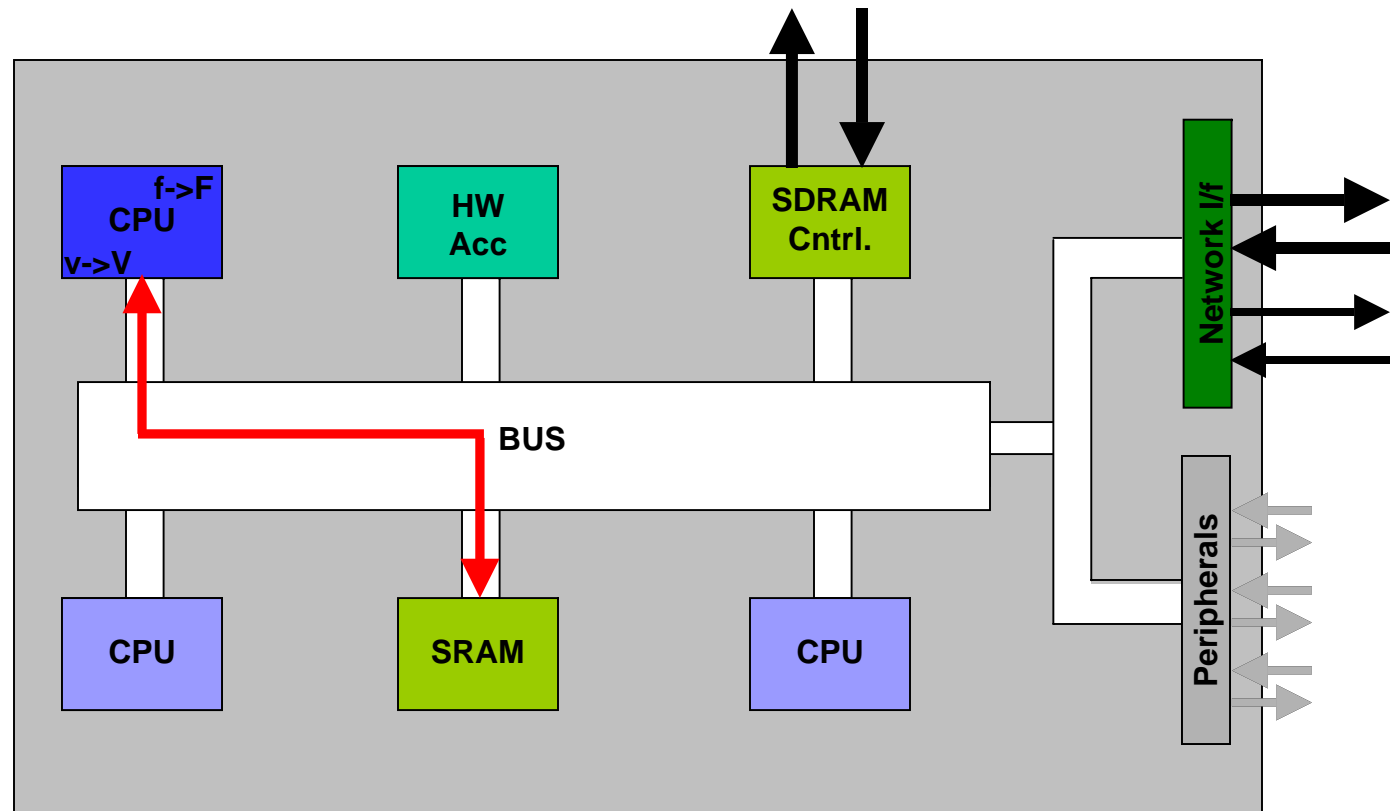
# ASoC Framework

## IBM Perspective on the State of Information Technology Autonomic System on Chip:

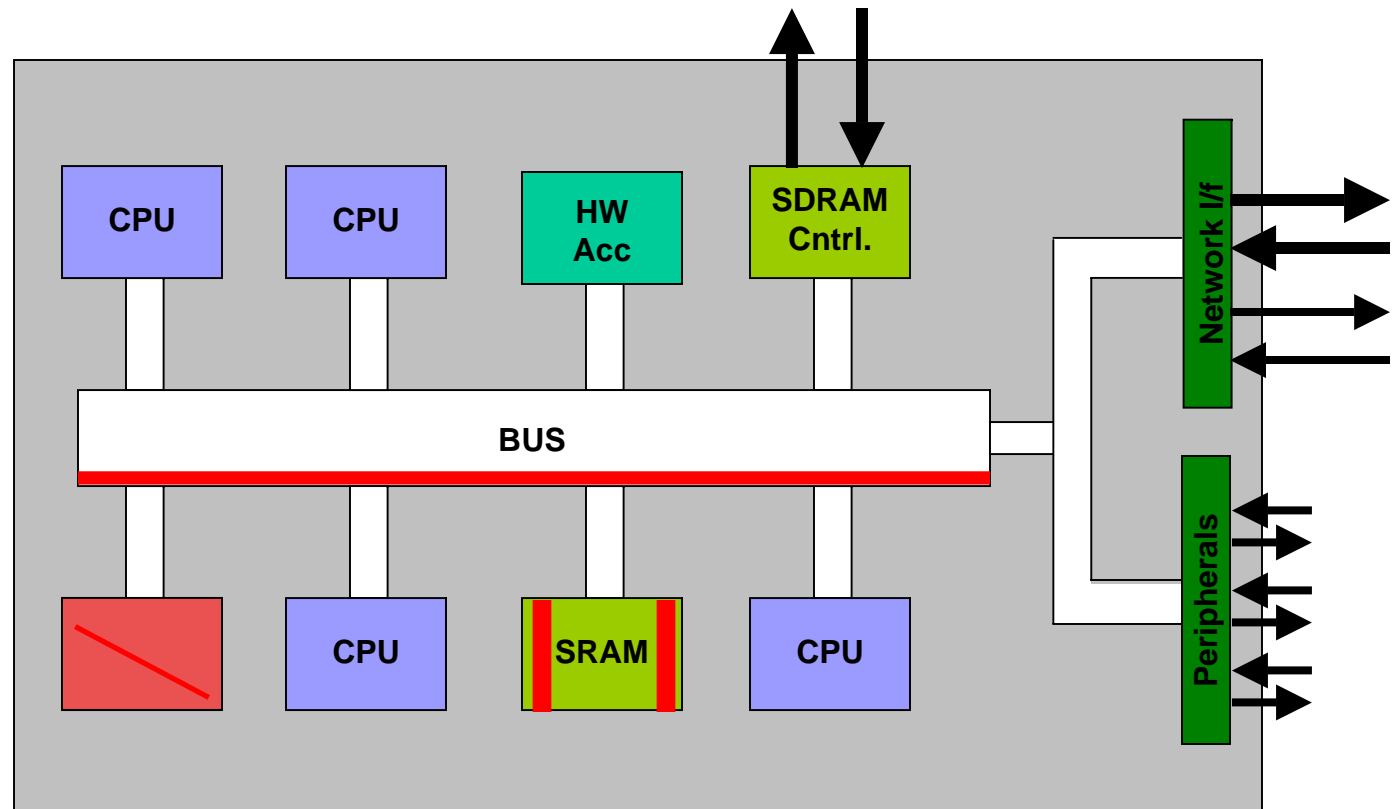
- must know itself
- must configure and reconfigure itself under varying and unpredictable conditions
- will always look for ways to optimize its working
- must perform self-healing
- knows its environment and the context surrounding its activity



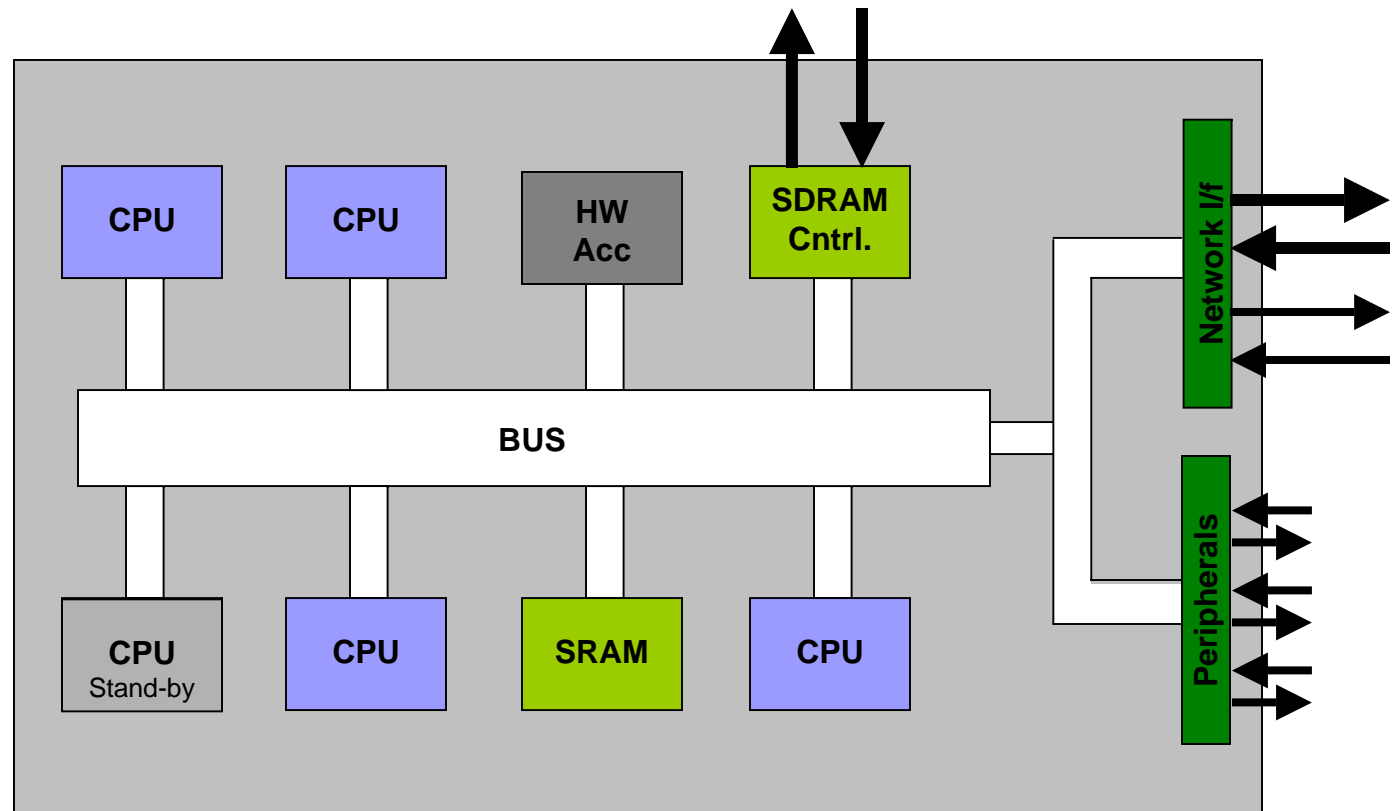
# ASoC Framework Overview



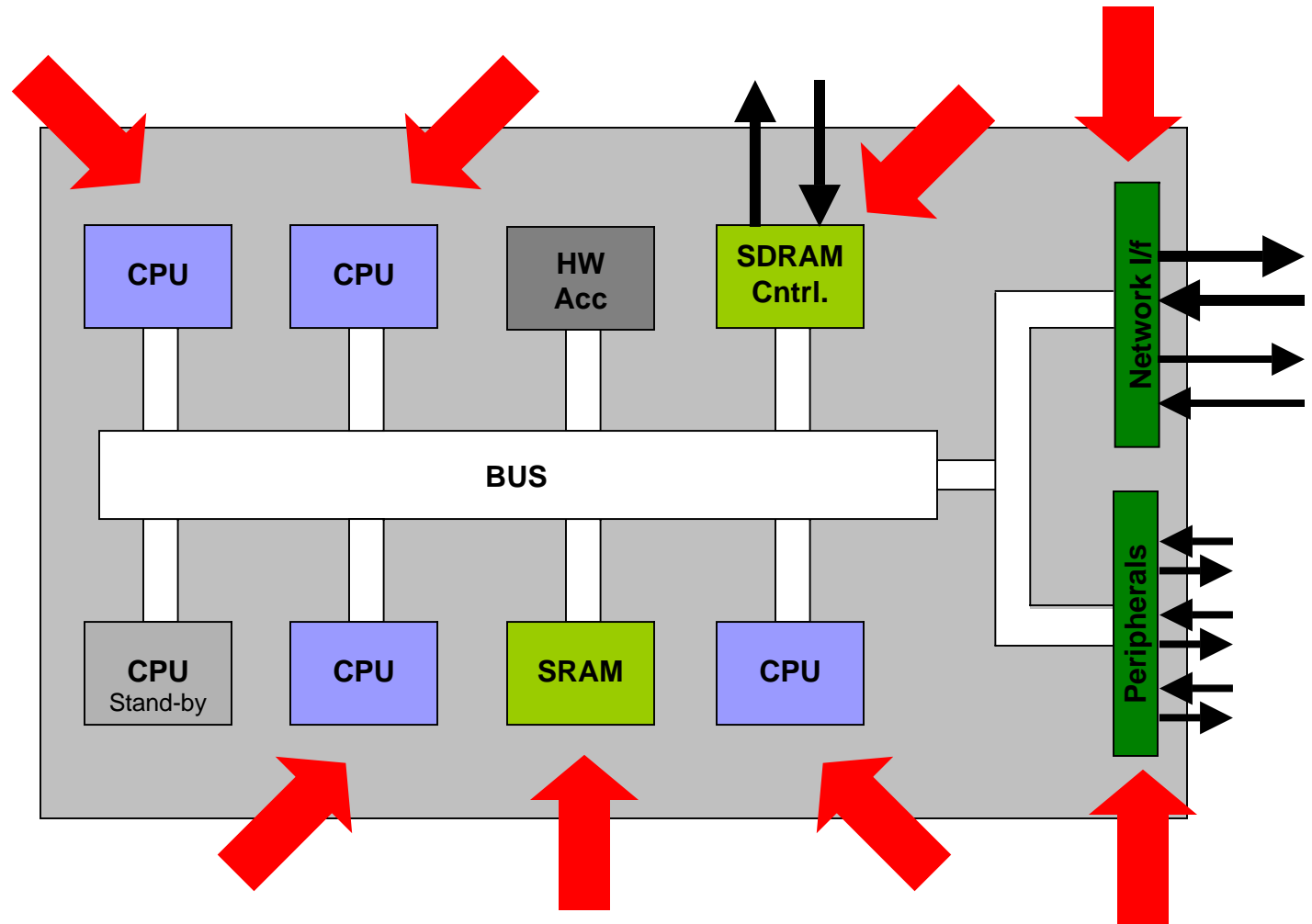
# ASoC Framework Overview



# ASoC Framework Overview

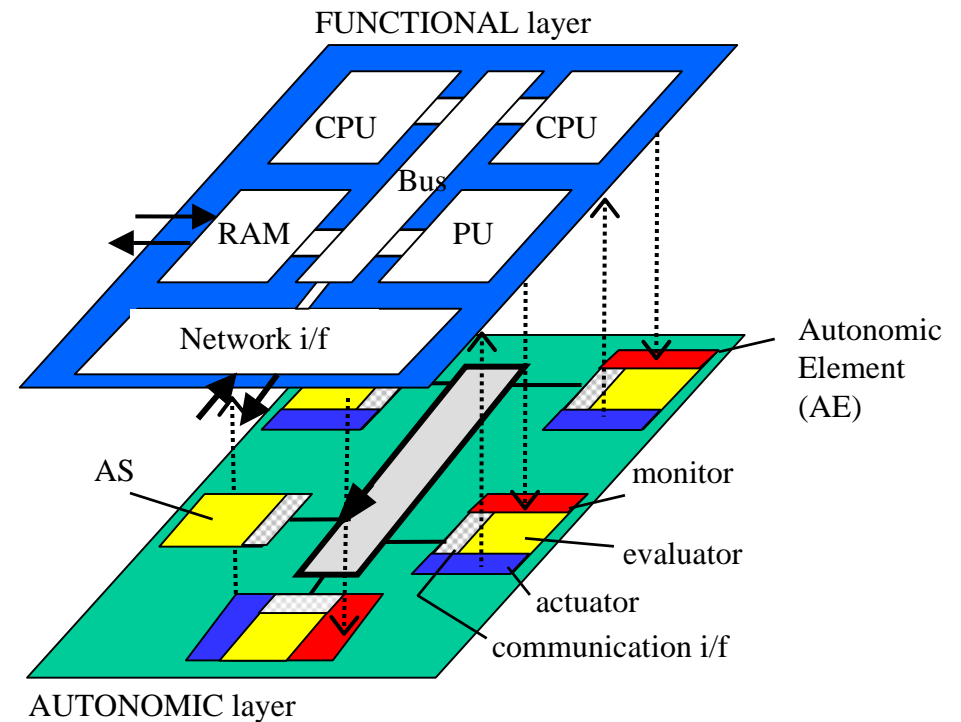


# ASoC Framework Overview



# ASoC Architecture

- Rededicate part of the SoC capacity to the self-x and control functions
  - Autonomic Control Elements (ACE) library
  - Closed HW loops between the ACE and their corresponding elements
  - Autonomic SW functions have access to the ACE parameters



## Autonomic Layer

- Autonomic Element
  - Monitor
    - senses signal or state information from the associated FE
  - Evaluator
    - merges and processes the locally obtained information with state from other AEs
  - Actuator
    - executes a possibly necessary action on the local FE
- Autonomic Supervisor
  - monitors the operations of and interaction between other AEs

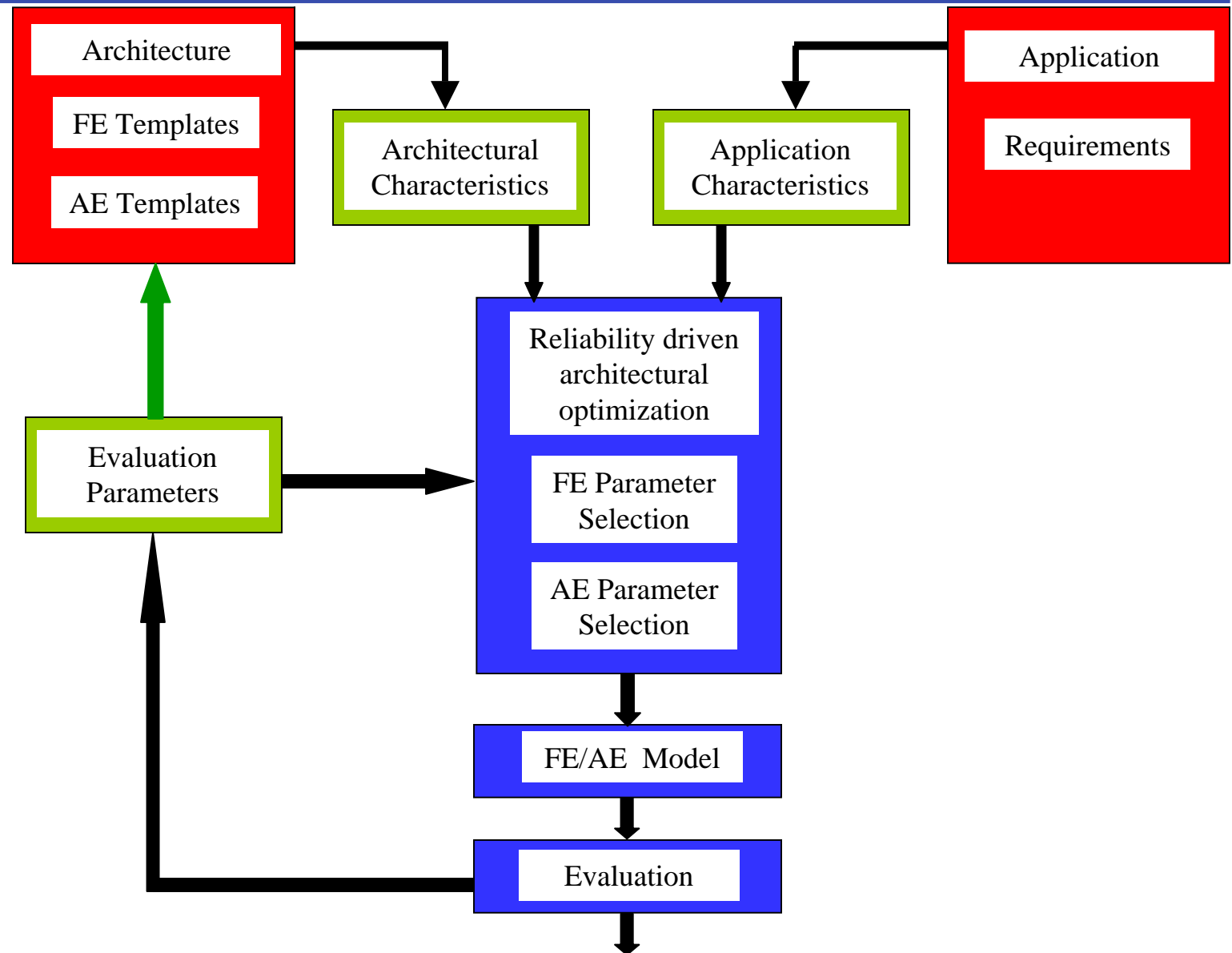
## Functional Layer

- Today's IP Blocks with little modifications for connecting with the Autonomic Layer

# ASoC Architecture

- Questions:
  - Is the clear separation between autonomic and functional layer really feasible and realistic?
  - Will there be specific AEs per functional IP component, or is it possible and more efficient to come up with a generic AE serving several functional IPs?
  - What is the right level of granularity for stand-by elements at the functional level?

# ASoC Design Methodology



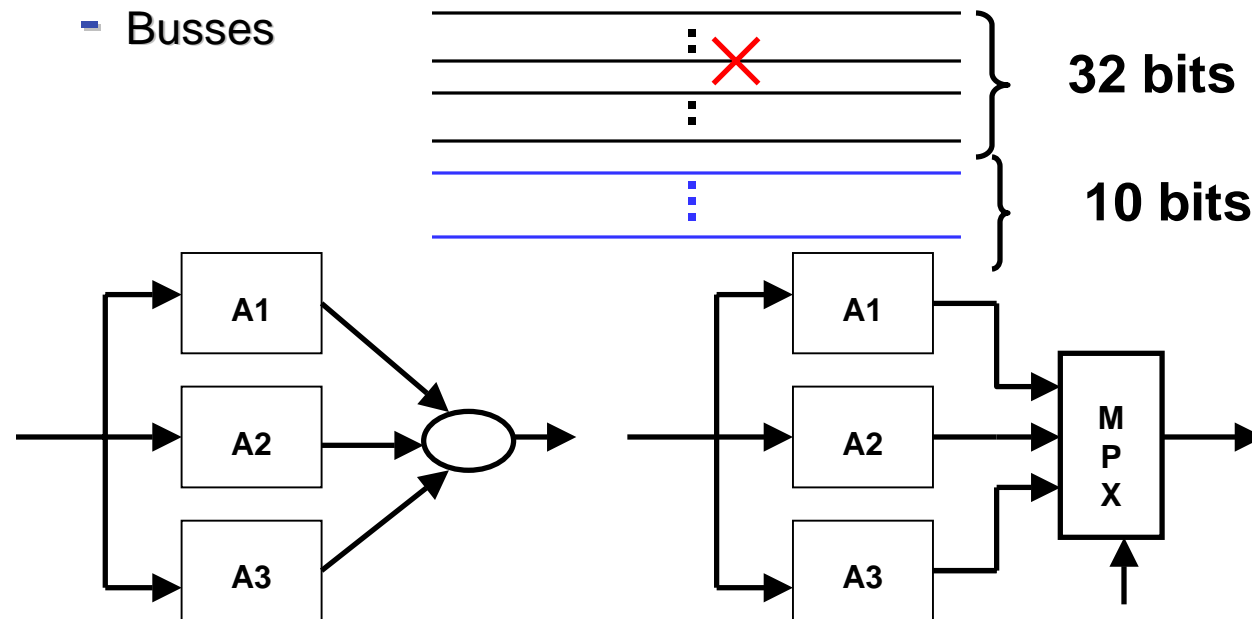
## FE Templates

- Clock frequency
- Area
- Distribution of failure rate (reliability)
- Parameters defining the functional element
  - Number and type of processing units
  - Type of memories and capacity
  - Width for busses
  - Level of granularity

## FE Templates

### Examples

- CPU
  - Replaced
  - Kept with limited functionality
- Memories
  - Local self-healing memories
  - Distributed self-healing memories
- Busses



## AE Templates

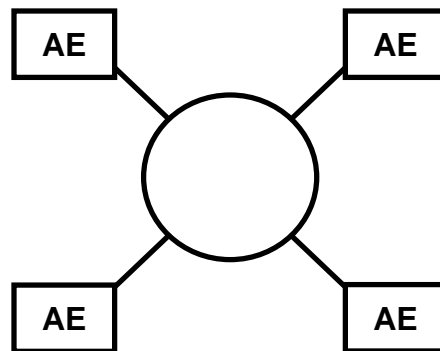
- Area
- Clock frequency
- Response time
- Self-organization characteristics
  - How good is the solution after a failure
  - After how much time will it find the best solution
- Autonomic Supervisor
- Autonomic Interconnect Structure (AIConS)



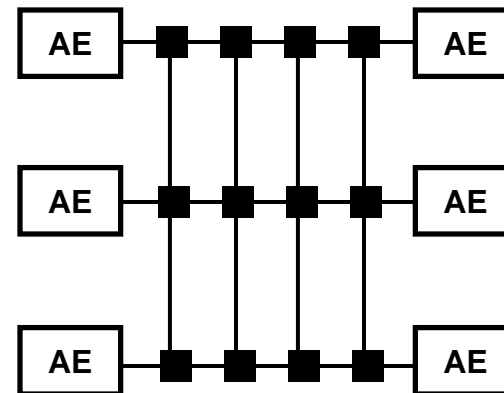
## AE Templates

- Autonomic Supervisor
  - Self-organizing algorithm
  - Emergent properties
- Autonomic Interconnect Structure

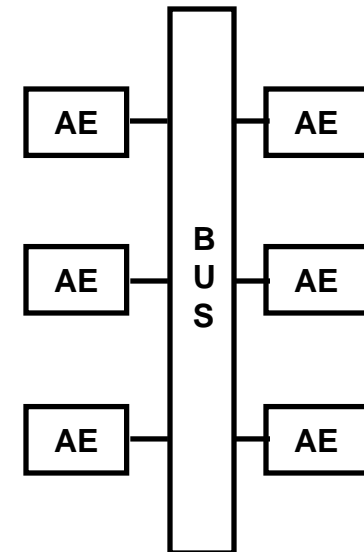
Shared Ring



Multi-port Switch



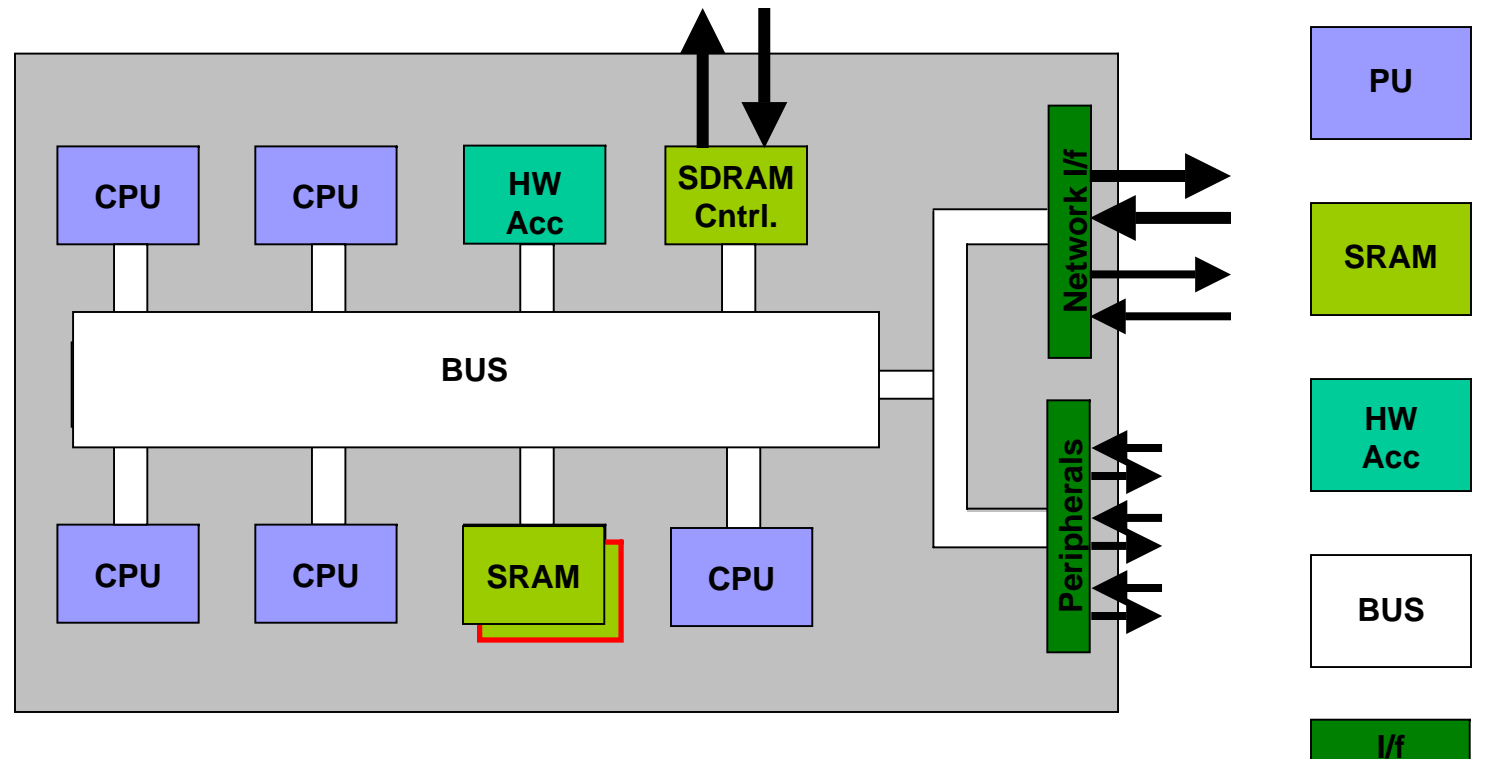
Shared Bus



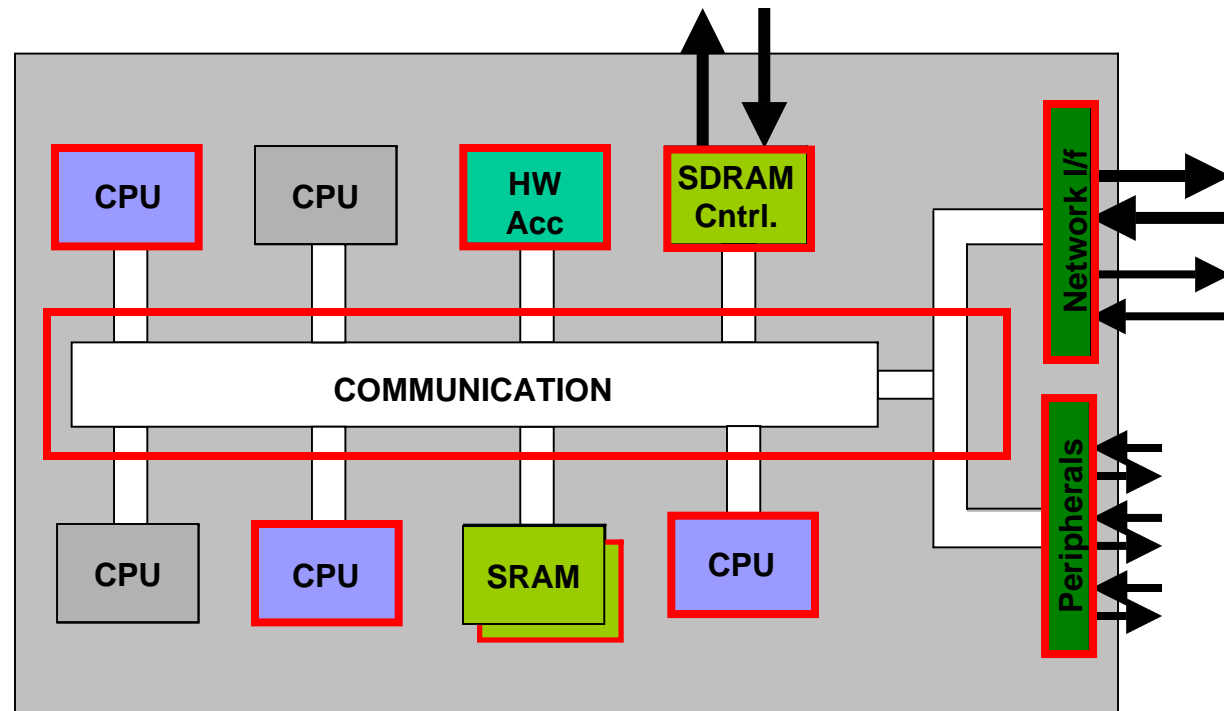
## Application Requirements

- Execution times
- Resource usage
- Priorities
- Task dependencies
- A-priori knowledge
- Time to function

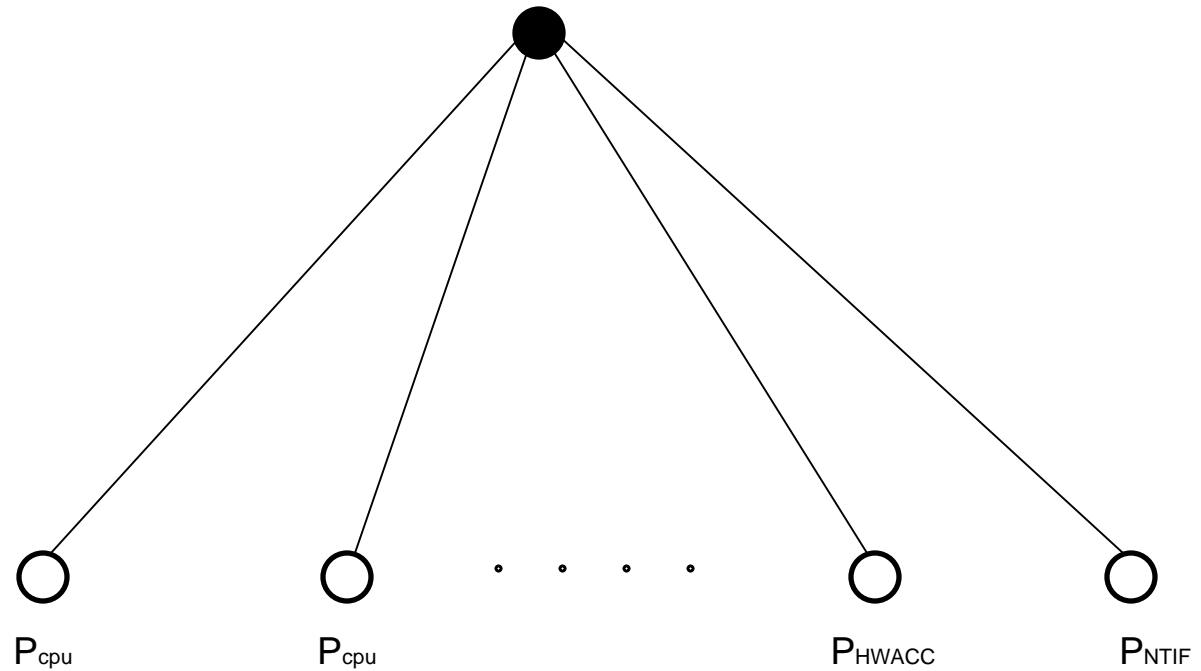
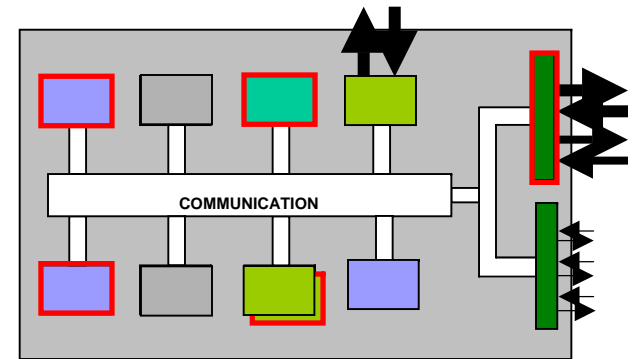
## Reliability driven architectural optimization



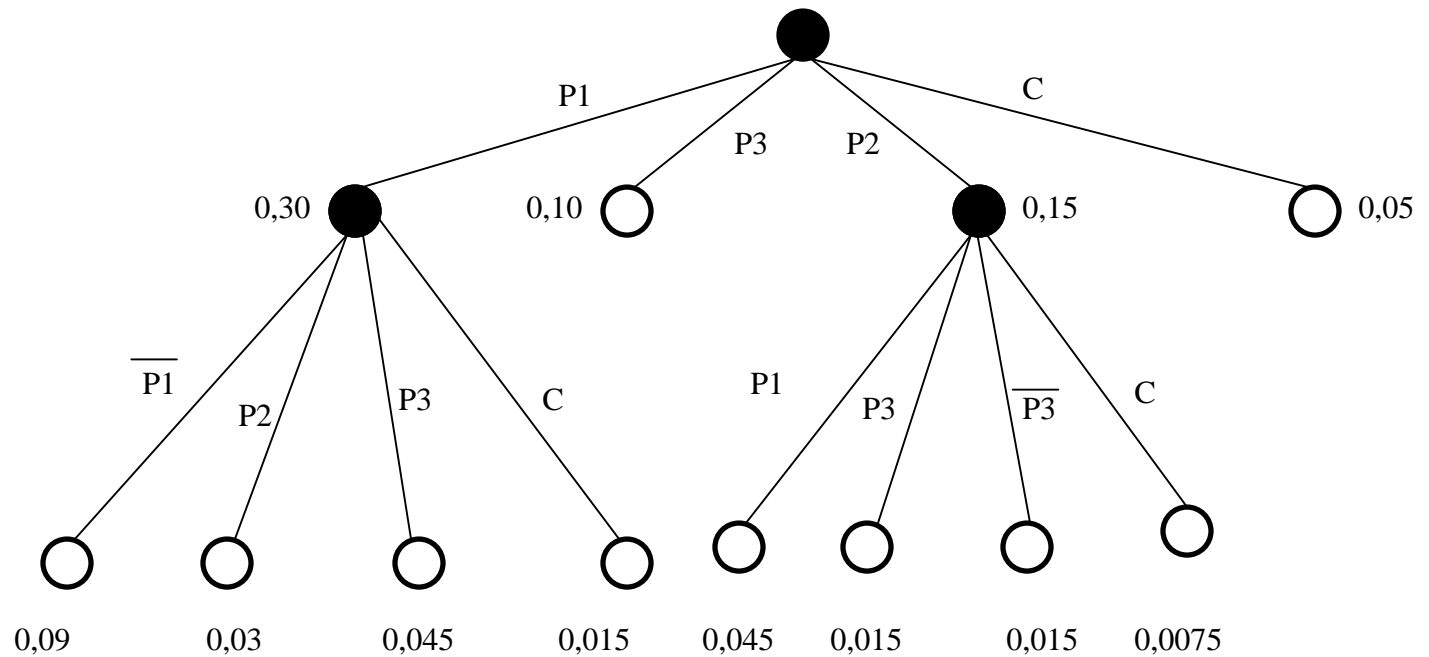
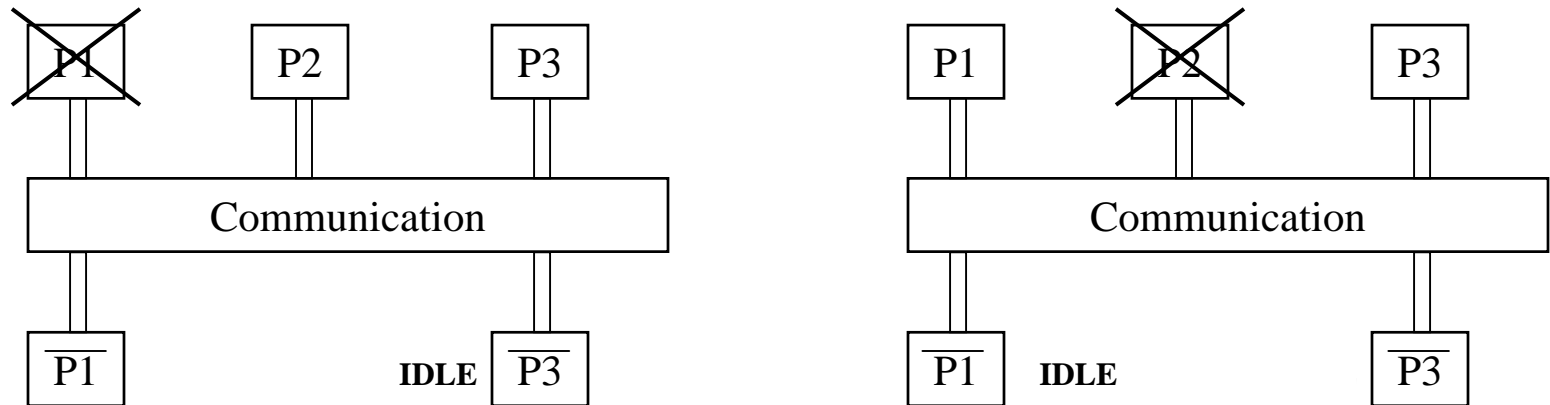
## Evaluation

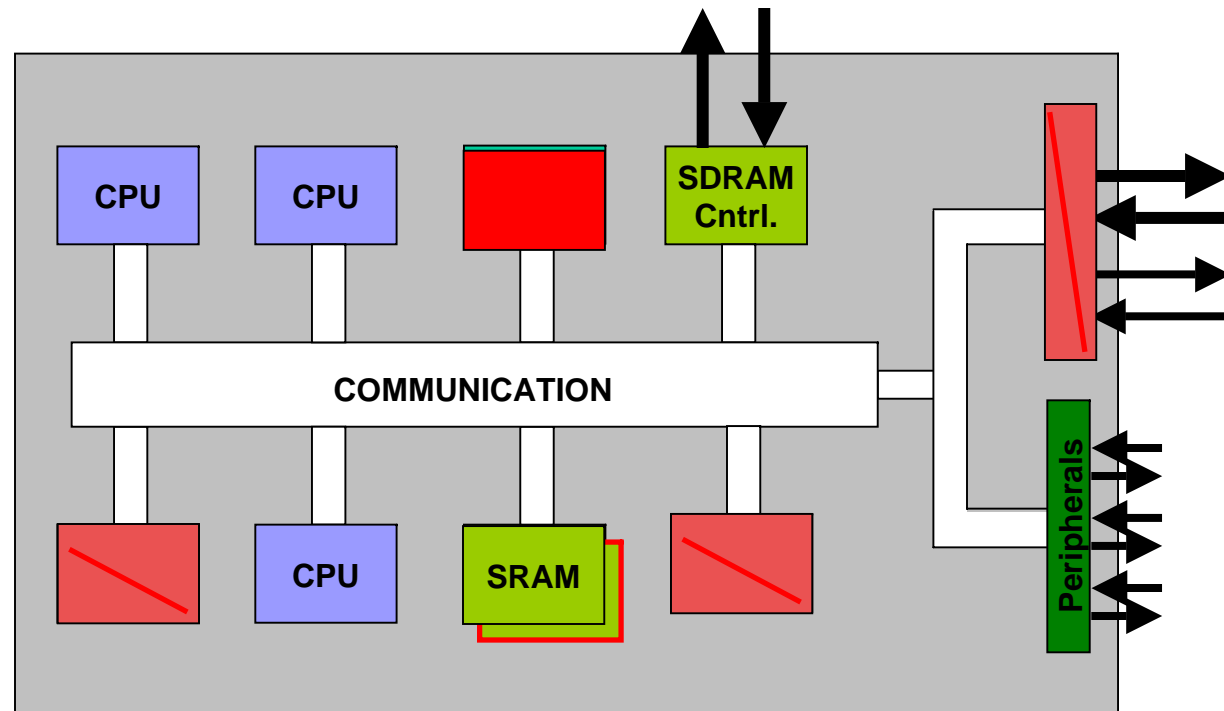


## Evaluation



## Evaluation







# ASoC Research Challenge

- 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 management
- 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