Architecture and Design Methodology for Autonomic System-on-Chip (ASoC)

A. Bernauer, W. Rosenstiel
Wilhelm-Schickard Institute for Informatics
Computer Engineering
University of Tübingen

O. Bringmann, B. Sander
Forschungszentrum Informatik
Karlsruhe

A. Bouajila, J. Zeppenfeld, W. Stechele, A. Herkersdorf
Institute for Integrated Systems
Munich University of Technology

asoc@informatik.uni-tuebingen.de

Using organic principles to solve hardware and design problems of future Systems-on-Chip (SoC)

Hardware problem

Nanometer sized transistors are **increasingly vulnerable** to ionizing radiation, manufacturing variations and environmental changes.

Future SoCs learn to work around or **live with permanent and temporary defects**.

Design problem

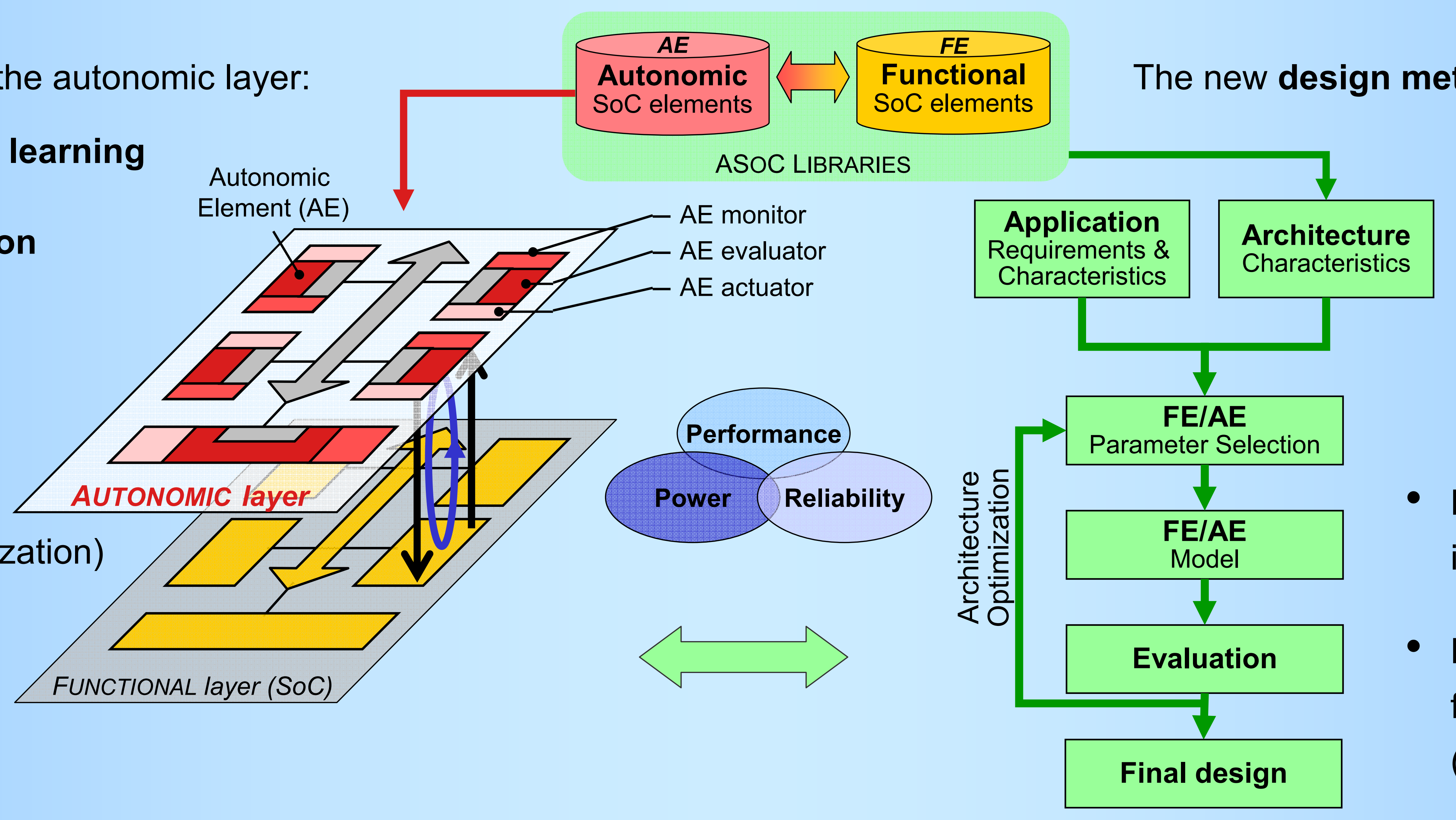
Design tools can hardly keep pace with **increasing complexity** of SoC as transistor counts and function complexity increase (design gap).

Future SoCs **find their optimal operating point on their own**.

Our solution:
Add an **autonomic layer** to the SoC

The new **architecture** with the autonomic layer:

- Integrates **flexibility** and **learning**
- Allows for **error correction (self-repair)**
- Allows for system-wide **run-time optimization** of performance, power and reliability (self-optimization)
- Creates **emergence** between the AEs



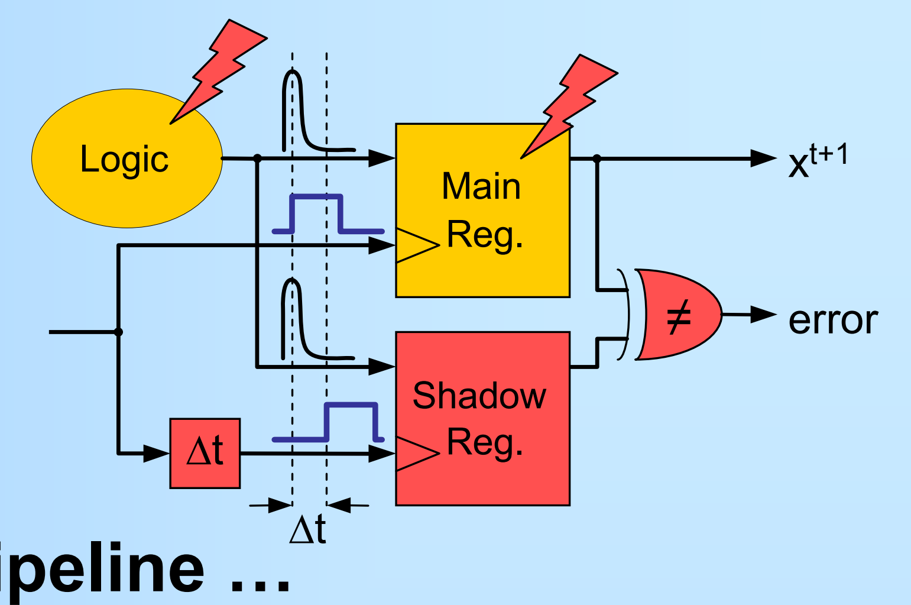
The new **design methodology** with new design tools:

- Is **aware of the hardware's new abilities** (autonomic layer, self-x properties)
- Treats **reliability as a first class optimization goal**
- Designs for the **typical case** that is usually correct
- Designs run-time **emergence** to find the optimal operating point (self-awareness)

Our project will show: A SoC based on the proposed two-layer architecture has the self-x properties necessary to handle the challenges of future SoCs while keeping the overheads acceptable. We will demonstrate this on a modern FPGA prototyping platform.

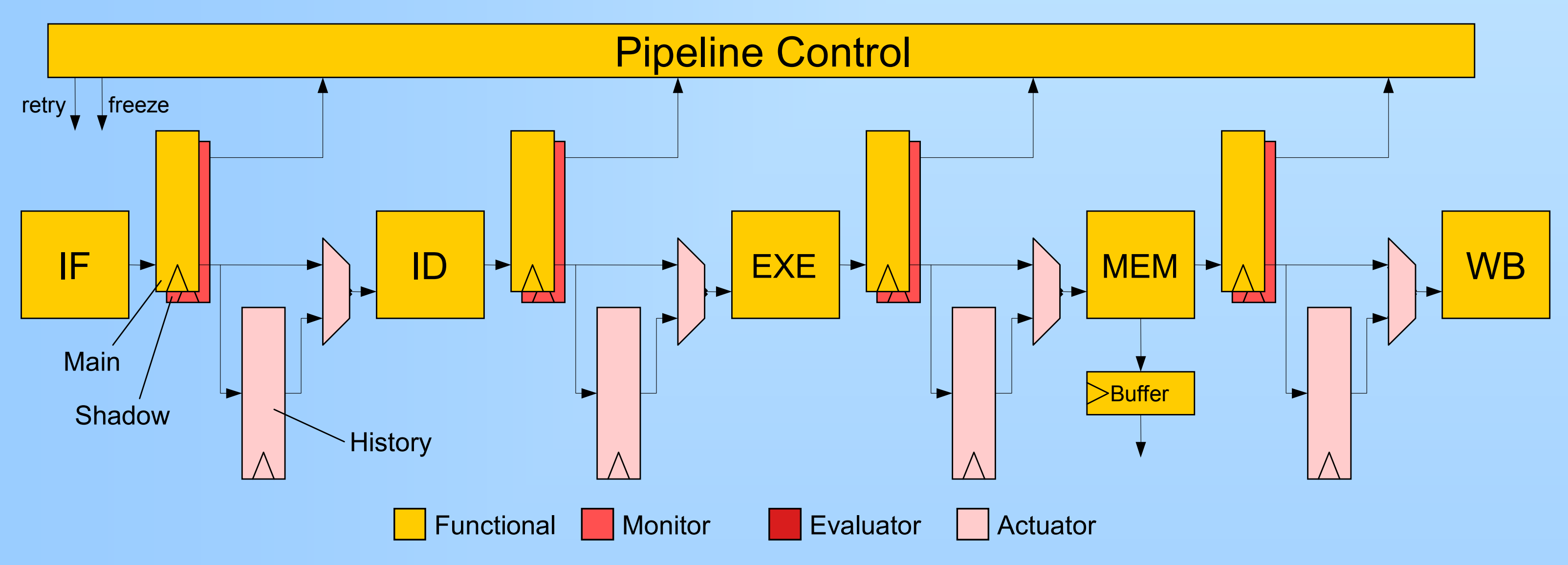
Fundamental Error Detection Technique…

- Shadow Register Based [Nicolaidis'99]
  - One-cycle detection latency
- Error Coverage
  - Transient Errors (SEU, SET)
  - Timing Violations

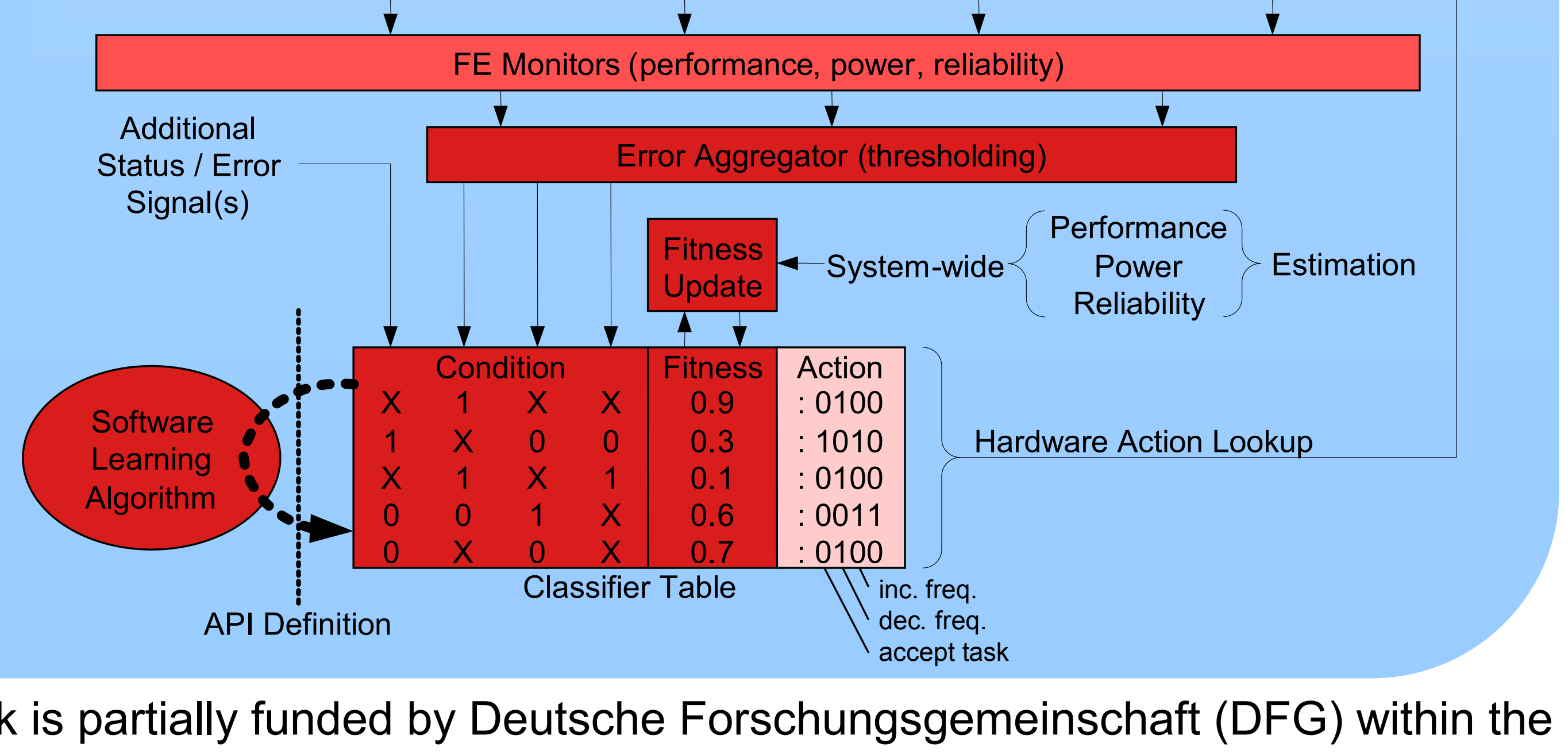


… Applied to a Leon2 RISC Pipeline …

- Enhancements [VLSISoC'06]
  - History registers for error recovery through micro-rollback
  - Constant two-cycle performance penalty on error
- Result: self-healing CPU pipeline (transient and timing errors)

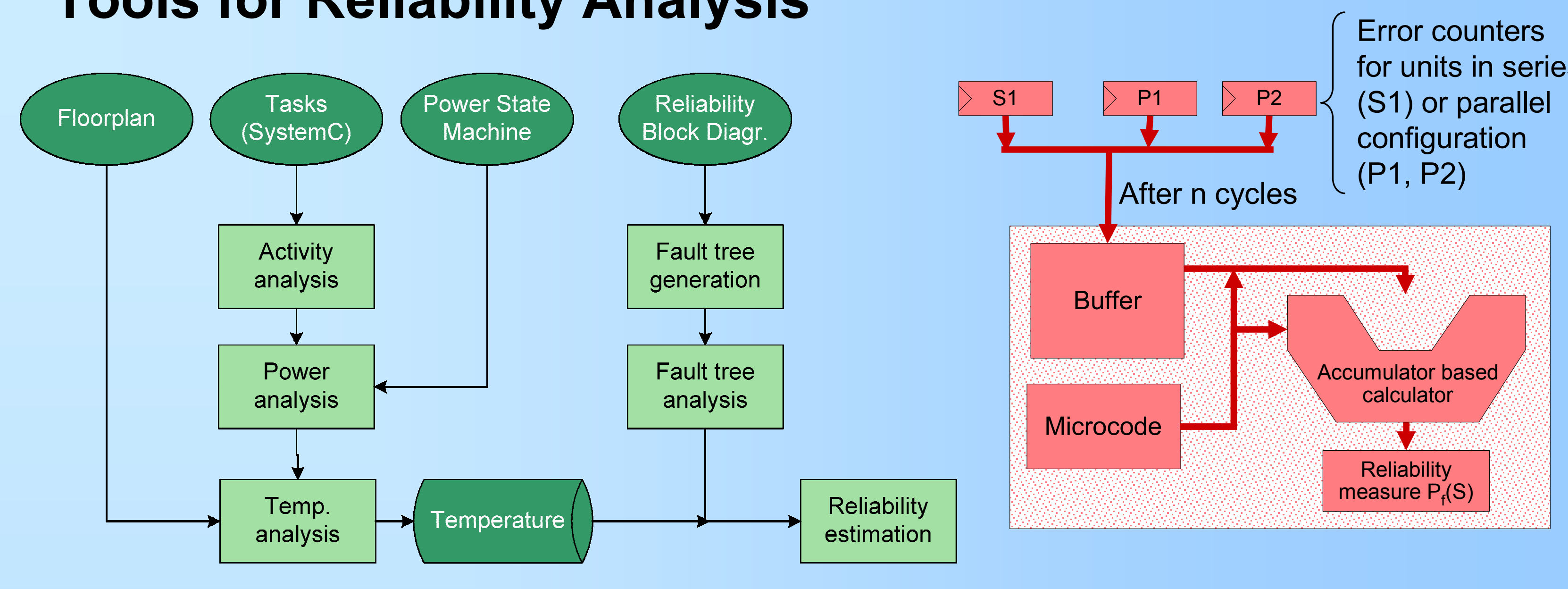


… and Integrated into ASoC Architecture with Learning Capabilities



Condition				Fitness	Action
X	1	X	X	0.9	: 0100
1	X	0	0	0.3	: 1010
X	1	X	1	0.1	: 0100
0	0	1	X	0.6	: 0011
0	X	0	X	0.7	: 0100

Tools for Reliability Analysis



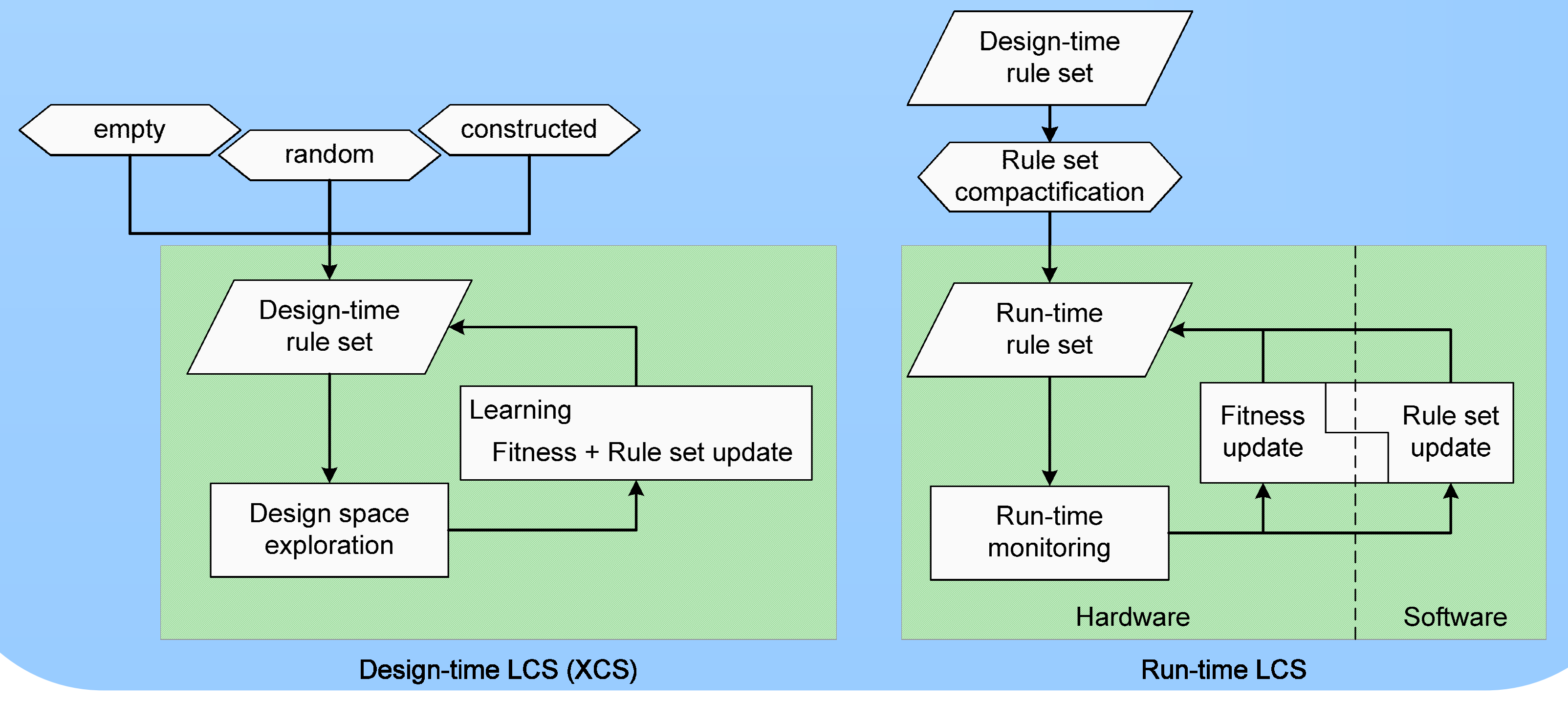
At design time

At run time [Bernauer06]

- At design time: optimize reliability application-dependently
- At run time: trigger actions if reliability drops below a threshold (e.g., reduce clock frequency or switch to a redundant unit)

Design of Learning System

- Based on **learning classifier systems** (LCS)
- LCS map conditions to actions (rules)
- At design time: learn rule set with **XCS** (high learning rate)
- At run time: adapt rule set with **simple LCS** (low area overhead)
- Support run-time adaptation with **software learning algorithm**
- Interplay of different LCS creates **emergent behavior**



This work is partially funded by Deutsche Forschungsgemeinschaft (DFG) within the priority program "Organic Computing" (SPP 1183) under the grants HE 4584/3-1 and RO 1030/14-1.

[Nicolaidis'99] M. Nicolaidis, "Time Redundancy Based Soft-Error Tolerance to Rescue Nanometer Technologies", 17th IEEE VLSI Test Symposium, 1999
[VLSISoC'06] A. Bouajila, J. Zeppenfeld et al., "Organic Computing at the System on Chip Level", VLSI-SOC, 2006
[Bernauer06] A. Bernauer, O. Bringmann et al. "An Architecture for Runtime Evaluation of SoC Reliability." GI-Edition - Lecture Notes in Informatics, vol. P-93, p.177-185, Bonn, September 2006, Köllen Verlag