


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

J. Zeppenfeld, A. Bernauer, S. Eisenhardt, A. Bouajila,
O. Bringmann, W. Stechele, W. Rosenstiel, A. Herkersdorf

Technische Universität München

Universität Tübingen

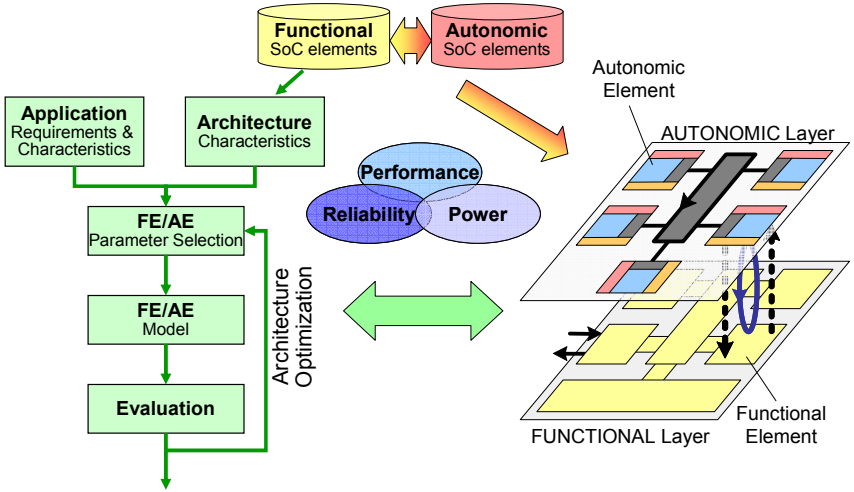
FZI - Forschungszentrum Informatik

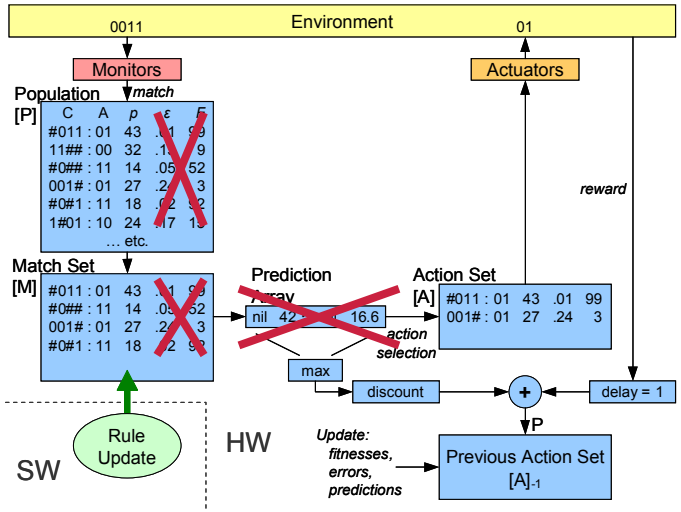
ASoC Highlights

- Concepts
  - Applying machine-learning techniques during SoC design flow and on chip with moderate resource overheads
  - Extension of functional components by autonomic elements
  - Adaptation of XCS to LCT
- Evaluation
  - Rule generation, translation and selection strategies
  - Effect and benefits of relative vs. absolute rules
  - Extended theoretical work of XCS for complex problems
  - Distributed learning on multi-core chips
- Prototype
  - Multi-core Leon3-based hardware demonstrator
  - Graphical interface for interactive evaluation of results
  - AE interconnect for sharing of global data between AEs

ASoC Highlights: Concepts

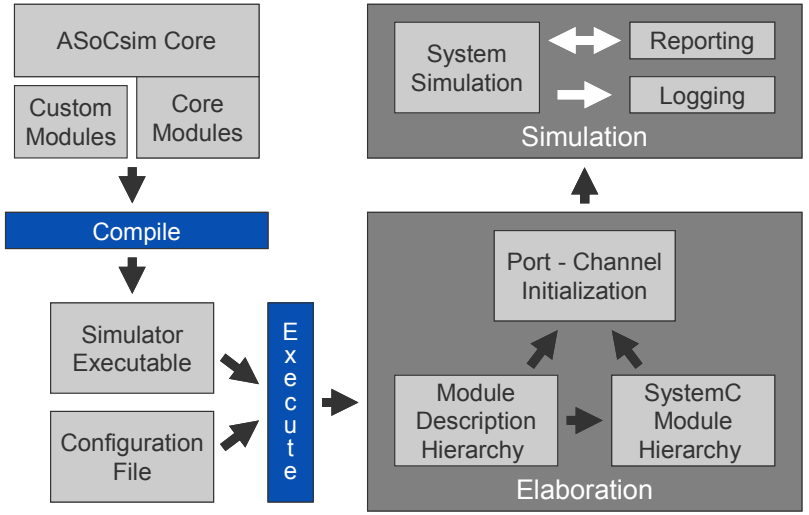
- Employ bio-inspired principles for autonomic performance, power and reliability optimization
  - Functional layer containing typical SoC components
  - Autonomic layer that provides various self-x properties
- Develop design flow that incorporates autonomic capabilities
- Adapt XCS to allow for fast, hardware-optimized learning at run time

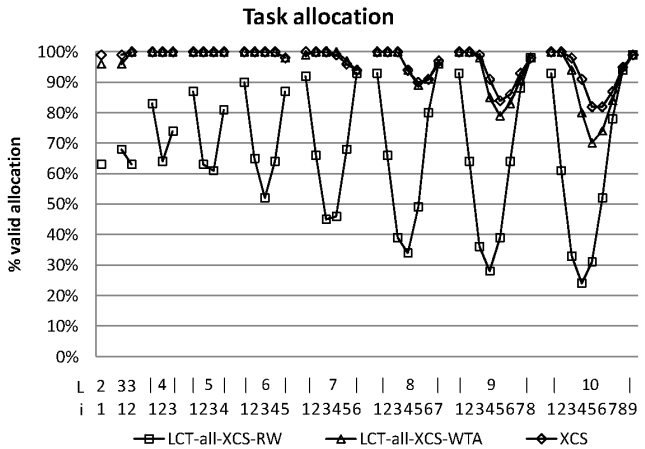




ASoC Highlights: Evaluation

- Development of SystemC-based simulator for high-level evaluation of autonomic concepts
- Exploration of:
  - Various system configurations
  - Various rule selection strategies
  - Relative vs. absolute rules
- Generation of an initial rule set
  - Create an initial rule set during design time using full XCS
  - Translate the initial rule set for use in the run-time system
  - Continue learning at run-time to react to unforeseen events
- Distributed learning in multi-core systems

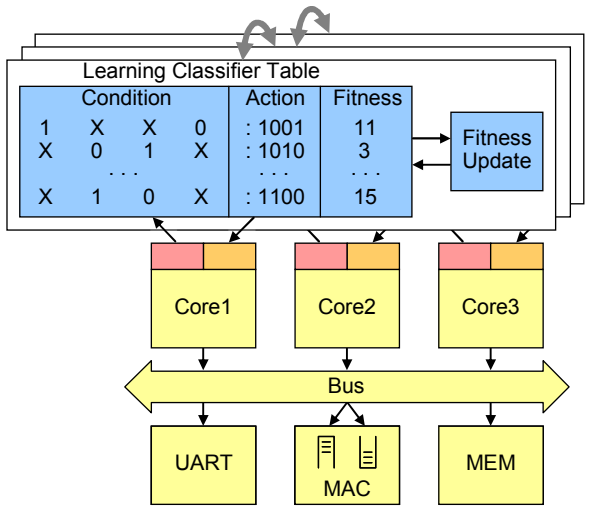


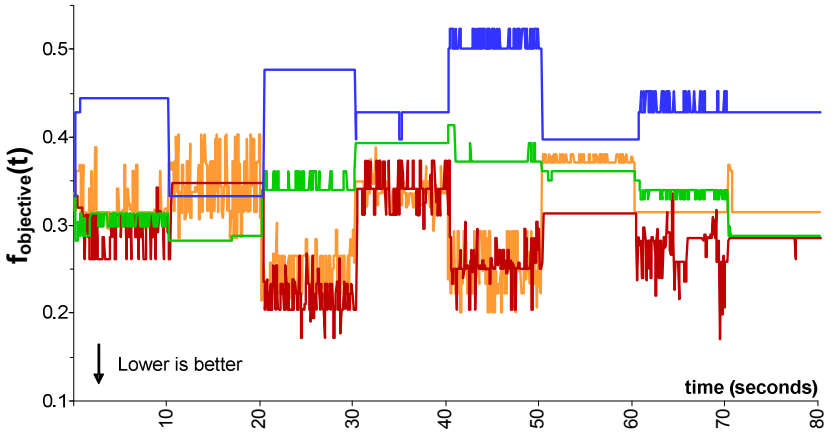


[BICC2010]

ASoC Highlights: Prototype

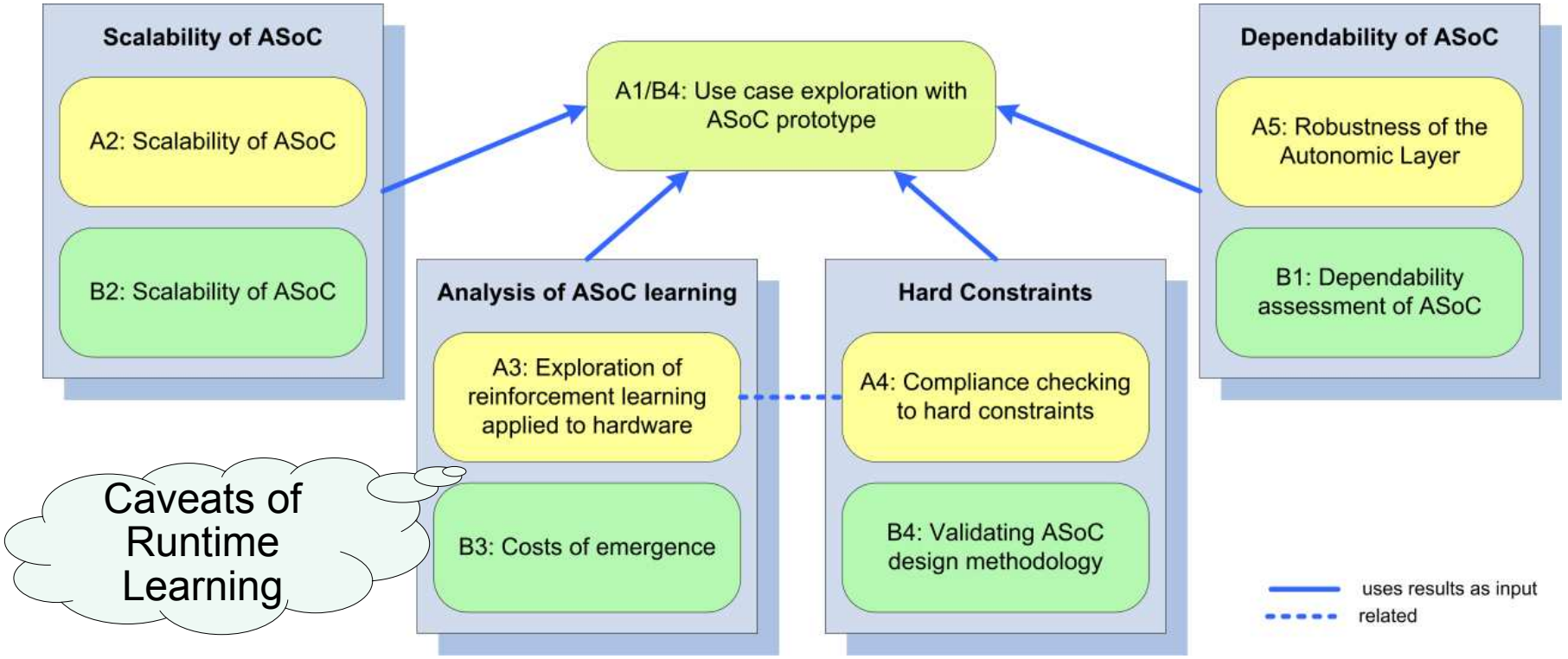
- Leon3-based multi-core FPGA prototype for the evaluation of concepts on a physical system
- Verification of high-level simulation results on actual hardware
- Exploration of:
  - Autonomic adjustment of frequency and task distribution under changing workloads
  - Global vs. local optimization
  - Various aspects of run-time learning





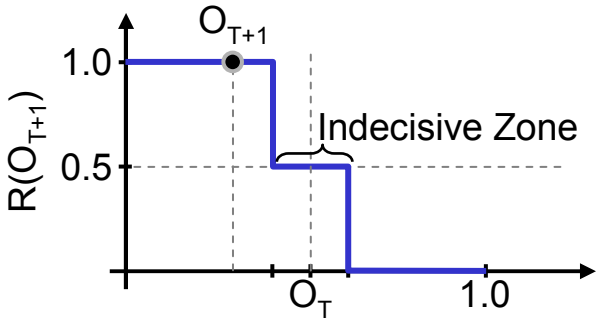
[ICAC2011a]

Phase 3 Work Packages



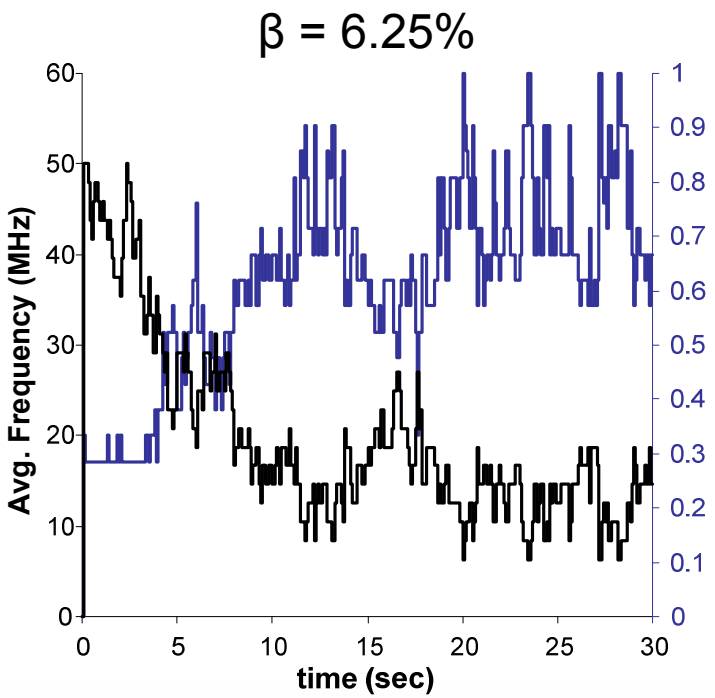
Combating Noise

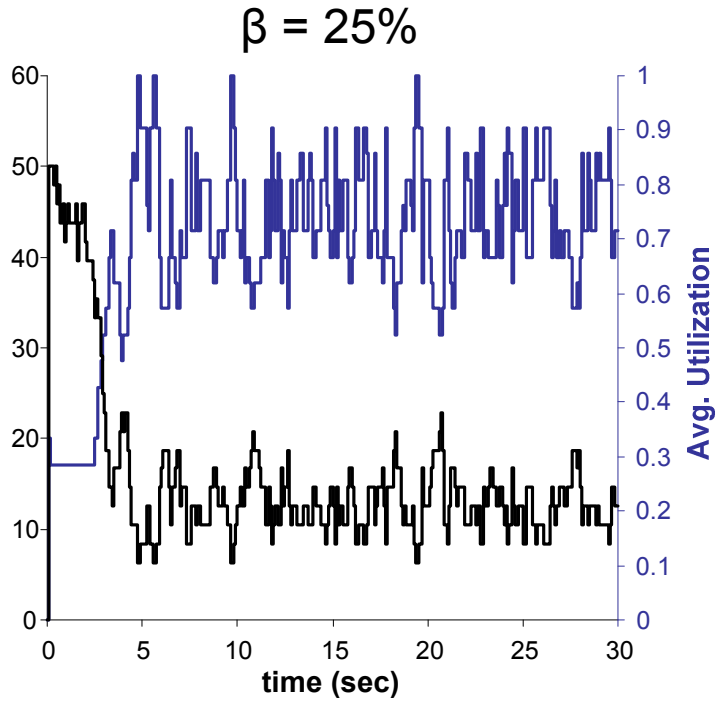
- Caveats of digital systems
  - Monitor precision is discrete, resulting in fluctuations
  - Perfect parameterization is physically not possible
- System dynamics are a challenge
  - Doing the same thing in the same (measured) situation does not always yield the same (measured) result
  - Even more complex when multiple components interact
  - Precision of objective function (and reward / fitness) suffers
- Use of a tolerant reward function
  - Stepwise to ensure strong differentiation between good and bad rules
  - Possibility to give no reward when indecisive



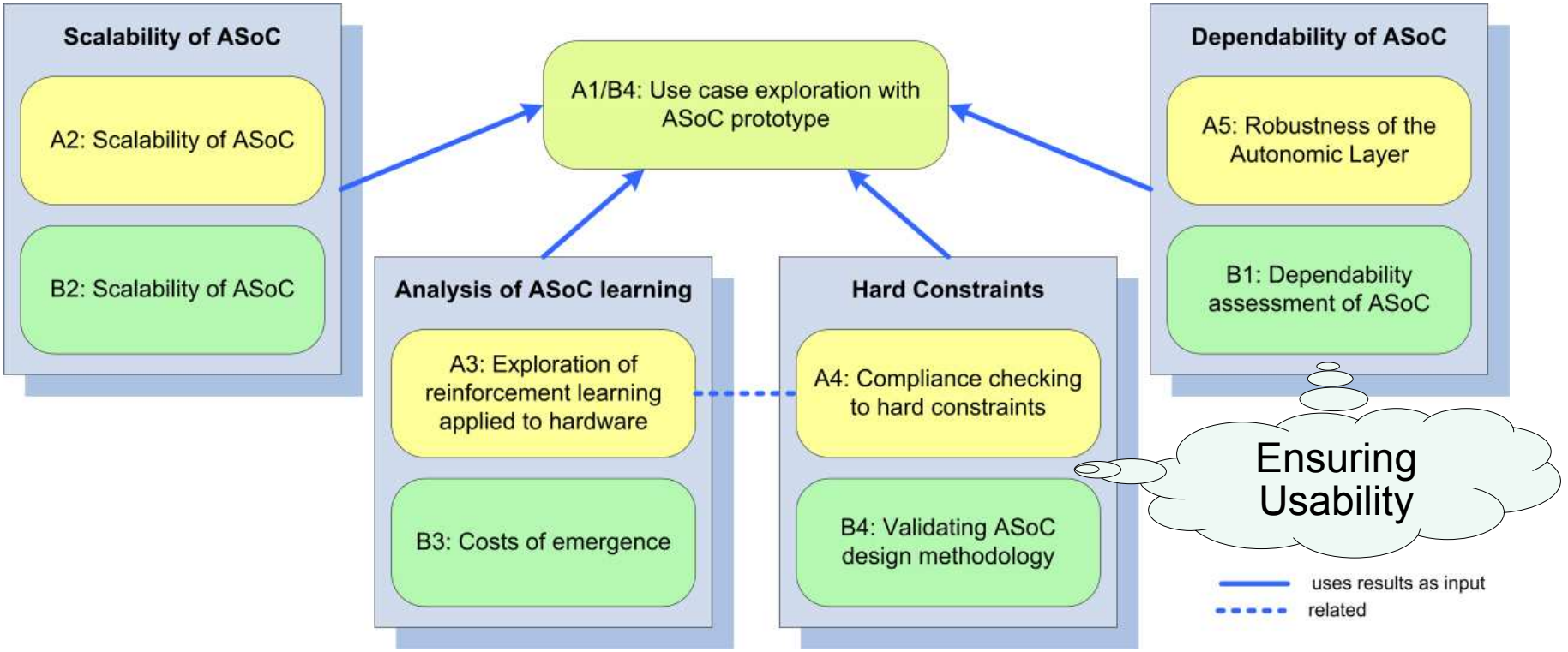
Learning Rate – β

- Indicates influence of reward on fitness
  - $Fitness_T = \beta \cdot Reward_T + (1-\beta) \cdot Fitness_{T-1}$
    - β ~ 0: better averaging of noise
    - β ~ 1: fast learning





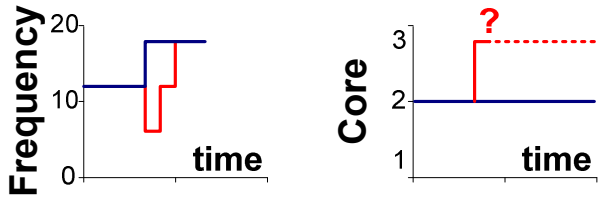
Phase 3 Work Packages



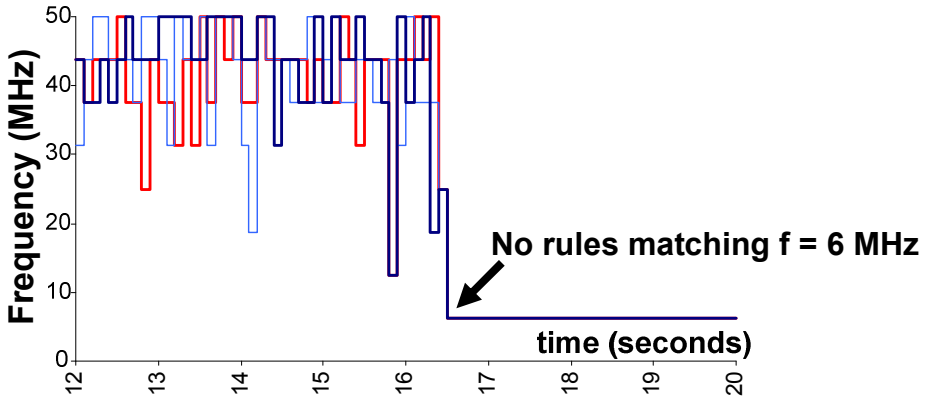
Ensuring Usability

- Actions should be reversible
  - Occasional actions by bad rules should only have temporary effect

	Condition	Action	Fitness
1	XX001XXX	00001	197
2	XX001XXX	01111	4
3	XX000XXX	00001	246

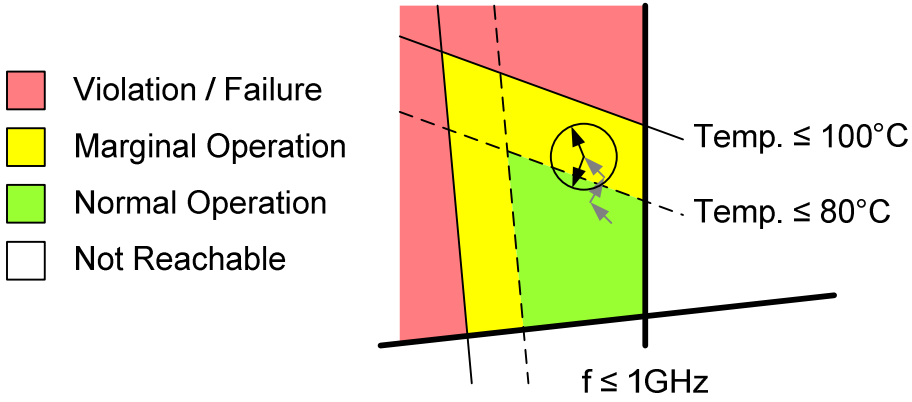


- Covering must be ensured by initial ruleset

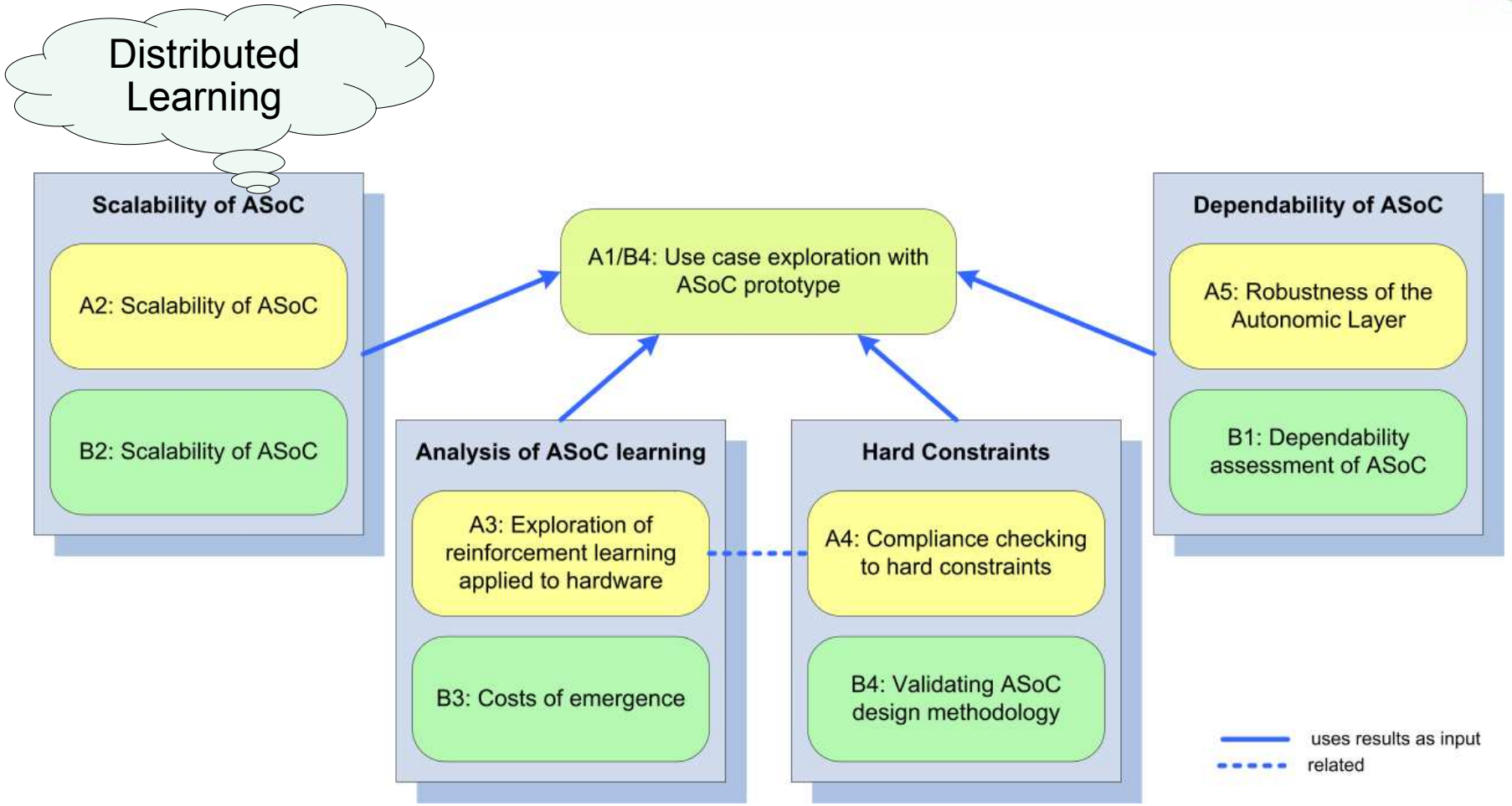


Stability Considerations

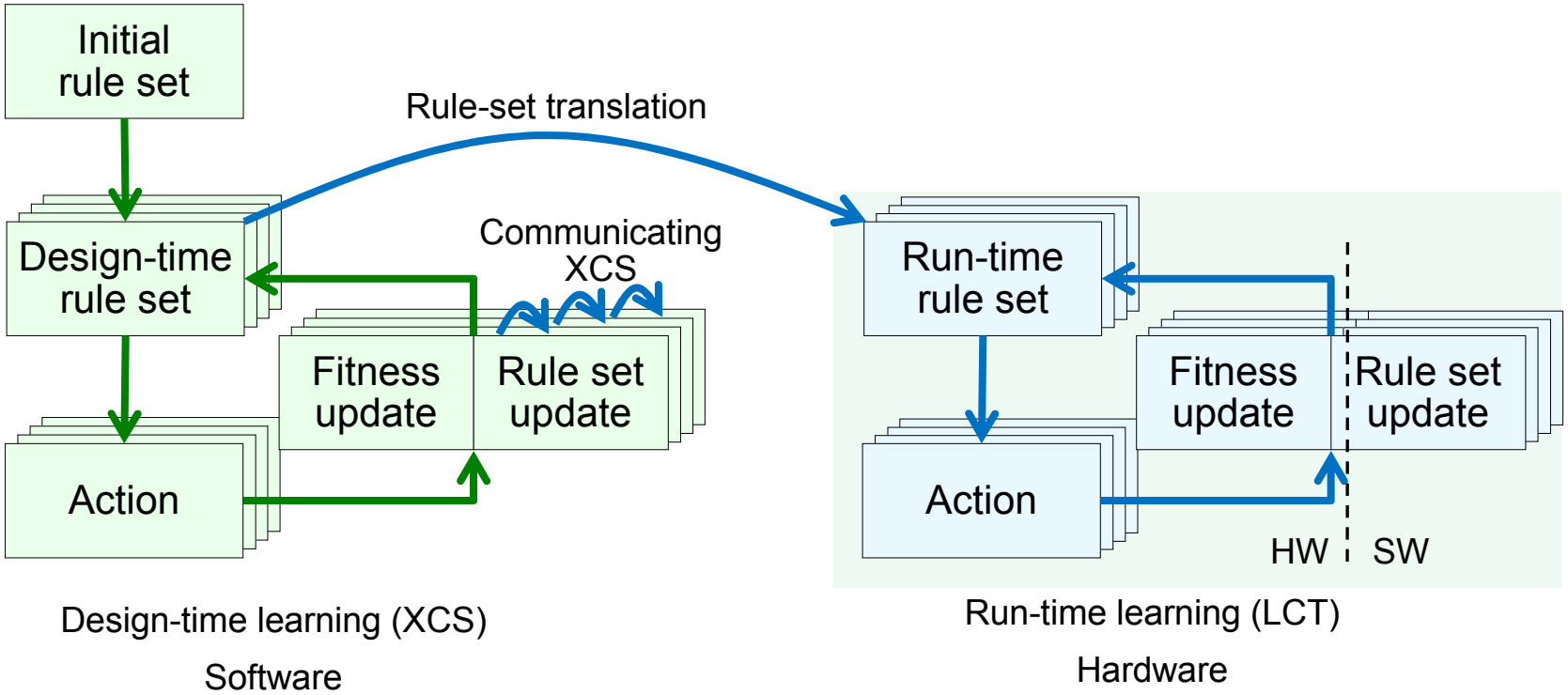
- Changing parameters does not mean the system is unstable
  - – Learning requires trying out new things
  - – Oscillations around an unreachable “perfect” operating state
- Many small steps are better than few large ones
  - \+ Large steps can theoretically reach a target state more quickly
  - – However, trying out large steps can have stronger side effects



Phase 3 Work Packages



Distributed Learning



- Sharing of rules between design- and run-time
- Distribution of rules among AEs

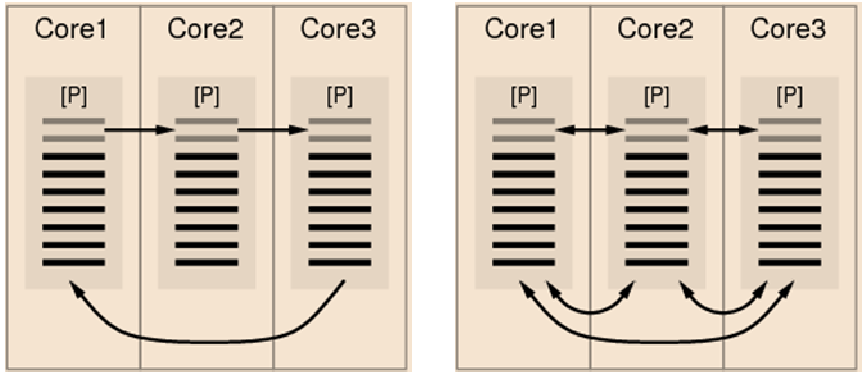
Distributed XCS: Experimental Setup

- Simulation of Cell processor
  - Goal: maximize frequency, avoiding temperature-dependent timing errors
  - Variations in load and ambient temperature
- Two learning phases: design- and run-time
- Explore
  - Communication topologies

    uni-, bidirectional, complete graph
  - Classifier migration strategies

    random, numerosity, fitness, prediction error
  - Classifier deletion strategies

    prediction error, fitness



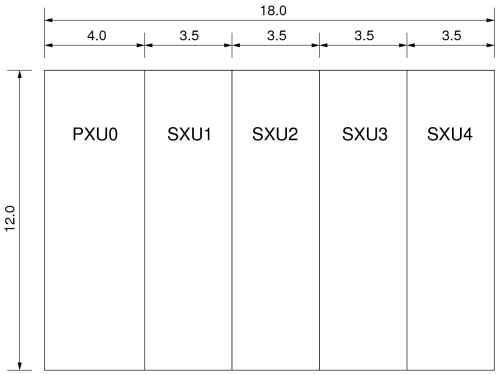
Topologies of communicating XCS

Distributed XCS: 10-Core Extension

- Procedure

① Explore using 5-core model ✔

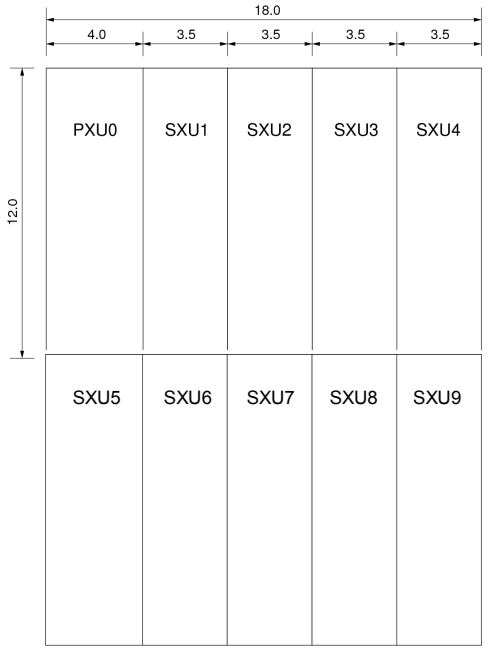
② Apply best strategies to 10-core model



5-core model

Best configuration

- Migration strategy:
  Emigrate by fitness
- Deletion strategy:
  Deletion by prediction Error



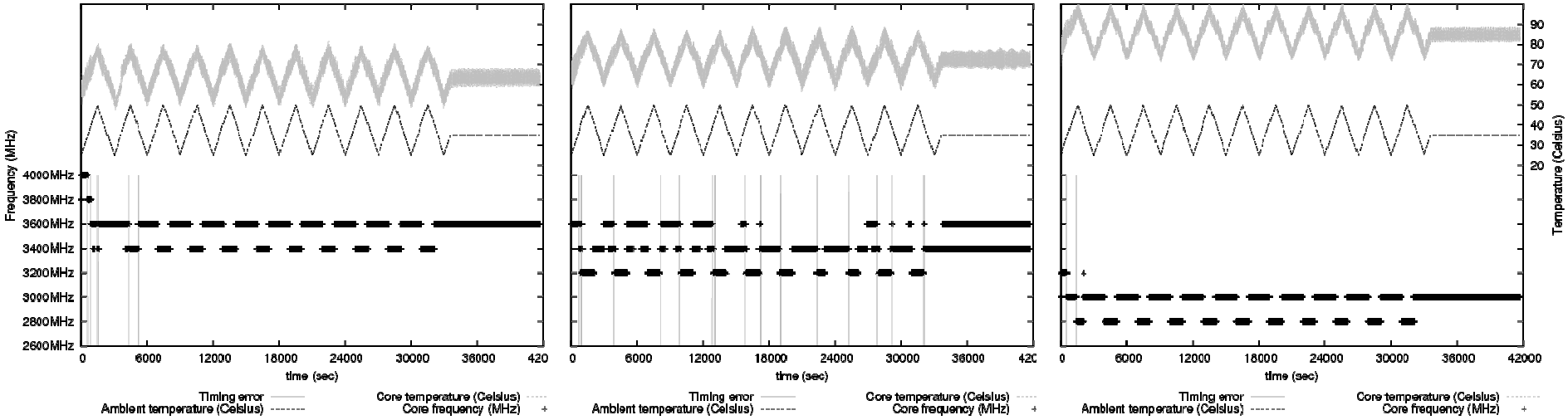
10-core model

[ICAC2011b]

Distributed XCS: Results
10-core model, topology with 3 neighbors

Thermal convection resistance:

r = 0.05 r = 0.10 r = 0.20

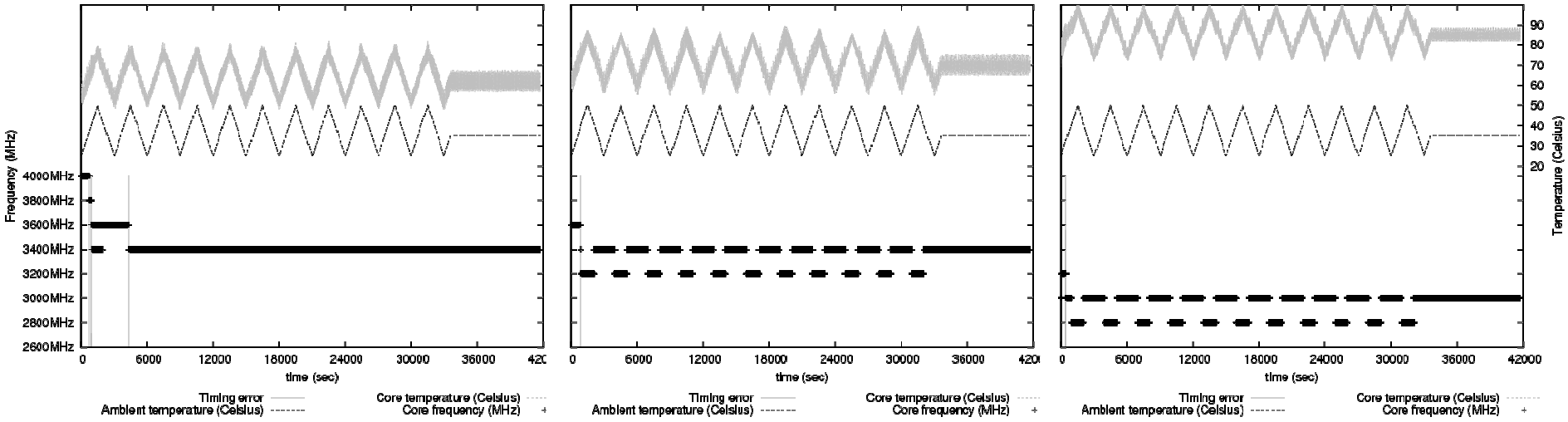


[ICAC2011b]

Distributed XCS: Results
10-core model, topology with 5 neighbors

Thermal convection resistance:

r = 0.05 r = 0.10 r = 0.20

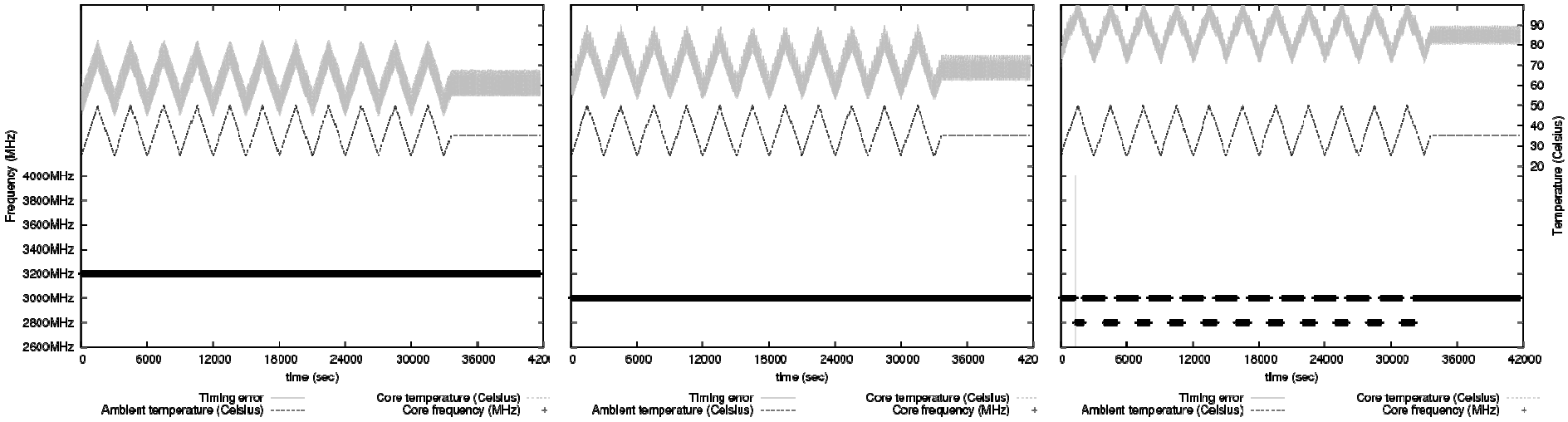


[ICAC2011b]

Distributed XCS: Results
10-core model, topology with 9 neighbors

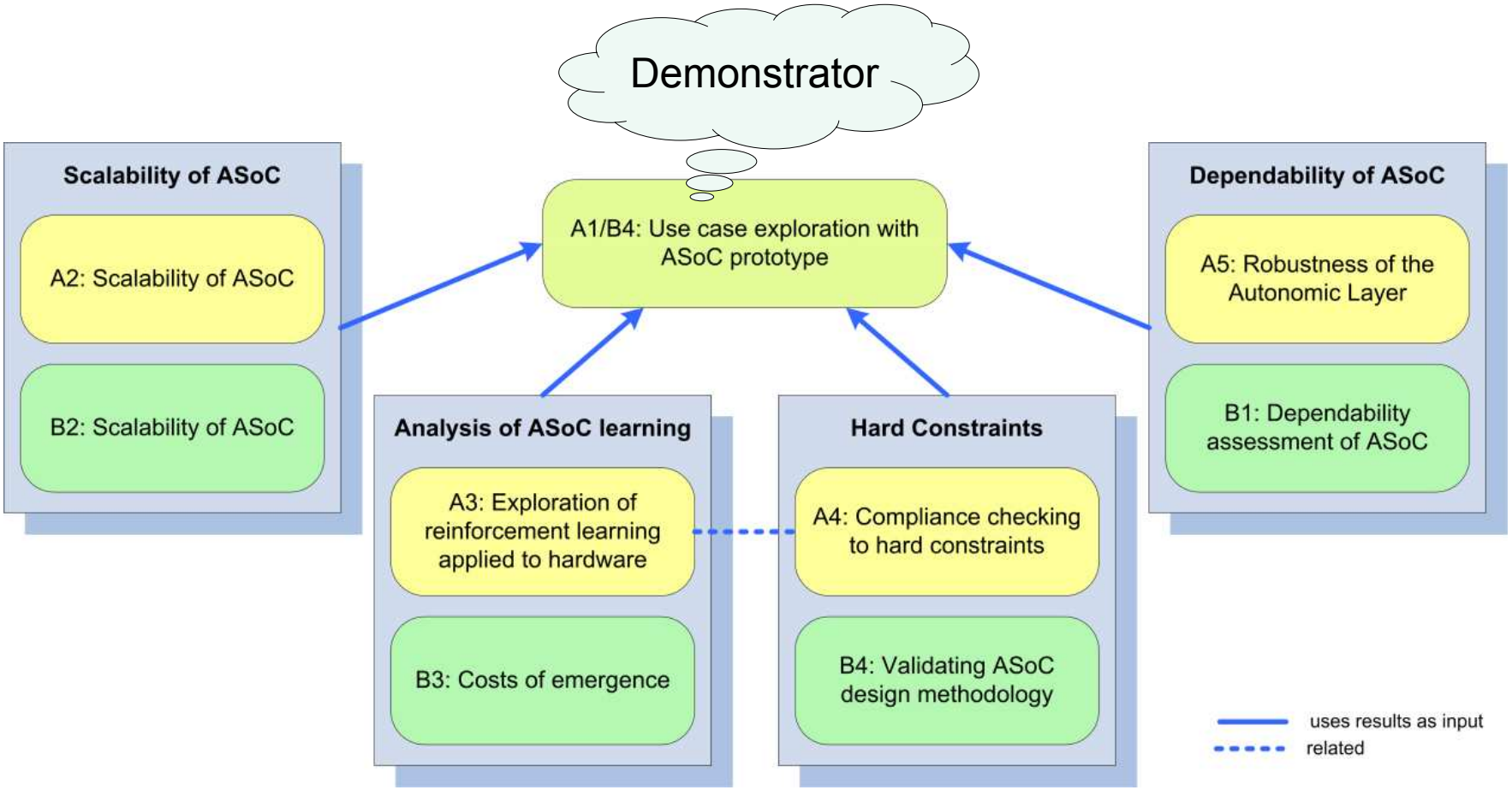
Thermal convection resistance:

r = 0.05 r = 0.10 r = 0.20



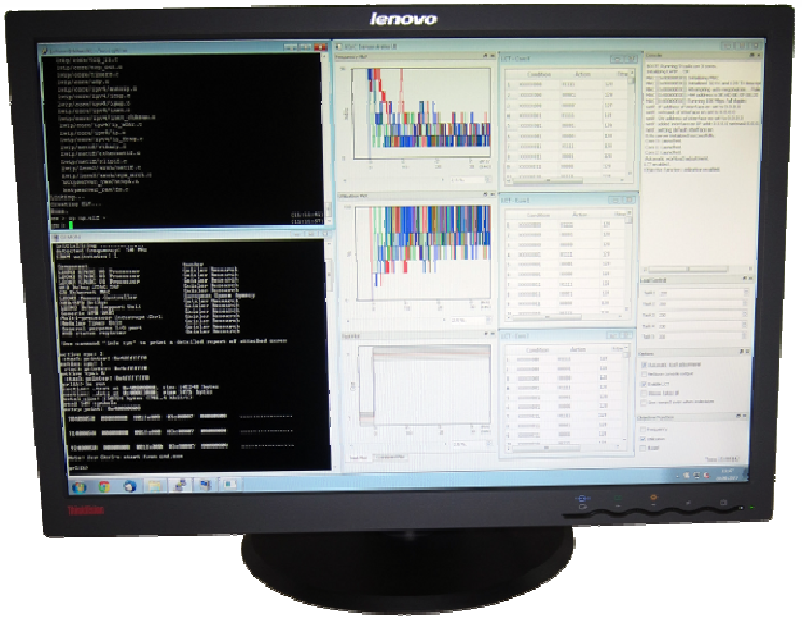
[ICAC2011b]

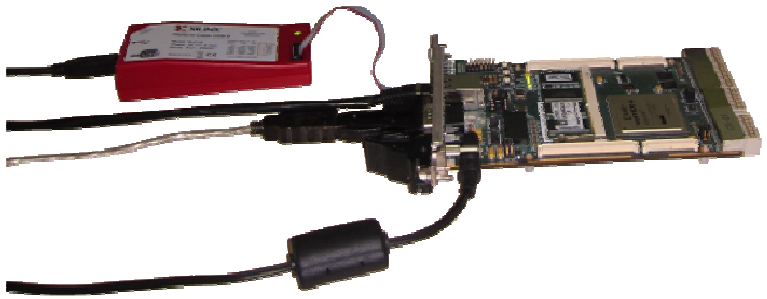
Phase 3 Work Packages



Demonstrator

- Leon3-based FPGA prototype
  - 3 cores with independent AEs
  - Network support using LwIP
  - Various tasks to generate workload
- GUI for online interaction and visualization of results





	FFs	LUTs	BRAM/mul	%
Leon3	1749	8936	28/1	–
L3+AE	2122	10213	29/2	14.3%
LCT	66	116	1/1	1.4%
Monitors	55	114	0/0	1.3%
Actuators	64	318	0/0	3.7%
AE IF	173	399	0/0	4.5%

Synthesis results for Xilinx Virtex 4 VLX100, overhead % calculated from FPGA slices

Possibilities for Further Work

- Interoperability between ASoC and OS
  - Allow AE to make use of task migration capabilities of OS
  - Give AE more action possibilities for migrating tasks
  - Possible solution to the irreversible task migration problem
- Extension of ASoC prototype to more cores
  - Hierarchically structured system of both functional and autonomic elements
  - Differentiation of local and system-wide actions and goals
- Autonomic elements for other functional components
  - Possibilities include bus, memory controller, I/O, etc.
- Action timing
  - Better handling of monitors that take a long time to react to actions
  - Allow overlapping actions instead of forced delays

Summary

- ASoC: From concept to prototype
  - Make use of distributed machine learning techniques within the hardware confines of a system on chip
  - Consider autonomic enhancements during the design process to provide the necessary configuration of the run-time system
- What ASoC can't do
  - Provide hard real-time guarantees in respect to performance
  - Eradicate the need for design-time
- What ASoC can do
  - Alleviate the burden on the designer by moving certain design-time decisions to run-time
  - Dynamically adapt to changing environments and operating modes
  - Learn to cope with situations that were not intended by the designer, and in which ordinary systems would typically fail

Phase 3 Publications

- [ICAC11a] J. Zeppenfeld, A. Herkersdorf. Applying Autonomic Principles for Workload Management in Multi-Core Systems on Chip, International Conference on Autonomic Computing, ICAC, Karlsruhe, Germany, June 14-18, 2011.
- [ICAC11b] A. Bernauer, G. Arndt, O. Bringmann, W. Rosenstiel. Autonomous Multi-Processor-SoC Optimization with Distributed Learning Classifier Systems XCS, International Conference on Autonomic Computing, ICAC, Karlsruhe, Germany, June 14-18, 2011.
- [BICC10] A. Bernauer, J. Zeppenfeld, O. Bringmann, A. Herkersdorf, W. Rosenstiel. Combining software and hardware LCS for lightweight on-chip learning, DIPES/BICC 2010, IFIP AICT 329, p.279-290, Brisbane, Australia, September 20-23, 2010.
- [IJCNN10] B. Rakitsch, A. Bernauer, O. Bringmann, W. Rosenstiel. Pruning population size in XCS for complex problems, International Joint Conference on Neural Networks at the World Congress on Computational Intelligence (WCCI), Barcelona, Spain, July 18-23, 2010.
- [SORT10] J. Zeppenfeld, A. Bouajila, A. Herkersdorf, W. Stechele. Towards Scalability and Reliability of Autonomic Systems on Chip, 1st IEEE Workshop on Self-Organizing Real-Time Systems, Carmona, Spain, May 4, 2010.
- [ARCS10] J. Zeppenfeld, A. Herkersdorf. Autonomic Workload Management for Multi-Core Processor Systems, ARCS, Hannover, Germany, February 22-25, 2010.