System Level Simulation of Autonomic SoCs with TAPES

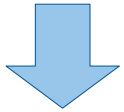
Technische Universität München
Institute for Integrated Systems

Andreas Lankes

Table of Content

Paradigm Shift in IC Design

- CMOS Technology Evolution leads to
  - Decreasing feature size
  - Increasing component count / complexity
  - Gap between complexity and designer productivity

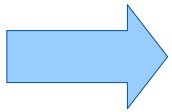


Decreased Reliability because of
- Design errors
- Manufacturing faults
- Soft errors

- Autonomic SoCs
  - ensures **reliable** system
  - optimizes **performance**
  - ... and/or **power**

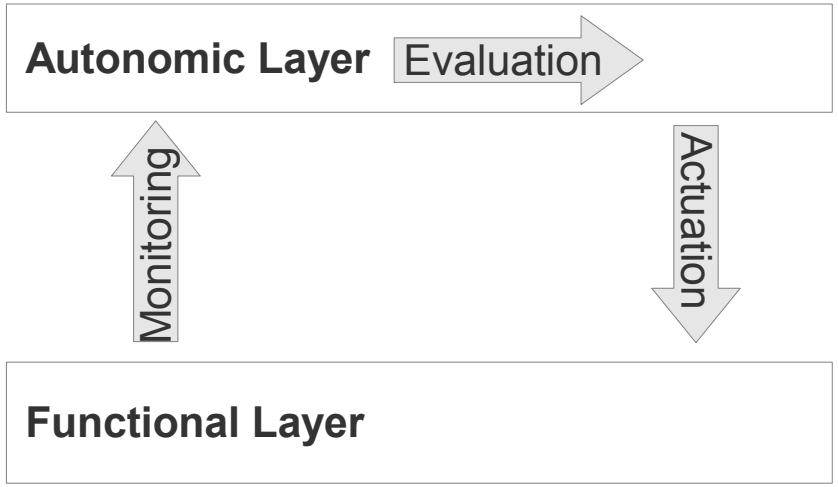
What is an Autonomic SoC?

- Autonomic SoCs adapts itself at runtime to
  - changing conditions, e.g. workloads, ...
  - occurrence of errors



to reach optimization goals:
power, performance, reliability

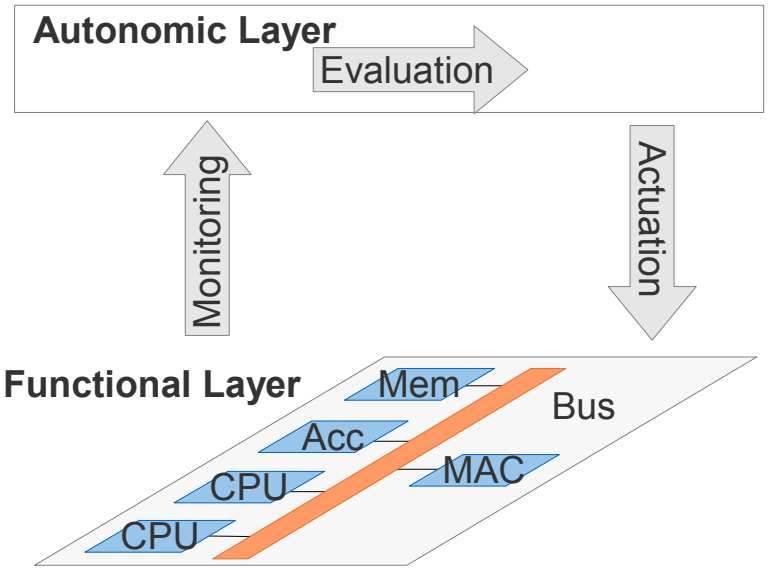
- Basic structure
  - 2 Layer Model
    - Functional Layer (regular IPs)
    - Autonomic Layer (supervisor)
  - Control Loop between layers



Design of Autonomic SoCs

Designing an Autonomic SoC requires

- Definition of suitable system architecture
- Adaptation strategy to reach optimization target at run-time



IC Design becomes even more complex

Simulation tools for exploration required

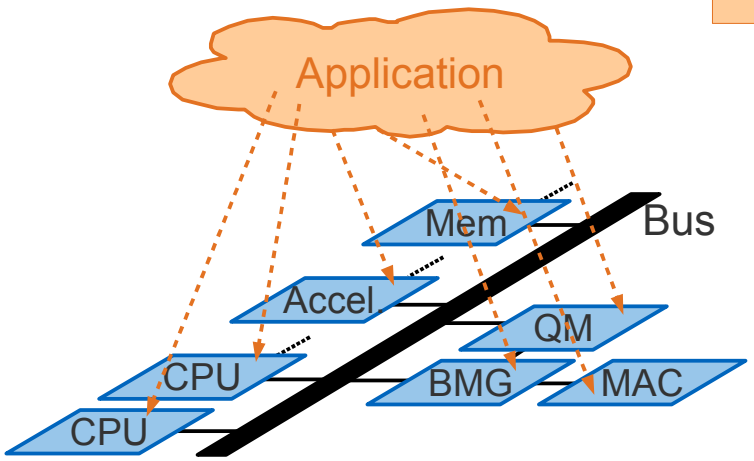
Table of Content

- Introduction
- **TAPES Simulator**
- Autonomic Extension of TAPES
- Experiment
- Conclusion

TAPES – System Level SoC Simulator

Trace-based **A**rchitecture **P**erformance **E**valuation with **S**ystemC

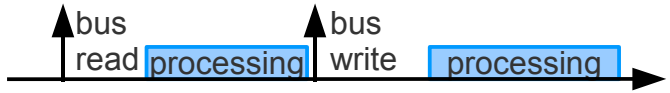
- Abstract simulation
  - modules modeled as black boxes
  - no data processing



Application i.e. Functionality abstracted by Application Traces

- internal (= processing): delay
- external: transactions

e.g. cpu trace

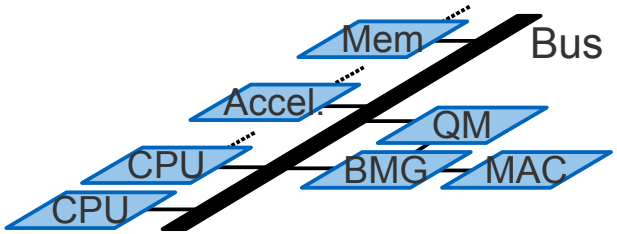


Specification of traces

- Task binding/ scheduling
- Memory hierarchy

TAPES – System Level SoC Simulator

Trace-based **A**rchitecture **P**erformance **E**valuation with **S**ystemC



Application

Specification of Application Traces

System configuration
- Resource allocation
- Resource properties (clock, buffer sizes, power, ...)

External stimuli

TAPES Simulator

SystemC TLM

Simulation output:
- Processing run times
- Resource utilization
- Latencies
- Power dissipation

Table of Content

- Introduction
- TAPES Simulator
- **Autonomic Extension of TAPES**
- Experiment
- Conclusion

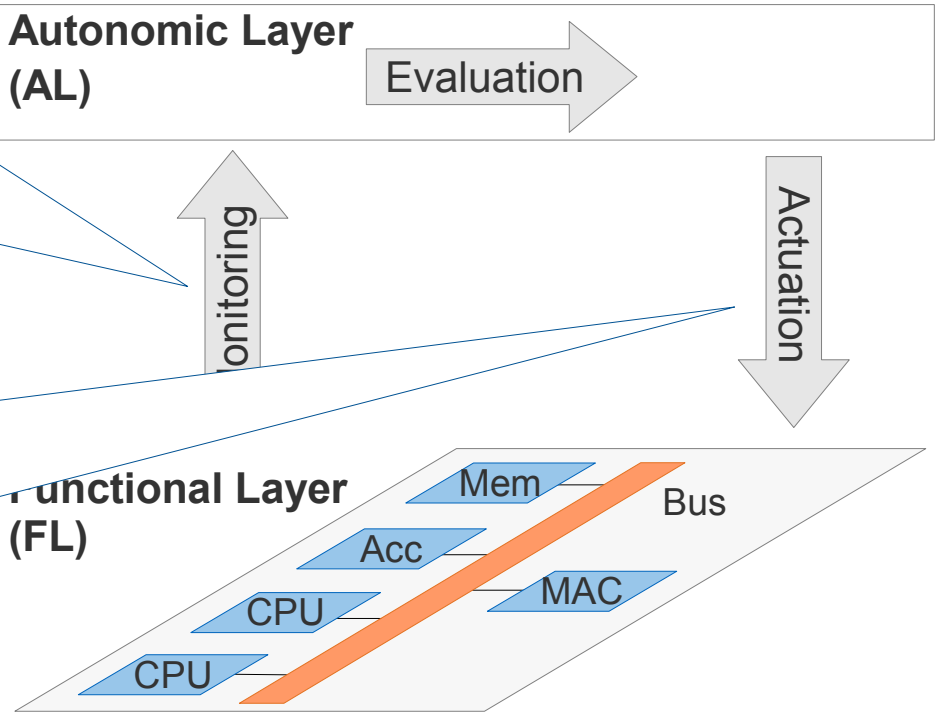
Metrics

Monitoring: information currently provided to the AL by the FL.

- Load and activity of modules
- Occurrence of errors in modules
- Memory usage and buffer fill levels

Actuation: list of currently adaptable parameters.

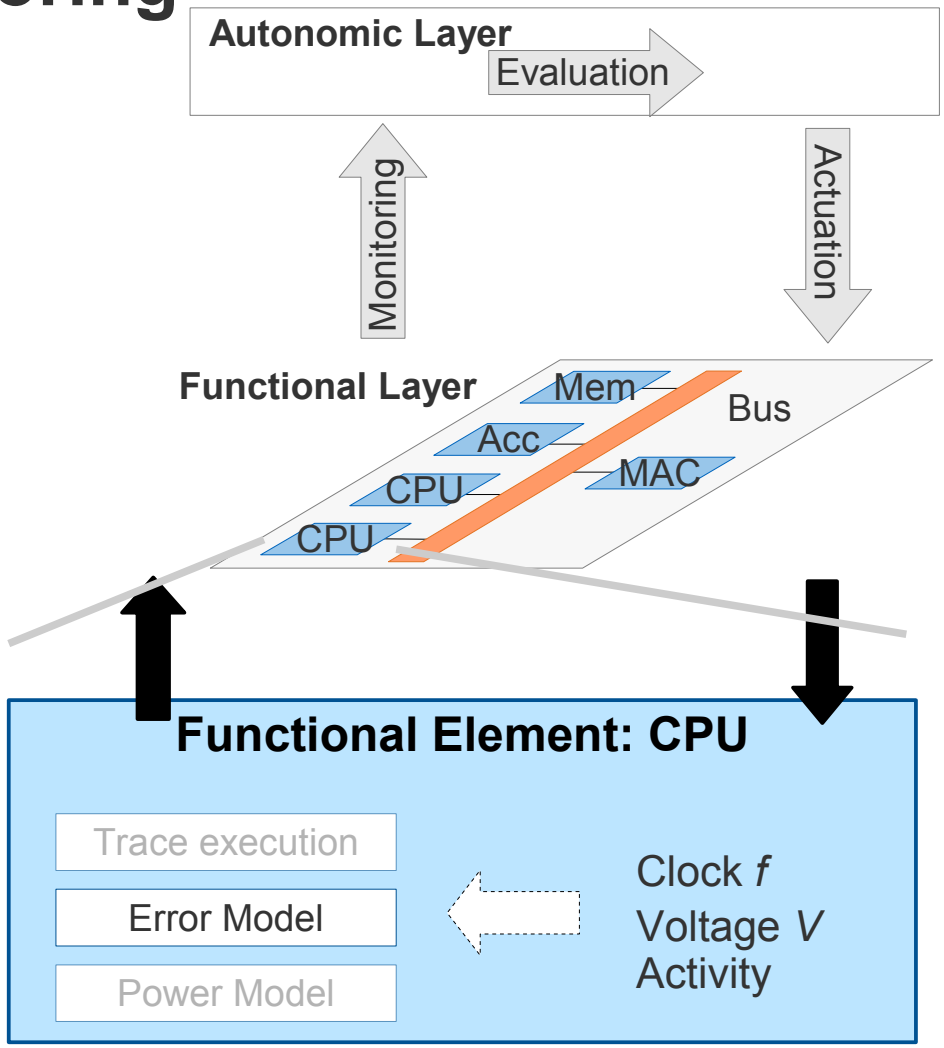
- Activation and Deactivation of modules
- Modification of clock frequency *f*
- Adaptation of supply voltage
- Change SoC bus properties



Functional Layer: Monitoring

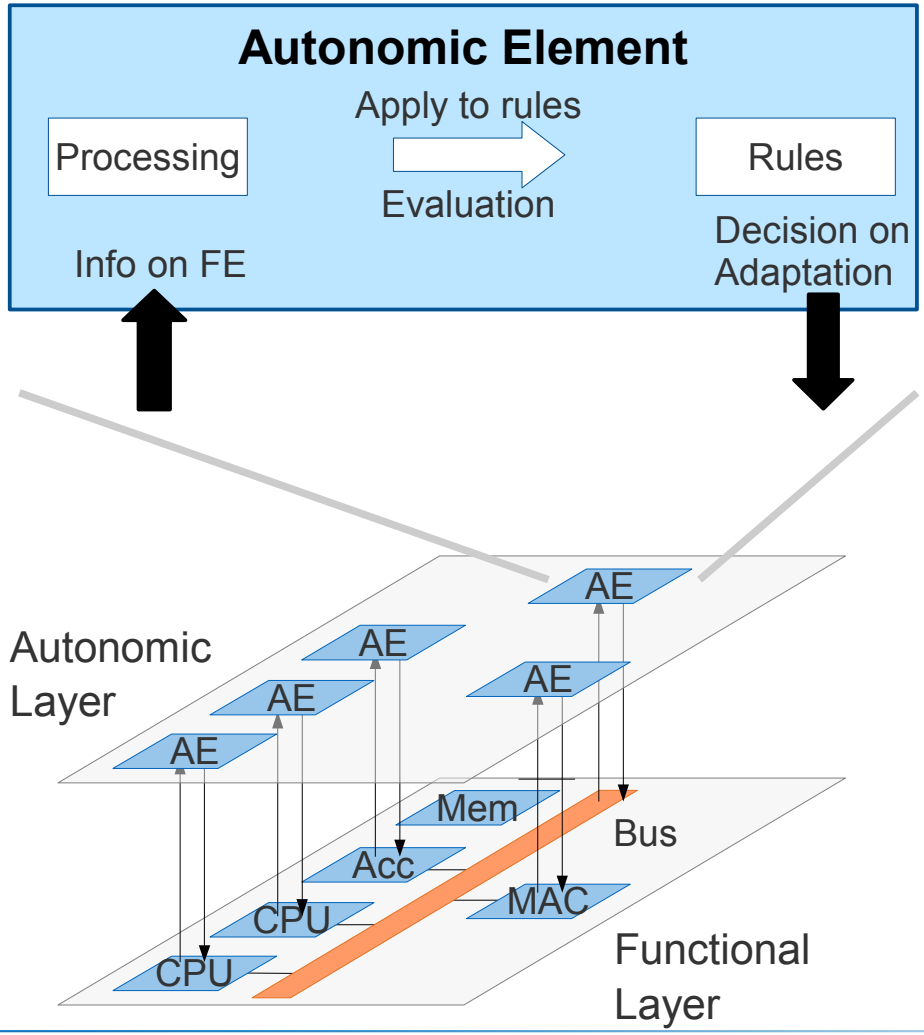
Extensions of the Functional Layer

- Functionality to push monitored data to AL
- Error Models simulating occurrence of timing violations, particle hits, ...
  - Up to now: static error model (independent of current module state)
  - Goal: dynamic error models (error rate depends on state of module)



Autonomic Layer: Evaluation

- Autonomic Layer
  - Built of Autonomic Elements (AEs)
  - Implements adaptation strategy
- Autonomic Elements
  - Fully functional
    <> FL's modules are abstract
  - Processing of information:
    averaging, event counting, ...
  - Rules:
    compare to thresholds, intervals, ...



Functional Layer: Actuation

Extensions of the Functional Layer

- Enable adaptation of Parameters at runtime
  - Functional behavior of modules
  - Performance (-> Power)
- Adaptable parameters
  - Clock frequency *f:*
    requires adaptation of
    application traces
  - Voltage *V*:
    influence on power model
  - Power state: running, standby modes, ...
    influence on power model and functional behavior

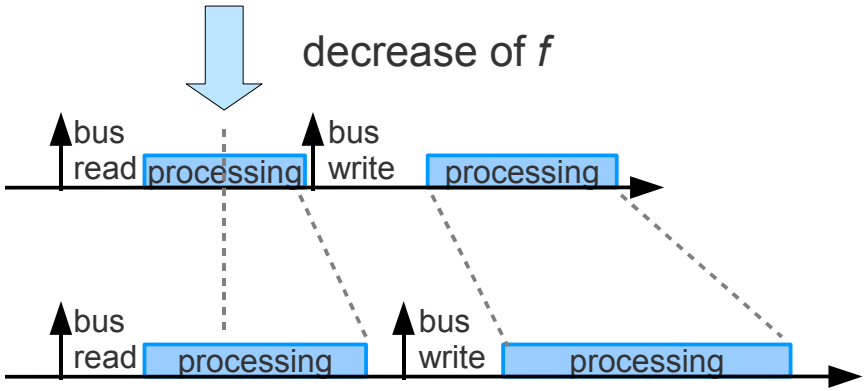


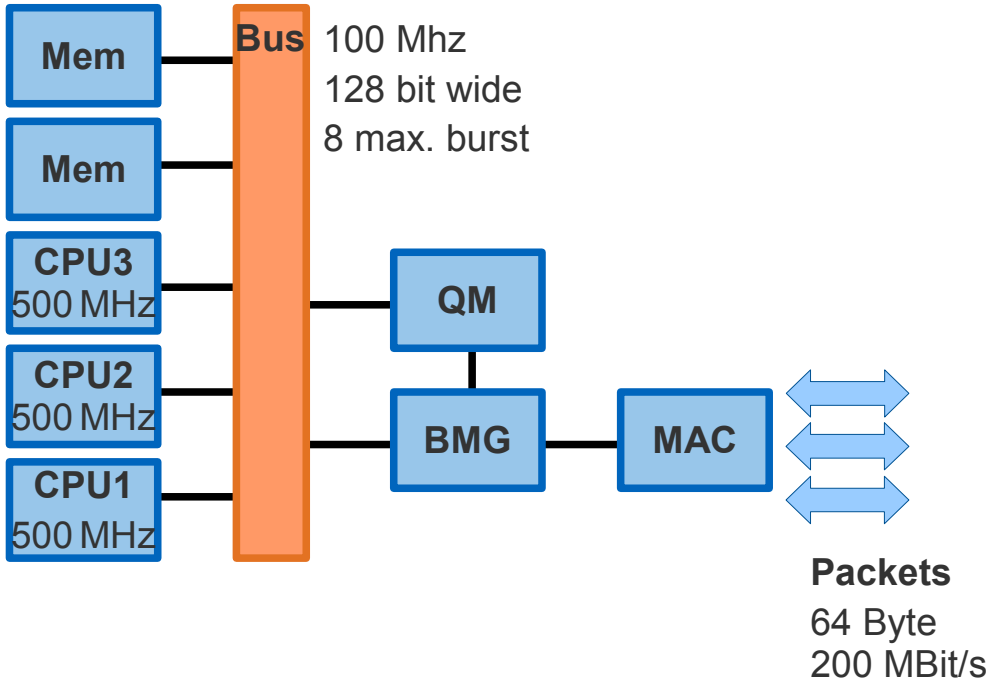
Table of Content

- Introduction
- TAPES Simulator
- Autonomic Extension of TAPES
- **Experiment**
- Conclusion

Simulation of Network Processor SoC

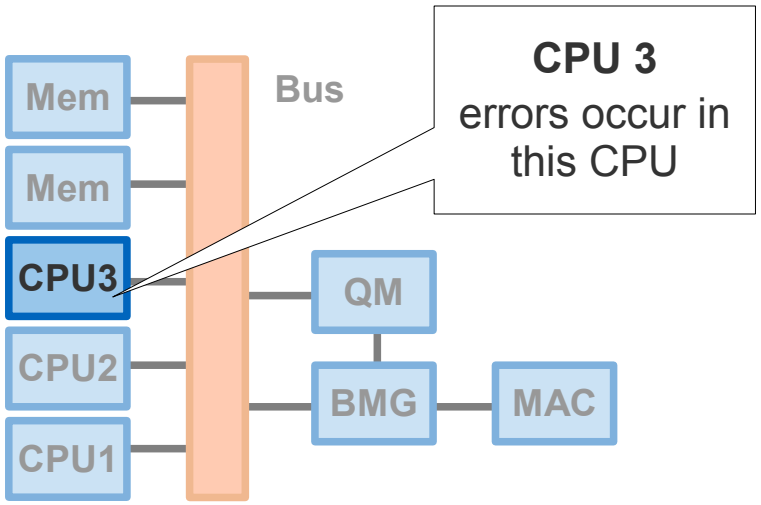
SoC adapting itself to occurring errors

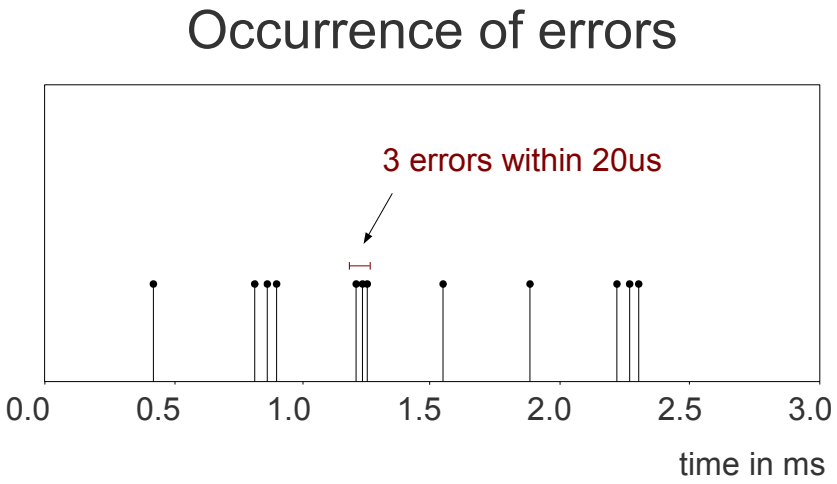
- Network processor SoC
  - Bus based architecture
  - Basic IP forwarding
- Example Adaptation Strategy
  - deactivate CPUs if error frequency gets too high
  - adapt clock frequency of CPUs to hold load of CPUs at 85%



Injection of Errors

- Occurrence of errors specified in error file

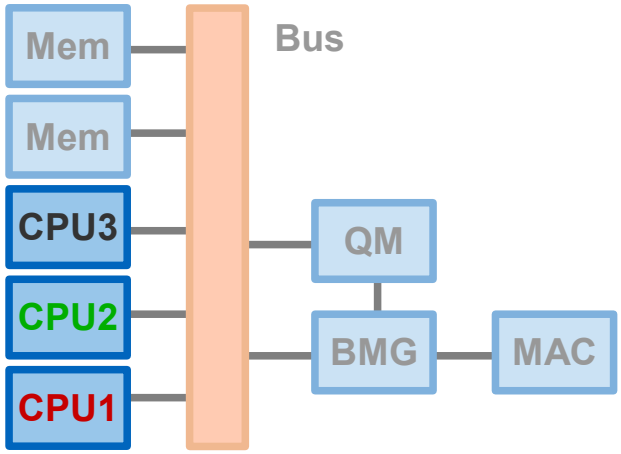


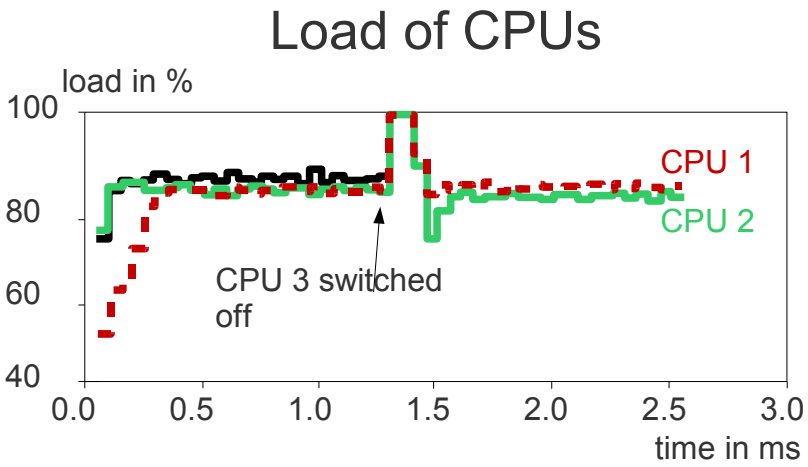


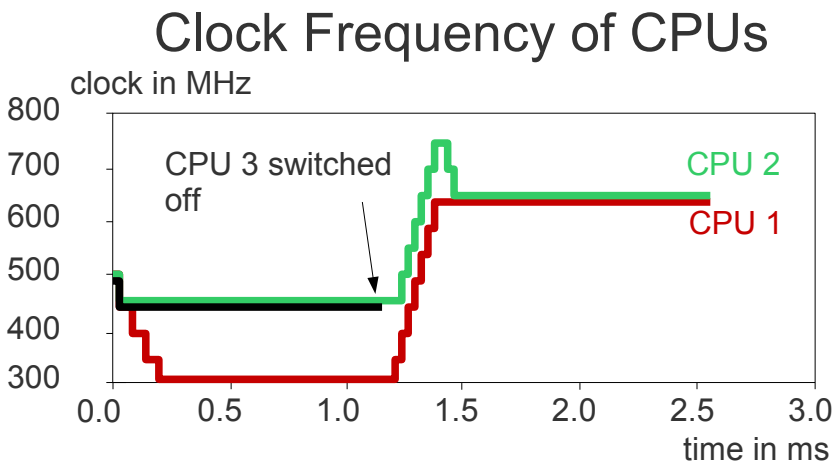
- Adaptation Rule
  > 2 errors within 20 us => deactivation of that module

Simulation Results: CPUs

- Adaptation of CPUs
  keep load at approx 85%

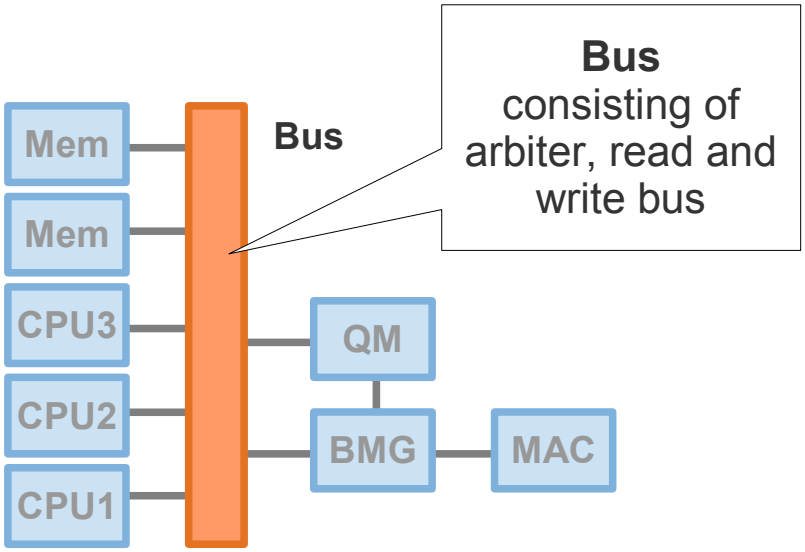


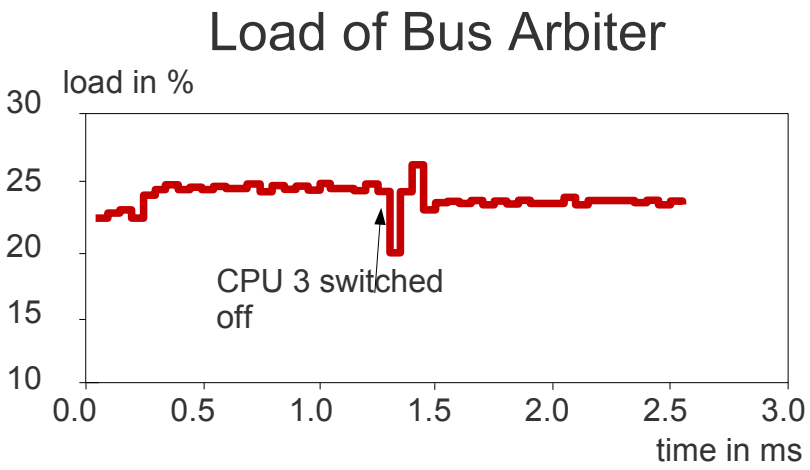


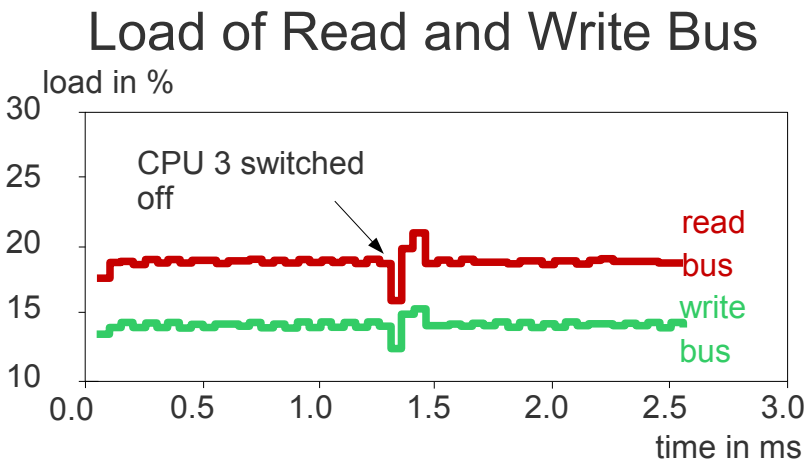


Simulation Results: Bus

- Effects on bus load

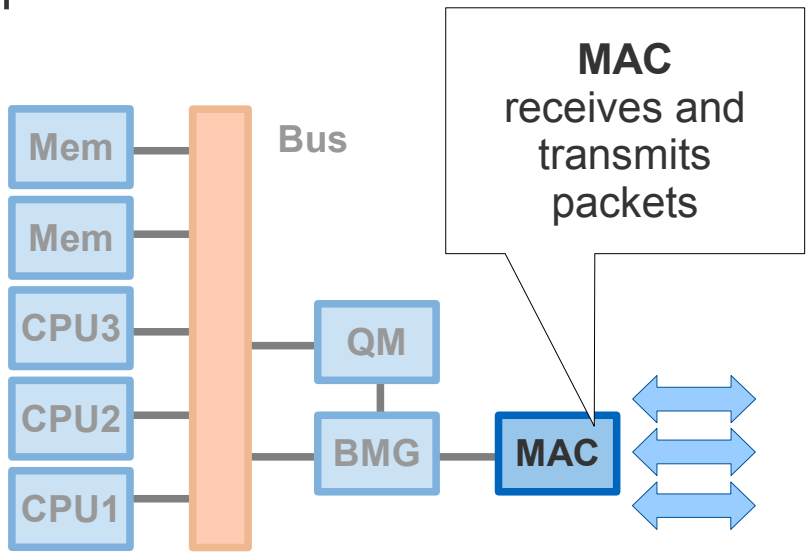




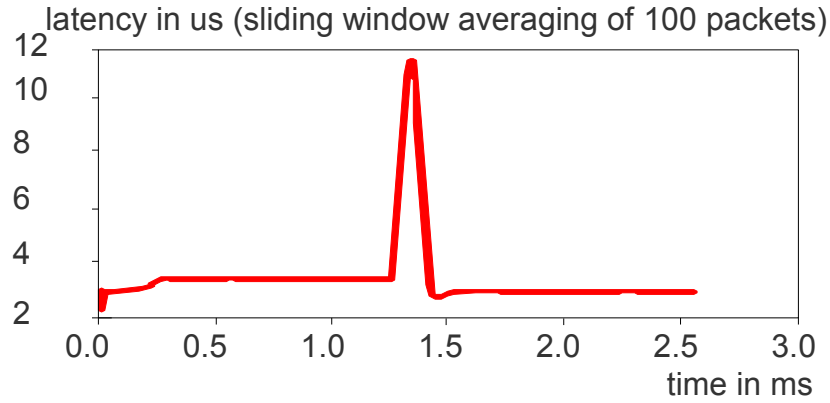


Simulation Results: Traffic

- Effect on in- and out-coming packets



Packet Latency



Incoming and Outgoing Packet Rate

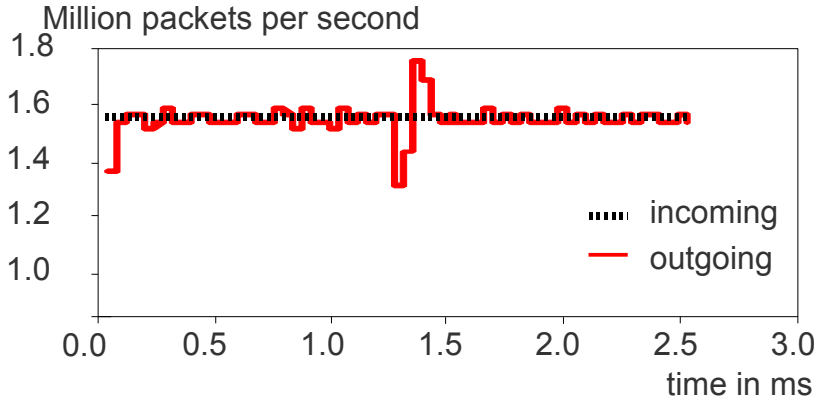


Table of Content

- Introduction
- TAPES Simulator
- Autonomic Extension of TAPES
- Experiments
- **Conclusion**

Conclusion

- System level simulator supporting simulation of autonomic SoCs
  - System adapts architectural parameters itself ...
  - to changing workloads or occurrence of errors

  - Investigation of adaptation strategies
  - Evaluation in terms of performance and power
- Interactive architecture exploration
  - High simulation performance (example above requires 1 sec on P4M)
  - High configurability
    XML configuration file for architectural parameters
- Future Work
  - dynamic error models
  - configurable autonomic rules

Thank You!

Any Questions?