Year 1 Review
Brussels, January 23rd, 2008

Transversal Activity

Achievements and Perspectives :

Design for Predictability and Performance

leader : Bengt Jonsson

Uppsala University

High-Level Objectives

- technology and design techniques for achieving predictability of systems
  - especially on modern platforms
- trade-offs between performance and predictability

Predictability transverses all levels of abstraction in embedded systems design:

- Verification, modeling, compilation, OS, platforms.

Industrial Sectors

- safety-critical systems
  - transportation, power automation, medical systems, ...
  - Market of over $900 million in 2008 [int. ARC Advisory Group]
- sectors, where systems failure may lead to economic consequences
  - consumer electronics, telecom, ...

Partners

Modeling & Validation:

- EPFL - Lausanne (Tom Henzinger)
- INRIA – France (Alain Girault)
- Uppsala (Bengt Jonsson)
- PARADES – Italy
  (Alberto Sangiovanni–Vincentelli)

Code Generation & Timing analysis

- Dortmund (Peter Marwedel)
- Saarland (Reinhard Wilhelm)
- TU Vienna (Peter Puschner)

OS & Networks

- Cantabria
  (Michael Gonzalez–Harbour)
- York (Alan Burns)

Hardware Platforms & MPSoC

- Bologna (Luca Benini)
- Braunschweig (Rolf Ernst)
  [affiliated]
- ETHZ – Zürich (Lothar Thiele)
- IMEC – Belgium
  (Stylianos Mamagkakis)
- Linköping (Petru Eles)

What is Predictability?

- Predictability == Determinism?, or
- Predictability == Conformance to Specification?
- Wikipedia: Predictability == degree to which a correct prediction or forecast of a system's state can be made either qualitatively or quantitatively.
  - i.e.,: How precisely can it be analyzed?
  - A function of:
    - System architecture, features
    - What quantity to predict
    - Power of Analysis techniques

A closer view

- Predicting a quantity (e.g., worst-case execution time) must consider a large number of details.
- predictability of details add up
- Contribution of, e.g., memory access, depends on
  - Uncertainty of prediction (e.g., of cache hit/miss)
    - depends on system structure **and** analysis technique
  - ”cost” for misprediction (difference miss - hit)

Building Excellence

- Most existing work is within one system level, e.g,:
  - Modeling and verification of timed component-based systems,
  - Timing analysis for programs
  - Compiler techniques for timing and memory predictability
  - OS Scheduling and resource management
  - Predictability in memory

Main Aim:

- Integrate research across different levels of abstraction

System Modeling and Validation

Integrate Component-based modeling (µCCM component model) [Thales] and Contract-Based Scheduling [FRESCOR]

Integrate Timed Automata Techniques and Real-Time Calculus (MPA-RTC), for predictability analysis of distributed systerms

Conceptual viewpoint [EPFL]

- Defines predictability as (a form of) determinism,
- Robustness as (a form of) continuity.

Modeling & Validation:
- EPFL - Lausanne (Tom Henzinger)
- INRIA – France (Alain Girault)
- Uppsala (Bengt Jonsson)
- PARADES (Alberto Sangiovanni-Vincentelli)

Code Generation & Timing analysis
- Dortmund (Peter Marwedel)
- Saarland (Reinhard Wilhelm)
- TU Vienna (Peter Puschner)

OS & Networks
- Cantabria (Michael Gonzalez–Harbour)
- York (Alan Burns)

Hardware Platforms & MPSoC
- Bologna (Luca Benini)
- Braunschweig (Rolf Ernst) [affiliated]
- ETHZ – Zürich (Lothar Thiele)
- IMEC – Belgium (Stylianos Mamagkakis)
- Linköping (Petru Eles)

Compiling and Timing Analysis

Integration of

- WCET timing analysis tool [AbsInt]
- compiler for Infineon TriCore 1.3 [Dortmund]

to build the leading WCET-aware compiler.

Analysis of Hardware Platform features

- cache architectures [Saarland]
- predictable multicore designs [Predator]

Improved WCET analysis techniques

- Interplay with task scheduling [Saarl]
- Dependency on input parameters [Absint, MdH, Saar]

Modeling & Validation:

- EPFL - Lausanne (Tom Henzinger)
- INRIA – France (Alain Girault)
- Uppsala (Bengt Jonsson)
- PARADES (Alberto Sangiovanni–Vincentelli)

Code Generation & Timing analysis

- Dortmund (Peter Marwedel)
- Saarland (Reinhard Wilhelm)
- TU Vienna (Peter Puschner)

OS & Networks

- Cantabria (Michael Gonzalez–Harbour)
- York (Alan Burns)

Hardware Platforms & MPSoC

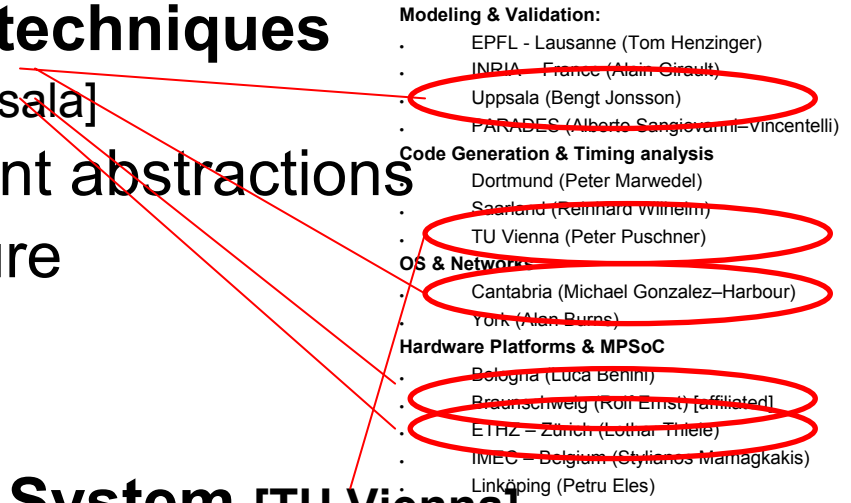
- Bologna (Luca Benini)
- Braunschweig (Rolf Ernst) [affiliated]
- ETHZ – Zürich (Lothar Thiele)
- IMEC – Belgium (Stylianos Mamagkakis)
- Linköping (Petru Eles)

OS/MiddleWare/Networks

Compare different analysis techniques

[Braunschweig, Cantabria, ETHZ, Uppsala]

- Understand effect of different abstractions
- in relation to system structure



Time-Predictable Operating System [TU Vienna]

Realization of simple OS prototype

- All decisions (control flow, communication) at compile-time
- Evaluation on code transformed to single-path code

Shows that accurate timing analysis can be achieved.

System and Platform Architecture

Generating predictable (fault-tolerant) schedules: [Linköping, DTU Copenhagen]

- supporting soft and hard real-time constraints
- maximizing the overall utility.

quasi-static scheduling strategy:

- synthesize set of schedules off-line
- select appropriate schedule at run-time

Predictability for MPSoC Architectures [Linköping, Bologna, Braunschweig]

- Bus access policy and bus access schedule
  - Guaranteeing both predictability and efficiency.
- Analysis of L2 cache architectures
  - wrp. to impact on overall system behaviour

Modeling & Validation:
- EPFL - Lausanne (Tom Henzinger)
- INRIA – France (Alain Girault)
- Uppsala (Bengt Jonsson)
- PARADES (Alberto Sangiovanni–Vincentelli)

Code Generation & Timing analysis
- Dortmund (Peter Marwedel)
- Saarland (Reinhard Wilhelm)
- TU Vienna (Peter Puschner)

OS & Networks
- Cantabria (Michael Gonzalez–Harbour)
- York (Alan Burns)

Hardware Platforms & MPSoC
- Bologna (Luca Benini)
- Braunschweig (Rolf Ernst) [affiliated]
- ETHZ – Zürich (Lothar Thiele)
- IMEC – Belgium (Stylianos Mamagkakis)
- Linköping (Petru Eles)

Overall Assessment and Vision at Y0+1

- Many collaborations working very well
  - WCC Compiler, MPSoC architecture analysis, Definition and assessment of “predictability”, Predictable implementation for Model-based Design,
  - Several more have been initiated:
    - Impact on timing analysis of Scheduling, MPSoC, and other features
- 9 joint publications, several mutual visits and joint projects
  - MORE, Predator, COSTA, HIPEAC

To be improved

- Wider transversal integration across levels of abstraction
  - E.g., Towards implementation of model-based design, timing aware compiler, predictable multicore architecture and operating system
- Global event

Scientific Highlights

- Comparison of different analysis methods
- WCC-aware compiler
- Analysis of L2 Cache performance

Influence of different system abstractions on the performance analysis of distributed hard real-time systems

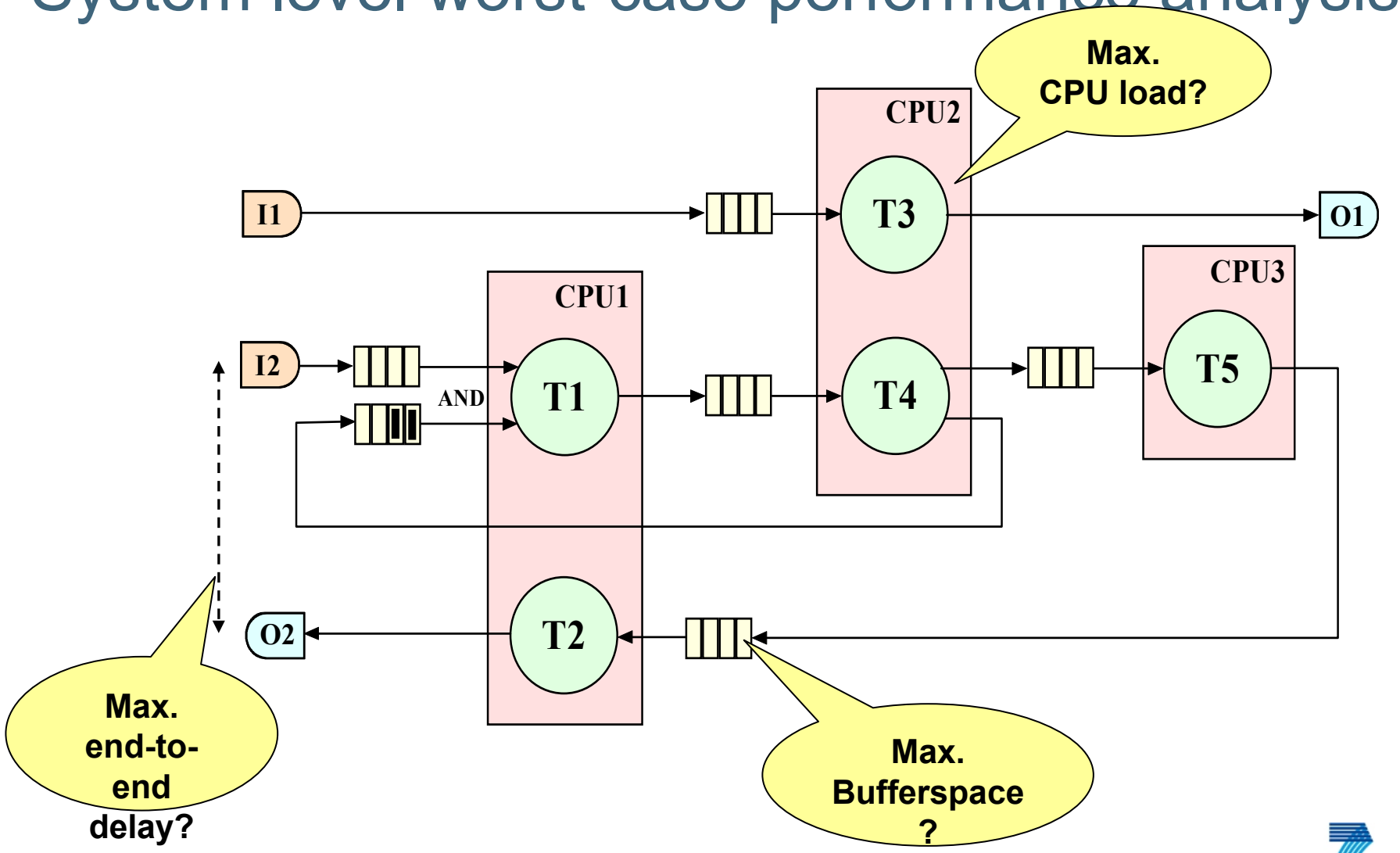
Simon Perathoner [1], Ernesto Wandeler [1], Lothar Thiele [1]

Arne Hamann [2], Simon Schliecker [2], Rafik Henia [2], Razvan Racu [2], Rolf Ernst [2]

Michael González Harbour [3]

[1] ETH Zürich [2] TU Braunschweig [3] Universidad de Cantabria

System level worst-case performance analysis



Motivation

- Scalable analysis only possible by abstracting
- Abstraction causes imprecision for some systems

Major Question:

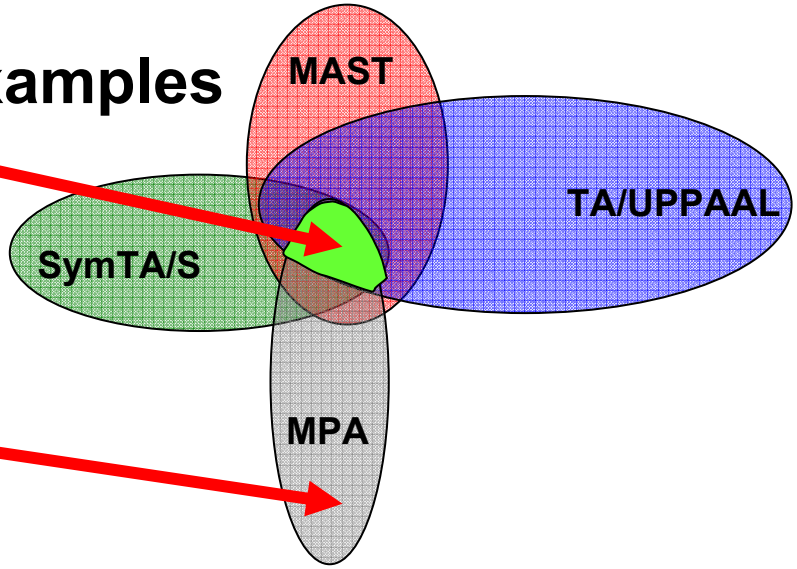
- Which system structures yield imprecise analysis for which modeling abstractions?

Approach

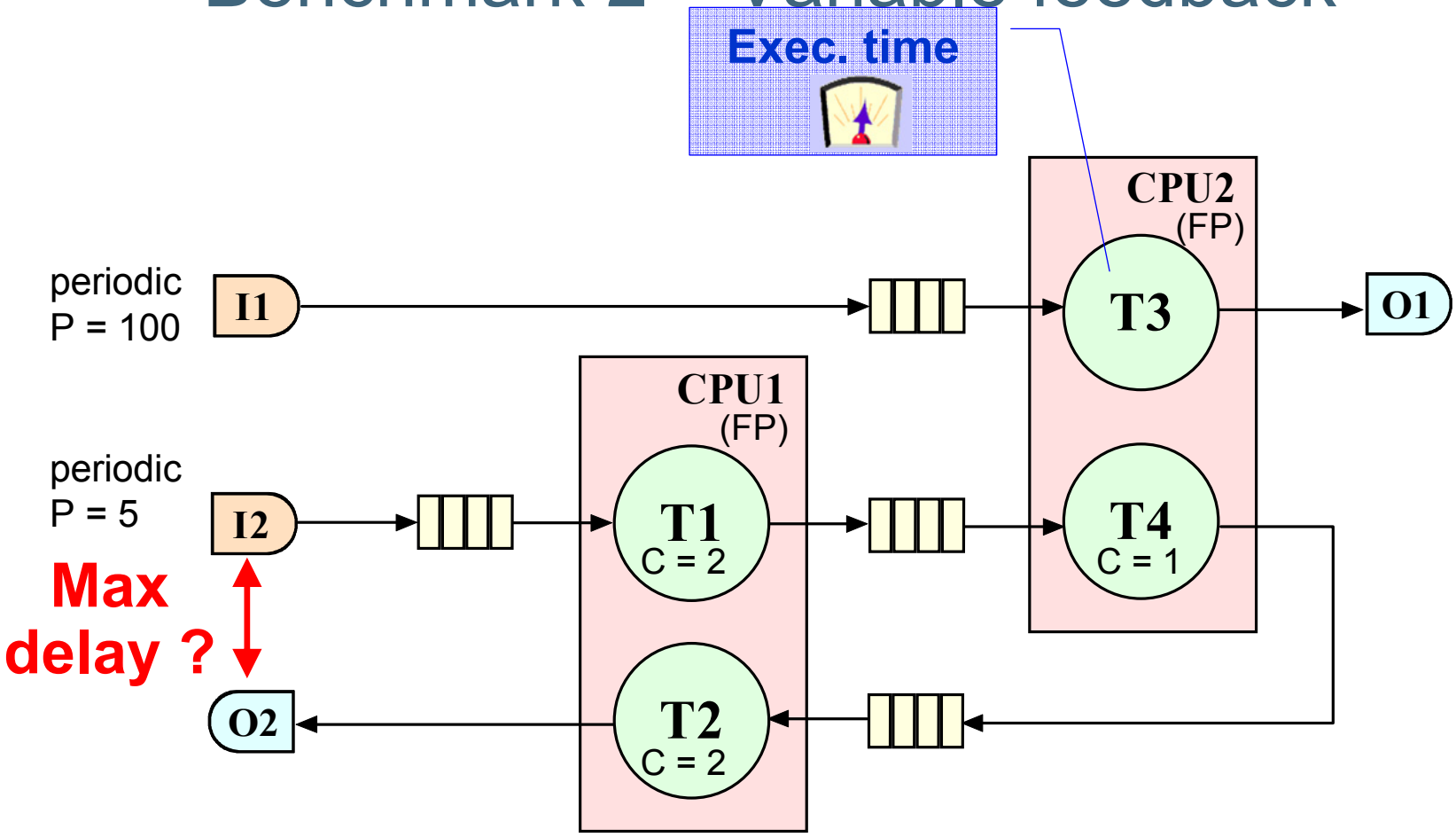
- **Starting point: Leiden Workshop on Distributed Embedded Systems: http://www.tik.ee.ethz.ch/~leiden05/**

- **Define a set of benchmark examples that cover common area**

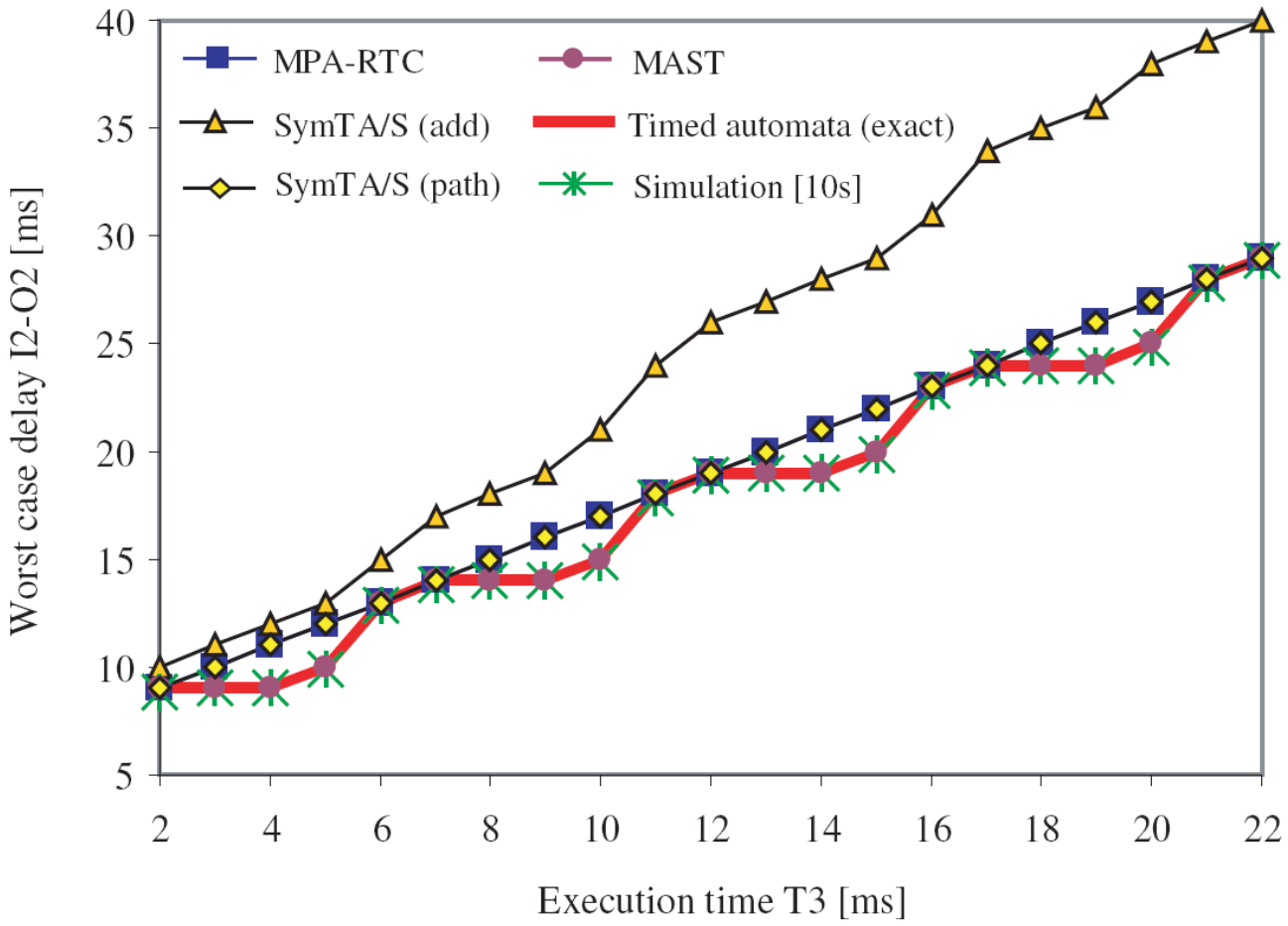
- **Define benchmark examples that show the power of each method**

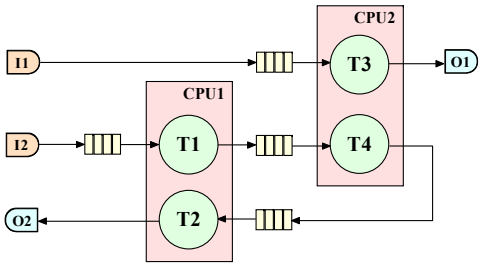


Benchmark 2 – Variable feedback

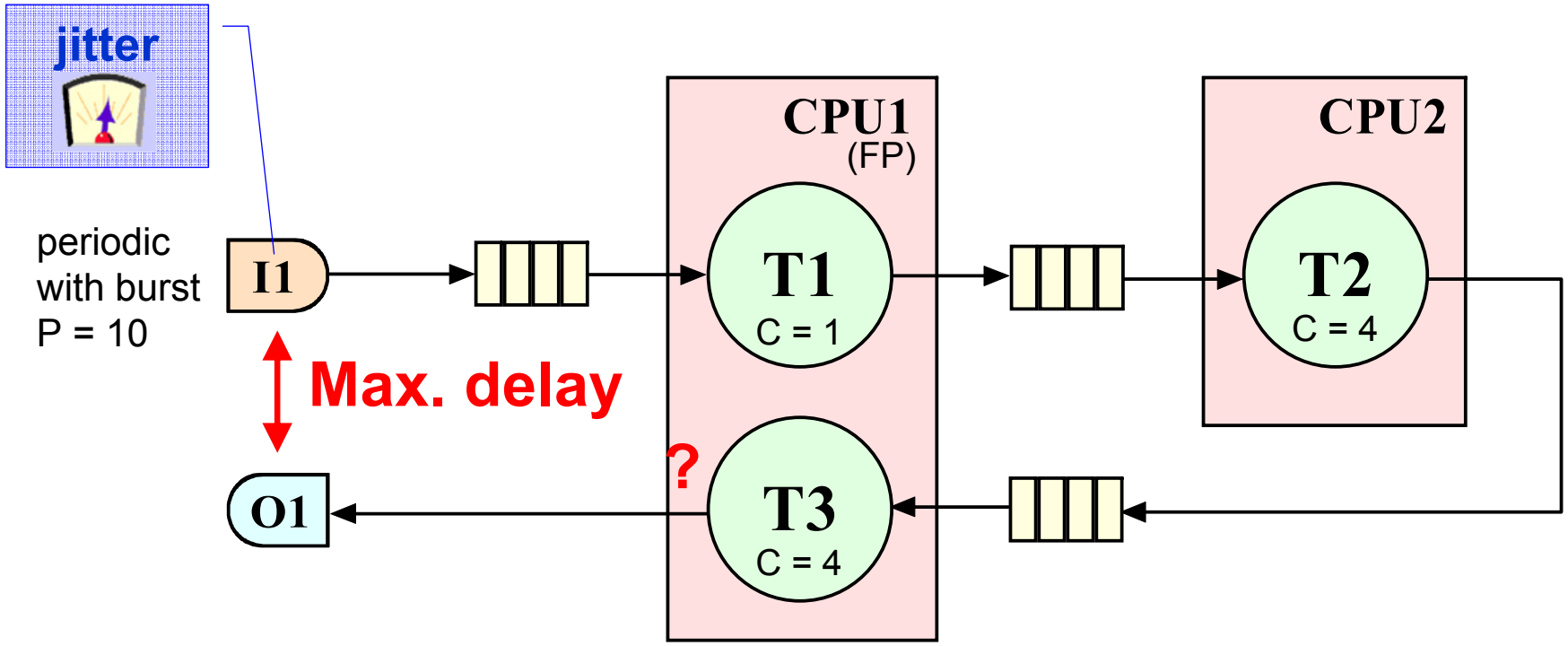


Benchmark 2 – Analysis results



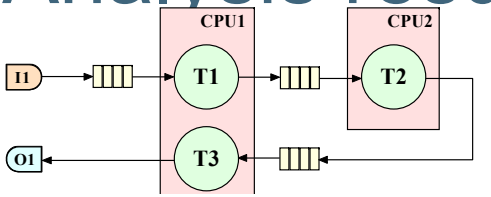


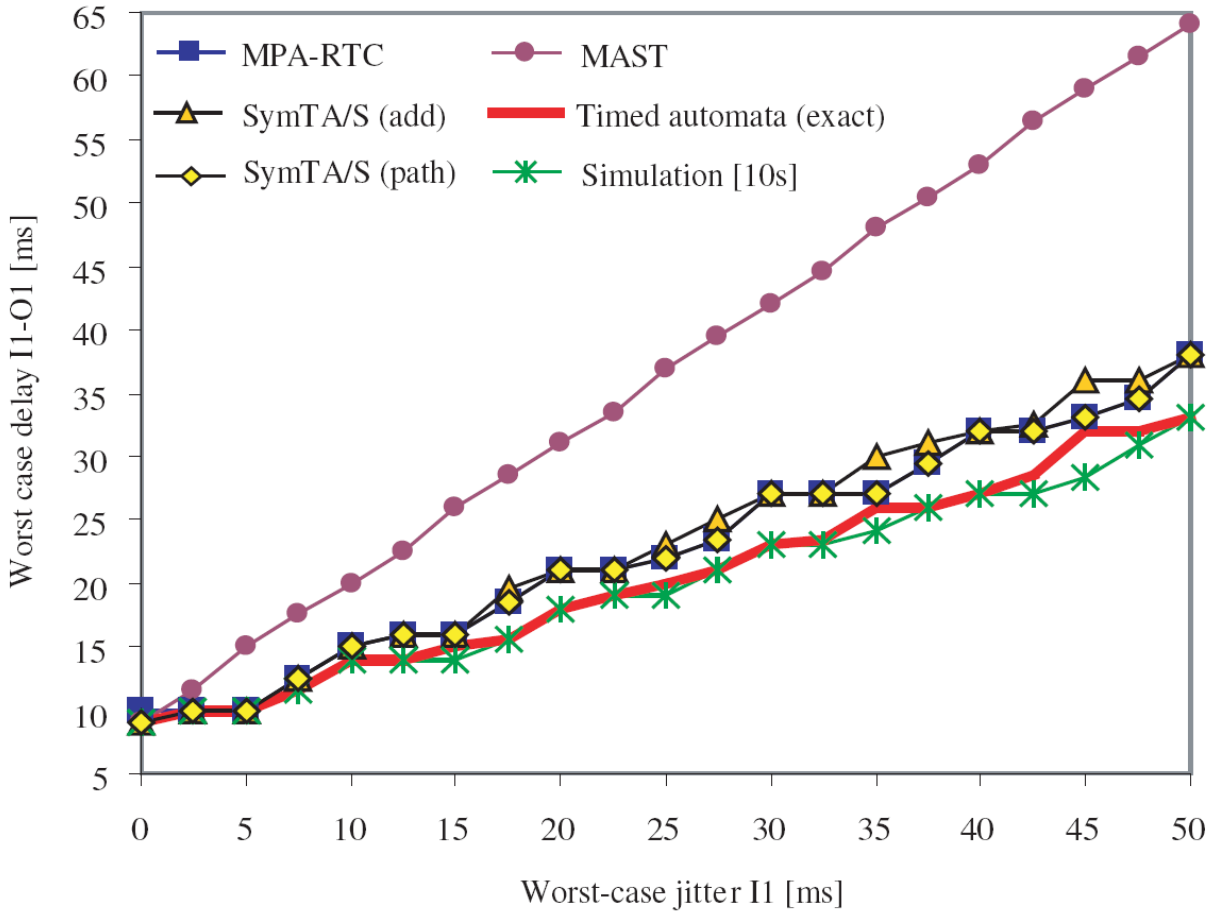
Benchmark 3 – Cyclic dependencies



Benchmark 3 – Analysis results

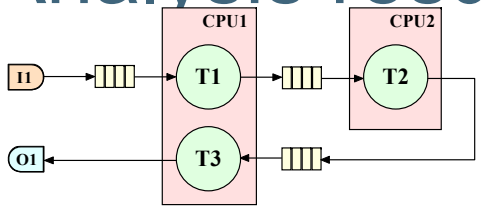
Scenario 1: priority T1 = high
priority T3 = low

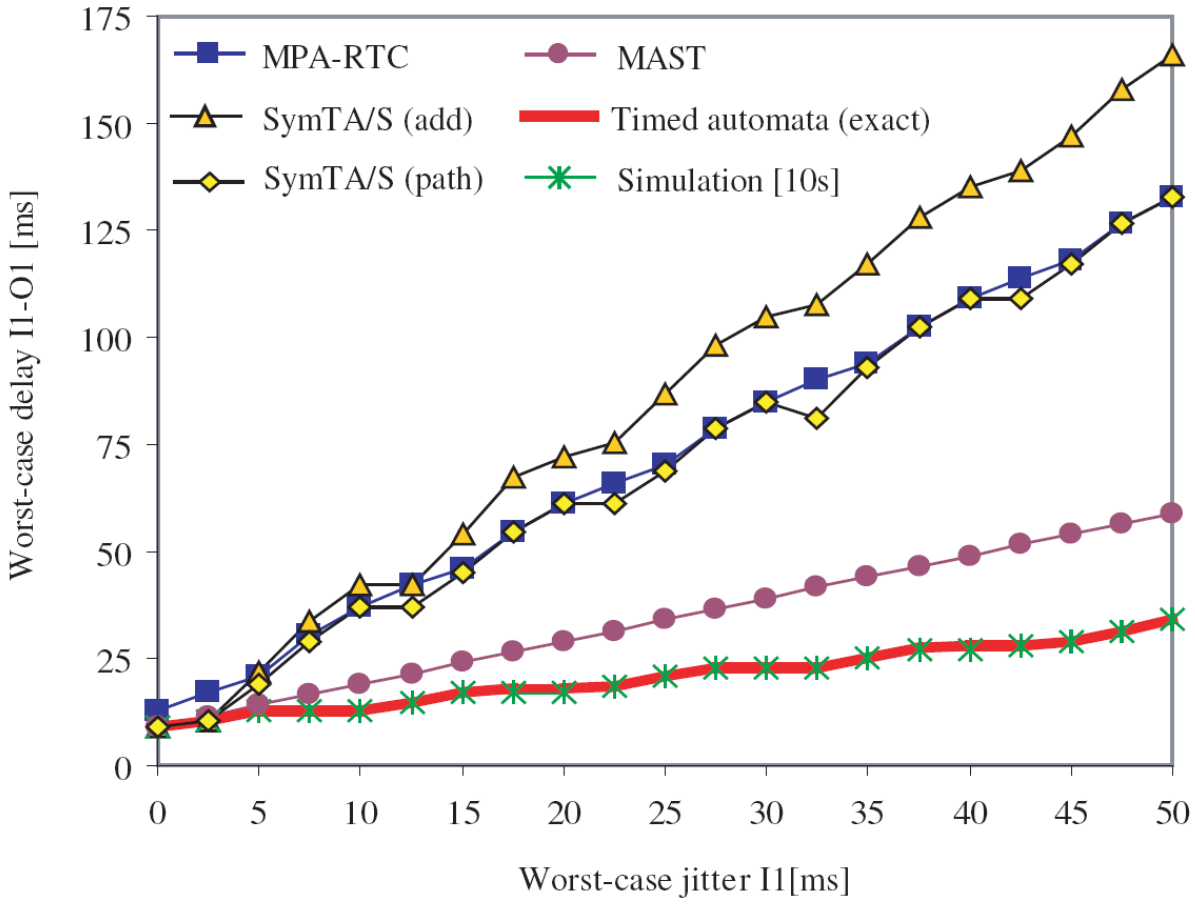




Benchmark 3 – Analysis results

Scenario 2: priority T1 = low
priority T3 = high





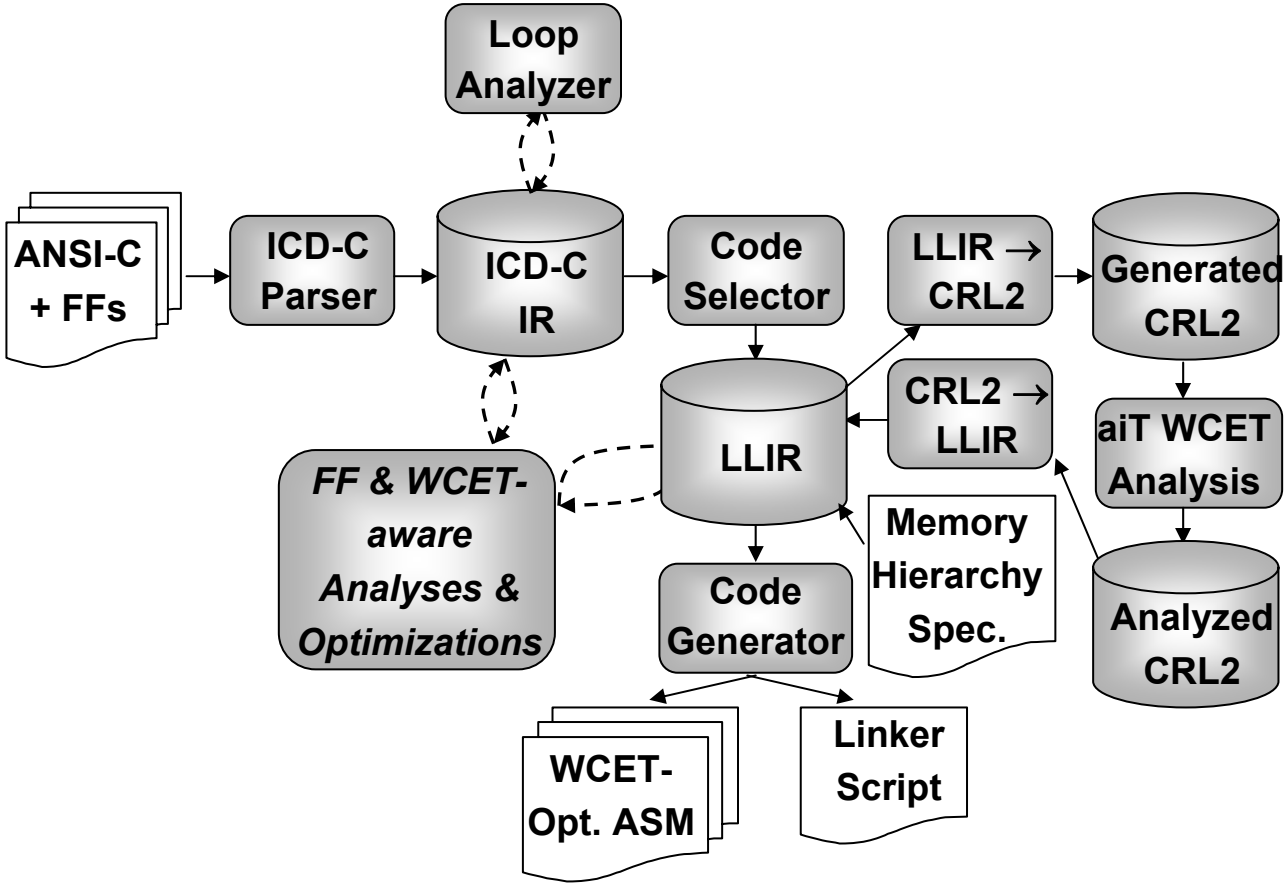
Analysis times [s]

		B2	B3 (sc.1)	B3 (sc.2)
MPA-RTC	min	0.03	0.01	0.04
	med	0.04	0.01	0.15
	max	0.08	0.04	0.30
SymTA/S	min	0.03	0.03	0.03
	med	0.05	0.06	0.34
	max	0.23	0.09	0.80
MAST	min	< 0.5	< 0.5	< 0.5
	med	< 0.5	< 0.5	< 0.5
	max	< 0.5	< 0.5	< 0.5
Timed aut.	min	< 0.5	< 0.5	< 0.5
	med	< 0.5	1.0	< 0.5
	max	< 0.5	52.0	5.5
Simulation	min	< 0.5	0.5	0.5
	med	< 0.5	0.5	0.5
	max	< 0.5	0.5	0.5

Conclusions

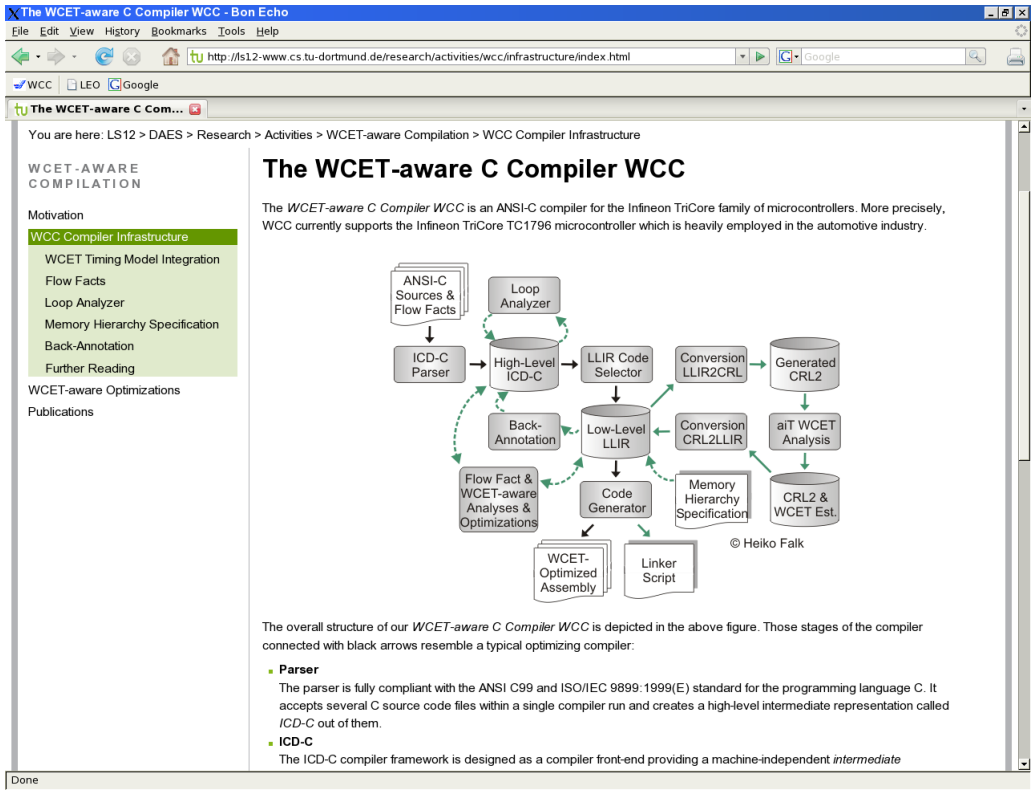
- Analysis results of different approaches are **remarkably different** even for basic systems
- Analysis accuracy and time depend highly on specific system characteristics
- Holistic approaches: better for correlations among task activations, worse for large jitter of actual releases.
- Modular approaches: worse for cyclic dependencies (giving correlations)
- The quest for “optimal” abstraction far from solved!!!

WCET-Aware Compiler *WCC*



- *WCC* for Infineon TriCore 1.3, including ICD-C & LLIR Irs
- Integrates static WCET analyzer aiT from AbsInt

Website on WCET-aware Compilation



- ✓ Visibility by Google:

  “WCET aware compilation”:
  First 10 hits all related to UDORT, website on place #1

  “WCC compiler”:
  First 3 hits related to UDORT, website on place #1

- ✓ Setup of completely new website:

  `http://ls12-www.cs.tu-dortmund.de/research/activities/wcc`

WCC's Static Loop Analyzer

- **Goal**
  - Automatic loop bound determination for WCET analysis
- **Strategy**
  - Perform static program analysis using Abstract Interpretation, polyhedral models and Program Slicing
  - Apply analysis at ICD-C level, i.e. at source code level
- **Results**
  - 98% of all loops of 93 representative benchmarks successfully analysed
  - Analysis results of superior quality, better than competitors
  - Publications at CGO 09 (to appear) and Report on Artist2 WCET Tool Challenge 2008

WCET-Aware Procedure Positioning

- **Goal**
  - Change order of functions such that worst-case I-cache performance is improved
- **Strategy**
  - Group any 2 functions calling each other frequently close to each other in memory, consider WCET data
- **Results**
  - WCET-aware Procedure Cloning: 37% WCET reduction
  - WCET-aware Procedure Cloning & Positioning: 42% WCET reduction
  - Publications at SCOPES 2008 and ECRTS 2008

WCET-Aware Scratchpad Memory Allocation

- **Goal**
  - Move global data, basic blocks and functions to data and code scratchpad memories *(SPMs)*
- **Strategy**
  - Minimize WCET via integrated ILP formulation, inherent consideration of worst-case execution path *(WCEP)*
- **Results**
  - Static SPM allocation of data & code work
  - Data SPM allocation: up to 20% WCET reduction for 14 benchmarks
  - Code SPM allocation: up to 40% WCET reduction for 73 benchmarks
  - Paper submitted to DAC 09

WCET-Aware Register Allocation

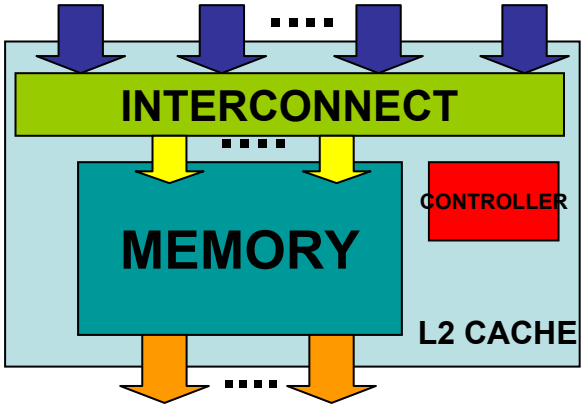
- **Goal**
  - Extend state-of-the-art graph coloring towards avoiding spill code generation along WCEP
- **Strategy**
  - Re-evaluate current WCET after register allocation of each basic block
- **Results**
  - WCET-aware register allocation fully operational
  - 27% WCET reduction on average for 27 benchmarks
  - Paper submitted to DAC 09

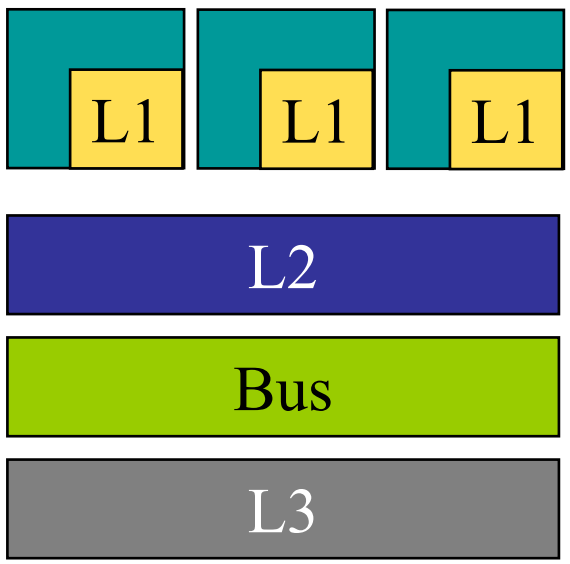
Trade-offs analysis of L2 on-chip cache architectures for embedded MPSoCs

- On-chip memory organization is one of the most important aspects
  - It can influence the overall system behavior in MPSoCs
  - It must provide the required bandwidth to the software tasks
  - Under chip area, power consumption and cost constraints
  - Predictability
- While there is general consensus on L1 private cache organization, for L2 there is still not a dominant paradigm!

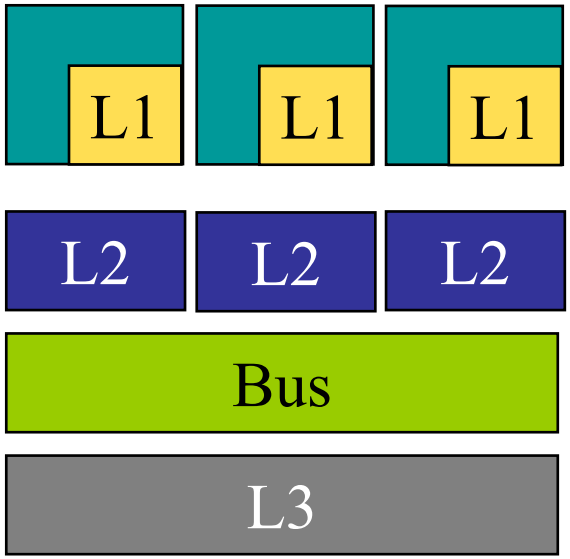
L2 cache modules

- **The High level template made up of:**
  - Configurable number of ports
  - Interconnect
  - Memory
  - Bus master ports
  - Cache controller logic
- **Different available configurations**

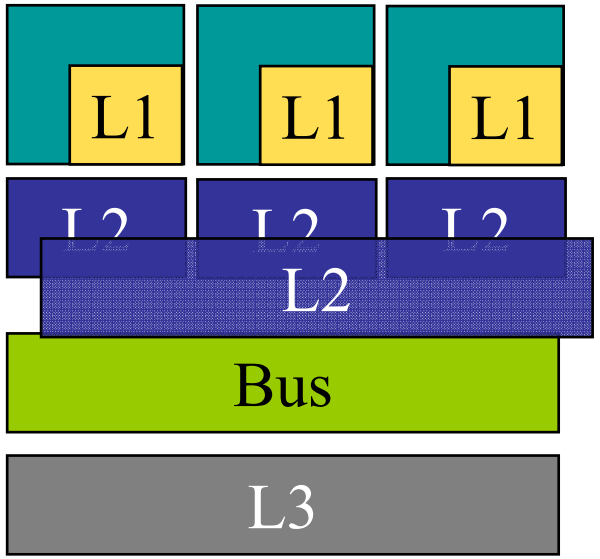




Shared



Private



Hybrid

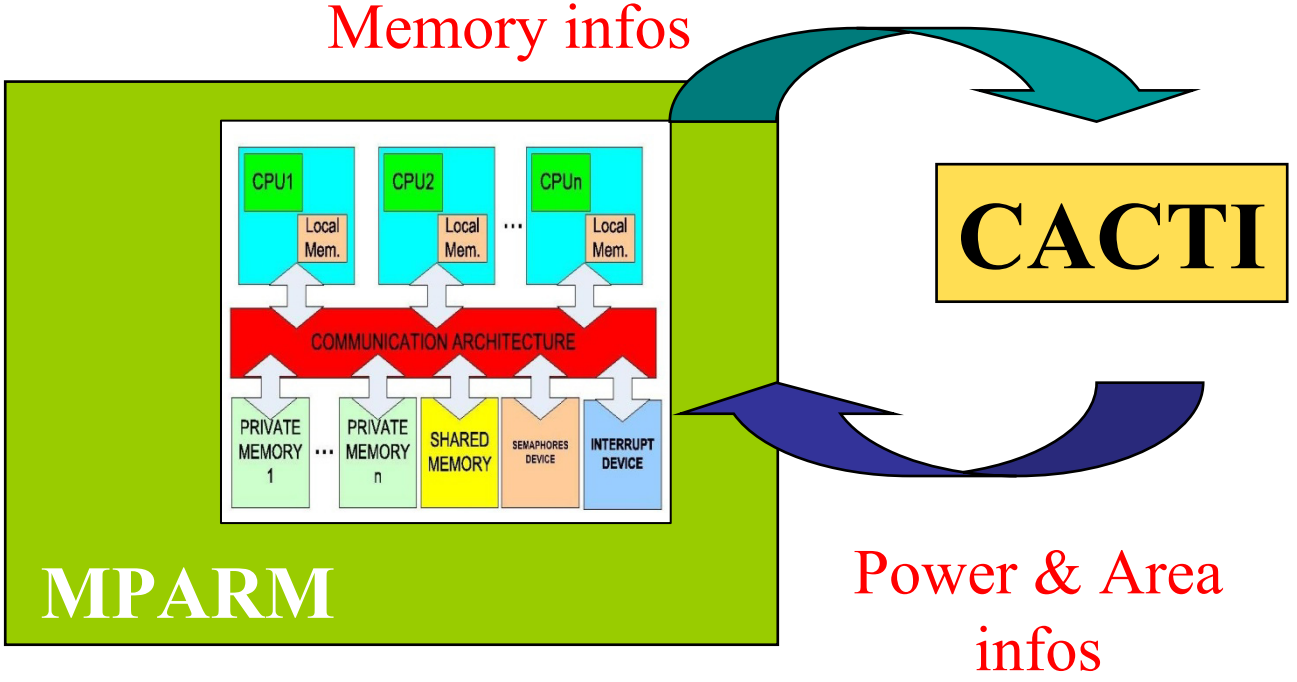
Memory Power Modeling

INPUTS:

- Sizes
- Technology
- Associativity
- Number of ports
- Port width
- Tag size
- Type of cache
- Number of banks
- Operating temperature

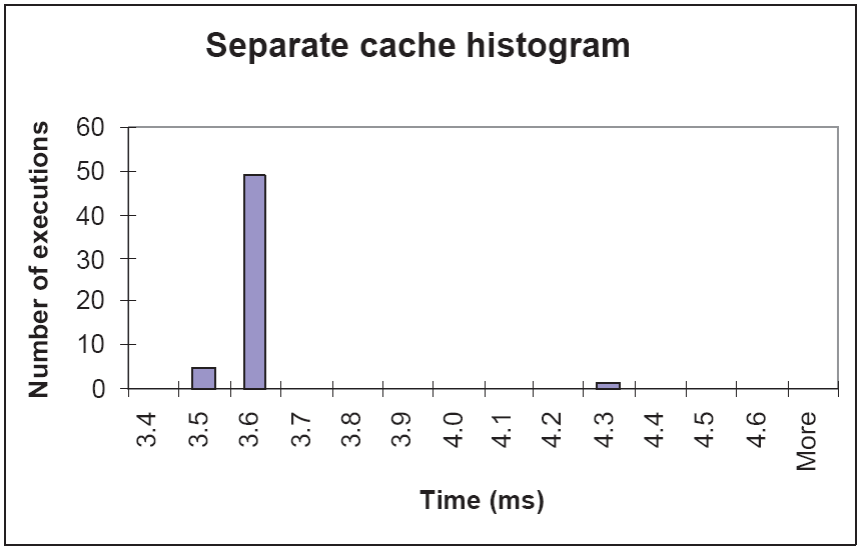
OUTPUTS:

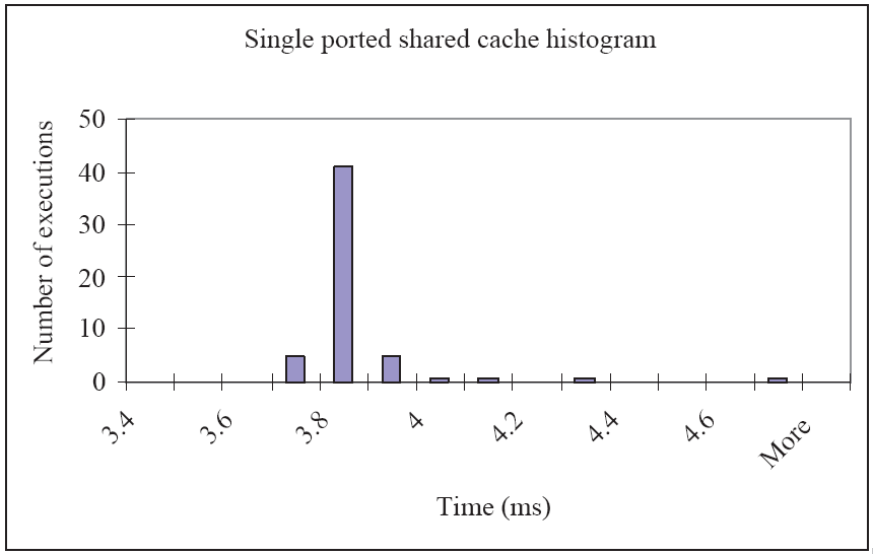
- Total area,
- Dynamic power
- Leakage power

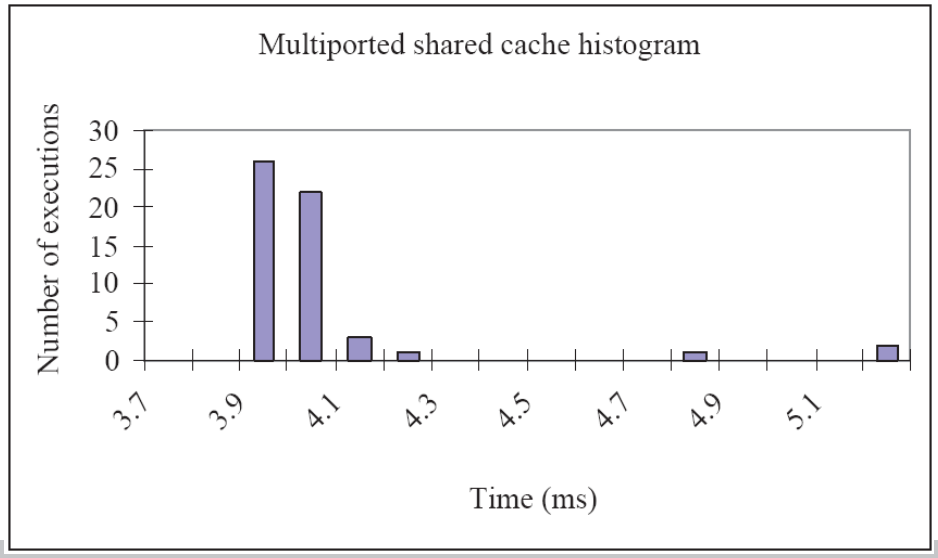


Results on Predictability (as variance)

- The separate L2 cache has lower variance
  - Some minorities: due to cold cache effect at the first run
- The shared caches have higher variance







Joint Meetings and Visits

- **Timing analysis, scheduling and architecture** *Pisa; Oct. 2-3, 2008*
- **Mapping of applications for MPSoCs** *Düsseldorf, Germany; Nov. 27-28, 2008*
- **Visit DTU -> Linköping** *Several visits during 2008*
  Common research on predictable fault tolerant systems.
- **Visit Bologna -> Linköping** *Jan.-April 2008*
  Synthesis of bus controllers for predictable MPSoC
- **Visit Dortmund -> ETHZ,** *Nov. 2008*
  Modeling memory accesses in MPA
- **Casteness 2008 Workshop,** *Rome, Jan. 15-18, 2008:*
- Many short visits for collaboration

KeyNotes

- **DATE 2008** (Petru Eles, Linköping)
- **Royal Society Meeting** (Tom Henzinger, EPFL)
- **CASA08** (Peter Marwedel, Dortmund)
- **MPSoC Conference 2008** (L. Thiele, ETHZ)
- **Mapping Algorithms to MPSoC, Workshop, June 2008** (L. Thiele, ETHZ)

Summer Schools and Tutorials

- **Summer Schools**
  - *Florianopolis, Brasil, August, 2008.*(several speakers),
  - *ARTIST Summer School, Sept 2008.* (several speakers),
  - *ARTIST Summer School, Shanghai 2008.* (M. Gonzalez-Harbour, Cantabria)
- **Tutorials**
  - **Timing Analysis and Timing Predictability (**R. Wilhelm, Saarland) *Embedded Networked Systems: Theory and Application, Heraklion, Crete – July 21–25, 2008*
  - **Abstract Interpretation with Applications to Timing Validation (**R. Wilhelm, Saarland) *Princeton, July 7–14, 2008*
  - **MPARM tutorial** (M. Ruggiero, Bologna) *Pisa, Nov 5-6, 2008*

Tools and Platforms

- **AiT**, the leading tool for computing WCETs
  - [AbsInt, Dortmund, Saarland]
- **MAST,** Modeling and Analysis Suite for Real-Time Applications [Cantabria]
- **MPA toolbox,** analysis of distributed embedded real-time systems, based on the real-time calculus [ETHZ]
- **MPARM,** virtual SoC platform, written in SystemC, to model system HW and SW [Bologna]
- **UPPAAL,** leading tool for precise automata-based analysis of timed systems [Uppsala, Aalborg]

Plans for Y2

Continued transversal integration

- More compositional, scalable, and precise analysis techniques
- Resource interfaces to specify resource requirements and constraints.
- Principles for the design of software with analyzable timing behaviour.
- Design predictable and performant multi-core architecture
- Fault-tolerance

Global event

Whitepaper