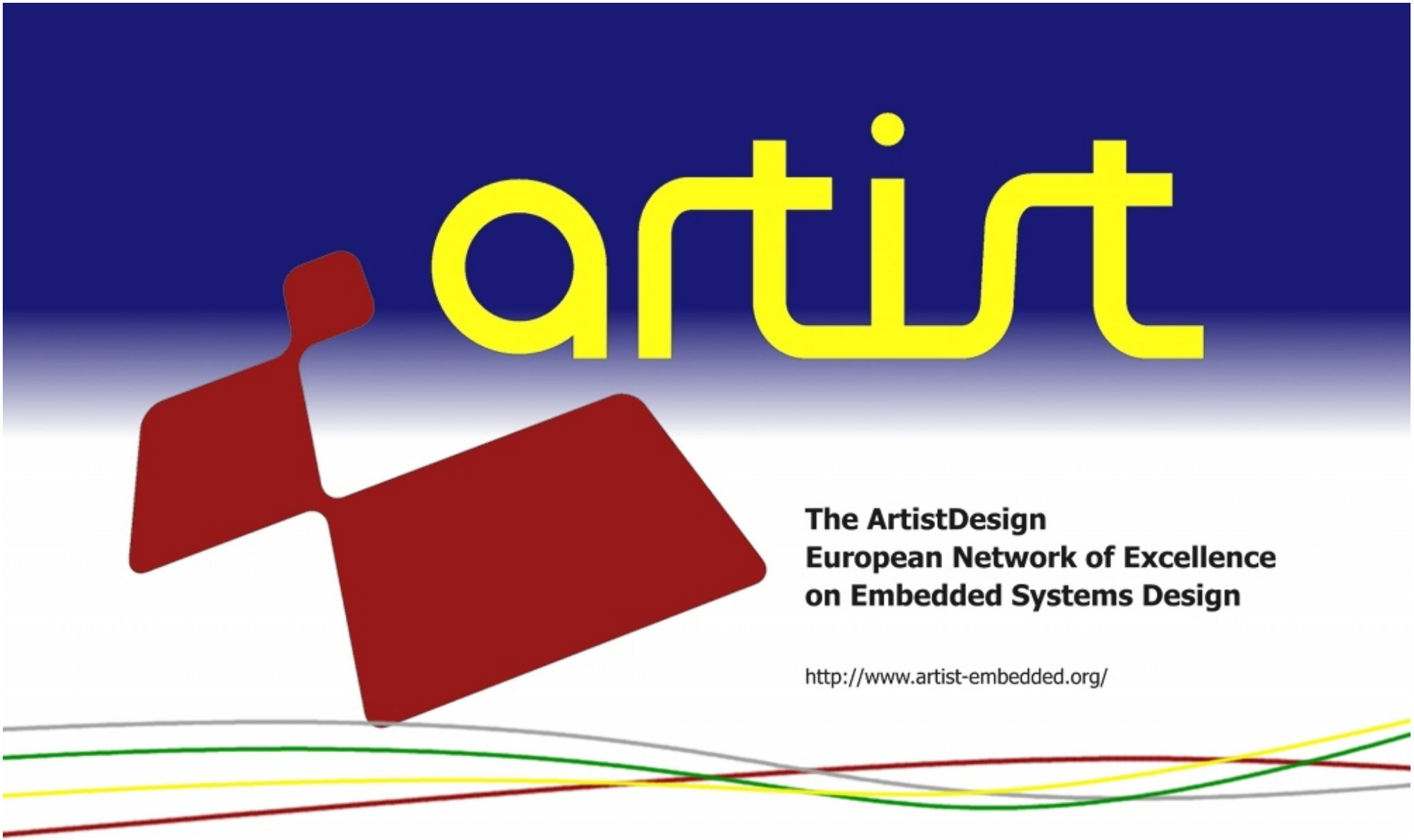


Showcase of the Main Results

DATE Conference, March 15th, 2012

SEVENTH FRAMEWORK PROGRAMME

ArtistDesign European NoE: Showcase of the Main Results

DATE Conference, March 15th, 2012

Achievements and Perspectives :

Software Synthesis, Code Generation and Timing Analysis

leader : Peter Marwedel

TU Dortmund

Scope of the cluster

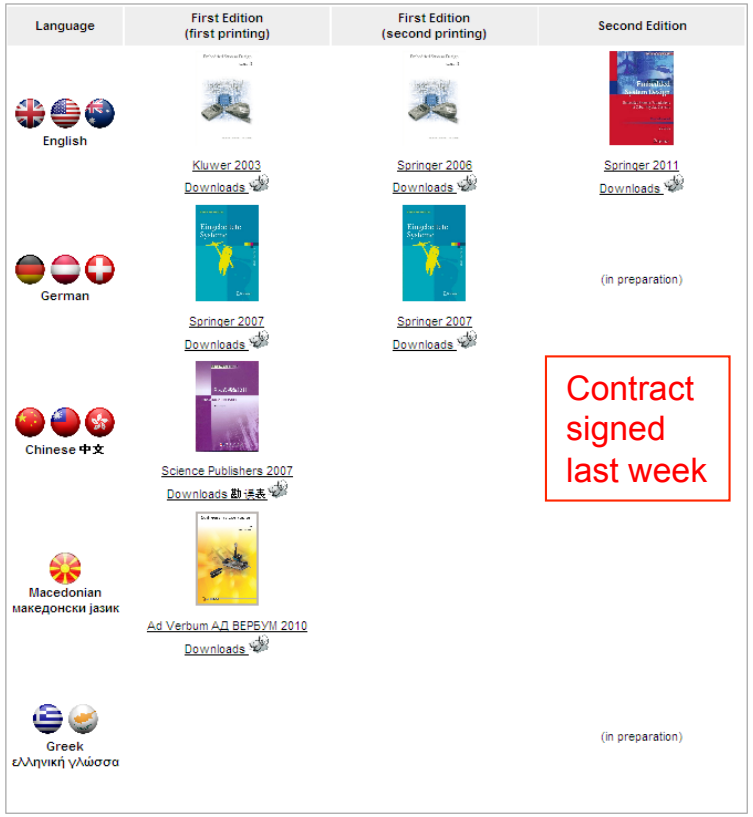
- Contributions to global activities
  (Education, spreading excellence, transversal clusters)
- Code generation:
  All types of generation of executable code from standard (frequently imperative) languages
- Linking timing analysis and compilers
- Software Synthesis:
  Generation of software from higher level specifications, e.g. in MATLAB or UML
- Timing analysis
  Computation of safe bounds on the execution time

Contributions toward education

- Organization of the workshop on embedded system education (WESE)
  - Published in the ACM Digital Library
  - Main forum for embedded system education
  - Will be continued beyond the end of the NoE
- Contribution at summer schools
  - Summer schools in China (2x), Brazil, Morocco, Europe
  - Will be continued beyond the end of the NoE
- Joint teaching
  E.g. at ALARI, Lugano
- Text book on embedded systems …

Embedded System Text Book

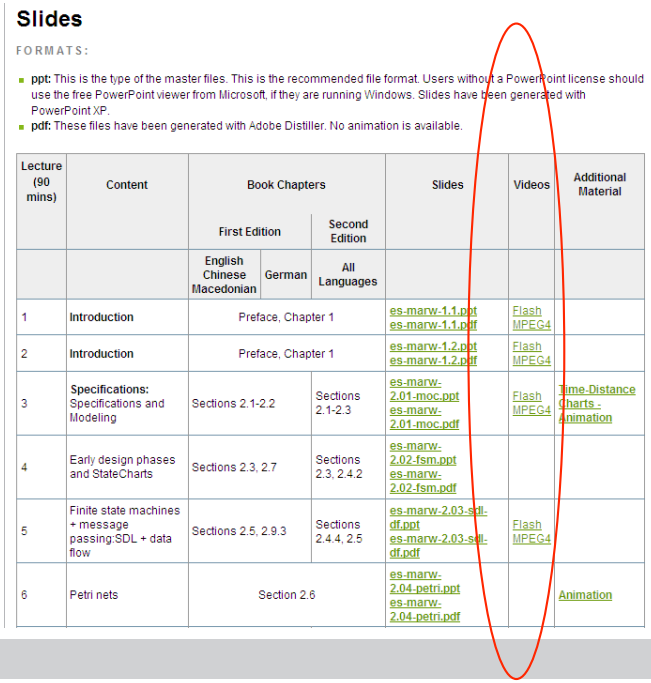
- Textbook, slides, video recorded lectures (http://ls12-www.cs.tu-dortmund. de/daes/daes/mitarbeiter/prof-dr-peter-marwedel/embedded-system-text-book/slides/slides-2011.html)



Language	First Edition (first printing)	First Edition (second printing)	Second Edition
English	Kluwer 2003 Downloads	Springer 2006 Downloads	Springer 2011 Downloads
German	Springer 2007 Downloads	Springer 2007 Downloads	(in preparation)
Chinese 中文	Science Publishers 2007 Downloads 勘误表		
Macedonian македонски јазик	Ad Verbum АД ВЕРБУМ 2010 Downloads		
Greek ελληνική γλώσσα			(in preparation)

Contract signed last week

Downloads: ~1400
Copies: 500



Slides

FORMATS:

- **ppt:** This is the type of the master files. This is the recommended file format. Users without a PowerPoint license should use the free PowerPoint viewer from Microsoft, if they are running Windows. Slides have been generated with PowerPoint XP.
- **pdf:** These files have been generated with Adobe Distiller. No animation is available.

Lecture (90 mins)	Content	Book Chapters			Slides	Videos	Additional Material
		First Edition		Second Edition			
		English Chinese Macedonian	German	All Languages			
1	Introduction	Preface, Chapter 1			es-marw-1.1.ppt es-marw-1.1.pdf	Flash MPEG4	
2	Introduction	Preface, Chapter 1			es-marw-1.2.ppt es-marw-1.2.pdf	Flash MPEG4	
3	Specifications: Specifications and Modeling	Sections 2.1-2.2		Sections 2.1-2.3	es-marw-2.01-moc.ppt es-marw-2.01-moc.pdf	Flash MPEG4	Time-Distance Charts - Animation
4	Early design phases and StateCharts	Sections 2.3, 2.7		Sections 2.3, 2.4.2	es-marw-2.02-fsm.ppt es-marw-2.02-fsm.pdf		
5	Finite state machines + message passing:SDL + data flow	Sections 2.5, 2.9.3		Sections 2.4.4, 2.5	es-marw-2.03-sdl-df.ppt es-marw-2.03-sdl-df.pdf	Flash MPEG4	
6	Petri nets	Section 2.6			es-marw-2.04-petri.ppt es-marw-2.04-petri.pdf		Animation

Scope of the cluster

- Contributions to global activities
  (Education, spreading excellence, transversal clusters)
- Code generation:
  All types of generation of executable code from standard (frequently imperative) languages
- Linking timing analysis and compilers
- Software Synthesis:
  Generation of software from higher level specifications, e.g. in MATLAB or UML
- Timing analysis
  Computation of safe bounds on the execution time

Target Platforms

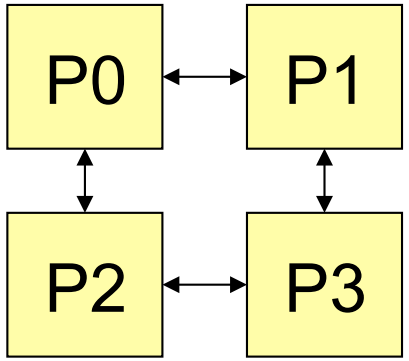
- Trend toward increased performance requirements for systems, in particular, embedded systems
- Due to power and thermal constraints, single processors cannot provide the required performance

→ Multi-processors have to be used

- For embedded systems, they are usually integrated on one chip

→ Multi-processor systems on a chip (MPSoCs) are the target for design processes

→ Techniques for mapping applications to MPSoC urgently needed



A Simple Classification

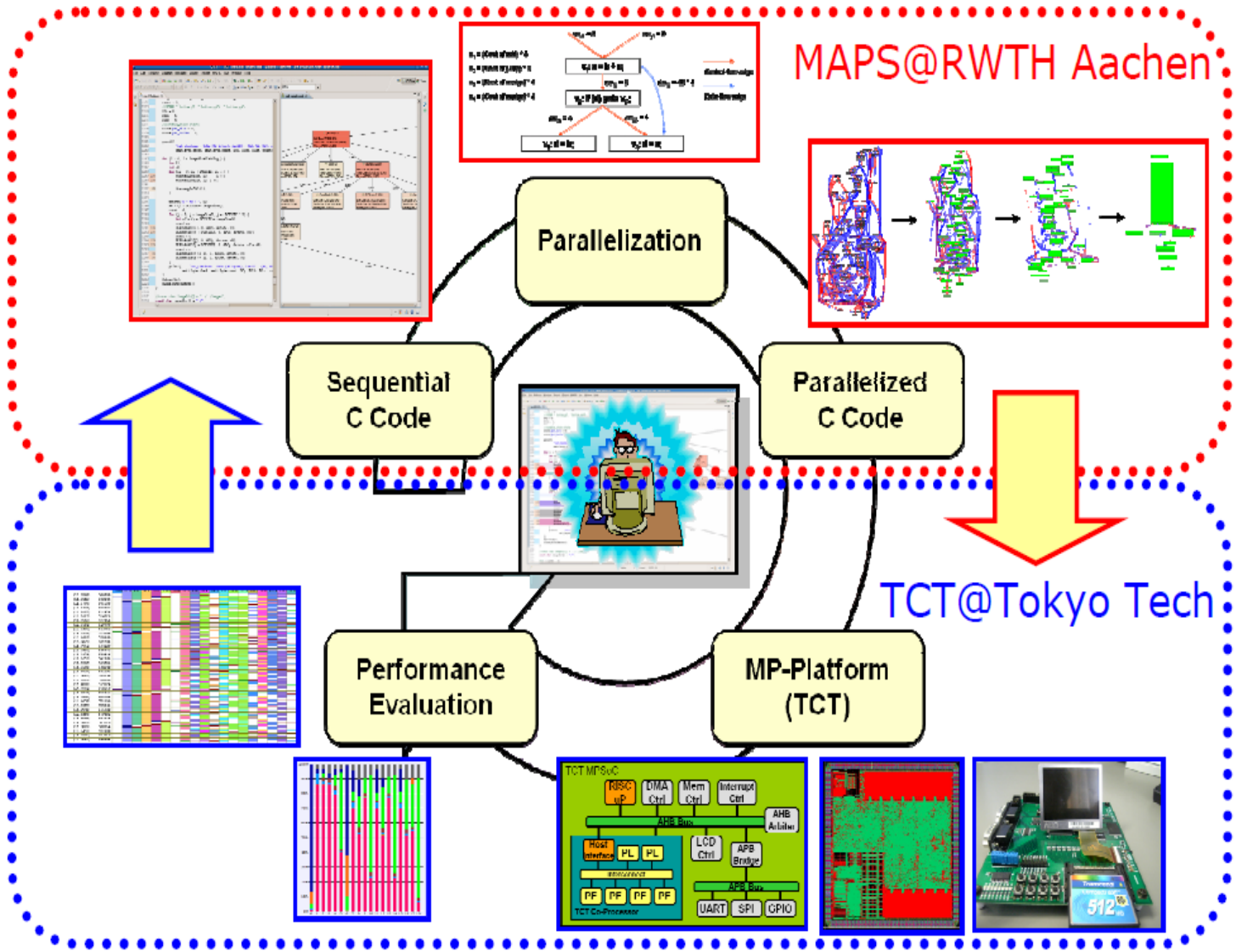
Architecture fixed/ Auto-parallelizing	Fixed Architecture	Architecture to be designed
Starting from given task graph	Map to CELL, Hopes (SNU), Qiang XU (HK), Simunic (UCSD)	DOL, SystemCodesigner
Auto-parallelizing	Mnemee (Dortmund) Franke (Edinburgh) MAPS	Daedalus

MAPS

Several tools for mapping applications to MPSoCs have become available

- MAPS (RWTH Aachen)
  Huawei has consulted RWTH Aachen to develop a 3-years technology roadmap on their MPSoC programming flow. The roadmap was tailored to future directions for Huawei wireless products.

MAPS-TCT Framework



Rainer Leupers, Weihua Sheng: MAPS: An Integrated Framework for MPSoC Application Parallelization, *1st Workshop on Mapping of Applications to MPSoCs, Rheinfels Castle*, 2008

Software Analysis & Transformation

IMEC

- MH parallelization assistant

U. Passau

- Establishment of the polyhedron model for loop parallelization with several entries in the Encyclopedia of Parallel Computing, September 2011
- First steps of making the polyhedron model multicore-ready (polly.llvm.org)
- Moving the polyhedron model further towards practical embedded systems

Scope of the cluster

- Contributions to global activities
  (Education, spreading excellence, transversal clusters)
- Code generation:
  All types of generation of executable code from standard (frequently imperative) languages
- ☞ Linking timing analysis and compilers
- Software Synthesis:
  Generation of software from higher level specifications, e.g. in MATLAB or UML
- Timing analysis
  Computation of safe bounds on the execution time

Reconciling compilers and timing analysis

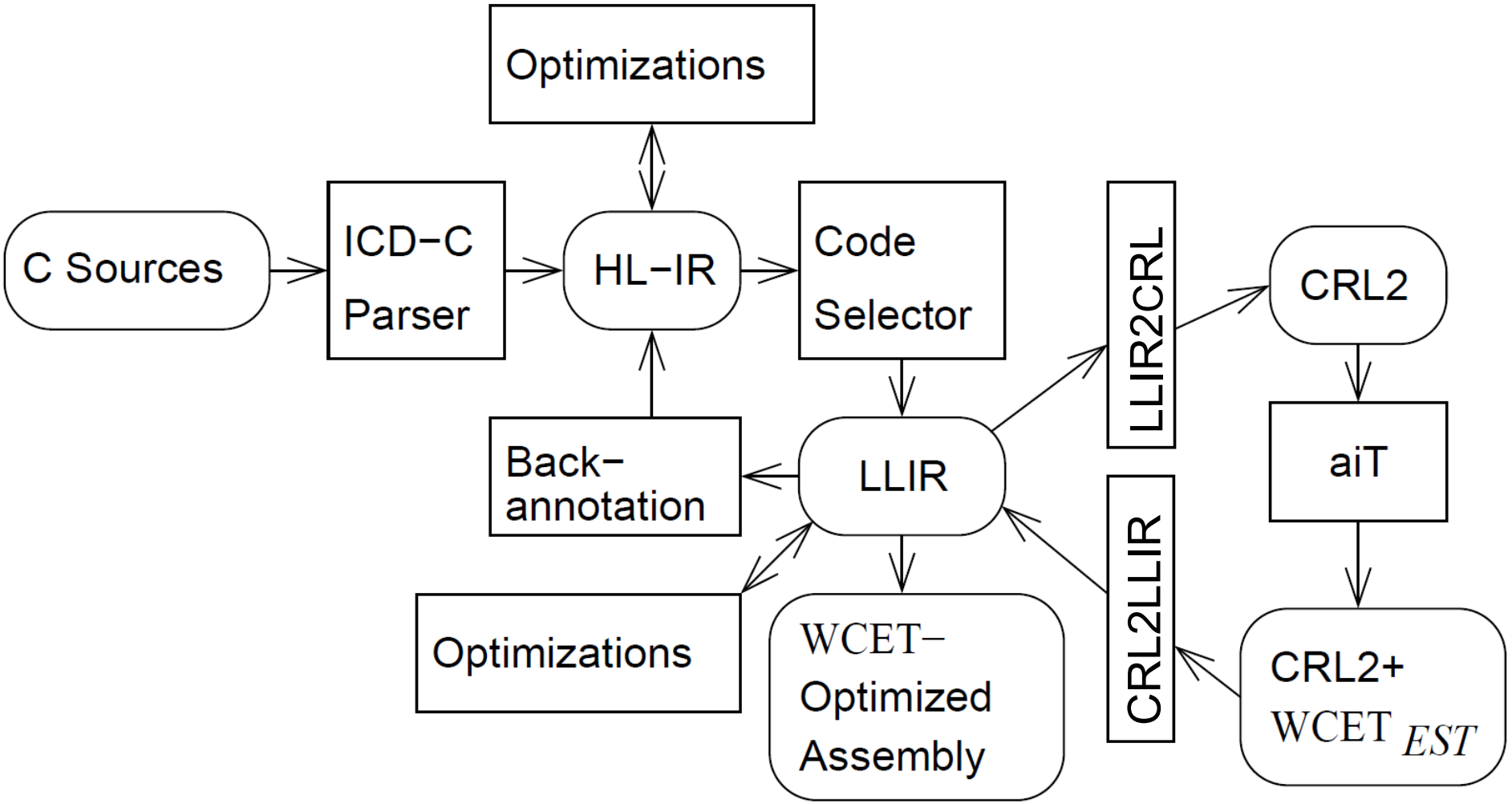
Compilers mostly unaware of execution times $WCET_{EST}$

- Execution times are checked in a “trial-and-error” loop:
  {try: compile – run – check – error: change}*

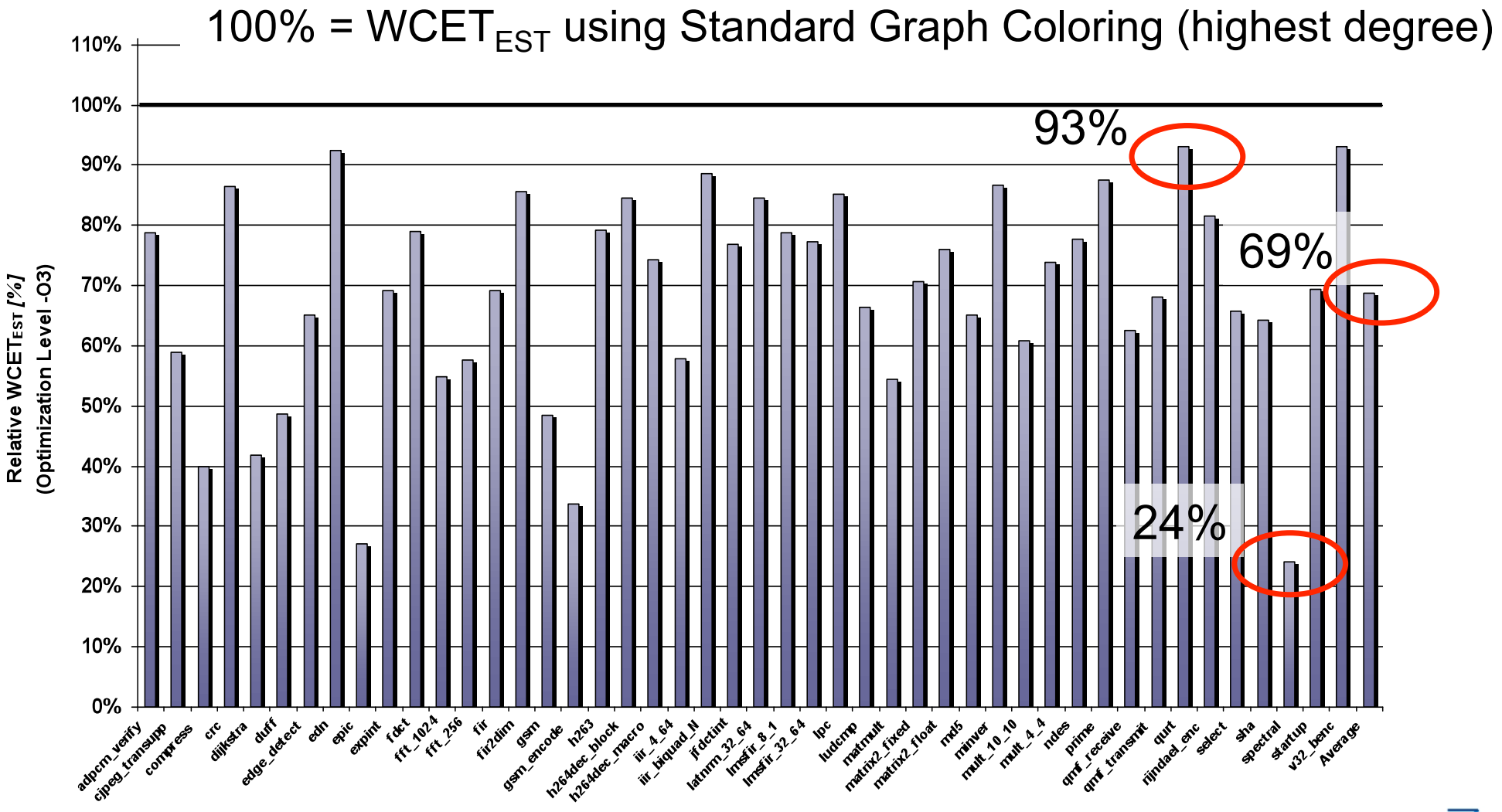
☞ Integration of safe, static timing analysis into compiler

- Getting rid of loops (if everything works well)
- Potential for optimizing for the WCET

Structure of WCC (WCET-aware C-compiler)



Register Allocation

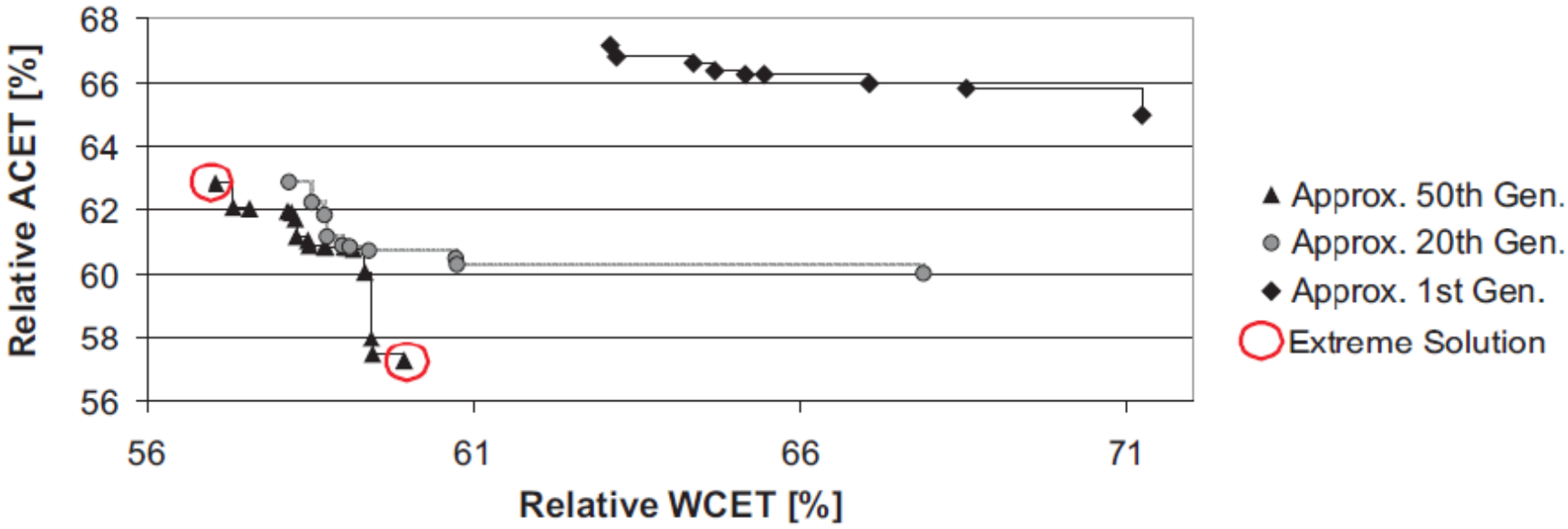


[H. Falk]

WCET-aware SPM allocation

- Setup
  - Bosch Democar: Runnable: IgnitionSWCSync
    Part of task actuator
    activated every 90° of crankshaft angle ☞ time-critical
  - WCET-aware SPM allocation of program code by WCC, including fully automated WCET analyses using aiT and solution of the ILP for SPM allocation
  - WCET reduced to about 50%, compared to gcc.
- ☞ PREDATOR project partners Bosch & Airbus interested in WCC

Timing Analysis and Compiler Techniques



Automatic Pareto-optimal trade-off between WCET, ACET (and code size)

- ***Result:*** trade-off WCET ↔ ACET reveals that (for the considered standard optimizations) WCET performs similar to ACET; achieved WCET and ACET reductions are very similar

Scope of the cluster

- Contributions to global activities
  (Education, spreading excellence, transversal clusters)
- Code generation:
  All types of generation of executable code from standard (frequently imperative) languages
- Linking timing analysis and compilers
- ☞ Software Synthesis:
  Generation of software from higher level specifications, e.g. in MATLAB or UML
- Timing analysis
  Computation of safe bounds on the execution time

Software Synthesis

1 call for a special journal issue

3 Workshops on Software Synthesis;
program of the 3rd workshop:

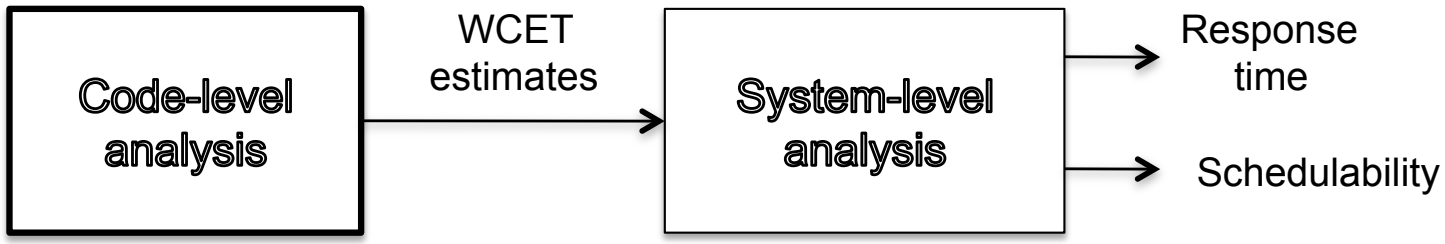
- Shuvra Bhattacharyya:
  Software Synthesis from Dataflow Graphs: State of the Art and Emerging Trends
- Kaushik Ravindran and Hugo Andrade:
  From Streaming Models to Hardware and Software Implementations
- Marco di Natale:
  From analysis to optimization in the deployment of real-time distributed functions in modern automotive systems
- Rajeev Alur: Interfaces for control components
- Nicolas Halbwachs:

Scope of the cluster

- Contributions to global activities
  (Education, spreading excellence, transversal clusters)
- Code generation:
  All types of generation of executable code from standard (frequently imperative) languages
- Linking timing analysis and compilers
- Software Synthesis:
  Generation of software from higher level specifications, e.g. in MATLAB or UML
- ☞ Timing analysis
  Computation of safe bounds on the execution time

Timing Analysis Scientific Highlights

- Timing analysis divided into code- and system level
- We deal with *code level*: estimate longest execution time of code (WCET)



- Safe bounds essential for verification of hard RT systems (automotive, avionics, ...)
- Unsafe bounds can also be acceptable in some situations (soft RT)

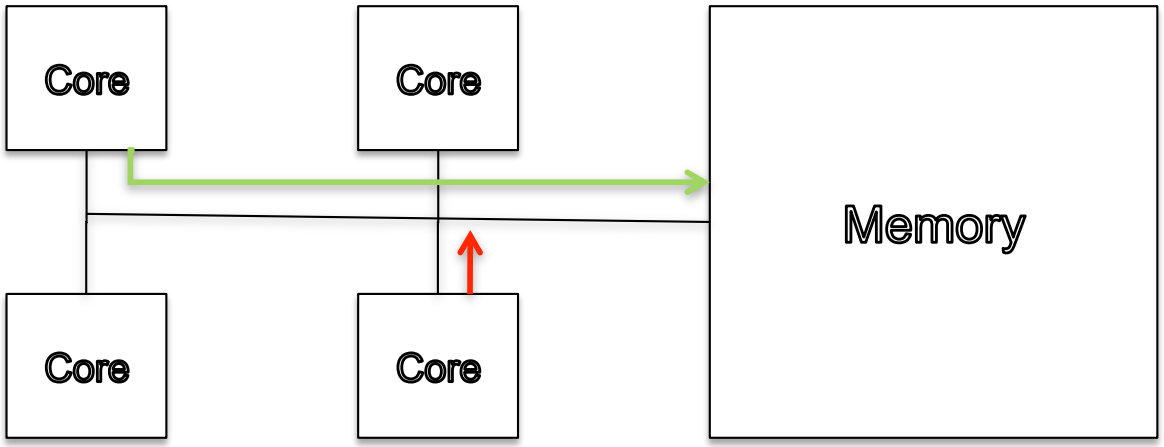
Challenges for WCET analysis

- WCET depends on:
  - Program code
  - Hardware
  - Runtime environment (input data, concurrent activities, ...)
- Challenges:
  - Rapid development in HW architecture:
    - Multicore/MPSoC, parallel architectures
    - Complex processors & memory systems: superscalar, cache, ...
  - SW: complex code, many abstraction layers, unpredictable program flow

Achievements in the network

- Foundational issues
- Design principles for timing-predictable systems
- Advanced analysis methods

The multicore timing predictability problem

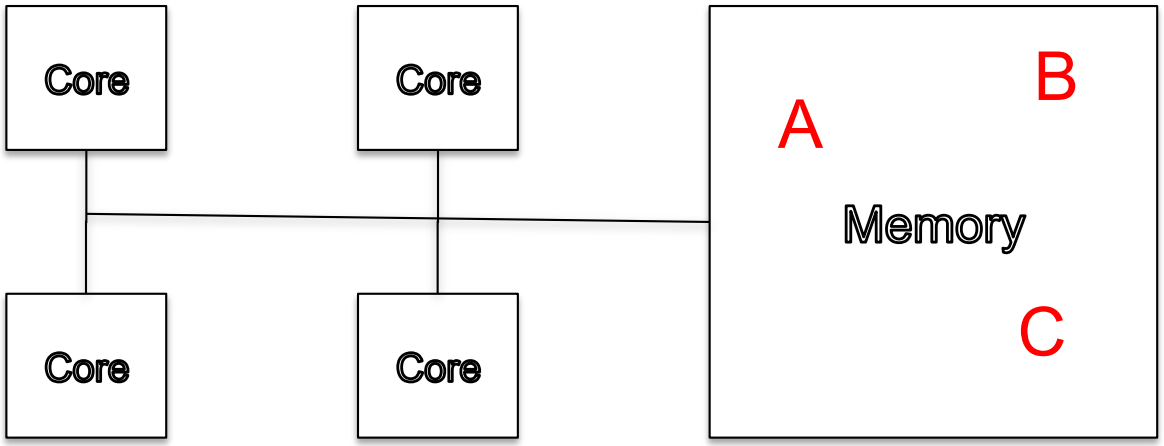


- Competition for shared resources makes access times dependent on all other possible concurrent accesses
- Timing composability is lost. Very detrimental to timing predictability

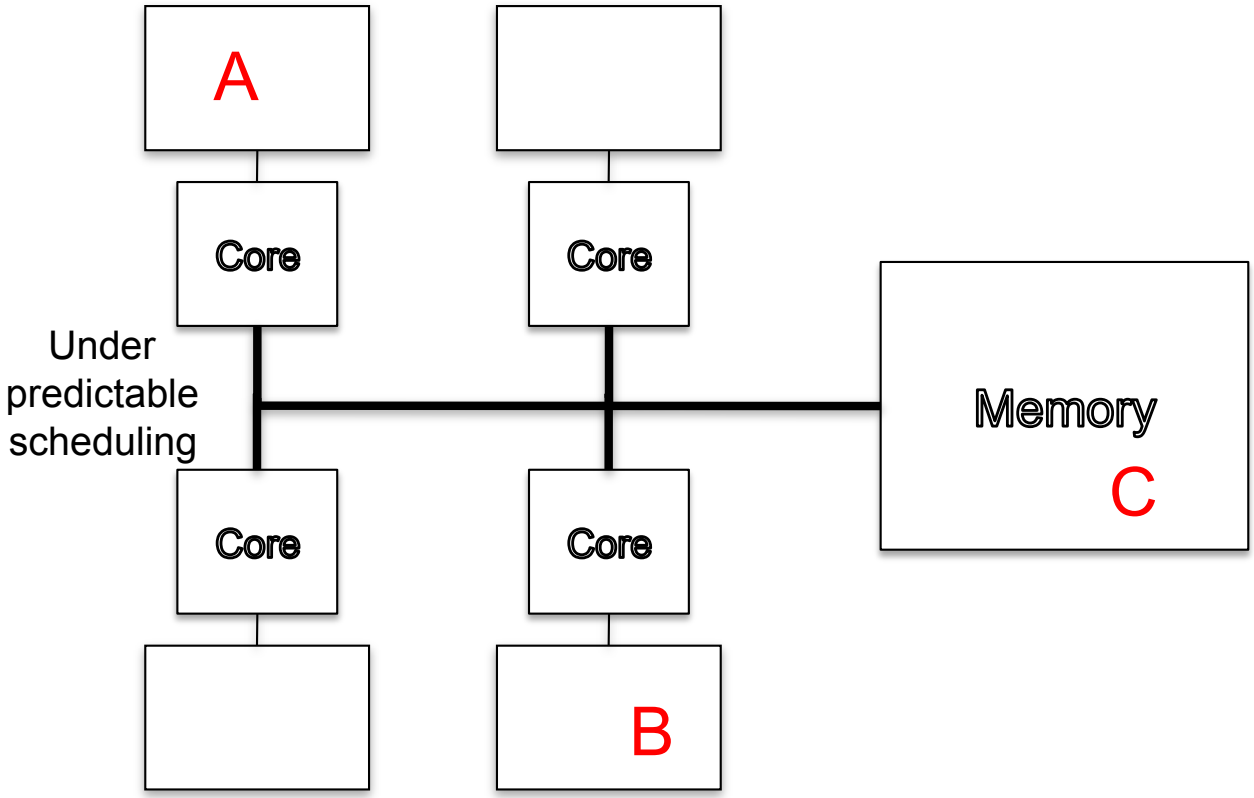
Design principles

- **PROMPT design principles** for predictable HW/SW parallel architectures
- **Main idea**: avoid sharing data and resources whenever possible
- **SW localization**: localize data. *Avoid shared data* when not strictly needed
- **HW localization**: Use *local memories* (scratchpads) for local data. Avoid shared caches
- **Whatever remains shared**: put under *strict scheduling control*. Predictable time slots for all activities (e.g., TDMA)
- **Result**: timing composability is regained. WCET analysis can again be done separately, for each core

Design principles (II)



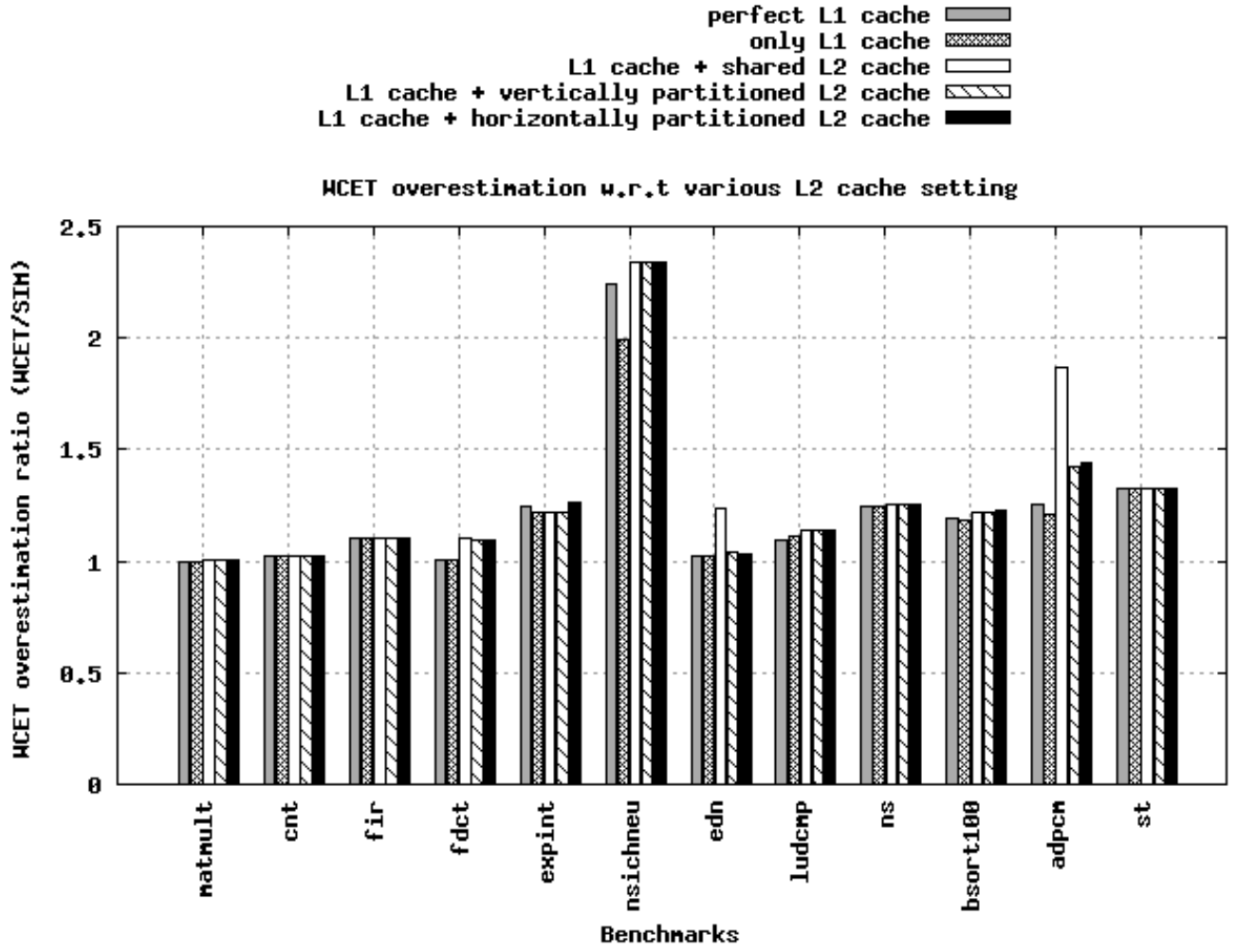
Design principles (III)



WCET analysis framework for multi-cores

- A unified WCET analysis framework for multi-cores
- Assumes TDMA bus scheduling
- Can deal with shared caches
- Can also deal with modern processor core features (exhibiting timing anomalies)

Experimental results



Cache analysis

- Cache analysis by a combination of abstract interpretation and model checking
- **Cache analysis**: classify memory accesses as hit/miss/unknown, needed for WCET analysis
- **Abstract interpretation** (AI) yields a scalable but somewhat imprecise analysis
- **Model checking** (MC) can give potentially very precise results, but has scalability problems
- **Idea**: First analyze by AI, then refine imprecise parts with MC. Keep complexity of MC under control by restricting its use to where it makes a difference
- Experiments show very good improvements of resulting WCET estimate

Predictability of caches

- Fundamental results regarding predictability of cache replacement strategies
- Introduced notion of "cache sensitivity" (lasting influence of initial cache state on cache hits/misses)
- Computed this for LRU, FIFO, PLRU, MRU
- Technique: model checking over automaton describing pairs of possible cache states, and transitions between pairs for same memory accesses:
  (c1,c'1) → (c2,c'2) → (c3,c'3) → ….
- Sensitivity ratios can be computed for paths in this transitivity system

Results

- For LRU, no lasting differences in cache hit/miss ratio
- For FIFO, PLRU, MRU there are sequences of cache states where hit/miss ratios also in "steady state" differ by a factor > 1
- Implies that difference in execution time can be big due only to initial cache state
- Makes it harder to estimate WCET from measurements, as it becomes very important to find the "worst" initial cache state
- The notion of cache-related preemption delay becomes very dubious for these replacement policies

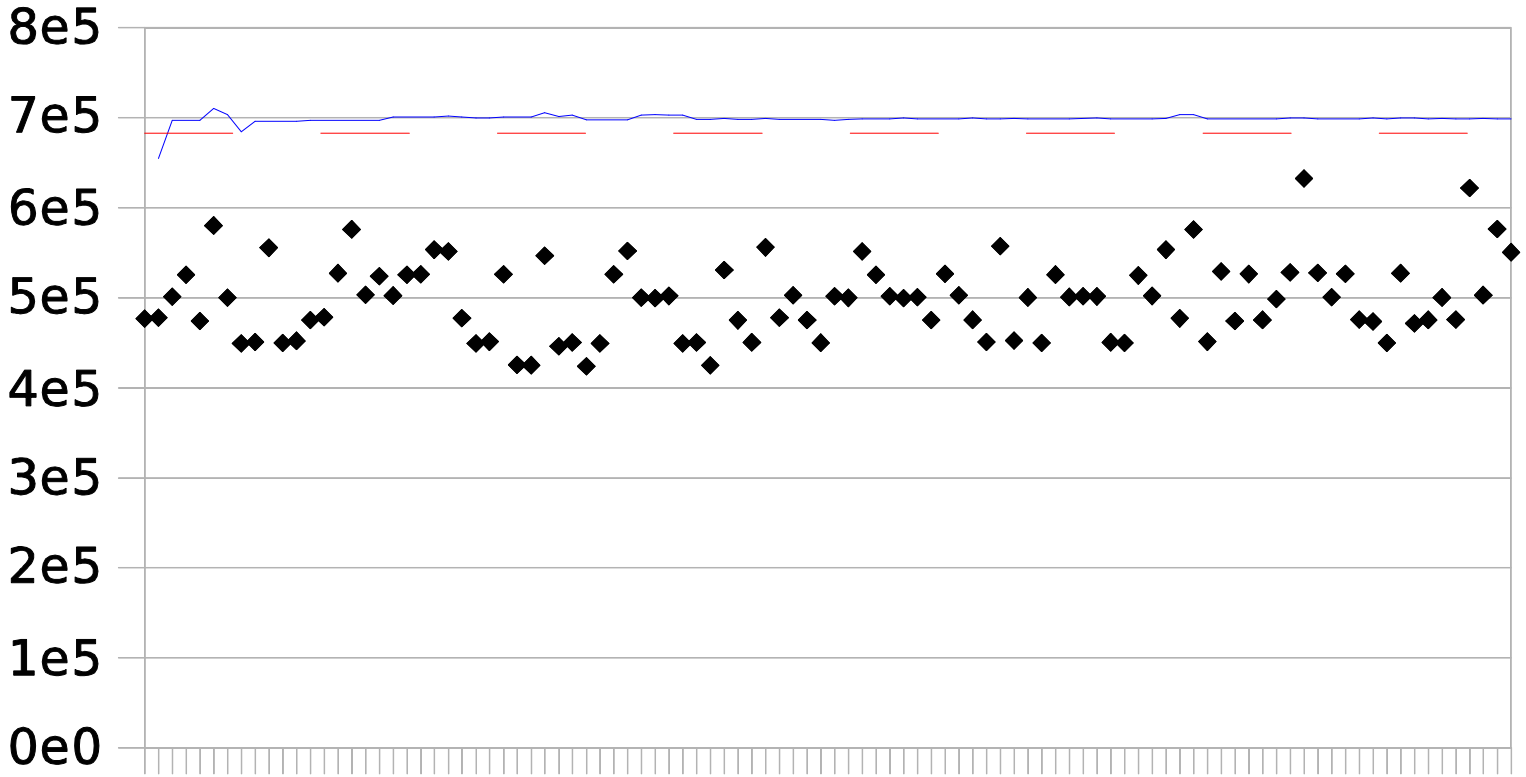
Hybrid WCET analysis

- Combines measurements with static analysis
- Replaces difficult microarchitectural modelling with measurements of time
- Unsafe in general, but may be acceptable depending on requirements
- Will typically measure time for small program fragments (basic block level), and then use static analysis techniques to produce WCET estimate
- Fine-grained measurements are problematic: large probe effects, hard to record measured data in real-time, ...

Model identification for hybrid WCET analysis

- A method to estimate basic-block execution times from end-to-end measurements
- Based on a linear timing model. Uses model identification by linear programming. Resulting model will never underestimate any observed execution times
- Allows for context-sensitive basic block execution times in analysis (increases precision)
- Evaluation shows WCET overestimation in range 0-10% on suite of benchmarks

Example



Conclusions

Mapping of applications to MPSoCs

- Turned an empty landscape into one with several tools with a different focus: MAPS, DOL, SystemCoDesigner, HOPES

Software Synthesis

- Linked experts, published papers

Efficient Design: Energy-awareness etc.

Timing analysis:

Many new insights in timing analysis for multi-core, cache, timing predictability, identification of timing models