Year 3 Review
Brussels, February 24th, 2011

Cluster

Achievements and Perspectives:

Software Synthesis, Code Generation
and Timing Analysis

leader: Peter Marwedel
TU Dortmund
High-Level Objectives

- Limitations of increasing clock speeds any further
  - focus on using MP platforms, in particular MPSoCs
    - increased importance since proposal writing, industrially relevant
- Different from multi-core situation: Multiple applications, heterogeneous processors, and multiple objectives
- MP platforms pose threats to timing predictability;
  - develop MP/MPSoC design principles for maximal predictability
  - develop models/methods for timing analysis of parallel software
- Efficient design + software synthesis also in the scope
  (see deliverables; coordination with other projects)
- Partners contribution to transversal clusters
  (e.g. for predictability: WCC)
Building Excellence

- Rheinfels workshop, St. Goar, June 2008/09/10 (Incorporating external experts), TA meetings, Tutorials, summer schools, teaching at ALARI, WESE, Software Synthesis Workshop (2nd year), ...
- 84 publications (14 joint)
- Textbook, slides, video recorded lectures (http://is12-www.cs.tu-dortmund.de/~marwedel/es-book/slides10/)
Overall Assessment and Vision at Y0+3

What went well:

- Extension toward software synthesis
- Abundant amount of results on resource-efficiency
- Successful formation of tool chains for timing analysis
- New timing-predictable architectures
- Energy efficiency issues beyond processors
- While ArtistDesign partners acted as a core, the involvement of researchers went far beyond ArtistDesign
- Extension of well-visible workshop
- Results from mapping tools
Scientific Highlights: Mapping to MPSOCS

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.

- MPMH/Mnemee-Integration (IMEC, Dortmund, Eindhoven, Athens)

- SystemCoDesigner (U. Erlangen-Nuremberg)

- Daedalus (U. Leiden)

- HOPES (Seoul National University)
Timing Analysis and Compiler Techniques

WCET-aware C Compiler WCC: Overall structure

- **Tight integration of WCET analyzer** aiT into compiler backend

- **WCET timing data** available in compiler: WCET of program, function, basic blocks; worst-case execution frequencies; cache hits/misses; ...

- **Flow facts** required by aiT can be specified directly in the source code or auto-generated by Loop Analyzer; they are kept correct during all compiler optimizations.

- **Back-Annotation** makes WCET timing data available for high-level optimization

- Properties of **memory hierarchies** (e.g. cache sizes, associativity, scratchpad sizes, ...) can be specified and are respected during optimization and WCET analysis.

- **WCET-aware code optimization** at two abstraction levels: high-level optimization of ANSI-C code (ICD–C), and low-level optimization of assembly code (LLIR)
WCET-aware superblock optimizations

- **Superblocks**: well-known concept from compilers dating back to 1991
- **WCC’s superblocks**: are proposed for the very first time at high-level ANSI-C code, and rely on WCET timing data
- **WCET-aware superblock optimizations**: Standard compiler optimizations (Common Subexpression Elimination (CSE) and Dead Code Elimination (DCE)) were ported to WCC’s superblocks

Superblock activity received 3 awards
Timing Analysis and Compiler Techniques

WCET-aware, ILP-based register allocation

- **Register allocation**: considered the most important compiler optimization since exploitation of processor registers is crucial to achieve high performance.
- **Problem**: current register allocators are unaware of a program’s timing and have no idea of the impact of spill code on WCETs.
- **Year 2**: WCET-aware graph coloring leads to large WCET reductions, compared to traditional graph coloring.
- **Now**: first ILP-based WCET-aware register allocator; the cost of spill code in terms of WCET cycles is taken into account.
Timing Analysis and Compiler Techniques

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

- **Motivation:** Compilers apply optimizations in a fixed sequence, with very few opportunities for compiler user to steer optimizations.
- **Problem:** Optimizations can influence each other; sequence matters; different positive/negative impact on different criteria like e.g. WCET or ACET
- **Joint work of Dortmund and ETHZ:** Use evolutionary algorithms to find highly specific sequences of optimizations for aggressive reduction of one criterion; statistical assessment and detection of Pareto-optimal optimization sequences trading one criterion with the other.
- **WCC’s internals:** Turned into an adaptive compiler where optimizations can be applied in any arbitrary sequence; WCC’s adaptive optimizer consisting of 41 standard, WCET-unaware optimizations, coupled with ETHZ’s tool box of evolutionary algorithms (EAs); ETHZ’s evolutionary algorithms coupled with WCET analyzer aiT and cycle-true simulator to get WCET/ACET per EA individual.
Timing Analysis and Compiler Techniques

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

- **Result 1:** trade-off WCET ↔ ACET reveals that (for the considered standard optimizations) WCET performs similar to ACET; achieved WCET and ACET reductions are very similar

- **Result 2:** trade-off between WCET and code size more interesting – aggressive WCET reduction implies huge code size increases;
  Interesting: function inlining and procedure cloning are frequently included in optimization sequences for code size reduction
  Interesting, too: loop unrolling is never applied when optimizing for reduced code size
Scientific Highlights: Timing Predictability

- Cooperation with OS cluster strengthened
- Cache-aware scheduling (U. Saar, SSSA)
- Timing analysis for multi-core systems with caches
- Projects on tradeoffs between real-time issues and reliability supported by Deutsche Forschungsgemeinschaft
- L. Thiele, R. Wilhelm: Architectural Aspects of Deriving Performance Guarantees, ISCA 2010
Scientific Highlights: Early WCET analysis

- WCET analysis typically applied late in the design chain (timing verification)
- If timing requirements are not met, costly redesigns may be needed
- A need to estimate WCET earlier, to direct design decisions
- Will have to give up safety (otherwise too pessimistic estimates)
- Two approaches:
  - Exploring different HW configurations through fast, approximate WCET analysis
  - Approximate source level timing models for WCET analysis
- Initial results indicate that 0-20% deviation is achievable
Scientific Highlights: WCET Challenge

- A tool evaluation event: WCET tools are evaluated on a set of common benchmarks
- Initiated during 2010, evaluation takes place right now
- Nine tools on the starting line (academic & commercial)
- This time, we use real embedded code from Daimler for the evaluation
Joint Technical Meetings (1)

- Course: Retargetable Compilation, Lugano, Switzerland, Feb. 16-19 & Feb 23-25, 2010
- ICT 4 Energy Efficiency, Brussels, Belgium – Feb. 23rd, 2010
- CeBIT, Hannover, Germany – March 9th, 2010
- DATE 2010: Brief meeting; Session: Cool MPSoC Design; Posters
- Industrial Workshop, Stockholm, Sweden – April 12th, 2010
- Tutorial: SystemC for Holistic System Design, Anaheim, CA – June 18th, 2010
Joint Technical Meetings (2)

- R. Wilhelm: Timing Analysis and Timing Predictability, ARTIST DESIGN Summer School, Beijing, 2010
Plans for Y4

Workshops:

• 14th Workshop on Software and Compilers for Embedded Systems
• 4th Workshop on Mapping of Applications to MPSoCs
• 7th Workshop on Embedded Systems Education
• 3rd Workshop on Software Synthesis
• Workshop PPES 2011: Bringing Theory to Practice: Predictability and Performance in Embedded Systems, March 18, 2011, Grenoble
• WCET Tool Challenge 2011

Research & Integration as per deliverable