Year 4 Review
Dresden, March 16th, 2012

Transversal Integration Activity

Achievements and Perspectives:

Design for Predictability

and Performance

leader: Bengt Jonsson

Uppsala University
High-Level Objectives

**Background:**
- Predictability important for embedded systems, but threatened by processor development

**Objectives:**
- Technology and design techniques for achieving predictability of systems (especially on multi-core platforms)
- Trade-offs between performance and predictability

**Expected Impact:**
- Tools and Techniques for building predictable systems
- Awareness about predictability issues in system and platform design

Predictability transverses levels of abstraction
  - Verification, modeling, compilation, OS, execution 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 leads to severe economic consequences:
  – consumer electronics, telecom, ...

● Systems that require both precise execution time and high throughput
Partners

**Modeling & Validation**
- IST (Tom Henzinger)
- INRIA (Alain Girault)
- Uppsala (Bengt Jonsson, Wang Yi)
- Trento (Alberto Sangiovanni–Vincentelli)

**Code Generation & Timing analysis**
- Dortmund (Peter Marwedel)
- Saarland (Reinhard Wilhelm, Jan Reineke)
- Vienna (Peter Puschner)

**OS & Networks**
- Cantabria (Michael Gonzalez–Harbour)
- SSSA (Giorgio Buttazzo)
- York (Alan Burns)

**Hardware Platforms & MPSoC**
- Bologna (Luca Benini)
- Braunschweig (Rolf Ernst)
- ETH Zürich (Lothar Thiele)
- IMEC (Maya d’Hondt)
- Linköping (Petru Eles)
Integration: Aims

Most existing work was 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
- Sharing of resources on multi-cores

Main goal of the predictability activity:
- To integrate research across different levels of abstraction
Integration: Some Achievements

- Quantitative definition of “Predictability”
- Predictability of cache replacement policies
- Integrating Timing analysis into compilation
- Timing-predictable languages (PRET_C)
- Predictable software on multicores
  - Isolation of memory accesses and of bus accesses
  - Multicore scheduling
- Design Principles for Industrial Practice
- Standardization (MARTE)
- Predictability/Reliability of Embedded Networked Systems
- Tools: aiT, WCC, MST, MPA, MPARM, UPPAAL
- European projects: Predator, T-Crest,
Quantitative Definition of Predictability

Some tentative suggestions:

- $\text{Predictability} \approx \text{Determinism}$?
- $\text{Predictability} \approx \text{Analyzability}$?

Towards quantitative definition for architectural elements: [Grund 11]
- Inherent to the element considered
- Independent of analysis method
- Provides quantitative measure
Towards Definition of Predictability [Grund 11]

Predictability $\approx$ variability of considered quantity under explicitly given sources of uncertainty

Examples:
- Execution time of task with uncertain initial state and/or input:
  
  $BCET/WCET$ over the possible initial states and/or inputs
Towards Definition of Predictability [Grund 11]

Predictability \(\approx\) variability of considered quantity under explicitly given sources of uncertainty

Examples:
- Execution time of task with uncertain initial state and/or input:
  \[ \frac{BCET}{WCET} \text{ over the possible initial states and/or inputs} \]
- Cache replacement policy with initial state uncertainty
  \[ \text{Min} / \text{Max number of cache misses for a program of length } n \]
  - Consider this ratio as \( n \to \infty \)
  - Results for different policies (8-way associative caches) [Reineke, Grund 08]

<table>
<thead>
<tr>
<th>LRU</th>
<th>FIFO</th>
<th>PLRU</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/8</td>
<td>0</td>
</tr>
</tbody>
</table>

- Highly predictable \(\equiv 1\) (eventual independence of initial state)
- Otherwise analysis of preemptive code difficult
Caches and Preemptive Scheduling

Analysis must predict if cache blocks survive preemption
- Improved calculation of Cache-Related Preemption Delay (CRPD)
  - Considers how many accesses a reempted block can tolerate
  - Implemented in aiT for several architectures

Arbitrary preemptions decrease schedulability due to CRPD
- Allow context switches only at fixed preemption points (FPP)
  - FPP can be placed to minimize CRPD
  - or to minimize system stack usage
  - Implemented in aiT

[AbsInt, U. Saarland, SSSA]
MRU Replacement [Guan Lv Yi 12]

- Used in Intel Nehalem
- As good average-case performance as LRU [2]
  - superior to FIFO and PLRU
- But considered “un-predictable”
- MRU is a kind of approximation of LRU

Analyzability of MRU replacement policy

- Used in commercial processors, e.g., Nehalem
- Low-cost "approximation" of LRU
- Previously considered "unpredictable"
- New result makes MRU predictable and analyzable
- IDEA: new classification of memory access: $k$-Miss
  - Always Hit in LRU $\Rightarrow$ (at most) $k$-Miss in MRU
- Can be analyzed using state-of-the-art LRU analysis
- On a considered benchmark, estimated WCET under MRU is only 5%~10% more than under LRU

[Uppsala]
Timing-Aware Compilation

[WCC compiler: integrates compilation [Dortmund] and timing analysis [AbsInt]]

Makes programs timing-aware, and allows to develop optimizations for WCET

Work in the last year includes

- WCET-driven cache-aware memory content selection
- WCET-aware superblock optimizations
  (awarded three times, e.g. as best computer science thesis in Germany)
- Basic block reordering for improved branch prediction
- Loop-invariant code motion ported towards WCET, based on machine learning
- WCET- and pipeline-aware register allocation using integer-linear programming (ILP)
- Adaptive WCET-aware compilation: automatic computation of Pareto-optimal solutions trading off WCET, ACET and code size
- Scratchpad allocation for multi-task programs
Deterministic Programming with Timing Semantics

[INRIA, U. Kiel, U. Saarland, U Bamberg]

- **PRET-C and SC**: extensions of C with primitives for multi-threading, reactive inputs and outputs, tick barrier, predictable loops…
- **Synchronous semantics** providing deterministic and thread-safe communication through shared variables.
- Programs can run on a dedicated reactive processor (RP) or on a general purpose processor (GPP).
- The synchronous hypothesis is validated by computing the WCRT and comparing with the execution time constraint.
- Papers published at DATE’09, EMSOFT’09, CASES’09, DATE’10, MEMOCODE’10, DATE’11, DATE’12.
WCRT analysis for PRET-C and SC

- WCRT analysis is based on UPPAAL.
- Infeasible execution paths are pruned thanks to a new data-flow analysis.
- Benchmarks made with speculative features off.
- Sizes between 400 and 1600 LOCs.
- Execution on our RP improves the WCRT by 26% vs GPP.
- The WCRT achieved with PRET-C is 20% better than with ProtoThreads and 50% better than with Esterel.
Predictability for MultiCores

- Timing analysis for single tasks must consider or eliminate interference on shared buses, memories, ...
- Eliminate interference on shared memories (L3 cache)
  - Scratchpads for shared data
  - or: Cache coloring
- Interference on bus:
  - Consider interference in analysis
  - or: Bound it, e.g., using TDMA
- Use suitable scheduling policy
Control Sharing by Cache-Coloring [Uppsala]

No sharing of cache lines between cores
Cache-Coloring

Logical view
Cache-Coloring

- E.g. LINUX – Power5 (16 colors)

- Logical Pages of Task A
- Logical Pages of Task B
- Physical Pages
- L2 Cache

controlled by software (OS)
indexed by hardware
Considering Interference on Shared Bus

Timed model of Bus Arbitration (FCFS, TDMA)
Considering Interference on Shared Bus

Timed model of Bus Arbitration (FCFS, TDMA)

Global Timing Analysis using UPPAAL

Timing bounds for tasks
Considering Interference on Shared Bus

Timed model of Bus Arbitration (FCFS, TDMA)

Global Timing Analysis using UPPAAL

Timing bounds for tasks

On Mälardalen benchmark, much better than pessimistic analysis
Implemented in tool McAIT http://www.neu-rtes.org/mcait
Construct the timing model of programs

the TA for each program is generated from its binary code (de-compiling, CFG construction, cache analysis)
Experiments and Evaluation

- WCET Benchmark programs [MDH]

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th># instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td>bs</td>
<td>Binary search algorithm for an array</td>
<td>78</td>
</tr>
<tr>
<td>edn</td>
<td>Finite Impulse Response (FIR) filter calculations</td>
<td>896</td>
</tr>
<tr>
<td>fdct</td>
<td>Fast Discrete Cosine Transform</td>
<td>647</td>
</tr>
<tr>
<td>insertsort</td>
<td>Insertion sort on a reversed array</td>
<td>106</td>
</tr>
<tr>
<td>jfdctint</td>
<td>Discrete Cosine Transformation on a pixel block</td>
<td>691</td>
</tr>
<tr>
<td>matmult</td>
<td>Matrix multiplication</td>
<td>287</td>
</tr>
</tbody>
</table>
Results for the TDMA Bus

- Results for a 4-core system with slot size 100

<table>
<thead>
<tr>
<th>Programs</th>
<th>WCET</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AI + MC</td>
<td>AI + Worst-Case</td>
</tr>
<tr>
<td>bs</td>
<td>16,082</td>
<td>30,244</td>
</tr>
<tr>
<td>edn</td>
<td>18,428,441</td>
<td>34,946,900</td>
</tr>
<tr>
<td>fdct</td>
<td>529,682</td>
<td>1,005,350</td>
</tr>
<tr>
<td>insertsort</td>
<td>31,641</td>
<td>50,902</td>
</tr>
<tr>
<td>jfdctint</td>
<td>624,482</td>
<td>1,182,740</td>
</tr>
<tr>
<td>matmult</td>
<td>179,241</td>
<td>231,790</td>
</tr>
<tr>
<td><strong>Average</strong></td>
<td><strong>75%</strong></td>
<td></td>
</tr>
</tbody>
</table>
Results for the TDMA Bus

- Results for a 4-core system with slot size 200

<table>
<thead>
<tr>
<th>Programs</th>
<th>WCET</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>AI + MC</td>
<td>AI + Worst-Case</td>
</tr>
<tr>
<td>bs</td>
<td>16082</td>
<td>53644</td>
</tr>
<tr>
<td>edn</td>
<td>18404164</td>
<td>62519600</td>
</tr>
<tr>
<td>fdct</td>
<td>529682</td>
<td>1793450</td>
</tr>
<tr>
<td>insertsort</td>
<td>32082</td>
<td>82702</td>
</tr>
<tr>
<td>jfdctint</td>
<td>628164</td>
<td>2110940</td>
</tr>
<tr>
<td>matmult</td>
<td>179241</td>
<td>317890</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bounding Interference by TDMA

- Task graph
- TDMA bus schedule

→ WCET calculation and task scheduling

→ Worst-case global delay
Bounding Interference by TDMA

- Task graph
- TDMA bus schedule

\[ \text{WCET calculation and task scheduling} \]

\[ \text{Worst-case global delay} \]

Techniques for optimizing ACET under requirements on WCET

[Linköping]
Superblock Model

- Timing of accesses inside tasks important
- Good to structure tasks as sequences of superblocks
  - bound on execution and communication requirements

\[ \begin{align*}
\rho_{1,1} & \quad \text{computation} \quad exec_{\text{max}} \\
\rho_{1,1} + W_1 & \\
\end{align*} \]

- (A)cquisition phase to read data
- (E)xecution phase to perform computation
- (R)eplication phase to write data
Superblock Model (contn.)

- dedicated model - **DSS**
- general model - **GSS**
- hybrid model - **HSS**

Superblocks execute **sequentially (S)** or **time-triggered (T)**
Multiprocessor Scheduling

Global Scheduling

new task
waiting queue

Partitioned Scheduling

Partitioned Scheduling with Task Splitting

cpu 1  cpu 2  cpu 3

2  3  4
5  8  9
1  6  7

2  5  2
3  1  2
4  6  3

2  3  2
5  3  7
1  6  4
Achieving Liu/Layland bound f. multiprocessors

Lehoczky et al. CMU
ECRTS 2009
Liu and Layland’s Utilization Bound

  (the 19th most cited paper in computer science)

\[ \Theta(N) = N\left(2^{\frac{1}{N}} - 1\right) \]

- \( N \) : the number of tasks, \( N \to \infty \), \( \Theta(N) \approx 69.3\% \)

\[ \sum C_i/T_i \leq N\left(2^{1/N} - 1\right) \]

\[ \Rightarrow \text{the task set is schedulable} \]
Achieving Liu/Layland bound f. multiprocessors

[OPODIS'08] 38
[TPDS'05] 50
[ECRTS'03] 50
[RTSS'04] 50

Liu and Layland’s Utilization Bound

[RTCSA'06] 66

Fixed Priority
Dynamic Priority
Fixed Priority
Dynamic Priority
Fixed Priority
Dynamic Priority

Global
Partitioned

Multiprocessor Scheduling

Task Splitting

[Guan Stigge Yi Gu 09]
Achieving Liu/Layland bound f. multiprocessors

Main idea:
- Partition tasks up to the LL utilization bound
- Split (few) remaining tasks on several processors
- Needs (some) task migration

Liu and Layland’s Utilization Bound

[OPODIS’08]

Fixed Priority Dynamic Priority Fixed Priority Dynamic Priority Fixed Priority Dynamic Priority

Global Partitioned Task Splitting

Multiprocessor Scheduling

[Guan Stigge Yi Gu 09]
Achieving Liu/Layland bound f. multiprocessors

Main idea:
- Partition tasks up to the LL utilization bound
- Split (few) remaining tasks on several processors
- Needs (some) task migration

Tasks can be split and migrated with low overhead

[Liu and Layland’s Utilization Bound]

[OPODIS’08]

[38]

[Zhang Guan Zhao Yi 11]
Integrated Analysis for MultiProcessor System

Tools for timing analysis have been integrated.

Here: report from demonstration on real automotive application “DemoCar” [Bosch] in the Predator project

- WCC: compilation and WCET-aware allocation of code to scratchpads
- aiT: Analyze compiled code to generate model of local timing and bus access patterns
- MPA: Calculate actual timing for each core from its timing model and bus access patterns of interfering cores

[AbsInt, Bosch, ETH Zurich, U. Saarland]
Application Configuration

- 4 cores on bus w. FlexRay
- Code in Shared L2 Flash
- One task on each core,
- Tasks comprise 15 runnables from engine control by Bosch
WCRT using WCC/aiT and RTC

- Runnables in Application
- Compilation by WCC
- Executables
- Analysis by aiT
- Global configuration with bus access parameters
- Timing model of interference
- WCRT analysis for FlexRay
- Analysis by MPA toolbox
- WCRT analysis for TDMA schedule

- Worst-case Execution Times and Bus Access Parameters
Representation of bus interference in MPA

- Core 1: under analysis
- Cores 2, 3, 4 Interferers
Conclusions: Tool Integration Works!

- WCET-aware compilation, optimization, WCET analysis practical:
  - 50,000 lines of industrial code takes only 1 minute,
- WCET-aware optimizations outperform GCC by up to 45% in terms of WCET
- Fully automatic integration of
  - Compilation (WCC)
  - Static WCET analysis of individual tasks (aiT)
  - Compositional timing analysis on system level (MPA)
Smart Configuration: Application on P4080

Explore PROMPT guidelines

- Increase predictability on the single core level
  - Partial cache locking, static branch prediction

- Privatization
  - Each core allocates required data in its private L2 cache
  - Accesses to main memory only allowed within time slots determined by TDMA-based resource scheduling (cf. Schranzhofer et al.)

- Improved Predictability

![Diagram showing access slots for cores c1 to c4 with read and write slots, and access to the memory controller.](AbsInt)
Picture of the P4080
Standardization of UML MARTE [Canabria]

- Participation to MARTE, Real-time and Embedded systems profile for UML
  - Continuation of effort in ARTIST, ARTIST2,
  - Major role of Univ. Cantabria in the development of this standard
  - Evolution into MARTE 1.2

- Impact
  - OMG standard
  - Several PhD Thesis in Europe
  - Usage in several companies
  - Interest shown by around 75 issues being raised in this year

- Participation in SySML standard
  - Trying to align it with UML MARTE
Dissemination of MARTE

- Built a collaborative web page for dissemination of the standard
- Now it’s the official OMG web page for MARTE
  
  http://www.omgmarte.org

- Organized: ArtistDesign Workshop on Real-Time System Models for Schedulability Analysis

- Backends for Analysis:
  - **marte2mast**: a new tool for obtaining schedulability analysis models from MARTE systems
  - Using the MAST modelling technology
    - http://mast.unican.es/umlmast/marte2mast
  - Backend also exists for SymTaVision
ES reliability issues: coping with errors

• Errors and fault-tolerance impact real-time constraints

• reliability depends on error coverage and runtime overhead for error handling
  • Are results **logically correct** even if errors occur?
  • Are results provided **in time** even if errors occur?

• if error coverage is very high (e.g. EDC on CAN):
  • timing failures are the crucial part (e.g. deadline misses)
  • for **safety-critical functions**: timing failure rate must be bounded
    → safety requirements specified in standards such as IEC 61508
Reliability analysis: a formal approach (2/2)

- (a) derive transmission or execution trace based on the given task set → **timing prediction** for each job

- (b) for each job $\tau_{i,j}$: enumerate all error situations which do not cause to miss $\tau_{i,j}$ its deadline → **working set** $W_{i,j}$

- (c) for each job $\tau_{i,j}$: calculate the probability that $\tau_{i,j}$ do not miss its deadline → **success probability** $S_{i,j}$

- (d) compose all success probabilities within the hyperperiod using **AND-composition** → **reliability function** $R(t)$
Survey Papers

Survey paper
Members of Transversal Activity on Predictability:

*Building Predictable Embedded Systems*

being submitted

Some other papers


● C. Cullmann, C. Ferdinand, et al.: *Predictability considerations in the design of multi-core embedded systems*, Presented at ERTS², Toulouse, May 2010
KeyNotes, Invited Talks, Tutorials

- **RECOMP Technical Day** (Jonas Diemer)
- **SafeTRANS Industrial Day** (Rolf Ernst)
- **EMSOFT 2011** (Tom Henzinger)
- **FSE 2011** (Tom Henzinger)
- **GAMES Workshop 2011** (Tom Henzinger)
- **SVARM Workshop 2011** (Tom Henzinger)
- **DATE 2011** (Tom Henzinger)
- **Nordic Workshop 2011** (Björn Lisper)
- **LCTES 2011** (Petru Eles)
- **ARTIST Summer School China 2010** (Several speakers)
- **ARTIST Summer School Europe 2011** (Several speakers)
Position Papers and Special Issues


- C. Cullmann, C. Ferdinand, et al.: *Predictability considerations in the design of multi-core embedded systems*, Presented at ERTS², Toulouse, May 2010

- Members of Predictability Activity: *Building Predictable Embedded Systems*, being submitted to ACM TECS
Workshops

- **PPES 2011**: Industrial Workshop Bringing Theory to Practice: Performance and Predictability in Embedded Systems
  Conf. on Design, Automation & Test in Europe (DATE), Grenoble, France, March 18, 2011

- **ArtistDesign Workshop** on Real-Time System Models for Schedulability Analysis
  Santander, Spain – February, 2011

- **SCOPES 2011**: 14th International Workshop on Software and Compilers for Embedded Systems
  Schloss Rheinfels, St. Goar, Germany – June 27-28, 2011

- **3rd Workshop on Software Synthesis, 2011**
  Taipei, Taiwan – October 14th, 2011
Tools and Platforms

- **AiT**, the leading tool for computing WCETs [AbsInt, Dortmund, Saarland]
- **WCC**, the WCET aware compiler [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]
- **PRET_C**, predictable multithreaded programming in C [INRIA, Auckland]
LASTING IMPACTS

- ARTISTDesign has contributed to an integrated view on achieving predictability in embedded systems design
- By bringing researchers from diverse fields together
- Lasting collaborations have been established
- Several tools have been developed and integrated into tool chains
- Next step: integrated attack on building predictable systems
- Emerging projects focus on Mixed-Criticality,
  - E.g., the CERTAINTY project