The time is now:
Timing Verification for Safety-Critical Multi-Cores

Sebastian Altmeyer
Safety-critical real-time systems

- must react **correctly**
- must react within stringent **time bounds**

A single failure may lead to a catastrophe

→ Timing verification needed
State-of-the-Art Timing Verification

1. Step: Timing Analysis
Derives worst-case execution time (WCET) of each task.

2. Step: Scheduling Analysis
Checks if all tasks scheduled together meet their timing constraints.

Task: smallest individual software component
Advent of Multi-Cores

- Better performance
- Lower heat dissipation
- Reduced power consumption
- Lower costs

Problem:
What happens if each core accesses the communication bus at the same time?
State-of-the-art Timing Verification fails due to interference on shared resources.

1) WCETs not independent of other tasks
2) Scheduling model assumes no interference costs
My Proposal: Integrated Timing Verification Framework

Scheduling Analysis
- Computes co-running tasks
- Explicitly considers interference

Timing Analysis
Bounds WCET and Interferences

for multi-core systems
Key Objectives

- **Static timing and interference analysis**
  Quantification of WCET and interference of individual tasks

- **Interference-aware scheduling analysis**
  Compute system performance (response times) including interference

- **Classification of hardware components**
  Which communication arbiter / network topology / memory hierarchy provides which performance/predictability/analysability?
Approach: Gradual Refinement of Models

**State-of-the-art**

abstract task model

no interference

**Result of the project**

concrete task model

- interference on local caches $\alpha$
- interference on shared bus $\beta$
- interference on shared cache $\gamma$

\[
R_i = C_i + \sum_{j \in h p(i)} \left[ \frac{R_i}{T_j} \right] (C_j + \alpha + \beta + \gamma)
\]
Approach: Gradual Refinement of Models

abstract task model
no interference

refined task model
- interference on local caches $\alpha$

refined task model
- interference on local caches $\alpha$
- interference on shared bus $\beta$

concrete task model
- interference on local caches $\alpha$
- interference on shared bus $\beta$
- interference on shared cache $\gamma$

$R_i = C_i + \sum_{j \in hp(i)} \left\lfloor \frac{R_i}{T_j} \right\rfloor (C_j)$

$R_i = C_i + \sum_{j \in hp(i)} \left\lfloor \frac{R_i}{T_j} \right\rfloor (C_j + \alpha)$

$R_i = C_i + \sum_{j \in hp(i)} \left\lfloor \frac{R_i}{T_j} \right\rfloor (C_j + \alpha + \beta)$

$R_i = C_i + \sum_{j \in hp(i)} \left\lfloor \frac{R_i}{T_j} \right\rfloor (C_j + \alpha + \beta + \gamma)$
Right Expertise Needed

1. Step: Timing Analysis
   Derives worst-case execution time (WCET) of each task

2. Step: Scheduling Analysis
   Checks if all tasks scheduled together meet their timing constraints

Compiler design community

Operations research community

Timing Analysis
Bounds WCET and Interferences

Scheduling Analysis
• Computes co-running tasks
• Explicitly considers interference

Combined knowledge from both communities required
Expected Outcomes

1) Integrated Timing Verification Framework

2) New, realistic real-time scheduling model/theory

3) Safety-critical systems exploit full potential of multicores (shown using prototype and use case)
Workplan and Timetable

**Work-package 1** Static Analysis of Bus/Memory/Cache usage

**Work-package 2** Inter-dependency analysis of various interferences

**Work-package 3** Interference-aware Schedulability Analysis

**Work-package 4** Classification of multi-core components

**Work-package 5** Prototype dissemination

The following gantt-chart describes the chronological order of the workpackages:
1) **Real-Time Systems Group** at the University of York for real-time scheduling

2) **Compiler Design Lab** at the Saarland University for static timing analysis

3) **Computer Systems Architecture Group** at the University of Amsterdam for multi-core design and implementation
Knowledge Utilization

Publications
(main real-time conference, main computer design/automation conferences)

Software Release
(working prototype of the timing validation)

Industrial Use Cases
(collaboration with Industrial Partners)
Schedulability Analysis

strongly depends on specific scheduler
(static or dynamic, priority driven, static/dynamic priority assignment …)

Fixed priority pre-emptive scheduling
set of \textit{n tasks}, each \textit{task} \textit{i} has
- exec time bound $C_i$,
- period (or minimal inter-arrival time) $T_i$
- relative deadline $D_i$
- priority $pr_i$

Response time $R$: \textbf{maximal time between task release and task completion}
Task-set schedulable, iff $\forall i: R_i \leq D_i$

$$R_i = C_i + \sum_{j \in hp(i)} \left\lfloor \frac{R_i}{T_j} \right\rfloor C_j$$

- own execution time of \textit{task} \textit{i}
- all tasks with \textbf{higher priority}
- one complete execution of task $j$ during $R_i$
- how often \textit{task} $j$ can pre-empt \textit{task} $i$