A Framework for the Derivation of WCET Analyses for Multi-Core Processors

Michael Jacobs, Sebastian Hahn, Sebastian Hack

Department of Computer Science
Saarland University

July 7, 2016
Context of Our Work

- Timing verification
  - Worst-case execution time (WCET) analysis
  - Scheduling theory / response time analysis

Multi-core processors

- Shared resources: buses, caches, ...

- Shared-resource interference

- Strong impact on performance

Must be considered in timing verification

Scope of our work

- WCET analysis for multi-core processors

- Static analysis

- Non-probabilistic

- Not (yet) integrated with response time analysis
Context of Our Work

- **Timing verification**
  - Worst-case execution time (WCET) analysis
  - Scheduling theory / response time analysis

- **Multi-core processors**
  - Shared resources: buses, caches, ...
  - **Shared-resource interference**
    - Strong impact on performance
  - Must be considered in timing verification
Context of Our Work

- **Timing verification**
  - Worst-case execution time (WCET) analysis
  - Scheduling theory / response time analysis

- **Multi-core processors**
  - Shared resources: buses, caches, …
  - *Shared-resource interference*
    - Strong impact on performance
  - Must be considered in timing verification

- **Scope of our work**
  - WCET analysis for multi-core processors
  - Static analysis
  - Non-probabilistic
  - Not (yet) integrated with response time analysis
Motivation
Existing Work

WCET Analysis and Response Time Analysis for Multi-Core Processors

- [Kelter and Marwedel, 2014]
- [Chattopadhyay et al., 2012]
- [Schranzhofer et al., 2010]
- [Schliecker et al., 2009]
- [Schliecker and Ernst, 2010]
- [Pellizzoni et al., 2010]
- [Schranzhofer et al., 2011]
- [Dasari et al., 2011]
- [Giannopoulou et al., 2012]
- [Liang et al., 2012]
- [Dasari and Nélis, 2012]
- [Nowotsch, 2014]
- [Altmeyer et al., 2015]
Existing Work

WCET Analysis and Response Time Analysis for Multi-Core Processors

- [Kelter and Marwedel, 2014] → enumeration of all interleavings
- [Chattopadhyay et al., 2012]
- [Schranzhofer et al., 2010]
- [Schliecker et al., 2009]
- [Schliecker and Ernst, 2010]
- [Pellizzoni et al., 2010]
- [Schranzhofer et al., 2011]
- [Dasari et al., 2011]
- [Giannopoulou et al., 2012]
- [Liang et al., 2012]
- [Dasari and Nélis, 2012]
- [Nowotsch, 2014]
- [Altmeyer et al., 2015]
Existing Work

WCET Analysis and Response Time Analysis for Multi-Core Processors

- [Kelter and Marwedel, 2014] \{ enumeration of all interleavings
- [Chattopadhyay et al., 2012]
- [Schranzhofer et al., 2010]
- [Schliecker et al., 2009]
- [Schliecker and Ernst, 2010]
- [Pellizzoni et al., 2010]
- [Schranzhofer et al., 2011]
- [Dasari et al., 2011]
- [Giannopoulou et al., 2012]
- [Liang et al., 2012]
- [Dasari and Nélis, 2012]
- [Nowotsch, 2014]
- [Altmeyer et al., 2015]

\{ only support TDMA bus arbitration

Michael Jacobs

WCET Analyses for Multi-Core Processors

July 7, 2016 3 / 31
Existing Work

WCET Analysis and Response Time Analysis for Multi-Core Processors

- [Kelter and Marwedel, 2014] enumeration of all interleavings
- [Chattopadhyay et al., 2012]
- [Schranzhofer et al., 2010]
- [Schliecker et al., 2009]
- [Schliecker and Ernst, 2010]
- [Pellizzoni et al., 2010]
- [Schranzhofer et al., 2011]
- [Dasari et al., 2011]
- [Giannopoulou et al., 2012]
- [Liang et al., 2012]
- [Dasari and Nélis, 2012]
- [Nowotsch, 2014]
- [Altmeyer et al., 2015]

Only support TDMA bus arbitration

Rely on compositionality
Motivating Example

All 6 Behaviors of a Simple Toy Program:

- Non-interfered execution
- Direct interference effect
- Indirect interference effect

▶ only as consequence of direct interference

WCET = 11

☐ = non-interfered execution
Motivating Example
All 6 Behaviors of a Simple Toy Program:

- non-interfered execution
- direct interference effect

WCET = 11
Motivating Example

All 6 Behaviors of a Simple Toy Program:

\[
\begin{array}{cccccccc}
\text{non-interfered} & \text{direct interference} & \text{indirect interference} \\
\hline
\checkmark & \checkmark & \checkmark \\
\checkmark & \checkmark & \checkmark \\
\end{array}
\]

WCET = 11

- = non-interfered execution
- = direct interference effect
- = indirect interference effect
  - only as consequence of direct interference
Classical Compositional Timing Analysis

For our Example:

Typical compositional analysis

WCET = 11
Classical Compositional Timing Analysis

For our Example:

Typical compositional analysis

WCET = 11

\[ \text{WCET} = 10 \text{ time units} \]
Classical Compositional Timing Analysis

For our Example:

Typical compositional analysis

\[ \text{Typical compositional analysis} = 10 \text{ time units} \]

Unsoundness

- Underestimates WCET

WCET = 11
Increasing Penalty in Compositional Analysis

For our Example:

Compositional analysis

\[ WCET = 11 \]
Increasing Penalty in Compositional Analysis

For our Example:

Compositional analysis

Add indirect effects to penalty

\[ \text{WCET} = 11 \]
Increasing Penalty in Compositional Analysis

For our Example:

Compositional analysis
Add indirect effects to penalty

\[ \text{WCET} = 11 \]

Limitations

Imprecision
Increasing Penalty in Compositional Analysis

For our Example:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

WCET = 11

- Compositional analysis
- Add indirect effects to penalty

\[ \text{WCET} = 15 \text{ time units} \]

Limitations

- Imprecision
- How to bound indirect effects per direct effect for a HW?
Increasing Penalty in Compositional Analysis

For our Example:

Compositional analysis

Add indirect effects to penalty

\[ \text{WCET} = 11 \]

Limitations

- Imprecision
- How to bound indirect effects per direct effect for a HW?
- Not possible for HW with domino effects!
A Novel Analysis by Us

"WCET Analysis for Multi-Core Processors with Shared Buses and Event-Driven Bus Arbitration"
at RTNS 2015 [Jacobs et al., 2015]

- not compositional
  - explicitly models interference in core pipeline
A Novel Analysis by Us

"WCET Analysis for Multi-Core Processors with Shared Buses and Event-Driven Bus Arbitration"
at RTNS 2015 [Jacobs et al., 2015]

- not compositional
  - explicitly models interference in core pipeline
- sound & precise
- scalable
  - octa-core processors
  - out-of-order execution
Focus of This Talk

- Concepts
  - used during derivation of [Jacobs et al., 2015]

Our Paper

- embeds concepts in **formal framework**
- rigorous **soundness proofs**
The Derivation of a WCET Analysis
Set *Traces* of system behaviors
The Actual WCET

- Maximum execution time over all system behaviors

![Diagram showing the relationship between execution time and WCET/BCET with traces highlighted]
Approximation of System Behavior

- Set $\hat{\text{Traces}}$ of abstract traces

- $\hat{t} \in \hat{\text{Traces}}$ describes ($\gamma_{\text{trace}}$):
  - system behaviors and/or
  - spurious behaviors
Soundness of an Approximation

- Traces must overapproximate all system behaviors
Time Bounds per Abstract Trace

- sound w.r.t. everything $\hat{t}$ describes
\[
\max_{t \in \text{Traces}} \overline{UBtime}(\hat{t})
\]
Infeasible Abstract Traces

\[ \text{Infeas} = \{ \hat{t} \mid \hat{t} \in \text{Traces} \land \gamma_{\text{trace}}(\hat{t}) \cap \text{Traces} = \emptyset \} \]

- describe **only spurious** behavior
Impact of Infeasible Abstract Traces

- might **dominate** WCET bound
Impact of Infeasible Abstract Traces

- might **dominate** WCET bound

**Goal:** prune them
  - How to detect them?

WCET Analyses for Multi-Core Processors

Michael Jacobs

July 7, 2016
System Property

- Property $P$
  - boolean predicate on execution behaviors
System Property

- Property $P$
  - boolean predicate on execution behaviors

- System property $P$
  - holds for each system behavior
Lifted System Property

- Property $\hat{P}$
  - boolean predicate on abstract traces
Lifted System Property

- Property $\hat{P}$
  - boolean predicate on abstract traces

- Criterion:

  $\gamma^{\text{trace}} \Rightarrow \hat{P}(\hat{t})$
Detect Infeasible Abstract Trace $\hat{t}$

- by $\neg \hat{P}(\hat{t})$
Detect Infeasible Abstract Trace $\hat{t}$

- by $\neg \hat{P}(\hat{t})$

- **sound** because of:

\[ \neg \hat{P}(\hat{t}) \Rightarrow \]

\[ Traces \]

\[ P \]

\[ trace \]
Detect Infeasible Abstract Trace $\hat{t}$

- by $\neg \hat{P}(\hat{t})$

- sound because of:

  $$\neg \hat{P}(\hat{t}) \Rightarrow$$

- not necessarily complete
Analysis Derivation Workflow

1. pessimistic baseline approximation
Analysis Derivation Workflow

1. pessimistic **baseline approximation**
2. **identify** system properties
Analysis Derivation Workflow

1. pessimistic baseline approximation
2. identify system properties
3. lift them to approximation
Analysis Derivation Workflow

1. **pessimistic baseline approximation**
2. **identify** system properties
3. **lift** them to approximation
4. **implement** the analysis
Property Lifting Examples
Bounding Shared-Bus Delay

- round-robin bus arbitration
  - $n$-core processor

\[ P(t) = \#\text{blocked} \leq (n-1) \cdot \text{lat}_{\text{acc}} \cdot \#\text{accesses} \]

\[ \hat{P}(\hat{t}) = \text{LB} \#\text{blocked} \leq (n-1) \cdot \text{lat}_{\text{acc}} \cdot \text{UB} \#\text{accesses} \]
Bounding Shared-Bus Delay

- **round-robin bus arbitration**
  - **n-core processor**

\[ P(t) = \sum \#\text{blocked}_{C_i}(t) \leq (n - 1) \cdot \text{lat}_{\text{acc}} \cdot \#\text{accesses}_{C_i}(t) \]
Bounding Shared-Bus Delay

- round-robin bus arbitration
  - $n$-core processor

- $P(t) = \#\text{blocked}_{C_i}(t) \leq (n - 1) \cdot \text{lat}_{\text{acc}} \cdot \#\text{accesses}_{C_i}(t)$

- $\hat{P}(\hat{t}) = \text{LB} \#\text{blocked}_{C_i}(\hat{t}) \leq (n - 1) \cdot \text{lat}_{\text{acc}} \cdot \text{UB} \#\text{accesses}_{C_i}(\hat{t})$
Bounding Loop Iterations

- loop bound $B_L$ for loop $L$
  - back edge of $L$ at most taken $B_L$ times before $L$ is left

$P(t) = \#\text{backEdge}_L(t) \leq B_L \cdot \#\text{entered}_L(t)$

$P(\hat{t}) = \hat{L} \#\text{backEdge}_L(\hat{t}) \leq B_L \cdot \hat{U}_L \#\text{entered}_L(\hat{t})$
Bounding Loop Iterations

- Loop bound $B_L$ for loop $L$
  - Back edge of $L$ at most taken $B_L$ times before $L$ is left

- $P(t) = \#backEdge_L(t) \leq B_L \cdot \#entered_L(t)$
Bounding Loop Iterations

- loop bound $B_L$ for loop $L$
  - back edge of $L$ at most taken $B_L$ times before $L$ is left

- $P(t) =$
  \[
  \#\text{backEdge}_L(t) \leq B_L \cdot \#\text{entered}_L(t)
  \]

- $\hat{P}(\hat{t}) =$
  \[
  L^B \#\text{backEdge}_L(\hat{t}) \leq B_L \cdot U^B \#\text{entered}_L(\hat{t})
  \]
Experimental Evaluation
Experimental Setup

- **Hardware platforms**
  - ARM® instruction set
  - four processor-core configurations
  - round-robin shared bus
  - SRAM latency: 10 cycles
  - dual-, quad-, and octa-core
Experimental Setup

- **Hardware platforms**
  - ARM® instruction set
  - four processor-core configurations
  - round-robin shared bus
  - SRAM latency: 10 cycles
  - dual-, quad-, and octa-core

- **Benchmarks**
  - 31 from Mälardalen suite
  - 6 generated from SCADE models
Experimental Setup

- Hardware platforms
  - ARM® instruction set
  - four processor-core configurations
  - round-robin shared bus
  - SRAM latency: 10 cycles
  - dual-, quad-, and octa-core

- Benchmarks
  - 31 from Mälardalen suite
  - 6 generated from SCADE models

- Analysis
  - co-runner-insensitive WCET bounds
  - per benchmark
  - per hardware configuration
Average Analysis-Runtime Increase
Compared to Compositional Analysis

- **Increasing complexity** of processor cores

<table>
<thead>
<tr>
<th>2-Core</th>
<th>in-order execution</th>
<th>out-of-order execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>local instruction scratchpad</td>
<td>3.3%</td>
<td>5.4%</td>
</tr>
<tr>
<td>local instruction cache</td>
<td>5.0%</td>
<td>15.9%</td>
</tr>
</tbody>
</table>
### Average Analysis/Runtime Increase

Compared to Compositional Analysis

- **increasing complexity** of processor cores

<table>
<thead>
<tr>
<th></th>
<th>2-Core</th>
<th>in-order execution</th>
<th>out-of-order execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>local instruction scratchpad</td>
<td>3.3%</td>
<td>5.4%</td>
<td></td>
</tr>
<tr>
<td>local instruction cache</td>
<td>5.0%</td>
<td>15.9%</td>
<td></td>
</tr>
</tbody>
</table>

- **increasing number** of processor cores
  - out-of-order execution, local instruction cache

<table>
<thead>
<tr>
<th></th>
<th>2-Core</th>
<th>4-Core</th>
<th>8-Core</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>15.9%</td>
<td>15.2%</td>
<td>14.9%</td>
</tr>
</tbody>
</table>
What else is in the paper?
Co-Runner-Sensitive Analysis

- In this talk
  - co-runner-insensitive analysis
Co-Runner-Sensitive Analysis

In this talk

- co-runner-insensitive analysis

Goal

- co-runner-sensitive analysis
- e.g. under work-conserving bus arbitration
Co-Runner-Sensitive Analysis

■ In this talk
  ▸ co-runner-insensitive analysis

■ Goal
  ▸ co-runner-sensitive analysis
  ▸ e.g. under work-conserving bus arbitration

■ Challenge
  ▸ avoid enumerating all interleavings of access requests
Co-Runner-Sensitive Analysis

- In this talk
  - co-runner-insensitive analysis

- Goal
  - co-runner-sensitive analysis
  - e.g. under work-conserving bus arbitration

- Challenge
  - avoid enumerating all interleavings of access requests

- In our paper: iterative overapproximation algorithm
  - give up some precision
  - keep analysis runtime manageable
Summary
Summary

- Formal framework
  - sound
  - modular
  - **applicable** to any hardware
Summary

- **Formal framework**
  - sound
  - modular
  - applicable to any hardware

- **Results for prototype analysis**
  - **scalability** shown for
    - octa-core processors
    - non-trivial processor-core features
Summary

- **Formal framework**
  - sound
  - modular
  - **applicable** to any hardware

- **Results for prototype analysis**
  - **scalability** shown for
    - octa-core processors
    - non-trivial processor-core features

- **Future work**
  - shared caches
  - more processor-core features
  - integrate with response time analysis
References I


Timed model checking with abstractions: Towards worst-case response time analysis in
resource-sharing manycore systems.

WCET analysis for multi-core processors with shared buses and event-driven bus arbitration.

Parallelism analysis: Precise WCET values for complex multi-core systems.
In Revised Selected Papers of the 3rd International Workshop on Formal Techniques for

Timing analysis of concurrent programs running on shared cache multi-cores.

Interference-sensitive Worst-case Execution Time Analysis for Multi-core Processors.
PhD thesis.

Worst case delay analysis for memory interference in multicore systems.


What Is an Abstract Trace?

- sequence of abstract states in **micro-architectural analysis**
What Is an Abstract Trace?

- sequence of abstract states in micro-architectural analysis
- path through abstract graph representation
What Is an Abstract Trace?

- sequence of abstract states in **micro-architectural analysis**
- path through **abstract graph representation**
- ILP valuation in **implicit path enumeration**
  - lifted property implemented by constraints