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

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Context of Our Work

- Timing verification
  - Worst-case execution time (WCET) analysis
  - Scheduling theory / 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] } only support TDMA bus arbitration
- [Schranzhofer et al., 2010] } only support TDMA bus arbitration
- [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]

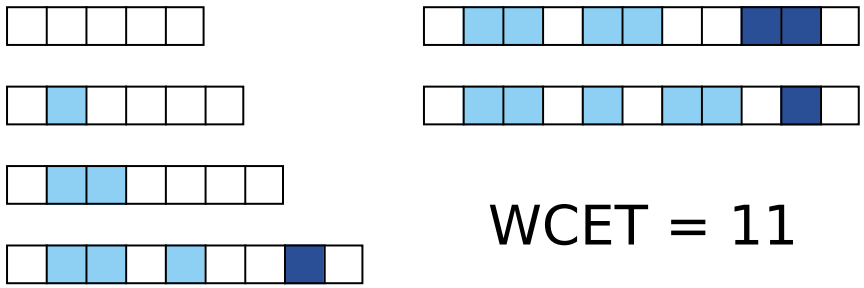
Existing Work

WCET Analysis and Response Time Analysis for Multi-Core Processors

- [Kelter and Marwedel, 2014] } enumeration of all interleavings
- [Chattopadhyay et al., 2012] } only support TDMA bus arbitration
- [Schranzhofer et al., 2010] } only support TDMA bus arbitration
- [Schliecker et al., 2009] } rely on compositionality
- [Schliecker and Ernst, 2010] } rely on compositionality
- [Pellizzoni et al., 2010] } rely on compositionality
- [Schranzhofer et al., 2011] } rely on compositionality
- [Dasari et al., 2011] } rely on compositionality
- [Giannopoulou et al., 2012] } rely on compositionality
- [Liang et al., 2012] } rely on compositionality
- [Dasari and Nélis, 2012] } rely on compositionality
- [Nowotsch, 2014] } rely on compositionality
- [Altmeyer et al., 2015] } rely on compositionality

Motivating Example

All 6 Behaviors of a Simple Toy Program:

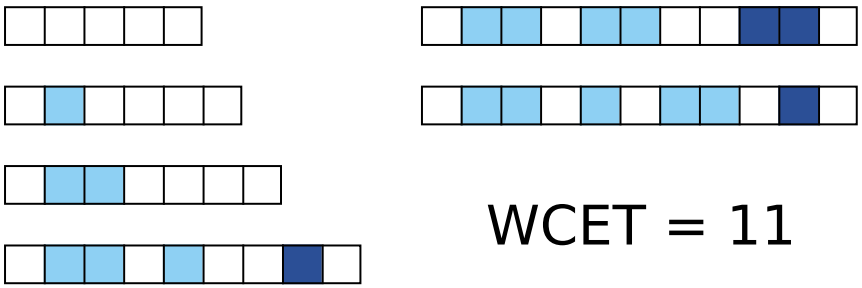


WCET = 11

□ = **non-interfered** execution

Motivating Example

All 6 Behaviors of a Simple Toy Program:



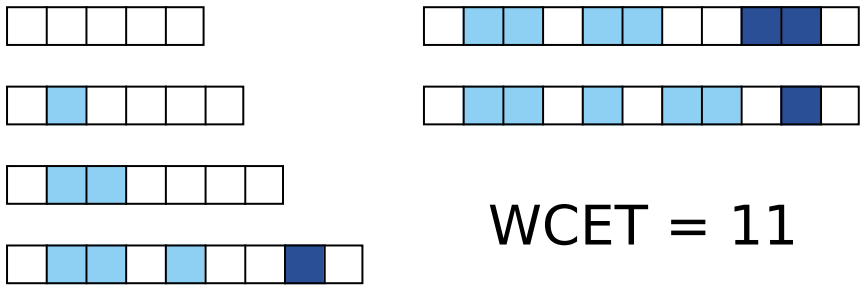
WCET = 11

□ = **non-interfered** execution

■ = **direct interference** effect

Motivating Example

All 6 Behaviors of a Simple Toy Program:



WCET = 11

□ = **non-interfered** execution

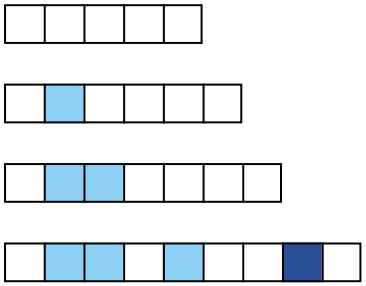
■ = **direct interference** effect

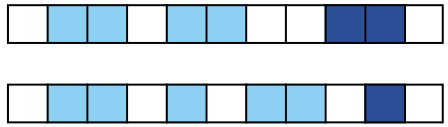
■ = **indirect interference** effect

- only as **consequence of** direct interference

Classical Compositional Timing Analysis

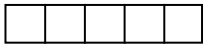
For our Example:





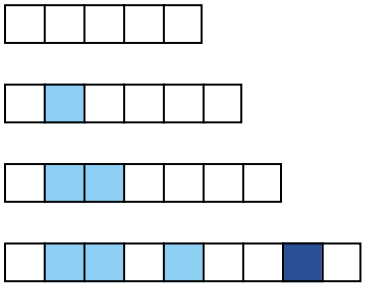
WCET = 11

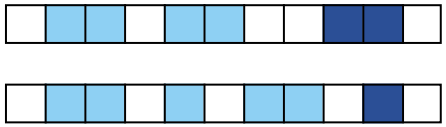
- Typical **compositional analysis**



Classical Compositional Timing Analysis

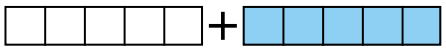
For our Example:





WCET = 11

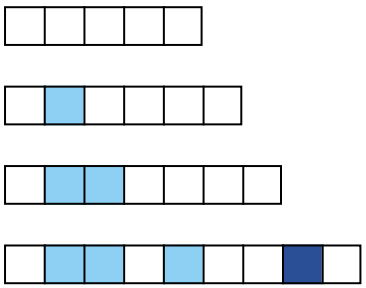
- Typical **compositional analysis**

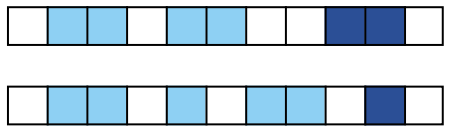
 = 10 time units

Classical Compositional Timing Analysis

SAARLAND UNIVERSITY
COMPUTER SCIENCE

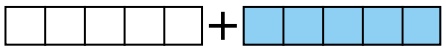
For our Example:





WCET = 11

- Typical **compositional analysis**



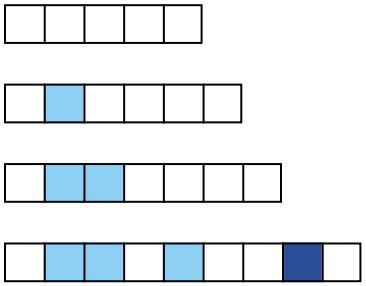
= 10 time units

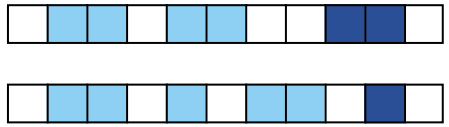
Unsoundness

- Underestimates WCET

Increasing Penalty in Compositional Analysis

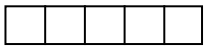
For our Example:





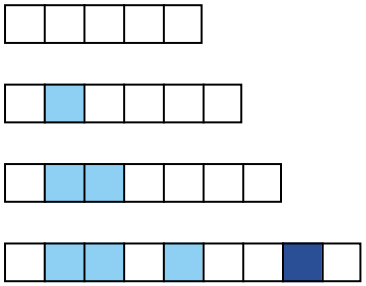
WCET = 11

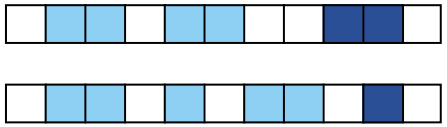
- Compositional analysis



Increasing Penalty in Compositional Analysis

For our Example:





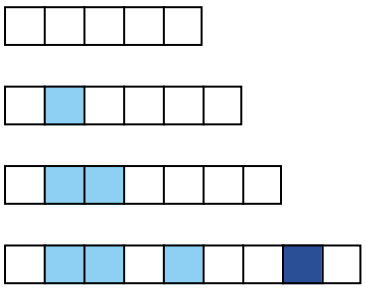
WCET = 11

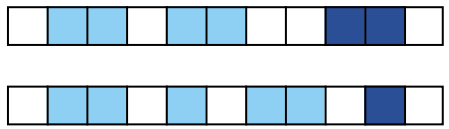
- Compositional analysis
- Add indirect effects to penalty

= 15 time units

Increasing Penalty in Compositional Analysis

For our Example:





WCET = 11

- Compositional analysis
- Add indirect effects to penalty

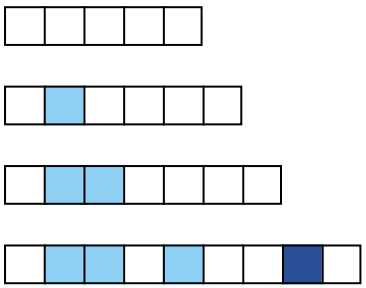
 = 15 time units

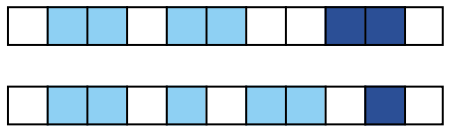
Limitations

- Imprecision

Increasing Penalty in Compositional Analysis

For our Example:





WCET = 11

- Compositional analysis
- Add indirect effects to penalty

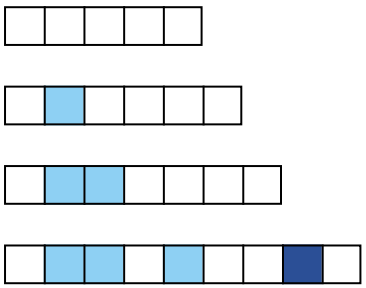
= 15 time units

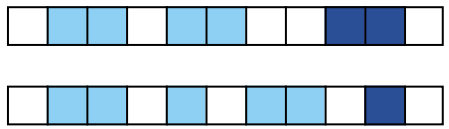
Limitations

- Imprecision
- How to bound indirect effects per direct effect for a HW?

Increasing Penalty in Compositional Analysis

For our Example:





WCET = 11

- Compositional analysis
- Add indirect effects to penalty

= 15 time units

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**

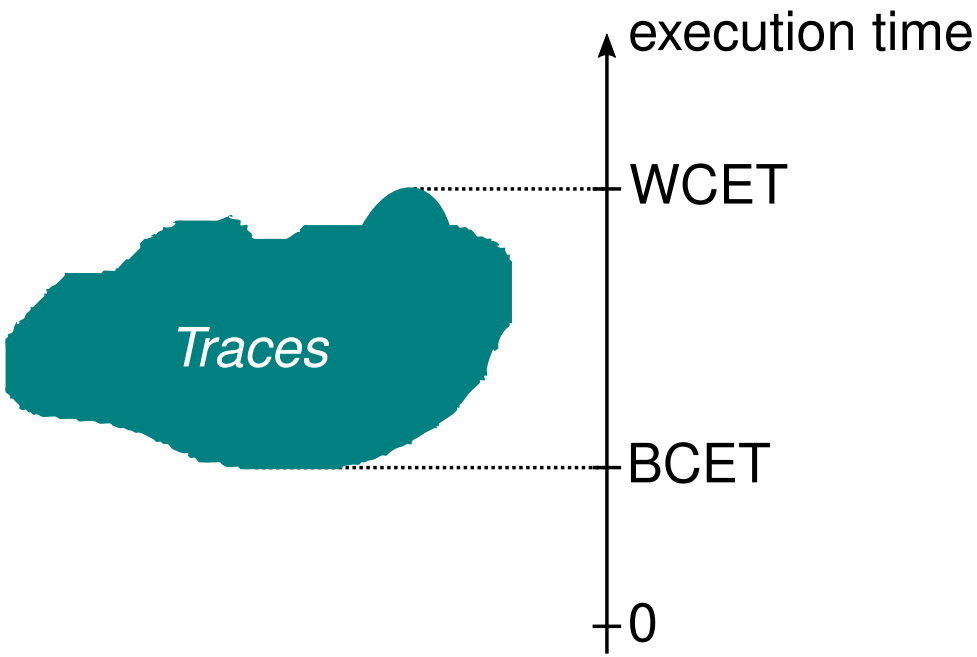
The Derivation of a WCET Analysis

Concrete-System Behavior

- Set *Traces* of system behaviors

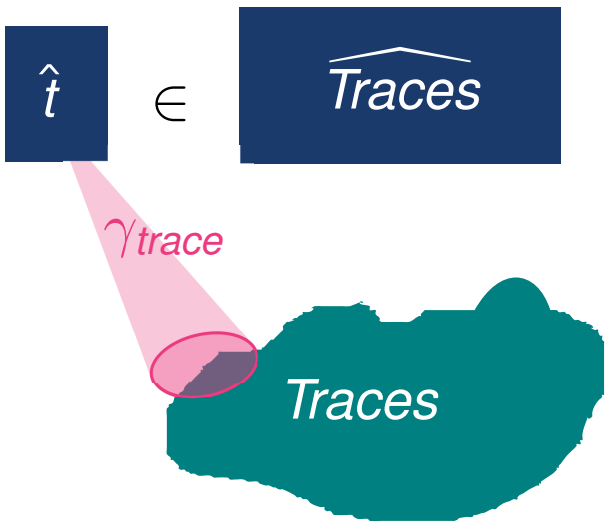
The Actual WCET

- Maximum execution time over all system behaviors



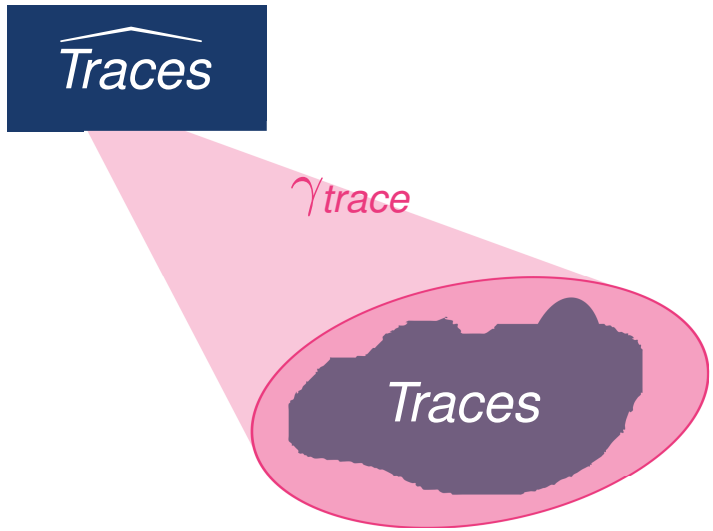
Approximation of System Behavior

- Set $\widehat{Traces}$ of **abstract traces**
- A $\hat{t} \in \widehat{Traces}$ describes (γ_{trace}):
  - system behaviors and/or
  - spurious behaviors



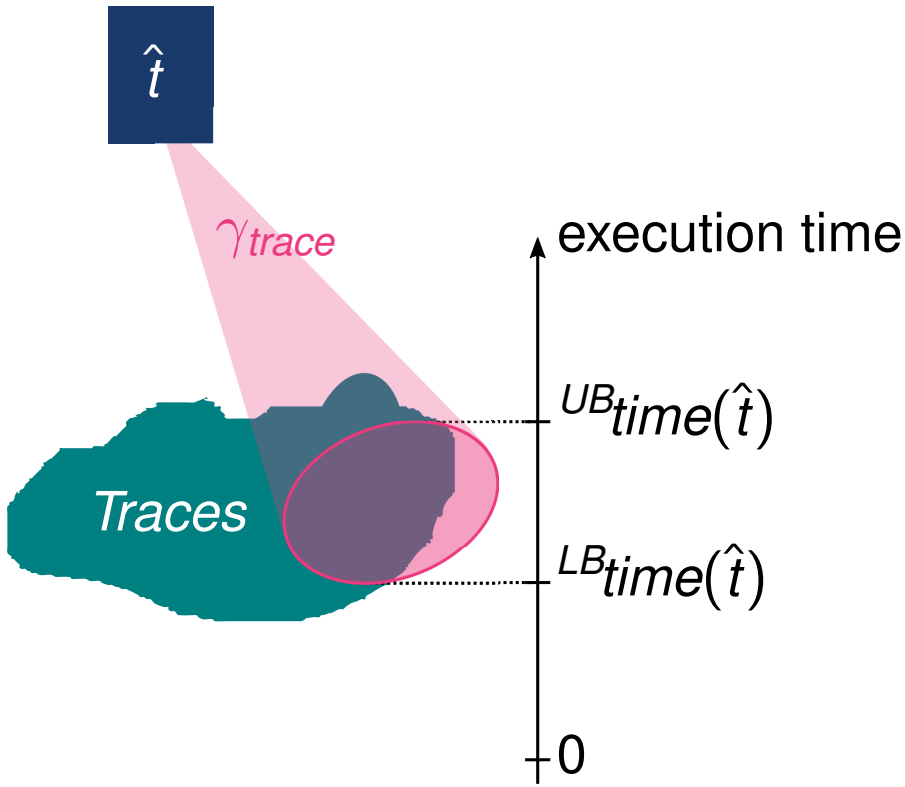
Soundness of an Approximation

- $\widehat{Traces}$ must overapproximate **all** system behaviors



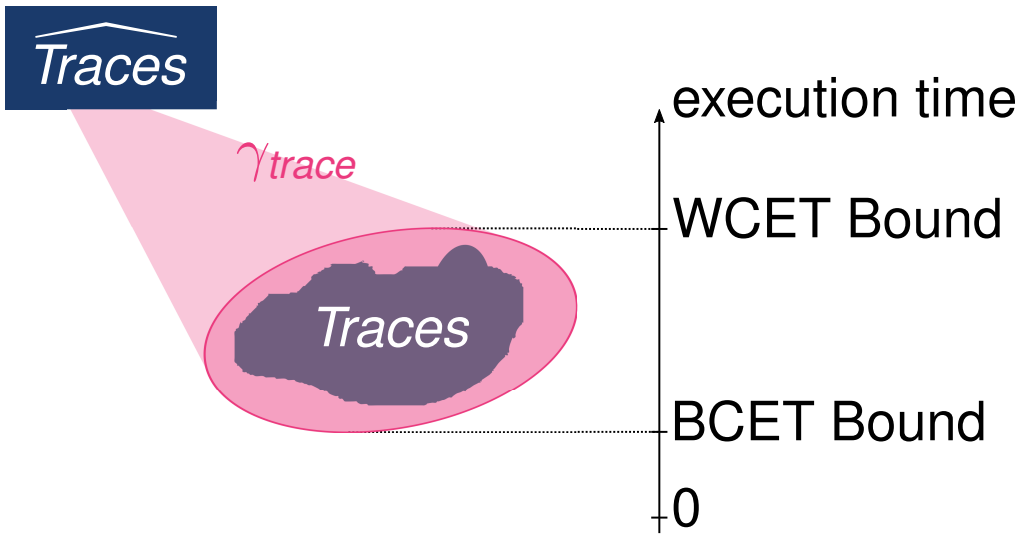
Time Bounds per Abstract Trace

- sound w.r.t. everything $\hat{t}$ describes



WCET Bound

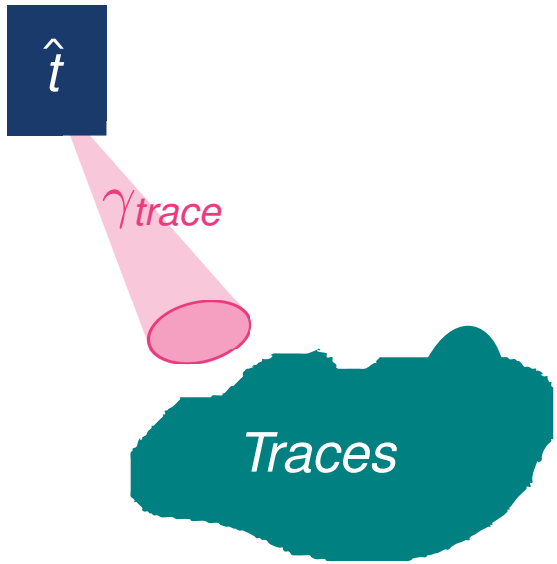
- $\max_{\hat{t} \in \widehat{Traces}} {}^{UB}time(\hat{t})$



Infeasible Abstract Traces

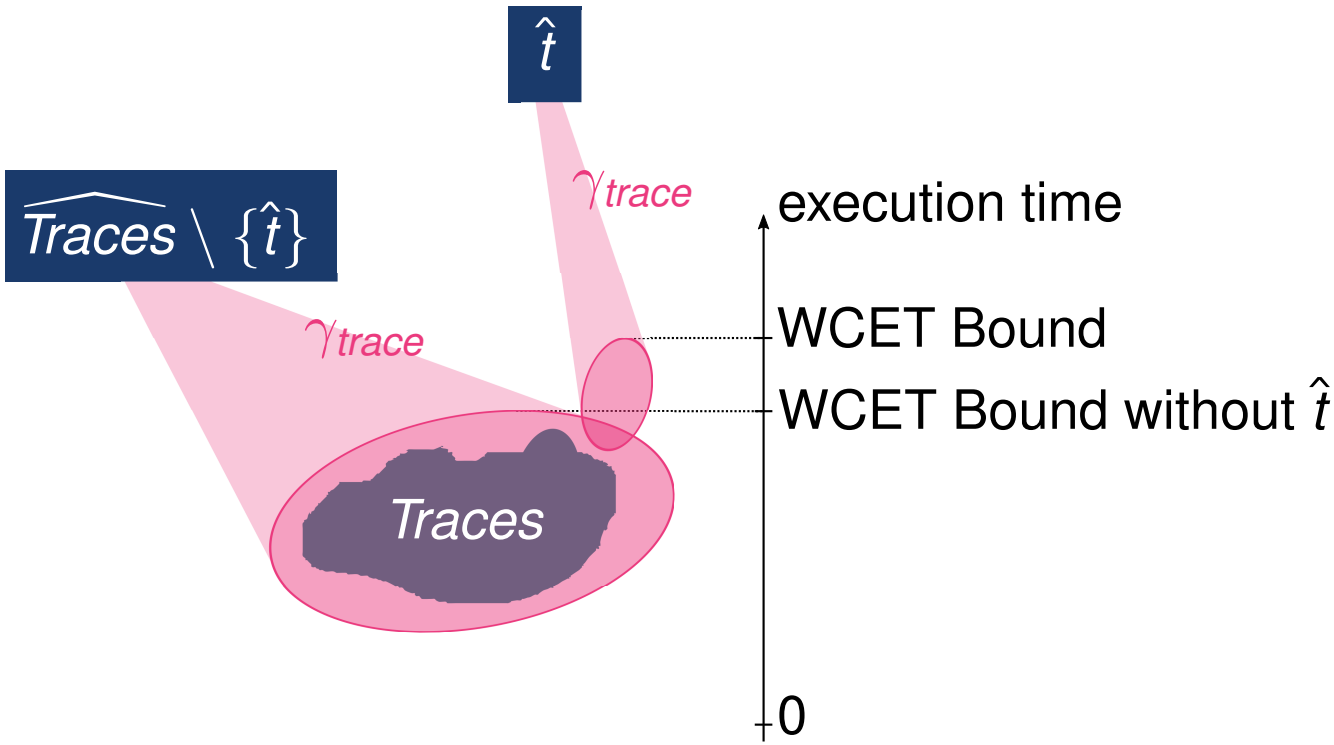
$$\widehat{Infeas} = \{\hat{t} \mid \hat{t} \in \widehat{Traces} \wedge \gamma_{trace}(\hat{t}) \cap Traces = \emptyset\}$$

- describe **only spurious** behavior



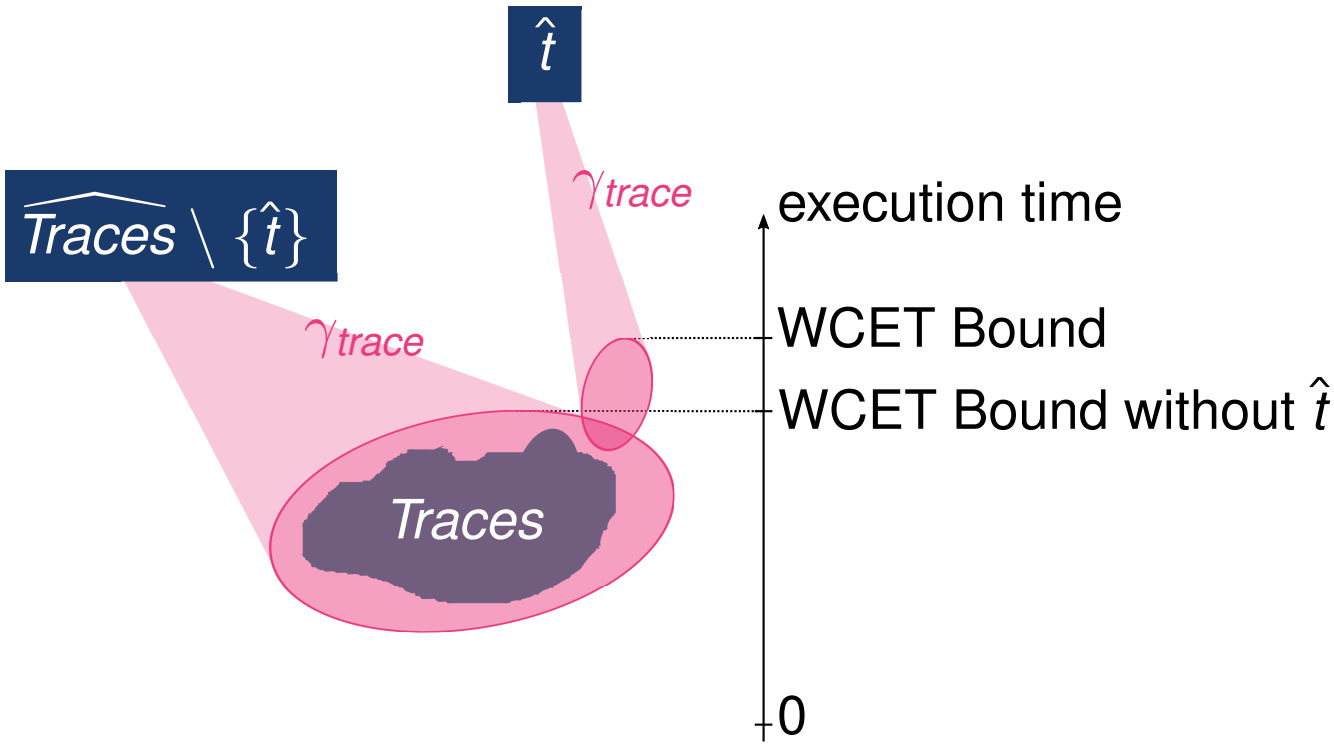
Impact of Infeasible Abstract Traces

- might **dominate** WCET bound



Impact of Infeasible Abstract Traces

- might **dominate** WCET bound



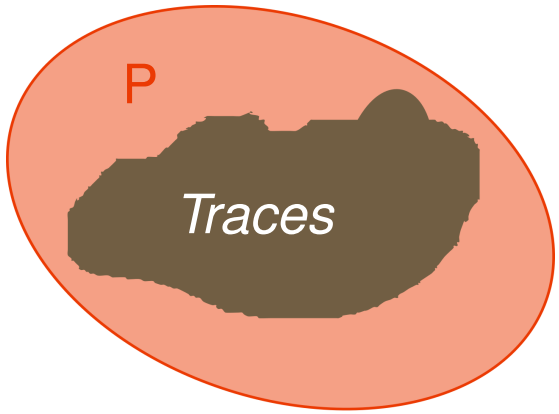
- Goal: **prune** them
  - How to detect them?

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

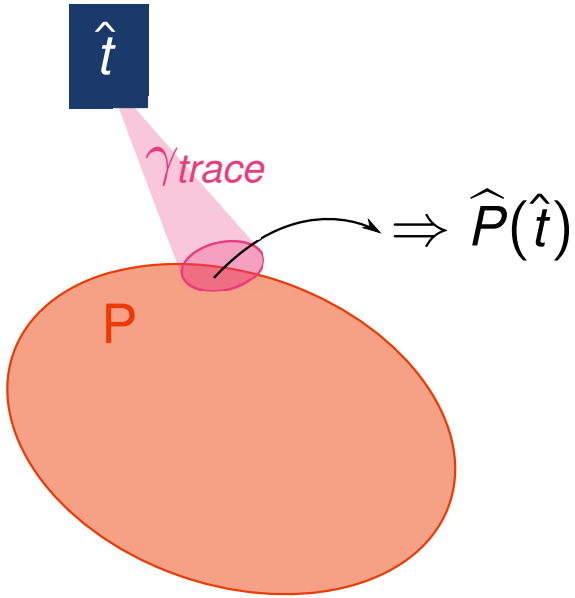


Lifted System Property

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

Lifted System Property

- Property $\widehat{P}$
  - boolean predicate on abstract traces
- Criterion:

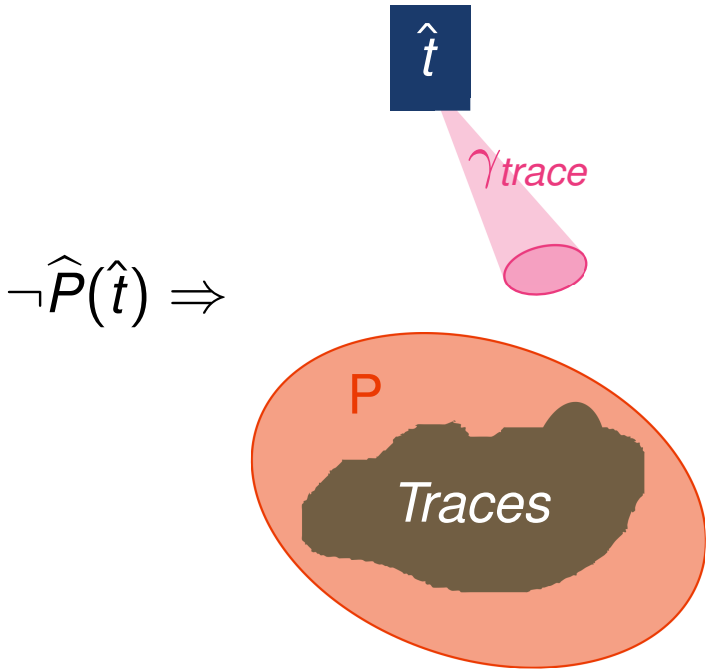


Detect Infeasible Abstract Trace $\hat{t}$

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

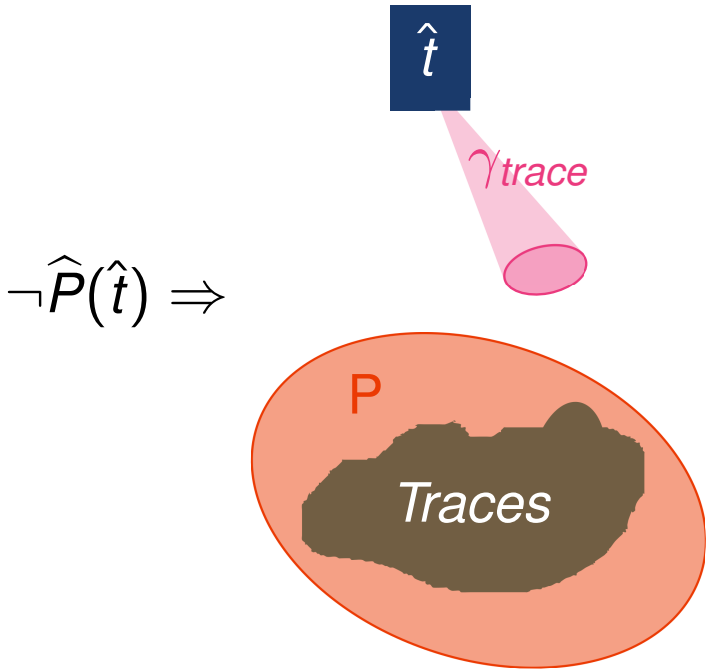
Detect Infeasible Abstract Trace $\hat{t}$

- by $\neg\widehat{P}(\hat{t})$
- **sound** because of:



Detect Infeasible Abstract Trace $\hat{t}$

- by $\neg\widehat{P}(\hat{t})$
- **sound** because of:



- 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

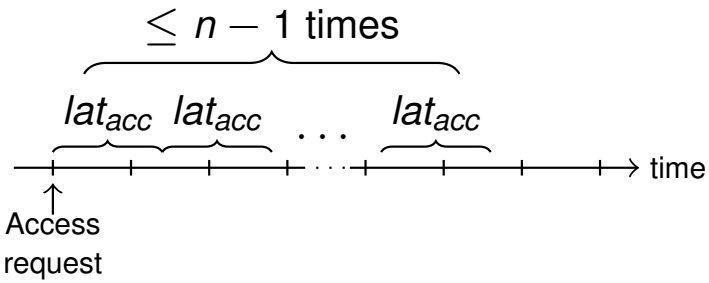
Analysis Derivation Workflow

1. pessimistic **baseline approximation**
2. **identify** system properties
3. **lift** them to approximation
4. **implement** the analysis

Property Lifting Examples

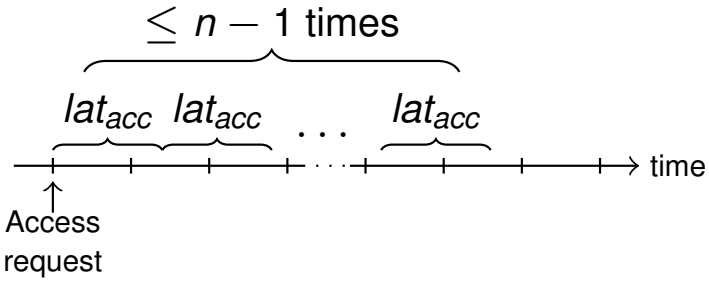
Bounding Shared-Bus Delay

- round-robin bus arbitration
  - n-core processor



Bounding Shared-Bus Delay

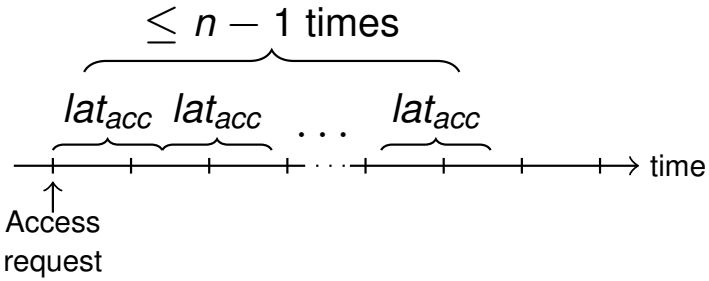
- round-robin bus arbitration
  - n-core processor



- $P(t) =$
  $\#blocked_{C_i}(t) \leq (n-1) \cdot lat_{acc} \cdot \#accesses_{C_i}(t)$

Bounding Shared-Bus Delay

- round-robin bus arbitration
  - n-core processor

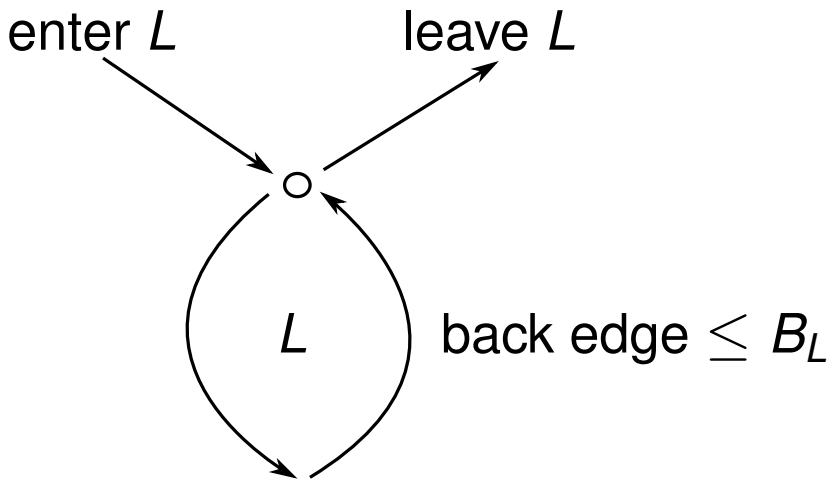


- $P(t) =$
  $\#blocked_{C_i}(t) \leq (n-1) \cdot lat_{acc} \cdot \#accesses_{C_i}(t)$

- $\widehat{P}(\hat{t}) =$
  ${}^{LB}\#blocked_{C_i}(\hat{t}) \leq (n-1) \cdot lat_{acc} \cdot {}^{UB}\#accesses_{C_i}(\hat{t})$

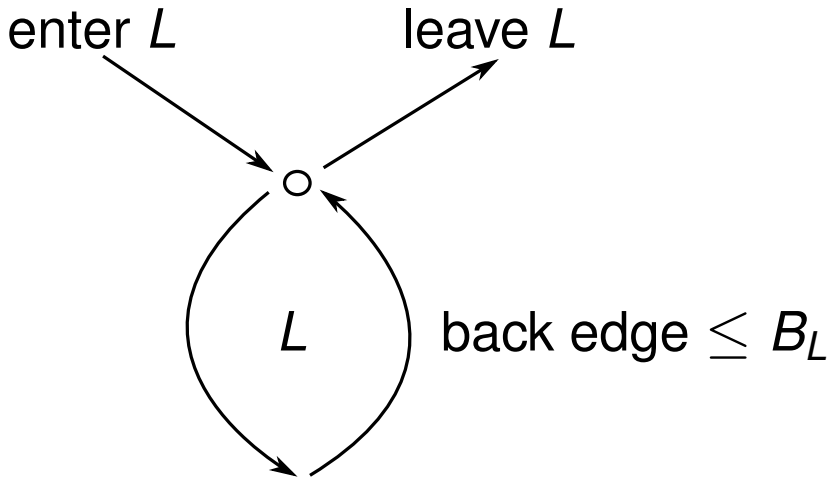
Bounding Loop Iterations

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



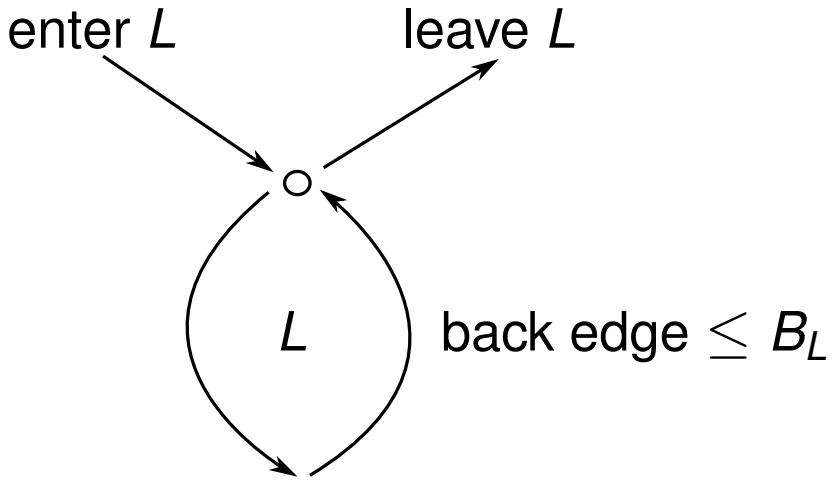
Bounding Loop Iterations

- 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

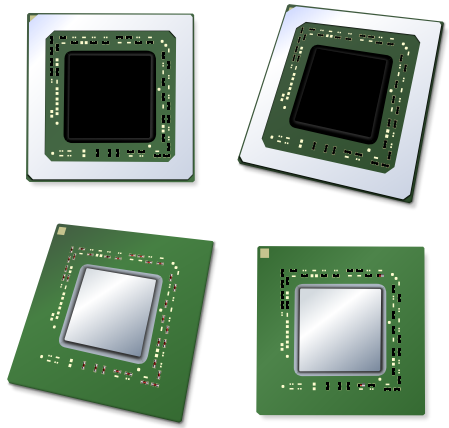
- $P(t) = \#backEdge_L(t) \leq B_L \cdot \#entered_L(t)$

- $\widehat{P}(\hat{t}) = {}^{LB}\#backEdge_L(\hat{t}) \leq B_L \cdot {}^{UB}\#entered_L(\hat{t})$

Experimental Evaluation

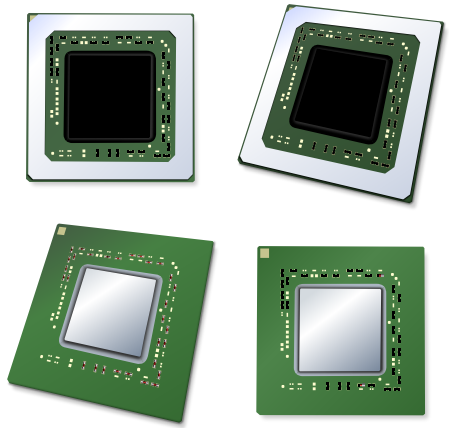
Experimental Setup

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



Experimental Setup

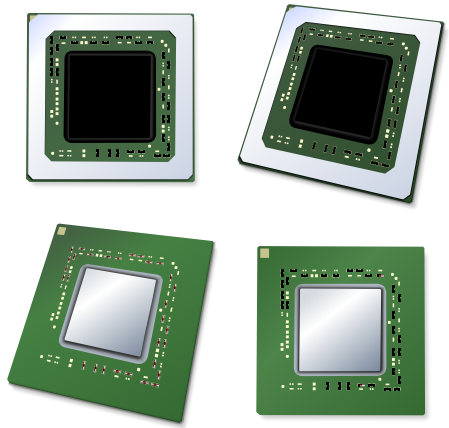
- Hardware platforms
  - ARM® instruction set
  - four processor-core configurations
  - round-robin shared bus
  - SRAM latency: 10 cycles
  - dual-, quad-, and octa-core



- 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

2-Core	in-order execution	out-of-order execution
local instruction scratchpad	3.3%	5.4%
local instruction cache	5.0%	15.9%

Average Analysis-Runtime Increase

Compared to Compositional Analysis

- increasing **complexity** of processor cores

2-Core	in-order execution	out-of-order execution
local instruction scratchpad	3.3%	5.4%
local instruction cache	5.0%	15.9%

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

2-Core	*4-Core*	*8-Core*
15.9%	15.2%	14.9%

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


Altmeyer, S., Davis, R. I., Indrusiak, L. S., Maiza, C., Nélis, V., and Reineke, J. (2015).
A generic and compositional framework for multicore response time analysis.
In *Proceedings of the 23rd International Conference on Real Time Networks and Systems*, pages 129–138.


Chattopadhyay, S., Kee, C., Roychoudhury, A., Kelter, T., Marwedel, P., and Falk, H. (2012).
A unified WCET analysis framework for multi-core platforms.
In *Proceedings of the 18th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 99–108.


Dasari, D., Andersson, B., Nélis, V., Petters, S. M., Easwaran, A., and Lee, J. (2011).
Response time analysis of COTS-based multicores considering the contention on the shared memory bus.
In *Proceedings of the 10th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*, pages 1068–1075.


Dasari, D. and Nélis, V. (2012).
An analysis of the impact of bus contention on the WCET in multicores.
In *Proceedings of the 14th IEEE International Conference on High Performance Computing and Communication & the 9th IEEE International Conference on Embedded Software and Systems*, pages 1450–1457.

References II


Giannopoulou, G., Lampka, K., Stoimenov, N., and Thiele, L. (2012).
Timed model checking with abstractions: Towards worst-case response time analysis in resource-sharing manycore systems.
In *Proceedings of the 10th ACM International Conference on Embedded Software*, pages 63–72.


Jacobs, M., Hahn, S., and Hack, S. (2015).
WCET analysis for multi-core processors with shared buses and event-driven bus arbitration.
In *Proceedings of the 23rd International Conference on Real Time Networks and Systems*, pages 193–202.


Kelter, T. and Marwedel, P. (2014).
Parallelism analysis: Precise WCET values for complex multi-core systems.
In *Revised Selected Papers of the 3rd International Workshop on Formal Techniques for Safety-Critical Systems*, pages 142–158.


Liang, Y., Ding, H., Mitra, T., Roychoudhury, A., Li, Y., and Suhendra, V. (2012).
Timing analysis of concurrent programs running on shared cache multi-cores.
Real-Time Systems, 48:638–680.


Nowotsch, J. (2014).
Interference-sensitive Worst-case Execution Time Analysis for Multi-core Processors.
PhD thesis.


Pellizzoni, R., Schranzhofer, A., Chen, J.-J., Caccamo, M., and Thiele, L. (2010).
Worst case delay analysis for memory interference in multicore systems.
In *Proceedings of the 13th Conference on Design, Automation and Test in Europe*, pages 741–746.

References III

Schliecker, S. and Ernst, R. (2010).
Real-time performance analysis of multiprocessor systems with shared memory.
ACM Trans. Embedded Comput. Syst., 10(2):22.

Schliecker, S., Negrean, M., and Ernst, R. (2009).
Response time analysis in multicore ECUs with shared resources.
IEEE Trans. Industrial Informatics, 5(4):402–413.

Schranzhofer, A., Chen, J.-J., and Thiele, L. (2010).
Timing analysis for TDMA arbitration in resource sharing systems.
In *Proceedings of the 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 215–224.

Schranzhofer, A., Pellizzoni, R., Chen, J.-J., Thiele, L., and Caccamo, M. (2011).
Timing analysis for resource access interference on adaptive resource arbiters.
In *Proceedings of the 17th IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 213–222.

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