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

Michael Jacobs

Department of Computer Science
Saarland University

August 20, 2014

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Multi-Core Processors

- Cores share common HW resources
  - Mostly caches and buses
- Reduces:
  - Weight
  - Energy consumption
  - Production costs
- Thus also interesting for embedded systems
- Downside: Interference
  - Cores no longer behave independently
- Precise WCET analysis is challenging
  - Exact consideration of all interference effects too costly

WCET Analysis for Multi-Core Processors

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Existing approaches not sufficient:
  - Compositionality assumed
  - Cache analysis not integrated with pipeline analysis
  - No dynamic bus arbitration

  ⇒ Modern processors not supported!

State of the Art

- Need for new WCET analyses
- Various HW platforms
  - Each requires specific analysis
- Hand-crafting analyses is tedious
- Each analysis needs soundness proof

Challenges

- We propose a framework
  - Derive analyses according to it
  - Soundness as consequence of derivation

Solution

A Framework for the Derivation of WCET Analyses

for Multi-Core Processors

- Start with sound analysis
  - Similar to single-core analysis
  - Pessimistic about resource sharing
  - But simple to prove sound
- Consider system properties
  - Hold for the concrete system
  - Bound **shared resource interference**
- Lift the system properties
  - To abstraction level of the existing analysis
  - According to **formal criterion**
- Improved analysis
  - Based on existing analysis
  - Incorporates lifted properties
  - Prunes infeasible behavior

Baseline Analysis

+ System Properties

+ Property Lifting

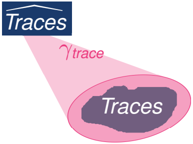
= Improved Analysis

Outline

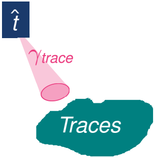
Concrete System and a WCET Analysis

- Concrete-system behavior
  - Set *Traces* of **execution behaviors**
  - $WCET_{C_i} = \max_{t \in Traces} et_{C_i}(t)$

- Existing sound analysis for core C_i
  - Similar to single-core analysis
  - Pessimistic about resource sharing
  - Result forms set $\widehat{Traces}$ of **abstract traces**
  - **Sound**: $\bigcup_{\hat{t} \in \widehat{Traces}} \gamma_{trace}(\hat{t}) \supseteq Traces$
  - **WCET bound**: $\max_{\hat{t} \in \widehat{Traces}} {}^{UB}et_{C_i}(\hat{t}) \geq WCET_{C_i}$

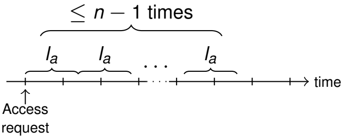


- Problem: infeasible abstract traces
  - $\widehat{Infeas} = \{\hat{t} \mid \hat{t} \in \widehat{Traces} \wedge \gamma_{trace}(\hat{t}) \cap Traces = \emptyset\}$
  - Increase WCET bound
  - Can safely be pruned
  - How to detect them?



System Properties and Lifting Them

- Set of **system properties**
  - $Prop = \{P_1, \ldots, P_p\}$
  - Hold for all execution behaviors:
    $\forall P_k \in Prop : \forall t \in Traces : P_k(t)$



Example: **Round-Robin** bus arbitration

- A Round-Robin property:
  $P_{rr}(t) \Leftrightarrow [\,\#blockedCycles_{C_i}(t) \leq \#accesses_{C_i}(t) \cdot (n-1) \cdot l_a\,]$
- **Lifted** to abstract traces:
  $\widehat{P_{rr}}(\hat{t}) \Leftrightarrow [\,{}^{LB}\#blockedCycles_{C_i}(\hat{t}) \leq {}^{UB}\#accesses_{C_i}(\hat{t}) \cdot (n-1) \cdot l_a\,]$

- Intuition:
  - $\neg\widehat{P_{rr}}(\hat{t}) \Rightarrow [\,\forall t \in \gamma_{trace}(\hat{t}) : \neg P_{rr}(t)\,]$
  - $\neg\widehat{P_{rr}}(\hat{t}) \Rightarrow \hat{t} \in \widehat{Infeas}$

Pruning Infeasible Abstract Traces

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Formal criterion for lifted properties $\widehat{P_k}$

$\forall \hat{t} \in \widehat{Traces}$:
$[\, \exists t \in \gamma_{trace}(\hat{t}) : P_k(t) \,] \Rightarrow \widehat{P_k}(\hat{t})$

- Consequence:
  $\neg \widehat{P_k}(\hat{t}) \Rightarrow \hat{t} \in \widehat{Infeas}$
- Prune infeasible abstract traces:
  $\widehat{LessTraces} = \{\hat{t} \mid \hat{t} \in \widehat{Traces} \wedge \bigwedge_{P_k \in Prop} \widehat{P_k}(\hat{t})\}$
- WCET bound still sound:
  $\max_{\hat{t} \in \widehat{LessTraces}} {}^{UB}et_{C_i}(\hat{t}) \geq WCET_{C_i}$

Outline

Properties Arguing across Core Boundaries

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Example: **work conserving** bus arbitration

- A work conserving property:
  $$P_{wc}(t) \Leftrightarrow [\#blockedCycles_{C_i}(t) \leq \sum_{C_j \in (Cores \setminus \{C_i\})} \#accessCycles_{C_j}(t)]$$
- **Lifted** to abstract traces:
  $$\widehat{P_{wc}}(\hat{t}) \Leftrightarrow [\,{}^{LB}\#blockedCycles_{C_i}(\hat{t}) \leq \sum_{C_j \in (Cores \setminus \{C_i\})} {}^{UB}\#accessCycles_{C_j}(\hat{t})\,]$$

- Abstract traces need to **argue about all cores**
- Introduce **compound abstract traces**
  - $\widehat{Traces} = \widehat{Traces^{C_1}} \times \cdots \times \widehat{Traces^{C_n}}$
  - $\gamma_{trace}(\,(\widehat{t^{C_1}}, \ldots, \widehat{t^{C_n}})\,) = \bigcap_{C_i \in Models} \gamma^{C_i}_{trace}(\widehat{t^{C_i}})$
- Obtain $\widehat{LessTraces}$ as before
- Problem: compound set $\widehat{Traces}$ likely **unmanageably large**

Overapproximating the compound set $\widehat{LessTraces}$

SAARLAND UNIVERSITY COMPUTER SCIENCE

- Idea: Analyze **each core on its own**
  - But consider **cumulative information from** the analysis of **other cores**

- Use **iterative approach**:
  - Approximation variable $\widehat{Approx}^{C_i}$ per core $C_i \in Cores$
  - Initialization: $\widehat{Approx}^{C_i} \leftarrow \widehat{Traces}^{C_i}$
  - Update: $\widehat{Approx}^{C_i} = \{\widehat{t^{C_i}} \mid \widehat{t^{C_i}} \in \widehat{Traces}^{C_i} \wedge \bigwedge_{P_k \in Prop} \widetilde{P_k^{C_i}}(\widehat{t^{C_i}}, (\widehat{Approx}^{C_1}, \ldots, \widehat{Approx}^{C_n}))\}$

- The iterative approach...
  - **overapproximates projections** of $\widehat{LessTraces}$
    $\forall C_i \in Cores: \quad \widehat{Approx}^{C_i} \supseteq \pi^{C_i}(\widehat{LessTraces})$
  - is an **anytime algorithm**
  - allows **any iteration strategy**

Overapproximating the compound set $\widehat{LessTraces}$

- Idea: Analyze **each core on its own**
  - But consider **cumulative information from** the analysis of **other cores**

- Use **iterative approach**:
  - Approximation variable $\widehat{Approx}^{C_i}$ per core $C_i \in Cores$
  - Initialization: $\widehat{Approx}^{C_i} \leftarrow \widehat{Traces}^{C_i}$
  - Update: $\widehat{Approx}^{C_i} = \{\widehat{t^{C_i}} \mid \widehat{t^{C_i}} \in \widehat{Traces}^{C_i} \wedge \bigwedge_{P_k \in Prop} \widetilde{P_k^{C_i}}(\widehat{t^{C_i}}, (\widehat{Approx}^{C_1}, \ldots, \widehat{Approx}^{C_n}))\}$

    specially lifted for iterative approach

- The iterative approach...
  - **overapproximates projections** of $\widehat{LessTraces}$

    $\forall C_i \in Cores : \widehat{Approx}^{C_i} \supseteq \pi^{C_i}(\widehat{LessTraces})$
  - is an **anytime algorithm**
  - allows **any iteration strategy**

Lifting Properties for the Iterative Approach

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- **Further lift** properties $\widehat{P_k}$ to $\widetilde{P_k^{C_i}}$ per core $C_i \in Cores$
- According to **formal criteria**:
  - **Soundness criterion**
  - **Monotonicity criterion**

Example: further lifted versions of $\widehat{P_{wc}}$

- $\widetilde{P_{wc}^{C_i}}(\widehat{t^{C_i}}, (\widehat{Approx^{C_1}}, \ldots, \widehat{Approx^{C_n}}))$
  $\Leftrightarrow [\, {}^{LB}\#blockedCycles_{C_i}^{C_i}(\widehat{t^{C_i}}) \leq$
  $\sum_{C_j \in (Cores \setminus \{C_i\})} \max_{\widehat{t^{C_j}} \in \widehat{Approx^{C_j}}} {}^{UB}\#accessCycles_{C_j}^{C_j}(\widehat{t^{C_j}})\,]$
- $\forall C_j \neq C_i : \;\; \widetilde{P_{wc}^{C_j}}(\widehat{t^{C_j}}, (\widehat{Approx^{C_1}}, \ldots, \widehat{Approx^{C_n}})) \Leftrightarrow 1$

Lifting Properties for the Iterative Approach

- **Further lift** properties $\widehat{P_k}$ to $\widetilde{P_k^{C_i}}$ per core $C_i \in Cores$
- According to **formal criteria**:
  - **Soundness criterion**
  - **Monotonicity criterion**

Non-compound abstract trace (Core C_i)

Example: further lifted versions of $\widehat{P_{wc}}$

- $\widetilde{P_{wc}^{C_i}}(\widehat{t^{C_i}}, (\widehat{Approx^{C_1}}, \ldots, \widehat{Approx^{C_n}}))$
  $\Leftrightarrow [\ {}^{LB}\#blockedCycles_{C_i}^{C_i}(\widehat{t^{C_i}}) \leq$
  $\sum_{C_j \in (Cores \setminus \{C_i\})} \max_{\widehat{t^{C_j}} \in \widehat{Approx^{C_j}}} {}^{UB}\#accessCycles_{C_j}^{C_j}(\widehat{t^{C_j}})\ ]$
- $\forall C_j \neq C_i :\ \widetilde{P_{wc}^{C_j}}(\widehat{t^{C_j}}, (\widehat{Approx^{C_1}}, \ldots, \widehat{Approx^{C_n}})) \Leftrightarrow 1$

Lifting Properties for the Iterative Approach

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- **Further lift** properties $\widehat{P_k}$ to $\widetilde{P_k^{C_i}}$ per core $C_i \in Cores$
- According to **formal criteria**:
  - **Soundness criterion**
  - **Monotonicity criterion**

Abstract traces of the other cores

Example: further lifted versions of $\widehat{P_{wc}}$

- $\widetilde{P_{wc}^{C_i}}(\widehat{t^{C_i}}, (\widehat{Approx^{C_1}}, \ldots, \widehat{Approx^{C_n}}))$
  $\Leftrightarrow [\,{}^{LB}\#blockedCycles_{C_i}^{C_i}(\widehat{t^{C_i}}) \leq$
  $$\sum_{C_j \in (Cores \setminus \{C_i\})} \max_{\widehat{t^{C_j}} \in \widehat{Approx^{C_j}}} {}^{UB}\#accessCycles_{C_j}^{C_j}(\widehat{t^{C_j}})\,]$$
- $\forall C_j \neq C_i: \quad \widetilde{P_{wc}^{C_j}}(\widehat{t^{C_j}}, (\widehat{Approx^{C_1}}, \ldots, \widehat{Approx^{C_n}})) \Leftrightarrow 1$

Lifting Properties for the Iterative Approach

SAARLAND UNIVERSITY COMPUTER SCIENCE

- **Further lift** properties $\widehat{P_k}$ to $\widetilde{P_k^{C_i}}$ per core $C_i \in Cores$
- According to **formal criteria**:
  - **Soundness criterion**
  - **Monotonicity criterion**

Cumulative information

Example: further lifted versions of $\widehat{P_{wc}}$

- $\widetilde{P_{wc}^{C_i}}(\widehat{t^{C_i}}, (\widehat{Approx^{C_1}}, \ldots, \widehat{Approx^{C_n}}))$
  $\Leftrightarrow [\,{}^{LB}\#blockedCycles_{C_i}^{C_i}(\widehat{t^{C_i}}) \leq \sum_{C_j \in (Cores \setminus \{C_i\})} \max_{\widehat{t^{C_j}} \in \widehat{Approx^{C_j}}} {}^{UB}\#accessCycles_{C_j}^{C_j}(\widehat{t^{C_j}})\,]$
- $\forall C_j \neq C_i : \widetilde{P_{wc}^{C_j}}(\widehat{t^{C_j}}, (\widehat{Approx^{C_1}}, \ldots, \widehat{Approx^{C_n}})) \Leftrightarrow 1$

Outline

Advantages of Using our Framework

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- A framework for the derivation of WCET analyses for multi-core processors
- Advantages:
  - Standard derivation procedure
    - Does not inherently rely on the restricting assumptions of previous approaches
  - Soundness guarantee
  - Trade-off between efficiency and precision
  - Modular design of WCET analyses
  - Assumptions about the system always explicit

How Does our Framework Compare to Existing Analyses?

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Not at all!
  - It is no analysis, but a framework to derive analyses.
- How do derived analyses compare to existing analyses?
  - It depends!
  - Many parameters influence precision and performance:
    - ★ Baseline analysis
    - ★ Considered system properties
    - ★ Iterative or non-iterative approach
    - ★ Lifting decisions
    - ★ Implementation details
  - Existing analyses can likely also be derived
    - ★ Although we did not try to so far
- Many aspects still **very abstract**
  - E.g. of which form *Traces* and $\widehat{Traces}$ are
  - Just concrete enough to show:
    - ★ the technical soundness of derived analyses
    - ★ the high-level use of our framework, not of a particular derived analysis

Future Work

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Framework so far only conceptual
- Step towards actual analysis implementations
  - **Execution behaviors** are **sequences of HW states**
    - as obtained by cycle-wise transition
  - **Abstract traces** are **sequences of abstract states**
    - abstract states as in abstract-interpretation-based WCET analysis
  - Further **lift properties** from abstract traces **to abstract states**
    - In a way that: if a property holds for an abstract trace, its abstract state version holds for all abstract states in the trace
  - Goal: **prune abstract states** for which a property does not hold
- Incorporate **arrival curves**
  - To express cumulative information in the iterative approach
- Experiment with different concrete system properties
  - E.g. arguing about:
    - shared caches
    - cache coherence