WCET Analysis for Multi-Core Processors

Michael Jacobs, Sebastian Hack, Jan Reineke, Reinhard Wilhelm

Department of Computer Science
Saarland University

February 28, 2013
Outline

1. WCET Analysis
2. Multi-Core Processors
3. Bounding Bus Interference
4. Bounding Cache Interference
5. A Classification of Approaches to Interference Bounding
6. Summary
Need for WCET Analysis

- Embedded systems
- Safety-critical applications
  - E.g. in automotive or medical industry
- Strict timing requirements
  - Dictated by the physical environment
- Sound execution time bounds for programs needed

⇒ Worst-Case Execution Time (WCET) analysis
Execution Time of a Computer Program

- Execution time
  - Number of processor cycles
  - Needed to execute a given program
  - On a given hardware platform

⇒ Bounds are specific to a hardware platform

- Execution time depends on
  - Program input
    Which path through the program is taken?
  - Initial system state
    E.g. load of a cached memory block faster

⇒ Sound bounds must hold for all possible combinations

![Diagram of execution time vs. frequency with bounds labeled as LB, BCET, WCET, and UB]
Exact Behavior of a Computer System
A Formalization

- Exact behavior of a Core under Consideration (CuC)
  - Set of actual system states: $S$
  - Transitions under cycles of the CuC: $\text{Transitions} \subseteq S \times S$
  - A trace describes one execution behavior

- Not suitable for timing analysis
  - Realistic systems are complex
  - Large space of initial system states and program inputs
    - Exhaustive simulation is no option
  - Many details irrelevant to timing

$\Rightarrow$ Abstract timing models needed
Abstract timing model of the CuC

- Set of abstract system states: \( \hat{S} \)
- Abstract cycle semantics of the CuC:
  \( \text{Transitions} \subseteq \hat{S} \times \hat{S} \)

An abstract state may describe several concrete states

\[
\begin{align*}
\hat{s}_0 \quad &\quad \hat{s}_1 \\
\downarrow \gamma &\quad \downarrow \gamma \\
S_0, S_2 &\quad S_1, S_3
\end{align*}
\]

\( \text{Transitions} \) subsumes \( \text{Transitions} \)

An abstract trace may describe several concrete traces
Infeasible Traces

- Abstract models over-estimate the concrete execution behavior

- Example
  - Concrete system
    \[ s_0 \rightarrow s_1, \ s_2 \rightarrow s_3 \]
  - Abstraction
    \[
    \begin{align*}
    \hat{s}_0 & \rightarrow \hat{s}_1 \\
    s_0, s_2 & \rightarrow s_1, s_3
    \end{align*}
    \]
  - Abstract model
    \[ \hat{s}_0 \rightarrow \hat{s}_1 \]
  - Described concrete traces
    \[ s_0 \rightarrow s_1, s_2 \rightarrow s_3, \ s_0 \rightarrow s_3, s_2 \rightarrow s_1 \]

- Abstraction has introduced infeasible traces
Sound and precise analyses exist

Already high analysis complexity
  ▶ Uncertainty about successor states
    ★ Non-determinism introduced by abstraction
    ★ Many case splits needed
  ▶ State space explosion

\[
\hat{s}_0 \quad \hat{s}_1 \quad \hat{s}_2
\]

cache miss

\[
\hat{s}_1 \quad \hat{s}_2
\]

cache hit
Outline

1  WCET Analysis
2  Multi-Core Processors
3  Bounding Bus Interference
4  Bounding Cache Interference
5  A Classification of Approaches to Interference Bounding
6  Summary
Transition to Multi-Core Processors

- **Motivation**
  - Reduce price, weight and energy consumption
    - Compared to multiple single-core processors
  - Use processors from the mass market
    - Further price reduction

- **Several cores share common resources**
  - Buses
  - Caches

- **Disadvantage: Interference on shared resources**
  - Subject of WP4 in R2
Bus Arbitration

- Shared bus
- One core at a time allowed to access
- Bus arbitration avoids bus conflicts
  - Grants access to one core
  - Blocks other cores requesting access
- Different arbitration protocols
  - Round-robin
  - FCFS (First-Come-First-Serve)
  - TDMA (Time-Division-Multiple-Access)
Bus Interference

- Bus interference
  - Arbitration influences a core’s behavior
  - Core behaves differently than with dedicated bus
Cache Sharing

- Shared cache
- Several cores use the same cache lines
- Core A evicts a block loaded by core B
  - Core B may suffer an additional cache miss
- Core A preloads a block for core B
  - Core B enjoys an additional cache hit
Cache Interference

- **Cache capacity interference**
  - These additional misses and hits influence a core’s behavior
  - Core behaves differently than with dedicated cache lines

- **Cache access interference**
  - Similar to bus interference
  - Not considered for now
  - Assume cache access resolved through shared bus arbitration
Challenges for WCET Analysis

- Traditional WCET analysis
  - Only considers one program executed on one core
- Analysis for multi-core processors has to consider interference
  - Bus interference
  - Cache interference
- Goal: Analyze program on one core
- However, programs on other cores may influence its behavior

→ Need for special timing analysis techniques
Existing Approaches

WCET Analysis for Multi-Core Processors

- Only applicable to compositional processor architectures
  - WCET analysis as if in isolation
  - Add penalties
    - Cycles blocked at the bus
    - Additional cycles on cache misses
  - Such processors are rarely available
- Only consider a very abstract model of computation
  - Based on superblocks
    
    \[
    \begin{array}{ccc}
    s_0 & s_1 & s_2 \\
    \text{exec}_0 = 4 & \text{exec}_1 = 6 & \text{exec}_2 = 3 \\
    \mu_0 = 2 & \mu_1 = 3 & \mu_2 = 3
    \end{array}
    \]

- Often only consider a single shared resource
  \[\Rightarrow\] Too restrictive
Considering Interference

- First approach
  - Analyzing the exact system behavior of all cores simultaneously
- Cores access shared resources
  - Different access interleavings exhibit different timing
- Must consider all interleavings
- State space explosion
  - Similar to verification of parallel programs
- Need for abstraction
  - Which is the right level of abstraction?
Precision versus Complexity

Exact system behavior

Coarse abstraction
Precision versus Complexity

- Precision
- Exact system behavior
- Coarse abstraction

Where is a good trade-off?
Precision versus Complexity

Exact system behavior

Coarse abstraction

Precision

Complexity
Precision versus Complexity

- Precision
- Complexity
- Exact system behavior
- Coarse abstraction

- Where is a good trade-off?
Coarse Abstraction + Interference Bounds

- Coarse abstraction as baseline
  - Consider only the analyzed core
  - Unknown state of the rest of the system
  - A lot of infeasible interference

- Improve by bounds on the interference
  - Precise enough
  - Efficiently obtained

- Exclude infeasible traces
Outline

1 WCET Analysis
2 Multi-Core Processors
3 Bounding Bus Interference
4 Bounding Cache Interference
5 A Classification of Approaches to Interference Bounding
6 Summary
Bus Interference on a Concrete System

- A core executes a program part
- Execution is blocked for a number of cycles at the bus
- For many systems, this number can be bounded
- Bounds can be based on
  - The arbitration logic
  - The concurrent access behavior
Number of Blocked Cycles on a Concrete System

Bounds Based on the Arbitration Logic

- Example: Round-robin arbitration
- Worst-case scenario [Pellizzoni and Caccamo, 2010]
  - All other cores are granted one access first
- An access may last at most $l_a$ cycles
- Invariant for every feasible bus access
  - $\#\text{blocked}(access) \leq (n - 1) \times l_a$

\[
\begin{array}{cccccc}
\text{time} & \cdots & \text{time} \\
\hline
l_a & l_a & \cdots & l_a \\
\text{Access request} & \cdots & \text{Access request} \\
\end{array}
\]
Number of Blocked Cycles on a Concrete System
Bounds Based on the Arbitration Logic

- Example: Round-robin arbitration
- Worst-case scenario [Pellizzoni and Caccamo, 2010]
  - All other cores are granted one access first
- An access may last at most $l_a$ cycles
- Invariant for every feasible bus access
  - $\#\text{blocked}(\text{access}) \leq (n - 1) \times l_a$

\[
\leq n - 1 \text{ times}
\]
\[
\begin{array}{cccc}
 & l_a & l_a & \ldots & l_a \\
\hline
\end{array}
\]

- Lift to an abstract trace
  - $\forall t \in \gamma(\text{trace}) :$
    - feasible$(t) \Rightarrow \#\text{blocked}(t) \leq UBNumAccesses(\text{trace}) \times (n - 1) \times l_a$
Consider event-driven bus arbitration

- Only blocked if another core has been granted access

Invariant for every feasible concrete trace

- It is not blocked longer than other cores access the bus
- Corresponds to the constraint

\[
\#\text{blocked}(\text{trace}) \leq \#\text{concurrentBusAccesses}(\text{trace})
\]
Number of Blocked Cycles on a Concrete System
Bounds Based on the Concurrent Access Behavior

- Consider event-driven bus arbitration
  - Only blocked if another core has been granted access
- Invariant for every feasible concrete trace
  - It is not blocked longer than other cores access the bus
  - Corresponds to the constraint
    \[
    \# \text{\texttt{blocked}}(\text{\texttt{trace}}) \leq \# \text{\texttt{concurrentBusAccesses}}(\text{\texttt{trace}})
    \]
- Lift to an abstract trace
  - Exact amount of concurrent bus access cycles not known
  - Pre-analyze the co-running tasks for an upper bound
    [Wandeler et al., 2006, Pellizzoni et al., 2010]
    - Per number of execution cycles
    - \( \text{UBNumConcurrentBusAccesses} : \mathbb{N} \to \mathbb{N} \)
    - \( \forall t \in \gamma(\hat{\text{\texttt{trace}}}) : \)
      \[
      \text{feasible}(t) \Rightarrow \# \text{\texttt{blocked}}(t) \leq \text{UBNumConcurrentBusAccesses}(\# \text{\texttt{cycles}}(\hat{\text{\texttt{trace}}}))
      \]
Bus Interference in an Abstract Model

- Consider an abstract trace

\[ \geq 4 \geq 3 \gamma \]

And the concrete traces it describes

Number of blocked cycles known for each concrete trace

Annotate abstract trace with a lower bound on them

Michael Jacobs

Without looking at the concrete traces

WCET Analysis for Multi-Core Processors

February 28, 2013 24 / 33
Bus Interference in an Abstract Model

- Consider an abstract trace

- And the concrete traces it describes
Bus Interference in an Abstract Model

- Consider an abstract trace

\[ \geq 4 \geq 3 \geq 4 \gamma \]

- And the concrete traces it describes

- Number of blocked cycles known for each concrete trace
Bus Interference in an Abstract Model

- Consider an abstract trace

\[ \overline{\ldots} \overline{\ldots} \geq 4 \quad \gamma \]

- And the concrete traces it describes

- **Number of blocked cycles** known for each concrete trace

- Annotate abstract trace with a **lower bound** on them
Bus Interference in an Abstract Model

- Consider an abstract trace

\[ \hat{\gamma} \geq 3 \]

- And the concrete traces it describes

  - Number of blocked cycles known for each concrete trace
  - Annotate abstract trace with a lower bound on them
  - Approximate coarse lower bound
Bus Interference in an Abstract Model

Consider an abstract trace

\[ \vdash \vdash \vdash \vdash \vdash \vdash \vdash \vdash \geq 3 \]

And the concrete traces it describes

■ Number of blocked cycles known for each concrete trace
■ Annotate abstract trace with a lower bound on them
■ Approximate coarse lower bound
  ▶ Without looking at the concrete traces
Detecting Infeasible Abstract Traces

- Set of abstract traces for a program part

- Annotate lower bounds on the number of blocked cycles
- Annotate upper bounds on the number of blocked cycles

- Derived for the concrete system and the arbitration protocol
- Hold for all feasible concrete traces described
- Lower bound and upper bound contradict
- All concrete traces described are infeasible
- Then so is the abstract trace

- Remove infeasible abstract traces
Detecting Infeasible Abstract Traces

- Set of abstract traces for a program part
  - \[\overrightarrow{\ldots} \overrightarrow{\ldots} \overrightarrow{\ldots} \overrightarrow{\ldots} \overrightarrow{\ldots} \geq 3\]
  - \[\overrightarrow{\ldots} \overrightarrow{\ldots} \overrightarrow{\ldots} \overrightarrow{\ldots} \overrightarrow{\ldots} \geq 6\]
  - \[\overrightarrow{\ldots} \overrightarrow{\ldots} \overrightarrow{\ldots} \overrightarrow{\ldots} \overrightarrow{\ldots} \geq 5\]
  - \[\overrightarrow{\ldots} \overrightarrow{\ldots} \overrightarrow{\ldots} \overrightarrow{\ldots} \overrightarrow{\ldots} \geq 0\]

- Annotate lower bounds on the number of blocked cycles

- Derived for the concrete system and the arbitration protocol
- Hold for all feasible concrete traces described
- Lower bound and upper bound contradict
- All concrete traces described are infeasible
- Then so is the abstract trace
Detecting Infeasible Abstract Traces

- Set of abstract traces for a program part

- Annotate lower bounds on the number of blocked cycles
- Annotate upper bounds on the number of blocked cycles
  - Derived for the concrete system and the arbitration protocol
  - Hold for all feasible concrete traces described
Detecting Infeasible Abstract Traces

- Set of abstract traces for a program part
  
  \[ \begin{align*}
  &\geq 3 \quad \leq 4 \\
  &\geq 6 \quad \leq 3 \\
  &\geq 5 \quad \leq 4 \\
  &\geq 0 \quad \leq 3
  \end{align*} \]

- Annotate lower bounds on the number of blocked cycles
- Annotate upper bounds on the number of blocked cycles
  
  - Derived for the concrete system and the arbitration protocol
  - Hold for all feasible concrete traces described

- Lower bound and upper bound contradict
  
  - All concrete traces described are infeasible
  - Then so is the abstract trace
Detecting Infeasible Abstract Traces

- Set of abstract traces for a program part

\[
\begin{align*}
\hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} & \geq 3 \leq 4 \\
\hat{\bullet} \cdot \hat{\bullet} \cdot \hat{\bullet} \cdot \hat{\bullet} \cdot \hat{\bullet} & \geq 6 \leq 3 \\
\hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} & \geq 5 \leq 4 \\
\hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} & \geq 0 \leq 3
\end{align*}
\]

- Annotate lower bounds on the number of blocked cycles
- Annotate upper bounds on the number of blocked cycles
  - Derived for the concrete system and the arbitration protocol
  - Hold for all feasible concrete traces described

- Lower bound and upper bound contradict
  - All concrete traces described are infeasible
  - Then so is the abstract trace

- Remove infeasible abstract traces

\[
\begin{align*}
\hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} \cdot \hat{\circ} & \geq 3 \leq 4 \\
\hat{\bullet} \cdot \hat{\bullet} \cdot \hat{\bullet} \cdot \hat{\bullet} \cdot \hat{\bullet} & \geq 0 \leq 3
\end{align*}
\]
Outline

1 WCET Analysis
2 Multi-Core Processors
3 Bounding Bus Interference
4 Bounding Cache Interference
5 A Classification of Approaches to Interference Bounding
6 Summary
Bounding Cache Interference

- Assume *only* cache capacity interference
  - Cache not accessed by two cores at the same time
- Coarse abstraction
  - Each access could *hit* or *miss* the cache
- Goal: Predict more cache hits
  - Exclude some case splits

![Diagram showing cache misses and hits]

\[
\hat{s}_0, \hat{s}_1, \hat{s}_2
\]

- cache miss
- cache hit
Shared Cache Analysis
Independent of Co-Running Tasks

- Analysis as if on a single-core processor [Ferdinand and Wilhelm, 1997]
  - Is an accessed block still a cache hit?
    - Consider maximum time since last access
    - How many changes to the cache could arbitrary programs on other cores maximally make in that time?
    - Is it enough to evict the cache block?
    - If not, still a cache hit

- Very conservative if co-running tasks do not make full use of the cache
Shared Cache Analysis
Depending on Co-Running Tasks

- Same idea as before
- Slight change to the classification
  - How many changes to the cache can the co-running tasks maximally make in that time?
- Use an upper bound on the concurrent cache access behavior
  - Per number of execution cycles
- Pre-analyze the concurrent cores for this bound
Outline

1. WCET Analysis
2. Multi-Core Processors
3. Bounding Bus Interference
4. Bounding Cache Interference
5. A Classification of Approaches to Interference Bounding
6. Summary
A Classification
Approaches to Interference Bounding

<table>
<thead>
<tr>
<th>Bus interference</th>
<th>Cache interference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent of co-running tasks</td>
<td>Based on arbitration protocol</td>
</tr>
<tr>
<td>Depending on co-running tasks</td>
<td>Based on concurrent access behavior</td>
</tr>
<tr>
<td></td>
<td>Shared cache analysis independent of co-running tasks</td>
</tr>
<tr>
<td></td>
<td>Shared cache analysis depending on co-running tasks</td>
</tr>
</tbody>
</table>
Outline

1. WCET Analysis
2. Multi-Core Processors
3. Bounding Bus Interference
4. Bounding Cache Interference
5. A Classification of Approaches to Interference Bounding
6. Summary
Summary

- WCET analysis for multi-core processors is important
- There is potential for improvements
- Find good trade-off
  - Precision
  - Complexity
- Considering co-running tasks may lead to more precise results

Outlook
- Implement a WCET analysis for a multi-core processor
- Evaluate different levels of precision
Fast and efficient cache behavior prediction.

Impact of peripheral-processor interference on WCET analysis of real-time embedded systems.

Worst case delay analysis for memory interference in multicore systems.

System architecture evaluation using modular performance analysis: a case study.