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

Michael Jacobs, Sebastian Hahn, Sebastian Hack
Saarland University, Saarbrücken, Germany
Email: jacobs@cs.uni-saarland.de, sebastian.hahn@cs.uni-saarland.de, hack@cs.uni-saarland.de

***Abstract*—Multi-core processors share common hardware resources between several processor cores. As a consequence, the performance of one processor core is influenced by the programs executed on the concurrent cores. We refer to this phenomenon as *shared-resource interference*. An explicit consideration of all such interference effects is in general combinatorially infeasible. This makes a precise worst-case execution time (WCET) analysis for multi-core processors challenging.**

In order to reduce the complexity, WCET analyses for multi-core processors coarsely approximate the behavior of the considered applications. However, current approaches are only applicable to rather restricted classes of hardware platforms. We propose a framework for the derivation of WCET analyses for multi-core processors. It relaxes the restricting assumptions that existing approaches are based on.

The derivation starts from a WCET analysis that makes maximally pessimistic assumptions about the shared-resource interference. More precise interference bounds for the concrete system are subsequently lifted to the approximation of the analysis. The lifted bounds are finally incorporated in the analysis in order to model the interference in a more precise way.

I. Introduction

For a timing-critical application it is important that the time needed to deliver the results of its calculations does not exceed a deadline dictated by the physical environment. A timing-critical application may consist of several programs that interact. Knowledge about the worst-case execution time (WCET) [1] of each such program allows us to verify the timeliness of the overall application. It is safe to replace the WCET of a program by an upper bound on its execution times (a so-called WCET bound) in this verification step. However, the timeliness of an application can often only be verified if the WCET bounds are relatively tight. WCET analyses are used for the calculation of WCET bounds.

The execution times of a program depend on the possible execution behaviors at the micro-architectural level of the processor that executes the program. Modern processors are too complex to exhaustively simulate or measure the execution times of all possible behaviors. WCET analyses for those processors need to approximate some of the micro-architectural details in order to reduce the inherent complexity [2], [3]. Approximation often comes at the cost of a less tight WCET bound.

The use of multi-core processors can reduce the weight, the energy consumption and the production costs of computer systems. Hence, they are likely to also be used for timing-critical applications in the long run.

However, multi-core processors consist of several processor cores, which share common resources such as buses or caches. The resource sharing has a significant impact on the overall performance of a system [4] because the cores compete for the shared resources. For example, an access request to a shared bus may be blocked for some cycles before it is granted because a concurrent core is granted access first. This effect is commonly referred to as *shared-resource interference*.

The WCET analysis of programs executed on multi-core processors needs to take into account all possible interference effects due to resource sharing. An exact consideration of all such effects requires in general an exhaustive enumeration of all possible interleavings of accesses to the shared resources by the different processor cores. Such an enumeration is combinatorially infeasible.

Most of the current approaches to WCET analysis for multi-core processors [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16] try to find a level of approximation that avoids this complexity without sacrificing precision too much. Unfortunately, they are restricted to Time-Division-Multiple-Access (TDMA) bus arbitration or not sound in the presence of indirect interference effects, which most modern multi-core platforms exhibit.

Contributions

We propose a framework for the derivation of WCET analyses for multi-core processors. An instance of our framework—derived according to the criteria proposed in this paper—is guaranteed to be a *sound* WCET analysis. The derivation starts from a baseline WCET analysis that makes maximally pessimistic assumptions about the shared-resource interference. We can infer more precise interference bounds from the specification of the concrete system. Lifting these bounds to the approximation of the baseline analysis avoids overly pessimistic assumptions about the interference.

Our iterative overapproximation analyzes each processor core on its own and still incorporates cumulative information about the concurrent cores in the lifted interference bounds. In this way, it finds a trade-off between the performance of analyzing each core in isolation and the precision of simultaneously considering all processor cores.

Our framework has been successfully used in the development of a novel analysis [17] that avoids the restrictions of the existing approaches.

II. Related Work

Most approaches [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15] to WCET analysis or response time analysis for multi-core processors rely on compositionality [18] in the sense that they start with a timing analysis that ignores the shared-resource interference. Subsequently, they add bounds on the direct interference effects to their results. In modern micro-processors, however, the overall impact of the interference can exceed the direct interference effects [19]. Thus, these approaches are not applicable to current hardware platforms.

An approach by Chattopadhyay et al. [16] supports complex processor core pipelines. It is restricted to TDMA bus arbitration. Most multi-core processors on the market, however, implement event-driven bus arbitration protocols.

A recent approach by Kelter and Marwedel [20] supports complex multi-core processors equipped with event-driven bus arbitration. However, it relies on the enumeration of all interleavings of accesses to the shared bus by the different cores. Therefore, we expect it to not scale to realistic application scenarios.

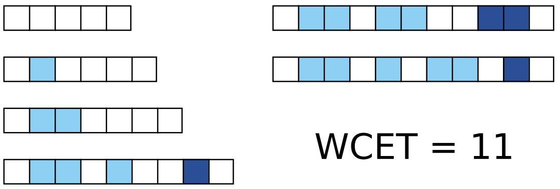

Figure 1: All six behaviors of an example program executed on a hardware platform. The program has a WCET of eleven time units.

A novel analysis developed by our group [17] overcomes the restrictions and simplifying assumptions of previous approaches. To the best of our knowledge, it is the first approach to the calculation of co-runner-sensitive WCET bounds that scales to multi-core processors with out-of-order execution and event-driven bus arbitration.

This paper presents the concepts we applied during the derivation of our novel analysis. They are embedded in a general and formally sound framework for the derivation of WCET analyses for multi-core processors.

III. Motivation

We motivate the key principle of our framework by considering an example program executed on a hardware platform. Figure 1 shows all six possible execution behaviors of the program. Each sequence of boxes represents one execution behavior. White boxes stand for time units of non-interfered execution. The boxes colored in light blue represent *direct interference effects* like cycles blocked at a shared bus or needed to serve a miss in the shared cache. Dark boxes denote the prolonging effects of timing anomalies that are a consequence of earlier interference. A processor core pipeline might, for example, only speculate in a particular situation if it is blocked at the shared bus. If the prediction turns out as false, the execution time is prolonged by more than the blocked cycles. Such *indirect interference effects* can be observed in modern multi-core processors [19]. Sound WCET analyses for such platforms have to take these indirect effects into account.

The example program has a WCET of eleven time units as its longest execution behavior takes this long.

In this example, we assume that no execution behavior of the example program can exhibit more than five direct interference effects. Such assumptions can for instance be inferred from the specification of a bus arbiter if the actual set of execution behaviors is unknown.

A. Classical Compositional Analysis

Existing compositional analyses [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15] first calculate a basic timing bound that does not take into account behaviors exhibiting interference effects. Subsequently, they add an upper bound on the direct interference effects to the result. This principle is depicted in Figure 2. The longest behavior without interference takes five time units. The maximum of five direct interference effects is subsequently added.

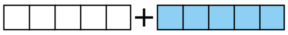

Figure 2: Compositional analyses typically only consider behaviors without interference. Subsequently, they add an upper bound on the direct interference effects. In the presence of indirect effects, this is unsound.

This shows that the common way of decomposing analyses is efficient but unsound in the presence of indirect interference effects. In order to be sound, an upper bound on the indirect effects has to be additionally incorporated.

The indirect interference effects highly depend on the interaction between the shared resources and the processor cores. As this interaction is typically not considered by compositional analyses, they can at best provide very pessimistic bounds on the indirect interference effects. In our example system, the amount of indirect interference effects cannot exceed the amount of direct ones. Note that this is hard or impossible to show for real-world hardware, for example due to domino effects [21]. The analysis results in a high overestimation of the WCET as shown in Figure 3.

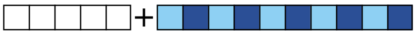

Figure 3: In order to be sound, such a decomposition of timing analysis would have to pessimistically assume that each direct effect leads to an indirect effect. However, this results in a high overestimation.

As a consequence, the common way of decomposing timing analysis is not able to provide sound and precise WCET bounds for modern multi-core processors.

B. Key Principle of our Framework

The derivation of a WCET analysis in our framework starts from an overapproximation of all behaviors of the program. This overapproximation is maximally pessimistic with respect to the shared-resource interference. Each access request to a shared bus can, for example, be blocked arbitrarily long before being granted by the arbiter. Similarly, each access to a shared cache can be a hit or a miss. Figure 4 shows such an overapproximation for our example program. It contains two infeasible behaviors that cannot be observed when actually executing the program (dashed box).

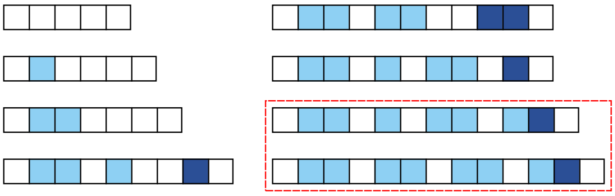

Figure 4: The pessimistic overapproximation of all behaviors of the example program contains two infeasible behaviors (dashed box).

We exploit the upper bound on the direct interference effects to prune the two infeasible behaviors, which exhibit more than five direct interference effects. The remaining execution behaviors result in an exact WCET bound of eleven time units.

In this way, our framework supports the development of timing analyses that explicitly model the impact of the interference on the processor cores and, thus, precisely bound the indirect interference effects.

Note that the pessimistic overapproximation is only a conceptual starting point. The implementation of the analysis derived from our framework will not materialize this overapproximation. Instead, it can directly leave out many of the infeasible behaviors during analysis.

Our example program only exhibits six execution behaviors. Programs executed on real-world hardware platforms, however, exhibit too many execution behaviors to exhaustively enumerate them. Thus, it is common to approximate some of the micro-architectural details [2]. In the next section, we formally show how the principle presented above can be applied to such approximations.

IV. Property Lifting

This section describes the concept of property lifting, which is the central part of our framework.

A. Concrete Execution Behavior and Time

We consider a multi-core processor consisting of the set *Cores* of n processor cores.

$$\mathit{Cores} = \{C_1, \ldots, C_n\}$$

For simplicity, we assume that each core only runs one program and that each program may at most be executed once per system run. This restriction is not inherent to our framework but only made to simplify the notation. For a more detailed discussion on how to overcome this restriction, we refer to [22]. In the following, we use the term *system* to refer to the combination of the hardware including the multi-core processor and the software executed on it.

The system may exhibit different execution behaviors depending on its initial state, external input parameters and clock drift effects. Let *Traces* be the set of all execution behaviors of the system. Its superset *Universe* additionally contains the spurious behaviors that might be described by imprecise analyses. Spurious behaviors can, for example, be sequences of concrete system states that cannot be observed during any execution of the concrete system.

$$\mathit{Universe} \supseteq \mathit{Traces}$$

The program executed on processor core C_i can be assigned an execution time per execution behavior. This time is given by the function et_{C_i}.

$$\mathit{et}_{C_i} : \mathit{Universe} \to \mathbb{N} \cup \{\infty\}$$

The WCET of the program executed on core C_i is its maximal execution time over all execution behaviors of the considered system.

$$\mathit{WCET}_{C_i} = \max_{t \in \mathit{Traces}} \mathit{et}_{C_i}(t) \tag{1}$$

B. Approximation by Abstract Traces

Modern processors usually exhibit too many execution behaviors to allow for an exhaustive consideration of all of them. The set *Traces* is simply too large. Therefore, it is mandatory to introduce some kind of approximation. The goal is to not have to argue separately about each concrete execution behavior.

In our view, an abstract model of the considered system is given by the tuple $(\widehat{\mathit{Traces}}, \gamma_{\mathit{trace}})$. $\widehat{\mathit{Traces}}$ is the set of abstract traces of the model. Depending on the chosen way of approximation, an abstract trace might for example be a sequence of abstract states in an analysis based on abstract interpretation [23] or the combination of a sequence of superblocks [5] and a corresponding sequence of blocking cycle counts. The function γ_{trace} maps those abstract traces to subsets of the universe of execution behaviors. Note that $\mathcal{P}(\mathit{Universe})$ denotes the power set of this universe of execution behaviors.

$$\gamma_{\mathit{trace}} : \widehat{\mathit{Traces}} \to \mathcal{P}(\mathit{Universe})$$

We say that an abstract model $(\widehat{\mathit{Traces}}, \gamma_{\mathit{trace}})$ is an overapproximation of *Traces* iff:

$$\bigcup_{\hat{t} \in \widehat{\mathit{Traces}}} \gamma_{\mathit{trace}}(\hat{t}) \supseteq \mathit{Traces} \tag{2}$$

We assume that for each core C_i there is an upper bound on its execution times per abstract trace. This bound shall be given by ${}^{\mathit{UB}}\mathit{et}_{C_i}$.

$$\begin{aligned} {}^{\mathit{UB}}\mathit{et}_{C_i} &: \widehat{\mathit{Traces}} \to \mathbb{N} \cup \{\infty\} \\ \forall \hat{t} \in \widehat{\mathit{Traces}} &: {}^{\mathit{UB}}\mathit{et}_{C_i}(\hat{t}) \geq \max_{t \in \gamma_{\mathit{trace}}(\hat{t})} \mathit{et}_{C_i}(t) \end{aligned} \tag{3}$$

From (2) and (3) it follows that the abstract model provides an upper bound to the WCET as defined in (1) by:

$$\max_{\hat{t} \in \widehat{\mathit{Traces}}} {}^{\mathit{UB}}\mathit{et}_{C_i}(\hat{t}) \geq \mathit{WCET}_{C_i} \tag{4}$$

From now on we only consider abstract models that are overapproximations of *Traces*.

C. Infeasible Abstract Traces

The method used to obtain the set of abstract traces (e.g. a static analysis exploring abstract states) might introduce imprecision. Therefore, there may be abstract traces that do not describe any execution behavior of the considered system. We call them *infeasible* abstract traces.

$$\widehat{\mathit{Infeas}} = \{\hat{t} \in \widehat{\mathit{Traces}} \mid \gamma_{\mathit{trace}}(\hat{t}) \cap \mathit{Traces} = \emptyset\} \tag{5}$$

Correspondingly, we refer to $\widehat{\mathit{Traces}} \setminus \widehat{\mathit{Infeas}}$ as the set of *feasible* abstract traces. In fact, it follows from (5) that the set of feasible abstract traces is an overapproximation of *Traces*.

$$\bigcup_{\hat{t} \in \widehat{\mathit{Traces}} \setminus \widehat{\mathit{Infeas}}} \gamma_{\mathit{trace}}(\hat{t}) \supseteq \mathit{Traces} \tag{6}$$

Based on an abstract model $(\widehat{\mathit{Traces}}, \gamma_{\mathit{trace}})$, which is an overapproximation of *Traces*, we define a set $\mathit{Deriv}_{(\widehat{\mathit{Traces}}, \gamma_{\mathit{trace}})}$ of further abstract models as follows:

$$\begin{aligned} &\mathit{Deriv}_{(\widehat{\mathit{Traces}}, \gamma_{\mathit{trace}})} = \\ &\{(\widehat{\mathit{Traces}}', \gamma_{\mathit{trace}}) \mid \widehat{\mathit{Traces}} \supseteq \widehat{\mathit{Traces}}' \supseteq \widehat{\mathit{Traces}} \setminus \widehat{\mathit{Infeas}}\} \end{aligned} \tag{7}$$

Intuitively, each element of $\mathit{Deriv}_{(\widehat{\mathit{Traces}}, \gamma_{\mathit{trace}})}$ is an overapproximation of *Traces*. So we can calculate an upper bound to the WCET based on any member of $\mathit{Deriv}_{(\widehat{\mathit{Traces}}, \gamma_{\mathit{trace}})}$:

$$\begin{aligned} &\forall (\widehat{\mathit{Traces}}', \gamma_{\mathit{trace}}) \in \mathit{Deriv}_{(\widehat{\mathit{Traces}}, \gamma_{\mathit{trace}})} : \\ &\qquad \max_{\hat{t} \in \widehat{\mathit{Traces}}'} {}^{\mathit{UB}}\mathit{et}_{C_i}(\hat{t}) \geq \mathit{WCET}_{C_i} \end{aligned} \tag{8}$$

As a consequence, we can ignore an arbitrarily chosen set of infeasible abstract traces in an abstract model. A WCET bound based on the remaining abstract traces is still guaranteed to be sound.

The calculation of WCET bounds is based on upper bounds on the execution times per abstract trace (3). If an abstract model makes conservative assumptions about the behavior at the shared resources, some infeasible abstract traces might assume an amount of shared-resource interference that exceeds the maximum possible amount for the concrete system. As upper bounds on the execution times of such infeasible abstract traces are likely to be very pessimistic, ignoring those abstract traces—as in (8)—might improve the tightness of the resulting WCET bound.

However, it depends heavily on the particular abstract model $(\widehat{\mathit{Traces}}, \gamma_{\mathit{trace}})$ and the upper bounds on the execution times per abstract trace whether the WCET bound can be tightened by leaving out some infeasible abstract traces.

We introduced the abstract model to not have to materialize the set *Traces*. The definition of infeasible abstract traces, however, is also based on *Traces*. Therefore, we cannot directly use this definition to detect infeasible abstract traces. The following subsection describes how we can use properties of the system under consideration to detect some infeasible abstract traces.

D. System Properties

We assume properties to be boolean predicates on execution behaviors. System properties are properties that hold for each execution behavior of a concrete system. The existence of a bound on the shared-resource interference may for example be a system property. Let *Prop* be a set of properties of the system under consideration:

$$\begin{aligned} &\mathit{Prop} = \{P_1, \ldots, P_p\} \\ &\forall t \in \mathit{Traces} : \forall P_k \in \mathit{Prop} : P_k(t) \end{aligned} \tag{9}$$

We want to use these system properties to detect some infeasible abstract traces. But so far, they only argue about execution behaviors of the concrete system. Therefore, we need to *lift* them to abstract traces. This means, we need to find $\widehat{P_k}$ such that the following criterion holds.

<u>*Soundness Criterion* (C1)*:*</u>

$$\begin{aligned} &\forall \hat{t} \in \widehat{\mathit{Traces}} : \\ &\quad [\, \exists t \in \gamma_{trace}(\hat{t}) : P_k(t) \,] \Rightarrow \widehat{P_k}(\hat{t}) \end{aligned} \tag{C1}$$

The intuition behind soundness criterion (C1) gets more clear if we have a look at what it means if $\widehat{P_k}$ does not hold for an abstract trace $\hat{t} \in \widehat{\mathit{Traces}}$:

$$\begin{aligned} &\neg\widehat{P_k}(\hat{t}) \\ &\underset{(C1)}{\Rightarrow} \forall t \in \gamma_{trace}(\hat{t}) : \neg P_k(t) \\ &\underset{(9)}{\Rightarrow} \gamma_{trace}(\hat{t}) \cap \mathit{Traces} = \emptyset \\ &\underset{(5)}{\Leftrightarrow} \hat{t} \in \widehat{\mathit{Infeas}} \end{aligned} \tag{10}$$

So if a lifted system property does not hold for an abstract trace, this means that the abstract trace is infeasible. From now on, the lifted version of any system property shall be identified by the name of the system property with an additional hat on top.

E. Property Lifting Example

The following example will illustrate how we can find a $\widehat{P_k}(\hat{t})$ satisfying (C1) without using $\gamma_{trace}(\hat{t})$ directly, which is mandatory for an efficient use of an abstract model.

<u>*Example:*</u> Assume that we have an upper bound on the number of bus accesses performed by a particular processor core C_i per abstract trace.

$$\begin{aligned} &\forall \hat{t} \in \widehat{\mathit{Traces}} : \\ &\quad \forall t \in \gamma_{trace}(\hat{t}) : \\ &\qquad {}^{\mathit{UB}}\#\mathit{accesses}_{C_i}(\hat{t}) \geq \#\mathit{accesses}_{C_i}(t) \end{aligned} \tag{a}$$

We only use γ_{trace} to argue about the soundness of the bounds. But we assume that each bound is given by a preceding analysis in the same way as the corresponding abstract trace is.

In addition, we assume to have a lower bound on the number of cycles that core C_i is blocked at a shared bus per abstract trace.

$$\begin{aligned} &\forall \hat{t} \in \widehat{\mathit{Traces}} : \\ &\quad \forall t \in \gamma_{trace}(\hat{t}) : \\ &\qquad {}^{\mathit{LB}}\#\mathit{blockedCycles}_{C_i}(\hat{t}) \leq \#\mathit{blockedCycles}_{C_i}(t) \end{aligned} \tag{b}$$

Now assume that the concrete system we consider uses a Round-Robin policy to arbitrate its shared bus. Therefore, all its execution behaviors fulfill the property P_{rr}:

$$\begin{aligned} P_{rr}(t) \Leftrightarrow [\, &\#\mathit{blockedCycles}_{C_i}(t) \\ &\leq \#\mathit{accesses}_{C_i}(t) \cdot (n-1) \\ &\quad \cdot \mathit{maxCyclesPerAccess}\,] \end{aligned} \tag{c}$$

The intuition behind this system property (implicitly assumed in [24]) is that with Round-Robin arbitration, each concurrent core (there are $n-1$ of them) can at most perform one access to the bus before an access of core C_i is granted. Together with an upper bound on the number of cycles that a granted bus access can at most take to complete on the concrete system, we arrive at an upper bound on the number of cycles that any access of core C_i can be blocked at the bus. Knowledge about how many accesses to the bus are performed by core C_i allows us to bound the overall amount of bus blocking experienced by core C_i in a particular execution behavior.

We can safely lift P_{rr} to abstract traces in a way that satisfies soundness criterion (C1) by applying (a) and (b):

$$\begin{aligned} \widehat{P_{rr}}(\hat{t}) \Leftrightarrow [\, &{}^{\mathit{LB}}\#\mathit{blockedCycles}_{C_i}(\hat{t}) \\ &\leq {}^{\mathit{UB}}\#\mathit{accesses}_{C_i}(\hat{t}) \cdot (n-1) \\ &\quad \cdot \mathit{maxCyclesPerAccess}\,] \end{aligned} \tag{d}$$

According to (10) any abstract trace $\hat{t}$ with $\neg\widehat{P_{rr}}(\hat{t})$ can safely be considered as infeasible. *[Example end]*

F. Removing Infeasible Abstract Traces

We define a compound property $\widehat{P}$ for abstract traces to be the conjunction over the lifted versions of the considered system properties.

$$\begin{aligned} &\forall \hat{t} \in \widehat{\mathit{Traces}} : \\ &\quad \widehat{P}(\hat{t}) \Leftrightarrow \forall P_k \in \mathit{Prop} : \widehat{P_k}(\hat{t}) \end{aligned} \tag{11}$$

If $\widehat{P}$ does not hold for an abstract trace $\hat{t}$ then this means that $\hat{t}$ is infeasible:

$$\begin{aligned} &\neg\widehat{P}(\hat{t}) \\ &\underset{(11)}{\Leftrightarrow} \exists P_k \in \mathit{Prop} : \neg\widehat{P_k}(\hat{t}) \\ &\underset{(10)}{\Rightarrow} \hat{t} \in \widehat{\mathit{Infeas}} \end{aligned} \tag{12}$$

We can use $\widehat{P}$ to define an alternative set $\widehat{\mathit{LessTraces}}$ of abstract traces based on $\widehat{\mathit{Traces}}$:

$$\widehat{\mathit{LessTraces}} = \{\hat{t} \mid \hat{t} \in \widehat{\mathit{Traces}} \wedge \widehat{P}(\hat{t})\} \tag{13}$$

$\widehat{\mathit{LessTraces}}$ is the subset of abstract traces in $\widehat{\mathit{Traces}}$ that cannot be classified as infeasible by any of the $\widehat{P_k}$.

$$\widehat{\mathit{LessTraces}} \supseteq \widehat{\mathit{Traces}} \setminus \widehat{\mathit{Infeas}} \tag{14}$$

Consequently, we can derive a sound WCET bound from the abstract model $(\widehat{\mathit{LessTraces}}, \gamma_{trace})$:

$$\max_{\hat{t} \in \widehat{\mathit{LessTraces}}} {}^{\mathit{UB}}\mathit{et}_{C_i}(\hat{t}) \geq \mathit{WCET}_{C_i} \tag{15}$$

$(\widehat{\mathit{LessTraces}}, \gamma_{trace})$ can improve the precision, as the set $\widehat{\mathit{LessTraces}}$ potentially prunes some of the infeasible abstract traces still included in $\widehat{\mathit{Traces}}$. In that context, $(\widehat{\mathit{Traces}}, \gamma_{trace})$ is referred to as *baseline abstract model* as it is the starting point for further improvements of precision.

This concludes the description of the concept of property lifting. Intuitively, the main idea is to start with a sound approximation as baseline. Lifted versions of system properties are used to detect some infeasible abstract traces of the baseline approximation. Removing them may result in more precise WCET bounds.

V. Iterative Overapproximation

Property lifting—as described in Section IV—requires a baseline abstract model arguing about all processor cores in detail in order to profit from system properties that interrelate the behaviors of all processor cores. Section V-A uses an exemplary system property to illustrate this requirement.

In Section V-B, we derive a compound abstract model from a set of abstract models—each focusing on one processor core. The compound abstract model argues about all cores in detail. Hence, system properties interrelating the behaviors of all cores can effectively be lifted to it.

However, the high number of abstract traces in the compound abstract model will likely become unmanageable. Thus, we project the analysis results from the compound abstract model back to the different component abstract models (Section V-C). Finally, we present an iterative approach to overapproximate these projections without having to materialize the compound abstract model (Section V-D).

A. Relating the Behavior of one Processor Core to that of Other Cores

Consider system properties that relate the behavior of one processor core to that of other cores. Such properties are typical for systems that do not provide performance isolation between their cores [24], [6].

Example: We introduce a property P_{wc} that holds for certain systems that enforce a *work conserving* bus arbitration policy.

$$\begin{aligned} P_{wc}(t) \Leftrightarrow [\, & \#\mathit{blockedCycles}_{C_i}(t) \\ & \leq \sum_{C_j \in (\mathit{Cores} \setminus \{C_i\})} \#\mathit{accessCycles}_{C_j}(t)\,] \end{aligned} \tag{e}$$

Essentially, it states that the number of cycles processor core C_i is blocked at the shared bus cannot exceed the number of cycles in which concurrent cores (here C_j) are granted access to the shared bus.

Assume that we have an upper bound on the number of bus access cycles performed by a particular processor core C_j per abstract trace.

$$\begin{aligned} & \forall C_j \in \mathit{Cores} : \\ & \quad \forall \hat{t} \in \widehat{\mathit{Traces}} : \\ & \quad\quad \forall t \in \gamma_{\mathit{trace}}(\hat{t}) : \\ & \quad\quad\quad {}^{\mathit{UB}}\#\mathit{accessCycles}_{C_j}(\hat{t}) \geq \#\mathit{accessCycles}_{C_j}(t) \end{aligned} \tag{f}$$

Using these upper bounds, we can lift the property P_{wc} to abstract traces.

$$\begin{aligned} \widehat{P_{wc}}(\hat{t}) \Leftrightarrow [\, & {}^{\mathit{LB}}\#\mathit{blockedCycles}_{C_i}(\hat{t}) \\ & \leq \sum_{C_j \in (\mathit{Cores} \setminus \{C_i\})} {}^{\mathit{UB}}\#\mathit{accessCycles}_{C_j}(\hat{t})\,] \end{aligned} \tag{g}$$

If an abstract model only focuses on one processor core, it has to assume arbitrary behaviors for the other cores. For now, assume that the abstract model $(\widehat{\mathit{Traces}}, \gamma_{\mathit{trace}})$ is only focused on core C_i. Thus, it cannot exclude arbitrarily high numbers of bus access cycles for all other cores.

$$\begin{aligned} & \forall C_j \in (\mathit{Cores} \setminus \{C_i\}) : \\ & \quad \forall \hat{t} \in \widehat{\mathit{Traces}} : \\ & \quad\quad {}^{\mathit{UB}}\#\mathit{accessCycles}_{C_j}(\hat{t}) = \infty \end{aligned} \tag{h}$$

As a consequence, the lifted property $\widehat{P_{wc}}$ holds for all abstract traces of the abstract model. Hence, it does not detect any infeasible abstract traces. *[Example end]*

This shows that property lifting only profits from properties interrelating the behaviors of several processor cores if it is applied to a baseline abstract model that argues about all those cores at the same time.

B. A Compound Abstract Model

Let *Models* be a set of identifiers of abstract models.

$$\mathit{Models} = \{M_1, \ldots, M_m\}$$

For each $M_a \in \mathit{Models}$ there shall be a corresponding abstract model $(\widehat{\mathit{Traces}^{M_a}}, \gamma^{M_a}_{\mathit{trace}})$ that is an overapproximation of *Traces*.

$$\begin{aligned} & \forall M_a \in \mathit{Models} : \\ & \quad \bigcup_{\widehat{t^{M_a}} \in \widehat{\mathit{Traces}^{M_a}}} \gamma^{M_a}_{\mathit{trace}}(\widehat{t^{M_a}}) \supseteq \mathit{Traces} \end{aligned} \tag{16}$$

$$\begin{aligned} & \forall M_a \in \mathit{Models} : \forall C_i \in \mathit{Cores} : \\ & \quad \forall \widehat{t^{M_a}} \in \widehat{\mathit{Traces}^{M_a}} : \\ & \quad\quad {}^{\mathit{UB}}\mathit{et}_{C_i}(\widehat{t^{M_a}}) \geq \max_{t \in \gamma^{M_a}_{\mathit{trace}}(\widehat{t^{M_a}})} \mathit{et}_{C_i}(t) \end{aligned} \tag{17}$$

The different abstract models may describe different aspects of the overall system behavior in detail. In the context of WCET analysis for multi-core processors, *Models* could be identical to *Cores* and each abstract model could focus on one particular core. The formalism, however, is not restricted to such an assumption.

Based on the abstract trace sets of previous abstract models, we can define a set $\widehat{\mathit{Traces}}$ of compound abstract traces.

$$\widehat{\mathit{Traces}} = \widehat{\mathit{Traces}^{M_1}} \times \cdots \times \widehat{\mathit{Traces}^{M_m}} \tag{18}$$

We use projection functions $\pi^{M_a}_{\mathit{trace}}$ to access the components of compound abstract traces.

$$\begin{aligned} & \forall (\widehat{t^{M_1}}, \ldots, \widehat{t^{M_m}}) \in \widehat{\mathit{Traces}^{M_1}} \times \cdots \times \widehat{\mathit{Traces}^{M_m}} : \\ & \quad \forall M_a \in \mathit{Models} : \\ & \quad\quad \pi^{M_a}_{\mathit{trace}}((\widehat{t^{M_1}}, \ldots, \widehat{t^{M_m}})) = \widehat{t^{M_a}} \end{aligned} \tag{19}$$

The mapping of compound abstract traces to subsets of the universe of execution behaviors can be defined as the intersection over the mappings of its components. Intuitively, a compound abstract trace only describes the execution behaviors that all of its components describe.

$$\gamma_{\mathit{trace}}(\hat{t}) = \bigcap_{M_a \in \mathit{Models}} \gamma^{M_a}_{\mathit{trace}}(\pi^{M_a}_{\mathit{trace}}(\hat{t})) \tag{20}$$

It follows from (16), (18) and (20) that the resulting compound abstract model $(\widehat{\mathit{Traces}}, \gamma_{\mathit{trace}})$ is also an overapproximation of *Traces* and thereby fulfills (2) and all its implications. Hence, we can apply property lifting to it as demonstrated in Section IV.

Our examples for lifted properties were so far based on upper and lower bounds per abstract trace. As soon as abstract traces are compositions, we may derive those bounds based on corresponding bounds for their components. According to (20), a particular bound for an abstract trace of the compound model can be obtained by taking the most precise bound value over its components.

$${}^{\mathit{LB}}\mathit{something}(\hat{t}) = \max_{M_a \in \mathit{Models}} {}^{\mathit{LB}}\mathit{something}(\pi^{M_a}_{\mathit{trace}}(\hat{t})) \tag{21}$$

$${}^{\mathit{UB}}\mathit{something}(\hat{t}) = \min_{M_a \in \mathit{Models}} {}^{\mathit{UB}}\mathit{something}(\pi^{M_a}_{\mathit{trace}}(\hat{t})) \tag{22}$$

Example: Reconsider the lifted example property $\widehat{P_{wc}}$ as defined in (g). Let us resume that example after formula (g). This time, we assume a compound abstract model as baseline. It shall be composed of one abstract model per processor core.

$$\mathit{Models} = \mathit{Cores} \tag{i}$$

Further assume that each abstract model can only provide detailed information about the processor core it is specialized on. In particular, this means:

$$\begin{aligned}
&\forall C_i \in \mathit{Cores} : \\
&\quad \forall \widehat{t^{C_i}} \in \widehat{\mathit{Traces}^{C_i}} : \\
&\quad\quad \forall C_j \in (\mathit{Cores} \setminus \{C_i\}) : \\
&\quad\quad\quad {}^{\mathit{LB}}\#\mathit{blockedCycles}_{C_j}(\widehat{t^{C_i}}) = 0 \wedge \\
&\quad\quad\quad {}^{\mathit{UB}}\#\mathit{accessCycles}_{C_j}(\widehat{t^{C_i}}) = \infty
\end{aligned} \tag{j}$$

In combination with (21) and (22) this implies the following equalities:

$$\begin{aligned}
&\forall C_i \in \mathit{Cores} : \\
&\quad {}^{\mathit{LB}}\#\mathit{blockedCycles}_{C_i}(\hat{t}) = {}^{\mathit{LB}}\#\mathit{blockedCycles}_{C_i}(\pi^{C_i}_{\mathit{trace}}(\hat{t})) \wedge \\
&\quad {}^{\mathit{UB}}\#\mathit{accessCycles}_{C_i}(\hat{t}) = {}^{\mathit{UB}}\#\mathit{accessCycles}_{C_i}(\pi^{C_i}_{\mathit{trace}}(\hat{t}))
\end{aligned} \tag{k}$$

This allows us to rewrite the lifted property $\widehat{P_{wc}}$ as follows:

$$\begin{aligned}
&\forall \hat{t} \in \widehat{\mathit{Traces}} : \\
&\quad \widehat{P_{wc}}(\hat{t}) \\
&\quad \underset{(g)}{\Leftrightarrow} [\, {}^{\mathit{LB}}\#\mathit{blockedCycles}_{C_i}(\hat{t}) \\
&\quad\quad \leq \sum_{C_j \in (\mathit{Cores} \setminus \{C_i\})} {}^{\mathit{UB}}\#\mathit{accessCycles}_{C_j}(\hat{t}) \,] \\
&\quad \underset{(k)}{\Leftrightarrow} [\, {}^{\mathit{LB}}\#\mathit{blockedCycles}_{C_i}(\pi^{C_i}_{\mathit{trace}}(\hat{t})) \\
&\quad\quad \leq \sum_{C_j \in (\mathit{Cores} \setminus \{C_i\})} {}^{\mathit{UB}}\#\mathit{accessCycles}_{C_j}(\pi^{C_j}_{\mathit{trace}}(\hat{t})) \,]
\end{aligned} \tag{m}$$

This time, the lifted property $\widehat{P_{wc}}$ is not guaranteed to hold for all abstract traces. Hence, it can effectively detect infeasible abstract traces. *[Example end]*

However, the cross product in the definition of $\widehat{\mathit{Traces}}$ already gives a hint that $\widehat{\mathit{Traces}}$ might become quite large. Thus, the compound consideration of several abstract models is likely too complex in most cases.

C. Projections of the Compound Results

Taking a closer look at the set $\widehat{\mathit{LessTraces}}$ derived from the compound abstract model, it turns out that we are not really interested in the set of all combinations of abstract traces from the different abstract models. It would be sufficient to know for each $M_a \in \mathit{Models}$ which members of $\widehat{\mathit{Traces}^{M_a}}$ are contained in a compound abstract trace of $\widehat{\mathit{LessTraces}}$. Those subsets can be obtained by projecting the members of $\widehat{\mathit{LessTraces}}$ to their different components. We define the projections in a general way on arbitrary subsets $\widehat{\mathit{SomeTraces}}$ of $\widehat{\mathit{Traces}}$.

$$\begin{aligned}
&\forall M_a \in \mathit{Models} : \\
&\quad \pi^{M_a}(\widehat{\mathit{SomeTraces}}) = \{\pi^{M_a}_{\mathit{trace}}(\hat{t}) \mid \hat{t} \in \widehat{\mathit{SomeTraces}}\}
\end{aligned} \tag{23}$$

Obviously, each projection $\pi^{M_a}(\widehat{\mathit{SomeTraces}})$ is a subset of the set of abstract traces of the corresponding abstract model.

$$\begin{aligned}
&\forall \widehat{\mathit{SomeTraces}} \subseteq \widehat{\mathit{Traces}} : \\
&\quad \forall M_a \in \mathit{Models} : \\
&\quad\quad \pi^{M_a}(\widehat{\mathit{SomeTraces}}) \subseteq \widehat{\mathit{Traces}^{M_a}}
\end{aligned} \tag{24}$$

Please note that $\widehat{\mathit{SomeTraces}}$ is a subset of the cross product over its projections.

$$\begin{aligned}
&\forall \widehat{\mathit{SomeTraces}} \subseteq \widehat{\mathit{Traces}} : \\
&\quad \widehat{\mathit{SomeTraces}} \\
&\quad\quad \subseteq \pi^{M_1}(\widehat{\mathit{SomeTraces}}) \times \cdots \times \pi^{M_m}(\widehat{\mathit{SomeTraces}})
\end{aligned} \tag{25}$$

Furthermore, it is rather obvious that the projection functions π^{M_a} are monotone.

$$\begin{aligned}
&\forall \widehat{\mathit{SomeTraces}}, \widehat{\mathit{OtherTraces}} \subseteq \widehat{\mathit{Traces}} : \\
&\quad \forall M_a \in \mathit{Models} : \\
&\quad\quad [\, \widehat{\mathit{SomeTraces}} \subseteq \widehat{\mathit{OtherTraces}} \,] \\
&\quad\quad\quad \Rightarrow [\, \pi^{M_a}(\widehat{\mathit{SomeTraces}}) \subseteq \pi^{M_a}(\widehat{\mathit{OtherTraces}}) \,]
\end{aligned} \tag{26}$$

Each projection $\pi^{M_a}(\widehat{\mathit{LessTraces}})$ is a superset of the feasible abstract traces of the corresponding $\widehat{\mathit{Traces}^{M_a}}$. Consider (27) for a formal proof of this statement. According to (7), (24) and (27), each abstract model $(\pi^{M_a}(\widehat{\mathit{LessTraces}}), \gamma^{M_a}_{\mathit{trace}})$ is a member of $\mathit{Deriv}_{(\widehat{\mathit{Traces}^{M_a}}, \gamma^{M_a}_{\mathit{trace}})}$.

$$\begin{aligned}
&\forall M_a \in \mathit{Models} : \\
&\quad (\pi^{M_a}(\widehat{\mathit{LessTraces}}), \gamma^{M_a}_{\mathit{trace}}) \in \mathit{Deriv}_{(\widehat{\mathit{Traces}^{M_a}}, \gamma^{M_a}_{\mathit{trace}})}
\end{aligned} \tag{28}$$

Thus, each abstract model $(\pi^{M_a}(\widehat{\mathit{LessTraces}}), \gamma^{M_a}_{\mathit{trace}})$ can be used to calculate a WCET bound based on it.

$$\begin{aligned}
&\forall M_a \in \mathit{Models} : \\
&\quad \max_{\widehat{t^{M_a}} \in \pi^{M_a}(\widehat{\mathit{LessTraces}})} {}^{\mathit{UB}}\mathit{et}_{C_i}(\widehat{t^{M_a}}) \geq \mathit{WCET}_{C_i}
\end{aligned} \tag{29}$$

But how precise is a WCET bound based on the projection of $\widehat{\mathit{LessTraces}}$ compared to one that is directly based on $\widehat{\mathit{LessTraces}}$? In general, we might lose precision by restricting ourselves to the projections $\pi^{M_a}(\widehat{\mathit{LessTraces}})$.

$$\begin{aligned}
&\mathit{WCET}_{C_i} \\
&\underset{(15)}{\leq} \max_{\hat{t} \in \widehat{\mathit{LessTraces}}} {}^{\mathit{UB}}\mathit{et}_{C_i}(\hat{t}) \\
&\underset{(22)}{=} \max_{\hat{t} \in \widehat{\mathit{LessTraces}}} \min_{M_a \in \mathit{Models}} {}^{\mathit{UB}}\mathit{et}_{C_i}(\pi^{M_a}_{\mathit{trace}}(\hat{t})) \\
&\underset{(31)}{\leq} \min_{M_a \in \mathit{Models}} \max_{\hat{t} \in \widehat{\mathit{LessTraces}}} {}^{\mathit{UB}}\mathit{et}_{C_i}(\pi^{M_a}_{\mathit{trace}}(\hat{t})) \\
&\underset{(23)}{=} \min_{M_a \in \mathit{Models}} \max_{\widehat{t^{M_a}} \in \pi^{M_a}(\widehat{\mathit{LessTraces}})} {}^{\mathit{UB}}\mathit{et}_{C_i}(\widehat{t^{M_a}})
\end{aligned} \tag{30}$$

We can prove the second $\leq$ relation used in (30) by assuming its opposite and deriving a statement from it that contradicts to the definition of the minimum.

$$\begin{aligned}
&\max_{\hat{t} \in \widehat{\mathit{LessTraces}}} \min_{M_a \in \mathit{Models}} {}^{\mathit{UB}}\mathit{et}_{C_i}(\pi^{M_a}_{\mathit{trace}}(\hat{t})) \\
&> \min_{M_a \in \mathit{Models}} \max_{\hat{t} \in \widehat{\mathit{LessTraces}}} {}^{\mathit{UB}}\mathit{et}_{C_i}(\pi^{M_a}_{\mathit{trace}}(\hat{t})) \\
&\Rightarrow \exists \widehat{t^*} \in \widehat{\mathit{LessTraces}} : \exists M_* \in \mathit{Models} : \\
&\quad \min_{M_a \in \mathit{Models}} {}^{\mathit{UB}}\mathit{et}_{C_i}(\pi^{M_a}_{\mathit{trace}}(\widehat{t^*})) \\
&\quad > \max_{\hat{t} \in \widehat{\mathit{LessTraces}}} {}^{\mathit{UB}}\mathit{et}_{C}(\pi^{M_*}_{\mathit{trace}}(\hat{t})) \\
&\quad \underset{\substack{\text{def.} \\ \max}}{\geq} {}^{\mathit{UB}}\mathit{et}_{C}(\pi^{M_*}_{\mathit{trace}}(\widehat{t^*})) \quad \lightning
\end{aligned} \tag{31}$$

However, we additionally assume each abstract model is focused on one processor core.

$$\mathit{Models} = \mathit{Cores} \tag{32}$$

$$\boxed{\begin{aligned}
&\pi^{M_a}(\widehat{\mathit{LessTraces}})\\
\underset{\substack{(14)\\(26)}}{\supseteq}\;&\pi^{M_a}(\widehat{\mathit{Traces}} \setminus \widehat{\mathit{Infeas}})\\
\underset{(5)}{=}\;&\pi^{M_a}(\,\{\hat{t} \mid \hat{t} \in \widehat{\mathit{Traces}} \wedge \gamma_{\mathit{trace}}(\hat{t}) \cap \mathit{Traces} \neq \emptyset\}\,)\\
\underset{(23)}{=}\;&\{\pi^{M_a}_{\mathit{trace}}(\hat{t}) \mid \hat{t} \in \widehat{\mathit{Traces}} \wedge \gamma_{\mathit{trace}}(\hat{t}) \cap \mathit{Traces} \neq \emptyset\}\\
\underset{\substack{(18)\\(19)}}{=}\;&\{\widehat{t^{M_a}} \mid (\widehat{t^{M_1}}, \dots, \widehat{t^{M_a}}, \dots, \widehat{t^{M_m}}) \in \widehat{\mathit{Traces}^{M_1}} \times \dots \times \widehat{\mathit{Traces}^{M_m}} \wedge \gamma_{\mathit{trace}}(\widehat{t^{M_1}}, \dots, \widehat{t^{M_a}}, \dots, \widehat{t^{M_m}}) \cap \mathit{Traces} \neq \emptyset\}\\
\underset{\substack{(20)\\(19)}}{=}\;&\{\widehat{t^{M_a}} \mid (\widehat{t^{M_1}}, \dots, \widehat{t^{M_a}}, \dots, \widehat{t^{M_m}}) \in \widehat{\mathit{Traces}^{M_1}} \times \dots \times \widehat{\mathit{Traces}^{M_m}} \wedge \bigcap_{M_b \in \mathit{Models}} \gamma^{M_b}_{\mathit{trace}}(\widehat{t^{M_b}}) \cap \mathit{Traces} \neq \emptyset\}\\
\underset{(16)}{=}\;&\{\widehat{t^{M_a}} \mid \widehat{t^{M_a}} \in \widehat{\mathit{Traces}^{M_a}} \wedge \gamma^{M_a}_{\mathit{trace}}(\widehat{t^{M_a}}) \cap \mathit{Traces} \neq \emptyset\}\\
\underset{(5)}{=}\;&\widehat{\mathit{Traces}^{M_a}} \setminus \widehat{\mathit{Infeas}^{M_a}}
\end{aligned}} \tag{27}$$

This in particular means that each abstract model has to make maximally pessimistic assumptions about the execution times of the cores it is not focused on.

$$\begin{aligned}
&\forall C_i \in \mathit{Cores}:\\
&\quad \forall \widehat{t^{C_i}} \in \widehat{\mathit{Traces}^{C_i}}:\\
&\qquad \forall C_j \in (\mathit{Cores} \setminus \{C_i\}):\\
&\qquad\quad {}^{\mathit{UB}}\#\mathit{et}_{C_j}(\widehat{t^{C_i}}) = \infty
\end{aligned} \tag{33}$$

Under those additional assumptions, we are guaranteed to not lose any precision by restricting ourselves to WCET bounds based on the projections $\pi^{M_a}(\widehat{\mathit{LessTraces}})$.

$$\begin{aligned}
&\mathit{WCET}_{C_i}\\
\underset{(15)}{\leq}\;& \max_{\hat{t} \in \widehat{\mathit{LessTraces}}} {}^{\mathit{UB}}\mathit{et}_{C_i}(\hat{t})\\
\underset{(22)}{=}\;& \max_{\hat{t} \in \widehat{\mathit{LessTraces}}} \min_{M_a \in \mathit{Models}} {}^{\mathit{UB}}\mathit{et}_{C_i}(\pi^{M_a}_{\mathit{trace}}(\hat{t}))\\
\underset{(32)}{=}\;& \max_{\hat{t} \in \widehat{\mathit{LessTraces}}} \min_{C_j \in \mathit{Cores}} {}^{\mathit{UB}}\mathit{et}_{C_i}(\pi^{C_j}_{\mathit{trace}}(\hat{t}))\\
\underset{(33)}{=}\;& \max_{\hat{t} \in \widehat{\mathit{LessTraces}}} {}^{\mathit{UB}}\mathit{et}_{C_i}(\pi^{C_i}_{\mathit{trace}}(\hat{t}))\\
\underset{(23)}{=}\;& \max_{\widehat{t^{C_i}} \in \pi^{C_i}(\widehat{\mathit{LessTraces}})} {}^{\mathit{UB}}\mathit{et}_{C_i}(\widehat{t^{C_i}})
\end{aligned} \tag{34}$$

So we see that, in general, the projections $\pi^{M_a}(\widehat{\mathit{LessTraces}})$ can be used to derive WCET bounds based on them. We do not need to know all combinations of abstract traces contained in $\widehat{\mathit{LessTraces}}$. Under the additional assumptions (32) and (33), we do not lose any precision compared to WCET bounds derived from $\widehat{\mathit{LessTraces}}$ directly. However, in most cases we will not be able to precisely obtain the projections $\pi^{M_a}(\widehat{\mathit{LessTraces}})$ without first materializing the set $\widehat{\mathit{LessTraces}}$. As discussed before, it is expected to be computationally too expensive to derive the set $\widehat{\mathit{LessTraces}}$. Therefore, we are interested in overapproximations of these projections.

D. Overapproximating the Projections

This subsection describes an iterative approach that overapproximates the projections $\pi^{M_a}(\widehat{\mathit{LessTraces}})$. It starts with very conservative assumptions about all projections. Intuitively, the overapproximation of a particular projection can be improved by incorporating information from the overapproximations of the other projections.

Clearly, it is possible to obtain an overapproximation of a projection $\pi^{M_a}(\widehat{\mathit{LessTraces}})$ by considering the abstract model $(\widehat{\mathit{Traces}^{M_a}}, \gamma^{M_a}_{\mathit{trace}})$ in isolation and providing the set $\widehat{\mathit{LessTraces}^{M_a}}$. In this case, however, the lifted versions $\widehat{P_k^{M_a}}$ of properties P_k do not help us in detecting infeasible abstract traces if the P_k need to argue about aspects of the system that are not modeled by $(\widehat{\mathit{Traces}^{M_a}}, \gamma^{M_a}_{\mathit{trace}})$. Therefore, the overapproximation of a projection $\pi^{M_a}(\widehat{\mathit{LessTraces}})$ should be able to incorporate (likely cumulative) information from the overapproximations of the other projections.

We propose an approach that overapproximates each projection $\pi^{M_a}(\widehat{\mathit{LessTraces}})$ by a corresponding approximation variable $\widehat{\mathit{Approx}^{M_a}}$. We use $\overrightarrow{\mathit{Approx}}$ to refer to the vector of all approximation variables.

$$\overrightarrow{\mathit{Approx}} = (\widehat{\mathit{Approx}^{M_1}}, \dots, \widehat{\mathit{Approx}^{M_m}}) \tag{35}$$

In the beginning, each $\widehat{\mathit{Approx}^{M_a}}$ is initialized to the corresponding $\widehat{\mathit{Traces}^{M_a}}$.

$$\overrightarrow{\mathit{Approx}} \leftarrow (\widehat{\mathit{Traces}^{M_1}}, \dots, \widehat{\mathit{Traces}^{M_m}}) \tag{36}$$

Then, the approximation variables are updated according to the following recursive equation system.

$$\begin{aligned}
&\forall M_a \in \mathit{Models}:\\
&\quad \widehat{\mathit{Approx}^{M_a}}\\
&\quad = \{\widehat{t^{M_a}} \mid \widehat{t^{M_a}} \in \widehat{\mathit{Traces}^{M_a}} \wedge \widetilde{P^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{\mathit{Approx}})\}\\
&\quad =: F^{M_a}(\overrightarrow{\mathit{Approx}})
\end{aligned} \tag{37}$$

We refer to F^{M_a} as the update function of $\widehat{\mathit{Approx}^{M_a}}$. From (36) and (37) we can immediately follow that each $\widehat{\mathit{Approx}^{M_a}}$ is guaranteed to always be a subset of $\widehat{\mathit{Traces}^{M_a}}$.

$$\begin{aligned}
&\forall M_a \in \mathit{Models}:\\
&\quad \widehat{\mathit{Approx}^{M_a}} \subseteq \widehat{\mathit{Traces}^{M_a}}
\end{aligned} \tag{38}$$

Therefore, the value range of the vector of approximation variables can be restricted as follows.

$$\overrightarrow{\mathit{Approx}} \in \mathcal{P}(\widehat{\mathit{Traces}^{M_1}}) \times \dots \times \mathcal{P}(\widehat{\mathit{Traces}^{M_m}}) \tag{39}$$

The boolean predicate $\widetilde{P^{M_a}}$ used in F^{M_a} takes an abstract trace from $\widehat{\mathit{Traces}^{M_a}}$ and the current vector of approximation variables as parameters. It is defined as follows.

$$\begin{aligned}
&\forall M_a \in \mathit{Models} : \\
&\quad \forall \widehat{t^{M_a}} \in \widehat{\mathit{Traces}^{M_a}} : \\
&\quad\quad \widetilde{P^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{\mathit{Approx}}) \\
&\quad\quad\quad \Leftrightarrow \forall P_k \in \mathit{Prop} : \widetilde{P_k^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{\mathit{Approx}})
\end{aligned} \tag{40}$$

The $\widetilde{P_k^{M_a}}$ are properties that overapproximate the $\widehat{P_k}$ lifted to the compound abstract model. They shall fulfill the following criterion with respect to the $\widehat{P_k}$.

<u>*Soundness Criterion* (C2)*:*</u>

$$\begin{aligned}
&\forall \overrightarrow{\mathit{Approx}} \in \mathcal{P}(\widehat{\mathit{Traces}^{M_1}}) \times \cdots \times \mathcal{P}(\widehat{\mathit{Traces}^{M_m}}) : \\
&\quad \forall \widehat{t^{M_a}} \in \widehat{\mathit{Traces}^{M_a}} : \\
&\quad\quad [\, \exists (\widehat{t^{M_1}}, \ldots, \widehat{t^{M_{a-1}}}) \in \widehat{\mathit{Approx}^{M_1}} \times \cdots \times \widehat{\mathit{Approx}^{M_{a-1}}} : \\
&\quad\quad\quad \exists (\widehat{t^{M_{a+1}}}, \ldots, \widehat{t^{M_m}}) \in \widehat{\mathit{Approx}^{M_{a+1}}} \times \cdots \times \widehat{\mathit{Approx}^{M_m}} : \\
&\quad\quad\quad\quad \widehat{P_k}(\widehat{t^{M_1}}, \ldots, \widehat{t^{M_{a-1}}}, \widehat{t^{M_a}}, \widehat{t^{M_{a+1}}}, \ldots, \widehat{t^{M_m}}) \,] \\
&\quad\quad \Rightarrow \widetilde{P_k^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{\mathit{Approx}})
\end{aligned} \tag{C2}$$

Criterion (C2) allows us to show that each approximation variable is guaranteed to be an overapproximation of the corresponding projection after arbitrary sequences of updates of the approximation variables:

$$\begin{aligned}
&\forall M_a \in \mathit{Models} : \\
&\quad \pi^{M_a}(\widehat{\mathit{LessTraces}}) \subseteq \widehat{\mathit{Approx}^{M_a}}
\end{aligned} \tag{H1}$$

Proof: As a consequence of (24), the claim in (H1) trivially holds for the initial values of the approximation variables as specified in (36). For the general case, however, we assume the hypothesis (H1) to hold after a given sequence of approximation variable updates. In an inductive way, we can use this assumption to show that the hypothesis is preserved by an additional simultaneous update of an arbitrarily chosen set of the approximation variables. For the details of this induction step, please refer to (41) and (42). The inequation chain in (41) shows that all sets contained in it are in fact equal. This information is used in (42) to show that the $F^{M_a}(\overrightarrow{\mathit{Approx}})$ are guaranteed to be supersets of the projections $\pi^{M_a}(\widehat{\mathit{LessTraces}})$. According to the equation system given by (37), the approximation variables $\widehat{\mathit{Approx}^{M_a}}$ are updated to the values of the functions $F^{M_a}(\overrightarrow{\mathit{Approx}})$. This proves that the simultaneous update of an arbitrarily chosen set of approximation variables is guaranteed to preserve the hypothesis given by (H1). □

As a consequence of (42), we can optionally use the alternative definitions of the update functions F^{M_a} given in (43). Their use is equivalent to the use of the definitions in (37). Depending on the implementation details of an instance of our framework, it could be more straightforward to use one style of definition or the other.

$$\begin{aligned}
&\forall M_a \in \mathit{Models} : \\
&\quad F^{M_a}(\overrightarrow{\mathit{Approx}}) \\
&\quad := \{\widehat{t^{M_a}} \mid \widehat{t^{M_a}} \in \widehat{\mathit{Approx}^{M_a}} \wedge \widetilde{P^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{\mathit{Approx}})\}
\end{aligned} \tag{43}$$

It follows from hypothesis (H1) that we can bound the content of the sets $\widehat{\mathit{Approx}^{M_a}}$ from above and from below.

$$\begin{aligned}
&\widehat{\mathit{Traces}^{M_a}} \\
&\underset{(38)}{\supseteq} \widehat{\mathit{Approx}^{M_a}} \\
&\underset{(\mathrm{H1})}{\supseteq} \pi^{M_a}(\widehat{\mathit{LessTraces}}) \\
&\underset{(27)}{\supseteq} \widehat{\mathit{Traces}^{M_a}} \setminus \widehat{\mathit{Infeas}^{M_a}}
\end{aligned} \tag{44}$$

As a consequence of (7) and (44), we see that each abstract model $(\widehat{\mathit{Approx}^{M_a}}, \gamma_{\mathit{trace}}^{M_a})$ is a member of $\mathit{Deriv}_{(\widehat{\mathit{Traces}^{M_a}}, \gamma_{\mathit{trace}}^{M_a})}$.

$$\begin{aligned}
&\forall M_a \in \mathit{Models} : \\
&\quad (\widehat{\mathit{Approx}^{M_a}}, \gamma_{\mathit{trace}}^{M_a}) \in \mathit{Deriv}_{(\widehat{\mathit{Traces}^{M_a}}, \gamma_{\mathit{trace}}^{M_a})}
\end{aligned} \tag{45}$$

Thus, each abstract model $(\widehat{\mathit{Approx}^{M_a}}, \gamma_{\mathit{trace}}^{M_a})$ can be used to calculate a WCET bound based on it.

$$\begin{aligned}
&\forall M_a \in \mathit{Models} : \\
&\quad \max_{\widehat{t^{M_a}} \in \widehat{\mathit{Approx}^{M_a}}} {}^{\mathit{UB}}\mathit{et}_{C_i}(\widehat{t^{M_a}}) \geq \mathit{WCET}_{C_i}
\end{aligned} \tag{46}$$

In addition, the $\widetilde{P_k^{M_a}}$ shall fulfill the following criterion.

<u>*Monotonicity Criterion* (C3)*:*</u>

$$\begin{aligned}
&\forall \overrightarrow{\mathit{Approx}}, \overrightarrow{\mathit{Approx}'} \in \mathcal{P}(\widehat{\mathit{Traces}^{M_1}}) \times \cdots \times \mathcal{P}(\widehat{\mathit{Traces}^{M_m}}) : \\
&\quad \forall \widehat{t^{M_a}} \in \widehat{\mathit{Traces}^{M_a}} : \\
&\quad\quad [\, \forall M_b \in \mathit{Models} : \\
&\quad\quad\quad \widehat{\mathit{Approx}'^{M_b}} \subseteq \widehat{\mathit{Approx}^{M_b}} \,] \\
&\quad\quad \Rightarrow [\, \widetilde{P_k^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{\mathit{Approx}'}) \Rightarrow \widetilde{P_k^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{\mathit{Approx}}) \,]
\end{aligned} \tag{C3}$$

Let $\overrightarrow{\mathit{Approx}}$ be the vector of approximation variables after an arbitrary sequence of updates of the approximation variables. Criterion (C3) allows us to show that the following additional hypothesis holds:

$$\begin{aligned}
&\forall M_a \in \mathit{Models} : \\
&\quad F^{M_a}(\overrightarrow{\mathit{Approx}}) \subseteq \widehat{\mathit{Approx}^{M_a}}
\end{aligned} \tag{H2}$$

Proof: According to (36) and (37), the hypothesis (H2) trivially holds for a vector $\overrightarrow{\mathit{Approx}}$ just initialized. For the inductive step, assume that hypothesis (H2) holds for a given vector $\overrightarrow{\mathit{Approx}}$ of approximation variables. Let $\overrightarrow{\mathit{Approx}'}$ be the successor of $\overrightarrow{\mathit{Approx}}$ after the simultaneous update of an arbitrarily chosen set of approximation variables:

$$\overrightarrow{\mathit{Approx}'} = (\widehat{\mathit{Approx}'^{M_1}}, \ldots, \widehat{\mathit{Approx}'^{M_m}}) \tag{47}$$

$$\begin{aligned}
&\forall M_a \in \mathit{Models} : \\
&\quad \widehat{\mathit{Approx}'^{M_a}} \in \{\widehat{\mathit{Approx}^{M_a}}, F^{M_a}(\overrightarrow{\mathit{Approx}})\}
\end{aligned} \tag{48}$$

It follows from (H2) and (48) that each component of $\overrightarrow{\mathit{Approx}'}$ is a subset of its corresponding counterpart in $\overrightarrow{\mathit{Approx}}$.

$$\begin{aligned}
&\forall M_a \in \mathit{Models} : \\
&\quad \widehat{\mathit{Approx}'^{M_a}} \subseteq \widehat{\mathit{Approx}^{M_a}}
\end{aligned} \tag{49}$$

$$
\begin{aligned}
&\pi^{M_a}(\widehat{LessTraces})\\
&\underset{(23)}{=}\{\pi^{M_a}_{trace}(\hat{t}) \mid \hat{t} \in \widehat{LessTraces}\}\\
&\underset{(13)}{=}\{\pi^{M_a}_{trace}(\hat{t}) \mid \hat{t} \in \widehat{LessTraces} \wedge \widehat{P}(\hat{t})\}\\
&\underset{(25)}{\subseteq}\{\pi^{M_a}_{trace}(\hat{t}) \mid \hat{t} \in \pi^{M_1}(\widehat{LessTraces}) \times \cdots \times \pi^{M_m}(\widehat{LessTraces}) \wedge \widehat{P}(\hat{t})\}\\
&\underset{(H1)}{\subseteq}\{\pi^{M_a}_{trace}(\hat{t}) \mid \hat{t} \in \widehat{Approx}^{M_1} \times \cdots \times \widehat{Approx}^{M_m} \wedge \widehat{P}(\hat{t})\}\\
&\underset{(38)}{\subseteq}\{\pi^{M_a}_{trace}(\hat{t}) \mid \hat{t} \in \widehat{Traces}^{M_1} \times \cdots \times \widehat{Traces}^{M_m} \wedge \widehat{P}(\hat{t})\}\\
&\underset{(18)}{=}\{\pi^{M_a}_{trace}(\hat{t}) \mid \hat{t} \in \widehat{Traces} \wedge \widehat{P}(\hat{t})\}\\
&\underset{(13)}{=}\{\pi^{M_a}_{trace}(\hat{t}) \mid \hat{t} \in \widehat{LessTraces}\}
\end{aligned}
\tag{41}
$$

$$
\begin{aligned}
&\pi^{M_a}(\widehat{LessTraces})\\
&\underset{(41)}{=}\{\pi^{M_a}_{trace}(\hat{t}) \mid \hat{t} \in \widehat{Approx}^{M_1} \times \cdots \times \widehat{Approx}^{M_m} \wedge \widehat{P}(\hat{t})\}\\
&\underset{(19)}{=}\{\widehat{t^{M_a}} \mid (\widehat{t^{M_1}}, \ldots, \widehat{t^{M_a}}, \ldots, \widehat{t^{M_m}}) \in \widehat{Approx}^{M_1} \times \cdots \times \widehat{Approx}^{M_m} \wedge \widehat{P}(\widehat{t^{M_1}}, \ldots, \widehat{t^{M_a}}, \ldots, \widehat{t^{M_m}})\}\\
&\underset{(11)}{=}\{\widehat{t^{M_a}} \mid (\widehat{t^{M_1}}, \ldots, \widehat{t^{M_a}}, \ldots, \widehat{t^{M_m}}) \in \widehat{Approx}^{M_1} \times \cdots \times \widehat{Approx}^{M_m} \wedge \forall P_k \in Prop : \widehat{P_k}(\widehat{t^{M_1}}, \ldots, \widehat{t^{M_a}}, \ldots, \widehat{t^{M_m}})\}\\
&\underset{(C2)}{\subseteq}\{\widehat{t^{M_a}} \mid \widehat{t^{M_a}} \in \widehat{Approx}^{M_a} \wedge \forall P_k \in Prop : \widetilde{P_k^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{Approx})\}\\
&\underset{(40)}{=}\{\widehat{t^{M_a}} \mid \widehat{t^{M_a}} \in \widehat{Approx}^{M_a} \wedge \widetilde{P^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{Approx})\}\\
&\underset{(38)}{\subseteq}\{\widehat{t^{M_a}} \mid \widehat{t^{M_a}} \in \widehat{Traces}^{M_a} \wedge \widetilde{P^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{Approx})\}\\
&\underset{(37)}{=}F^{M_a}(\overrightarrow{Approx})
\end{aligned}
\tag{42}
$$

We assume that the update functions F^{M_a} can be applied to $\overrightarrow{Approx'}$ in the same way as to $\overrightarrow{Approx}$. Equation (50) shows that $F^{M_a}(\overrightarrow{Approx'})$ is a subset of $F^{M_a}(\overrightarrow{Approx})$.

$$
\begin{aligned}
&F^{M_a}(\overrightarrow{Approx'})\\
&\underset{(37)}{=}\{\widehat{t^{M_a}} \mid \widehat{t^{M_a}} \in \widehat{Traces}^{M_a} \wedge \widetilde{P^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{Approx'})\}\\
&\underset{(40)}{=}\{\widehat{t^{M_a}} \mid \widehat{t^{M_a}} \in \widehat{Traces}^{M_a}\\
&\qquad\qquad \wedge \forall P_k \in Prop : \widetilde{P_k^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{Approx'})\}\\
&\underset{\substack{(C3)\\(49)}}{\subseteq}\{\widehat{t^{M_a}} \mid \widehat{t^{M_a}} \in \widehat{Traces}^{M_a}\\
&\qquad\qquad \wedge \forall P_k \in Prop : \widetilde{P_k^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{Approx})\}\\
&\underset{(40)}{=}\{\widehat{t^{M_a}} \mid \widehat{t^{M_a}} \in \widehat{Traces}^{M_a} \wedge \widetilde{P^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{Approx})\}\\
&\underset{(37)}{=}F^{M_a}(\overrightarrow{Approx})
\end{aligned}
\tag{50}
$$

Based on those results, it is straightforward to show that the hypothesis also holds for $\overrightarrow{Approx'}$, which concludes the inductive proof of (H2):

$$
\begin{aligned}
&\widehat{Approx'}^{M_a}\\
&\underset{(48)}{=}\begin{cases}\widehat{Approx}^{M_a}\\ F^{M_a}(\overrightarrow{Approx})\end{cases}\\
&\underset{(H2)}{\supseteq}F^{M_a}(\overrightarrow{Approx})\\
&\underset{(50)}{\supseteq}F^{M_a}(\overrightarrow{Approx'})
\end{aligned}
\tag{51}
$$

□

The intuition behind (H2) is that the update of an approximation variable is guaranteed to never increase its set of abstract traces. As the calculation of the WCET bounds is based on the abstract trace sets, we can be sure that the update of some approximation variables can never result in worse WCET bounds.

Example: Coming back to the example of Sections V-A and V-B, we can further lift the property $\widehat{P_{wc}}$—as defined in equation (m)—in a way that satisfies criteria (C2) and (C3):

$$
\begin{aligned}
&\widetilde{P_{wc}^{C_i}}(\widehat{t^{C_i}}, (\widehat{Approx}^{C_1}, \ldots, \widehat{Approx}^{C_n}))\\
&\Leftrightarrow [\,{}^{LB}\#blockedCycles_{C_i}(\widehat{t^{C_i}}) \leq\\
&\qquad \sum_{C_j \in (Cores \setminus \{C_i\})} \max_{\widehat{t^{C_j}} \in \widehat{Approx}^{C_j}} {}^{UB}\#accessCycles_{C_j}(\widehat{t^{C_j}})\,]
\end{aligned}
\tag{n}
$$

Note that the right-hand side of the inequation in property $\widetilde{P_{wc}^{C_i}}$ does not depend on the abstract trace $\widehat{t^{C_i}}$. It contains cumulative information about the processor cores competing against C_i. Thus, this right-hand side is constant over all evaluations of $\widetilde{P_{wc}^{C_i}}$ during an update of $\widehat{\mathit{Approx}^{C_i}}$. Therefore, we can precompute the constant right-hand side based on the other approximation variables before updating $\widehat{\mathit{Approx}^{C_i}}$. The constant right-hand side can subsequently be used in an integer linear programming constraint. [17] *[Example end]*

Moreover, we can use hypothesis (H2) to show that all sets contained in the following cyclic inequation chain are equal.

$$\begin{aligned}
&F^{M_a}(\overrightarrow{\mathit{Approx}}) \\
&\underset{(37)}{=} \{\widehat{t^{M_a}} \mid \widehat{t^{M_a}} \in F^{M_a}(\overrightarrow{\mathit{Approx}}) \wedge \widetilde{P^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{\mathit{Approx}})\} \\
&\underset{(H2)}{\subseteq} \{\widehat{t^{M_a}} \mid \widehat{t^{M_a}} \in \widehat{\mathit{Approx}^{M_a}} \wedge \widetilde{P^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{\mathit{Approx}})\} \\
&\underset{(38)}{\subseteq} \{\widehat{t^{M_a}} \mid \widehat{t^{M_a}} \in \widehat{\mathit{Traces}^{M_a}} \wedge \widetilde{P^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{\mathit{Approx}})\} \\
&\underset{(37)}{=} F^{M_a}(\overrightarrow{\mathit{Approx}})
\end{aligned} \tag{52}$$

An interesting consequence of (52) is that, in particular, the following equation holds.

$$\begin{aligned}
&\{\widehat{t^{M_a}} \mid \widehat{t^{M_a}} \in \widehat{\mathit{Approx}^{M_a}} \wedge \widetilde{P^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{\mathit{Approx}})\} \\
\underset{(52)}{=} &\{\widehat{t^{M_a}} \mid \widehat{t^{M_a}} \in \widehat{\mathit{Traces}^{M_a}} \wedge \widetilde{P^{M_a}}(\widehat{t^{M_a}}, \overrightarrow{\mathit{Approx}})\}
\end{aligned} \tag{53}$$

Using (53) in the induction step (42) of the soundness proof instead of (38) allows us to replace the last $\subseteq$ by an equal. With this improvement in place, we see that (42) has only left a single $\subseteq$. This means, we can exactly point out where the update of an approximation variable may lose precision compared to the projections of the compound consideration of all models at once. The compound consideration of all models only keeps those $\widehat{t^{M_a}}$ that occur in a combination with abstract traces from the other models that fulfills all lifted properties $\widehat{P_k}$. However, the specially lifted properties $\widetilde{P_k^{M_a}}$ may lead to different results. In their presence, an abstract trace $\widehat{t^{M_a}}$ is not pruned as soon as for each $\widehat{P_k}$ there is a particular combination with abstract traces from the other models that fulfills $\widehat{P_k}$—this is a consequence of (C2). The intersection of those sets of combinations of abstract traces for the different $\widehat{P_k}$ could possibly be empty for a particular $\widehat{t^{M_a}}$. In that case, this $\widehat{t^{M_a}}$ is not contained in $\pi^{M_a}(\widehat{\mathit{LessTraces}})$, but in its overapproximation $\widehat{\mathit{Approx}^{M_a}}$.

Please note that the additional abstract traces introduced in this way by overapproximation are inherent to considering each abstract model on its own. They can occur even if we choose the $\widetilde{P_k^{M_a}}$ in a way that the ⇒ in criterion (C2) can be shown to be replaceable by a ⇔. This inherent amount of overapproximation only depends on the abstract models $(\widehat{\mathit{Traces}^{M_a}}, \gamma_{\mathit{trace}}^{M_a})$ and the way in which the $\widehat{P_k}$ are chosen. Of course, it might lead to further overapproximation if we choose the $\widetilde{P_k^{M_a}}$ in a way that we cannot prove the additional equivalence relation.

VI. Advantages of the Framework

This section highlights the benefits of using our framework.

Standard derivation procedure: The framework is a common starting point for the derivation of future WCET analyses for multi-core processors. It has been successfully used in the development of a novel analysis that avoids the restrictions of previous approaches (cf. Section II).

Soundness guarantee: We show in this paper that an instance of our framework is a sound WCET analysis. This soundness is a consequence of a sound baseline analysis and the property lifting according to the criteria presented above. A sound baseline analysis is easily obtained by adapting a single-core WCET analysis in a way that makes it maximally pessimistic with respect to the shared-resource interference [17]. Hence, our framework essentially reduces the soundness proofs of its instances to showing the soundness of the property lifting steps involved in their derivations.

Assumptions about the system always explicit: The declarative style of our framework makes it mandatory to explicitly list all properties that a derived analysis assumes about the system under analysis. This makes sure that a derived analysis does not rely on implicit assumptions (except those that its baseline analysis already relies on).

Clean separation between concrete system and approximation: Existing analyses often try to directly incorporate properties of the concrete system in their level of approximation. However, this is mostly based on intuition and, thus, very error-prone. The concept of property lifting, in contrast, provides a clean separation between system properties and their implications on the approximation.

Trade-off between efficiency and precision: The iterative over-approximation (Section V-D) forms a trade-off between the efficiency of analyzing the programs of one processor core in isolation (Section V-A) and the precision of performing a simultaneous consideration of the detailed behaviors of the programs executed on all processor cores (Section V-B).

Not limited to multi-core processors: The principles presented throughout this paper are not limited to the analysis of multi-core processors. Some of the techniques used in single-core WCET analysis can also be seen as instances of our framework. The micro-architectural analysis, for example, typically has no notion of loop bounds. Thus, it pessimistically assumes that each loop body in the program can be executed indefinitely. Loop bounds of the concrete program are subsequently lifted to the level of approximation that the path analysis operates on. The lifted loop bounds are typically implemented as additional constraints in an implicit path enumeration [25].

VII. Framework Instantiation Workflow

Figure 5 sketches the typical workflow of deriving WCET analyses as instances of our framework. A derivation that only relies on the concept of property lifting (cf. Section IV) comprises two logical steps. The derivation of an analysis that iteratively overapproximates the results of properties lifted to a compound abstract model (cf. Section V) requires an additional lifting step.

We successfully used our framework for the derivation of two novel WCET analyses for multi-core processors with a shared bus and Round-Robin bus arbitration: a co-runner-insensitive analysis and a co-runner-sensitive one [17]. This section describes—at a high level—how the derivation of each of these analyses follows the instantiation workflow sketched in Figure 5.

The derivation of our *co-runner-insensitive* analysis comprises two steps:

Step 1: The derivation starts from a baseline analysis focusing on one core and assuming that each access request to the shared bus can be blocked indefinitely by the bus arbiter. Furthermore, we consider a system property that bounds the maximum amount of blocked cycles per access to the shared bus under Round-Robin arbitration.

Step 2: The Round-Robin property is lifted to the baseline analysis. The lifted property is subsequently added to the implementation of

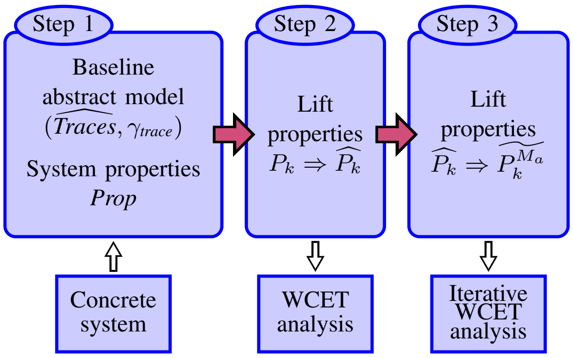

Figure 5: Framework Instantiation Workflow

the baseline analysis in order to prune infeasible behavior at its level of approximation.

The derivation of our *co-runner-sensitive* analysis, in contrast, comprises three steps:

Step 1: The derivation starts from a compound baseline abstract model that consists of one co-runner-insensitive analysis per processor core. Thus, the compound baseline analysis argues about all processor cores. We consider a property that bounds the blocked cycles of the core under analysis based on the access cycles of the concurrent cores assuming a work-conserving bus arbitration (like e.g. Round-Robin).

Step 2: The work-conserving property is lifted to the baseline abstract model. However, it would be unpractical to enumerate all combinations of abstract traces of the analyses for the different cores (since the compound abstract model is defined as cross product over its components).

Step 3: Hence, we further lift the already lifted property to the component analysis only focusing on the core under analysis. To this end, we assume per concurrent core the maximum amount of access cycles possible in any interval no longer than the current WCET bound of the core under analysis. The resulting analysis starts by calculating the co-runner-insensitive WCET bound for the core under analysis. Then, it calculates upper bounds on the concurrent access cycles and subsequently recalculates the WCET bound. This process is repeated until a fixed point is reached.

VIII. Experimental Evaluation

We evaluate our analysis prototype for multi-core processors with ARM® instruction set, a shared memory bus, and Round-Robin bus arbitration. Our experiments consider cores with in-order pipelines (five stages) as well as cores that support out-of-order execution (Tomasulo dynamic scheduling, three functional units, and speculative execution). We also consider two scenarios with respect to the local instruction memories of the cores. First, we assume a local instruction scratchpad that is statically initialized with all programs executed on the core. Secondly, we consider a local instruction cache (1KiB size, least-recently-used [LRU] replacement policy) that is connected to the shared bus. Table II lists the four resulting core configurations. All core configurations assume a local LRU data cache of size 1KiB. The shared bus connects the cores to an SRAM background memory that serves accesses with a fixed latency of ten cycles. Note that we do not precisely model any particular commercial processor.

We consider a dual-core, a quad-core, and an octa-core processor per core configuration. For each resulting hardware configuration, we calculate a co-runner-insensitive WCET bound per benchmark. Our benchmark suite contains 31 benchmarks of the Mälardalen suite [26] and six larger programs generated from SCADE models[1]. We assume non-preemptive task scheduling as providing timing guarantees for preemptive multitasking is an unsolved problem for realistic hardware platforms. Note that we do not perform response time analysis. This work focuses on WCET analysis.

Our experiments take about 107 minutes on a quad-core Intel® Core™ i7 processor clocked at 2.4 GHz and provided 8 GiB of main memory.

We normalize the WCET bound and the analysis runtime per benchmark and considered processor to the corresponding values of an analysis that ignores the shared-bus interference. Table I lists the geometric means of the resulting factors for the considered hardware platforms.

The results underline the strong impact of the shared-bus interference on the WCET bounds: the average deviation factors of the WCET bounds from bounds ignoring the interference reach up to 1.756 (3.267, 6.284) for dual-core (quad-core, octa-core) processors. However, note that the calculated WCET bounds are co-runner-insensitive. Thus, they implicitly assume arbitrarily aggressive bus access behavior of the programs executed on the concurrent cores. As we have shown before [17], a co-runner-sensitive analysis can lead to a significant reduction of the WCET bounds.

In contrast to classical compositional approaches [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], our analysis prototype supports hardware platforms exhibiting indirect interference effects (cf. Section III). We estimate the additional cost of considering indirect interference effects by comparing the analysis runtime to the runtime of an analysis that ignores all interference effects (which is the main part of classical compositional timing analysis). The average increase in analysis runtime is moderate (up to 5.4 percent) for hardware platforms with in-order execution or instruction scratchpads ($Conf_{is}^{io}$, $Conf_{is}^{ooo}$, $Conf_{ic}^{io}$). The combination of out-of-order execution and instruction caches ($Conf_{ic}^{ooo}$), however, leads to a significantly higher—though still bearable—increase in analysis runtime (up to 15.9 percent on average). Intuitively, the complexity of modeling the pipeline features multiplies with the complexity of modeling the shared-bus interference by non-determinism. Note that these runtime results are a significant improvement compared to the numbers we reported in our earlier work. The improvement stems from engineering improvements (which are not in the scope of this paper) of the implementation of our analysis.

The average runtime increase factors for dual-core, quad-core, and octa-core processors with the same core configuration are essentially identical for all our experiments (the small deviations are caused by the heavy use of hash sets in our prototype implementation). Intuitively, for the considered processor core configurations, the core pipelines already converge for each access to the shared bus in a dual-core processor. As a consequence, further cycles blocked at the shared bus do not result in new pipeline states. An optimization (fast-forwarding of converged chains [17]) in our analysis prototype exploits this convergence. For analyses relying on the enumeration of all interleavings of bus access requests by the different processor cores [20], in contrast, each additional core increases the analysis runtime by a factor. Thus, such analyses do not scale to high numbers of processor cores.

IX. Future Work

Our current prototype implementation only takes into account shared-bus interference. We plan to also consider shared caches and

[1] http://www.esterel-technologies.com/products/scade-suite

	$Conf^{io}_{is}$			$Conf^{ooo}_{is}$			$Conf^{io}_{ic}$			$Conf^{ooo}_{ic}$		
	2-*Core*	4-*Core*	8-*Core*	2-*Core*	4-*Core*	8-*Core*	2-*Core*	4-*Core*	8-*Core*	2-*Core*	4-*Core*	8-*Core*
WCET bound	1.579	2.728	5.022	1.660	2.978	5.609	1.677	3.024	5.714	1.756	3.267	6.284
analysis runtime	1.033	1.033	1.028	1.054	1.046	1.051	1.050	1.037	1.038	1.159	1.152	1.149

Table I: Average deviation factors of WCET bound and analysis runtime of the calculation of co-runner-insensitive WCET bounds with respect to an analysis assuming no interference

	in-order execution	out-of-order execution
local instruction scratchpad	$Conf^{io}_{is}$	$Conf^{ooo}_{is}$
local instruction cache	$Conf^{io}_{ic}$	$Conf^{ooo}_{ic}$

Table II: Evaluated processor core configurations

cache coherence in a future version of our tool. A long-term goal is the modeling of commercially available multi-core processors with our techniques.

Furthermore, we plan to study the impact of complex processor core features like store buffers and speculation on the performance of our analysis approach. In this context, we will investigate performance improvements of our tool in order to further reduce the performance overhead due to the consideration of shared-resource interference. As a result of our studies, we will give recommendations for the design of future multi-core hardware platforms to enable their use in timing-critical embedded system.

X. Summary

We present a framework for the derivation of WCET analyses for multi-core processors. It centers around the concept of property lifting. Instances of the framework are sound WCET analyses. The framework has been successfully used in the development of a novel analysis that avoids the restrictions of existing approaches.

Acknowledgments

The authors would like to thank Mihail Asavoae, Florian Haupenthal, Max John, Jan Reineke, and Reinhard Wilhelm for many comments and interesting discussions. This work was supported by the DFG as part of the Transregional Collaborative Research Centre SFB/TR 14 (AVACS) and by the Saarbrücken Graduate School of Computer Science which receives funding from the DFG as part of the Excellence Initiative of the German Federal and State Governments.

References

[1] R. Wilhelm *et al.*, “The worst-case execution-time problem — overview of methods and survey of tools,” *ACM Trans. Embed. Comput. Syst.*, vol. 7, no. 3, pp. 36:1–36:53, 2008.

[2] S. Thesing, “Safe and precise WCET determination by abstract interpretation of pipeline models,” Ph.D. dissertation, 2004.

[3] X. Li *et al.*, “Modeling out-of-order processors for WCET analysis,” *Real-Time Syst.*, vol. 34, no. 3, pp. 195–227, 2006.

[4] A. Abel *et al.*, “Impact of resource sharing on performance and performance prediction: A survey,” in *Proceedings of the 24th Conference on Concurrency Theory*, 2013, pp. 25–43.

[5] A. Schranzhofer *et al.*, “Timing analysis for TDMA arbitration in resource sharing systems,” in *Proceedings of the 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2010, pp. 215–224.

[6] R. Pellizzoni *et al.*, “Worst case delay analysis for memory interference in multicore systems,” in *Proceedings of the 13th Conference on Design, Automation and Test in Europe*, 2010, pp. 741–746.

[7] A. Schranzhofer *et al.*, “Timing analysis for resource access interference on adaptive resource arbiters,” in *Proceedings of the 17th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2011, pp. 213–222.

[8] G. Giannopoulou *et al.*, “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*, 2012, pp. 63–72.

[9] Y. Liang *et al.*, “Timing analysis of concurrent programs running on shared cache multi-cores,” *Real-Time Systems*, vol. 48, pp. 638–680, 2012.

[10] J. Nowotsch, “Interference-sensitive worst-case execution time analysis for multi-core processors,” Ph.D. dissertation, 2014.

[11] D. Dasari *et al.*, “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*, 2011, pp. 1068–1075.

[12] D. Dasari and V. Nélis, “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*, 2012, pp. 1450–1457.

[13] S. Schliecker and R. Ernst, “Real-time performance analysis of multiprocessor systems with shared memory,” *ACM Trans. Embedded Comput. Syst.*, vol. 10, no. 2, p. 22, 2010.

[14] S. Schliecker *et al.*, “Response time analysis in multicore ECUs with shared resources,” *IEEE Trans. Industrial Informatics*, vol. 5, no. 4, pp. 402–413, 2009.

[15] S. Altmeyer *et al.*, “A generic and compositional framework for multicore response time analysis,” in *Proceedings of the 23rd International Conference on Real Time Networks and Systems*, 2015, pp. 129–138.

[16] S. Chattopadhyay *et al.*, “A unified WCET analysis framework for multi-core platforms,” in *Proceedings of the 18th IEEE Real-Time and Embedded Technology and Applications Symposium*, 2012, pp. 99–108.

[17] M. Jacobs *et al.*, “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*, 2015, pp. 193–202.

[18] S. Hahn *et al.*, “Towards compositionality in execution time analysis – definition and challenges,” in *Proceedings of the 6th International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems*, 2013.

[19] T. Lundqvist and P. Stenstrom, “Timing anomalies in dynamically scheduled microprocessors,” in *Proceedings of the 20th IEEE Real-Time Systems Symposium*, 1999, pp. 12–21.

[20] T. Kelter and P. Marwedel, “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*, 2014, pp. 142–158.

[21] J. Reineke and R. Sen, “Sound and efficient WCET analysis in the presence of timing anomalies,” in *Proceedings of the 9th International Workshop on Worst-Case Execution Time Analysis*, 2009, pp. 98–108.

[22] M. Jacobs, “Improving the precision of approximations in WCET analysis for multi-core processors,” in *Proceedings of the 7th Junior Researcher Workshop on Real-Time Computing*, 2013, pp. 1–4.

[23] P. Cousot and R. Cousot, “Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints,” in *Conference Record of the 4th Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, 1977, pp. 238–252.

[24] R. Pellizzoni and M. Caccamo, “Impact of peripheral-processor interference on WCET analysis of real-time embedded systems,” *IEEE Transactions on Computers*, vol. 59, pp. 400–415, 2010.

[25] Y.-T. S. Li and S. Malik, “Performance analysis of embedded software using implicit path enumeration,” in *Proceedings of the 32nd Annual ACM/IEEE Design Automation Conference*, 1995, pp. 456–461.

[26] J. Gustafsson *et al.*, “The mälardalen WCET benchmarks - past, present and future,” in *Proceedings of the 10th International Workshop on Worst-Case Execution Time Analysis*, 2010.