Precise Worst-Case Execution Time Analysis for Processors with Timing Anomalies

Raimund Kirner, Albrecht Kadlec, Peter Puschner

Institut für Technische Informatik
Technische Universität Wien, Austria
{raimund,albrecht,peter}@vmars.tuwien.ac.at

Abstract

This paper explores timing anomalies *in WCET analysis. Timing anomalies add to the complexity of WCET analysis and make it hard to apply divide-and-conquer strategies to simplify the WCET assessment.*

So far, timing anomalies have been described as a problem that occurs when the WCET of a control-flow graph is computed from the WCETs of its subgraphs, i.e., from a series decomposition*. This paper extends the state of the art by (i) showing that timing anomalies can as well occur in a* parallel decomposition *of the WCET problem, i.e., when complexity is reduced by splitting the hardware state space and performing a separate WCET analysis for hardware components that work in parallel, (ii) proving that the potential occurrence of parallel timing anomalies makes the parallel decomposition technique unsafe (i.e., one cannot guarantee that the calculated WCET bound does not underestimate the WCET), and (iii) identifying special cases of parallel timing anomalies for which the parallel decomposition technique is safe. The latter provides an important hint to hardware designers on their way to constructing predictable hardware components.*

1. Introduction

The knowledge of the worst-case execution time (WCET) of software components is a prerequisite for ensuring the timeliness of a real-time system. Since the end of the 1980s significant effort has been spent on research towards the development of WCET analysis tools.

The research leading to these results has received funding from the Austrian Science Fund (Fonds zur Förderung der wissenschaftlichen Forschung) within the research project "Compiler-Support for Timing Analysis" (COSTA) under contract P18925-N13.

The two main tasks of WCET analysis tools are the *control-flow analysis* (also called *path analysis* [1] or *high-level analysis*) that determines the (in)feasible paths in a program and the *processor-behavior analysis* (also known as *hardware modeling* [1] or *low-level analysis*) that assesses instruction timing [2].

Within this paper we discuss an open problem of *processor-behavior analysis*, namely the occurrence of so-called *timing anomalies* [3], [4], [5]. Timing anomalies are a challenge for WCET analysis, because they violate the continuity properties "proportionality" and "monotony" of program execution time.

Timing anomalies so far have only been discussed in the context of composing instruction sequences. We motivate in Section 3 that they are also a challenge for multiple phases of processor-behavior analysis, which we call *parallel composition*. In Section 4 we define *parallel timing anomalies* as timing effects due to changes of the initial state and show that they can occur in practice. In Section 5 we discuss two fundamental techniques of parallel composition and prove as an impossibility result that in case of arbitrary forms of parallel timing anomalies parallel composition does not provide safe WCET bounds. But we also prove that in case of restricted forms of parallel timing anomalies it is still possible to obtain safe WCET bounds. In Section 6 we discuss practical issues of timing anomalies on WCET analysis and discuss methods of how to avoid anomalous behavior.

2. Worst-Case Execution Time Analysis

WCET analysis is about finding the longest feasible path through a program, where length means execution time [1], [2]. For example, the *implicit path-enumeration technique* allows to consider arbitrary linear flow constraints [6], [7]. One or more program analysis phases

precede the longest path search to calculate the instruction timing [8], [9].

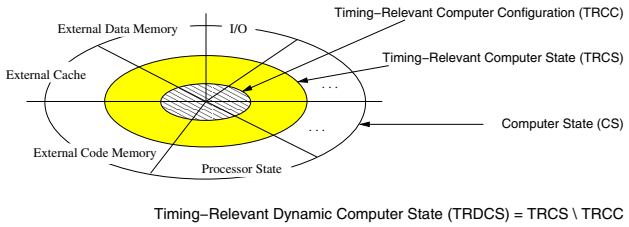


Figure 1. The Timing-Relevant State TRDCS

On modern processors with peak-performance improving features like caches or pipelines, the WCET of an instruction sequence I depends on the set of initial computer states that potentially reach the beginning of instruction sequence I. Since a precise notion of computer state is important for WCET analysis, we introduce the *timing-relevant dynamic computer state* (TRDCS) for a program scope S. As shown in Figure 1, the TRDCS includes only those parts of the timing-relevant computer state that are changed within a program scope S. Those parts of the timing-relevant computer state that are changed only outside of S are called computer configuration and are not part of the TRDCS.

2.1. Notation

The following sections discuss several formal properties on WCET analysis. To keep the definition of these properties short and intuitive we use the following notation:

$T(I, s)$ ... the execution time of an instruction sequence $I = I_0 \circ I_1 \circ \ldots \circ I_n$ with the initial TRDCS s. The operator $\circ$ combines individual instructions or instruction sequences to a combined instruction sequence.

$T_{hw_A}(I, a)$ ... the *component latency* of processor component hw_A when executing instruction sequence I with initial local state $a \in A$, where A is the TRDCS state space of processor component hw_A. The component latency of a processor component is the cumulated time where this component performs some data processing. That part of the execution of I where hw_A is inactive does not contribute to its component latency.
For example, the component latency of a data cache when executing an instruction sequence I is the cumulated time the data cache needs to exchange data with the main memory and with the processor registers (including waiting time to get data ready). The TRDCS of the hardware model can be optimized for the instruction sequences of interest. In the data cache, for example, all addresses that will not be used by the instruction sequences of interest can be subsumed as "not part of the TRDCS".

$T_{max}(I, S) = max(\{T(I, s) \mid s \in S\})$ ... the WCET of instruction sequence I where S is the set of potential initial states for execution of I.

$\Delta(I, s, s') = T(I, s') - T(I, s)$ ... the difference of execution time of instruction sequence I for different initial states s and s'.

$\Delta_{hw_A}(I, a, a') = T_{hw_A}(I, a') - T_{hw_A}(I, a)$ ... the difference of component latency of processor component hw_A when executed from different local initial states a and a' with $a, a' \in A$.

3. Parallel Decomposition

WCET analysis with parallel decomposition is a technique to reduce the complexity of processor-behavior analysis.

The idea of parallel decomposition is to calculate the WCET $T_{max}(I, S)$ of an instruction sequence $I = I_0 \circ I_1 \circ \ldots \circ I_n$ in two steps. Before this calculation, the TRDCS S is partitioned into $S = A \cup B$, where A is the state space of a processor component hw_A and B is the state space of other components hw_B in the processor. For example, the hardware component hw_A may be the instruction cache and the state fraction B may cover the pipeline and the other processor components. For the timing function $T(I, s)$ of an instruction sequence I given in Figure 2, the corresponding timing function with the partitioned state space A and B is shown in Figure 3. The state spaces of hw_A and hw_B may also overlap ($A \cap B \neq \emptyset$).

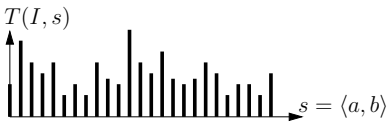


Figure 2. Timing of Non-Partitioned TRDCS

In the first step, the timing of processor component hw_A is analyzed and one state $a \in A$ is chosen (details describing the choice of a will follow below). Based on this result, the overall processor timing is analyzed in the second step by searching the state space B while using the result for state $a \in A$ to model the timing of hw_A.

The challenge is to find a composable timing calculation method that can be used to calculate safe WCET bounds for the target processor of interest. Concrete calculation methods are discussed in Section 5.

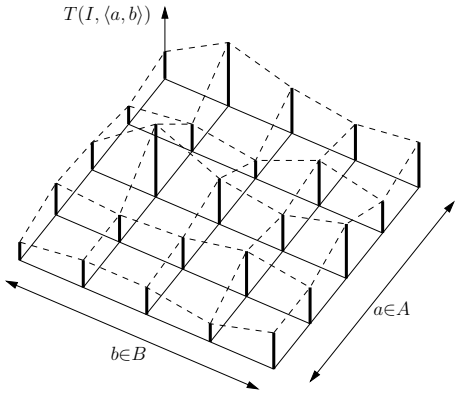


Figure 3. Timing of Partitioned TRDCS

4. Timing Anomalies

The name *timing anomalies* is used to describe system behavior where relaxing some constraints leads to an increase of the system timing. This is typically caused due to a greedy scheduler that cannot foresee the future impact of its local decisions. With respect to WCET analysis, for example, such a constraint may require the execution of two instruction sequences to finish within a given deadline. Decreasing the execution time of the first instruction sequence relaxes the contraint for the second instruction sequence to finish within the deadline, which can lead to timing anomalies.

4.1. Related Work on Timing Anomalies

Program execution time is not the first field where *timing anomalies* have been observed. For example, Graham described this effect for task scheduling [10]. He has shown that a greedy task scheduler can produce a longer schedule if the scheduling constraints are weakened, e.g., by using shorter tasks, less dependencies, or more processors.

Lundqvist and Stenström first described *timing anomalies* in the context of WCET analysis [3]. Their definition of timing anomalies is semi-formal. They have shown an example where a change from a cache hit to a cache miss of the first instruction of an instruction sequence on a processor with out-of-order pipeline and instruction cache can result in a decrease of the total execution-time. However, it has been shown that it is rather challenging to understand the potential triggers of timing anomalies. For example, Lundqvist and Stenström believed that it requires an out-of-order pipeline to trigger timing anomalies [3], which later turned out as too specific, see below.

Schneider developed an integrated WCET analysis method, i.e., he integrated response-time analysis with WCET analysis. He did this on a PowerPCC 755, where he demonstrated that timing anomalies occur on real processors [11]. Besides, he has also demonstrated the occurrence of the so-called *domino effect*, i.e., different states at the header of a loop do never converge during execution of the loop. Domino effects do not necessarily cause timing anomalies. In the concrete example shown by Schneider it did result in a *strong timing anomaly*, as he showed that a delay caused in the loop header results in a constant delay in each loop iteration, resulting in a total delay that is at least a linear function of the loop bound. Berg has shown an example of a *domino effect* that results in a *weak timing anomaly* with a constant execution-time change each loop iteration [12].

Wenzel et al. have analyzed different patterns of processor architectures to gain knowledge about the possible triggers of timing anomalies [13], [4]. They have shown that timing anomalies can occur even on processors with in-order execution. Further, Wenzel et al. provide a necessary precondition for the potential occurrence of timing anomalies, the *resource allocation criterion*. However, the concrete formulation of the criterion was a bit too restrictive as it covers only cases in which exactly one instruction changes its timing. This criterion needs to be generalized to cover timing anomalies caused by speculative execution [5] or certain cache replacement policies like *pseudo-round robin* [14].

Reineke et al. for the first time provide a formal definition of timing anomalies in the context of program execution time. Their definition of timing anomalies is based on a transition system as processor model where a timing anomaly occurs if the WCET path within a local scope is not part of the WCET path of a surrounding scope [5]. This was an important step towards improving the understanding of timing anomalies. However, this formalization of timing anomalies is rather complex, making it clumsy to use as a tool for exploring safeness properties of WCET analyses. Further, the authors enumerated three different sources of timing anomalies without claiming this to be a complete list: speculation, scheduling, and cache effects. Reineke et al. only discuss the type of timing anomaly that Eisinger et al. call a *strong timing anomaly*: the case where a local increase of execution time leads to a global decrease [15]. The other case, where a local increase of execution time leads to an even larger global increase - called *weak timing anomaly* - is not treated in the context of WCET analysis because it does not hinder an efficient search of the worst-case. The definition of a timing anomaly given by Reineke et al. is specific to WCET analysis, while the original definition given by Lundqvist includes also processor behavior that is not a challenge for WCET analysis.

The research described above discusses timing anomalies only in the context of serial decomposition of WCET

analysis. Kirner et al. for the first time describe a class of timing anomalies not considered so far: timing anomalies in the context of WCET analysis using parallel decomposition. A timing anomaly in case of parallel decomposition is defined as the situation where the worst-case initial state of a hardware/processor sub-component is not part of the worst-case initial state of the total hardware/processor [16]. So far the authors formalized this new type of timing anomalies without discussing it in the context of the previously mentioned timing anomalies based on local changes within an instruction sequence.

4.2. Parallel Timing Anomalies

Recently it was found that timing anomalies do not only occur between the timing of two subsequent instruction sequences, but also between the component latency of processor components and the total execution time [16]. We call these timing anomalies "parallel timing anomalies" because they are challenging for the parallel decomposition described in Section 3.

Parallel timing anomalies are formally defined by Definition 4.1. The timing anomalies are defined over the component latency $T_{hw_A}(I, s)$ and the total execution time $T(I, s)$ of an instruction sequence I. Component latency is explained in Section 2.1. Concrete examples of timing anomalies are given in Section 4.3.

Definition 4.1: **(Parallel Timing Anomalies)** Given a partitioned TRDCS $S = A \cup B$ with the timing behavior (component latency) of hardware component hw_A modeled as $T_{hw_A}(I, a)$, the timing behavior $T(I, \langle a, b\rangle)$ of an instruction sequence I on a processor is called a *parallel timing anomaly*, iff at least one of the following two properties holds:

TA-P-I "Parallel Inversion"
$$\exists a, a' \in A, b \in B.$$
$$\Delta_{hw_A}(I, a, a') > 0 \ \wedge \ \Delta(I, \langle a, b\rangle, \langle a', b\rangle) < 0$$

TA-P-A "Parallel Amplification"
$$\exists a, a' \in A, \exists b \in B.$$
$$0 < \Delta_{hw_A}(I, a, a') < \Delta(I, \langle a, b\rangle, \langle a', b\rangle)$$

The parallel timing anomaly **TA-P-I** states that a change of the component latency of instruction sequence I for the processor component hw_A results in a change in the opposite direction for the execution time over the state $\langle a, b\rangle \in A \times B$. Due to this behavior **TA-P-I** is also called "parallel inversion".

Analog to the series amplification, the parallel timing anomaly **TA-P-A** states that a change of the component latency of instruction sequence I for the processor component hw_A results in a larger change in the same direction for the execution time over the state $\langle a, b\rangle \in A \cup B$. Thus, the parallel timing anomaly **TA-P-A** is also called "parallel amplification".

In case of parallel timing anomalies, both **TA-P-I** and **TA-P-A** can be considered to be "strong (parallel) timing anomalies", since both potentially invalidate the parallel decomposition described in Section 3. However, as described in Section 5, not all occurrences of **TA-P-I** and of **TA-P-A** are challenging.

Analogously to series timing anomalies, there exists also a more strict definition for parallel timing anomalies than that given in Definition 4.1. This more strict definition of parallel timing anomalies is given in Definition 4.2. These timing anomalies are called *worst-case parallel timing anomalies*, since they describe exactly the cases that cause problems for efficient WCET analysis with parallel decomposition.

Definition 4.2: **(Worst-Case Parallel Timing Anomalies)** Given a partitioned TRDCS $S = A \cup B$ with the timing behavior (component latency) of hardware component hw_A modeled as $T_{hw_A}(I, a)$, the timing behavior $T(I, \langle a, b\rangle)$ of an instruction sequence I on a processor is called a *worst-case parallel timing anomaly*, iff at least one of the following two properties holds:

TAW-P-I **"Worst-Case Parallel Inversion"**
$$\exists a \in A, b \in B, \forall a' \in A_{max}, \forall b' \in B_{A,max}(a').$$
$$\Delta_{hw_A}(I, a, a') > 0 \ \wedge \ \Delta(I, \langle a, b\rangle, \langle a', b\rangle) < 0$$

TAW-P-A **"Worst-Case Parallel Amplification"**
$$\exists a \in A, b \in B, \forall a' \in A_{min}, \forall b' \in B_{A,max}(a').$$
$$0 < \Delta_{hw_A}(I, a', a) < \Delta(I, \langle a', b'\rangle, \langle a, b\rangle)$$

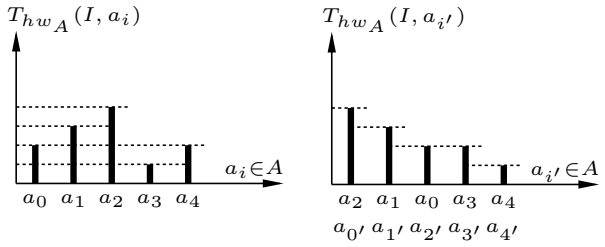
with

$$A_{min} = \{a \in A \mid \forall a' \in A.\ T_{hw_A}(I, a') \geq T_{hw_A}(I, a)\}$$
$$A_{max} = \{a \in A \mid \forall a' \in A.\ T_{hw_A}(I, a') \leq T_{hw_A}(I, a)\}$$
$$B_{A,max}(a) = \{\ b \in B \mid \forall b' \in B.\ T(I, \langle a, b\rangle) \geq T(I, \langle a, b'\rangle)\ \}$$

Note that the definition given in Definition 4.2 is almost identical with the definition given in Definition 4.1, with the small difference that it uses $\forall a' \in A_{max}$, $\forall b' \in B_{A,max}(a')$ respectively $\forall a' \in A_{min}$, $\forall b' \in B_{A,max}(a')$ instead of $\exists a \in A$. The worst-case timing anomalies are more specific than the others, which is, besides the specific elements to compare, due to the $\forall$ quantifier instead the $\exists$ quantifier. Analogous to series timing anomalies, the generic form of timing anomalies given in Definition 4.1 can imply for a specific program (with the right set of reachable states) the occurrence of the worst-case timing anomalies given in Definition 4.2.

4.2.1. Visualization of Parallel TAs. To visualize parallel timing anomalies we assume that the TRDCS is partitioned into A and B as explained in Section 3. The component latency of instruction sequence I on hardware component hw_A is assumed to be as shown in Figure 4.a. It is easier to identify the occurrence of parallel timing anomalies if the component latencies for the different states a_i are in a (decreasing) order. To get a decreasing order we

relabel the states a_i into states a'_i as shown in Figure 4.b. We have to compare decreasing component latency with the overall execution time to find occurrences of parallel timing anomalies.



a) arbitrary state order of component hw_A

b) re-ordered by execution time and renamed

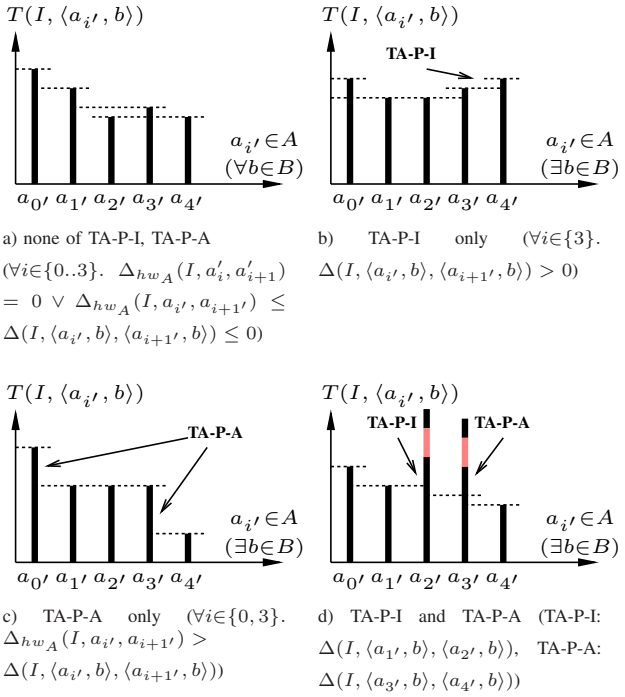
Figure 4. Component Latency of Processor Component hw_A for Instruction Sequence I (examples of corresponding overall execution times causing timing anomalies are given in Figure 5)

Figure 5.a shows how the execution time $T(I, \langle a'_i, b\rangle)$ has to look like in case there are no parallel timing anomalies: for all $b \in B$ the execution times are decreasing like the component latency does. The change of execution time is not larger than the change of the component latency. However, the only exception are those cases where the component latency does not change. Whatever the change of the execution time is, as long as the component latency does not change, it is not considered to be a timing anomaly. As described in Section 5, such cases can be handled by doing the parallel composition for multiple component latencies of hw_A.

Figure 5.b shows an example of timing anomaly **TA-P-I** (parallel inversion): the change from state a'_3 to state a'_4 where the execution time increases while the component latency decreases. Note that between state a'_2 and state a'_3 there is no timing anomaly, though the execution time also increases. This is not a timing anomaly because in this case the component latency of hw_A does not change.

Figure 5.c shows examples of timing anomaly **TA-P-A**: between states a'_0 and a'_1 and between states a'_3 and a'_4. In those cases the execution time decreases more than the component latency of hw_A does.

Of course, it can also happen that both parallel timing anomalies, **TA-P-I** and **TA-P-A**, occur. As described in Section 5, such a scenario in general does not invalidate parallel decomposition. But it turns problematic when **TA-P-I** and **TA-P-A** do occur for the same $b \in B$. The limitations of parallel decomposition in case of such a scenario are described in Section 5.3.1. Figure 5.d shows such a scenario: the execution time of state $\langle a'_3, b\rangle$ and state



a) none of TA-P-I, TA-P-A ($\forall i \in \{0..3\}$. $\Delta_{hw_A}(I, a'_i, a'_{i+1}) = 0 \vee \Delta_{hw_A}(I, a_{i'}, a_{i+1'}) \leq \Delta(I, \langle a_{i'}, b\rangle, \langle a_{i+1'}, b\rangle) \leq 0$)

b) TA-P-I only ($\forall i \in \{3\}$. $\Delta(I, \langle a_{i'}, b\rangle, \langle a_{i+1'}, b\rangle) > 0$)

c) TA-P-A only ($\forall i \in \{0, 3\}$. $\Delta_{hw_A}(I, a_{i'}, a_{i+1'}) > \Delta(I, \langle a_{i'}, b\rangle, \langle a_{i+1'}, b\rangle)$)

d) TA-P-I and TA-P-A (TA-P-I: $\Delta(I, \langle a_{1'}, b\rangle, \langle a_{2'}, b\rangle)$, TA-P-A: $\Delta(I, \langle a_{3'}, b\rangle, \langle a_{4'}, b\rangle)$)

Figure 5. Examples for Both Types of Parallel Timing Anomalies

$\langle a'_4, b\rangle$ are not bounded by the changes of the component latency of hw_A.

4.3. Examples of Timing Anomalies

In the previous section we have shown how a processor behaves in case of timing anomalies. In this section we show examples of concrete hardware patterns that can cause such timing anomalies. It is not fully understood how to determine efficiently whether a hardware exhibits timing anomalies. The following presents known instances of timing anomalies, which might help to identify further sources of timing anomalies.

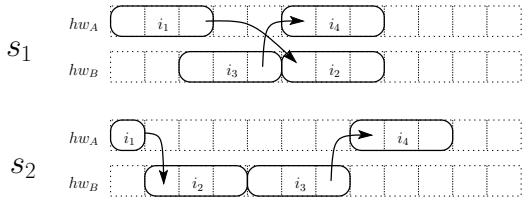


Figure 6. Example of TA-P-I (out-of-order pipeline + cache + data dependencies)

One of the first known potential sources of timing anomalies is out-of-order execution. Wenzel et al. has constructed simple patterns of hardware architectures and

studied whether they may exhibit timing anomalies [13], [4]. Figure 6 shows a simple example of *inversion timing anomaly*, which has been taken from [4]. The assumed processor has an out-of-order pipeline with two non-overlapping resources. Non-overlapping resources means that there are no instructions that can choose from more than one alternatives during each resource allocation. The bold arrows show data dependencies, which restrict the set of different possible executions through the pipeline. The timing anomalies in this example show up due to the combined effect of data dependencies and the out-of-order execution. Figure 7 shows for the same processor model an example of *amplification timing anomaly* which has been also taken from [4]. Both examples of timing anomalies can manifest as series timing anomalies or as parallel timing anomalies.

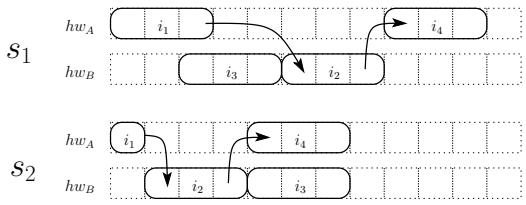


Figure 7. Example of TA-P-A (out-of-order pipeline + cache + data dependencies)

Most patterns of hardware and software that can cause series timing anomalies can also cause parallel timing anomalies. However, there is an interesting difference between them: in case of series timing anomalies, the inversion is challenging, but not the amplification. As we show by Theorem 5.7 and Theorem 5.9, in case of parallel timing anomalies, only the coupled occurrence of inversion and amplification is challenging.

5. WCET Analysis with Parallel Composition

In Section 3 we described the basic idea of reducing analysis complexity by using parallel decomposition of a state TRDCS into two sets A and B. The challenge is to find a composable timing calculation method that can be used to calculate safe WCET bounds for the target processor of interest. In the following we describe two different timing-composition techniques and analyze their correctness in case of parallel timing anomalies. These are the only two possible approaches of parallel composition that first search the state of hardware hw_A and hw_B independently based on the maxima, minima, and maximal variation of hardware component hw_A.

5.1. Delta-Composition

The first prototypical technique to derive the maximum overall instruction timing of an instruction sequence I based on a decomposition of the TRDCS into two state fractions A and B is called *Delta-Composition*. The principle of *Delta-Composition* is given in Figure 8:

1) $\Delta_{hw_A,max}$, the maximum variability (Δ_{hw_A}) of $T_{hw_A}(I,a)$ is determined:
   $\Delta_{hw_A,max} = \max_{a,a' \in A} |T_{hw_A}(I,a) - T_{hw_A}(I,a')|$
2) The set A_{min} of local states $a_{hw_A,min} \in A$ where $T_{hw_A}(I,a)$ is minimal, is determined: $A_{min} = \{a \in A \mid \forall a' \in A.\ T_{hw_A}(I,a') \geq T_{hw_A}(I,a)\}$.
3) For each $a_{hw_A,min} \in A_{min}$ the overall timing function $T(I, \langle a_{hw_A,min}, b \rangle)$ with fixed local state $a_{hw_A,min}$ is selected.
4) $b_{dc,max}$, the partial state $b \in B$ where $T(I, \langle a_{hw_A,min}, b \rangle)$ is maximal, is determined: $b \in B_{A,max}(a_{hw_A,min})$ with $B_{A,max}(a) = \{\ b \in B \mid \forall b' \in B.\ T(I, \langle a, b \rangle) \geq T(I, \langle a, b' \rangle)\ \}$.
5) $\Delta_{hw_A,max}$ is added to $T(I, \langle a_{hw_A,min}, b_{dc,max} \rangle)$.

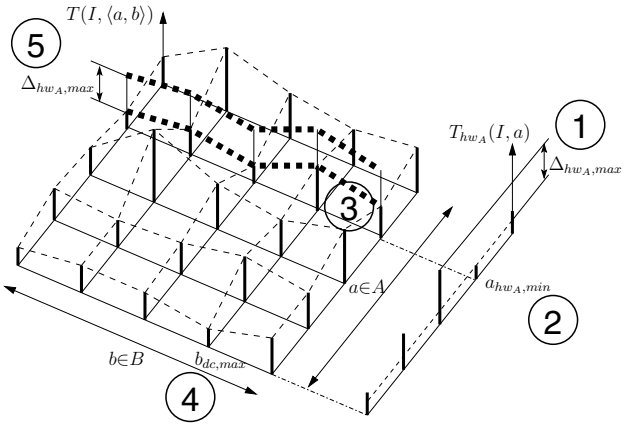


Figure 8. Delta-Composition of TRDCS

Equation 1 shows how the maximum instruction timing is calculated with *Delta-Composition* ($T_{dc}(I)$). Note that the Delta-Composition potentially overestimates the WCET: $T_{dc}(I) \geq T_{max}(I)$.

$$T_{dc}(I) = \max_{a \in A_{min}, b \in B} T(I, \langle a, b \rangle) + \Delta_{hw_A,max} \quad (1)$$

In case there are multiple states $a \in A_{min}$ of minimal latency then Delta-Composition evaluates each of these minima and takes the overall maxima. The computational cost for $T_{dc}(I)$ is $O((|A| + |A_{min}| \cdot |B|)) \cdot |I|)$. Thus, the more minima $a \in A_{min}$ exist, the higher is the computational cost of Delta-Composition. In the extreme case of $A = A_{min}$ the Delta-Composition degrades to searching all states $s \in A \times B$. However, this extreme case of $A = A_{min}$ rarely seems to be a real problem, because

in that case the whole set A has no influence on the timing and thus cannot be part of the TRDCS. However the worst case of complexity is the scenario $|A| - 1 = |A_{min}|$.

Theorem 5.1 describes the sufficient and necessary condition about the hardware behavior such that Delta-Composition ($T_{dc}(I, s)$) is safe, i.e., that it provides an upper bound for the execution time of an instruction sequence I.

Theorem 5.1: **Safeness of Delta-Composition:** Based on above definitions of A_{min}, $B_{A,max}(a)$, and $\Delta_{hw_A,max}$, the Delta-Composition allows to provide a safe WCET bound on processor hardware whose timing characteristics obey the following sufficient and necessary condition (proof given in [17]):

$$\forall a{\in}A, \forall b{\in}B, \exists a'{\in}A_{min}, \exists b'{\in}B_{A,max}(a').$$
$$\Delta_{hw_A}(I, a', a) > 0 \;\rightarrow$$
$$\Delta(I, \langle a', b'\rangle, \langle a, b\rangle) \leq \Delta_{hw_A,max} \quad (2)$$

Condition 2 states that there exists at least one local best-case state $a'{\in}A_{min}$ ($a_{hw_A,min}$) such that whenever the state of a hardware component hw_A changes from any state $a \notin A_{min}$ to this specific best-case state a', the resulting change in the execution time of instruction-sequence I ($\Delta(I, \langle a', b'\rangle, \langle a, b\rangle)$) is not higher than the maximum change that is possible in the component latency of the hardware component hw_A ($\Delta_{hw_A}(I, a', a)$).

The correctness condition given in Equation 2 is the negation of **TAW-P-A** (see Definition 4.1).

5.2. Max-Composition

The second prototypical technique to derive the maximum overall instruction timing of an instruction sequence I based on a decomposition of the TRDCS into two state fractions A and B is called *Max-Composition*. The principle of *Max-Composition* is given in Figure 9:

1) The set A_{max} of local states $a_{hw_A,max} \in A$ where $T_{hw_A}(I, a)$ is maximal, is determined: $A_{max} = \{a \in A \mid \forall a' \in A.\ T_{hw_A}(I, a') \leq T_{hw_A}(I, a)\}$.
2) For each $a_{hw_A,max} \in A_{max}$ the overall timing function $T(I, \langle a_{hw_A,max}, b\rangle)$ with fixed local state $a_{hw_A,max}$ is selected.
3) $b_{mc,max}$, the partial state $b \in B$ where $T(I, \langle a_{hw_A,max}, b\rangle)$ is maximal, is determined: $b{\in}B_{A,max}(a_{hw_A,max})$ with $B_{A,max}(a) = \{\ b \in B \mid \forall b' \in B.\ T(I, \langle a, b\rangle) \geq T(I, \langle a', b'\rangle)\ \}$.

Equation 3 shows how the maximum instruction timing is calculated with *Max-Composition* ($T_{mc}(I)$). Max-Composition provides a precise WCET bound: $T_{mc}(I) = T_{max}(I)$.

$$T_{mc}(I) = \max_{a \in A_{max}, b \in B} T(I, \langle a, b\rangle) \quad (3)$$

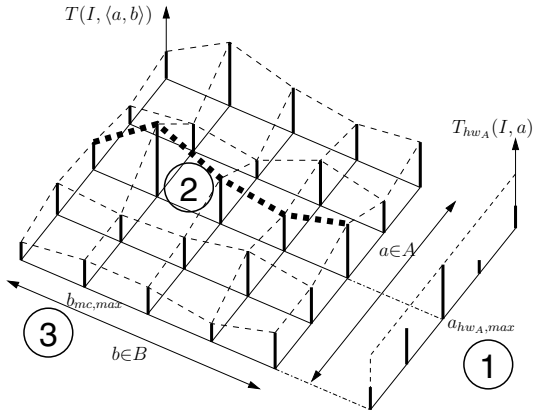


Figure 9. Max-Composition of TRDCS

In case there are multiple states $a \in A_{max}$ of maximal latency then Max-Composition evaluates each of these maxima and takes the overall maxima. The computational cost for $T_{mc}(I)$ is $O((|A| + |A_{max}| \cdot |B|)) \cdot |I|)$. Thus, the more maxima $a \in A_{max}$ exist, the higher is the computational cost of Max-Composition. In the extreme case of $A = A_{max}$ the Max-Composition degrades to searching all states $s \in A \times B$. However, this extreme case of $A = A_{max}$ rarely seems to be a real problem. Because in the case $A = A_{max}$ it is typically the case that the component latency of hardware component hw_B is independent of the component latency of hardware component hw_A, and hence the state space A is not part of the TRDCS.

Theorem 5.2 describes the sufficient and necessary condition about the hardware behavior such that Max-Composition ($T_{mc}(I, s)$) is safe, i.e., that it provides an upper bound for the execution time of an instruction sequence I.

Theorem 5.2: **Safeness of Max-Composition:** Based on above definition of A_{max} the Max-Composition allows to provide a safe WCET bound on processor hardware whose timing characteristics obey the following sufficient and necessary condition (proof given in [17]):

$$\forall a{\in}A, \forall b{\in}B, \exists a'{\in}A_{max}, \exists b'{\in}B_{A,max}(a').$$
$$\Delta_{hw_A}(I, a, a') > 0 \;\rightarrow\; \Delta(I, \langle a, b\rangle, \langle a', b'\rangle) \geq 0 \quad (4)$$

Condition 4 states that there exists at least one local worst-case state $a' \in A_{max}$ ($a_{hw_A,max}$) such that whenever the state of a hardware component hw_A changes from any state $a \notin A_{max}$ to this specific worst-case state a', then the execution time of instruction sequence I must not decrease when also changing the state of hardware component hw_B from any state $b \in B$ to one of the states $b' \in B_{A,max}(a')$ ($\Delta(I, \langle a, b\rangle, \langle a', b'\rangle) \geq 0$).

The correctness condition given in Equation 4 is the negation of **TAW-P-I** (see Definition 4.1).

5.3. Safeness of Parallel Composition

The following two theorems state which type of parallel timing anomaly are a challenge for the correctness of Delta-Composition and Max-Composition.

Theorem 5.3: **Timing-Composability without TA-P-I:** The absence of timing anomalies of type **TA-P-I** on a processor and a given instruction sequence is
a) sufficient for the correctness of Max-Composition,
b) not necessary for the correctness of Max-Composition,
c) not sufficient for the correctness of Delta-Composition
(proof given in [17]).

Theorem 5.4: **Timing-Composability without TA-P-A:** The absence of timing anomalies of type **TA-P-A** on a processor and a given instruction sequence is
a) sufficient for the correctness of Delta-Composition,
b) not necessary for the correctness of Delta-Composition,
c) not sufficient for the correctness of Max-Composition
(proof given in [17]).

Corollary 5.5: Concluding from Theorem 5.3 and Theorem 5.4, processor hardware exhibiting timing anomalies of at most one of the types **TA-P-I** or **TA-P-A**, without knowing which one it is, can safely be analyzed by applying both, *Delta-Composition* (Equation 1) and *Max-Composition* (Equation 3) simultaneously:

$$T_{dmc}(I) = \max\Big(T_{dc}(I),\ T_{mc}(I)\Big) \tag{5}$$

From Corollary 5.5 it follows that parallel timing anomalies are not a serious problem as long as only one type of them occurs. Note that since the Delta-Composition is not tight, $T_{dmc}(I)$ also does not have to be tight: $T_{dmc}(I) \geq T_{max}(I)$.

5.3.1. Coupled Parallel Timing Anomaly. In the previous section we have shown that parallel composition can be safe if at most one type of parallel timing anomalies is possible.

In the following we analyze in more detail what happens if both types of parallel timing anomalies (**TA-P-I** and **TA-P-A**) occur. In this case we can differ between the case where both types of parallel timing anomalies occur only for different states $b \in B$ and $b' \in B$ (discussed in Section 5.3.2) and the more severe case where they also occur for the same state $b \in B$. The latter case is discussed in the following.

A formal definition of the case where both types of timing anomalies **TA-P-I** and **TA-P-A** occur for the same state $b \in B$ is given in Definition 5.6. To simplify its reference, we have named this case as **TA-P-C** where "C" stands for the coupled (same $b \in B$) of both parallel timing anomalies.

Definition 5.6: **(TA-P-C: Coupled Parallel Timing Anomaly)** Given a partitioned TRDCS $S = A \cup B$ with the component latency of hardware component hw_A modeled as $T_{hw_A}(I, a)$, the timing behavior $T(I, \langle a, b\rangle)$ of an instruction sequence I on a processor is called a *coupled parallel timing anomaly*, iff the following property holds:

$$\begin{aligned} &\exists a_1, a_2, a_3, a_4 \in A, \exists b \in B.\ (\\ &\quad \Delta_{hw_A}(I, a_1, a_2) > 0 \wedge \\ &\quad \Delta(I, \langle a_1, b\rangle, \langle a_2, b\rangle) < 0) \wedge \\ &\quad (0 < \Delta_{hw_A}(I, a_3, a_4) < \Delta(I, \langle a_3, b\rangle, \langle a_4, b\rangle)) \end{aligned} \tag{6}$$

Above definition combines the definitions of TA-P-I and TA-P-A given in Definition 4.1. The word "coupled" signals that both types of timing anomalies occur for the same state $b \in B$.

Theorem 5.7 states that timing anomalies of type **TA-P-C** can only be bounded by searching the whole state space $A \times B$. This is actually an impossibility result for applying efficient parallel composition whenever the occurrence of **TA-P-C** is possible.

Theorem 5.7: **Non-Composability with Coupled Parallel Timing Anomalies:** On processor hardware exhibiting parallel timing anomalies of type **TA-P-C** as defined in Definition 5.6 with a state partitioning into two partions A and B, there are no safe parallel composition techniques without analyzing the whole combined state space $A \times B$. (proof given in [17])

5.3.2. Exclusive Parallel Timing Anomaly. We call the case where the two types of parallel timing anomalies (**TA-P-I** and **TA-P-A**) occur only for different states $b \in B$ and $b' \in B$ *exclusive parallel timing anomaly*, which is formally defined in Definition 5.8. To simplify its reference, we have named this case as **TA-P-E** where "E" stands for the exclusive occurrence of either **TA-P-I** or **TA-P-A**.

Definition 5.8: **(TA-P-E: Exclusive Parallel Timing Anomaly)** Given a partitioned TRDCS $S = A \cup B$ with the component latency of hardware component hw_A modeled as $T_{hw_A}(I, a)$, the timing behavior $T(I, \langle a, b\rangle)$ of an instruction sequence I on a processor is called an *exclusive parallel timing anomaly*, iff the following property holds:

$$\begin{aligned} &\exists a_1, a_2, a_3, a_4 \in A, \exists b_1, b_2 \in B.\quad (b_1 \neq b_2)\ \wedge \\ &\quad \Delta_{hw_A}(I, a_1, a_2) > 0 \wedge \\ &\quad \Delta(I, \langle a_1, b_1\rangle, \langle a_2, b_1\rangle) < 0 \wedge \\ &\quad (\ 0 < \Delta_{hw_A}(I, a_3, a_4) < \Delta(I, \langle a_3, b_2\rangle, \langle a_4, b_2\rangle)\ ) \end{aligned} \tag{7}$$

Above definition allows the occurrence of both, **TA-P-I** and **TA-P-A** as given in Definition 4.1. But the word "exclusive" signals that the two types of timing anomalies can only occur for different states $b_1, b_2 \in B$.

Theorem 5.9 states that timing anomalies of type **TA-P-E** can efficiently be bounded without having to search the whole state space $A \times B$. This is the most generic form of occurrence of parallel timing anomalies that can efficiently be bounded. As Theorem 5.7 states, this is not possible in

Composition Technique	*No TA*	*Timing Anomaly:* TA-P-I	TA-P-A	TA-P-C	TA-P-E
MC	×		×		
DC	×	×			
max(MC,DC)	×	×	×		×
Full State	×	×	×	×	×

Table 1. Applicability of Parallel Composition

a more generic form.

Theorem 5.9: **Composability with Exclusive Parallel Timing Anomalies:** Processor hardware exhibiting parallel timing anomalies of type **TA-P-E** as defined in Definition 5.8 with a state partitioning into two partions A and B, can safely be analyzed by applying both, *Delta-Composition* (Equation 1) and *Max-Composition* (Equation 3) simultaneously as described in Equation 5 (proof given in [17]).

5.4. Summary of Parallel Composition

Table 1 summarizes the situation where Parallel-Composition is safe. Max-Composition (MC) is safe if at most parallel timing anomalies of type **TA-P-A** are present and Delta-Composition (DC) is safe if at most parallel timing anomalies of type **TA-P-I** are present. If **TA-P-I** and **TA-P-A** can both occur, but only for different states $b \in B$ and $b' \in B$ (scenario **TA-P-E**) then the maximum of Max-Composition and Delta-Composition is a safe upper bound of the execution time. But if **TA-P-I** and **TA-P-A** can both occur for the same state $b \in B$ (scenario **TA-P-C**) then there is no efficient method that does not rely on examining the combined state space of A and B. The examination of the combined state space ("Full State") is not a composition method anymore, but is given to show the consequences in case of **TA-P-C**.

6. Implications on WCET Analysis

The potential occurrence of timing anomalies depends on the target hardware as well as on the program code. Thus, one solution to avoid timing anomalies on existing hardware is to rewrite the program such that no timing anomalies can occur [18], [19]. So far, only timing anomalies for series composition have been addressed. Avoiding parallel timing anomalies is open research.

The other way to avoid timing anomalies is to design predictable hardware that avoids timing anomalies by design. Besides the impossibility result, this paper also provides special cases of parallel timing anomalies where WCET analysis with parallel composition is safe. The latter provide important hints to hardware designers on their way to constructing predictable hardware components.

7. Conclusion and Outlook

The most challenging problem of WCET analysis is the high complexity of today's processors. Features like caches and pipelines create a huge state space. Even worse, effects like *timing anomalies* can make it impossible to construct an efficient *processor behavior analysis* that does not need to search the whole state space for the whole program at once.

In this paper we presented a new class of timing anomalies, which we call *parallel timing anomalies*. Parallel timing anomalies can occur on *parallel composition*, i.e., when analyzing the processor behavior in multiple phases based on a decomposition of the computer state (TRDCS). We have introduced the two fundamental techniques to perform parallel composition: Delta-Composition and Max-Composition. As an impossibility result we have proved that in case of arbitrary forms of parallel timing anomalies it is not possible to exclude underestimation of the WCET with parallel composition. Additionally, we have shown that parallel compostion provides safe WCET bounds for all types of timing anomalies except **TA-P-C**. These results provide the foundation for a useful tradeoff between flexibility and predictability on processor hardware design.

Future work is needed on identifying the concrete types of timing anomalies that might occur for a concrete processor implementation.

Acknowledgments We would like to thank Michael Zolda for useful advice on first-order logic.

References

[1] R. Kirner and P. Puschner, "Classification of WCET analysis techniques," in *Proc. 8th IEEE International Symposium on Object-oriented Real-time distributed Computing*, Seattle, WA, May 2005, pp. 190–199.

[2] R. Wilhelm, J. Engblom, A. Ermedahl, N. Holsti, S. Thesing, D. Whalley, G. Bernat, C. Ferdinand, R. Heckman, T. Mitra, F. Mueller, I. Puaut, P. Puschner, J. Staschulat, and P. Stenstrom, "The worst-case execution time problem - overview of methods and survey of tools," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 7, no. 3, Apr. 2008.

[3] T. Lundqvist and P. Stenström, "Timing analysis in dynamically scheduled microprocessors," in *Proc. 20th IEEE Real-Time Systems Symposium (RTSS)*, Dec. 1999, pp. 12–21.

[4] I. Wenzel, R. Kirner, P. Puschner, and B. Rieder, "Principles of timing anomalies in superscalar processors," in *Proc. 5th International Conference of Quality Software*, Melbourne, Australia, Sep. 2005.

[5] J. Reineke, B. Wachter, S. Tesing, R. Wilhelm, I. Polian, J. Eisinger, and B. Becker, "A definition and classification of timing anomalies," in *Proc. 6th International Workshop on Worst-Case Execution Time Analysis*, Dresden, Germany, July 2006.

[6] Y.-T. S. Li, S. Malik, and A. Wolfe, "Efficient microarchitecture modeling and path analysis for real-time software," in *Proc. IEEE Real-Time Systems Symposium*, Dec. 1995, pp. 298–307.

[7] P. Puschner and A. V. Schedl, "Computing maximum task execution times – a graph-based approach," *Journal of Real-Time Systems*, vol. 13, pp. 67–91, 1997.

[8] F. Mueller, "Timing analysis for instruction caches," *Journal of Real-Time Systems*, vol. 18, no. 2/3, pp. 209–239, May 2000.

[9] C. Ferdinand, R. Heckmann, M. Langenbach, F. Martin, M. Schmidt, H. Theiling, S. Thesing, and R. Wilhelm, "Reliable and precise WCET determination for a real-life processor," in *Proc. of the 1st International Workshop on Embedded Software (EMSOFT 2001)*, Tahoe City, CA, USA, Oct. 2001, pp. 469–485.

[10] R. L. Graham, "Bounds on multiprocessing timing anomalies," *SIAM Journal of Applied Mathematics*, vol. 17, no. 2, pp. 416–429, 1969.

[11] J. Schneider, "Combined schedulability and wcet analysis for real-time operating systems," PhD Thesis, Universität des Saarlandes, Saarbrücken, Germany, Dec. 2002.

[12] C. Berg, "PLRU cache domino effects," in *Proc. 6th International Workshop on Worst-Case Execution Time Analysis*, Dresden, Germany, July 2006.

[13] I. Wenzel, "Principles of timing anomalies in superscalar processors," Master's thesis, Technische Universität Wien, Vienna, Austria, 2003.

[14] S. Thesing, *Safe and Precide WCET Determination by Abstract Interpretation of Pipeline Models*. Pirrot Verlag, July 2004, iSBN: 3-937436-00-6.

[15] J. Eisinger, I. Polian, B. Becker, A. Metzner, S. Thesing, and R. Wilhelm, "Automatic identification of timing anomalies for cycle-accurate worst-case execution time analysis," in *Proc, 9th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems*. IEEE Computer Society, 2006, pp. 15–20.

[16] R. Kirner and P. Puschner, "Obstacles in worst-cases execution time analysis," in *Proc. 11th IEEE International Symposium on Object-oriented Real-time distributed Computing*, Orlando, Florida, May 2008, pp. 333–339.

[17] R. Kirner, A. Kadlec, and P. Puschner, "Worst-case execution time analysis for processors showing timing anomalies," Technische Universität Wien, Institut für Technische Informatik, Treitlstr. 1-3/182-1, 1040 Vienna, Austria, Research Report 01/2009, 2009.

[18] C. Rochange and P. Sainrat, "Code padding to improve the WCET calculability," in *Proc. 14th International Conference on Real-Time and Network Systems (RNTS)*, Poitiers, France, May 2006.

[19] A. Kadlec and R. Kirner, "Neutralizing timing anomalies in complex computer architectures," in *Proc. Junior Scientist Conference*, Vienna, Austria, Nov. 2008, pp. 119–120.