Definitions and Detection Procedures of Timing Anomalies for the Formal Verification of Predictability in Real-Time Systems

CAPITAL Workshop

Benjamin Binder

Mihail Asavoae [1], Belgacem Ben Hedia [1], Florian Brandner [2], and Mathieu Jan [1]

[1] *Université Paris-Saclay, CEA, List*

[2] *Institut Polytechnique de Paris, Télécom Paris, LTCI*

June 13, 2023

Outline

Real-Time (RT) Systems

Timing behavior

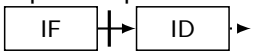
Strong timing requirements
$\longrightarrow$ *temporal correctness*

Formal modeling and verification

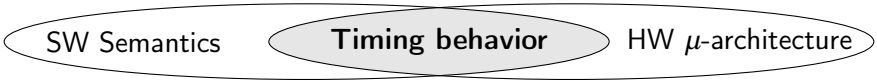
Real-Time (RT) Systems

Strong timing requirements
$\longrightarrow$ *temporal correctness*
Formal modeling and verification

Pipelined processor



Real-Time (RT) Systems



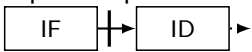

```
LD    r1, 0(r2)   ; A
ADD   r3, r1, r4  ; B
ADD   r5, r6, r7  ; C
LD    r8, (0)r5   ; D
```

Program (Instruction sequence)

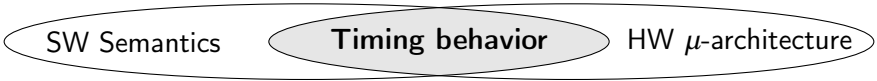
Strong timing requirements
$\longrightarrow$ *temporal correctness*

Formal modeling and verification

Pipelined processor



Real-Time (RT) Systems

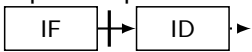



```
LD    r1, 0(r2)   ; A
ADD   r3, r1, r4  ; B
ADD   r5, r6, r7  ; C
LD    r8, (0)r5   ; D
```

Program (Instruction sequence)

Strong timing requirements
$\longrightarrow$ *temporal correctness*
Formal modeling and verification

Pipelined processor

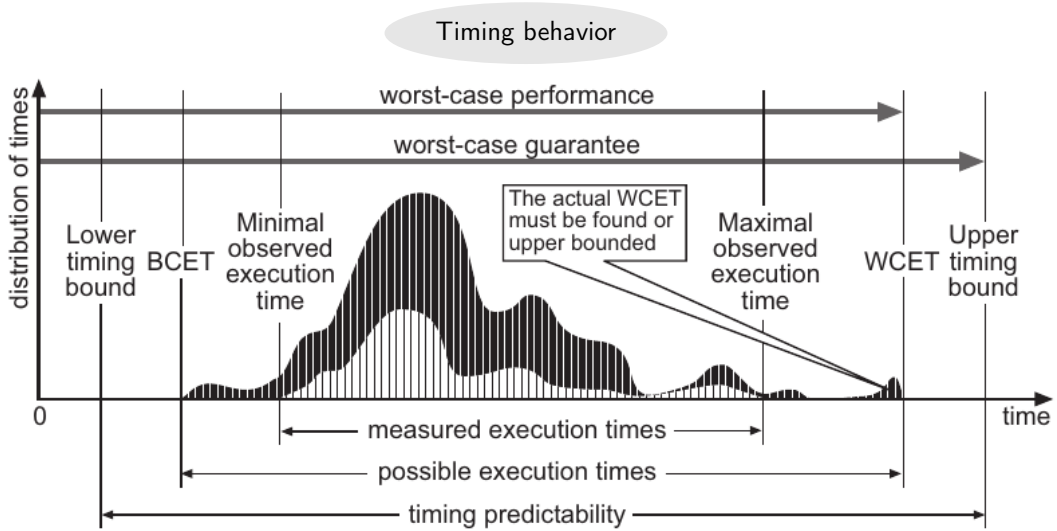


No real-time notion

Unknown input data and HW state

Performance enhancers

Predictability



WCET Analysis

Worst-Case-Execution-Time Analysis:

- Test measurements & Probabilistic analysis
- Static analysis
  ⟶ Microarchitecture (instructions → clock cycles)
- Compositional analysis (e.g., pipeline+cache)

WCET Analysis

Worst-Case-Execution-Time Analysis:

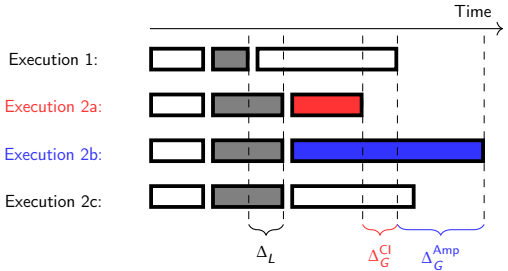
- Test measurements & Probabilistic analysis
- Static analysis
  ⟶ Microarchitecture (instructions → clock cycles)
- Compositional analysis (e.g., pipeline+cache)

No exhaustive exploration, but *estimates*.
⚠ **Timing Anomalies** (TAs) jeopardize all methods.

Timing Anomalies (TAs)

(At least) 2 *execution traces* of the same program trace:

- Same input program and data
- Distinct initial hardware states
- Variation in *latencies*



- *Counter-intuitive* TAs → jeopardize all WCET analyses
- *Amplification* TAs → jeopardize *compositional* timing analyses

Motivation

- COTS (Commercial off-the-shelf) processors are subject to TAs and are more and more used in RT systems.
- Understanding TAs is crucial for the analysis of the timing behavior of RT systems.
- The execution of specific binaries may hide some TAs.

  ⟶ **Need for a reliable code-specific detection of TAs**

Our Approach towards the Detection of TAs

1. How to set up a modeling and verification strategy for a *detection procedure*?
   ⟶ Formal tools for modeling systems and verifying timing properties
   ⟶ Case study: amplification TAs on an industrial in-order processor
2. How to *interpret* TAs wrt. WCET?
   ⟶ Nature of TAs
3. Which criteria for a reliable formal definition of TAs?
   ⟶ Case study: counter-intuitive TAs on an Out-of-Order (OoO) processor

State of the Art

- Formalization of timing anomalies in microarchitectures
- Hardware timing modeling
- Formal methods for hardware verification
- Modeling and verification tools

State of the Art

- Formalization of timing anomalies in microarchitectures
- Hardware timing modeling
- Formal methods for hardware verification
- Modeling and verification tools

Overview of the Existing Definitions & Procedures of TAs

Definitions	Time in FUs	Step heights	Intersections	Occupation	Locality	Analysis	Hardware	Amplifications	Procedure
	Variations					Interp.			
Lundqvist and Stenström	Unspec.						✓	✓	
Wenzel et al.	✓					✓		✓	
Gebhard		✓					✓		
Reineke and Sen		✓				✓			
Kirner et al.			✓				✓	✓	
Cassez et al.			✓				✓		
Eisinger et al.			✓			✓	✓		✓
Kirner et al.				✓			✓	✓	
Reineke et al.					✓	✓			

Various criteria are used for *latencies* and thus *variations*.

Definitions	Time in FUs	Step heights	Intersections	Occupation	Locality	Analysis	Hardware	Amplifications	Procedure
	Variations					Interp.			
Lundqvist and Stenström	Unspec.						✓	✓	
Wenzel et al.	✓					✓		✓	
Gebhard		✓					✓		
Reineke and Sen		✓				✓			
Kirner et al.			✓				✓	✓	
Cassez et al.			✓				✓		
Eisinger et al.			✓			✓	✓		✓
Kirner et al.				✓			✓	✓	
Reineke et al.					✓	✓			

The phenomenon is subjected to *interpretations*.

Definitions	Time in FUs	Step heights	Intersections	Occupation	Locality	Analysis	Hardware	Amplifications	Procedure
	Variations					Interp.			
Lundqvist and Stenström	Unspec.						✓	✓	
Wenzel et al.	✓					✓		✓	
Gebhard		✓					✓		
Reineke and Sen		✓				✓			
Kirner et al.			✓				✓	✓	
Cassez et al.			✓				✓		
Eisinger et al.			✓			✓	✓		✓
Kirner et al.				✓			✓	✓	
Reineke et al.					✓	✓			

Amplification TAs usually defined as a complement.

Definitions	Time in FUs	Step heights	Intersections	Occupation	Locality	Analysis	Hardware	Amplifications	Procedure
	Variations					Interp.			
Lundqvist and Stenström	Unspec.						✓	✓	
Wenzel et al.	✓					✓		✓	
Gebhard		✓					✓		
Reineke and Sen		✓				✓			
Kirner et al.			✓				✓	✓	
Cassez et al.			✓				✓		
Eisinger et al.			✓			✓	✓		✓
Kirner et al.				✓			✓	✓	
Reineke et al.					✓	✓			

Work around TAs widely theoretical and not *executable*

Definitions	Time in FUs	Step heights	Intersections	Occupation	Locality	Analysis	Hardware	Amplifications	Procedure
	Variations					Interp.			
Lundqvist and Stenström	Unspec.						✓	✓	
Wenzel et al.	✓					✓		✓	
Gebhard		✓					✓		
Reineke and Sen		✓				✓			
Kirner et al.			✓				✓	✓	
Cassez et al.			✓				✓		
Eisinger et al.			✓			✓	✓		✓
Kirner et al.				✓			✓	✓	
Reineke et al.					✓	✓			

Contributions

1. Heuristics for the detection of amplification-TA patterns
   - Binder et al., "Scalable Detection of Amplification Timing Anomalies for the Superscalar TriCore Architecture" (FMICS20)
   - Binder et al., "Formal Modeling and Verification for Amplification Timing Anomalies in the Superscalar TriCore Architecture" (STTT21)
2. Unified formal framework for assessing the various definitions
   - Binder et al., "Formal Processor Modeling for Analyzing Safety and Security Properties" (ERTS22)
   - Binder et al., "Is This Still Normal? Putting Definitions of Timing Anomalies to the Test" (RTCSA21)
3. Novel formal definition of counter-intuitive TAs with a detection procedure
   - Binder et al., "The Role of Causality in a Formal Definition of Timing Anomalies" (RTCSA22)

Outline

How to Concretely Apply the Definitions?

- HW models restricted to theoretical concepts
  $\longrightarrow$ Definitions usually not implemented as procedures
- Incomplete definitions per se and illustrated only through partial examples
- Different levels of *granularity* to define **latencies/variations**

Contributions

1. *Systematic* approach, precise assumptions
2. *Formal hardware model* (TLA$^+$)
   - Parameterizable formal model of a representative OoO pipeline template
   - An *executable procedure* in the form of a predicate for each definition
3. *Assessment* of the definitions by model checking (finding contradictions)

Benjamin Binder et al. "Is This Still Normal? Putting Definitions of Timing Anomalies to the Test". In: *IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. 2021, pp. 139–148. doi: 10.1109/RTCSA52859.2021.00024

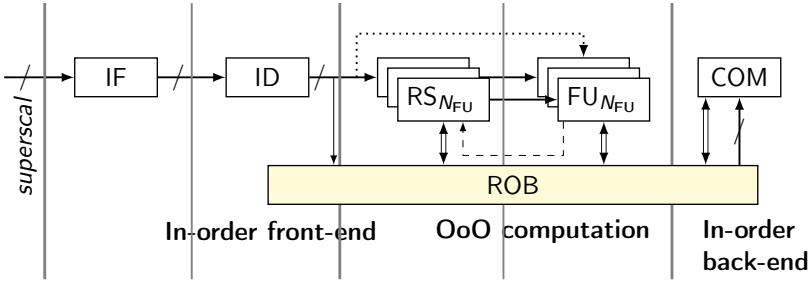
Modeling Requirements

- No functional aspects beyond the pipeline level
  $\longrightarrow$ *cycle-accurate* timing ($\mathbb{N}$)
- Abstract data path: only pipeline stages ($\mathscr{S}$)
- Abstract control path: signals impacting the stalling logic
- Focus on instruction ($\mathscr{I}$) progress (SW) through the pipeline stages (HW)
  Execution: $\mathscr{I} \times \mathbb{N} \rightarrow \mathscr{S}$

Benjamin Binder et al. "Formal Processor Modeling for Analyzing Safety and Security Properties". In: *11th European Congress Embedded Real Time Systems (ERTS)*. 2022

Out-of-Order (OoO) Pipeline Template

A concrete HW model as a simple *unified* framework.

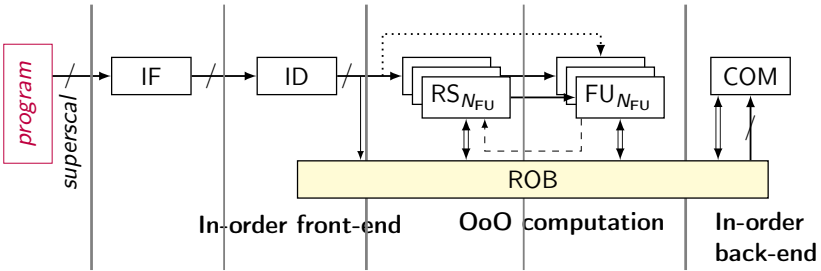


Representative **OoO pipeline template** based on Tomasulo's algorithm with N_{FU} functional units, fetching and committing *superscal* instructions per cycle.

Out-of-Order (OoO) Pipeline Template

A concrete HW model as a simple *unified* framework.

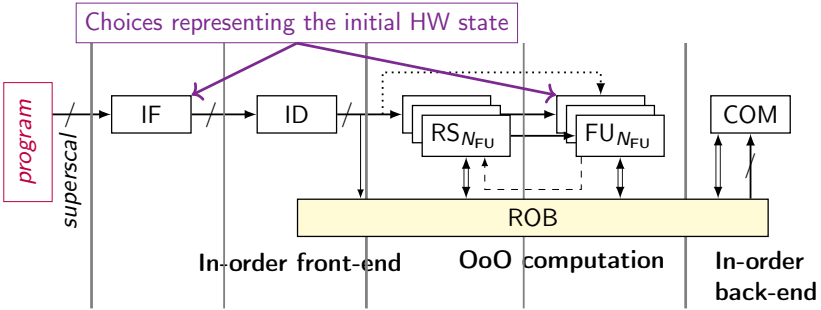
```
LD   r1, 0(r2)  ; A
ADD  r3, r1, r4 ; B
ADD  r5, r6, r7 ; C
LD   r8, (0)r5  ; D
```



Out-of-Order (OoO) Pipeline Template

A concrete HW model as a simple *unified* framework.

```
LD   r1, 0(r2)   ; A
ADD  r3, r1, r4  ; B
ADD  r5, r6, r7  ; C
LD   r8, (0)r5   ; D
```



A TA on the OoO Pipeline Template

A concrete example of a TA:

α	1	2	3	4	5	6	7	8	9	10	11	12	13
A	IF	ID	FU_1	COM									
B	IF	ID	RS_2	FU_2	FU_2	FU_2	COM						
C		IF	ID	RS_2	RS_2	RS_2	FU_2	FU_2	FU_2	COM			
D		IF	ID	RS_1	RS_1	RS_1	RS_1	RS_1	RS_1	FU_1	FU_1	FU_1	COM•

A TA on the OoO Pipeline Template

A concrete example of a TA:

		1	2	3	4	5	6	7	8	9	10	11	12	13
α	A	IF	ID	FU_1	COM									
	B	IF	ID	RS_2	FU_2	FU_2	FU_2	COM						
	C		IF	ID	RS_2	RS_2	RS_2	FU_2	FU_2	FU_2	COM			
	D		IF	ID	RS_1	RS_1	RS_1	RS_1	RS_1	RS_1	FU_1	FU_1	FU_1	COM•
β	A	IF	ID	FU_1	FU_1	FU_1	COM							
	B	IF	ID	RS_2	RS_2	RS_2	RS_2	FU_2	FU_2	FU_2	COM			
	C		IF	ID	FU_2	FU_2	FU_2	ROB	ROB	ROB	COM			
	D		IF	ID	RS_1	RS_1	RS_1	FU_1	FU_1	FU_1	ROB	COM•		

What Could be a *Latency*?

- Focus on specific events only:
  - Instruction *commit*
    Kirner, Kadlec, and Puschner, *Worst-Case Execution Time Analysis for Processors showing Timing Anomalies*
    Eisinger et al., "Automatic Identification of Timing Anomalies for Cycle-Accurate Worst-Case Execution Time Analysis"
    Gebhard, "Timing Anomalies Reloaded"
    Cassez, Hansen, and Olesen, "What is a Timing Anomaly?"
  - Instruction *fetch*
    Reineke and Sen, "Sound and Efficient WCET Analysis in the Presence of Timing Anomalies"
- Finer-grained comparison of the utilization of *local* resources
  Reineke et al., "A Definition and Classification of Timing Anomalies"
- Other viewpoint: not instruction latencies but a *total* resource use
  Kirner, Kadlec, and Puschner, "Precise Worst-Case Execution Time Analysis for Processors with Timing Anomalies"

Application of Various Formal Definitions (on the 2 traces)

		1	2	3	4	5	6	7	8	9	10	11	12	13
α	A	IF	ID	FU_1	COM									
	B	IF	ID	RS_2	FU_2	FU_2	FU_2	COM						
	C		IF	ID	RS_2	RS_2	RS_2	FU_2	FU_2	FU_2	COM			
	D		IF	ID	RS_1	RS_1	RS_1	RS_1	RS_1	RS_1	FU_1	FU_1	FU_1	COM•
β	A	IF	ID	FU_1	FU_1	FU_1	COM							
	B	IF	ID	RS_2	RS_2	RS_2	RS_2	FU_2	FU_2	FU_2	COM			
	C		IF	ID	FU_2	FU_2	FU_2	ROB	ROB	ROB	COM			
	D		IF	ID	RS_1	RS_1	RS_1	FU_1	FU_1	FU_1	ROB	COM•		

Figure: Execution traces

[1] Reineke et al., "A Definition and Classification of Timing Anomalies"

Application of Various Formal Definitions (on the 2 traces)

Commit events only[1]

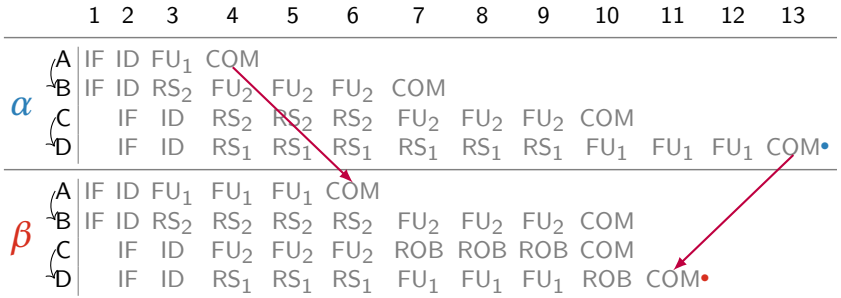


Figure: Execution traces showing the order of commits (→)

[1] Kirner, Kadlec, and Puschner, *Worst-Case Execution Time Analysis for Processors showing Timing Anomalies*; Cassez, Hansen, and Olesen, "What is a Timing Anomaly?"

Application of Various Formal Definitions (on the 2 traces)

Locality + Commit events [1]

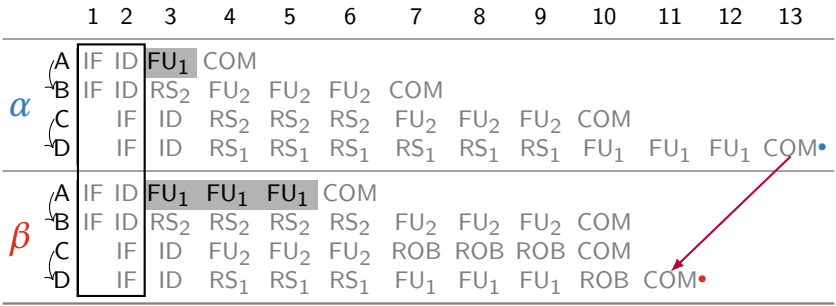


Figure: Execution traces showing the order of commits (→) and the common prefix (□).

[1] Reineke et al., "A Definition and Classification of Timing Anomalies"

Properties for the Absence of TAs

- 2 instances of the pipeline specification (hyperproperty → self-composition)
  ⟶ 2 traces (same program but different *actual choices*)
- Additional operators and state variables
- Example: definition based on commit events
  ⟶ Keep track of the instant of each commit event (*comTime*)

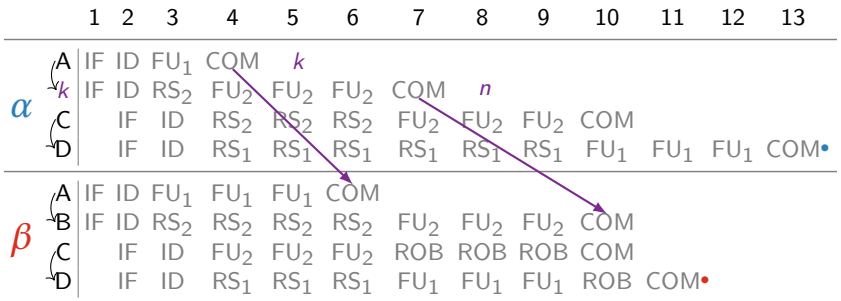
Commit Events: Invariant for the absence of TAs

$$
\begin{aligned}
NoTA \triangleq\ & \forall\, k \in 1\,..\,Len(Program) - 1 : \forall\, n \in k+1\,..\,Len(Program) : \\
& \wedge\ ProgDone(n) \\
& \wedge\ ComTime(1,k) < ComTime(2,k) \\
& \Longrightarrow\ ComTime(1,n) \leq ComTime(2,n)
\end{aligned}
$$

		1	2	3	4	5	6	7	8	9	10	11	12	13
α	A	IF	ID	FU_1	COM									
	A	IF	ID	RS_2	FU_2	FU_2	FU_2	COM						
	C		IF	ID	RS_2	RS_2	RS_2	FU_2	FU_2	FU_2	COM			
	D		IF	ID	RS_1	RS_1	RS_1	RS_1	RS_1	RS_1	FU_1	FU_1	FU_1	COM•
β	A	IF	ID	FU_1	FU_1	FU_1	COM							
	B	IF	ID	RS_2	RS_2	RS_2	RS_2	FU_2	FU_2	FU_2	COM			
	C		IF	ID	FU_2	FU_2	FU_2	ROB	ROB	ROB	COM			
	D		IF	ID	RS_1	RS_1	RS_1	FU_1	FU_1	FU_1	ROB	COM•		

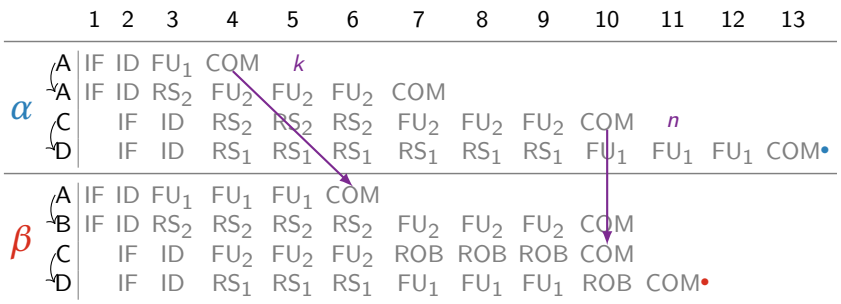
Commit Events: Invariant for the absence of TAs

$$
\begin{aligned}
NoTA \triangleq\ & \forall\, k \in 1\,..\,Len(Program) - 1 : \forall\, n \in k+1\,..\,Len(Program) : \\
& \wedge\ ProgDone(n) \\
& \wedge\ ComTime(1, k) < ComTime(2, k) \\
& \Longrightarrow\ ComTime(1, n) \leq ComTime(2, n)
\end{aligned}
$$



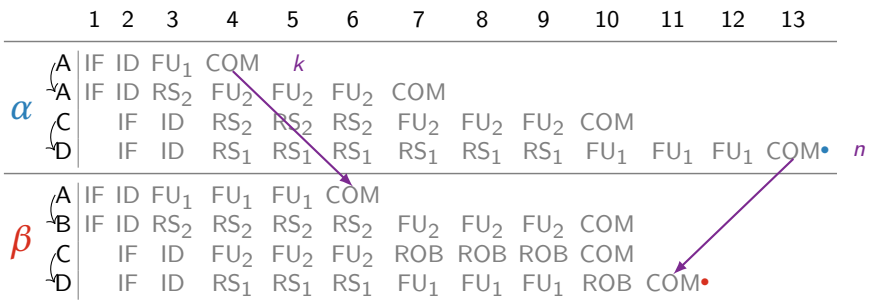
Commit Events: Invariant for the absence of TAs

$$
\begin{aligned}
NoTA \triangleq\ & \forall\, k \in 1\,..\,Len(Program) - 1 : \forall\, n \in k+1\,..\,Len(Program) : \\
& \wedge\ ProgDone(n) \\
& \wedge\ ComTime(1, k) < ComTime(2, k) \\
& \implies ComTime(1, n) \leq ComTime(2, n)
\end{aligned}
$$



Commit Events: Invariant for the absence of TAs

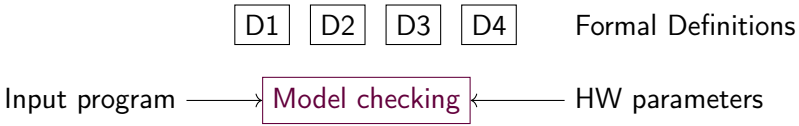
$$
\begin{aligned}
NoTA \triangleq\ & \forall\, k \in 1\,..\,Len(Program) - 1 : \forall\, n \in k+1\,..\,Len(Program) : \\
& \wedge\ ProgDone(n) \\
& \wedge\ ComTime(1, k) < ComTime(2, k) \\
& \implies ComTime(1, n) \leq ComTime(2, n)
\end{aligned}
$$



Assessment Methodology

- Model checking: looking for *inconsistent scenarios*
- Based on variations of an input program found in the literature

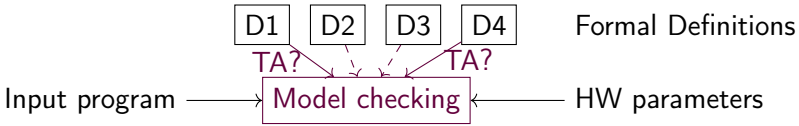
Counterexamples (CEX) derived from different invariants



Assessment Methodology

- Model checking: looking for *inconsistent scenarios*
- Based on variations of an input program found in the literature

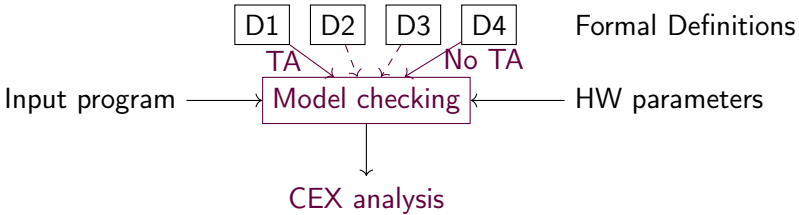
Counterexamples (CEX) derived from different invariants



Assessment Methodology

- Model checking: looking for *inconsistent scenarios*
- Based on variations of an input program found in the literature

Counterexamples (CEX) derived from different invariants

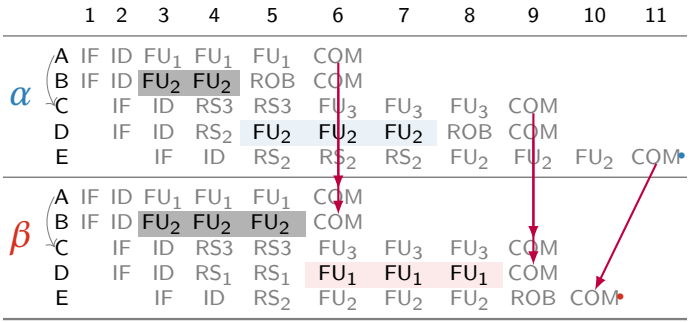


Contradictory Statements about TAs due to Different Granularities

		1	2	3	4	5	6	7	8	9	10	11
α	A	IF	ID	FU_1	FU_1	FU_1	COM					
	B	IF	ID	FU_2	FU_2	ROB	COM					
	C		IF	ID	RS3	RS3	FU_3	FU_3	FU_3	COM		
	D		IF	ID	RS_2	FU_2	FU_2	FU_2	ROB	COM		
	E			IF	ID	RS_2	RS_2	RS_2	FU_2	FU_2	FU_2	COM
β	A	IF	ID	FU_1	FU_1	FU_1	COM					
	B	IF	ID	FU_2	FU_2	FU_2	COM					
	C		IF	ID	RS3	RS3	FU_3	FU_3	FU_3	COM		
	D		IF	ID	RS_1	RS_1	FU_1	FU_1	FU_1	COM		
	E			IF	ID	RS_2	FU_2	FU_2	FU_2	ROB	COM	

Contradictory Statements about TAs due to Different Granularities

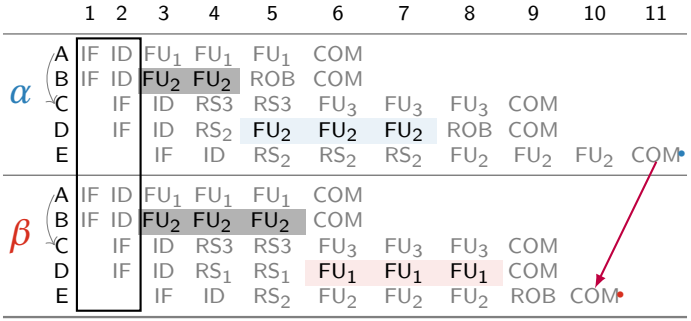
In particular the definitions based on instruction latencies vs. locality:



The formal definition based on commit events only (→) **does not** signal a TA

Contradictory Statements about TAs due to Different Granularities

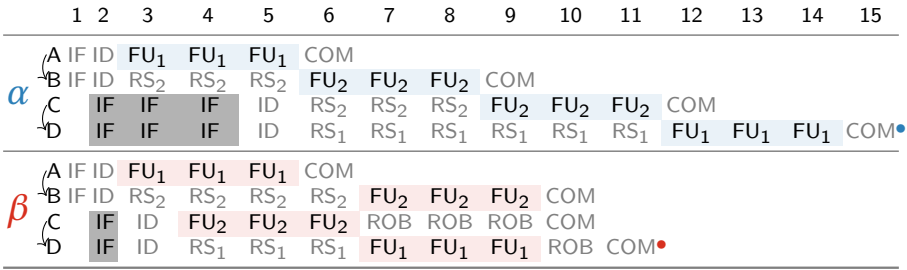
In particular the definitions based on instruction latencies vs. locality:



The formal definition based on commit events only (→) **does not** signal a TA, whereas the definition based on locality (□) **does**.

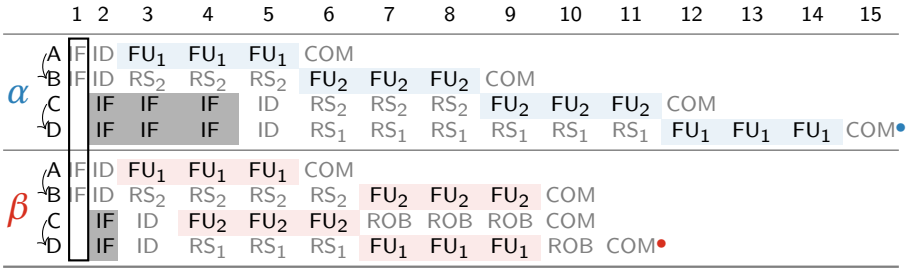
Commit Events Show a Particular Effect

Same TA pattern:



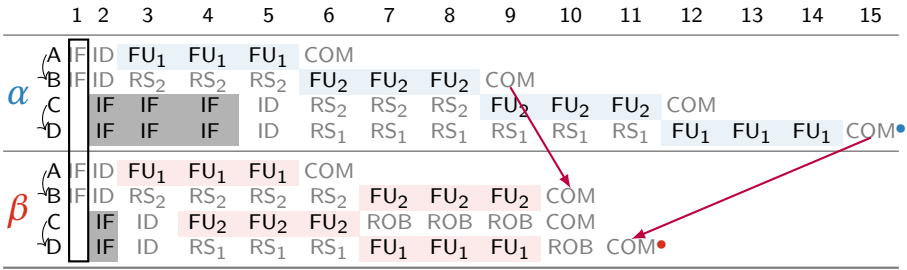
Commit Events Show a Particular Effect

Same TA pattern:



Commit Events Show a Particular Effect

Same TA pattern:



The TA is due to an effect on previous instructions.
The definitions that signal a TA **do not correctly capture** it.

No Definition is Reliable on an OoO Pipeline

- Contradictory statements about TAs
- **No definition dominates others**
- It is clear what a TA is *not* (intuitive understanding)

⟶ **No definition accurately characterizes TAs**.

No Definition is Reliable on an OoO Pipeline

- Contradictory statements about TAs
- **No definition dominates others**
- It is clear what a TA is *not* (intuitive understanding)

⟶ **No definition accurately characterizes TAs**.

- No restrictions on the applicability conditions
- Standard microarchitecture & simple examples
- Main deficiency: the notion of **causality**

No Definition is Reliable on an OoO Pipeline

- Contradictory statements about TAs
- **No definition dominates others**
- It is clear what a TA is *not* (intuitive understanding)

$\longrightarrow$ **No definition accurately characterizes TAs**.

- No restrictions on the applicability conditions
- Standard microarchitecture & simple examples
- Main deficiency: the notion of **causality**

$\Longrightarrow$ A precise formal definition was still needed.

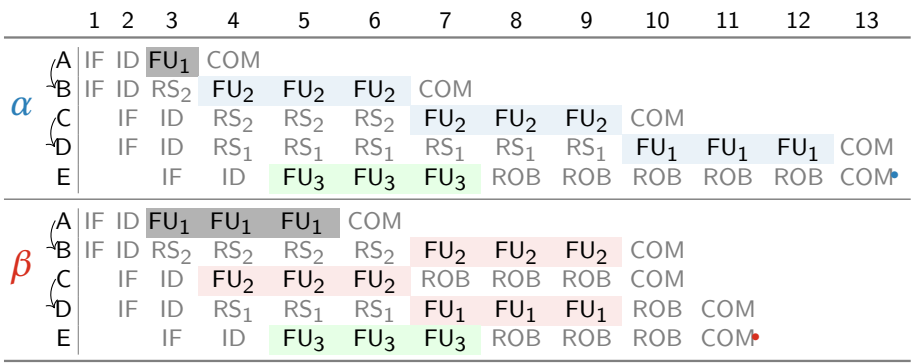
Outline

Requirements of Our Definition

- Independent of any WCET-analysis technique
- Integration of **causality**
- Generic notions
- Instantiation for a *well-specified* HW model
  → Clear assumptions for a **detection procedure** (OoO pipeline)

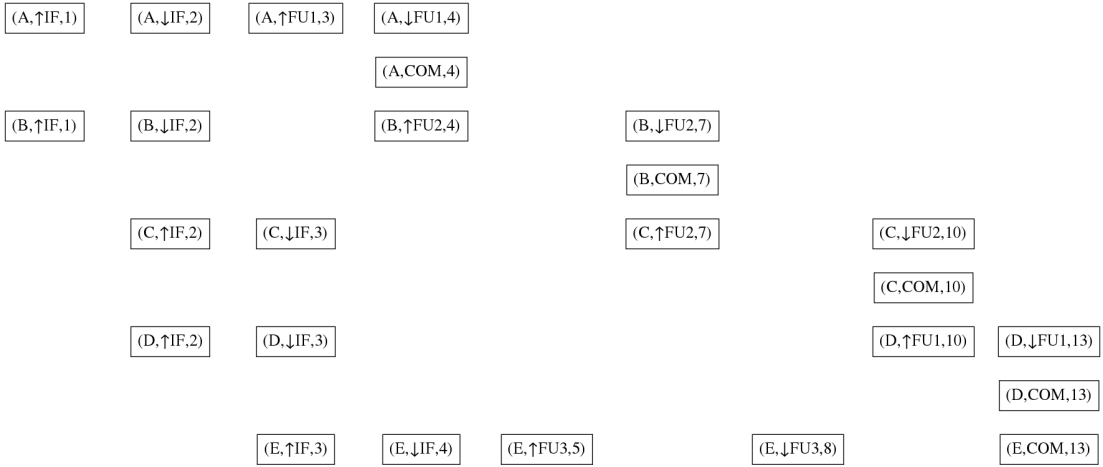
Benjamin Binder et al. "The Role of Causality in a Formal Definition of Timing Anomalies". In: *IEEE 28th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. 2022, pp. 91–102. doi: 10.1109/RTCSA55878.2022.00016

Basic Established TA Pattern



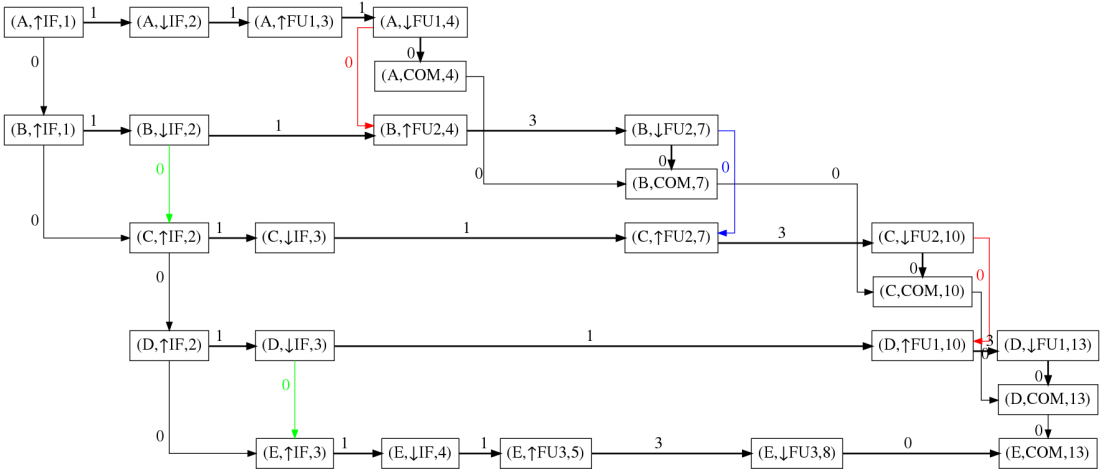
Nodes

1 – Events emerging from trace α



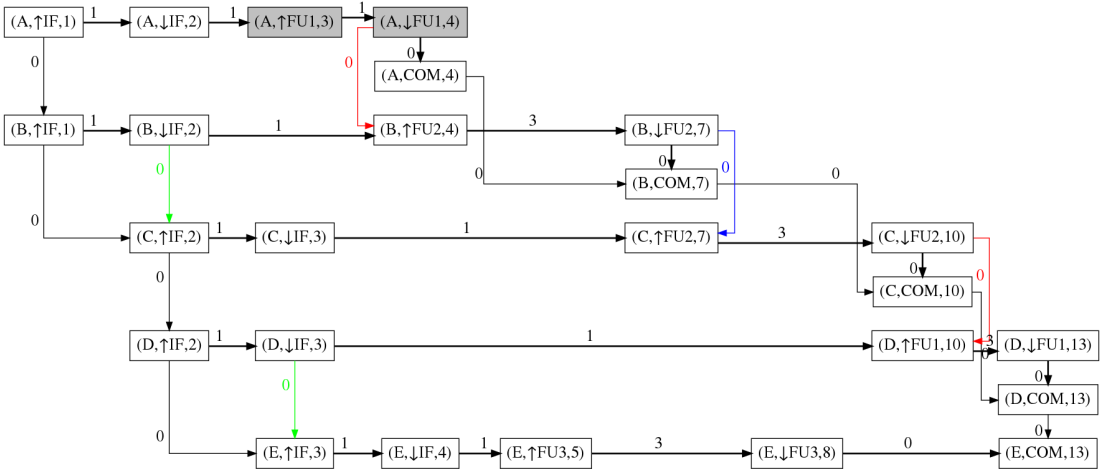
Arcs of the Event Time-Dependence Graph

2 – Minimal durations (imposed by the hardware)



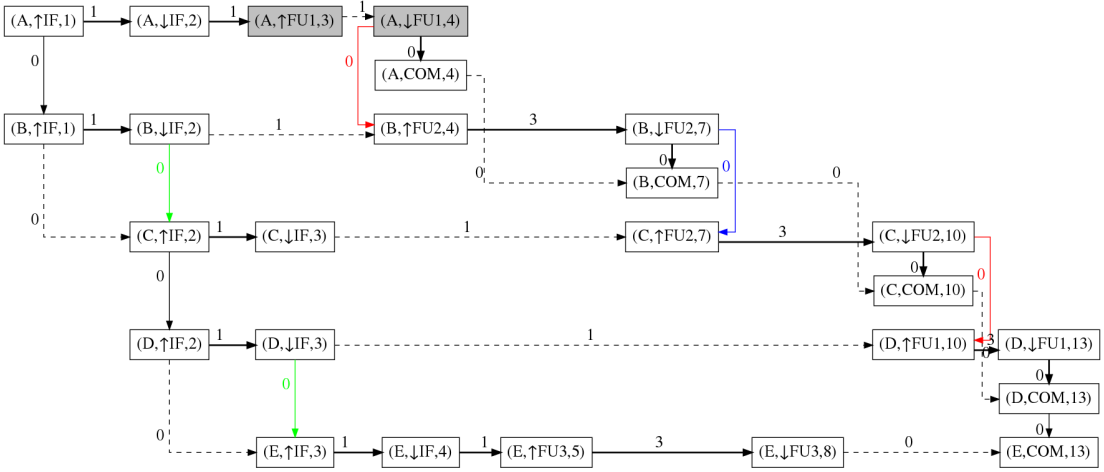
Identification of Latencies

3 – 1 favorable variation ( )



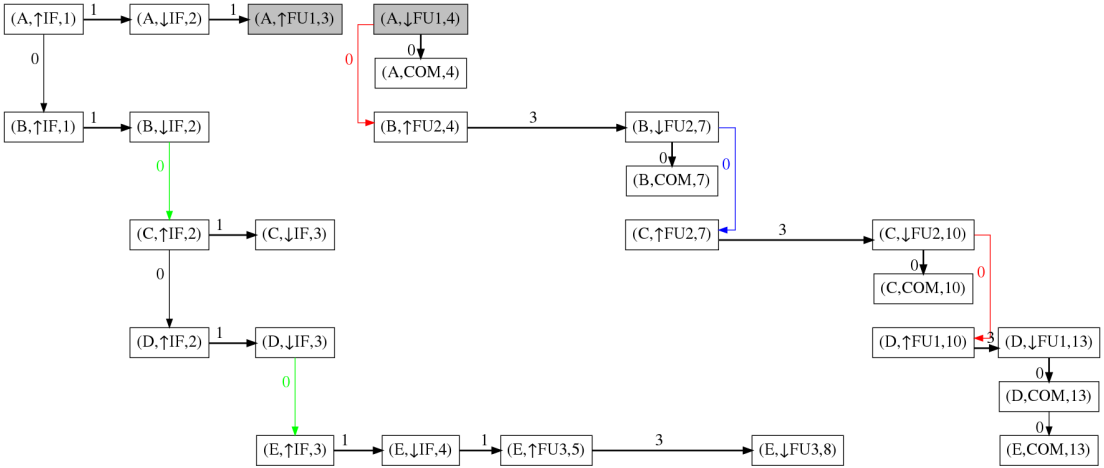
Causality Graph

4 – Only the arcs that *determine* the instants of the destination events



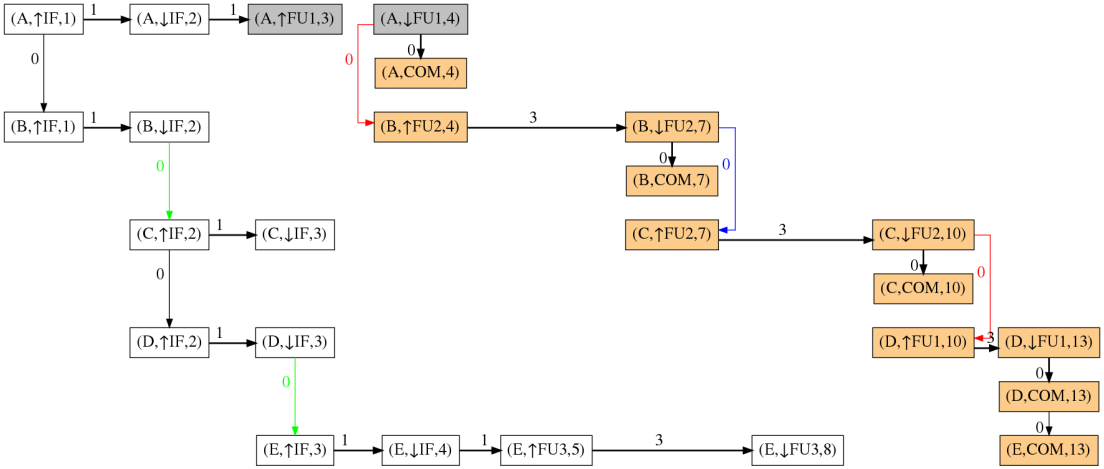
Causality Graph

4 – Only the arcs that *determine* the instants of the destination events

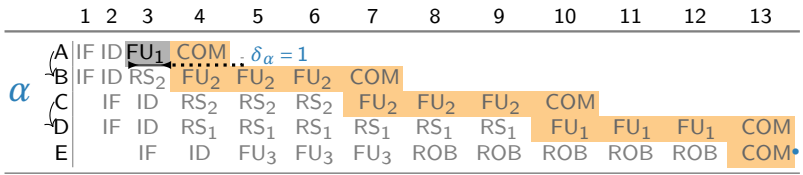


Causality Graph

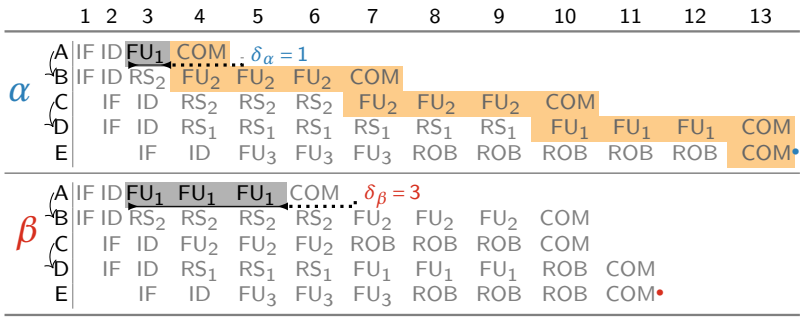
5 – Causal region ( ) of the favorable variation



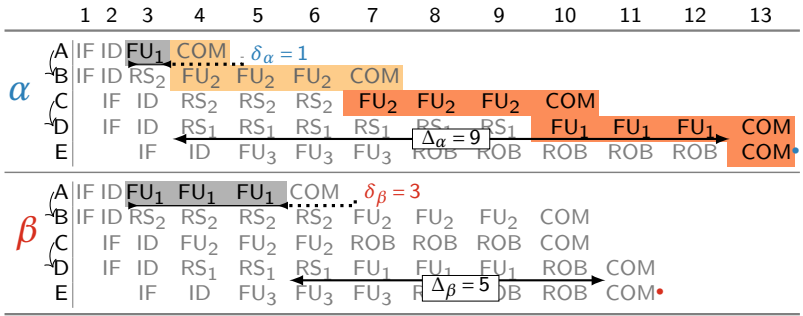
Final Verdict



Final Verdict



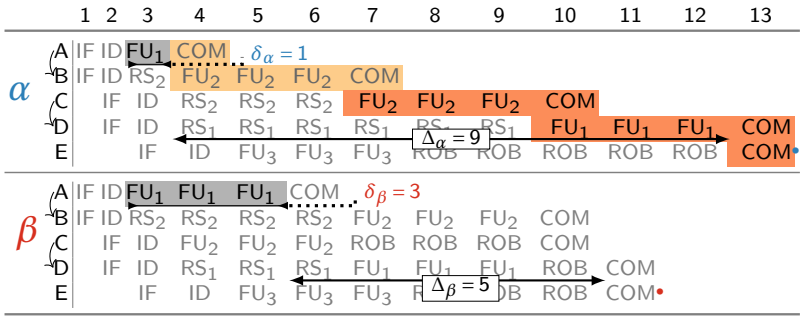
Final Verdict



6 – Identification of TAs (■)

- comparison of the *relative* time distances (Δ)

Final Verdict



Conforming with the intuitive definition, with:

- precisely defined *variations*;
- causal region limiting the *scope* of the verdict;
- comparison of the *relative* time distances (Δ)

Formalization of the Definition

For $\tau = \alpha$ or $\tau = \beta$, let $e_{\tau\uparrow} = (i, \uparrow r_\tau, t_{\tau\uparrow})$ be an acquisition event and $e_{\tau\downarrow} = (i, \downarrow r_\tau, t_{\tau\downarrow})$ be the matching release event, s.t. $e_{\beta\uparrow} = \mathit{CospEvent}(e_{\alpha\uparrow})$ and $e_{\beta\downarrow} = \mathit{CospEvent}(e_{\alpha\downarrow})$.

Definition (Counter-Intuitive Timing Anomaly)

Event $e_{\alpha\downarrow}$ triggers a *counter-intuitive* TA at event e wrt. β, iff:

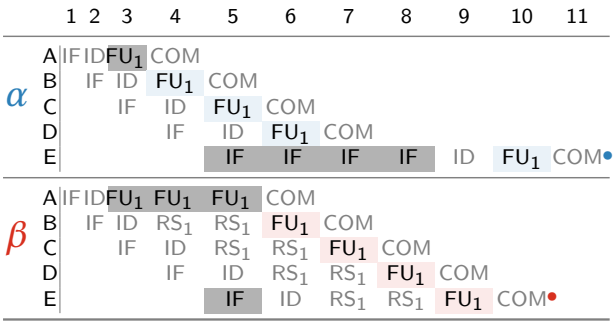
1. **Variation:** α exhibits a *favorable variation* at $e_{\alpha\downarrow}$, i.e.:
$$(\delta_\alpha = t_{\alpha\downarrow} - t_{\alpha\uparrow}) < (t_{\beta\downarrow} - t_{\beta\uparrow} = \delta_\beta)$$
2. **Causality:** e is a node of the *causal region* $C(e_{\alpha\downarrow})$ of the variation:
$$e \in \mathcal{N}_{C(e_{\alpha\downarrow})}$$
3. **Slowdown:** α exhibits a *relative* slowdown, expressed as:
$$\Delta(e_{\alpha\downarrow}, e) > \Delta(e_{\beta\downarrow}, \mathit{CospEvent}(e))$$

$\longrightarrow$ All definitions are introduced formally.

Features of the Procedure

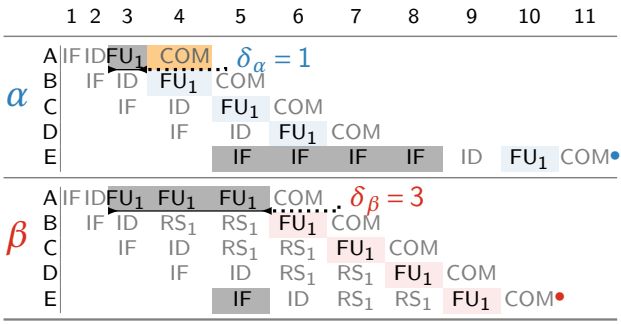
- Procedure integrated into TLA$^+$'s model checker
- Faithfully represents the established TA *patterns*, excluding false positives.
- Open problem: *composition* of variations and timing effects.

Illustration of the Separation of Unrelated Variations



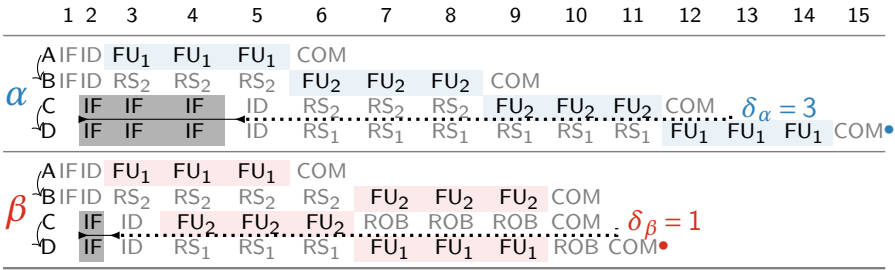
- Totally independent variations, same scheduling on FU_1
- All existing definitions surprisingly state a TA

Illustration of the Separation of Unrelated Variations



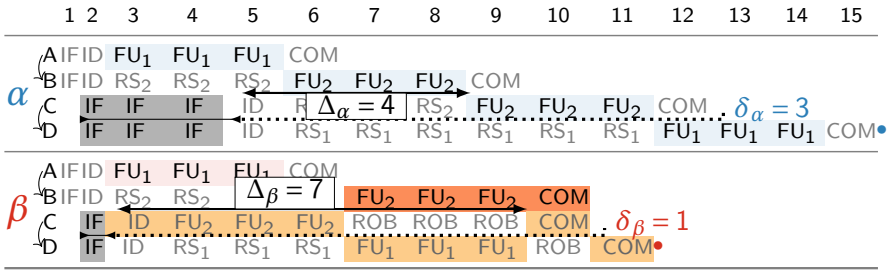
- Totally independent variations, same scheduling on FU_1
- All existing definitions surprisingly state a TA
- Our definition splits the traces into independent parts, hence no TA.

TAs May Be Limited in Scope



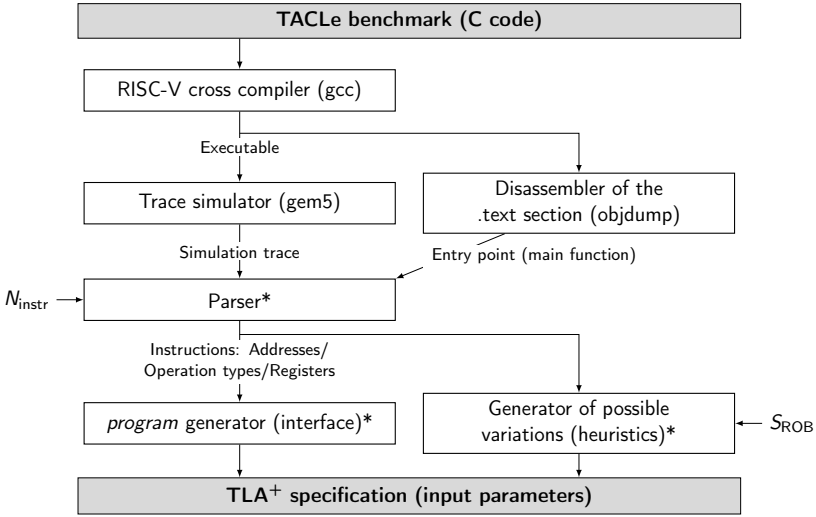
- Existing definitions: no TA or instruction B incriminated

TAs May Be Limited in Scope



- Existing definitions: no TA or instruction B incriminated
- We capture the resource contention in FU_2 in β that blocks B *due to* the variation in C.
- We also capture that the TA propagates in a *limited scope*.

Application on TACLe Benchmarks: Workflow



Synthesis of Some Results

#	benchmark	N_{instr}	$\lvert mayDMiss \rvert$	*superscal*	asym	result	time	diam.	states
5	countneg	50	4	4	✓	true	00:01:24	59	2105
6				2		simple cex	00:00:12	43	728
9	iir	100	4	4	✓	simple cex	00:03:10	100	4749
10	cosf	30	7	4	✓	true	00:23:24	86	118880
11				2		true	00:22:50	86	118991
12	fft	100	3	4	✓	true	00:00:26	98	834
13	fir2dim	100	4	4	✓	simple cex	00:05:42	81	1800
14	insertsort	30	6	4	✓	cex$^{\circ}$	00:00:31	29	3333
15					*	true	00:00:15	71	1159
16				2	✓	cex$^{\circ\circ}$	00:00:26	30	3632
17					*	simple cex	00:00:17	37	456
18	complexup	100	4	4	✓	simple cex	00:02:17	79	2563
19	bitonic	100	30	4	✓	simple cex	00:05:21	35	18008

$^{\circ}$: composition of variations $^{\circ\circ}$: + no COM event for TAs

asym: $IF \wedge NxtFU(i).ind \in mayDMiss$

(✓) $\wedge exec\_inst = 2 \implies NxtFU(i).ind \neq Min(mayDMiss)$

(*) $\wedge exec\_inst = 2$

Synthesis of Some Results

#	benchmark	N_{instr}	$\lvert mayDMiss \rvert$	*superscal*	asym	result	time	diam.	states
5	countneg	50	4	4	✓	true	00:01:24	59	2105
6				2		simple cex	00:00:12	43	728
9	iir	100	4	4	✓	simple cex	00:03:10	100	4749
10	cosf	30	7	4	✓	true	00:23:24	86	118880
11				2		true	00:22:50	86	118991
12	fft	100	3	4	✓	true	00:00:26	98	834
13	fir2dim	100	4	4	✓	simple cex	00:05:42	81	1800
14	insertsort	30	6	4	✓	cex$^{\circ}$	00:00:31	29	3333
15					*	true	00:00:15	71	1159
16				2	✓	cex$^{\circ\circ}$	00:00:26	30	3632
17					*	simple cex	00:00:17	37	456
18	complexup	100	4	4	✓	simple cex	00:02:17	79	2563
19	bitonic	100	30	4	✓	simple cex	00:05:21	35	18008

$^{\circ}$: composition of variations $^{\circ\circ}$: + no COM event for TAs

asym: $IF \wedge NxtFU(i).ind \in mayDMiss$

(✓) $\wedge exec\_inst = 2 \implies NxtFU(i).ind \neq Min(mayDMiss)$

(*) $\wedge exec\_inst = 2$

Synthesis of Some Results

#	benchmark	N_{instr}	$\lvert mayDMiss \rvert$	*superscal*	asym	result	time	diam.	states
5	countneg	50	4	4	✓	true	00:01:24	59	2105
6				2		simple cex	00:00:12	43	728
9	iir	100	4	4	✓	simple cex	00:03:10	100	4749
10	cosf	30	7	4	✓	true	00:23:24	86	118880
11				2		true	00:22:50	86	118991
12	fft	100	3	4	✓	true	00:00:26	98	834
13	fir2dim	100	4	4	✓	simple cex	00:05:42	81	1800
14	insertsort	30	6	4	✓	cex$^{\circ}$	00:00:31	29	3333
15					*	true	00:00:15	71	1159
16				2	✓	cex$^{\circ\circ}$	00:00:26	30	3632
17					*	simple cex	00:00:17	37	456
18	complexup	100	4	4	✓	simple cex	00:02:17	79	2563
19	bitonic	100	30	4	✓	simple cex	00:05:21	35	18008

$^{\circ}$: composition of variations $^{\circ\circ}$: + no COM event for TAs

asym: $IF \wedge NxtFU(i).ind \in mayDMiss$

(✓) $\wedge exec\_inst = 2 \implies NxtFU(i).ind \neq Min(mayDMiss)$

(*) $\wedge exec\_inst = 2$

Synthesis of Some Results

#	benchmark	N_{instr}	$\lvert mayDMiss \rvert$	*superscal*	asym	result	time	diam.	states
5	countneg	50	4	4	✓	true	00:01:24	59	2105
6				2		simple cex	00:00:12	43	728
9	iir	100	4	4	✓	simple cex	00:03:10	100	4749
10	cosf	30	7	4	✓	true	00:23:24	86	118880
11				2		true	00:22:50	86	118991
12	fft	100	3	4	✓	true	00:00:26	98	834
13	fir2dim	100	4	4	✓	simple cex	00:05:42	81	1800
14	insertsort	30	6	4	✓	cex$^{\circ}$	00:00:31	29	3333
15					*	true	00:00:15	71	1159
16				2	✓	cex$^{\circ\circ}$	00:00:26	30	3632
17					*	simple cex	00:00:17	37	456
18	complexup	100	4	4	✓	simple cex	00:02:17	79	2563
19	bitonic	100	30	4	✓	simple cex	00:05:21	35	18008

$^{\circ}$: composition of variations $^{\circ\circ}$: + no COM event for TAs

asym: $IF \wedge NxtFU(i).ind \in mayDMiss$

(✓) $\wedge exec\_inst = 2 \implies NxtFU(i).ind \neq Min(mayDMiss)$

(*) $\wedge exec\_inst = 2$

Synthesis of Some Results

#	benchmark	N_{instr}	$\lvert mayDMiss \rvert$	*superscal*	asym	result	time	diam.	states
5	countneg	50	4	4	✓	true	00:01:24	59	2105
6				2		simple cex	00:00:12	43	728
9	iir	100	4	4	✓	simple cex	00:03:10	100	4749
10	cosf	30	7	4	✓	true	00:23:24	86	118880
11				2		true	00:22:50	86	118991
12	fft	100	3	4	✓	true	00:00:26	98	834
13	fir2dim	100	4	4	✓	simple cex	00:05:42	81	1800
14	insertsort	30	6	4	✓	cex$^{\circ}$	00:00:31	29	3333
15					*	true	00:00:15	71	1159
16				2	✓	cex$^{\circ\circ}$	00:00:26	30	3632
17					*	simple cex	00:00:17	37	456
18	complexup	100	4	4	✓	simple cex	00:02:17	79	2563
19	bitonic	100	30	4	✓	simple cex	00:05:21	35	18008

$^{\circ}$: composition of variations $^{\circ\circ}$: + no COM event for TAs

asym: $IF \wedge NxtFU(i).ind \in mayDMiss$

(✓) $\wedge exec\_inst = 2 \implies NxtFU(i).ind \neq Min(mayDMiss)$

(*) $\wedge exec\_inst = 2$

Outline

Recap

- TAs = execution phenomena that jeopardize predictability
  - Counter-intuitive TAs
  - Amplification TAs
- Lack of *tool support* for their detection and their understanding

Recap

- TAs = execution phenomena that jeopardize predictability
  - Counter-intuitive TAs
  - Amplification TAs
- Lack of *tool support* for their detection and their understanding
- Limitations of the existing formal definitions of counter-intuitive TAs
  → Notion of *causality*
- Novel definition & detection procedure

Recap

- TAs = execution phenomena that jeopardize predictability
  - Counter-intuitive TAs
  - Amplification TAs
- Lack of *tool support* for their detection and their understanding
- Limitations of the existing formal definitions of counter-intuitive TAs
  → Notion of *causality*
- Novel definition & detection procedure
- Extension of the procedure based on a precond. for amplification TAs
  → Appropriate state-space *reductions*
- Strategies for multiple execution scenarios that help building culprit SW patterns

Future Work

- Improvement of the detection procedure (counter-intuitive TAs)
  - Speedup based on causality (on the fly)
  - Further benchmarking
  - Refinement of the model (composition) and the procedure:
    more concrete scheduler, side effects on the HW states, other HW resources (e.g., speculation)

Future Work

- Improvement of the detection procedure (counter-intuitive TAs)
  - Speedup based on causality (on the fly)
  - Further benchmarking
  - Refinement of the model (composition) and the procedure:
    more concrete scheduler, side effects on the HW states, other HW resources (e.g., speculation)
- Formal definition of amplification TAs → Causality

Future Work

- Improvement of the detection procedure (counter-intuitive TAs)
  - Speedup based on causality (on the fly)
  - Further benchmarking
  - Refinement of the model (composition) and the procedure:
    more concrete scheduler, side effects on the HW states, other HW resources (e.g., speculation)
- Formal definition of amplification TAs → Causality
- Utilization of the heuristics for SW patterns
  → Efficient counter-measures preserving convenient static analyses
  → Integration into a WCET analyzer

Thank you!

Definitions and Detection Procedures of Timing Anomalies for the Formal Verification of Predictability in Real-Time Systems

CAPITAL Workshop

Benjamin Binder

Mihail Asavoae [1], Belgacem Ben Hedia [1], Florian Brandner [2], and Mathieu Jan [1]

[1] *Université Paris-Saclay, CEA, List*

[2] *Institut Polytechnique de Paris, Télécom Paris, LTCI*

June 13, 2023

References I

[1] Benjamin Binder et al. "Formal Modeling and Verification for Amplification Timing Anomalies in the Superscalar TriCore Architecture". In: *International Journal on Software Tools for Technology Transfer (STTT)* 24 (2022), pp. 415–440. issn: 1433-2787. doi: 10.1007/s10009-022-00655-1.

[2] Benjamin Binder et al. "Formal Processor Modeling for Analyzing Safety and Security Properties". In: *11th European Congress Embedded Real Time Systems (ERTS)*. 2022.

[3] Benjamin Binder et al. "Is This Still Normal? Putting Definitions of Timing Anomalies to the Test". In: *IEEE 27th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. 2021, pp. 139–148. doi: 10.1109/RTCSA52859.2021.00024.

References II

[4] Benjamin Binder et al. "Scalable Detection of Amplification Timing Anomalies for the Superscalar TriCore Architecture". In: *Formal Methods for Industrial Critical Systems - 25th International Conference, FMICS 2020, Vienna, Austria, September 2-3, 2020, Proceedings*. Vol. 12327. Lecture Notes in Computer Science. Springer, 2020, pp. 151–169. doi: 10.1007/978-3-030-58298-2_6.

[5] Benjamin Binder et al. "The Role of Causality in a Formal Definition of Timing Anomalies". In: *IEEE 28th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. 2022, pp. 91–102. doi: 10.1109/RTCSA55878.2022.00016.

[6] Franck Cassez, René Rydhof Hansen, and Mads Chr. Olesen. "What is a Timing Anomaly?" In: *WCET*. Vol. 23. WCET'12. 2012. doi: 10.4230/OASIcs.WCET.2012.1.

References III

[7] J. Eisinger et al. "Automatic Identification of Timing Anomalies for Cycle-Accurate Worst-Case Execution Time Analysis". In: *DDECS*. 2006. doi: 10.1109/DDECS.2006.1649563.

[8] Gernot Gebhard. "Timing Anomalies Reloaded". In: *WCET*. WCET'10. 2010. doi: 10.4230/OASIcs.WCET.2010.1.

[9] Mathieu Jan et al. "Formal Semantics of Predictable Pipelines: a Comparative Study". In: *ASP-DAC*. 2020, pp. 103–108. doi: 10.1109/ASP-DAC47756.2020.9045351.

[10] R. Kirner, A. Kadlec, and P. Puschner. "Precise Worst-Case Execution Time Analysis for Processors with Timing Anomalies". In: *ECRTS*. ECRTS'09. July 2009. doi: 10.1109/ECRTS.2009.8.

[11] Raimund Kirner, Albrecht Kadlec, and Peter Puschner. *Worst-Case Execution Time Analysis for Processors showing Timing Anomalies*. Tech. rep. TU Wien, 2009.

References IV

[12] Jan Reineke and Rathijit Sen. "Sound and Efficient WCET Analysis in the Presence of Timing Anomalies". In: *WCET*. 2009.

[13] Jan Reineke et al. "A Definition and Classification of Timing Anomalies". In: *WCET*. WCET'06. 2006. doi: 10.4230/OASIcs.WCET.2006.671.

[14] Reinhard Wilhelm et al. "The Worst-Case Execution-Time Problem–Overview of Methods and Survey of Tools". In: *ACM Trans. Embed. Comput. Syst.* (May 2008). doi: 10.1145/1347375.1347389.