# Mixed Feelings about Mixed Criticality

Reinhard Wilhelm
Saarland University

WCET 2018, Barcelona


# more precisely

Mixed Feelings about
(a Popular Model of)
Mixed Criticality
(Systems)

or

soundness

It's ~~the economy~~, stupid

R.W.

-- ~~Bill Clinton~~

or

Seeking employment for an overpopulated community

# Contents

- Our WCET-analysis method
- Soundness
  - a bit of terminology
- Mixed criticality – the Vestal model
- Multi-core platforms – the disappearance of a very nice interface
- How to convince oneself and others of soundness – tool validation

# Deriving Run-Time Guarantees for Hard Real-Time Systems

Given:

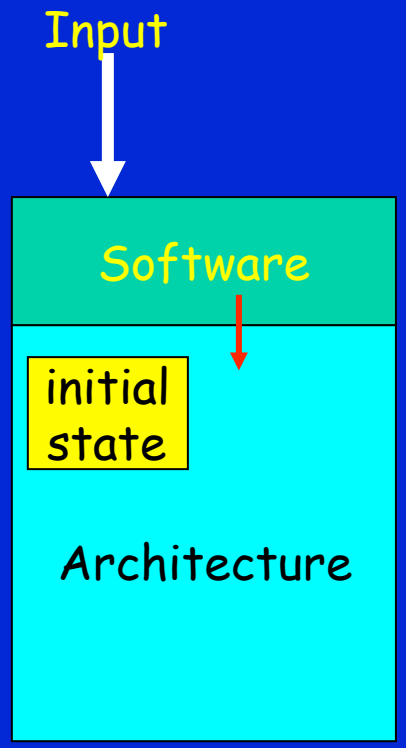
1. a software to produce a reaction,
2. a hardware platform, on which to execute the software,
3. a required reaction time.

Derive: a guarantee for timeliness

# What does Execution Time Depend on?

- the input,
- the initial and intermediate execution states of the platform,
- interferences from the environment – this depends on whether the system design admits it (preemptive scheduling, interrupts).

Caused by caches, pipelines, speculation etc. ⇒ Explosion of state space




"external" interference as seen from analyzed task

# High-Performance Microprosessors

- increase (average-case) performance by using: Caches, Pipelines, out-of-order execution, Branch Prediction, Speculation
- These features make timing analysis difficult: Execution times of instructions vary widely
  - Best case - everything goes smoothly: no cache miss, operands ready, resources free, branch correctly predicted
  - Worst case - everything goes wrong: all loads miss the cache, resources are occupied, operands not ready
  - Span may be several hundred cycles

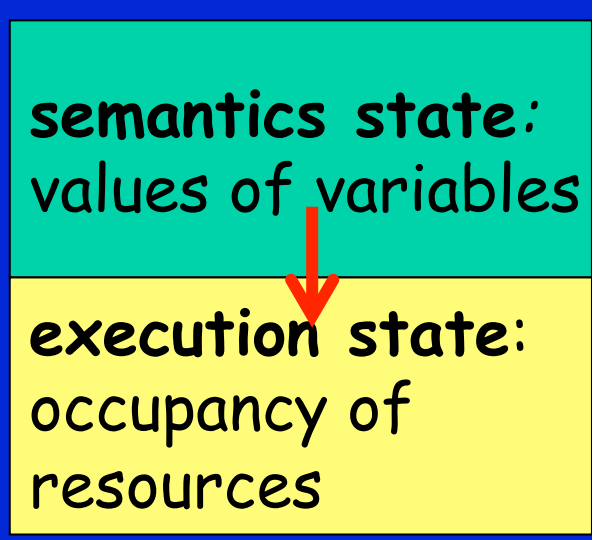
# Timing Analysis

- Sounds methods determine upper bounds for all execution times,
- can be seen as the search for a longest path,
  - through the huge combination of control-flow and architecture graphs,

1. I will show how this huge state space originates.
2. How and how far we can cope with this huge state space.

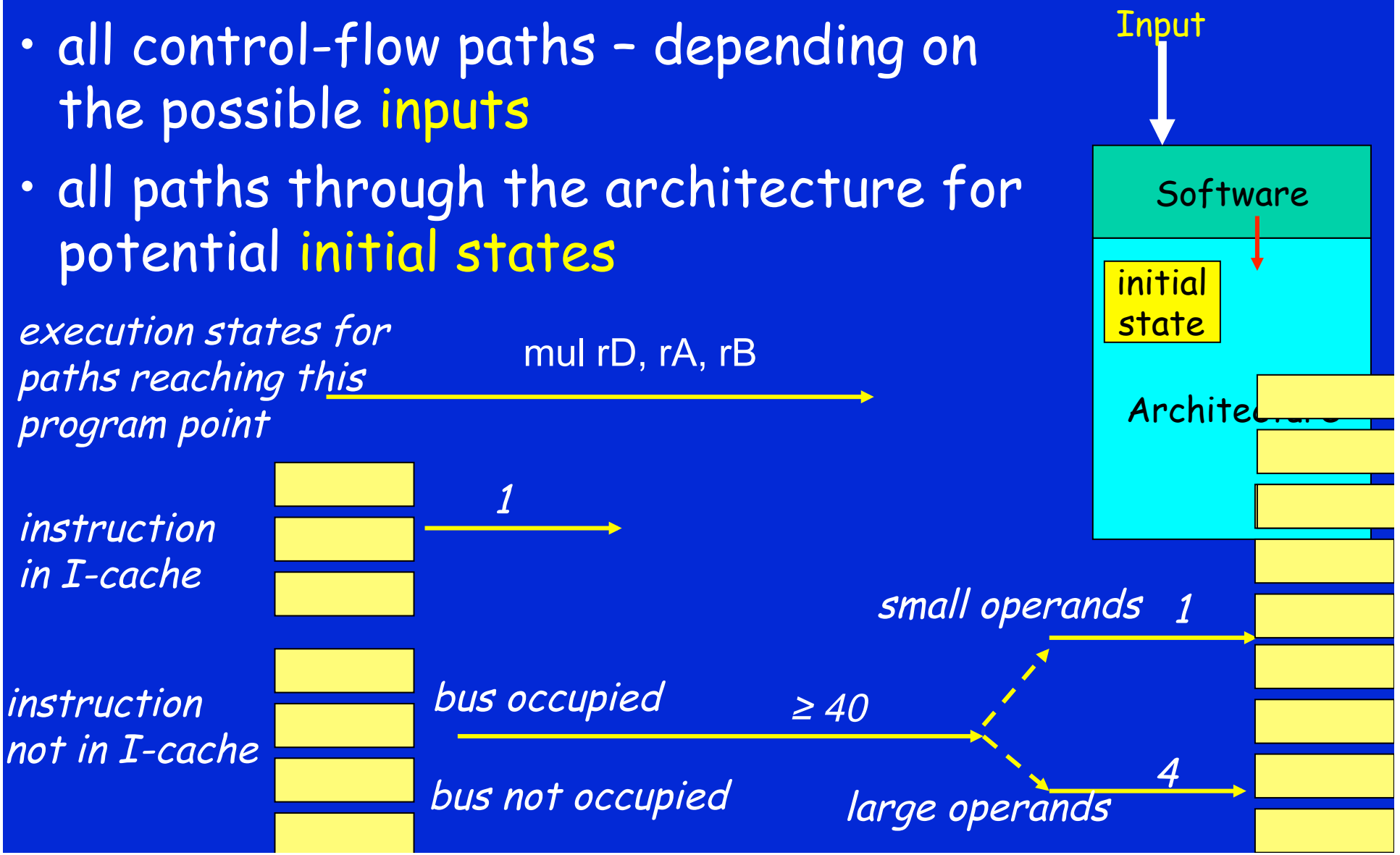
# State-dependent Execution Times

- Execution times of instructions depend on the execution state.
- Execution state
  - results from the execution history and
  - depends on the semantics state, e.g. computed address influences cache contents



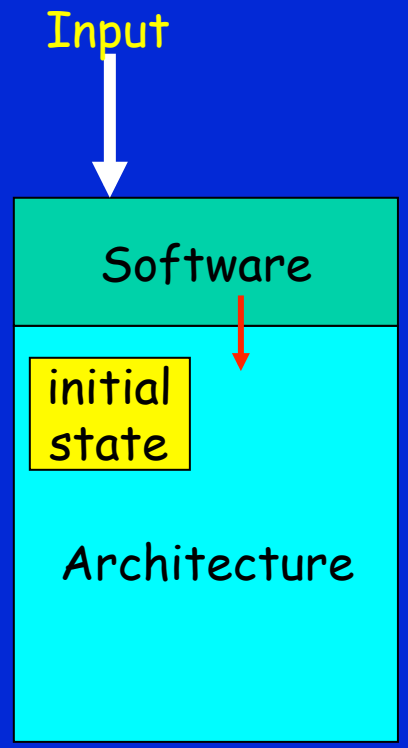
# Timing Analysis – the Search Space with State-dependent Execution Times

- all control-flow paths – depending on the possible inputs
- all paths through the architecture for potential initial states



# Timing Analysis – the Search Space with out-of-order execution

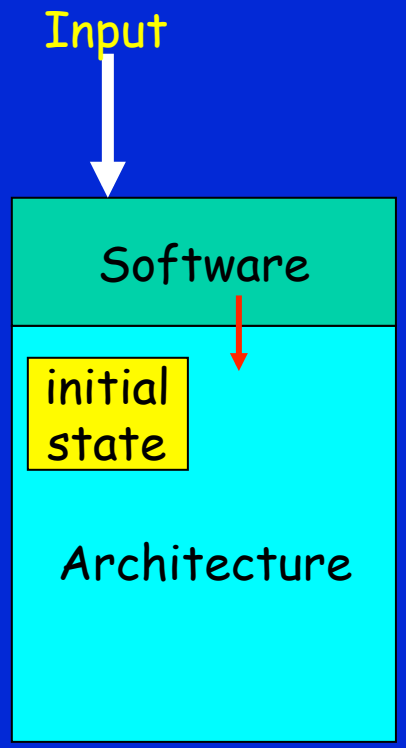
- all control-flow paths – depending on the possible inputs
- all paths through the architecture for potential initial states
- including different schedules for instruction sequences



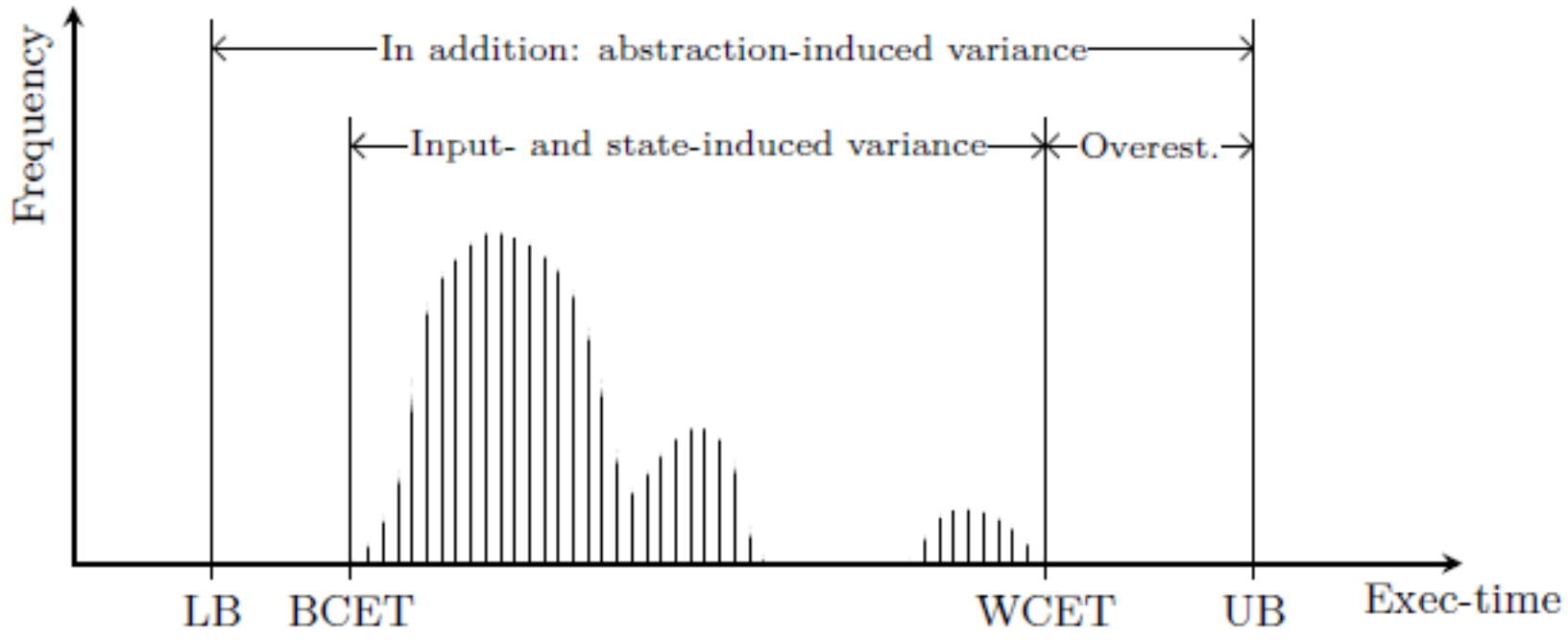
# Timing Analysis – the Search Space

with multi-threading or co-running tasks on multi-core platforms

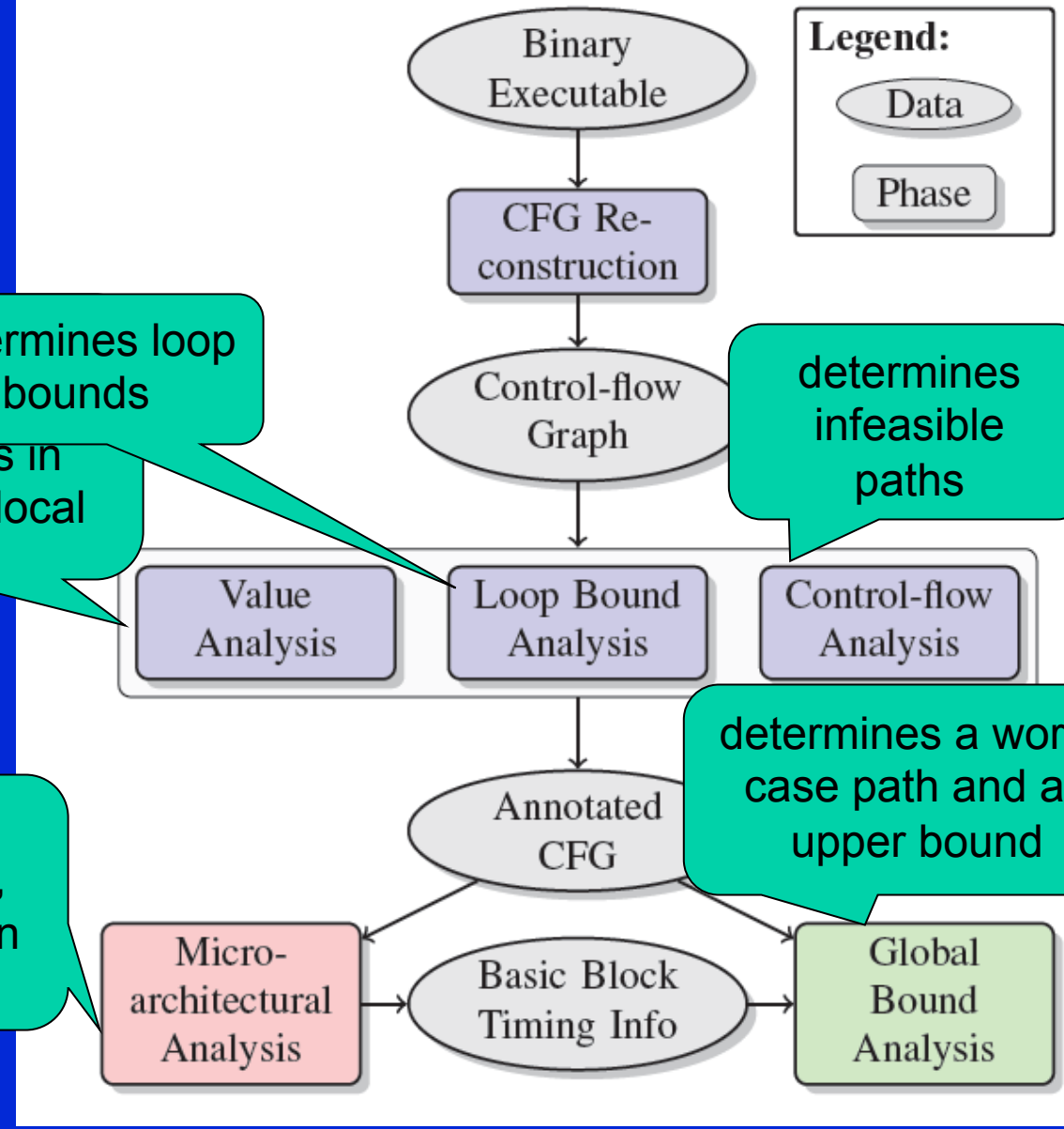
- all control-flow paths – depending on the possible inputs
- all paths through the architecture for potential initial states
- including different schedules for instruction sequences
- including different interleavings of accesses to shared resources



# Abstraction-Induced Imprecision



# Tool Architecture




*Abstract Interpretations*

*Abstract Interpretation*

*Integer Linear Programming*

# Timing Accidents and Penalties

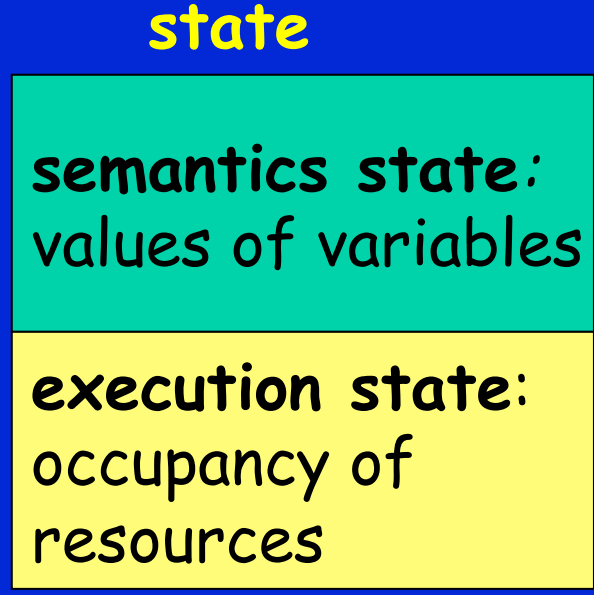
Timing Accident – cause for an increase of the execution time of an instruction

Timing Penalty – the associated increase

- Types of timing accidents
  - Cache misses
  - Pipeline stalls
  - Branch mispredictions
  - Bus collisions
  - Memory refresh of DRAM
  - TLB miss

# Our Approach

- Static Analysis of Programs for their behavior on the execution platform
- computes invariants about the set of all potential execution states at all program points,
- the execution states result from the execution history,
- static analysis explores all execution histories



# Deriving Run-Time Guarantees

- Our method and tool derives Safety Properties from these invariants:
  Certain timing accidents will never happen
  Example: At program point p, instruction fetch will never cause a cache miss
- The more accidents **excluded**, the **lower** the **upper** bound

# Implemented WCET Analysis

- The story told is conceptual
- Implementation:
  - analysis executes instructions of program u.a. on an abstraction of the execution platform
  - for each instruction it steps cycle-by-cycle through all possible, i.e. not excluded, transitions until retirement
  - follow all transitions whenever state information is missing (uncertainty, non-determinism)
  - iterates over the basic-block graph until convergence to a fixed point

# State of Affairs

- WCET analysis for single-core architectures is a solved problem
- Tools exist and have proved applicability in industrial practice
- No satisfactory solution for multi-core architectures with shared resources

# Abstraction and Decomposition

Components with domains of states $C_1, C_2, \ldots, C_k$

Analysis has to track domain $C_1 \times C_2 \times \ldots \times C_k$

Start with the powerset domain $2^{C_1 \times C_2 \times \ldots \times C_k}$

Find an abstract domain $C_1^{\#}$ transform into $C_1^{\#} \times 2^{C_2 \times \ldots \times C_k}$

This has worked for caches and cache-like devices and recently for strictly in-order pipelines

Find abstractions $C_{11}^{\#}$ and $C_{12}^{\#}$ factor out $C_{11}^{\#}$ and transform rest into $2^{C_{12}^{\# \times \ldots \times C_k}}$

This has worked for the arithmetic of the pipeline.


value analysis

microarchitectural analysis

# Mixed-Criticality Systems

- Safety-criticality systems have different levels of criticality,
  - Automotive Safety Integrity Level (ASIL) in ISO 26262
  - Development Assurance Level (DAL) in DO-178

# The Vestal Model of MCS, 2007

- The Vestal model of mixed-criticality systems (MCS) for schedulability analysis:
  Different WCETs are associated with different criticality levels of a task, the higher the criticality level, the higher the WCET estimate
- Motivation: better exploitation of HW resources
- His conjecture: *the higher the degree of assurance required that actual task execution times will never exceed the WCET parameters used for analysis, the larger and more conservative the latter values become in practice.*
- Note:
  - *no mention of soundness*
  - „More conservative" here does not mean sound!

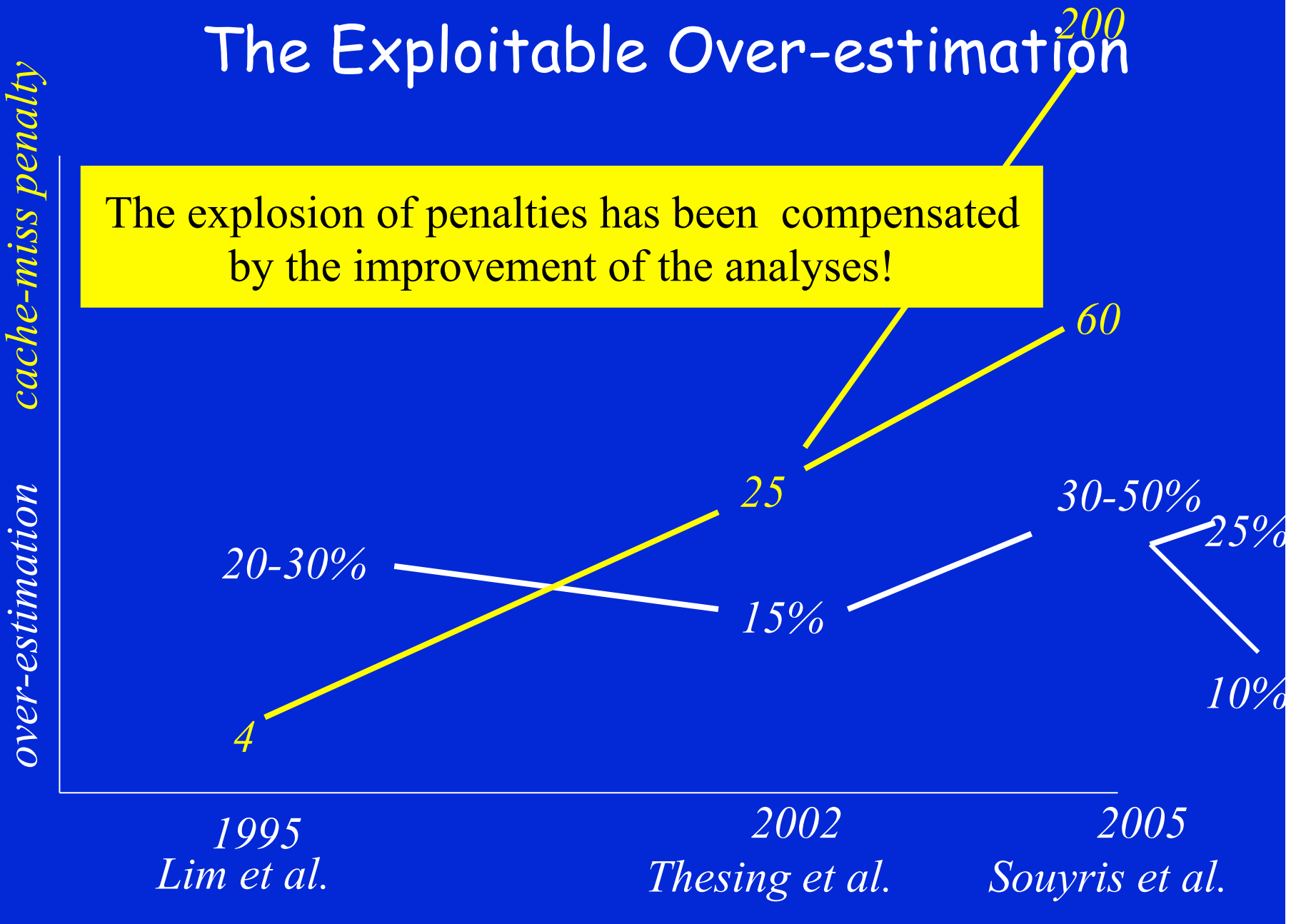
# Burns&Davis Survey of MCS, 2013

- Their assumption: *A key aspect of MCS is that system parameters, such as tasks' worst-case execution times (WCETs), become dependent on the criticality level of the tasks*
- Their doubts: *It would certainly be hard to estimate what increase in confidence would result from, say, a 10% increase in all Cs.*

# Better HW Exploitation

- Scheduling research has associated different budgets with the different criticality levels of tasks
- Allows scheduling of low-criticality tasks together with low-budgeted high-criticality tasks
- Assumes a big difference between a sound WCET estimate - the high budget – and the maximally observed WCET – the low budget – of a high-criticality task

# The Exploitable Over-estimation



# A Bit of Terminology

- A WCET estimate is conservative if no execution time exceeds it (a Boolean property!)
- A WCET-analysis method and associated tool are sound if they always produce conservative WCET estimates
- Sound WCET-analysis methods may be more or less accurate
- and more or less efficient

# Complaints against Sound Methods

- Implementing a sound WCET-analysis method, i.e. obtaining a correct abstract platform model, is so complex that one might use a simpler unsound method in the first place
- Sound methods produce high over-estimations
- Certification standards, e.g. DO178, are test based. So you do testing.

I'll come to that

Some measurement-based methods combine unsoundness with high over-estimation

Yes, but as a preprocess, more later

# What is the Background of the Vestal Model of MCS?

- My guess: measurement-based WCET analysis
- Mantra: Higher criticality requires higher assurance (apparently no soundness!)
- Folklore: Higher assurance obtained by larger sets of measurements
- Fact: Larger sets of measurements may produce higher maximal observed execution times.
- Still, no guarantee to obtain conservative estimates

# Alternative Reasons Given

- Longest paths often correspond to exceptional cases
- May be far off the "normal" execution times
- However, this should be dealt with as one instance of an operating mode, RW, Lucas, Parshin,Tan, Wachter 2012
- Failure modes may entail a different schedule with different instantiations of other tasks

# Criticism

- Scheduling researchers would claim soundness for their methods
- However, their soundness is only relative to the soundness of their inputs – which they don't care about
- Note: Soundness is not mentioned in Vestal 2007

# Multi-Core Execution Platforms

- The scheduling community has worked with a nice interface:
  - we determine sound/unsound WCET estimates
  - they use these as inputs to schedulability analyses
- Transition to multi-core execution platforms with shared resources invalidates this interface
- Different schedules lead to different interactions on shared resources and thus different execution times
- Different WCET estimates lead to different schedules
- This interface is still popular in the scheduling community

# Desirable: Separate WCET Estimation i.e. Compositionality

Certification Authorities Software Team. *Multi-core processors*, CAST-32A edition, November 2016, position paper

- requires Robust Partitioning of co-executing tasks as precondition for robust partitioning
- requires Mitigation (unspecified) if robust partitioning cannot be proven
- has some trouble understanding the characteristics of the different types of resources
  - bandwidth resources, e.g. buses
  - storage resources, e.g. caches
- Detailed treatment in Wilhelm R., Reineke J., Wegener S. (2018) Keeping up with Real Time. In: Durak U., Becker J., Hartmann S., Voros N. (eds) Advances in Aeronautical Informatics. Springer

# How justified is our Claim of Soundness?

- Consider tool qualification according to DO178, the intl. standard for safety-critical avionics systems
- WCET-analysis tools fare under verification tools,
- have lower certification requirements than development tools
- specification of functionality is required
- DO178 asks for testing, but
  - Testing as preprocessing
  - Coverage of ISA and architecture paths

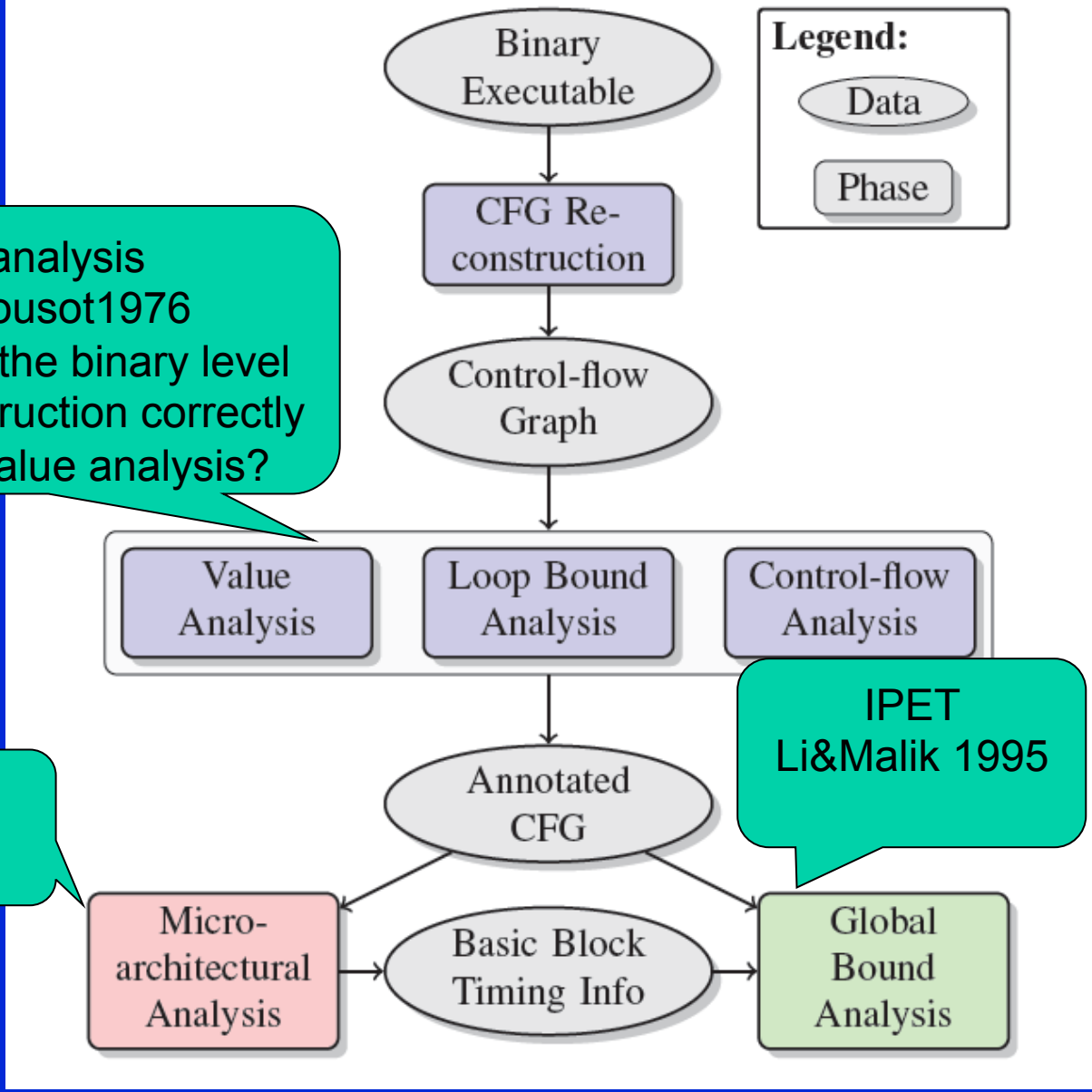
# The Claim of Soundness

- Certification more challenging through DO-333, the Formal-Methods Supplement to DO-178C
- Is a formal method including the underlying theory *adequate* for solving the verification problem, i.e. matches requirements?
- Positive answer enforces soundness of methods and tools

# Certification – Questions asked

- Underlying theory correct and adequate?:
  Abstract Interpretation
  40 years old, established, well-developed, accepted
- Implementation correct?

# Tool Architecture and used methods




interval analysis
Cousot&Cousot1976
Here applied at the binary level
Test: is each instruction correctly abstracted in value analysis?

*Abstract Interpretations Cousot&Cousot1976*

The critical component!

IPET
Li&Malik 1995

*Abstract Interpretation*

*Integer Linear Programming*

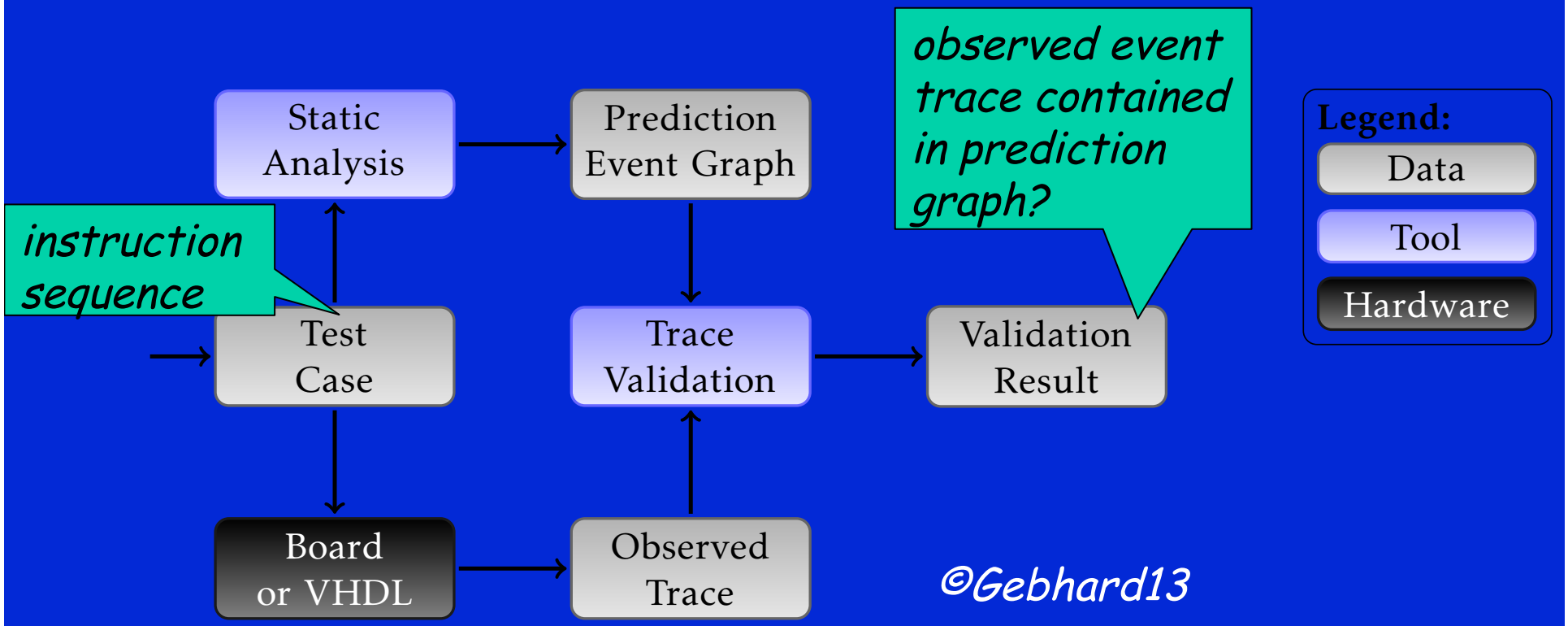
# Validation of Control-Flow Reconstruction

- Each instruction correctly disassembled?
- Additional compiler-dependent tests for typical control flows to show that switch tables and loops are correctly recognized

# Validation of Micro-architectural Analysis

- Essentially validation of the abstract execution platform
- Done by Trace Validation …
- at validation time
- Decouples effort (spent at validation time) from precision (achieved at analysis time)

# Trace Validation



# Trace Validation

- Prediction graph contains only events that can be externally observed
- Observed trace must be *contained* in the prediction graph
- Trace is extended cycle by cycle through interrupts
- The observed trace, the reached state, and the consumed time are checked against the prediction graph
- The predicted time may be larger than the observed time, but never smaller
- Not explicitly observable state components, e.g. cache states, are forced into a "deterministic" state by choosing the right initial states
- Altogether, a tremendous effort is spent on validation

# Conclusion

- I tried to pick up some loose ends on WCET analysis
- pointed to bizarre assumptions of the scheduling community
- Main focus was on soundness
- and how to convince oneself, customers, and the certification authorities of soundness

# Recent publications:




- S. Hahn, J. Reineke, R.Wilhelm: *Toward Compact Abstractions for Processor Pipelines*. in Correct System Design 2015: 205-220
- S. Hahn, J. Reineke, R. Wilhelm: *Towards compositionality in execution time analysis: definition and challenges*. SIGBED Review 12(1): 28-36 (2015)
- J. Reineke, R.Wilhelm: *Static Timing Analysis - What is Special? in Semantics, Logics, and Calculi 2016: 74-87*
- M. Lv, N.Guan, J.Reineke, R.Wilhelm, W. Yi: *A Survey on Static Cache Analysis for Real-Time Systems*. LITES 3(1): 05:1-05:48 (2016)
- R. Wilhelm, R. Reineke, S. Wegener: *Keeping up with Real Time*, in Advances in Aeronautical Informatics, Springer 2018