

# Timing analysis and timing predictability

## Introduction

Reinhard Wilhelm

Saarland University, Saarbrücken, Germany

ArtistDesign Summer School in China 2010



Safety critical applications:

- Avionics, automotive, train industries, manufacturing control



Sideairbag in car, Reaction in  $<10$  mSec



Embedded controllers must finish their tasks *within given time bounds*. Developers would like to know the *Worst-Case Execution Time (WCET)* to give a guarantee



Crankcraft-synchronous tasks,  
Reaction in  $<45$   $\mu$ Sec

## The Problem:

### ■ Given

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

### ■ Derive:

- ▶ a guarantee for timeliness.

# What does the execution time depends on?

- the input—this has always been so and will remain so,
- the initial state of the platform—this is (relatively new)
  
- interferences from the environment—this depends on whether the system design admits it (preemptive scheduling, interrupts, multi-core)

# What does the execution time depends on?

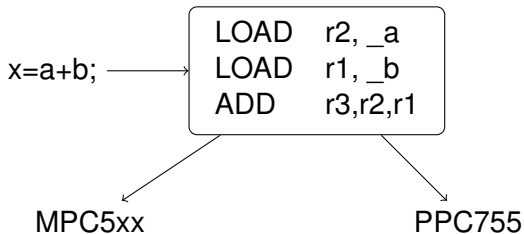
- the input—this has always been so and will remain so,
- the initial state of the platform—this is (relatively new)
  - ▶ Caused by caches, pipelines, speculation, etc.
  
- interferences from the environment—this depends on whether the system design admits it (preemptive scheduling, interrupts, multi-core)

# What does the execution time depends on?

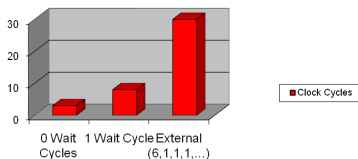
- the input—this has always been so and will remain so,
- the initial state of the platform—this is (relatively new)
  - ▶ Caused by caches, pipelines, speculation, etc.
  - ▶ Explosion of the space of inputs **and** initial states  
⇒ all exhaustive approaches infeasible
- interferences from the environment—this depends on whether the system design admits it (preemptive scheduling, interrupts, multi-core)

# What does the execution time depends on?

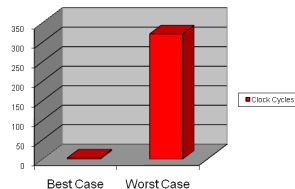
- the input—this has always been so and will remain so,
- the initial state of the platform—this is (relatively new)
  - ▶ Caused by caches, pipelines, speculation, etc.
  - ▶ Explosion of the space of inputs **and** initial states  
⇒ all exhaustive approaches infeasible
- interferences from the environment—this depends on whether the system design admits it (preemptive scheduling, interrupts, multi-core)
  - ▶ External interferences as seen from analyzed task



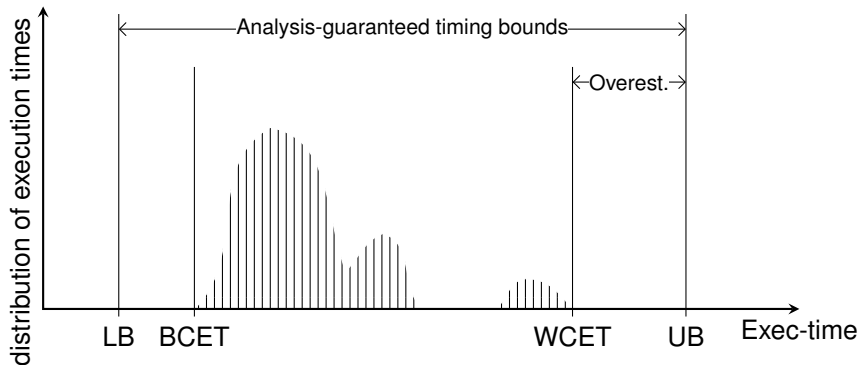
Execution Time depending on Flash Memory  
(Clock Cycles)



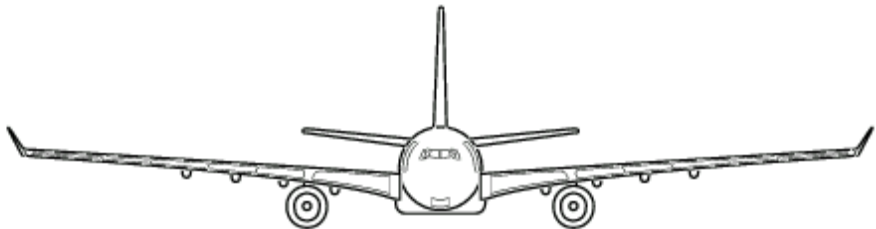
Execution Time (Clock Cycles)



# Timing Analysis

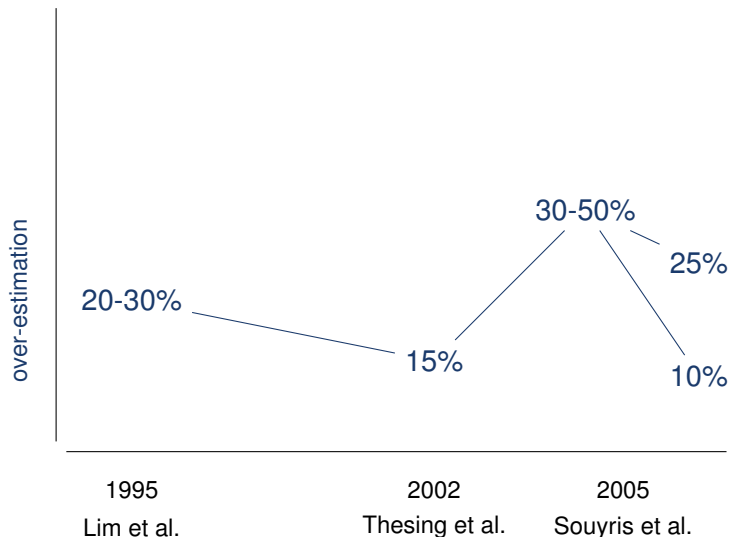


IST Project DAEDALUS final review report:  
*"The AbsInt tool is probably the best kind in  
the world and it is justified to consider this  
result as a breakthrough."*

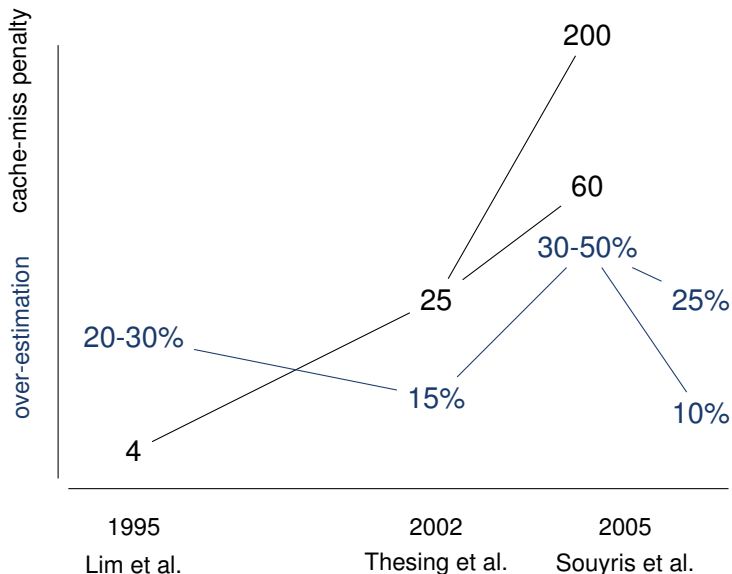


Several time-critical subsystems of the airbus A380 have been certified using aiT;  
aiT is the only validated tool for these applications.

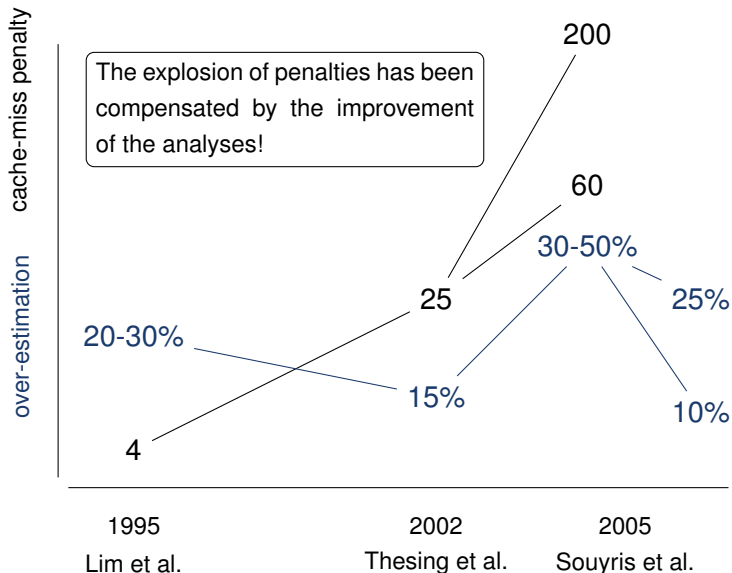
# Tremendous Progress during the 15 past Years

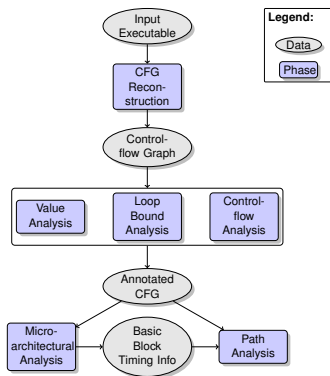


# Tremendous Progress during the 15 past Years



# Tremendous Progress during the 15 past Years





## ■ Value Analysis:

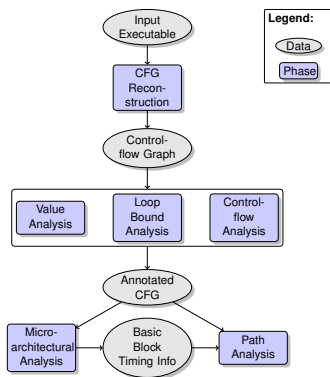
- ▶ determines enclosing intervals for the values in registers and local variables

## ■ Loop Bound analysis:

- ▶ determines loop bounds

## ■ Control Flow Analysis:

- ▶ determines infeasible paths



## ■ Value Analysis:

- ▶ determines enclosing intervals for the values in registers and local variables

## ■ Loop Bound analysis:

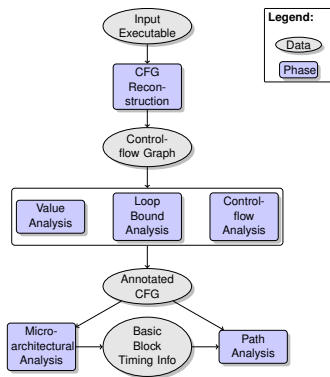
- ▶ determines loop bounds

## ■ Control Flow Analysis:

- ▶ determines infeasible paths

## ■ Micro-architectural Analysis:

- ▶ combined cache and pipeline analysis
- ▶ derives invariants about architectural execution states, computes bounds on execution times of basic blocks



## ■ Value Analysis:

- ▶ determines enclosing intervals for the values in registers and local variables

## ■ Loop Bound analysis:

- ▶ determines loop bounds

## ■ Control Flow Analysis:

- ▶ determines infeasible paths

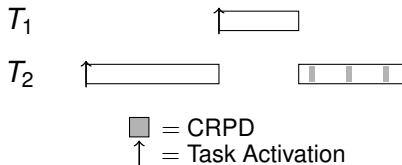
## ■ Micro-architectural Analysis:

- ▶ combined cache and pipeline analysis
- ▶ derives invariants about architectural execution states, computes bounds on execution times of basic blocks

## ■ Global Bound Analysis:

- ▶ determines a worst-case path and an upper bound

- Timing analysis:
  - ▶ upper-bound on the execution time (WCET)
  - ▶ uninterrupted execution
- Preemptive scheduling
  - ▶ WCET
  - ▶ Context-switch cost



- Predictability: not a boolean property
- Some performance-enhancing features, like certain
  - ▶ Caches
  - ▶ Pipelinesare analyzable, others are not...
- Explore trade-offs between (worst-case )predictability, (average-case) performance, and costs
- Goal:  
Design architectures with high worst-case performance that can be precisely and efficiently determined

- Predictable cores are a prerequisite for predictable multi-cores
- The main culprit: *Sharing* of resources
  - ▶ Main memory, caches
  - ▶ Busses
  - ▶ I/O
  - ▶ Flash memory
- introduces variability of execution times and, thus, imprecision,
- increases the state space to analyze and, thus, the complexity.

- Caches
- Bounding the preemption delay
- Pipeline analysis and Predictability of architectures