Static analysis of WCET in an experimental satellite software subsystem.

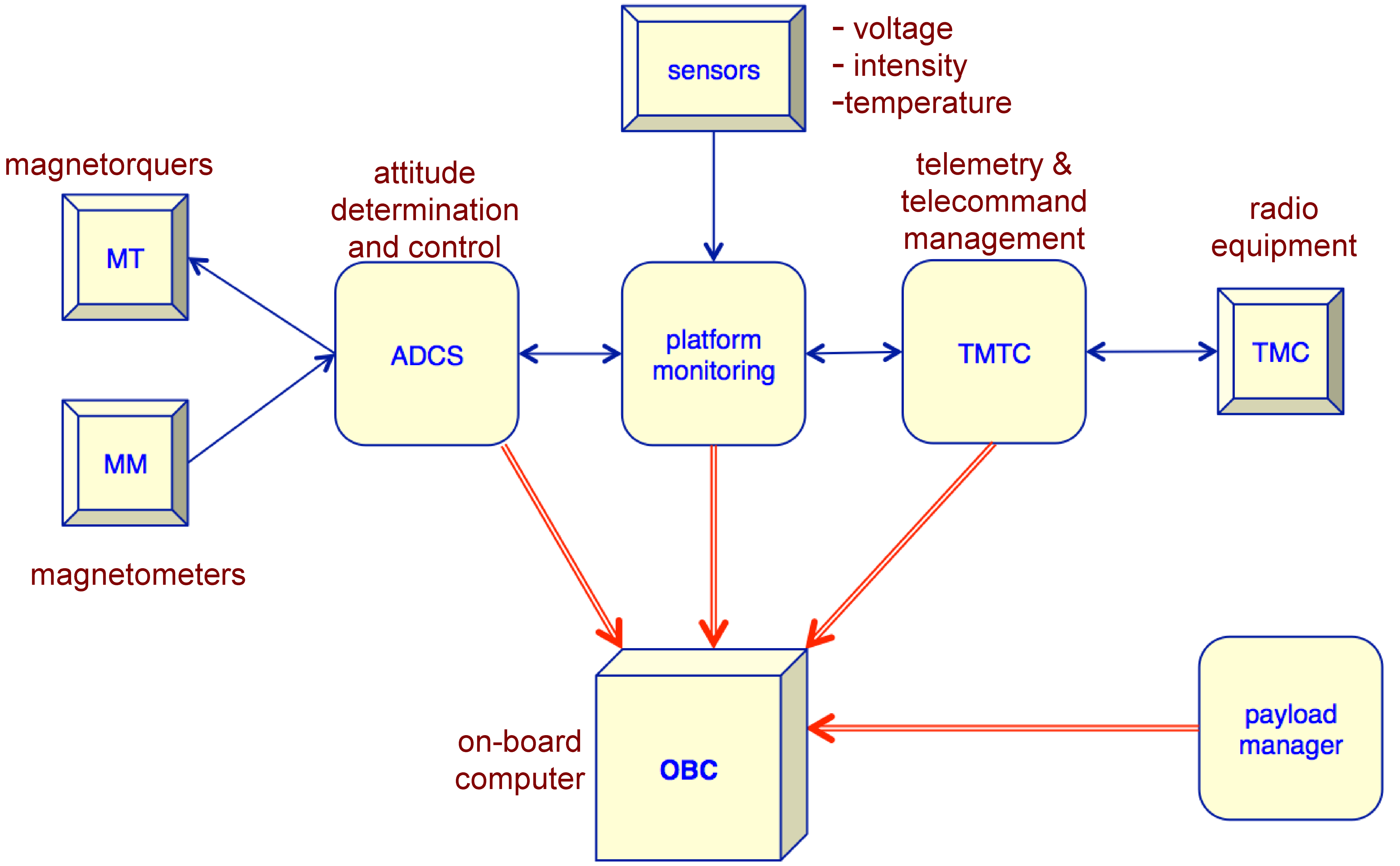
Jorge Garrido
Juan Zamorano
Juan A. de la Puente

Universidad Politécnica de Madrid, Spain

Aims

- To experiment with using static analysis WCET tools
- Study influence of LEON processors singularities
- Test system: UPMSat2 micro-satellite on-board computer
  - simple, but yet realistic system
  - software developed using an MDE approach
    - functional code auto-generated from Simulink
    - concurrency and real-time behaviour provided by containers
  - WCET analysis required for schedulability analysis
    - required by ESA standards

UPMSat2 on-board computer architecture



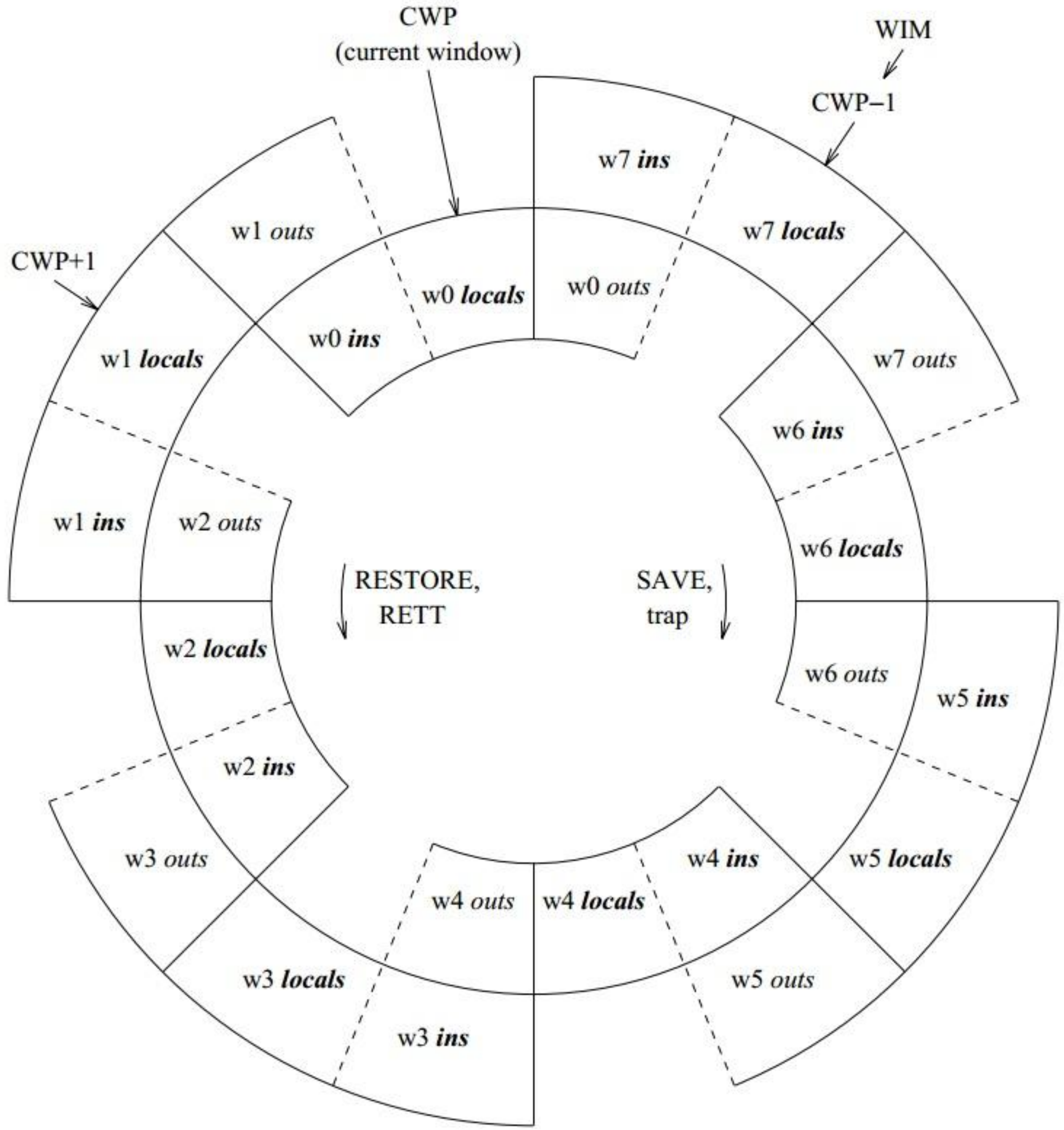
ADCS - Attitude Determination and Control System

- Orientation with respect to Earth
- Designed by aerospace engineers with Simulink
- C code autogenerated
  - Linear
  - Vector arithmetics
  - Embedded into Ada cyclic task

SPARC register windows

- Sets of 32 general purpose registers
  - Part of each set overlaps with the next one, allowing to pass parameters using registers
- Implemented as a circular buffer
  - Size is implementation dependant
  - When it gets full, next function call causes an overflow
  - Similarly, when it gets empty, next return causes an underflow
- Overflows and underflows trigger handler routines
  - Handler routines pose WCET overhead
  - Behavior is implementation dependant

SPARC register window



WCET static analyzers

- Allow early analysis of binary executables
- Can perform stack analysis
- Disadvantages:
  - Processor-specific
  - Need to be configured
  - Depend on assertions
  - Incomplete due to processors complexity

Analyzers used

- Static analyzers
  - a3
  - Bound-T
- Dynamic analyzers
  - Rapitime

a^3

- Developed by AbsInt

- SPARC register windows
  - Assumes an unlimited number of register windows
  - Stack analysis can obtain the max. depth of the register window stack
  - Register window overflow and underflow overhead has to be calculated by the user.

Bound-T

- Developed by Tidorum Ltd
- ERC32 support
- Terminal interface
- Graphical representation of results by 3rd party tools
- Rapitime integration

- SPARC register windows
  - Specific number of register windows support
  - Initial number of used register windows
  - Register window overflow and underflow prediction
    - Automatic register window overflow and underflow trap handler detection and analysis

Study strategy

- Compute a base WCET with a^3
- Measure a WCET for overflow and underflow trap routines by dynamic analysis
  - 156 cycles per overflow
  - 188 cycles per underflow
- Study the worst-case number of trap occurrences
  - Relevant information from Bound-T
  - Implementation dependent
    - Windows saved/restored in trap routine
    - Windows restored after context switch
- Compute the register window WCOH → WCET

Register windows overhead

- Number of traps in a function

$$N_f = n_f \times T_f$$

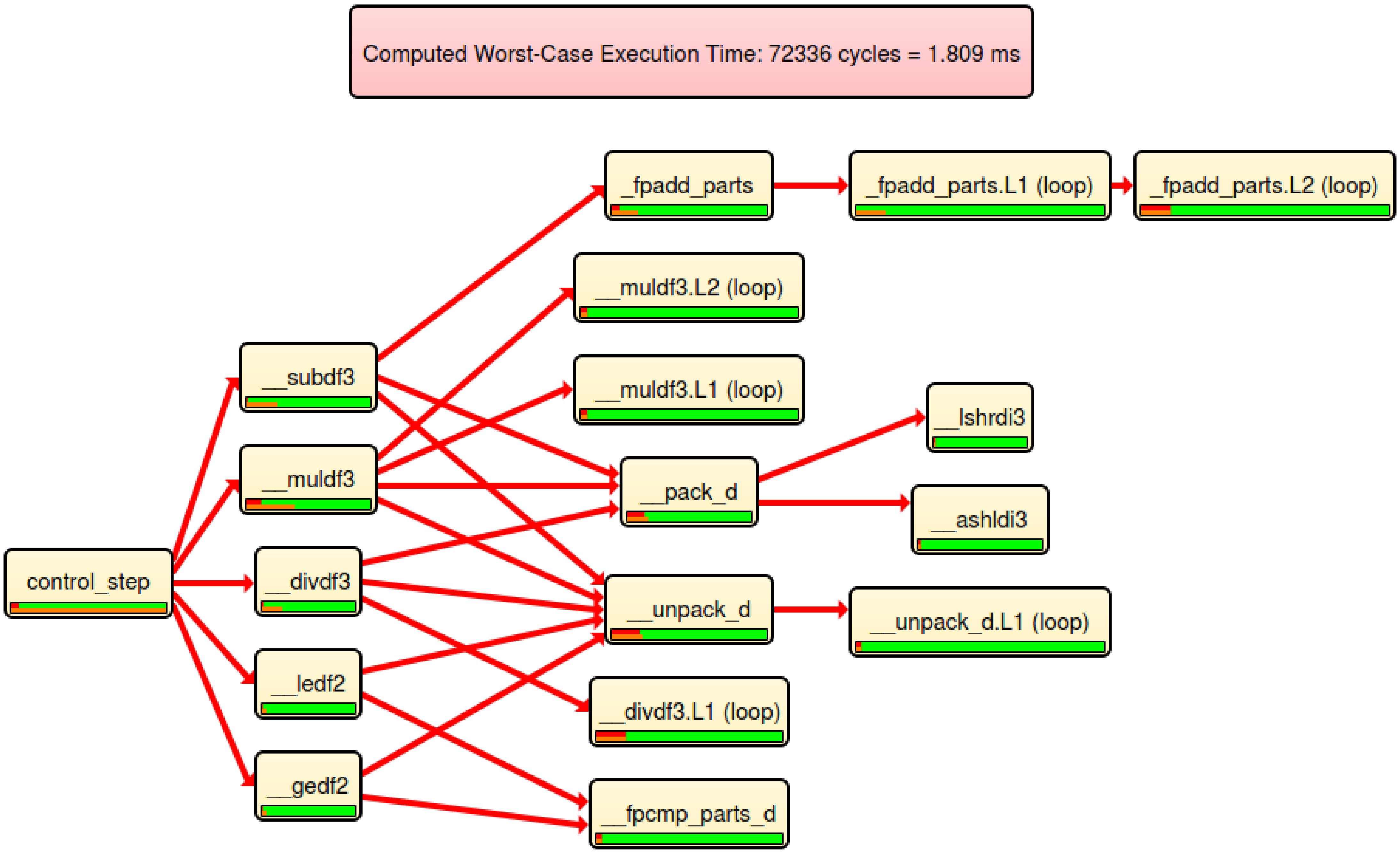
- Number of traps in the worst-case path

$$N = \sum_{f \in F} N_f$$

- Worst-case overhead

$$\mathrm{WCOH} = N \times \mathrm{WCET}_T$$

Call tree


Computed Worst-Case Execution Time: 72336 cycles = 1.809 ms
_fpadd_parts
_fpadd_parts.L1 (loop)
_fpadd_parts.L2 (loop)
__muldf3.L2 (loop)
__muldf3.L1 (loop)
__subdf3
__muldf3
__lshrdi3
__pack_d
__ashldi3
control_step
__divdf3
__unpack_d
__unpack_d.L1 (loop)
__ledf2
__divdf3.L1 (loop)
__gedf2
__fpcmp_parts_d

Case studies

- One window saved/restored on traps, only current window restored on context switches
- One window saved/restored on traps, full windows set restored on context switches
- Full set saved/restored on traps, full set restored on context switches

Case study I

- One window saved/restored on traps, only current window restored on context switches
  - a^3 reports 72366 cycles as $WCET_B$
  - Bound-T reports 25 overflows and 25 underflows

$$WCOH = 25 \times 156 + 25 \times 188 = 8600 \text{ cycles}$$

$$WCET = WCET_B + WCOH = 80966 \text{ cycles } (+11.56\%)$$

Case study II

- One window saved/restored on traps, full windows set restored on context switches
  - a^3 reports 72366 cycles as $WCET_B$
  - Number of overflows in worst-case is equal to max. depth of register windows that code may create.
  - Underflows only occur if depth is higher than processor number of register windows.

$$WCOH = 3 \times 156 + 0 \times 188 = 468 \text{ cycles}$$

$$WCET = WCET_B + WCOH = 72834 \text{ cycles } (+0.63\%)$$

Case study III

- Full set saved/restored on traps, full set restored on context switches
  - a^3 reports 72366 cycles as $WCET_B$
  - In case study, worst case happens when the controller is called using the last available window, so calls to floating point routines cause a trap

$$WCOH = 24 \times 156 + 24 \times 188 = 8256 \text{ cycles}$$

$$WCET = WCET_B + WCOH = 80622 \text{ cycles } (+11.28\%)$$

Comparison with dynamic analysis

- Improving former dynamic WCET analysis
  - Same code was previously analyzed using a hardware-in-the-loop approach.
    - Rapitime reported 8400 cycles as $WCET_B$
  - Refined results for Rapitime's WCET:

1 w. no restore	17000 cycles (+102.38%)
1 w. full restore	8868 cycles (+5.57%)
7 w. full restore	16656 cycles (+98.28%)

Analysis of results

- Implementation decisions have a strong influence on the overhead
- For dynamic analysis, register windows overhead can double measured WCET
- Even for more pessimistic WCET, the register windows overhead is far from trivial

Conclusions

- UPMSat2 good testbed for experimenting with high-integrity real-time technology
- Static analyzers good first-step WCET analysis, although more pessimistic
- Register window analysis has to be included in WCET measurements for LEON processors
  - Static analyzers provide useful information

Analysis of WCET in an experimental satellite software development

Jorge Garrido
Juan Zamorano
Juan A. de la Puente

Universidad Politécnica de Madrid, Spain