Worst-Case Execution Time and Reactive Systems

Claire Maiza

in collaboration with:
Nicolas Halbwachs, Pascal Raymond, Catherine Vigouroux,
Erwan Jahier, Fabienne Carrier,Hamza Rihani,
Matthieu Moy, Amaury Graillat, Wei Tsu Sun, Hugues Cassé

Journée Nicolas, 4/06/2018

Context: WCET and Reactive Systems

Worst-case execution time Estimation

- A guaranteed bound on the execution time
- Static estimation
  - Based on static analysis
  - At the binary level
  - Measurement to assess the WCET estimation

Reactive systems

- A periodic step function
  - Bounded memory
  - From input values (and memory state) computes output value (and memory update)
- Lustre/SCADE code and tools

In this talk

How a better knowledge on reactive systems helps timing analysis?

- Infeasible path: Semantic analysis
- WCET assessment: Environment simulator
- Multi-core timing analysis

Outline

1 Semantic analysis

2 Environment Simulator

3 Multi-Core timing analysis

Use of Lustre verification tool to check the feasibility of execution paths

Analysis flow

1. Find the set of Lustre expression that influence the binary execution path
2. Check the feasibility of paths
   - Pairwise properties
   - Full path properties
3. When infeasible path, give the property to WCET analysis (OTAWA)

What did we learn?

Results

- Up to 50% WCET improvement in case of automaton
- Mainly properties due to reactive systems that are hard or impossible to find at lower level

Our Related Work

- Semantic analysis at C level (using SMT encoding or abstract interpretation and Pagai tool)
- Properties that may be encoded in ILP

Outline

1 Semantic analysis

2 Environment Simulator

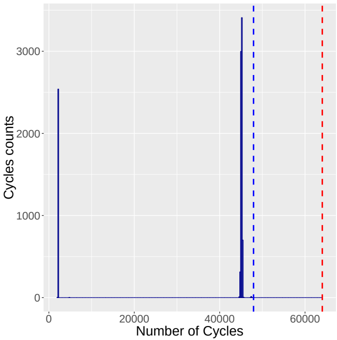
3 Multi-Core timing analysis

Use of simulation to assess the WCET estimation

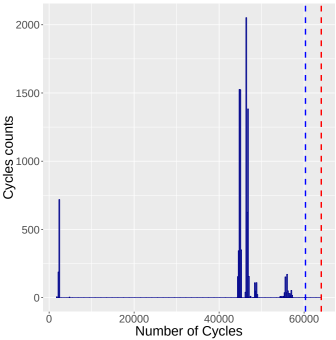
Where are the difficulties?

- Use the same platform model as in the WCET estimation: Osim, OTAWA
- Take into account infeasible path as in the WCET estimation: Lutin
- Take into account environment scenario: Lutin

Why an environment simulator to assess WCET estimations?



Only Semantic properties:
Estimation should be improved?



Scenario-guided environment:
Good WCET estimation :)

What did we learn?

- Naive simulation of reactive systems may lead to wrong assessment
- This assessment is very useful to know when the WCET estimation is precise “enough”

Outline

1 Semantic analysis

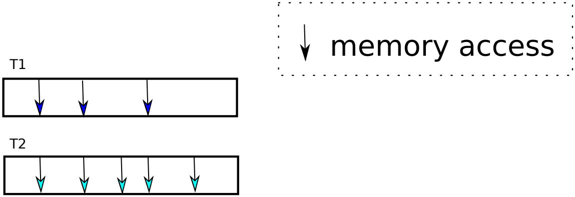
2 Environment Simulator

3 Multi-Core timing analysis

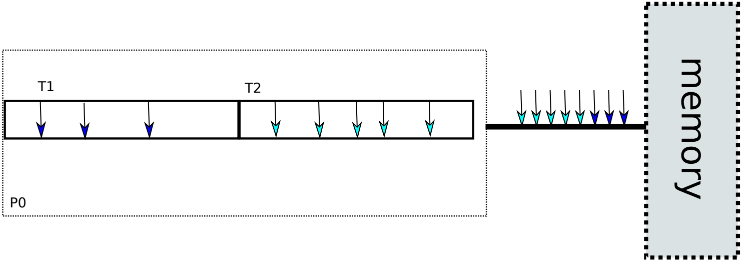
Timing analysis on multi-core

- Interferences on shared resources
- An intuition on why it is complex (complexity of the analysis and loss of precision)
- Our analysis on a cluster of the Kalray MPPA

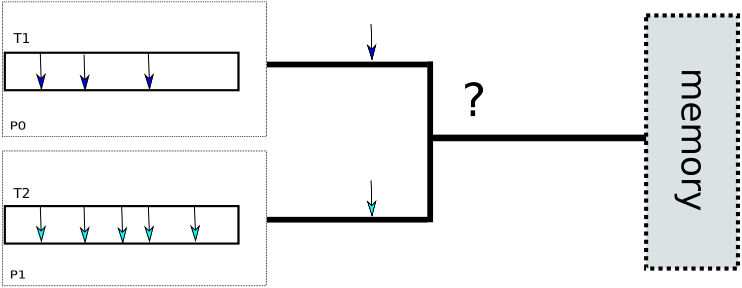
Very simple case study



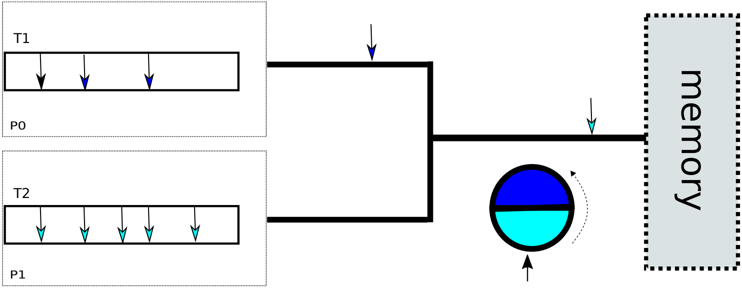
One core: sequential execution



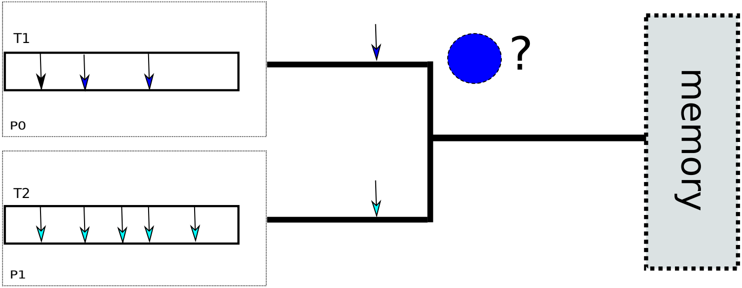
2 cores: the arbiter



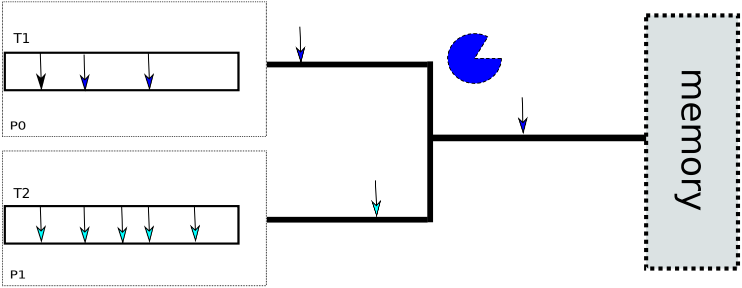
2 cores: TDMA



2 cores: Round Robin



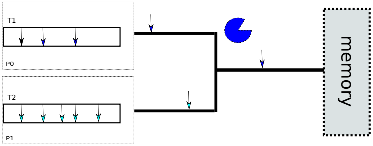
2 cores: Round Robin, global interference anlaysis



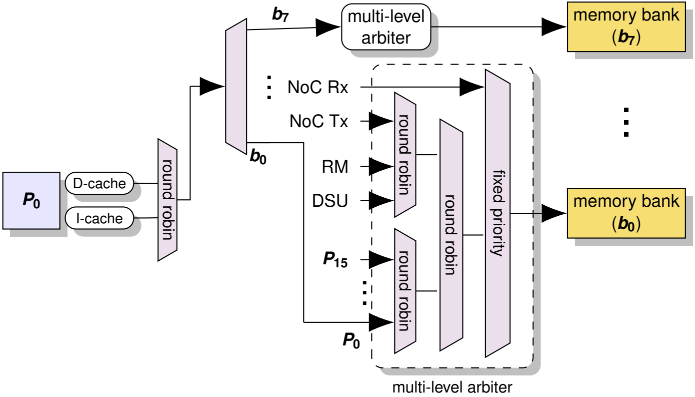
2 cores: Round Robin, global interference anlaysis

how to take the access memory delay into account?

- worst case: $access_{task2} = slotSize$
- n cores, worst case: $access_{task2} = (n - 1) \times slotSize$
- more precise: TDMA-like analysis = scalability issue
- more precise: minimum of the worst-case numbers of accesses (WA) of concurrent tasks
  $delay_{task2} = min(WA_{task2}, WA_{task1}) \times slotSize = min(5.3) \times slotSize = 3 \times slotSize$



An example of memory architecture: a cluster of the Kalray MPPA bostan



How to reduce the complexity?

Why is it complex?

- WCET analysis without any knowledge on the application (communication, resource sharing)
- Consider full resource sharing

In our context

- Lustre/SCADE code:
  - Data-flow: limited communication actors and better knowledge on the possible interfering nodes
- MPPA bostan:
  - Each memory bank assigned to one core
  - Local read + global write only by predecessor
  - Round robin

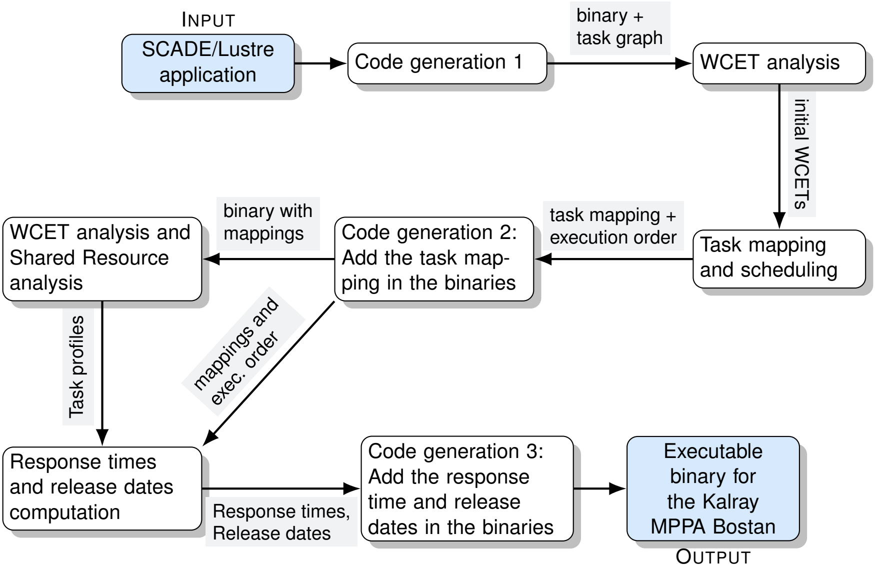
Response-time analysis with interference analysis

Algorithm

Input = Isolated WCET + Worst-case memory access number + Initial scheduling/mapping

1 Estimate current interference delay
  1.1 For all tasks that interfere on the current scheduling on each memry bank, due to execution or write phases
  1.2 Add this interference delay to the initial WCET
2 Reajust release dates preserving precedence constraint and restart [1] with the new scheduling

Capacites full framework


INPUT
SCADE/Lustre application
Code generation 1
binary + task graph
WCET analysis
initial WCETs
Task mapping and scheduling
task mapping + execution order
Code generation 2: Add the task mapping in the binaries
binary with mappings
WCET analysis and Shared Resource analysis
Task profiles
mappings and exec. order
Response times and release dates computation
Response times, Release dates
Code generation 3: Add the response time and release dates in the binaries
Executable binary for the Kalray MPPA Bostan
OUTPUT

What did we learn?

Multi-core timing analysis is feasible with a better knowledge on the application and a better knowledge/usage of the platform.
WCET analysis and Lustre/SCADE implementation are inter-dependent in this context.

Merci Nicolas!

- Response Time Analysis of Synchronous Data Flow Programs on a Many-Core Processor. Rihani, Hamza and Moy, Matthieu and Maiza, Claire and Davis, Robert I. and Altmeyer, Sebastian (in RTNS 2016, 2016)
- Timing analysis enhancement for synchronous program. Raymond, Pascal and Maiza, Claire and Parent-Vigouroux, Catherine and Carrier, Fabienne and Asavoae, Mihail (in Real-Time Systems, 2015)