WCET and interference analysis for multicore processor

Claire Maiza

Hardware interference and temporal determinism for modern SoC

Introduction

What does mean “Worst-case execution time” on a multi-core?

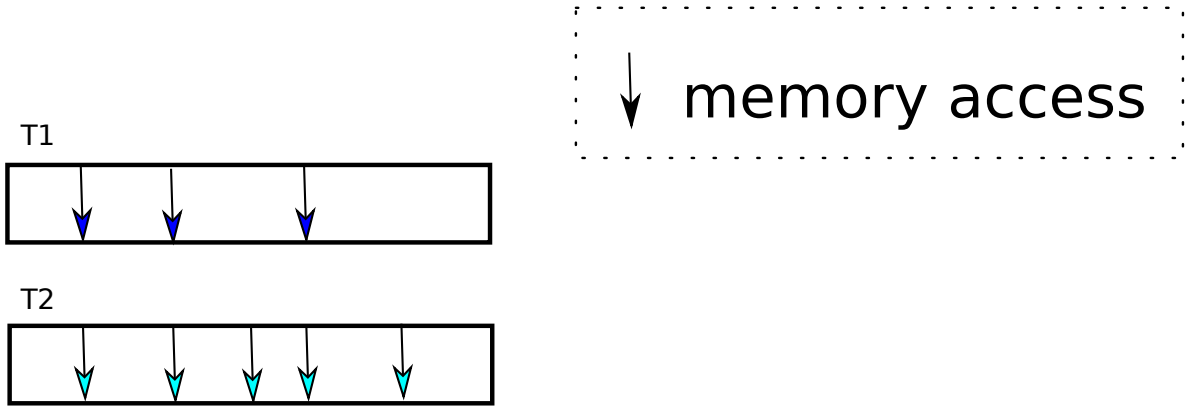
- Monocore: WCET = isolated execution time
- Multi-core: WCET + additional interference delay = “worst-case response time”

In this talk

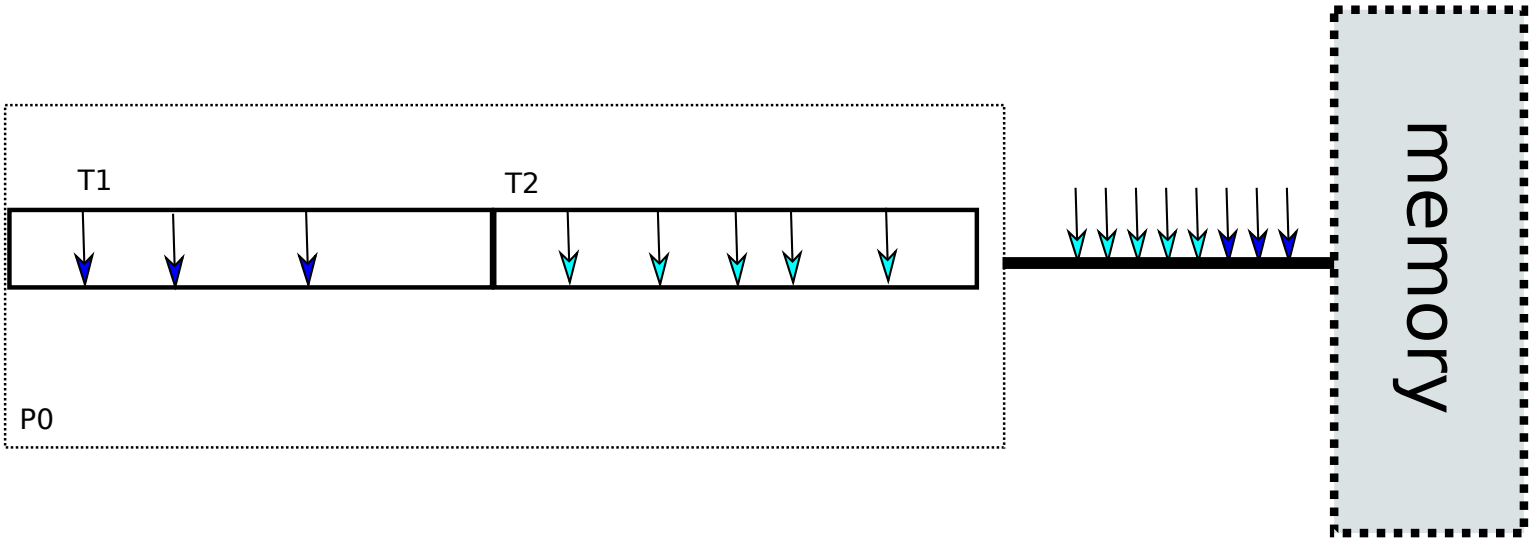
- How to bound the delay due to shared ressource? Example of memory access
- State of the art multicore timing analysis
- Our timing analysis on a MPPA2 cluster

Outline

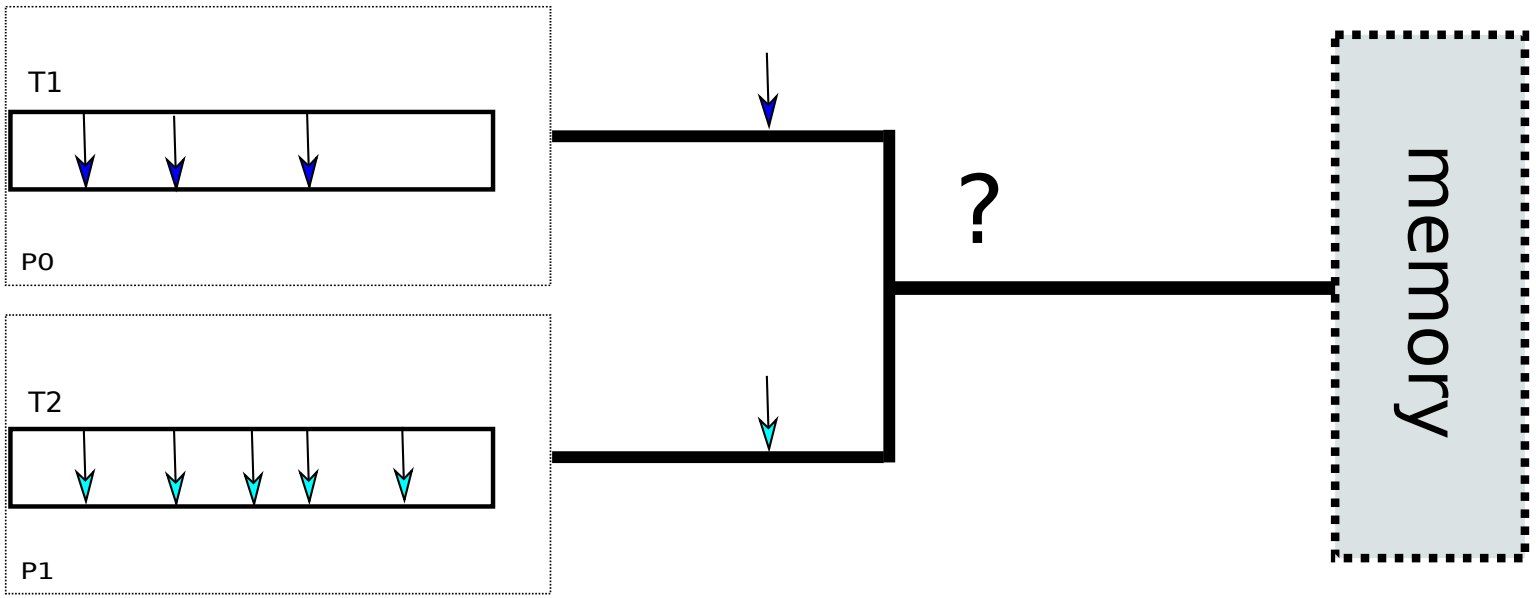
Very simple case study



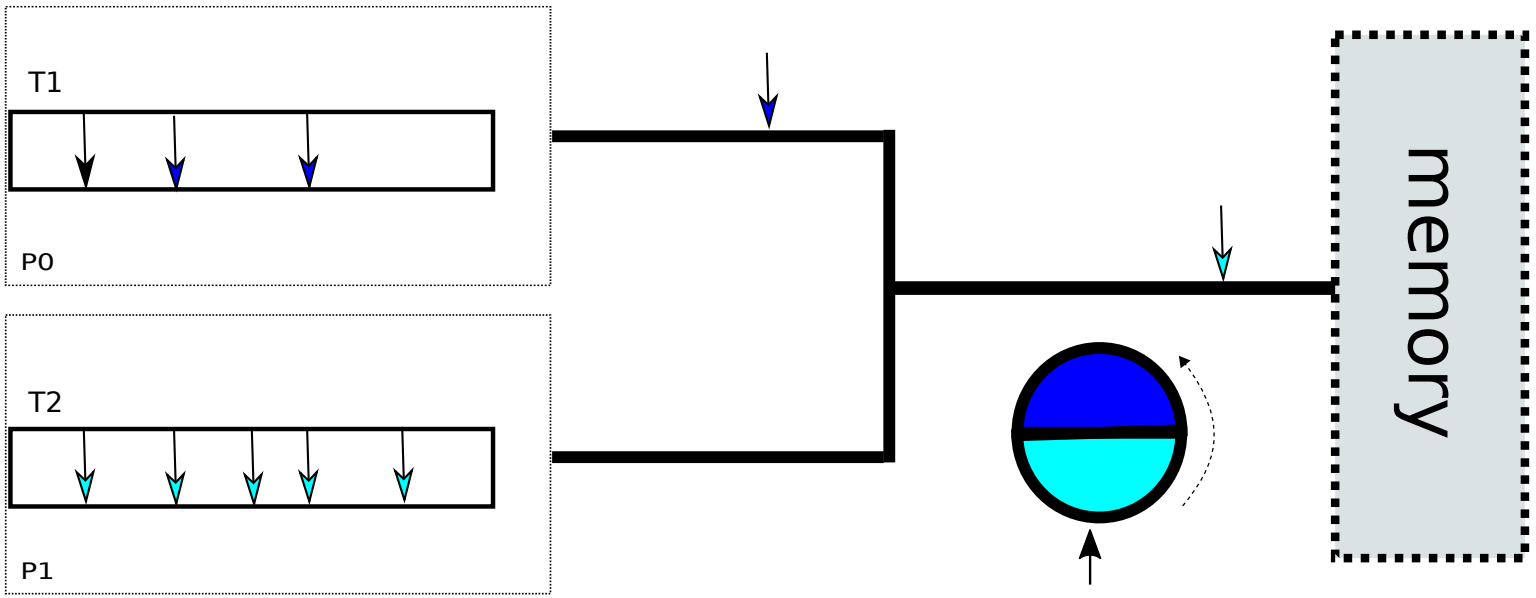
One core: sequential execution



2 cores: the arbiter



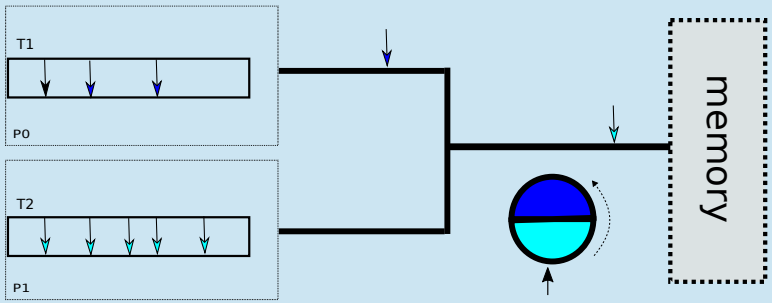
2 cores: TDMA



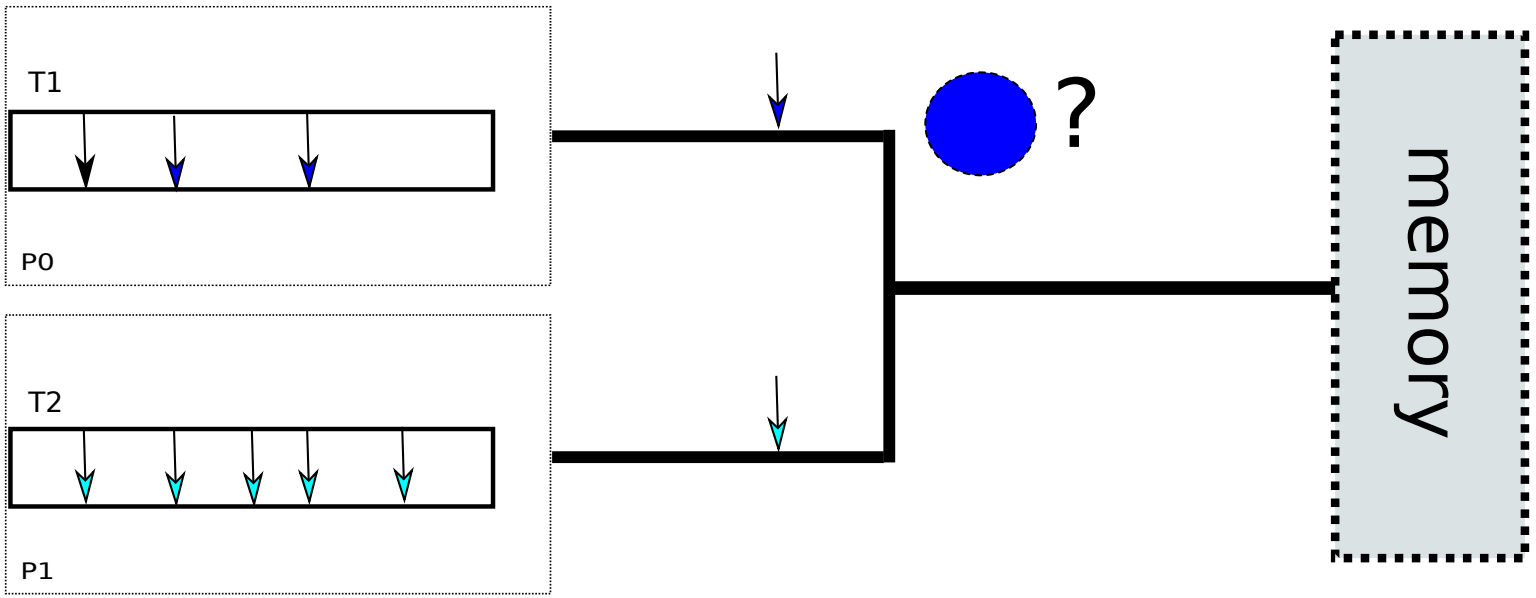
2 cores: TDMA, keeping isolated WCET

how to take the memory access delay into account?

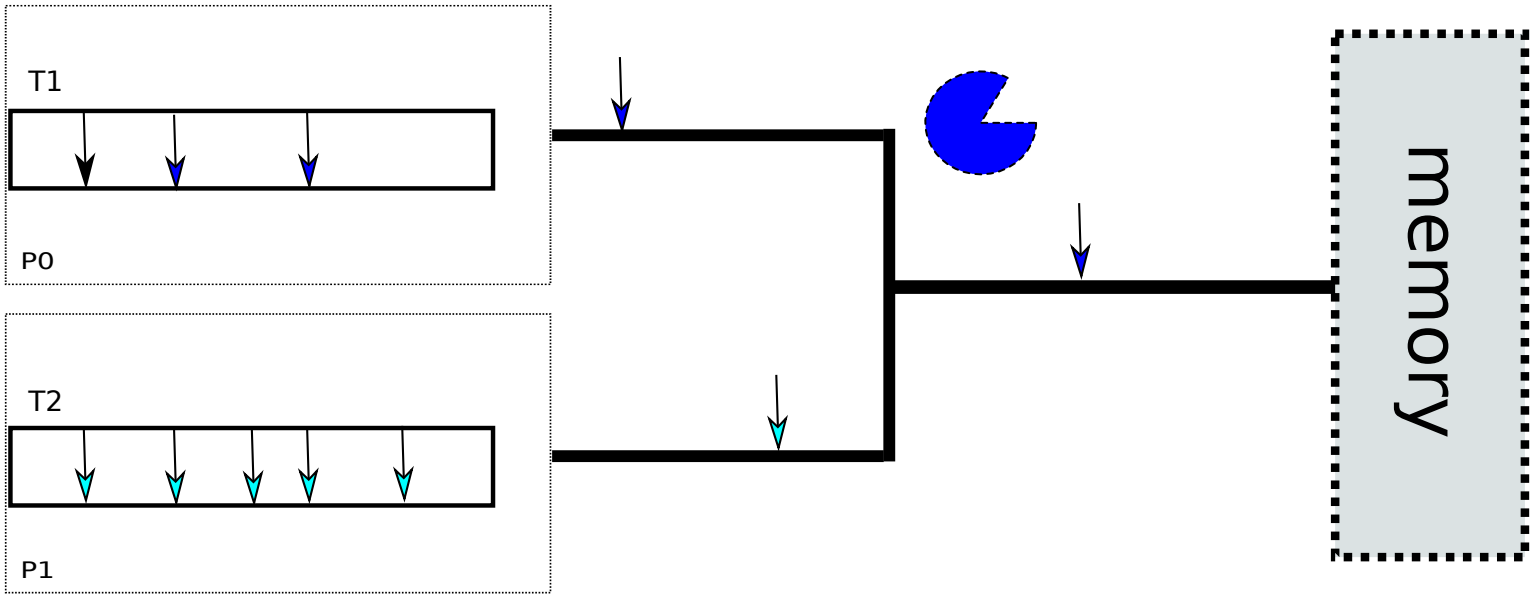
- worst case: $access_{task2} = slotSize$
- n cores, worst case: $access_{task2} = (n-1) \times slotSize$



2 cores: Round Robin



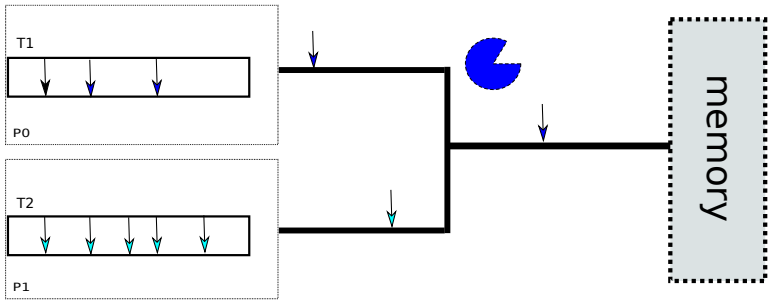
2 cores: Round Robin, global interference anlaysis



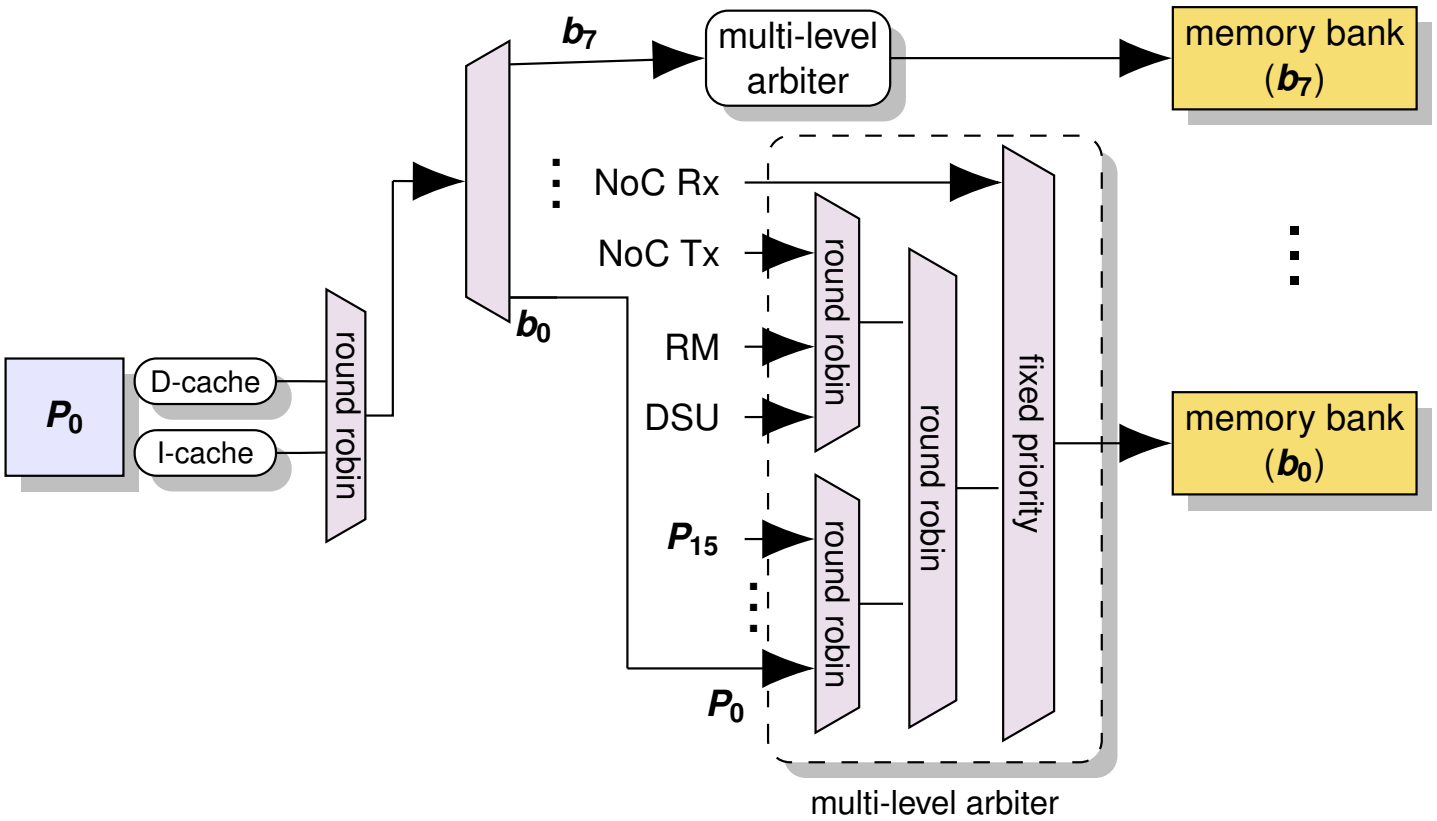
2 cores: Round Robin, global interference anlaysis

how to take the memory access delay into account?

- worst case: $access_{task2} = slotSize$
- n cores, worst case: $access_{task2} = (n-1) \times slotSize$
- more precise: minimum of the worst-case numbers of accesses (WA) of concurrent tasks



Interferences on a many-core platform Example of the MPPA Kalray



Growing analysis complexity

- How to get a precise and scalable analysis?

Outline

What is covered by the state-of-the-art

- Timing analysis (execution time, response time, scheduling, mapping)
- Analysis that takes into account the interference delay due to shared resources
- 110 papers, 11 last years

4 main categories

- Temporal isolation
- Full integration: all in one execution time bound
- Interference delay in the schedulability analysis
- Interference delay as a parameter for scheduling/mapping

[1] Maiza et al., submitted to ACM Computing Surveys
http://www-verimag.imag.fr/TR/TR-2018-9.pdf

Temporal isolation

Main idea

- Isolation = eliminate interference
- Examples: TDMA, phased execution

Pros/Cons

- Pros: partial or no interference
- Cons: low hardware utilization

Full integration

Main idea

- Knowledge on all concurrent tasks
- Estimation of one global inclusive bound
- Example: all possible interleavings of cache memory accesses

Pros/Cons

- Pros: precision
- Cons: very high analysis complexity

Interference delay into schedulability analysis

Main idea

- Interference delay estimation as an independent analysis
- Shared resource: memory and/or bus

Pros/Cons

- Pros: scalability, good precision
- Cons: usually needs to limit the interference scope

Interference delay as a parameter for scheduling/mapping

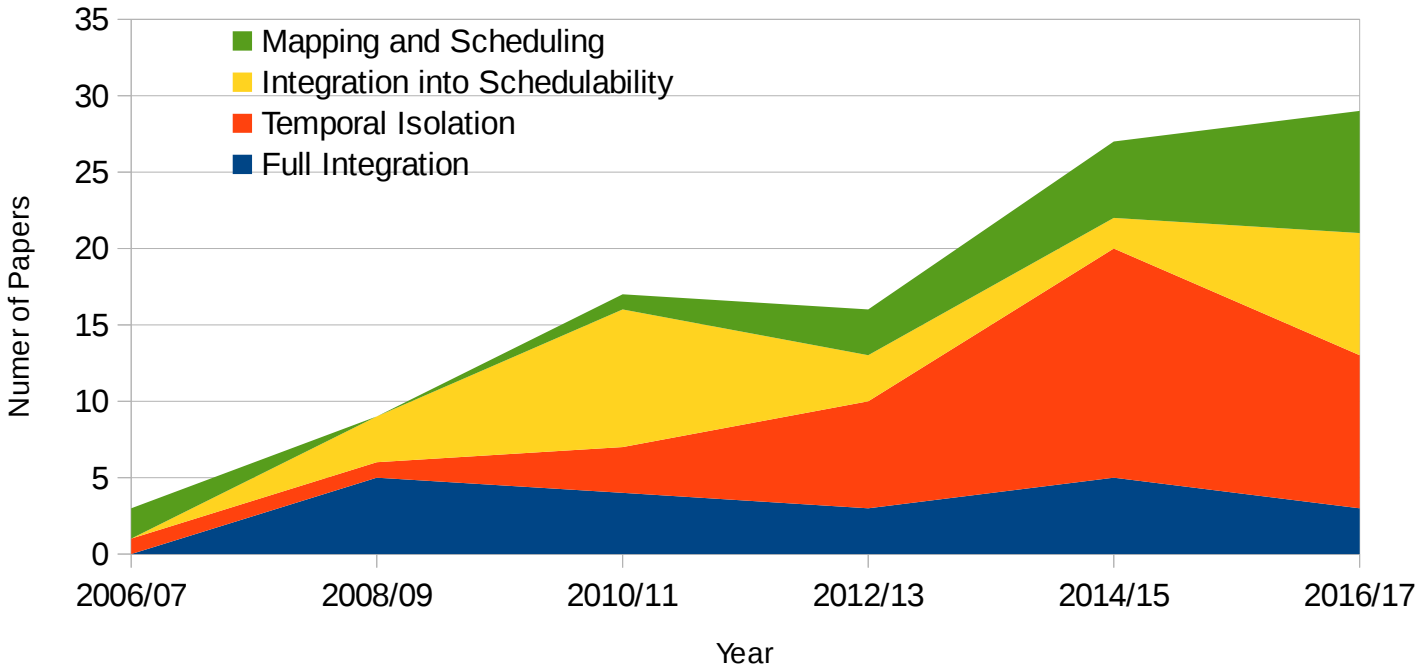
Main idea

- Orthogonal to schedulability
- Example: round-robin slot size

Pros/Cons

- Pros: better use of the platform
- Cons: may lead to isolation/low utilization

SoA timing analysis on multi-core



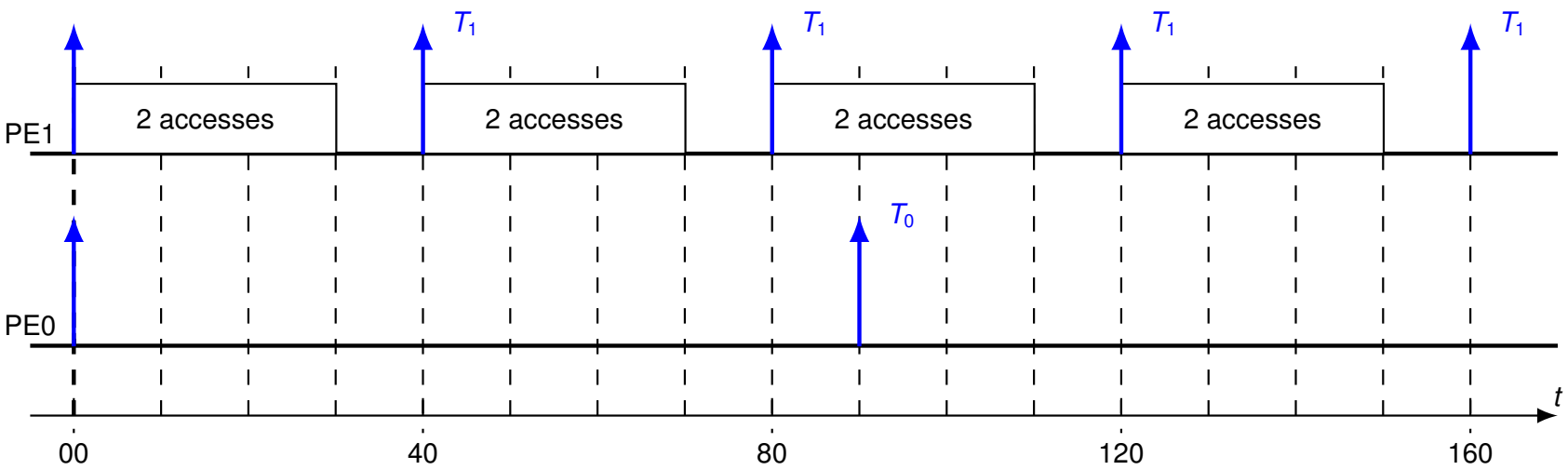
Feedback

- Need of timing compositionality
- Interference analysis scales!

Outline

Multicore Response Time Analysis

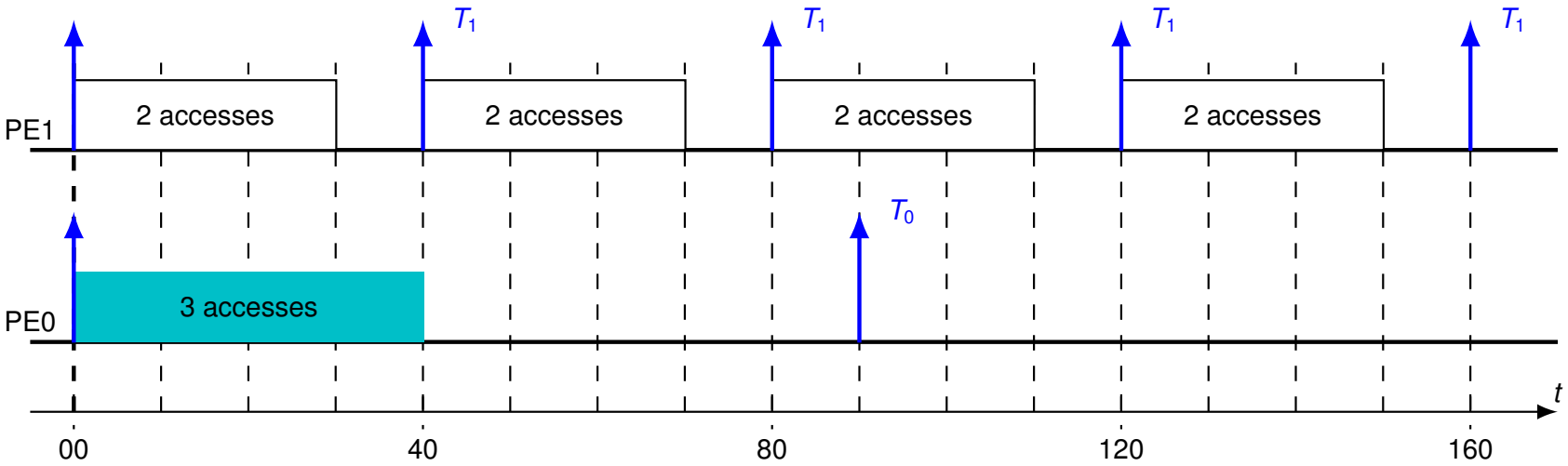
Interference effects into response-time analysis
Example: Fixed Priority bus arbiter, PE1 > PE0
Bus access delay = 10



[1] Davis et al., RTS 2017

Multicore Response Time Analysis

Interference effects into response-time analysis
Example: Fixed Priority bus arbiter, PE1 > PE0
Bus access delay = 10



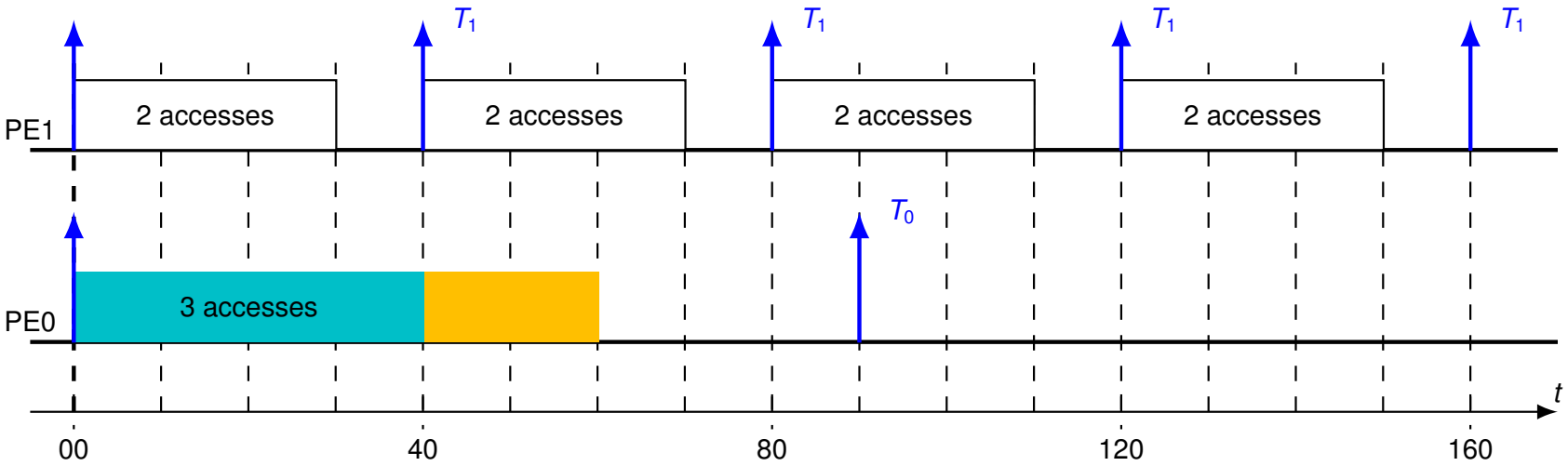
- Task of interest running on $PE0$:

  $R_0 = 10 + 3 \times 10$ (response time in isolation)

[1] Davis et al., RTS 2017

Multicore Response Time Analysis

Interference effects into response-time analysis
Example: Fixed Priority bus arbiter, PE1 > PE0
Bus access delay = 10



- Task of interest running on $PE0$:

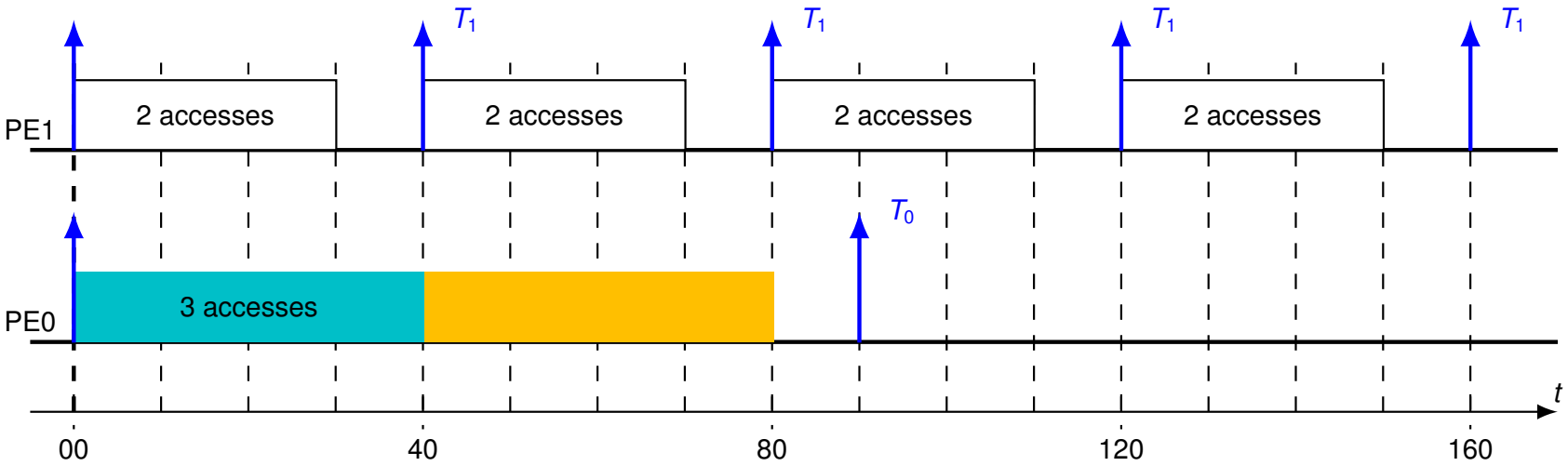
  $R_0 = 10 + 3 \times 10$ (response time in isolation)

  $R_1 = 10 + 3 \times 10 + 2 \times 10 = 60$

[1] Davis et al., RTS 2017

Multicore Response Time Analysis

Interference effects into response-time analysis
Example: Fixed Priority bus arbiter, PE1 > PE0
Bus access delay = 10



- Task of interest running on $PE0$:

  $R_0 = 10 + 3 \times 10$ (response time in isolation)

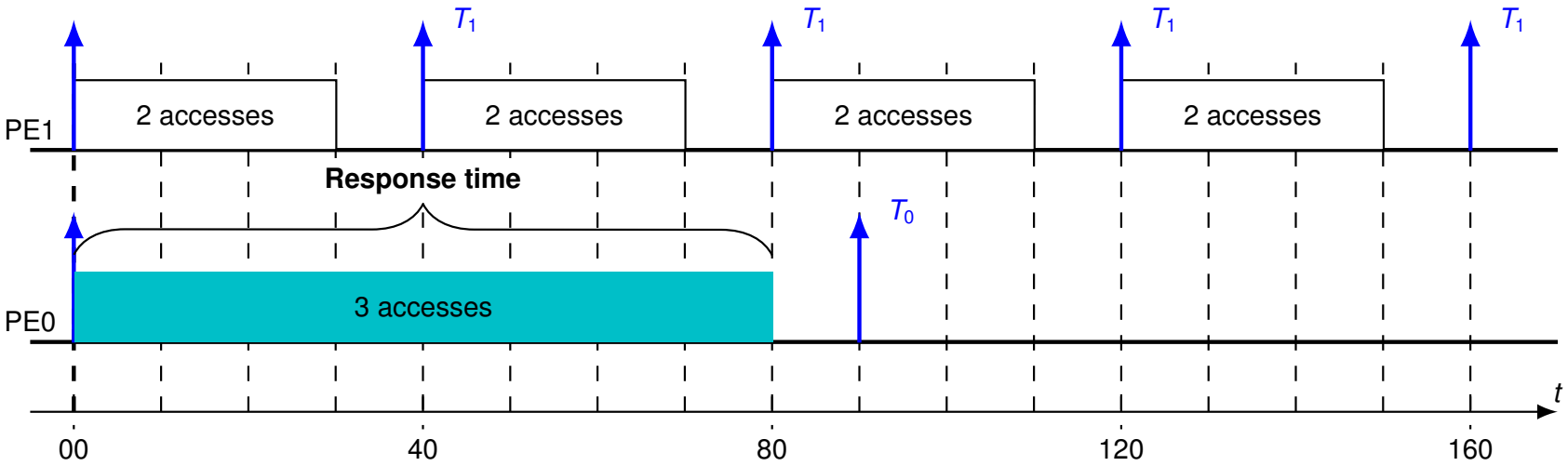
  $R_1 = 10 + 3 \times 10 + 2 \times 10 = 60$

  $R_2 = 10 + 3 \times 10 + 2 \times 10 + 2 \times 10 = 80$

[1] Davis et al., RTS 2017

Multicore Response Time Analysis

Interference effects into response-time analysis
Example: Fixed Priority bus arbiter, PE1 > PE0
Bus access delay = 10



- Task of interest running on *PE*0:

  $R_0 = 10 + 3 \times 10$ (response time in isolation)

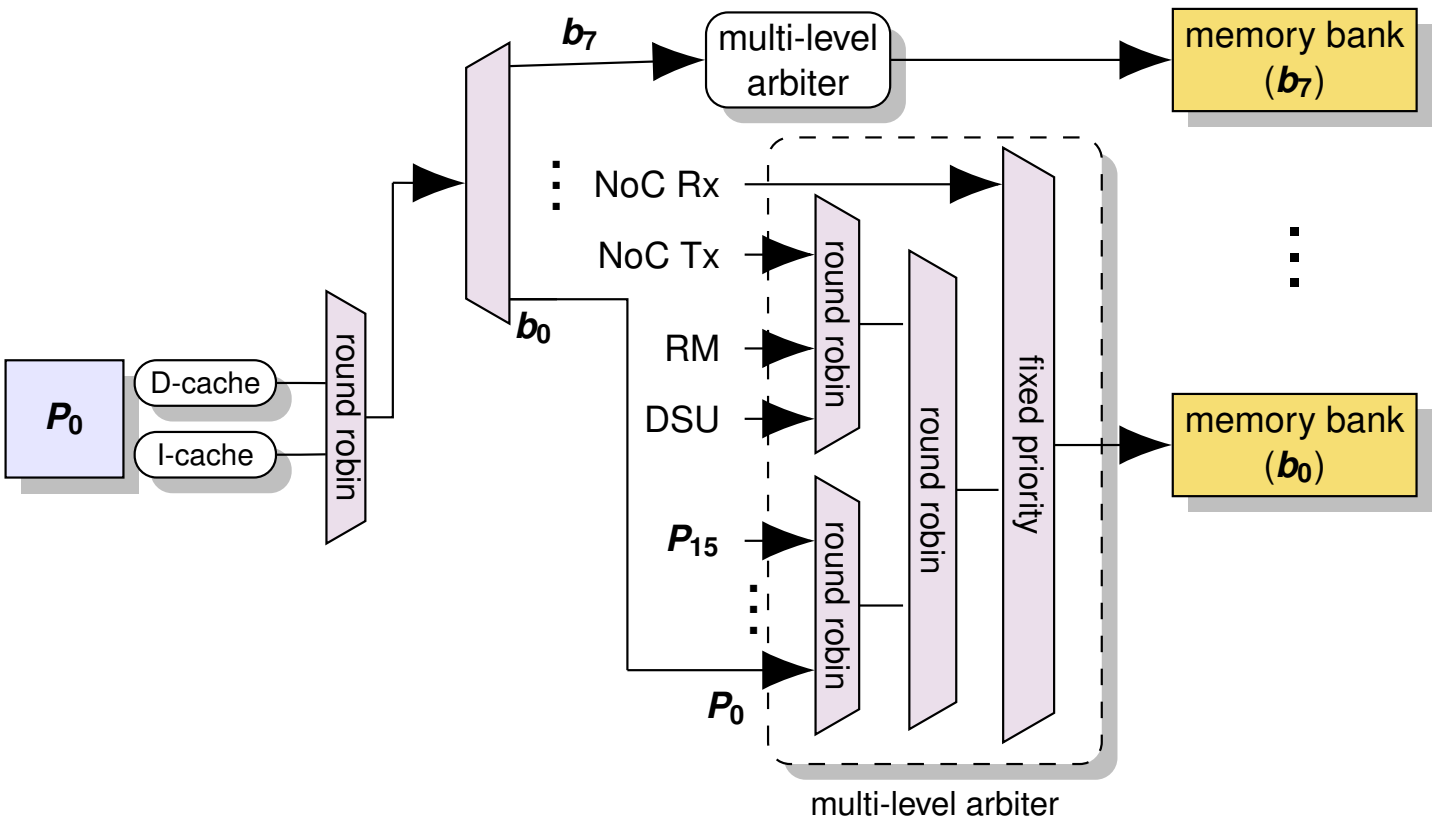
  $R_1 = 10 + 3 \times 10 + 2 \times 10 = 60$

  $R_2 = 10 + 3 \times 10 + 2 \times 10 + 2 \times 10 = 80$

  $R_3 = 10 + 3 \times 10 + 2 \times 10 + 2 \times 10 + 0 = 80$ (fixed-point)

[1] Davis et al., RTS 2017

Application to a cluster MPPA2 kalray



Too many delays to add:
Combine a precise interference analysis with a good implementation/configuration

Configuration/implementation for a precise interference delay

Context: Hard real-time system with guaranteed execution

Better knowledge on communications: who, what and when

- Data-flow application: precedence constraints implemented time-triggered
- (Phased application)
- Banked memory :
  One core-one bank, interference only on communication
- Proof-of-concept: bare metal implementation Lustre/SCADE

Interference analysis

Main idea

Input = Isolated WCET + WA + Initial scheduling/mapping

1 Estimate current interference

1.1 For all tasks that interfere on the current scheduling

1.2 Add this interference delay to the initial WCET

2.1 Reajust release dates and restart [1] with the new scheduling

[1] Rihani et al., RTNS 2016

Interference analysis

Main idea

Input = Isolated WCET + WA + Initial scheduling/mapping

1. Estimate current interference
   - 1.1 For all tasks that interfere on the current scheduling
   - 1.2 Add this interference delay to the initial WCET
2. If phase [1] differs from previous step
   - 2.1 Reajust release dates and restart [1] with the new scheduling

[1] Rihani et al., RTNS 2016

Interference analysis

Main idea

Input = Isolated WCET + WA + Initial scheduling/mapping

1. Estimate current interference
   - 1.1 For all tasks that interfere on the current scheduling
   - 1.2 Add this interference delay to the initial WCET
2. If phase [1] differs from previous step
   - 2.1 Reajust release dates and restart [1] with the new scheduling
3. Check final schedulability
   - 3.1 If the application is schedulable: the current schedule is guaranteed
   - 3.2 Otherwise, go back to the initial scheduling/mapping step

[1] Rihani et al., RTNS 2016

Conclusion

Interference analysis is possible!
There is a huge need of compositionality verification.

Conclusion

Interference analysis is possible!
There is a huge need of compositionality verification.

Future work:

- Proof-of-concept implementation inter-cluster
- Extend to other platforms

CAPACITES: workflow proof of concept