A Generic and Compositional Framework for Multicore Response Time Analysis

Sebastian Altmeyer, Robert I. Davis
Leandro Indrusiak, Claire Maiza
Vincent Nelis, Jan Reineke

RTNS 2015

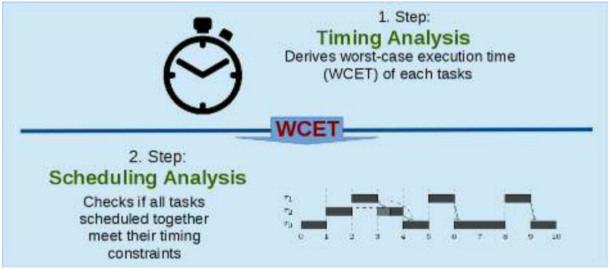
Motivation and Context

Multicore Response Time Analysis

Evaluation

Conclusions

Multicore Timing Verification: Traditional Approach



1. Step:
Timing Analysis
Derives worst-case execution time (WCET) of each tasks

WCET

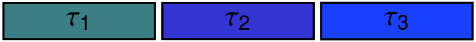
2. Step:
Scheduling Analysis
Checks if all tasks scheduled together meet their timing constraints

Implicit assumptions:

- Tasks can be analyzed independently
- WCETs are context independent

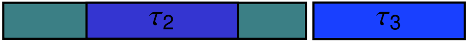
Problems with context-independent WCETs

Non-pre-emptive uniprocessor:



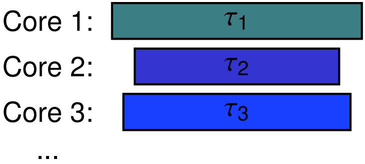
works well

Pre-emptive uniprocessor:

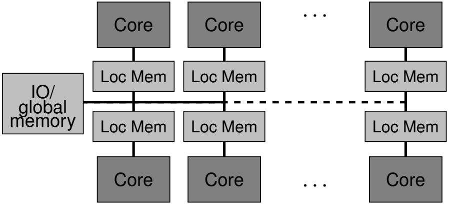


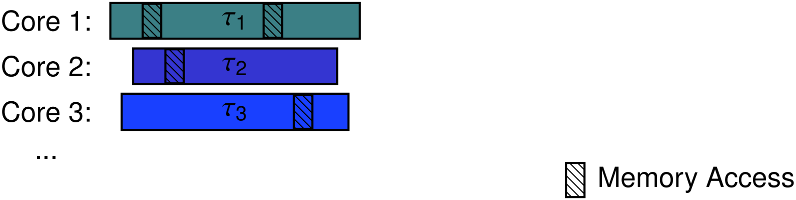
works relatively well

Multicore:

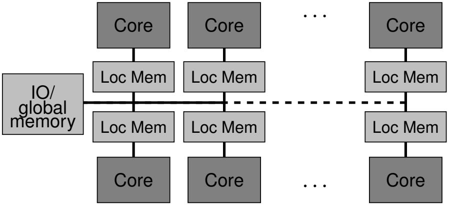


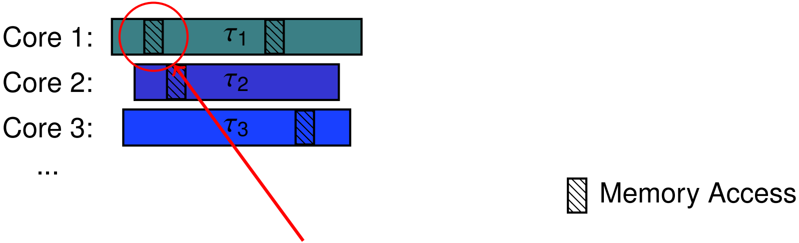
Problems with context-independent WCETs





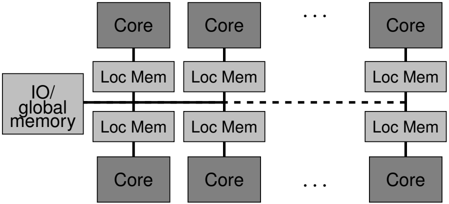
Problems with context-independent WCETs

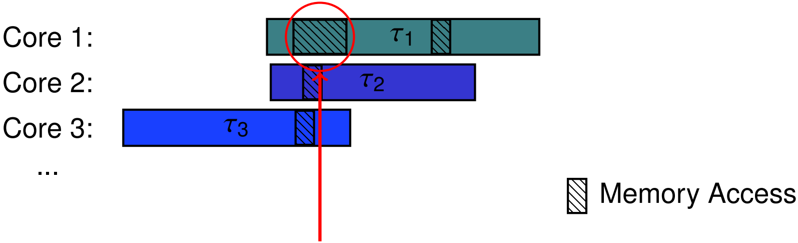




What is the context-independent worst-case delay?

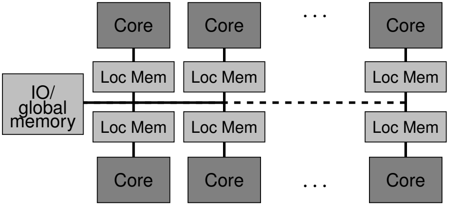
Problems with context-independent WCETs

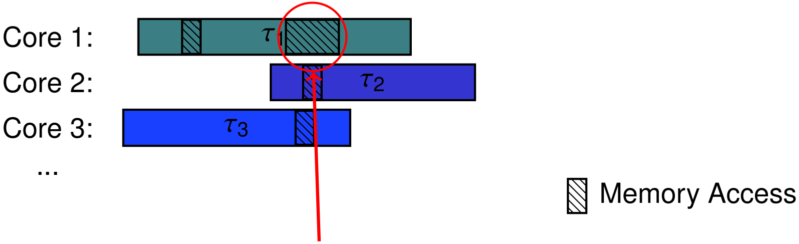




What is the context-independent worst-case delay?

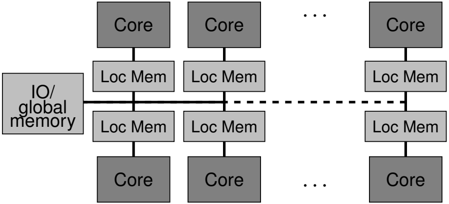
Problems with context-independent WCETs

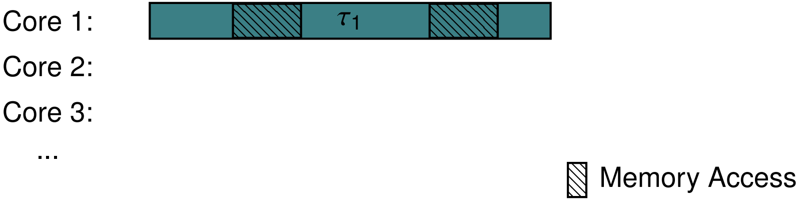




What is the context-independent worst-case delay?

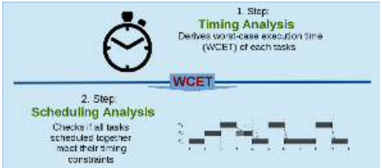
Problems with context-independent WCETs



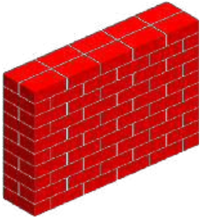


⇒ Highly inflated execution time bounds
(multicore may perform worse than single cores)

Multicore Timing Verification: Isolation

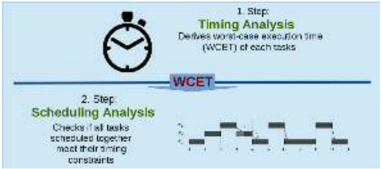

1. Step:
Timing Analysis
Derives worst-case execution time (WCET) of each tasks
WCET
2. Step:
Scheduling Analysis
Checks if all tasks scheduled together meet their timing constraints

+

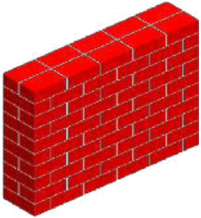


Isolate tasks from each other, remove interference

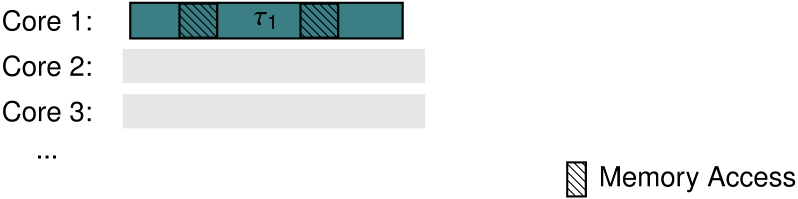
Multicore Timing Verification: Isolation


1. Step:
Timing Analysis
Derives worst-case execution time (WCET) of each tasks
WCET
2. Step:
Scheduling Analysis
Checks if all tasks scheduled together meet their timing constraints

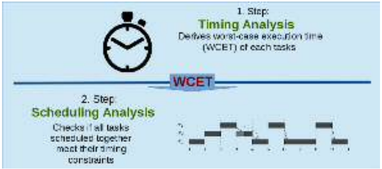
+



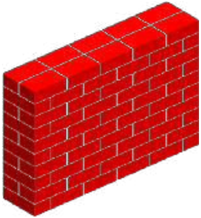
Isolate tasks from each other, remove interference



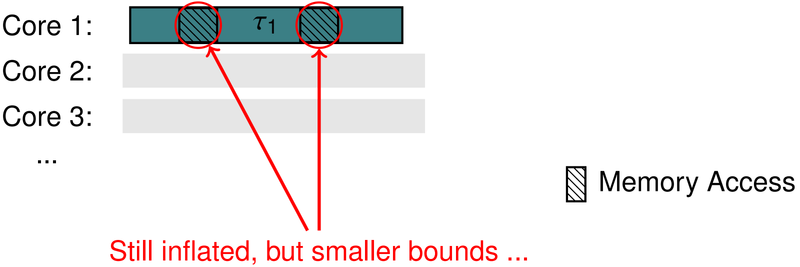
Multicore Timing Verification: Isolation


1. Step:
Timing Analysis
Derives worst-case execution time (WCET) of each tasks
WCET
2. Step:
Scheduling Analysis
Checks if all tasks scheduled together meet their timing constraints

+

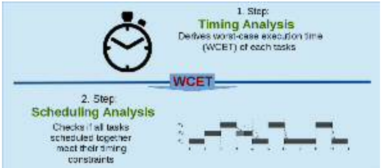


Isolate tasks from each other, remove interference



Still inflated, but smaller bounds ...

Multicore Timing Verification: Isolation

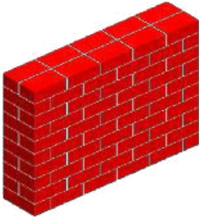


1. Step:
Timing Analysis
Derives worst-case execution time (WCET) of each tasks

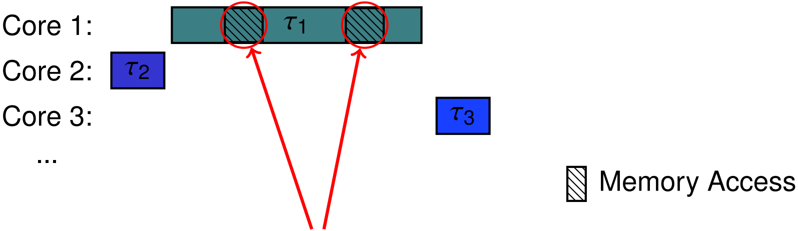
WCET

2. Step
Scheduling Analysis
Checks if all tasks scheduled together meet their timing constraints

+



Isolate tasks from each other, remove interference



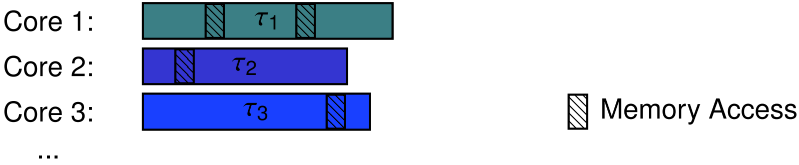
Pays for interference, even if there is none

Multicore Timing Verification: Fully Integrated Approach

- One, all-combining analysis
- Analyze exact interleavings

Multicore Timing Verification: Fully Integrated Approach

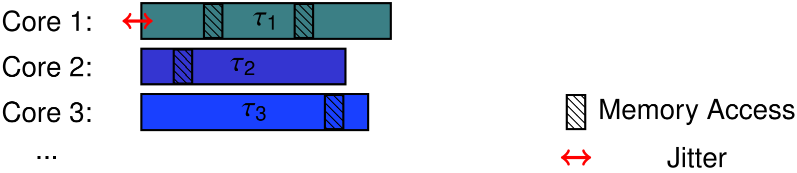
- One, all-combining analysis
- Analyze exact interleavings



Promises best precision

Multicore Timing Verification: Fully Integrated Approach

- One, all-combining analysis
- Analyze exact interleavings



Core 1: ↔ τ_1

Core 2: τ_2

Core 3: τ_3

...

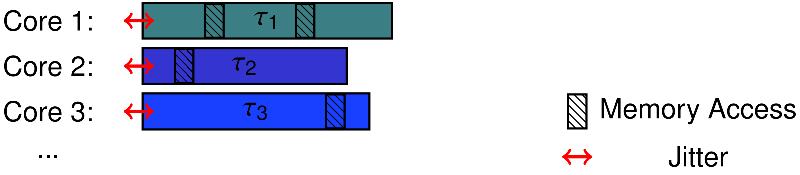
▨ Memory Access

↔ Jitter

Promises best precision

Multicore Timing Verification: Fully Integrated Approach

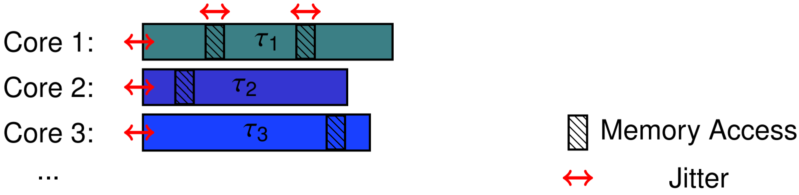
- One, all-combining analysis
- Analyze exact interleavings



Promises best precision

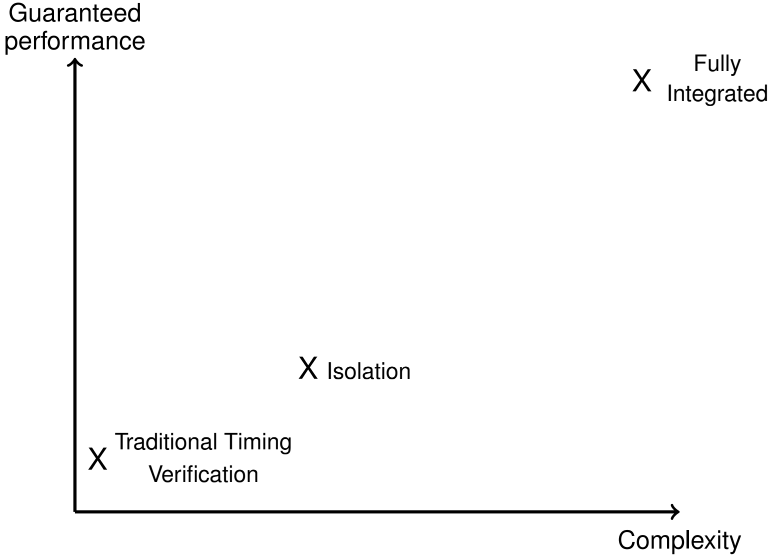
Multicore Timing Verification: Fully Integrated Approach

- One, all-combining analysis
- Analyze exact interleavings

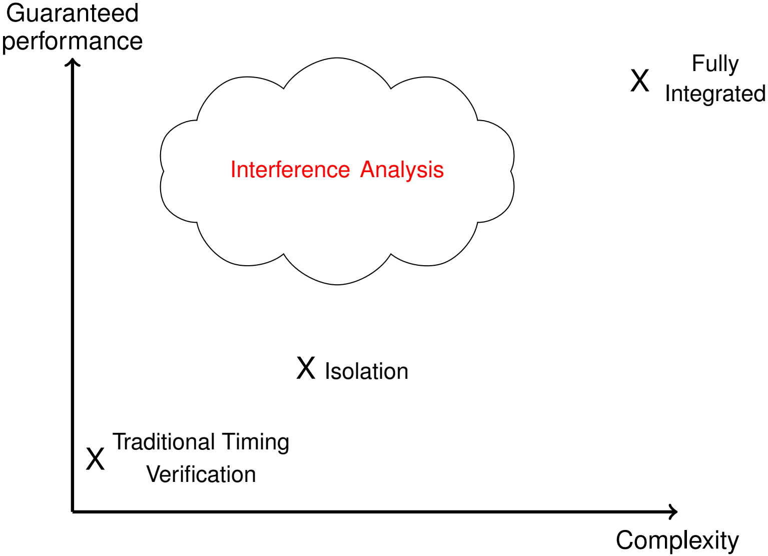


Promises best precision, but very high complexity. Too high?

Multicore Timing Verification: Comparisons

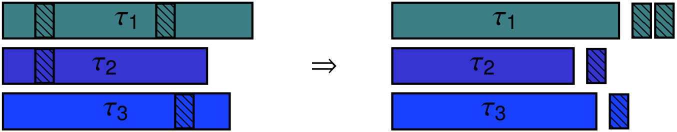


Multicore Timing Verification: Comparisons



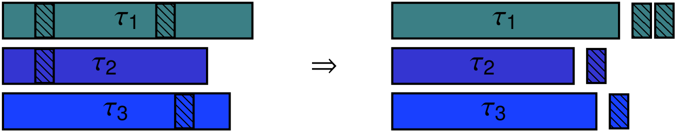
Interference Analysis

Decompose

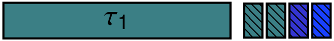


Interference Analysis

Decompose

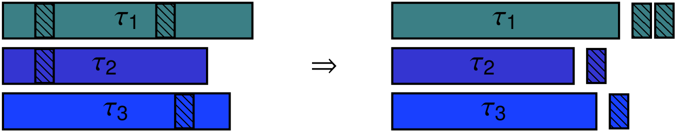


and re-assemble

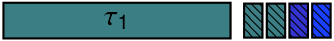


Interference Analysis

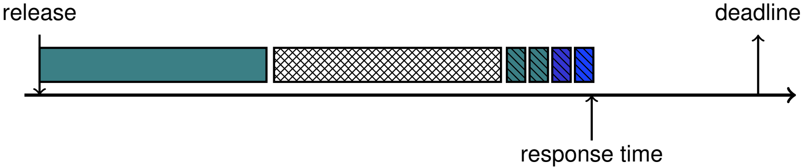
Decompose



and re-assemble



over the response time:



Motivation and Context

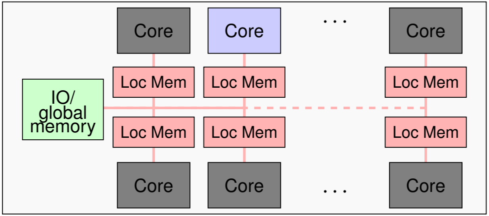
Multicore Response Time Analysis

Evaluation

Conclusions

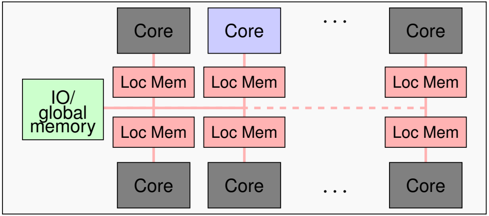
Analysis Framework

Multicore architecture with shared components:



Analysis Framework

Multicore architecture with shared components:



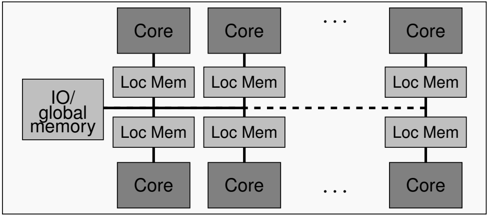
What is the impact of each component on a task's response time:

$R_i =$ Delay on the core $+$

Delay on the bus/local memory $+$

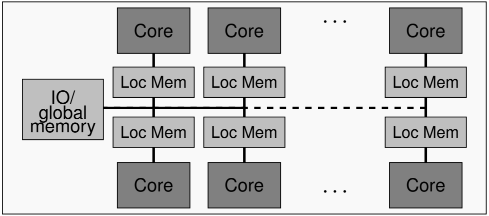
Delay on the global memory

Targeted Processor Model



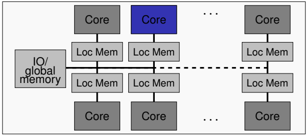
- ℓ identical cores $\{P_1, \ldots, P_\ell\}$,
- fixed-priority pre-emptive scheduling, partitioned tasks
- one shared bus
- local memories
- a global memory (DRAM)

Impact of the Multicore Components



Core	How long does it take to execute a task?
Local Memory	How many memory requests go to the bus?
Bus	How many competing accesses can occur?
Global Memory	How many DRAM refreshes can occur?

Core: Processor Demand



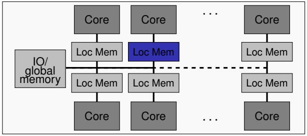
How long does it take to execute a task?

Provides:

- processor demand PD of a task
  i.e., execution time without any interference, memory delays, etc.

Local Memory: Memory Demand

$$\mathrm{MEM}(o) = (\mathrm{MD}, \overline{\mathrm{UCB}}, \mathrm{ECB})$$



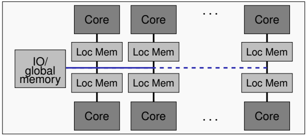
How many memory requests go to the bus?

Provides:

- memory demand MD, i.e., $\#$ bus accesses
- metrics for the pre-emptions costs ($\overline{\mathrm{UCB}}$,ECB)

Bus: Competing Accesses

$\mathrm{BUS}(i, x, t)$



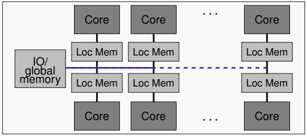
How many competing accesses can occur?

Provides:

- #bus accesses that delay task τ_i on processor P_x during time t

Bus: Competing Accesses

$\mathrm{BUS}(i, x, t)$



How many competing accesses can occur?

Provides:

- #bus accesses that delay task τ_i on processor P_x during time t

Uses

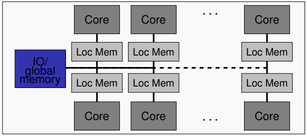
$S(t)$ #competing accesses on same core

$A(t)$ #competing accesses on all other cores

Derived using output of the memory function: MD, UCBs and ECBs

DRAM: Number of DRAM refreshes

DRAM(t, m)

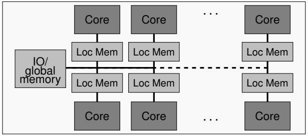


How many DRAM refreshes can occur?

Provides:

- #DRAM refreshes during time t with up to m memory accesses

Which components can we model so far?



Core: any timing-compositional core

Local Mem.: Scratchpads, LRU/DM caches, partitioned caches, uncached systems (all for instruction and data)

Bus: Fixed-Priority Bus, TDMA, Round-Robin, Processor Priority

DRAM: burst refreshes, distributed refreshes

and any combination thereof.

From Component Model to Interferences

$$I^C(i, x, R_i)$$

Interference/Delay of component C during the response time R_i of task τ_i executing on processor P_x

From Component Model to Interferences

$$I^{C}(i, x, R_i)$$

Interference/Delay of component C during the response time R_i of task τ_i executing on processor P_x

$$I^{\text{PROC}}(i, x, t) = \sum_{j \in \Gamma_x \wedge j \in \text{hp}(i)} \left\lceil \frac{t}{T_j} \right\rceil \text{PD}_j$$

From Component Model to Interferences

$$I^{C}(i, x, R_i)$$

Interference/Delay of component C during the response time R_i of task τ_i executing on processor P_x

$$I^{\text{PROC}}(i, x, t) = \sum_{j \in \Gamma_x \wedge j \in \text{hp}(i)} \left\lceil \frac{t}{T_j} \right\rceil \text{PD}_j$$

$$I^{\text{BUS}}(i, x, t) = \text{BUS}(i, x, t) \cdot d_{\text{main}}$$

where d_{main} is the bus access latency to the global memory.

From Component Model to Interferences

$$I^{C}(i, x, R_i)$$

Interference/Delay of component C during the response time R_i of task τ_i executing on processor P_x

$$I^{\mathrm{PROC}}(i, x, t) = \sum_{j \in \Gamma_x \wedge j \in \mathrm{hp}(i)} \left\lceil \frac{t}{T_j} \right\rceil \mathrm{PD}_j$$

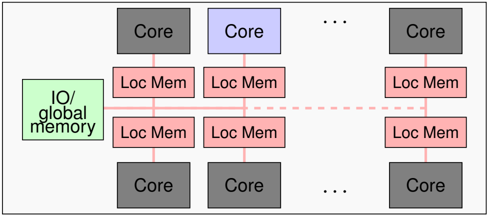
$$I^{\mathrm{BUS}}(i, x, t) = \mathrm{BUS}(i, x, t) \cdot d_{\mathrm{main}}$$

where d_{main} is the bus access latency to the global memory.

$$I^{\mathrm{DRAM}}(i, x, t) = \mathrm{DRAM}(t, \mathrm{BUS}((i, x, t)) \cdot d_{\mathrm{refresh}}$$

where d_{refresh} is the refresh latency.

Multicore Response Time Analysis



$$R_i = \boxed{\mathrm{PD}_i + I^{\mathrm{PROC}}(i, x, R_i)} + \boxed{I^{\mathrm{BUS}}(i, x, R_i)} + \boxed{I^{\mathrm{DRAM}}(i, x, R_i)}$$

(solved via fixed-point iteration)

Task set feasible, if:

$$\forall i : R_i \leq D_i$$

Motivation and Context

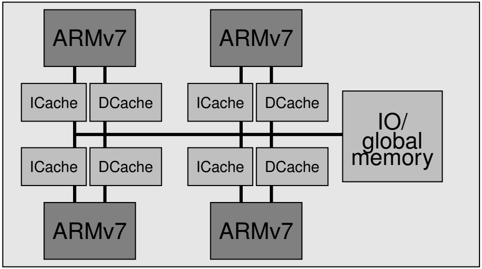
Multicore Response Time Analysis

Evaluation

Conclusions

Proof-of-Concept Instantiation

- System based on the ARM Cortex A5:



- 4 cores, separate instruction and data caches, FP/FIFO/TDMA bus, and distributed DRAM controller.
- Compared different configurations for a large number of randomly generated task sets

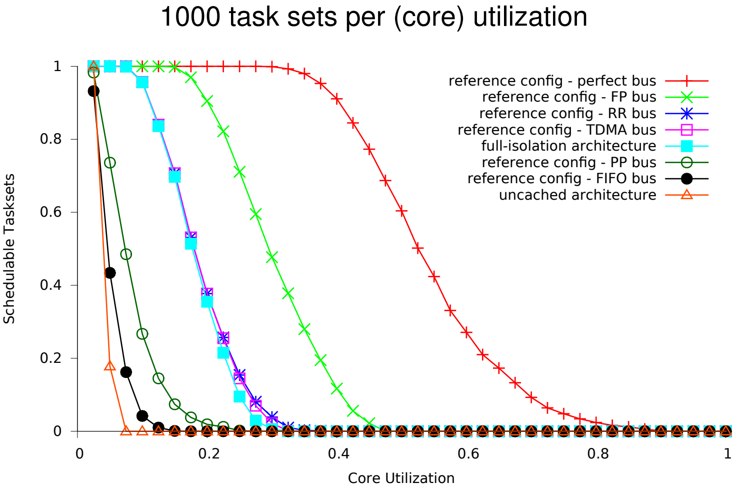
Randomly generated task sets

Task set parameters

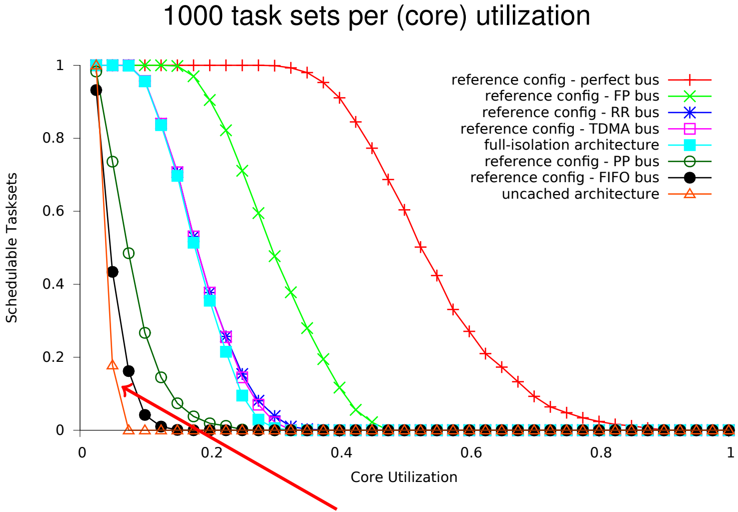
- 32 tasks in total, with 8 tasks per core, uniform core utilization
- each task was randomly assigned a task from Mälardalen benchmark suite (see table)
- implicit deadlines
- priorities in deadline monotonic order.

Name	# Instr. (PD)	Read/Write	MD	UCB	ECB
adpcm_enc	628795	124168	38729	155	346
bsort100	272715	1305613	25464	31	135
compress	8793	3358	993	74	174
fdct	5923	3098	1088	67	193
lms	3023813	373874	120821	150	276
nsichneu	8648	4841	1582	397	589
⋮	⋮	⋮	⋮	⋮	⋮

Results: Core Utilization

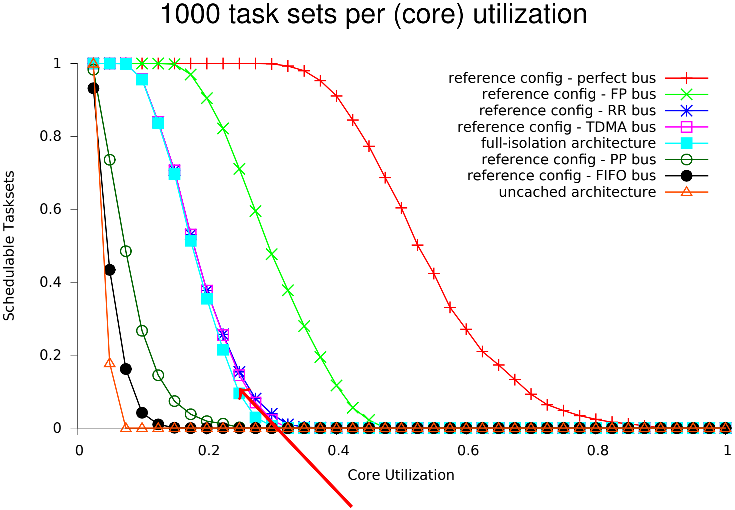


Results: Core Utilization



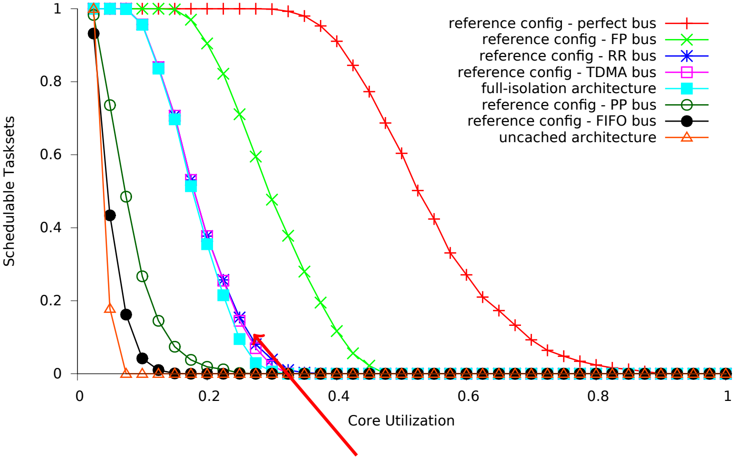
without local caches: worst performance

Results: Core Utilization



full isolation (TDMA bus + cache partitioning)

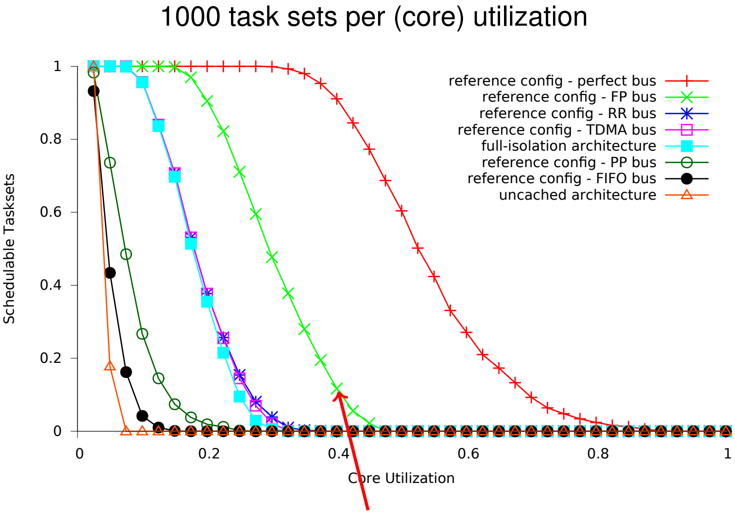
Results: Core Utilization



1000 task sets per (core) utilization

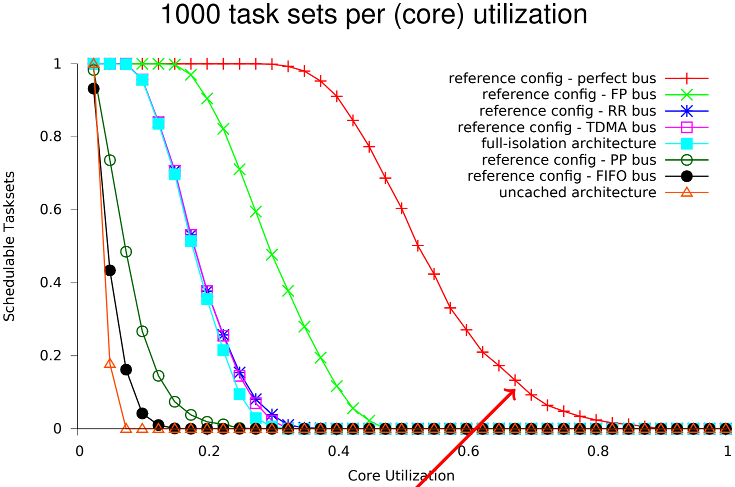
round-robin/TDMA bus

Results: Core Utilization



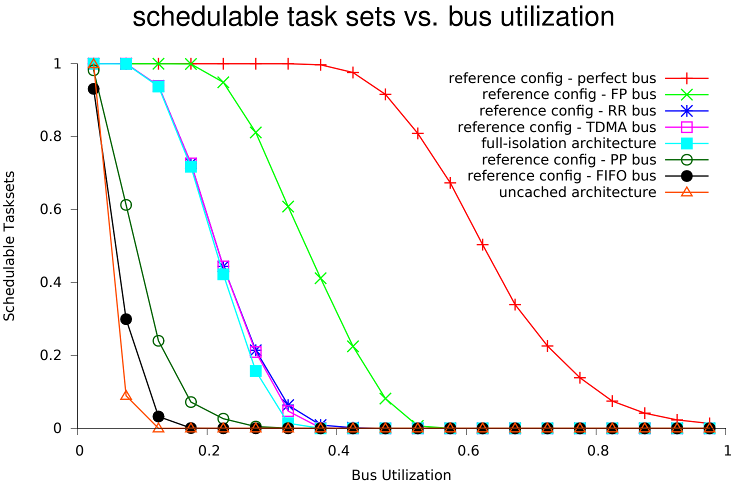
Fixed-Priority Bus: work-conserving, best performance

Results: Core Utilization



perfect bus: theoretical upper bound on the performance

Results: Bus Utilization



better results: bus/global memory is the bottleneck

Motivation and Context

Multicore Response Time Analysis

Evaluation

Conclusions

Conclusions

Multicore Response Time Analysis framework

- based on interference modelling
- directly aiming at response time
- parametric in the hardware configuration
- extensible to other sources of interference
- but ignores overlapping

Proof-Of-Concept Implementation

- based on ARM Cortex A5
- temporal isolation not needed
- promising results for work-conserving bus policies

Questions?