PROXIMA

Software-enforced Interconnect Arbitration for COTS Multicores

M. Ziccardi, A. Cornaglia, E. Mezzetti, and T. Vardanega
University of Padua - Italy

15th International Workshop on Worst-Case Execution Time Analysis (WCET 2015)
Lund, July 7th, 2015

This project and the research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7 / 2007-2013] under grant agreement 611085

www.proxima-project.eu

Introduction

Multicore Challenge

- Applications on distinct cores may contend for accessing shared HW resources incurring *inter-core interference*
- Interference cannot be disregarded as it largely affects the worst-case execution time (WCET) behavior

Research Paths

1. Precisely analyze the timing interference
   - Complex on complex architectures
2. Control the sources of inter-core interference in the system to make it more analyzable
   - *Resource partitioning*

Motivation

Shared interconnects (BUS, crossbar, …) are one of the main sources of interference

Interconnects time partitioning can be achieved via dedicated hardware arbitration policies (e.g. TDMA)

Hardware policies facilitating WCET analysis are not always available

Arbitration policies of COTS multicores focus on performance rather than analyzability

Might be challenging to analyze and might lead to pessimistic bounds

Our idea: software-enforced TDMA arbitration of interconnects

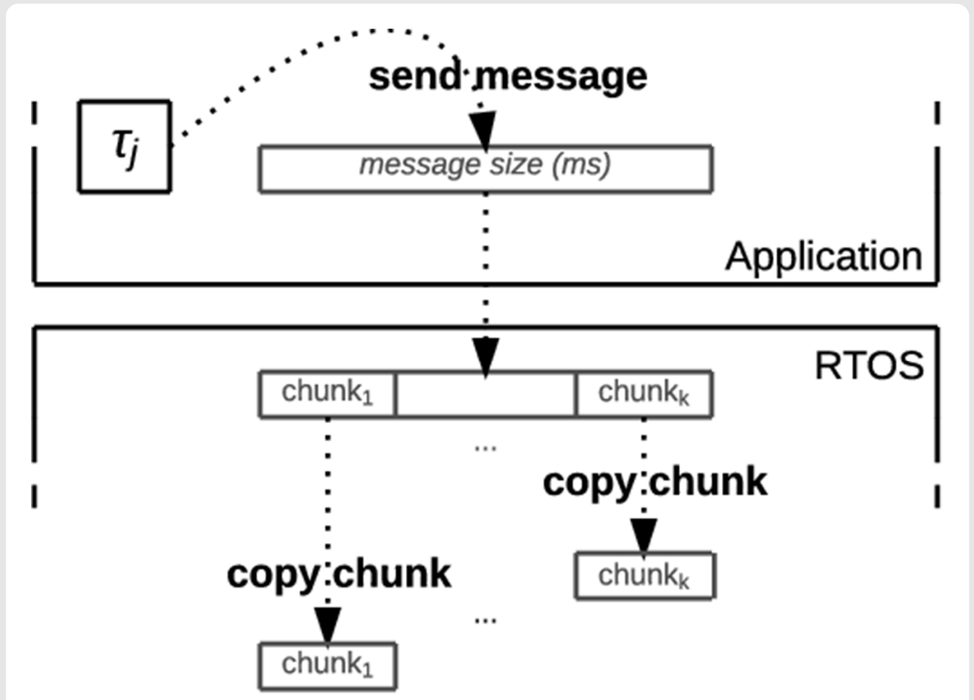
- To achieve partitioning
- No hardware-support needed
- TDMA and Bandwidth Reservation TDMA

System Model

- *n* cores C_0, … C_{n-1}
- Partitioned system
  - Each application is statically assigned to a core
- Each core has its local memory (e.g. scratchpad)
  - To store applications data and instructions
- Communication across tasks is only allowed through the RTOS (ARINC-653, AUTOSAR)
  - *receive_message*
    - Copy a message from a shared channel to local memory
  - *send_message*
    - Copy a message from local memory to a shared channel
- Cores share a time source (*TS*)

Software-enforced TDMA (1)

- Time is divided into frames (of size *fs*)
  - Within each frame *n* slots (of size *ss*), one for each core
  - Messages are split into chunks
  - Each chunk is sent separately and can be transferred within a slot



- Slot size must be such that a chunk transfer can start and terminate within it
  - Takes into account the cost for software arbitration

Software-enforced TDMA (2)

- At any time t_{req} the start time of the current frame (*fb*) is

$$fb(t_{req}) = fs \cdot \lfloor \frac{t_{req}}{fs} \rfloor$$

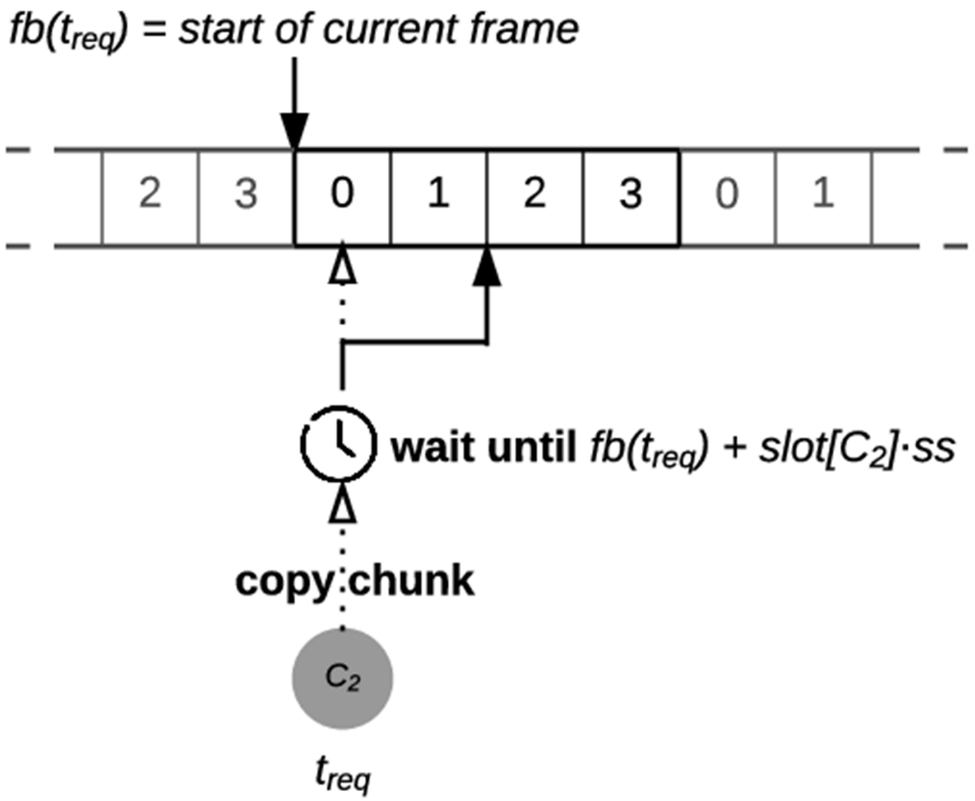
- We use an array (*slot*) to remember the slot in the frame allocated to each core C_i
- The beginning of the corresponding slot (*sb*) for a chunk request issued at time t_{req} by C_i is

$$\mathrm{sb}(C_i, t_{req}) = \begin{cases} fb(t_{req}) + slot[C_i] \cdot ss & \textbf{if } t_{req} \leq fb(t_{req}) + slot[C_i] \cdot ss \\ fb(t_{req}) + fs + slot[C_i] \cdot ss & \textbf{otherwise} \end{cases}$$

Software-enforced TDMA (3)

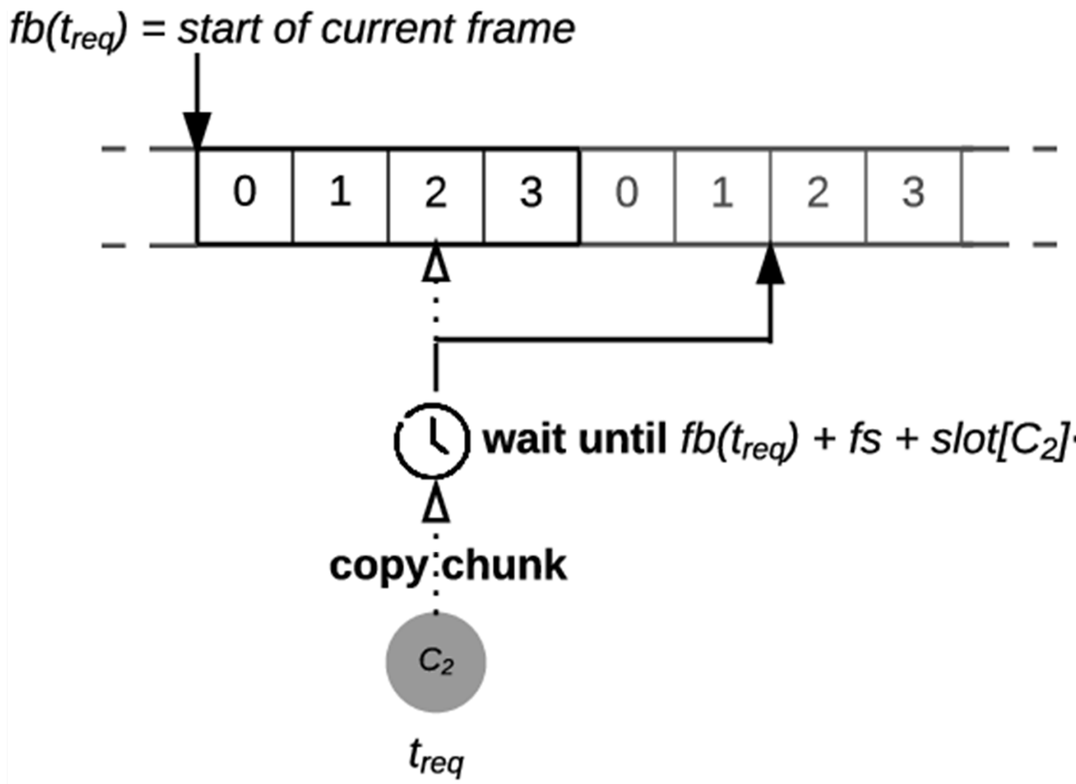
Chunk fit in current frame

$$t_{req} \leq fb(t_{req}) + slot[C_i] \cdot ss$$



Chunk delayed to next frame

$$t_{req} > fb(t_{req}) + slot[C_i] \cdot ss$$



Bandwidth Reservation TDMA (1)

- A variation over classic TDMA
  - Capable of reserving more slots to selected cores
    - To provide higher bandwidth guarantees
  - To meet mixed-criticality application requirements
- Frames are divided into *m* slots (with $m > n$)
  - Each core is associated to one slot
  - The remaining $m - n$ slots can be used to reserve more bandwidth to selected contenders
- We use an array *core* to remember which core is assigned to each slot *i*

Bandwidth Reservation TDMA (2)

The beginning of the corresponding slot (*sb*) for a chunk request issued at time t_{req} by C_i is

If a slot *first* for C_j that starts after t_{req} exists in the current frame

$$sb(C_i, t_{req}) = fb(t_{req}) + first \cdot ss$$

If a slot *first* for C_j that starts after t_{req} does not exist in the current frame. *First* is the first slot for C_j in the next frame

$$sb(C_i, t_{req}) = fb(t_{req}) + fs + first \cdot ss$$

Bandwidth Reservation TDMA (2)

The beginning of the corresponding slot (*sb*) for a chunk request issued at time t_{req} by C_i is

$$if\ \exists\ first:\ first = \min_{j \in [0,m-1]} \{core[j] = C_i\ and\ t_{req} \leq fb(t_{req}) + j \cdot ss\}$$

$$sb(C_i, t_{req}) = fb(t_{req}) + first \cdot ss$$

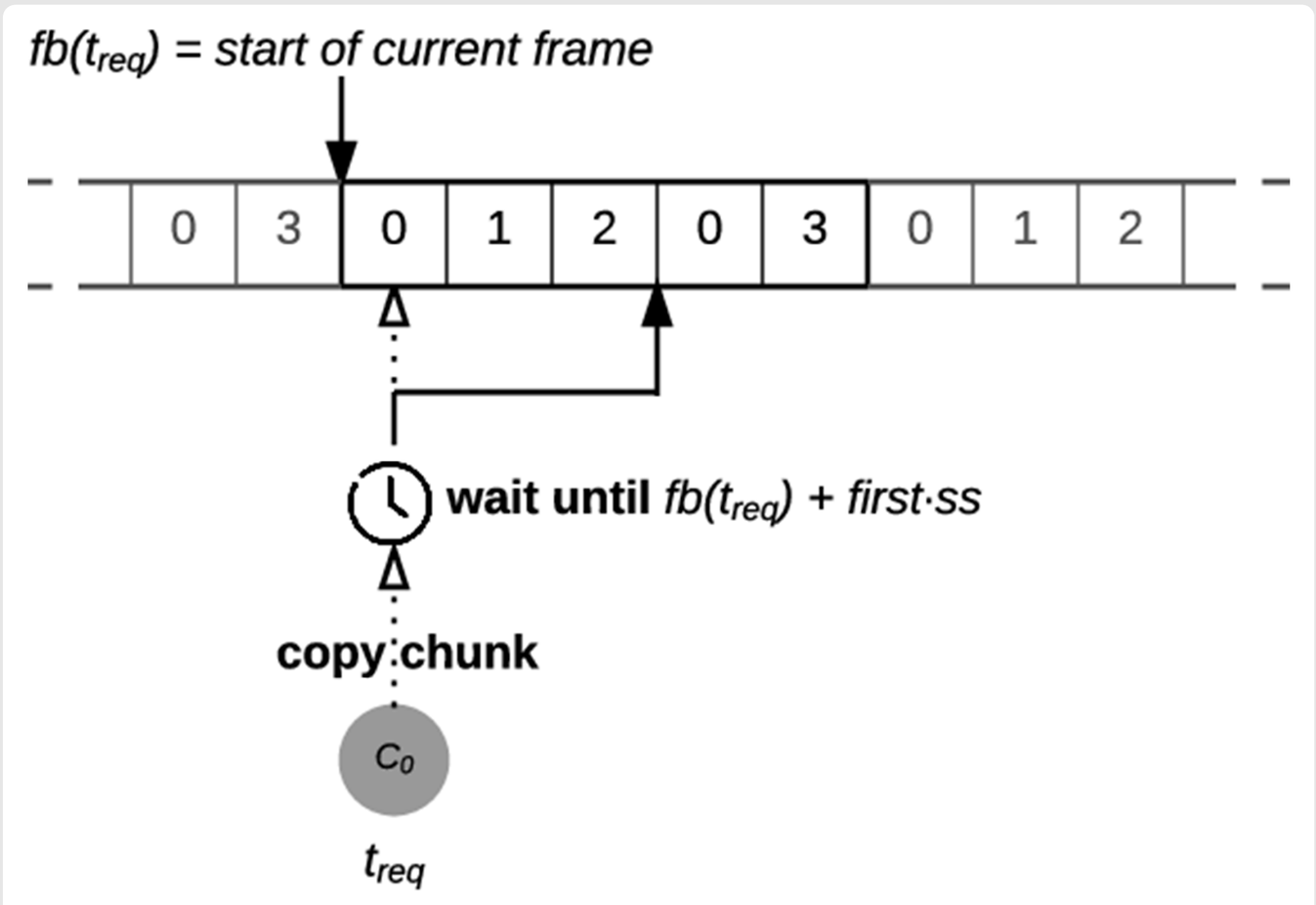
$$Otherwise\ (first = \min_{j \in [0,m-1]} \{core[j] = C_i\})$$

$$sb(C_i, t_{req}) = fb(t_{req}) + fs + first \cdot ss$$

Bandwidth Reservation TDMA (3)

If a slot *first* for C_j that starts after t_{req} exists in the current frame

$$\exists\, first:\ first = \min_{j\in[0,m-1]}\{core[j] = C_i\ and\ t_{req} \le fb(t_{req}) + j \cdot ss\}$$



Experimental Setup

Hardware

- Infineon AURIX TC277TU
- 3 cores connected through a Shared Resource Interconnect (SRI)
  - Connects all cores to the Local Memory Unit (LMU) that controls a 32KB SRAM
  - Divides transactions into request and data phases that can be pipelined
- Each core has local data and instruction scratchpads

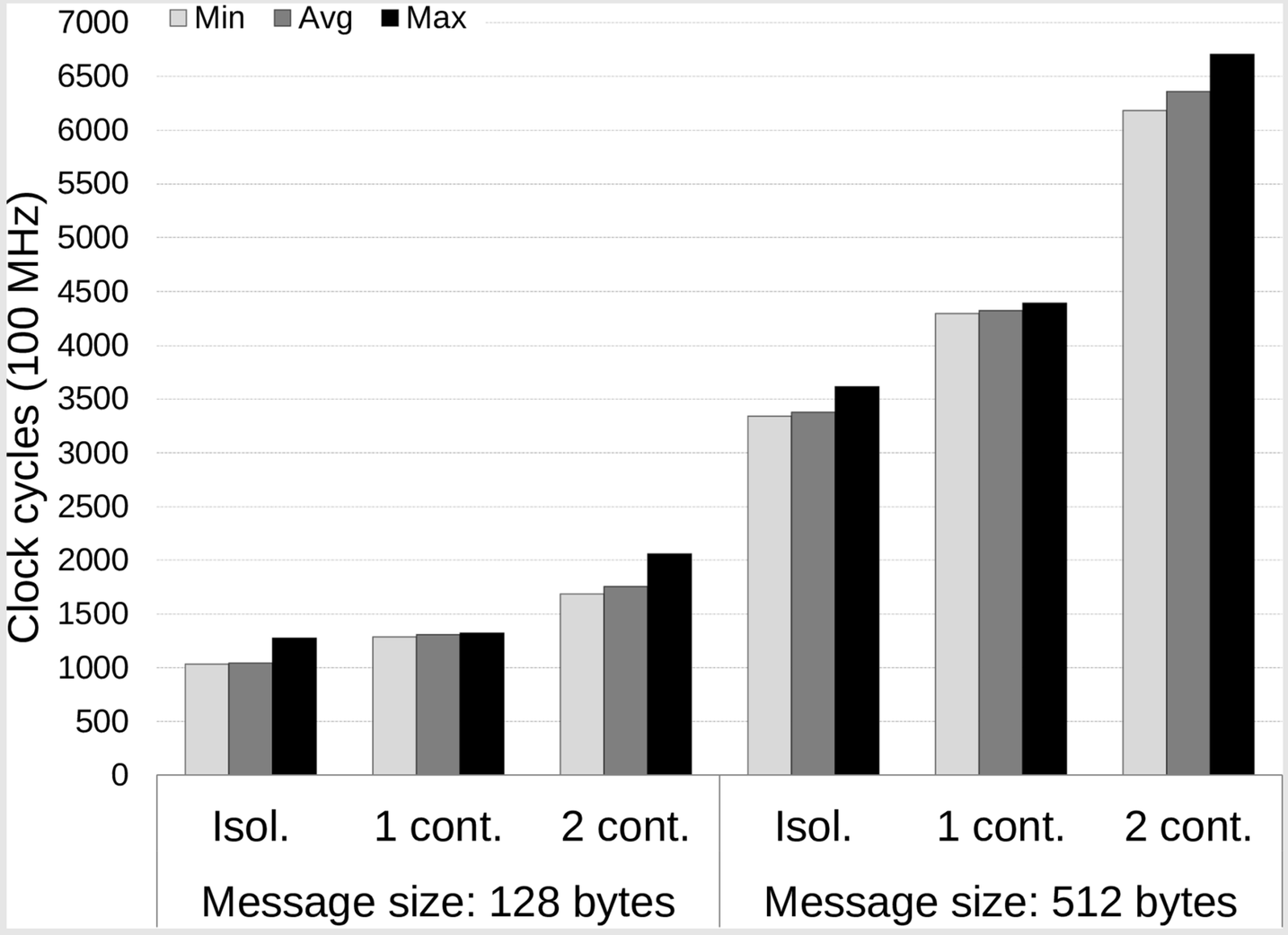
Software

- ERIKA Enterprise
  - OSEK compliant RTOS
  - User-level library implementing the AUTOSAR IOC for message passing API
- Tasks send messages of 128 and 512 bytes
  - With different number of contenders
- Task code and data are placed inside core-local scratchpads

Standard SRI HW Arbitration

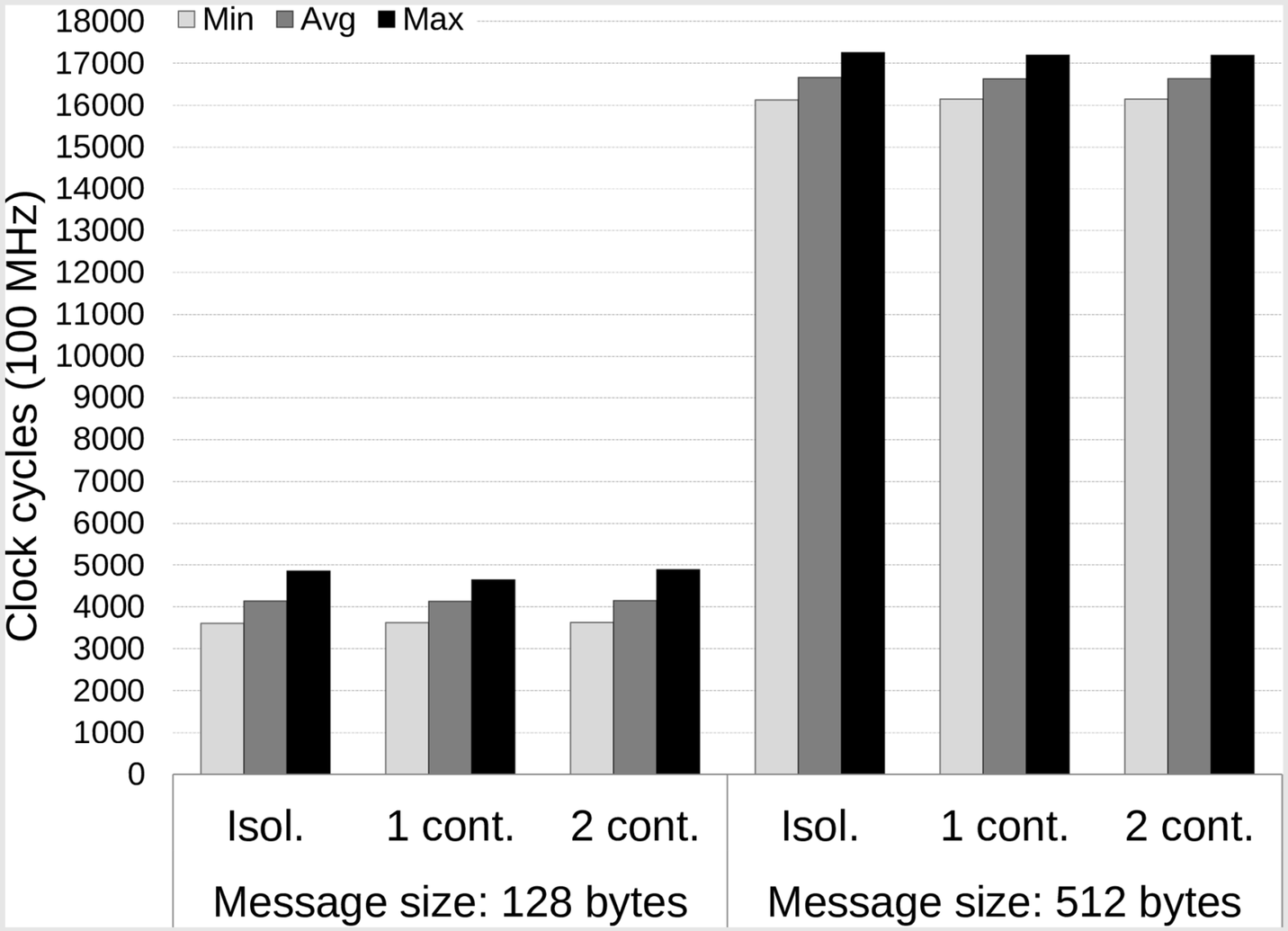
All cores in the same priority level, arbitrated via round-robin

- Execution times grow with the number of contenders
- Tree cores sending messages of 128 bytes cause 93% growth in execution times w.r.t. isolation
- Interference is limited by transaction splitting
- Different access pattern can cause more interference



SW-TDMA Arbitration (1)

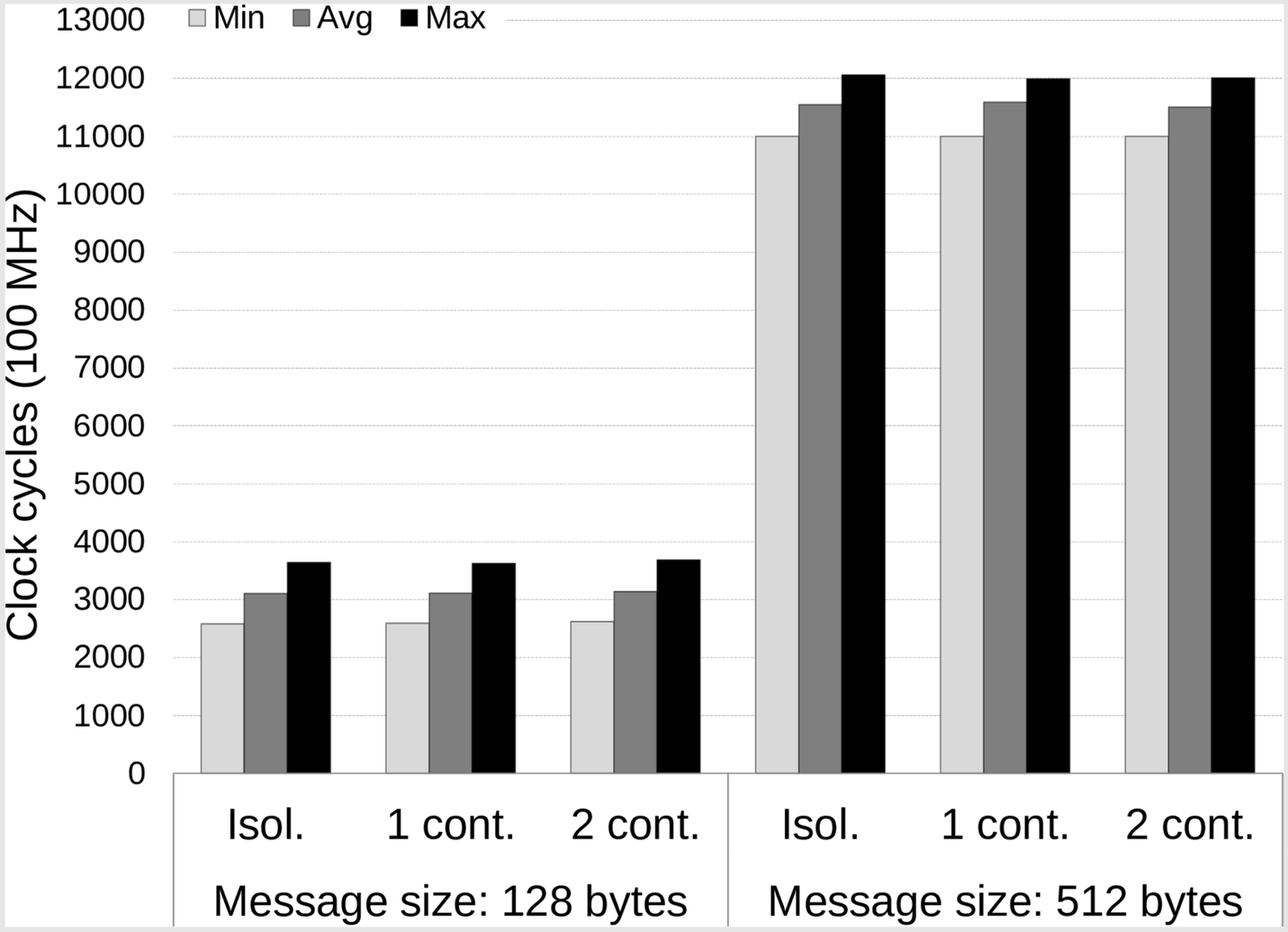
Frames of 1024 cycles divided in 3 slots, chunks of 32 bytes



- No interference
- Execution times are not affected by the number of contenders
- Inside a slot of ~342 cycles 32 bytes are transferred
- 44% of a slot is used to perform arbitration decisions

SW-TDMA Arbitration (2)

Frames of 1024 cycles divided in 3 slots, chunks of 48 bytes

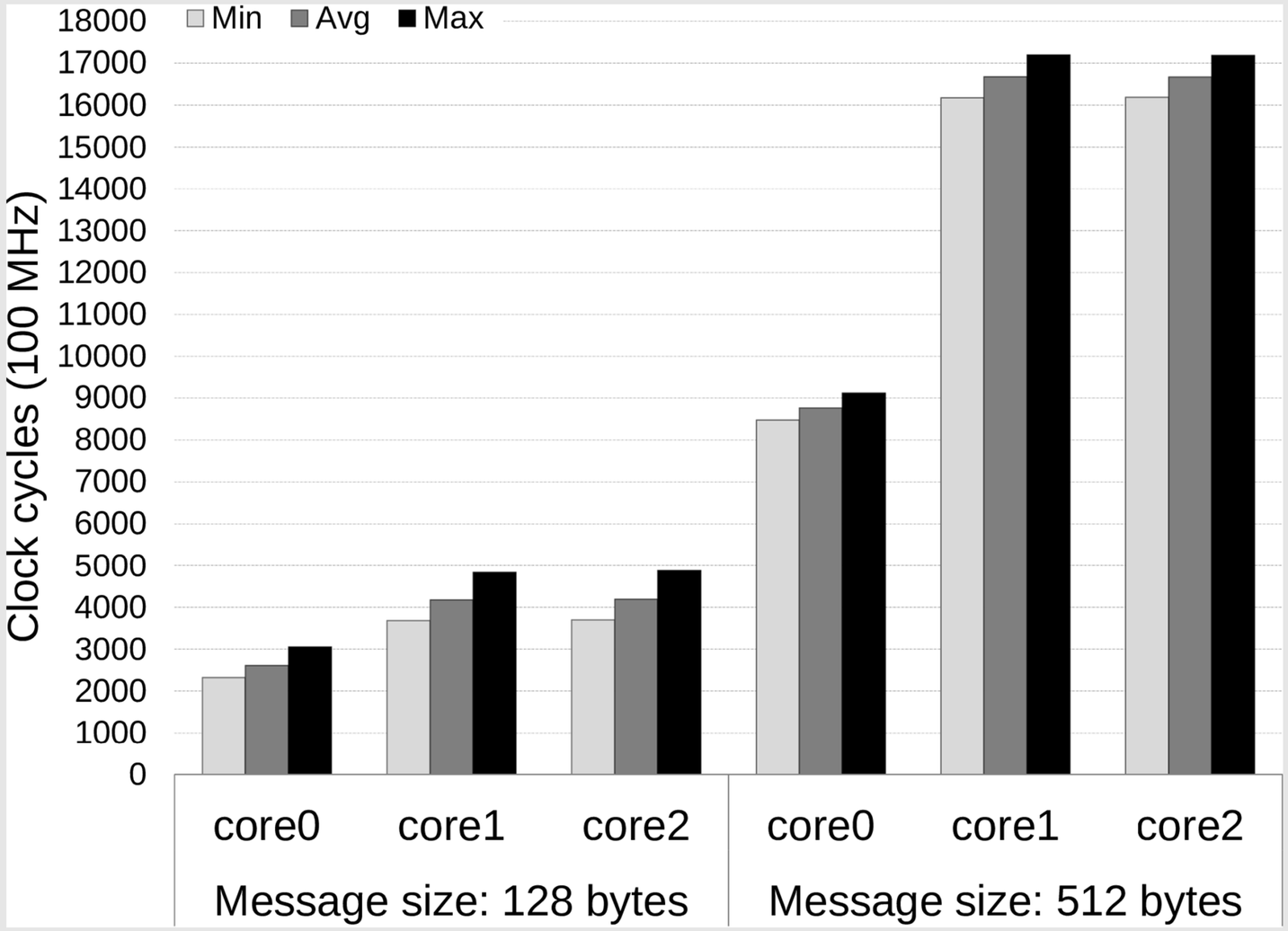


- Execution times are not affected by the number of contenders
- Inside a slot of ~342 cycles 48 bytes are transferred
  - Higher throughput can be achieved inside a slot
- Only 16% of a slot is used to perform arbitration decisions

Bandwidth Reservation TDMA

Frames of 1024 cycles divided in 4 slots, chunks of 32 bytes

- 2 slots are allocated to core 0
- On core 1 and 2 sending a message costs exactly as in SW-TDMA with chunks of 32 bytes
- Core 0 runs twice as fast (as it is allocated 2 slots)



Conclusions and Future Work

- A software solution to enforce isolation among cores accessing shared resources
  - No need for specific hardware
  - SW-TMDA experiments show a tolerable 16% throughput drop due to software arbitration
  - Bandwidth reservation TDMA meets mixed-criticality requirements
    - Better quality of service to critical tasks
- Assumes that the shared interconnect is only accessed for inter-task communication
    - Task data and instruction must be placed in core-local memories
- As a future work:
  - Target different architectures
  - Reduce software arbitration overhead
  - Investigate more complex schemes

PROXIMA

Software-enforced Interconnect Arbitration for COTS Multicores

M. Ziccardi, A. Cornaglia, E. Mezzetti, and T. Vardanega
University of Padua - Italy

15th International Workshop on Worst-Case Execution Time Analysis (WCET 2015)
Lund, July 7th, 2015

This project and the research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7 / 2007-2013] under grant agreement 611085

www.proxima-project.eu