CARP: A Data Communication Mechanism for Multi-Core Mixed-Criticality Systems

Anirudh M Kaushik, Paulos Tegegn, Zhuanhao Wu, and Hiren Patel
Motivation: Mixed Critical Systems (MCS)

High criticality  Medium criticality  Low criticality
Motivation: Data communication in MCS

Communication Centric Design in Complex Automotive Embedded Systems

Arne Hanmann1, Dukoshina Dasari2, Simon Kramer3, Michael Pressler4, and Falk Wurst5

1 Corporate Research, Robert Bosch GmbH, Germany
www.bosch.de/bosch.com
2 Corporate Research, Robert Bosch GmbH, Germany
3 Corporate Research, Robert Bosch GmbH, Germany
4 Corporate Research, Robert Bosch GmbH, Germany
5 Corporate Research, Robert Bosch GmbH, Germany

Abstract

Automotive embedded applications like the engine management system are composed of multiple functional components that are tightly coupled via secure communication (redundancy and intensive data sharing), while also having real-time requirements. In order to cope with complexity, especially in multi-core settings, various communication mechanisms are used to ensure data consistency and temporal determinism along functional cross-effect chains. However, existing testing analysis methods generally only support very basic communication models that need to be extended to handle the analysis of industry grade problems which involve more complex communication semantics. In this work, we give an overview of communication semantics used in the automotive industry and the different constraints to be considered in the design process. We also propose a method for model transformation to increase the expressiveness of current testing analysis methods enabling them to work with more complex communication semantics. We demonstrate this transformation approach for concrete implementations of two communication semantics, namely, implicit and LET communication. We discuss the impact on end-to-end latency and communication overheads based on a full blown engine management system.


Keywords and phrases: communication semantics, logical execution time, implicit communication, automotive, embedded systems, scheduling simulation, Анализ

(Hamann et al., ECRTS 2018)
Motivation: Data communication in MCS

Simultaneous use of multiple processing units (multi-cores, accelerators)

Evidence of data communication between real-time tasks deployed in automotive systems

(Hamann et al., ECRTS 2018)
Motivation: Standards on Data communication (1/2)

Section 2.7 of the AUTOSAR model constraints mandates that tasks sharing a memory partition must execute on the same core.
Motivation: Standards on Data communication (1/2)

Section 2.7 of the AUTOSAR model constraints mandates that tasks sharing a memory partition must execute on the same core.
Motivation: Standards on Data communication (1/2)

Section 2.7 of the AUTOSAR model constraints mandates that tasks sharing a memory partition must execute on the same core.

Current solution space
(Chisolm et al. RTSS 2016)

Applicable to emerging platforms?
Motivation: Standards on Data communication (1/2)

Section 2.7 of the AUTOSAR model constraints mandates that tasks sharing a memory partition must execute on the same core.

Current solution space
(Chisolm et al. RTSS 2016)

- Considerable underutilization of hardware parallelism
- Deploying tasks onto processing elements best suited for their execution is limited

Applicable to emerging platforms?
Motivation: Standards on Data communication (2/2)

No guidance in AUTOSAR on data communication between critical and non-critical tasks.
Motivation: Standards on Data communication (2/2)

No guidance in AUTOSAR on data communication between critical and non-critical tasks.

Critical tasks

Non-critical task

A.M.Kaushik, RTSS 2019

PAGE 10
Motivation: Standards on Data communication (2/2)

No guidance in AUTOSAR on data communication between critical and non-critical tasks.

Current solution space

- **Elevate** criticality level
- **Disallow** the communication

Critical tasks

- $\tau_1$
- $\tau_2$
- $\tau_3$
- $\tau_4$

Non-critical task

- $\tau_1$
- $\tau_2$
- $\tau_3$
- $\tau_4$
Motivation: Standards on Data communication (2/2)

No guidance in AUTOSAR on data communication between critical and non-critical tasks.

- Elevating non-critical tasks to critical tasks impacts temporal behavior of critical tasks.
- Disallowing communication can result in missed opportunities to realize advanced MCS functionality.
Main Contributions

Propose extensions to standards for data communication in MCS

Increases solution space for novel data communication mechanisms

CARP: A data communication mechanism that uses hardware cache coherence to enable communication in MCS

Enables predictable and high performance data communication between tasks of different criticality levels
Outline

- Proposed extensions to standards
- System Model
- Interference scenarios
- CARP design and implementation
- Overview of latency analysis
- Evaluation
- Conclusion
# Proposed Extensions for Data Communication

## Current guidelines

Section 2.7 of the AUTOSAR model constraints mandates that tasks sharing a memory partition must execute on the same core.

## Proposed extensions

Allow tasks sharing a memory partition to execute across different cores provided temporal requirements of tasks are satisfied.
Proposed Extensions for Data Communication

<table>
<thead>
<tr>
<th>Current guidelines</th>
<th>Proposed extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 2.7 of the AUTOSAR model constraints mandates that tasks sharing a memory partition must execute on the same core.</td>
<td>Allow tasks sharing a memory partition to execute across different cores provided temporal requirements of tasks are satisfied.</td>
</tr>
<tr>
<td><strong>No guidance</strong> in AUTOSAR on data communication between critical and non-critical tasks.</td>
<td><strong>Allow</strong> communication between non-critical and critical tasks provided temporal requirements of critical tasks are satisfied.</td>
</tr>
</tbody>
</table>
Proposed Extensions for Data Communication

<table>
<thead>
<tr>
<th>Current guidelines</th>
<th>Proposed extensions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section 2.7 of the AUTOSAR model constraints mandates that tasks sharing a memory partition must execute on the same core.</td>
<td>Allow tasks sharing a memory partition to execute across different cores provided temporal requirements of tasks are satisfied.</td>
</tr>
</tbody>
</table>

**1. Predictable Hardware Cache Coherence**

**No guidance** in AUTOSAR on data communication between critical and non-critical tasks.

**Allow** communication between non-critical and critical tasks provided temporal requirements of critical tasks are satisfied.
# Proposed Extensions for Data Communication

## Current guidelines

Section 2.7 of the AUTOSAR model constraints mandates that tasks sharing a memory partition must execute on the same core. No guidance in AUTOSAR on data communication between critical and non-critical tasks.

## Proposed extensions

- Allow tasks sharing a memory partition to execute across different cores provided temporal requirements of tasks are satisfied.

### 1. Predictable Hardware Cache Coherence

- Disallow timing interference from non-critical cores’ communication on critical cores’ timing requirements.

---

A.M.Kaushik, RTSS 2019
System Model

5 levels of criticalities: \textbf{A, B, C, D, E} \hfill (S. Vestal, RTSS’07)
5 levels of criticalities: \( \mathbf{A, B, C, D, E} \) (S. Vestal, RTSS’07)

Level \( \mathbf{E} \) requests serviced in slack

(Cilku et al., EBCCSP 2015)
System Model

5 levels of criticalities: \( A, B, C, D, E \) (S. Vestal, RTSS’07)

Weighted TDM phase

RR phase

Levels \( C-D \)

Level \( E \) requests serviced in slack
(Cilku et al., EBCCSP 2015)

Core 0

Core 1

Core \( k \)

Shared Snooping Bus

I$  D$  CC

I$  D$  CC

I$  D$  CC
5 levels of criticalities: \( A, B, C, D, E \) (S. Vestal, RTSS’07)

- Weighted TDM phase
- RR phase
- Levels \( C - D \)

Level \( E \) requests serviced in slack
(Cilku et al., EBCCSP 2015)

(Hassan et al. RTAS 2017)
System Model

5 levels of criticalities: A, B, C, D, E

Weighted TDM phase

RR phase

Levels C - D

Level E requests serviced in slack

(Cilku et al., EBCCSP 2015)

High

Low

(St' Vestal, RTSS'07)

Work-conserving RR

Address

Pending request

Pending write-back

(PR buffer)

(PWB) buffer

I$

D$

CC

Core o

Shared Snooping Bus

(Hassan et al. RTAS 2017)

Pending request lookup

(PR LUT)

Address

Core

Coherence

A.M.Kaushik, RTSS 2019

PAGE 23
Timing Interference Scenario (1/2)

Broadcast request ↑ Broadcast write-back ↓ Data transfer

Blocking latency due to level E task

1 2 3 4 5 6 7 8 9 10 11
A B C A B C A B C A B

$C^A_0$: Store X

Data latency
Timing Interference Scenario (1/2)

Broadcast request  Broadcast write-back  Data transfer

Blocking latency due to level E task

Mark X for WB

PR LUT

`c^A_0`: Store X

`c^E_3`: Load X

<table>
<thead>
<tr>
<th>Address</th>
<th>Core ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td><code>c^E_3</code></td>
</tr>
</tbody>
</table>
Timing Interference Scenario (1/2)

Broadcast request \[\uparrow\] Broadcast write-back \[\downarrow\] Data transfer

Blocking latency due to level E task

PR LUT

<table>
<thead>
<tr>
<th>Address</th>
<th>Core ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>(c^E_3)</td>
</tr>
<tr>
<td>X</td>
<td>(c^C_2)</td>
</tr>
</tbody>
</table>
Timing Interference Scenario (1/2)

Broadcast request
Broadcast write-back
Data transfer

Blocking latency due to level E task

$c^A_0$: Store X
$c^E_3$: Load X
$c^C_2$: Load X
$c^A_0$: Write back X

Blocking latency due to level A-D task
Timing Interference Scenario (1/2)

Broadcast request → Broadcast write-back ↓ ↓ Data transfer

Blocking latency due to level E task

<table>
<thead>
<tr>
<th>Address</th>
<th>Core ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>(c^A_0)</td>
</tr>
<tr>
<td>X</td>
<td>(c^C_2)</td>
</tr>
<tr>
<td>X</td>
<td>(c^B_1)</td>
</tr>
</tbody>
</table>

\(c^A_0\): Store X
\(c^E_3\): Load X
\(c^C_2\): Load X
\(c^A_0\): Write back X
\(c^B_1\): Store X
Timing Interference Scenario (1/2)

Broadcast request \[→\] Broadcast write-back \[↓\] Data transfer

Blocking latency due to level E task

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

\(c^A_0\): Store X
\(c^E_3\): Load X
\(c^C_2\): Load X
\(c^A_0\): Write back X
\(c^B_1\): Store X
\(c^E_3\): Receives X

Latency due to arbitration
Timing Interference Scenario (1/2)

Broadcast request  
Broadcast write-back  
Data transfer

Blocking latency due to level E task

1 2 3 4 5 6 7 8 9 10 11
A B C A B C A B C A B

$C^A_0$: Store X
$C^E_3$: Load X
$C^C_2$: Load X
$C^A_0$: Write back X
$C^B_1$: Store X
$C^E_3$: Receives X
$C^C_2$: Receives X
Timing Interference Scenario (1/2)

Broadcast request 🔄 Broadcast write-back ⬇️ ⬇️ Data transfer

**Blocking latency due to level E task**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

- **C^0_A:** Store X
- **C^3_E:** Load X
- **C^2_C:** Load X
- **C^0_A:** Write back X
- **C^1_B:** Store X
- **C^3_E:** Receives X
- **C^2_C:** Receives X
- **C^1_B:** Receives X
Timing Interference Scenario (1/2)

Broadcast request ⬆️ Broadcast write-back ⬇️ Data transfer

Blocking latency due to level E task

Pending requests to same address from memory are serviced based on **broadcasted order**

- \(C^A_0\): Store X
- \(C^E_3\): Load X
- \(C^C_2\): Load X
- \(C^A_0\): Write back X
- \(C^B_1\): Store X
- \(C^E_3\): Receives X
- \(C^C_2\): Receives X
- \(C^B_1\): Receives X
Timing Interference Scenario (2/2)

Broadcast request  
Broadcast write-back  
Data transfer

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

**Blocking latency due to level E task**

- **C^E_0**: Mark X for WB
- **C^E_1**: PWB of c^A_0
- **C^E_2**: Address
- **C^E_3**: Load X
### Timing Interference Scenario (2/2)

Broadcast request  
Broadcast write-back  
Data transfer

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

**Blocking latency due to level E task**

<table>
<thead>
<tr>
<th>Task</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C^A_0$</td>
<td>1</td>
</tr>
<tr>
<td>$C^B_1$</td>
<td>2</td>
</tr>
<tr>
<td>$C^C_2$</td>
<td>3</td>
</tr>
<tr>
<td>$C^E_3$</td>
<td>4</td>
</tr>
</tbody>
</table>

- Broadcast request
- Broadcast write-back
- Data transfer

- **Mark Y for WB**
- **PWB of $c^A_0$**

**Address**

- X
- Y

**Load X**

**Load Y**
Timing Interference Scenario (2/2)

Broadcast request \[\uparrow\] Broadcast write-back \[\downarrow\] Data transfer

Blocking latency due to level E task

Block latency due to level A-D task

\(C_E^3: \text{Load } X\)
\(C_C^2: \text{Load } Y\)
\(C_A^0: \text{Write back } X\)
Timing Interference Scenario (2/2)

Broadcast request ➖ Broadcast write-back ➖ Data transfer

Blocking latency due to level E task

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ C^A_0: \text{Write back X} \]
\[ C^B_1: \text{Load Z} \]
\[ C^C_2: \text{Load Y} \]
\[ C^E_3: \text{Load X} \]

Address

- X
- Y
- Z
Timing Interference Scenario (2/2)

Blocking latency due to level E task

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Broadcast request**
- **Broadcast write-back**
- **Data transfer**

- **C^E_3**: Load X
- **C^C_2**: Load Y
- **C^A_0**: Write back X
- **C^B_1**: Load Z
- **C^E_3**: Receive X

Latency due to arbitration

Address:
- X
- Y
- Z
Timing Interference Scenario (2/2)

Broadcast request ➕ Broadcast write-back ➖ Data transfer

Blocking latency due to level E task

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
</tbody>
</table>

- **C^E_3**: Load X
- **C^C_2**: Load Y
- **C^A_0**: Write back X
- **C^B_1**: Load Z
- **C^E_3**: Receive X
- **C^A_0**: Write back Y

Address:
- X
- Y
- Z

PWB of C^A_0
Timing Interference Scenario (2/2)

Broadcast request \[\uparrow\] Broadcast write-back \[\downarrow\] Data transfer

Blocking latency due to level E task

\[C^E_3: \text{Load X} \]
\[C^C_2: \text{Load Y} \]
\[C^A_0: \text{Write back X} \]
\[C^B_1: \text{Load Z} \]
\[C^E_3: \text{Receive X} \]
\[C^A_0: \text{Write back Y} \]
\[C^C_2: \text{Receive Y} \]
Timing Interference Scenario (2/2)

Broadcast request

Broadcast write-back

Data transfer

Blocking latency due to level E task

1 2 3 4 5 6 7 8 9 10 11

A B C A B C A B C A B

$C^A_0$: Write back X

$C^B_1$: Load Z

$C^E_3$: Receive X

$C^A_0$: Write back Y

$C^C_2$: Receive Y

$C^A_0$: Write back Z

A.M.Kaushik, RTSS 2019
Timing Interference Scenario (2/2)

Broadcast request

Broadcast write-back

Data transfer

Blocking latency due to level E task

1 2 3 4 5 6 7 8 9 10 11

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$C^A_0$</td>
<td>WB X</td>
<td></td>
<td>WB Y</td>
<td></td>
<td>WB Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C^B_1$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Z</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C^C_2$</td>
<td></td>
<td></td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$C^E_3$</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$c^E_3$: Load X
$c^C_2$: Load Y

$c^A_0$: Write back X
$c^B_1$: Load Z
$c^E_3$: Receive X
$c^A_0$: Write back Y
$c^C_2$: Receive Y
$c^A_0$: Write back Z

Pending write-back responses in PWB to different addresses are serviced in broadcasted order.
CARP: Key Design Overview

- Coherence messages contain criticality level of tasks → **Criticality Aware**

- **Abort-and-Retry** non-critical requests on observing higher critical requests

  Eliminates timing interference scenario 1

- **Partitioned write-back buffers** to isolate critical and non-critical write-back responses

  Eliminates timing interference scenario 2
CARP Design: Criticality Aware

Architectural support

- Core $c_0$
  - D-$\tau_A$ I-$\tau_A$ CC
  - MCS arbiter
  - Shared memory

- Core $c_1$
  - D-$\tau_E$ I-$\tau_E$ CC

- Criticality register (3 bits)

- Store X
  - Remote Store X
  - MCS arbiter

- Store-$\tau_A$ X
  - Remote Store-$\tau_E$ X
  - MCS arbiter

A.M. Kaushik, RTSS 2019
CARP Design: Abort-and-Retry

1. Remove pending E-level request from PR LUT
2. Retry request

Remote load-\{A, B, C, D\} 
/store-\{A, B, C, D\}

1. Wait for data
2. Done
3. Abort

Criticality register (3 bits)

PR LUT

<table>
<thead>
<tr>
<th>Address</th>
<th>Core ID</th>
<th>Coherence message</th>
<th>CL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>3 bits</td>
</tr>
</tbody>
</table>

Architectural support

Protocol support

A.M.Kaushik, RTSS 2019
CARP Design: Abort-and-Retry

Broadcast request

Broadcast write-back

Data transfer

Blocking latency due to level E task

A B C A B C A B C A B

1 2 3 4 5 6 7 8 9 10 11

$C^A_0$: Store X

Data latency

$C^A_0$: Store X

$C^B_1$

$C^C_2$

$C^E_3$
CARP Design: Abort-and-Retry

Broadcast request  ➖  Broadcast write-back  ➖  Data transfer

Blocking latency due to level E task

1 2 3 4 5 6 7 8 9 10 11

A B C A B C A B C A B

Mark X for WB

Address | Core ID
---|---
X | $c^E_3$

$C^A_0$: Store X

$C^E_3$: Load X

A.M. Kaushik, RTSS 2019
CARP Design: Abort-and-Retry

Broadcast request | Broadcast write-back | Data transfer

Blocking latency due to level E task

<table>
<thead>
<tr>
<th>Address</th>
<th>Core ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>$c^E_3$</td>
</tr>
<tr>
<td>X</td>
<td>$c^C_2$</td>
</tr>
</tbody>
</table>

$c^A_0$: Store X
$c^E_3$: Load X
$c^C_2$: Load X

A.M.Kaushik, RTSS 2019
CARP Design: Abort-and-Retry

Broadcast request \[\uparrow\] Broadcast write-back \[\downarrow\] Data transfer

Blocking latency due to level E task

1 2 3 4 5 6 7 8 9 10 11

A B C A B C A B C A B

\(c^A_0\): Store X
\(c^E_3\): Load X
\(c^C_2\): Load X
\(c^A_0\): Write back X
CARP Design: Abort-and-Retry

Broadcast request \( \uparrow \) Broadcast write-back \( \downarrow \) Data transfer

Blocking latency due to level E task

<table>
<thead>
<tr>
<th>Address</th>
<th>Core ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>( E_3 )</td>
</tr>
<tr>
<td>X</td>
<td>( C_2 )</td>
</tr>
<tr>
<td>X</td>
<td>( B_1 )</td>
</tr>
</tbody>
</table>

\( c^A_0 \): Store X
\( c^E_3 \): Load X
\( c^C_2 \): Load X
\( c^A_0 \): Write back X
\( c^B_1 \): Store X

A.M. Kaushik, RTSS 2019
CARP Design: Abort-and-Retry

Broadcast request \(\uparrow\) Broadcast write-back \(\downarrow\) Data transfer

Blocking latency due to level E task

<table>
<thead>
<tr>
<th>Address</th>
<th>Core ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>(c^E_3)</td>
</tr>
<tr>
<td>X</td>
<td>(c^C_2)</td>
</tr>
<tr>
<td>X</td>
<td>(c^B_1)</td>
</tr>
</tbody>
</table>

\(c^A_0\): Store X
\(c^E_3\): Load X
\(c^C_2\): Load X
\(c^A_0\): Write back X
\(c^B_1\): Store X
\(c^C_2\): Receive X

A.M.Kaushik, RTSS 2019
**CARP Design: Abort-and-Retry**

Broadcast request Broadcast write-back Data transfer

Blocking latency due to level E task

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

- \(c^A_0\): Store X
- \(c^E_3\): Load X
- \(c^C_2\): Load X
- \(c^A_0\): Write back X
- \(c^B_1\): Store X
- \(c^C_2\): Receives X
- \(c^E_3\): Retry Load X

<table>
<thead>
<tr>
<th>Address</th>
<th>Core ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td>(c^B_1)</td>
</tr>
<tr>
<td>X</td>
<td>(c^E_3)</td>
</tr>
</tbody>
</table>

A.M.Kaushik, RTSS 2019
CARP Design: Abort-and-Retry

Broadcast request \( \uparrow \) Broadcast write-back \( \downarrow \) Data transfer

Blocking latency due to level E task

\[ c^A_0: \text{Store X} \]
\[ c^E_3: \text{Load X} \]
\[ c^C_2: \text{Load X} \]
\[ c^A_0: \text{Write back X} \]
\[ c^B_1: \text{Store X} \]
\[ c^C_2: \text{Receives X} \]
\[ c^E_3: \text{Retry Load X} \]
\[ c^B_1: \text{Receives X} \]
CARP Design: Partitioned Write-Back Buffers

Architectural support

- Core $c_0$
- Core $c_1$
- Core $c_n$
- MCS arbiter
- Shared memory

Criticality register (3 bits)
- RR arbitration
- PR Address
- Critical PWB Address
- Non-Critical PWB Address
- PR LUT
- Address
- Core ID
- Coherence message
- CL
- 3 bits

Protocol support

Serviced in slack

- Mark for WB in non-critical PWB
- Remote $E$-Level cache miss
- Mark for WB in critical PWB
- Remote $\{A-D\}$-Level cache miss
**CARP Design: Partitioned Write-Back Buffers**

Broadcast request $\uparrow$ Broadcast write-back $\downarrow$ Data transfer

Blocking latency due to level E task

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

$C^A_0$: Mark X for WB

$C^B_1$: Critical PWB

$C^C_2$: Non-Critical PWB

$C^E_3$: Load X

A.M.Kaushik, RTSS 2019
**CARP Design: Partitioned Write-Back Buffers**

Broadcast request ↑ Broadcast write-back ↓ ↓ Data transfer

### Blocking latency due to level E task

|      | A | B | C | A | B | C | A | B | C | A | B | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| \(C^A_0\) | Mark Y for WB | Critical PWB | Non-Critical PWB |
| \(C^B_1\) | Y | X |
| \(C^C_2\) | Y | X |
| \(C^E_3\) | X | |

\(C^E_3\): Load X  
\(C^C_2\): Load Y
CARP Design: Partitioned Write-Back Buffers

Broadcast request → Broadcast write-back ↓ ↓ Data transfer

Blocking latency due to level E task

$C^A_{0}$: Write back Y

$C^E_{3}$: Load X

$C^C_{2}$: Load Y

A.M.Kaushik, RTSS 2019
CARP Design: Partitioned Write-Back Buffers

Blocking latency due to level E task

Broadcast request \[\uparrow\] Broadcast write-back \[\downarrow\] Data transfer

- \(C^E_3\): Load X
- \(C^C_2\): Load Y
- \(C^A_0\): Write back Y
- \(C^B_1\): Load Z

A.M.Kaushik, RTSS 2019
CARP Design: Partitioned Write-Back Buffers

Broadcast request

Broadcast write-back

Data transfer

Blocking latency due to level E task

$C^A_0$: Write back Y

$C^B_1$: Load Z

$C^C_2$: Receive Y

$C^E_3$: Load X

A.M. Kaushik, RTSS 2019
CARP Design: Partitioned Write-Back Buffers

Broadcast request

Broadcast write-back

Data transfer

Blocking latency due to level E task

\( c^E_3 \): Load X

\( c^C_2 \): Load Y

\( c^A_0 \): Write back Y

\( c^B_1 \): Load Z

\( c^C_2 \): Receive Y

\( c^A_0 \): Write back Z
CARP Design: Partitioned Write-Back Buffers

Broadcast request \( \uparrow \) Broadcast write-back \( \uparrow \) Data transfer

Blocking latency due to level E task

\[ C^E_3: \] Load X
\[ C^C_2: \] Load Y
\[ C^A_0: \] Write back Y
\[ C^B_1: \] Load Z
\[ C^C_2: \] Receive Y
\[ C^A_0: \] Write back Z
\[ C^B_1: \] Receive Z

A.M.Kaushik, RTSS 2019
CARP Design: Partitioned Write-Back Buffers

Broadcast request
Broadcast write-back
Data transfer

Blocking latency due to level E task

$C^A_0$: Write back Y
$C^B_1$: Load Z
$C^C_2$: Receive Y
$C^A_0$: Write back Z
$C^B_1$: Receive Z

$C^A_0$: Write back X

A.M.Kaushik, RTSS 2019
Latency Analysis: Key Takeaways

WCL of a critical memory access independent of non-critical cores

WCL = WC Request latency + WC Communication latency + WC Response latency

WC Request and Response latencies dependent on arbitration schedule \((\alpha N)\)

WC Communication latency dependent on memory activity of other cores \((\alpha N^2)\)
Overview of Latency Analysis of CARP

C_{ua} issues request \hspace{2cm} WCL_{Request}: Worst-case request latency \hspace{2cm} C_{ua} broadcasts request

\begin{align*}
\text{Total WCL} &= WCL_{Request} \\
\text{Time taken to broadcast a core’s request on the shared bus}
\end{align*}
Overview of Latency Analysis of CARP

$C_{ua}$ broadcasts request

WCL\textsubscript{Communication}: Worst-case communication latency

Last core completes write-back

Remaining cores broadcast write requests to same address as $C_{ua}$'s access (worst-case scenario)

Total WCL = WCL\textsubscript{Request} + WCL\textsubscript{Communication}

Time taken for interfering cores to receive and write-back requested data
Overview of Latency Analysis of CARP

Total WCL = WCL_{Request} + WCL_{Communication} + WCL_{Response}

Time taken for shared memory to **send** requested data to C_{ua}
Evaluation Methodology

- Gem5 micro-architectural simulator
- 8-core setup, in-order cores, 1 outstanding request per core
- 2 level cache hierarchy: private L1 caches, shared L2 cache
- Task to core mapping: 2 cores each to levels A–C, one core each to levels D–E
- Arbitration policy: Weighted TDM for levels A–B, Round-Robin for levels C–D, slack for level E
- Synthetic and SPLASH-2 benchmarks
Results: Observed worst-case latency

Observed worst-case latencies across all synthetic benchmarks

<table>
<thead>
<tr>
<th>Criticality level</th>
<th>Analytical WCL (cycles)</th>
<th>Observed WCL (cycles)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>6600</td>
<td>4348</td>
</tr>
<tr>
<td>B</td>
<td>6800</td>
<td>3701</td>
</tr>
<tr>
<td>C/D</td>
<td>11200</td>
<td>6699</td>
</tr>
</tbody>
</table>

Observed WCL under CARP is within analytics WCL
1. PMSI and CARP allow cores to simultaneously cache communicated data
2. CARP performs better than PMSI (30% average) due to slack allocation for level E cores; PMSI allocates slots for level E cores
1. CARP offers slight performance improvement (4% average) over PMSI
2. SPLASH-2 benchmarks have thread barriers that force all threads to converge at barrier before making forward progress
Conclusion

- CARP: A criticality-aware hardware cache coherence mechanism for MCS
- 2 key features of CARP:
  - Non-critical tasks **abort-and-retry** memory requests to data in the presence of simultaneous memory requests to the same data by critical tasks.
  - Write-back responses due to memory activity from non-critical tasks are **separated** from write-back responses from critical tasks; write-back responses from latter are prioritized over former
- Under CARP, data communication with non-critical tasks do **not** impact temporal behavior of critical tasks
- Evaluation shows that observed WCL of memory requests under CARP are within analytical WCL bounds, and CARP shows significant performance benefits over alternative data communication mechanisms for MCS
- CARP is publicly available at: https://git.uwaterloo.ca/caesr-pub/mcs-carp
THANKS AND QUESTIONS?