A Real-Time Scratchpad-centric OS for Multi-core Embedded Systems

Rohan Tabish, Renato Mancuso, Saud Wasly, Ahmed Alhammad, Sujit S. Phatak, Rodolfo Pellizzoni, Marco Caccamo

University of Illinois at Urbana-Champaign
University of Waterloo
Hitachi America Ltd

RTAS 2016 Vienna
Embedded code size statistics
Embedded code size statistics

~ 1.7 million lines of code in a F-22 Jet Fighter
Embedded code size statistics

~ 1.7 million lines of code in a F-22 Jet Fighter

~ 6.5 million lines of code in a Boeing 787
Embedded code size statistics

~ 1.7 million lines of code in a F-22 Jet Fighter

~ 6.5 million lines of code in a Boeing 787

~ 20 million lines of code in S Class Mercedes-Benz
Embedded code size statistics

~ 1.7 million lines of code in a F-22 Jet Fighter

~ 6.5 million lines of code in a Boeing 787

~ 20 million lines of code in S Class Mercedes-Benz

It is estimated that future cars will have more than 200 million lines of code
Complexity of automotive software
MOTIVATION

Complexity of automotive software
MOTIVATION
Complexity of automotive software

infotainment

body
MOTIVATION

Complexity of automotive software

infotainment

engine control

body
MOTIVATION

Complexity of automotive software

= tens of interconnected MCUs
From *multiple* single-core systems

MCU 1  MCU 2  MCU 3  MCU 4

Network
From *multiple* single-core systems

```
MCU 1  MCU 2  MCU 3  MCU 4
```

*Network*

To a *single* multi-core system

```
Core 1  Core 2
Core 3  Core 4
```
From multiple single-core systems

To a single multi-core system
From multiple single-core systems

MCU 1 — MCU 2 — MCU 3 — MCU 4

Network

To a single multi-core system

Core 1 — Core 2 — Core 3 — Core 4

Memory — I/O devices

✔ Integration
From multiple single-core systems

MCU 1  MCU 2  MCU 3  MCU 4

Network

✔ Integration
✔ Cost reduction

To a single multi-core system

Core 1  Core 2  Core 3  Core 4

Memory  I/O devices
From **multiple** single-core systems

- MCU 1
- MCU 2
- MCU 3
- MCU 4

Network

To a **single** multi-core system

- Core 1
- Core 2
- Core 3
- Core 4

- Memory
- I/O devices

- ✔ Integration
- ✔ Cost reduction
- ✔ Reduced hw complexity
Network predictability is performed in parallel, but I/O and memory shared.

From multiple single-core systems:

- MCU 1
- MCU 2
- MCU 3
- MCU 4

Network

To a single multi-core system:

- Core 1
- Core 2
- Core 3
- Core 4
- Memory
- I/O devices

✔ Integration
✔ Cost reduction
✔ Reduced hw complexity

Predictability:

Computation is performed in parallel, but shared I/O and memory.
In the automotive domain, new platforms offer a set of hardware features designed for hard real-time applications.
In the automotive domain, new platforms offer a set of hardware features designed for hard real-time applications.

+ Multiple general-purpose cores
In the automotive domain, new platforms offer a set of hardware features designed for hard real-time applications.

- Multiple general-purpose cores
- Static priority / round-robin policy at memory bus
Multi-core trends in Automotive

In the automotive domain, new platforms offer a set of hardware features designed for hard real-time applications.

- Multiple general-purpose cores
- Static priority / round-robin policy at memory bus
- SRAM and Flash instead of DRAM
In the automotive domain, new platforms offer a set of hardware features designed for hard real-time applications.

- Multiple general-purpose cores
- Static priority / round-robin policy at memory bus
- SRAM and Flash instead of DRAM
- Fast, private local memories - scratchpads
Multi-core trends in Automotive

In the automotive domain, new platforms offer a set of hardware features designed for hard real-time applications.

- Multiple general-purpose cores
- Static priority / round-robin policy at memory bus
- SRAM and Flash instead of DRAM
- Fast, private local memories - scratchpads
- Specialized core for I/O processing
Multi-core trends in Automotive

In the automotive domain, new platforms offer a set of hardware features designed for hard real-time applications.

- Multiple general-purpose cores
- Static priority / round-robin policy at memory bus
- SRAM and Flash instead of DRAM
- Fast, private local memories - scratchpads
- Specialized core for I/O processing
- Dedicated bus for I/O communication
Consider *Freescale MPC5777M*
ARCHITECTURE
System dimensions

Consider Freescale MPC5777M

2 application cores
Consider **Freescale MPC5777M**

- **System dimensions**
- **2** application cores
- **~500 KB**
ARCHITECTURE

System dimensions

Consider Freescale MPC5777M

2 application cores
~500 KB
~90 KB each
Consider Freescale MPC5777M

- 2 application cores
- ~500 KB
- ~90 KB each
- ~300 MHz
Consider **Freescale MPC5777M**

- **2 application cores**
- **~500 KB**
- **~90 KB each**
- **~300 MHz**

Focus on small and critical engine-control applications.
ARCHITECTURE

Memory performance

Synthetic Benchmarks – on Freescale MPC5777M

Execution time (ms)

- Instr. Hit, Data Hit
- Instr. Hit, Data Miss
- Instr Miss, Data Hit
- Instr. Miss, Data Miss
- Scratchpad

1 Active Core
3 Active Cores
Memory performance

Synthetic Benchmarks – on Freescale MPC5777M

Execution time (ms)

- Instr. Hit, Data Hit
- Instr. Hit, Data Miss
- Instr Miss, Data Hit
- Instr. Miss, Data Miss
- Scratchpad

1 Active Core
3 Active Cores

~ 3.5x
RTOS design principles
RTOS design principles

SPM-CENTRIC OS

AUGMENT EXISTING RTOS

ERIKA ENTERPRISE
RTOS design principles

AUGMENT EXISTING RTOS

ALWAYS EXECUTE TASKS FROM SPM

do not allow tasks to directly access shared memory resources as they execute.
RTOS design principles

- **AUGMENT EXISTING RTOS**

- **ALWAYS EXECUTE TASKS FROM SPM**
  
  do not allow tasks to directly access shared memory resources as they execute.

- **SELECTIVE I/O FORWARDING**
  
  do not allow I/O traffic to asynchronously access the application-side memory bus.
Task structure and execution

- **Core** with Scratchpad
- **SRAM**
- **DMA**
- **Mem. Bus**

- Blue: execution
- Gray: memory
- Green arrows: data/instruction references
- Red dashed arrow: DMA directed transfer

**Text:**

- Task 1 data/instruction references
- DMA directed transfer
Task structure and execution

- Core
  - Scratchpad
- DMA
- Mem. Bus
- SRAM
- Task 1

**Execution**

**Memory**

**Data/instruction references**

**DMA directed transfer**
Task structure and execution

Core
- Scratchpad
- Task 1

DMA
- Mem. Bus

SRAM
- Task 1

- execution
- memory
- data/instruction references
- DMA directed transfer

load to SPM (DMA)
Task structure and execution

- Core
  - Scratchpad
  - Task 1
- DMA
  - Mem. Bus
- SRAM
  - Task 1

Colors and markers:
- Blue: execution
- Gray: memory
- Green: data/instruction references
- Red: DMA directed transfer

Diagram includes:
- "load to SPM (DMA)"
- Arrows indicating data transfer
Task structure and execution

- **Core**
  - Scratchpad
  - Task 1

- **SRAM**
  - Task 1

- **DMA**
  - Mem. Bus

- **Load to SPM (DMA)**
- **Execution in SPM (CPU)**

**Task 1**
- Memory transfer:
  - Data/instruction references
  - DMA directed transfer

**SPM-CENTRIC OS**

- **Task 1 execution**
- **Task 2 execution**
Task structure and execution

- **Core**
  - Scratchpad
  - Task 1
- **SRAM**
  - Task 1
- **DMA**
  - Mem. Bus
- **SPM-CENTRIC OS**
- **Execution**
- **Memory**
- **Data/instruction references**
- **DMA directed transfer**
- **Load to SPM (DMA)**
- **Execution in SPM (CPU)**
SPM-CENTRIC OS

Task structure and execution

- SPM
- Core
- Scratchpad
- SRAM
- Mem. Bus
- DMA
- Task 1

- DMA directed transfer
- Data/instruction references

Execution:
- Load to SPM (DMA)
- Execution in SPM (CPU)
- Unload from SPM (DMA)
Task structure and execution

**SPM-CENTRIC OS**

- **Core**
  - Scratchpad
- **SRAM**
  - Task 1
- **DMA**
  - Mem. Bus

**3 stages, 2 resources**

\[ \tau_i = \{ L_i, C_i, U_i \} \]

- **load to SPM (DMA)**
- **execution in SPM (CPU)**
- **unload from SPM (DMA)**
Pipelining and memory bus scheduling
Memory operations can be parallelized with respect to execution.
Pipelining and memory bus scheduling

Memory operations can be parallelized with respect to execution.
Memory operations can be parallelized with respect to execution.
Memory operations can be parallelized with respect to execution.
Pipelining and memory bus scheduling

Memory operations can be parallelized with respect to execution.

**Partition in space**
Pipelining and memory bus scheduling

Memory operations can be parallelized with respect to execution.

Partition in space
Memory operations can be parallelized with respect to execution.

**Partition in space**
Pipelining and memory bus scheduling

Memory operations can be parallelized with respect to execution.

Partition in space
Memory operations can be parallelized with respect to execution.

PIPILINING

Task 1 load
Task 1 execution
Task 1 unload
Task 2 load
Task 2 execution
Task 2 unload

Scratchpad

Partition 1
next task

Partition 2
executing task

partition in space
Pipelining and memory bus scheduling

Memory operations can be parallelized with respect to execution.

Partition in space

Partition 1
- next task

Partition 2
- executing task

Partition in time (TDMA)

Task 1 load
Task 1 execution
Task 1 unload

Task 2 load
Task 2 execution
Task 2 unload

Memory Bus
Pipelining and memory bus scheduling

**Memory operations can be parallelized with respect to execution.**

**Partition in space**

- **Partition 1**: next task
- **Partition 2**: executing task

**Partition in time (TDMA)**

- **Core 1 load/unload**
- **Core 2 load/unload**
- **Core 1 load/unload**
- **Core 2 load/unload**
- **...**

**Memory Bus**
Memory operations can be parallelized with respect to execution.

Memory Bus

$$\sigma \geq \max_i (L_i, U_i)$$

### Pipelining

**Pipeline Operation**

- Task 1: load
- Task 1: execution
- Task 1: unload
- Task 2: load
- Task 2: execution
- Task 2: unload

**Partition in Space**

- Scratchpad
  - Partition 1: next task
  - Partition 2: executing task

**Partition in Time (TDMA)**

- Core 1: load/unload
- Core 2: load/unload
- Core 1: load/unload
- Core 2: load/unload
- ...
\( \tau_1, \tau_2, \tau_3 \) released

active queue

ready queue

App. Core

idle

scratchpad

SRAM

\( \tau_1 \) image

\( \tau_2 \) image

\( \tau_3 \) image

\( \tau_4 \) image

I/O Core

EDMA
$\tau_1, \tau_2, \tau_3$ released

**Active queue**

$\tau_3$, $\tau_2$, $\tau_1$

**Ready queue**

**App. Core**

- idle
- scratchpad
- $\tau_4$ image

**I/O Core**

**EDMA**

**Core 1 DMA slot**

**Core 1 DMA slot**

**Core 1 DMA slot**

**SRAM**

$\tau_1$ image

$\tau_2$ image

$\tau_3$ image

$\tau_4$ image
Load $\tau_1$

- **App. Core**
  - idle
  - scratchpad
  - $\tau_4$ image

- **Core 1 DMA slot**
  - $\tau_1$ image

- **Core 1 DMA slot**
  - $\tau_2$ image

- **Core 1 DMA slot**
  - $\tau_3$ image

- **SRAM**
  - $\tau_1$
  - $\tau_2$
  - $\tau_3$
  - $\tau_4$

- **I/O Core**

- **EDMA**

**Active queue**
- $\tau_1$
- $\tau_2$
- $\tau_3$

**Ready queue**
App. Core
idle

Core 1 DMA slot

Load $\tau_1$

Core 1 DMA slot

Core 1 DMA slot

time

active queue

$\tau_1$ $\tau_2$ $\tau_3$

ready queue

SRAM

$\tau_1$ image $\tau_2$ image

$\tau_3$ image $\tau_4$ image

I/O Core

program

EDMA

EDMA

$\tau_4$ image

scratchpad
The diagram illustrates the load completion process within a system with multiple DMA slots. The load is completed at time $\tau_4$. The active queue contains $\tau_3$ and $\tau_2$, and the ready queue contains $\tau_1$. The system features an active core, an idle core, a scratchpad, an I/O core, and an EDMA component.
Core 1 DMA slot | Core 1 DMA slot | Core 1 DMA slot
---|---|---
Unload $\tau_4$

**Active queue**

$\tau_3$ $\tau_2$

**Ready queue**

**App. Core**

running: $\tau_1$

$\tau_1$ image

$\tau_4$ image

**Scratchpad**

**SRAM**

$\tau_1$ image $\tau_2$ image $\tau_3$ image $\tau_4$ image

**I/O Core**

**EDMA**
App. Core

running: $\tau_1$

$\tau_1$ image

$\tau_4$ image

Core 1 DMA slot

Core 1 DMA slot

Core 1 DMA slot

time

active queue

ready queue

I/O Core

program

EDMA

$\tau_1$ image

$\tau_1$ image

$\tau_2$ image

$\tau_2$ image

$\tau_3$ image

$\tau_4$ image

$\tau_3$ image

$\tau_4$ image

scratchpad
App. Core

running: $\tau_1$

Core 1 DMA slot

Scratchpad

SRAM

$\tau_1$ image

$\tau_2$ image

$\tau_3$ image

$\tau_4$ image

Core 1 DMA slot

Core 1 DMA slot

Unloads completed

I/O Core

EDMA

active queue

$\tau_2$

$\tau_3$

ready queue

$\tau_1$ image

$\tau_2$ image

$\tau_3$ image

$\tau_4$ image
App. Core
idle

<table>
<thead>
<tr>
<th>τ₁ image</th>
</tr>
</thead>
</table>

Core 1 DMA slot

Scratchpad

<table>
<thead>
<tr>
<th>τ₁ image</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₂ image</td>
</tr>
<tr>
<td>τ₃ image</td>
</tr>
<tr>
<td>τ₄ image</td>
</tr>
</tbody>
</table>

SRAM

Core 1 DMA slot

Load τ₂

Active queue

| τ₂ |
| τ₃ |

Ready queue

Core 1 DMA slot

I/O Core

EDMA
Core 1 DMA slot

Core 1 DMA slot

Core 1 DMA slot

time

Load completed

App. Core

idle

scratchpad

Core 1

DMA slot

active queue

ready queue

SRAM

I/O Core

program

EDMA

𝜏₁

τ₁
image

𝜏₁
image

𝜏₂
image

𝜏₂
image

𝜏₃
image

𝜏₃
image

𝜏₄
image

𝜏₄
image
SPM-CENTRIC OS

I/O handling
do not allow I/O traffic to asynchronously access the application-side memory bus.
SELECTIVE I/O FORWARDING

don't allow I/O traffic to asynchronously access the application-side memory bus.
SELECTIVE I/O FORWARDING

do not allow I/O traffic to asynchronously access the application-side memory bus.
SELECTIVE I/O FORWARDING

do not allow I/O traffic to asynchronously access the application-side memory bus.
I/O handling

SELECTIVE I/O FORWARDING

Do not allow I/O traffic to asynchronously access the application-side memory bus.
SELECTIVE I/O FORWARDING

Do not allow I/O traffic to asynchronously access the application-side memory bus.
SELECTIVE I/O FORWARDING

Do not allow I/O traffic to asynchronously access the application-side memory bus.
I/O data dispatch
do not allow I/O traffic to asynchronously access the application-side memory bus.

SELECTIVE I/O FORWARDING
SPM–CENTRIC OS

I/O handling

SELECTIVE I/O FORWARDING

do not allow I/O traffic to asynchronously access the application-side memory bus.

I/O data dispatch

discard

Core 1

SPM

Memory Bus

SRAM

Task 1

Task 2

I/O Core

Task 1 - in

pkt 1

Task 2 - in

pkt 2

Dev. 1 - in

Dev. 2 - in

I/O Bus

Device 1

Device 2

Packet 3
I/O data dispatch

SELECTIVE I/O FORWARDING

do not allow I/O traffic to asynchronously access the application-side memory bus.
Analysis principles
SCHEDULABILITY ANALYSIS

Analysis principles

WORST-CASE RESPONSE TIME

from task release to end of unload phase
**Analysis principles**

**WORST-CASE RESPONSE TIME**
從任務釋放到卸載階段結束的時間

**SCHEDULING INTERVAL**

1. 只有一個阻塞或干擾任務運行
2. 開始與 CPU 執行
3. 結束時 CPU 完成執行，或當下一個任務的載入階段完成時
Analysis approach

- DMA load
- DMA unload
- Other core's slot

Time:
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
Analysis approach

SCHEDULABILITY ANALYSIS

Task priority

DMA load

DMA unload

Other core’s slot

Time

\( \tau_1 \)

\( \tau_2 \)

\( \tau_4 \)

\( \tau_5 \)
Analysis approach
Analysis approach

SCHEDULABILITY ANALYSIS

- \( \tau_1 \)
- \( \tau_2 \)
- \( \tau_3 \)
- \( \tau_4 \)
- \( \tau_5 \)

Task priority:
- High priority
- Under analysis
- Low priority

DMA load
DMA unload
Other core's slot

Time: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
Analysis approach

Task priority

DMA load

DMA unload

Other core's slot

Time

Task under analysis

High priority

Low priority

\( \tau_1 \)

\( \tau_2 \)

\( \tau_3 \)

\( \tau_4 \)

\( \tau_5 \)
Analysis approach

SCHEDULABILITY ANALYSIS

- Task priority
  - High priority
  - Low priority
- Interval 1
  - \( \tau_1 \)
  - \( \tau_2 \)
  - \( \tau_3 \)
  - \( \tau_4 \)
  - \( \tau_5 \)
- DMA load
- DMA unload
- Other core’s slot

Time:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27
Schedulability Analysis

Analysis approach

- Task priority
- DMA load
- DMA unload
- Other core’s slot

- Time

- Interval 1

- Priority
  - High priority
  - Under analysis
  - Low priority
Analysis approach

Schedulability Analysis

- Analysis approach
  - Task priority
    - High priority
    - Under analysis
    - Low priority
  - DMA load
  - DMA unload
  - Other core's slot

Interval 1

- $\tau_1$
- $\tau_2$
- $\tau_3$
- $\tau_4$
- $\tau_5$

Time

DMA unload
DMA load

20
SCHEDULABILITY ANALYSIS

Analysis approach

......

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>interval 1</th>
<th>interval 2</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>τ₁</td>
<td>τ₂</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>τ₃</td>
<td>τ₄</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>τ₅</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **τ₁**: High priority task under analysis
- **τ₂**: High priority task under analysis
- **τ₃**: High priority task under analysis
- **τ₄**: Low priority task
- **τ₅**: Low priority task

- DMA load
- DMA unload
- Other core's slot

- Task priority
- High priority
- Low priority
- Under analysis

- Time
- X

20
Analysis approach

SCHEDULABILITY ANALYSIS

Task priority

- High priority
- Under analysis
- Low priority

Time intervals:
- Interval 1
- Interval 2
- Interval 3

Tasks:
- τ₁
- τ₂
- τ₃
- τ₄
- τ₅

Events:
- DMA load
- DMA unload
- Other core's slot

Time scale: 0 to 27
Analysis approach

### Task Priority
- **High Priority**
- **Under Analysis**
- **Low Priority**

### Analysis Intervals
- **Interval 1**
- **Interval 2**
- **Interval 3**
- **Interval 4**

### DMA Operations
- **DMA Load**
- **DMA Unload**

### Other Core’s Slot

### Time Line

---

**τ₁**

**τ₂**

**τ₃**

**τ₄**

**τ₅**
Analysis approach

The diagram illustrates the schedulability analysis for tasks with different priorities. The tasks are categorized into high and low priority, and their intervals are marked with 'X' for under analysis and 'u' for analysis of task priority.

1. Task \( \tau_1 \) has an interval from time 1 to 4.
2. Task \( \tau_2 \) has an interval from time 2 to 4, with DMA unload at time 1.
3. Task \( \tau_3 \) has an interval from time 3 to 5, with DMA unload at time 1.
4. Task \( \tau_4 \) has an interval from time 4 to 6, with DMA unload at time 1.
5. Task \( \tau_5 \) has an interval from time 5 to 7, with DMA unload at time 1.

The chart also indicates DMA load and unload events and the slot busy time for other cores.
Analysis approach

**Schedulability Analysis**

Task priority:
- High priority
- Under analysis
- Low priority

Task names:
- \( \tau_1 \)
- \( \tau_2 \)
- \( \tau_3 \)
- \( \tau_4 \)
- \( \tau_5 \)

DMA load and unload:
- DMA load
- DMA unload

Other core's slot:
- X

Time intervals:
- Interval 1
- Interval 2
- Interval 3
- Interval 4
- Interval \( F \)

Graphical representation of task execution and scheduling.
Critical instant analysis

Critical instant:

- Task priorities:
  - High priority
  - Under analysis
  - Low priority

- Task intervals:
  - Interval 1
  - Interval 2

- Events:
  - DMA load
  - DMA unload
  - Other core's slot

- Time:
  - Time axis with intervals 0 to 9
Critical instant

Critical instant:

1. Task under analysis released with all the **high priority tasks**
Critical instant:

1. Task under analysis released with all the **high priority tasks**

2. **Right after** a lower priority tasks starts **loading**
Critical instant:

1. Task under analysis released with all the high priority tasks
2. Right after a lower priority tasks starts loading
3. When the other partition was previously loaded with another low priority task
SCHEDULABILITY ANALYSIS

Blocking term

blocking B

\[ \sigma \geq \max_i (L_i, U_i) \]

DMA slot size

B

interval 1

\( \tau_1 \)

\( \tau_2 \)

\( \tau_3 \)

\( \tau_4 \)

\( \tau_5 \)

DMA load

DMA unload

other core’s slot

task priority

high priority

under analysis

low priority

time
**Schedulability Analysis**

**Blocking term**

\[ B \]

- **Task priority**:
  - High priority
  - Under analysis
  - Low priority

- **DMA load**
- **DMA unload**
- **Other core’s slot**

- **Interval**:
  - Interval 1
  - Interval 2
  - Interval 3
  - Interval 4

- **Blocking B**

\[ \sigma \geq \max_i (L_i, U_i) \]

- **DMA slot size**

- **As long as** (interval 1) − \( \sigma \)

- **Time**
SCHEDULABILITY ANALYSIS

Blocking term

\[ \sigma \geq \max_i(L_i, U_i) \]

\begin{align*}
\text{blocking } B & \leq \max(C_5, 2\sigma) - \sigma \\
\text{as long as } (\text{interval } 1) - \sigma & \geq \max_i(L_i, U_i)
\end{align*}
SCHEDULABILITY ANALYSIS

Busy period

\[
\sigma \geq \max_i (L_i, U_i)
\]

DMA slot size

\(\tau_1\), \(\tau_2\), \(\tau_3\), \(\tau_4\), \(\tau_5\)

Under analysis

High priority

Low priority

Task priority

DMA load

DMA unload

Other core’s slot

Busy period \(H\)

Interval 2

Interval 3

Interval 4

Time
**Schedulability Analysis**

**Busy period**

![Diagram]

- **Task priority**:
  - High priority
  - Under analysis
  - Low priority

- **DMA slot size**
  - \( \sigma \)

- ** boury period \( H \)**

- **Equation**
  - \( \sigma \geq \max_i (L_i, U_i) \)

- **Conditions**
  - If \( C > 4\sigma \):
    - Interval \( \leq \max(C, 5\sigma) \)
SCHEDULABILITY ANALYSIS

Busy period

\[ \tau_1 \leq \tau_2 \leq \tau_3 \leq \tau_4 \leq \tau_5 \]

\[ \forall i \in \text{intervals} \leq \max_i (L_i, U_i) \]

\[ \sigma \]

DMA slot size

\[ \sigma \geq \max_i (L_i, U_i) \]

busy period \( H \)

\[ \text{if } C > 4\sigma: \]

\[ \text{interval } \leq \max(C, 5\sigma) \]

\[ \text{else: } \]

\[ \text{interval } \leq 4\sigma \]
Schedulability Analysis

Critical instant

\[ \tau_5 \geq \max_i (L_i, U_i) \]

DMA slot size

Final interval \( F \)

Task priority

High priority

Under analysis

Low priority

18 19 20 21 22 23 24 25 26 27

Interval \( F \)

DMA load

DMA unload

Other core's slot

Time
Schedulability Analysis

Critical instant

\[ \tau_5 \]

\[ \tau_4 \]

\[ \tau_3 \]

\[ \tau_2 \]

\[ \tau_1 \]

\[ \begin{align*}
\sigma & \geq \max_i (L_i, U_i) \\
\text{DMA slot size} & \\
\text{final interval } F & \\
\text{max between:} & \\
C_i + 5\sigma & \\
\end{align*} \]

Task priority

DMA load

DMA unload

Under analysis

High priority

Low priority

Interval \( F \)

18 19 20 21 22 23 24 25 26 27

Other core's slot

Time
Critical instant

\[ \tau_1 \geq \max_i (L_i, U_i) \]

DMA slot size

final interval \( F \)

max between:

\[ C_i + 5\sigma \]

\[ 7\sigma \]
EEMBC automotive benchmarks

The diagram compares the execution time of various benchmarks using SPM (Single Program Multiple Threads) and SRAM (Single Instruction Multiple Data) execution models. The benchmarks include:

- tblook
- matrix
- a2time
- pntrch
- ttspk
- irrft
- canrd
- bitmnp
- rspeed
- puwm
- aiffr
- aiftr
- aiffft
- idct

The x-axis represents the benchmarks, and the y-axis shows the execution time. The bars for each benchmark indicate the performance in SPM and SRAM execution modes.
EEMBC automotive benchmarks

**up to 1.1x**
EEMBC automotive benchmarks

- **up to 1.1x**
- **up to 1.4x**

For more information, please refer to the image.
EVALUATION

EEMBC automotive benchmarks

- up to 1.1x
- up to 1.4x
- up to 2.1x
Schedulability envelope

![Diagram showing schedulability envelope comparison between Contention and Our approach.](image)
Summary and future work
CONCLUSION

Summary and future work

- **RTOS** designed for performance isolation on multi-core
- Exploits specialization and scheduling of shared resources
- Predictability ensured, development process simplified
- Possible to achieve performance benefits
**Summary and future work**

- **RTOS** designed for performance isolation on multi-core
- Exploits specialization and scheduling of shared resources
- Predictability ensured, development process simplified
- Possible to achieve performance benefits

**I/O analysis**
CONCLUSION

Summary and future work

- RTOS designed for performance isolation on multi-core
- Exploits specialization and scheduling of shared resources
- Predictability ensured, development process simplified
- Possible to achieve performance benefits

I/O analysis

Relax strict TDMA
CONCLUSION
Summary and future work

- RTOS designed for performance isolation on multi-core
- Exploits specialization and scheduling of shared resources
- Predictability ensured, development process simplified
- Possible to achieve performance benefits

I/O analysis
Relax strict TDMA
Safety mechanisms
Thanks.

Renato Mancuso
rmancus2@illinois.edu