A Reliable and Predictable Scratchpad-Centric OS for Multi-Core Embedded Systems

Rohan Tabish*, Renato Mancuso*, Saud Wasly**, Sujit S. Phatak+, Rodolfo Pellizzoni** and Marco Caccamo*
MOTIVATION
In the **automotive** domain, new platforms feature a set of hardware characteristics for **real-time** applications.
In the automotive domain, new platforms feature a set of hardware characteristics for real-time applications.

- Multiple general-purpose cores
In the automotive domain, new platforms feature a set of hardware characteristics for real-time applications.

- Multiple general-purpose cores
- Static priority / round-robin policy at memory bus
In the automotive domain, new platforms feature a set of hardware characteristics for 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 feature a set of hardware characteristics for 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
In the automotive domain, new platforms feature a set of hardware characteristics for 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
In the automotive domain, new platforms feature a set of hardware characteristics for 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
In the automotive domain, new platforms feature a set of hardware characteristics for 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
- Support for error correction codes (ECC)
In the **automotive** domain, new platforms feature a set of hardware characteristics for **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
- Support for error correction codes (ECC)
In the automotive domain, new platforms feature a set of hardware characteristics for real-time applications. These platforms are designed with specific motivations in mind:

- 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
- Support for error correction codes (ECC)

**Scratchpad-based Platforms with Error Detection**

1. Predictability.
2. Reliability.
Consider Freescale MPC5777M
Consider Freescale MPC5777M

- **Core 1**
- **Core m**
- **I/O Core**

**Memory Bus**
- **Main Mem**
- **FLASH**

**Devices**

**I/O Bus**

**2 application cores**
Consider Freescale MPC5777M

MOTIVATION

2 application cores

500 KB
Consider Freescale MPC5777M

- **Core 1**
  - Memory Bus
  - Main Mem.
  - SPM
  - ECC

- **Core m**
  - Memory Bus
  - FLASH
  - SPM
  - ECC

- **I/O Core**
  - I/O Bus
  - Devices
  - SPM
  - ECC

- **Main Mem.**
  - 500 KB

- **SPM**
  - 90 KB each

- **2 application cores**
Consider Freescale MPC5777M

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

- **Core 1**
- **Core m**
- **I/O Core**
- **Main Mem.**
- **FLASH**
- **Devices**
- **Memory Bus**
- **I/O Bus**
- **SPM**
- **ECC**

**Motivation**

- 2 application cores
- 500 KB
- 90 KB each
- 300 MHz
- SEC-DED
**MOTIVATION – UNBOUNDED CONTENTION**

**Synthetic Benchmarks – on Freescale MPC5777M**

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

Legend:
- **1 Active Core**
- **3 Active Cores**
MOTIVATION – UNBOUNDED CONTENTION

Synthetic Benchmarks – on Freescale MPC5777M

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

~ 3.5

~ half a billion cars in 2010
>

> a billion cars by 2020

~ half a billion cars in 2010

> a billion cars by 2020

About 1 single-bit soft error per SoC every 8000000 hours
~ half a billion cars in 2010

> a billion cars by 2020

About 1 single-bit soft error per SoC every 8000000 hours

~ 5000 cars per day affected by an error


~ half a billion cars in 2010

~ 5000 cars per day affected by an error

> a billion cars by 2020
1. Half a billion cars in 2010
2. More than a billion cars by 2020
3. Approximately 5000 cars per day affected by an error

ERROR-Correcting Codes

- 1-bit detection (parity)
- 1-bit correction + 2-bit detection (SEC-DED)
- 2-bit correction (DEC)

MOTIVATION – ERROR RECOVERY

RTOS Design Approach
RTOS Design Approach

AUGMENT EXISTING RTOS
RTOS Design Approach

AUGMENT EXISTING RTOS

ALWAYS EXECUTE TASKS FROM SPM
do not allow tasks to directly access shared memory resources as they execute.
RTOS Design Approach

AUGMENT EXISTING RTOS

**ERIKA ENTERPRISE**

ALWAYS EXECUTE TASKS FROM SPM

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

Upon Uncorrectable Error Detection

perform appropriate recovery mechanism depending upon error location.
SPM-CENTRIC OS

Core
  Scratchpad

DMA
Mem. Bus

Main Memory
Task 2

activation
activation
SPM-CENTRIC OS

Core
  \hspace{1.5cm} Scratchpad
  \hspace{2.5cm} Task 2

DMA
  \hspace{1.5cm} Mem. Bus

Main Memory
  \hspace{1.5cm} Task 2

load to SPM (DMA)

activation

SPM
SPM-CENTRIC OS

Core
  Scratchpad
  Task 2

DMA
Mem. Bus
Main Memory
  Task 2

load to SPM (DMA)
activation

execution
memory
SPM-CENTRIC OS

Task 2 execution

Core
  Scratchpad
  Task 2

DMA

Mem. Bus

Main Memory
  Task 2

Load to SPM (DMA)

Execution in SPM (CPU)

Activation
SPM-CENTRIC OS

Core
- Scratchpad
- Task 2

Main Memory
- Mem. Bus
- Task 2

DMA

Task: 2 Mem. Bus

Activation

Load to SPM
(DMA)

Execution in SPM
(CPU)
SPM-CENTRIC OS

3 stages, 2 resources
multi-stage (a.k.a. flow-shop) task
SPM-CENTRIC OS

Core

Scratchpad

P1

P2
SPM-CENTRIC OS
1. Task 1 executing from local SPM
1. **Task 1** executing from local SPM

2. **Task 2** becomes ready
1. Task 1 executing from local SPM
2. Task 2 becomes ready
3. DMA loads Task 2 image into SPM
1. Task 1 executing from local SPM
2. Task 2 becomes ready
3. DMA loads Task 2 image into SPM
4. When DMA completes, Task 2 ready
1. Task 1 executing from local SPM
2. Task 2 becomes ready
3. DMA loads Task 2 image into SPM
4. When DMA completes, Task 2 ready
5. When Task 1 completes, context switch
1. Task 1 executing from local SPM
2. Task 2 becomes ready
3. DMA loads Task 2 image into SPM
4. When DMA completes, Task 2 ready
5. When Task 1 completes, context switch
6. DMA unloads Task 1 image to SRAM
1. Task 1 executing from local SPM
2. Task 2 becomes ready
3. DMA loads Task 2 image into SPM
4. When DMA completes, Task 2 ready
5. When Task 1 completes, context switch
6. DMA unloads Task 1 image to SRAM
1. Task 1 executing from local SPM
2. Task 2 becomes ready
3. DMA loads Task 2 image into SPM
4. When DMA completes, Task 2 ready
5. When Task 1 completes, context switch
6. DMA unloads Task 1 image to SRAM
Core Scratchpad
P1
P2
Main Memory
DMA
Task 1
Task 2
Mem. Bus

1. Task 1 executing from local SPM
2. Task 2 becomes ready
3. DMA loads Task 2 image into SPM
4. When DMA completes, Task 2 ready
5. When Task 1 completes, context switch
6. DMA unloads Task 1 image to SRAM

SPM-CENTRIC OS

Core 0
Load/Unload
Core 1
Load/Unload
Core 0
Load/Unload

Memory Bus

Main Memory

DMA

Scratchpad

≥ max(\(L_{\text{max}}\), \(U_{\text{max}}\))

DMA slot size
SPM-CENTRIC OS + ERROR RECOVERY
Integrate with hardware. Rely on detection capabilities to perform recovery in the OS.
SOFTWARE RECOVERY

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS.

| Flash Memory | Main Memory | Local Memory (SPM) |
Integrate with hardware. Rely on detection capabilities to perform recovery in the OS.

SPM-CENTRIC OS + ERROR RECOVERY

SOFTWARE RECOVERY

Flash Memory
- Task i
  - Read Only

Main Memory

Local Memory (SPM)
 SOFTWARE RECOVERY

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS

1. Divide task memory into Read Only and Read/Write part
SOFTWARE RECOVERY

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task

Flash Memory
- Task $i$
  - Read Only

Main Memory
- Task $i$
  - Read Only
  - Read/Write copy #1
  - Read/Write copy #2

Local Memory (SPM)

SPM-CENTRIC OS + ERROR RECOVERY
SOFTWARE RECOVERY

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task

Flash Memory

Task $i$
Read Only

Main Memory

Task $i$
Read Only
Read/Write copy #1
Read/Write copy #2

Local Memory (SPM)

load to SPM (DMA)
**SPM-CENTRIC OS + ERROR RECOVERY**

**SOFTWARE RECOVERY**

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS.

1. Under normal use the Read Only copy and Read/Write Copy #1 for load.

- **Flash Memory**
  - Task $i$
    - Read Only

- **Main Memory**
  - Task $i$
    - Read Only
    - Read/Write copy #1
    - Read/Write copy #2

- **Local Memory (SPM)**
  - Task $i$
    - Read Only
    - Read/Write
SOFTWARE RECOVERY

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS

1. Under normal use the Read Only copy and Read/Write Copy #1 for load

**load to SPM (DMA)**

**execution in SPM (CPU)**

---

**Flash Memory**

- Task $i$
  - Read Only

**Main Memory**

- Task $i$
  - Read Only
  - Read/Write copy #1
  - Read/Write copy #2

**Local Memory (SPM)**

- Task $i$
  - Read Only
  - Read/Write

---

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task
SOFTWRE RECOVERY

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS.

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task

Task i
- Read Only

Task i
- Read Only
- Read/Write copy #1
- Read/Write copy #2

Task i
- Read Only
- Read/Write

1. Under normal use the Read Only copy and Read/Write Copy #1 for load
2. At unload, download R/W part of task from SPM to update R/W copy #1

load to SPM (DMA)

execution in SPM (CPU)

unload from SPM to copy 1 (DMA)

SPM-CENTRIC OS + ERROR RECOVERY

SOFTWARE RECOVERY

Flash Memory

Main Memory

Local Memory (SPM)
SPM-CENTRIC OS + ERROR RECOVERY

SOFTWARE RECOVERY

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS.

Flash Memory

Main Memory

Local Memory (SPM)

1. Under normal use the Read Only copy and Read/Write Copy # 1 for load
2. At unload, download R/W part of task from SPM to update R/W copy # 1
3. Download R/W part of task using DMA from SPM to update R/W copy # 2

Task $i$

Task $i$

Task $i$

Read Only

Read Only

Read Only

Read/Write copy #1

Read/Write copy #2

Read/Write

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task
1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task
3. Under normal use the Read Only copy and Read/Write Copy #1 for load
4. At unload, download R/W part of task from SPM to update R/W copy #1
5. Download R/W part of task using DMA from SPM to update R/W copy #2

Software Recovery

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS.

Flash Memory
- Task i
  - Read Only

Main Memory
- Task i
  - Read Only
  - Read/Write Copy #1
  - Read/Write Copy #2

Local Memory (SPM)
- Task i
  - Read Only
  - Read/Write

Load to SPM (DMA)
- Execution in SPM (CPU)
- Unload from SPM to Copy 1 (DMA)
- Unload from SPM to Copy 2 (DMA)
**SOFTWARE RECOVERY**

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS.

**SPM-CENTRIC OS + ERROR RECOVERY**

1. **Error at load**
   - Load to SPM (DMA)
   - Execution in SPM (CPU)
   - Unload from SPM to copy 1 (DMA)
   - Unload from SPM to copy 2 (DMA)

2. **Have two copies of same task**

   - Divide task memory into Read Only and Read/Write part
   - Under normal use the Read Only copy and Read/Write Copy # 1 for load
   - At unload, download R/W part of task from SPM to update R/W copy # 1
   - Download R/W part of task using DMA from SPM to update R/W copy # 2

**Flash Memory**

- Task i
  - Read Only

**Main Memory**

- Task i
  - Read Only
  - Read/Write copy #1
  - Read/Write copy #2

**Local Memory (SPM)**

- Task i
  - Read Only
  - Read/Write
SPM-CENTRIC OS + ERROR RECOVERY

SOFTWARE RECOVERY

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task

1. Under normal use the Read Only copy and Read/Write Copy # 1 for load
2. At unload, download R/W part of task from SPM to update R/W copy # 1
3. Download R/W part of task using DMA from SPM to update R/W copy # 2

1. Error at load
2. Error at Execution

Task i

Flash Memory

Main Memory

Local Memory (SPM)

Read Only

Read Only

Read/Write copy #1

Read/Write copy #2

Read Only

Read/Write
SOFTWARE RECOVERY

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task

Task $i$

Flash Memory
Read Only

Main Memory
Task $i$
Read Only
Read/Write copy #1
Read/Write copy #2

Local Memory (SPM)
Task $i$
Read Only
Read/Write

1. Under normal use the Read Only copy and Read/Write Copy # 1 for load
2. At unload, download R/W part of task from SPM to update R/W copy # 1
3. Download R/W part of task using DMA from SPM to update R/W copy # 2

load to SPM (DMA)

execution in SPM (CPU)

unload from SPM to copy 1 (DMA)
unload from SPM to copy 2 (DMA)

1. Error at load
2. Error at Execution
3. Error at Unload
SOFTWARE RECOVERY

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS.

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task

SPM-CENTRIC OS + ERROR RECOVERY

- Task $i$
  - Read Only
  - Read/Write copy #1
  - Read/Write copy #2

Main Memory

- Task $i$
  - Read Only

Local Memory (SPM)
1. An error is either in the Read only memory or Read/Write Memory of task.

2. Have two copies of same task

1. Divide task memory into Read Only and Read/Write part

SOFTWARE RECOVERY
Integrate with hardware. Rely on detection capabilities to perform recovery in the OS
1. An error is either in the Read only memory or Read/Write Memory of task.
2. Error is detected when DMA finishes loading.

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task

SOFTWARE RECOVERY
Integrate with hardware. Rely on detection capabilities to perform recovery in the OS

load to SPM (DMA)
1. Error at load
**SPM-CENTRIC OS + ERROR RECOVERY**

**SOFTWARE RECOVERY**

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS

1. Error at load

- **Flash Memory**
  - Task *i*
    - Read Only

- **Main Memory**
  - Task *i*
    - Read Only
    - Read/Write copy #1
    - Read/Write copy #2

- **Local Memory (SPM)**
  - Task *i*
    - Read Only

1. An error is either in the Read only memory or Read/Write Memory of task.
2. Error is detected when DMA finishes loading.
3. The I/O core copies the word that caused error either from flash or from the second copy R/W copy to the main memory as well as to the SPM.

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task
SOFTWARE RECOVERY

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS

Flash Memory

Task $i$

Read Only

Main Memory

Task $i$

Read Only

Read/Write copy #1

Read/Write copy #2

Local Memory (SPM)

load to SPM (DMA)

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task
SPM-CENTRIC OS + ERROR RECOVERY

SOFTWARE RECOVERY
Integrate with hardware. Rely on detection capabilities to perform recovery in the OS

load to SPM (DMA)

Flash Memory
- Task i
  - Read Only

Main Memory
- Task i
  - Read Only
  - Read/Write copy #1
  - Read/Write copy #2

Local Memory (SPM)
- Task i
  - Read Only
  - Read/Write

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task
SOFTWARE RECOVERY

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS.

<table>
<thead>
<tr>
<th>Flash Memory</th>
<th>Main Memory</th>
<th>Local Memory (SPM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task $i$</td>
<td>Task $i$</td>
<td>Task $i$</td>
</tr>
<tr>
<td>Read Only</td>
<td>Read Only</td>
<td>Read Only</td>
</tr>
<tr>
<td></td>
<td>Read/Write</td>
<td>Read/Write</td>
</tr>
<tr>
<td></td>
<td>copy #1</td>
<td>copy #2</td>
</tr>
</tbody>
</table>

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of the same task

load to SPM (DMA)

execution in SPM (CPU)
SPM-CENTRIC OS + ERROR RECOVERY

SOFTWARE RECOVERY

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS

1. Task successfully loaded, error happens after load but before completion.

load to SPM (DMA)

execution in SPM (CPU)

2. Error at Execution

Flash Memory

Task i
Read Only

Main Memory

Task i
Read Only
Read/Write copy #1
Read/Write copy #2

Local Memory (SPM)

Task i
Read Only
Read/Write

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task

SOFTWARE RECOVERY

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS

1. Task successfully loaded, error happens after load but before completion.

load to SPM (DMA)

execution in SPM (CPU)

2. Error at Execution

Flash Memory

Task i
Read Only

Main Memory

Task i
Read Only
Read/Write copy #1
Read/Write copy #2

Local Memory (SPM)

Task i
Read Only
Read/Write

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task
1. Task successfully loaded, error happens after load but before completion.
2. The application core marks partition as empty.

SOFTWARE RECOVERY
Integrate with hardware. Rely on detection capabilities to perform recovery in the OS.

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task

Flash Memory
Task $i$
- Read Only

Main Memory
Task $i$
- Read Only
- Read/Write copy #1
- Read/Write copy #2

Local Memory (SPM)
Task $i$
- Read Only
- Read/Write

load to SPM (DMA)
execution in SPM (CPU)

Error
E!

1. Task successfully loaded, error happens after load but before completion.
2. The application core marks partition as empty.
1. Task successfully loaded, error happens after load but before completion.
2. The application core marks partition as empty.
3. Reschedule tasks with highest priority.

SOFTWARE RECOVERY
Integrate with hardware. Rely on detection capabilities to perform recovery in the OS.

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task

Flash Memory
Task $i$
Read Only

Main Memory
Task $i$
Read Only
Read/Write copy #1
Read/Write copy #2

Local Memory (SPM)
Task $i$
Read Only
Read/Write

2. Error at Execution

load to SPM (DMA)
execution in SPM (CPU)
Integrate with hardware. Rely on detection capabilities to perform recovery in the OS.

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task
3. Error at Unload
SPM-CENTRIC OS + ERROR RECOVERY

SOFTWARE RECOVERY
Integrate with hardware. Rely on detection capabilities to perform recovery in the OS

1. Task successfully completed and ready to be downloaded.

2. Have two copies of same task

3. Error at Unload

Flash Memory
- Task i
  - Read Only

Main Memory
- Task i
  - Read Only
  - Read/Write copy #1
  - Read/Write copy #2

Local Memory (SPM)
- Task i
  - Read Only
  - Read/Write

1. Divide task memory into Read Only and Read/Write part

SOFTWARE RECOVERY
- load to SPM (DMA)
- execution in SPM (CPU)
1. Task successfully completed and ready to be downloaded.
2. DMA downloads R/W section to R/W copy #1.

3. Error at Unload

SOFTWARE RECOVERY
Integrate with hardware. Rely on detection capabilities to perform recovery in the OS

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task

SPM-CENTRIC OS + ERROR RECOVERY

Flash Memory
Task $i$
- Read Only

Main Memory
Task $i$
- Read Only
- Read/Write copy #1
- Read/Write copy #2

Local Memory (SPM)
Task $i$
- Read Only
- Read/Write
- E!
SPM-CENTRIC OS + ERROR RECOVERY

SOFTWARE RECOVERY
Integrate with hardware. Rely on detection capabilities to perform recovery in the OS

Flash Memory
Task i
Read Only

Main Memory
Task i
Read Only
Read/Write copy #1
Read/Write copy #2

Local Memory (SPM)
Task i
Read Only
Read/Write

1. Task successfully completed and ready to be downloaded.
2. DMA downloads R/W section to R/W copy #1.
3. At error application core marks SPM partition empty and reschedules task with highest priority using second copy.

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task
3. Error at Unload
unload from SPM to copy 1 (DMA)
SPM-CENTRIC OS + ERROR RECOVERY

SOFTWARE RECOVERY

Integrate with hardware. Rely on detection capabilities to perform recovery in the OS.

Flash Memory

- Task $i$
  - Read Only

Main Memory

- Task $i$
  - Read Only
  - Read/Write copy #1
  - Read/Write copy #2

Local Memory (SPM)

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task
3. Error at Unload

load to SPM (DMA)

execution in SPM (CPU)

unload from SPM to copy 1 (DMA)
SPM-CENTRIC OS + ERROR RECOVERY

SOFTWARE RECOVERY
Integrate with hardware. Rely on detection capabilities to perform recovery in the OS

1. Task successfully completed and error happens after first download.

Flash Memory
- Task $i$
  - Read Only

Main Memory
- Task $i$
  - Read Only
  - Read/Write copy #1
  - Read/Write copy #2

Local Memory (SPM)
- Task $i$
  - Read Only
  - Read/Write
    - E!

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task
3. Error at Unload
1. Task successfully completed and error happens after first download.
2. Use first copy for next time. No need to reschedule the task.

3. Error at Unload
   - unload from SPM to copy 1 (DMA)
   - unload from SPM to copy 2 (DMA)

SOFTWARE RECOVERY
Integrate with hardware. Rely on detection capabilities to perform recovery in the OS.

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task

**SPM-CENTRIC OS + ERROR RECOVERY**

- **Flash Memory**
  - Task $i$
    - Read Only

- **Main Memory**
  - Task $i$
    - Read Only
    - Read/Write copy #1
    - Read/Write copy #2

- **Local Memory (SPM)**
  - Task $i$
    - Read Only
    - Read/Write
Integrate with hardware. Rely on detection capabilities to perform recovery in the OS.

1. Divide task memory into Read Only and Read/Write part
2. Have two copies of same task
3. Error at Unload

Worst case occurs when error happens as late as possible.
WORST-CASE RESPONSE TIME

from task release to end of unload phase
WORST-CASE RESPONSE TIME
from task release to end of unload phase

SCHEDULABILITY ANALYSIS

SCHEDULING INTERVAL
1. only one blocking or interfering task runs
2. starts with CPU execution
3. ends when CPU completes execution, or when the load phase for next task completes
ANALYSIS APPROACH

[Diagram showing time on the x-axis and task priority on the y-axis. The diagram includes symbols for DMA load, DMA unload, and other core's slot.]

DMA load
DMA unload
other core's slot
ANALYSIS APPROACH
ANALYSIS APPROACH

τ₁, τ₂, τ₃, τ₄, τ₅

DMA load
DMA unload
other core's slot
ANALYSIS APPROACH

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

- Analysis approach
- Under analysis
under analysis

ANALYSIS APPROACH
ANALYSIS APPROACH

\[ \tau_1 \]

\[ \tau_2 \]

\[ \tau_3 \]

\[ \tau_4 \]

\[ \tau_5 \]

DMA load

DMA unload

X other core’s slot

Under analysis

task

priority

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 28 29 30 31 32 33 34 35 36 37 38 39
ANALYSIS APPROACH

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

Interval 1
Interval 2
Interval 3

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

Time

DMA load
DMA unload
Other core's slot
ANALYSIS APPROACH

Task priority

DMA unload
DMA load
Other core's slot

Interval 1
Interval 2
Interval 3
Interval 4

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

Under analysis

Time

Task priority

DMA load
DMA unload
Other core's slot
ANALYSIS APPROACH

<table>
<thead>
<tr>
<th>Task</th>
<th>Priority</th>
<th>DMA unload</th>
<th>DMA load</th>
<th>other core's slot</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₁</td>
<td></td>
<td></td>
<td>▲</td>
<td>X</td>
</tr>
<tr>
<td>τ₂</td>
<td></td>
<td></td>
<td>▼</td>
<td>X</td>
</tr>
<tr>
<td>τ₃</td>
<td></td>
<td></td>
<td>▲</td>
<td>X</td>
</tr>
<tr>
<td>τ₄</td>
<td></td>
<td></td>
<td>▼</td>
<td>X</td>
</tr>
<tr>
<td>τ₅</td>
<td></td>
<td></td>
<td>▲</td>
<td>X</td>
</tr>
</tbody>
</table>

τ₁, τ₂, τ₃, τ₄, and τ₅ are tasks under analysis. Interval 1, Interval 2, Interval 3, and Interval 4 represent different time periods.
ANALYSIS APPROACH

Task priority

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

DMA load
DMA unload
Other core’s slot

Under analysis

Interval 1
Interval 2
Interval 3
Interval 4
Interval F

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 28 29 30 31 32 33 34 35 36 37 38 39
ANALYSIS APPROACH

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

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

- Time

- Interval 1
  - Task τ₁
  - Task τ₂
  - Task τ₃
  - Task τ₄
  - Task τ₅

- Interval 2

- Analysis

- DMA load
- DMA unload
- Other core's slot
critical instant:

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 low priority task starts loading.
critical instant:

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

Task priority

DMA unload
DMA load
other core's slot

\( \tau \)

Interval 1

B

0 1 2 3 4

high priority

under analysis

low priority

task priority

DMA load
DMA unload
other core's slot

time
ANALYSIS APPROACH

Blocking B

<table>
<thead>
<tr>
<th>Task Priority</th>
<th>DMA Load</th>
<th>DMA Unload</th>
<th>Other Core's Slot</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Under analysis</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

Interval 1

B

0 1 2 3 4

time
ANALYSIS APPROACH

Blocking B

Length of interval 1 - $\sigma$
ANALYSIS APPROACH

Blocking B

Length of interval 1 - $\sigma$

$\leq \max(C_5, 2\sigma) - \sigma$
ANALYSIS APPROACH

Busy Interval H

<table>
<thead>
<tr>
<th>Task</th>
<th>Priority</th>
<th>DMA unload</th>
<th>DMA load</th>
<th>Other core's slot</th>
</tr>
</thead>
<tbody>
<tr>
<td>τ₁</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>τ₂</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>τ₃</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>τ₄</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>τ₅</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Time:
5  6  7  8  9  10  11  12  13  14  15  16  17  18

Interval 2  Interval 3  Interval 4

τ₁  τ₂  τ₃  τ₄  τ₅

DMA load
DMA unload
X other core's slot
ANALYSIS APPROACH

Busy Interval $H$

if $C > 4\sigma$:

interval $\leq \max(C, 5\sigma)$
Analysys Approach

Busy Interval \( H \)

if \( C > 4 \sigma \):
interval \( \leq \max(C, 5\sigma) \)

else:
interval \( \leq 4\sigma \)
**ANALYSIS APPROACH**

![Diagram showing task priorities and intervals](image)

- **Task Priority**
  - DMA unload
  - DMA load
  - Other core's slot

- **Interval F**
  - τ₁
  - τ₂
  - τ₃
  - τ₄
  - τ₅

- **Max between:**
  \[ C + 5\sigma \]

- **Final Interval F**

**Notes:**
- Under analysis
- Time axis: 19, 20, 21, 22, 23, 24, 25, 26, 27

**Legend:**
- DMA load
- DMA unload
- Other core's slot

---

**Additional Information:**
- Interval F calculation involves statistical measures such as mean (C) and standard deviation (σ).
- The diagram illustrates how tasks are prioritized and scheduled over time.
ANALYSIS APPROACH

Final Interval F

Max between: $C + 5\sigma$

and

$7\sigma$
Response Time With No Error
Response Time of Task
Under Analysis
Response Time of Task Under Analysis
Response Time of Task
Under Analysis

\[ = B + H + F \]
Response Time With Error Recovery
Assumption
Assumption
Assumption

No more than one error can occur for two consecutive period of same task.
Assumption

No more than one error can occur for two consecutive period of same task.

Memory Error Classification Based on when it Occurs
Assumption

No more than one error can occur for two consecutive period of same task.

Memory Error Classification
Based on when it Occurs
Assumption

No more than one error can occur for two consecutive period of same task.

Memory Error Classification
Based on when it Occurs

Error Recovery Prior to the Final Interval
**Assumption**

No more than one error can occur for two consecutive period of same task.

**Memory Error Classification**
Based on when it Occurs

- Error Recovery Prior to the Final Interval
- Error Recovery in the Final Interval

**Response Time With Error Recovery**
Error Recovery Prior to Final Interval

(task)

priority

DMA unload
DMA load
other core's slot

time
Error Recovery Prior to Final Interval

- Task priority
- DMA load
- DMA unload
- Other core's slot
Error Recovery Prior to Final Interval

-DMA unload
-DMA load
-other core’s slot
Error Recovery Prior to Final Interval

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

Under analysis

Task 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 28 29 30 31 32 33 34 35 36 37 38 39
Error Recovery Prior to Final Interval

Diagram showing the timeline with tasks τ₁, τ₂, τ₃, τ₄, and τ₅. The diagram includes DMA load and DMA unload events, as well as other core's slots marked with 'X'. The timeline is divided into intervals for analysis.
Error Recovery Prior to Final Interval

- Task priority
- Under analysis

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

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

Time range: 0 to 39

Legend:
- DMA load
- DMA unload
- Other core's slot
Error Recovery Prior to Final Interval
Error Recovery Prior to Final Interval

Interval 1

τ₁

τ₂

τ₃

τ₄

τ₅

under analysis

task priority

DMA load

DMA unload

x other core's slot

Error Recovery Prior to Final Interval

Interval 1

τ₁

τ₂

τ₃

τ₄

τ₅

under analysis

task priority

DMA load

DMA unload

x other core's slot

Error Recovery Prior to Final Interval

Interval 1

τ₁

τ₂

τ₃

τ₄

τ₅

under analysis

task priority

DMA load

DMA unload

x other core's slot

Error Recovery Prior to Final Interval

Interval 1

τ₁

τ₂

τ₃

τ₄

τ₅

under analysis

task priority

DMA load

DMA unload

x other core's slot
Error Recovery Prior to Final Interval

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

Interval 1

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

Under analysis

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 28 29 30 31 32 33 34 35 36 37 38 39
Error Recovery Prior to Final Interval

- **Task Priority**
  - \(\tau_1\)
  - \(\tau_2\)
  - \(\tau_3\)
  - \(\tau_4\)
  - \(\tau_5\)

- **Under Analysis**

- **DMA Load**
- **DMA Unload**
- **Other Core’s Slot**

- **Interval 1**

- **Time**
Error Recovery Prior to Final Interval

- \( \tau_1 \) under analysis
- \( \tau_2 \) task
- \( \tau_3 \) priority
- \( \tau_4 \) DMA load
- \( \tau_5 \) DMA unload
- X other core's slot

Interval 1
- E Error Recovery
- R Other Core's Slot

Interval 2
- X DMA load
- X DMA unload
Error Recovery Prior to Final Interval

- **Task Priority**:DMA unload
- **DMA Load**: DMA unload
- **Other Core's Slot**: Other core's slot

**Interval 1**
- **Task 1 (τ1)**: under analysis
- **Task 2 (τ2)**: under analysis
- **Task 3 (τ3)**: under analysis
- **Task 4 (τ4)**: under analysis
- **Task 5 (τ5)**: under analysis

**Interval 2**
- **Task 1 (τ1)**: under analysis
- **Task 2 (τ2)**: under analysis
- **Task 3 (τ3)**: under analysis
- **Task 4 (τ4)**: under analysis
- **Task 5 (τ5)**: under analysis

**Error Recovery Prior to Final Interval**

- **E**: Error
- **R**: Recovery

**Time**

Error Recovery Prior to Final Interval

Interval 1

Interval 2

\( \tau_1 \)

\( \tau_2 \)

\( \tau_3 \)

\( \tau_4 \)

\( \tau_5 \)

DMA load

DMA unload

other core’s slot

under analysis

Error Recovery Prior to Final Interval
Error Recovery Prior to Final Interval

- Task priority
- Under analysis

- Interval 1
  - Interval 2
  - Interval 3

- Error Recovery
  - Prior to Final Interval

- DMA load
- DMA unload
- Other core's slot
Error Recovery Prior to Final Interval

Interval 1

Interval 2

Interval 3

\( \tau_1 \)

\( \tau_2 \)

under analysis

\( \tau_3 \)

\( \tau_4 \)

\( \tau_5 \)

DMA load

DMA unload

other core's slot

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 28 29 30 31 32 33 34 35 36 37 38 39

time
Error Recovery Prior to Final Interval

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

- DMA load and DMA unload
  - DMA load
  - DMA unload

- Other core's slot

- Time intervals:
  - Interval 1
  - Interval 2
  - Interval 3
  - Interval 4

- Analysis under

- Error Recovery Prior to Final Interval
Error Recovery Prior to Final Interval

- **Task Priority**
- **Under Analysis**

**Task Priority**

- **τ₁**
- **τ₂**
- **τ₃**
- **τ₄**
- **τ₅**

**Interval 1**

- DMA load

**Interval 2**

- DMA unload

**Interval 3**

- DMA load

**Interval 4**

- DMA unload

**Time**
Error Recovery Prior to Final Interval

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

Interval 1: 
- \( \tau_1 \) under analysis
-\( \tau_2 \)
- \( \tau_3 \)
- \( \tau_4 \)
- \( \tau_5 \)

Interval 2:
- \( \tau_1 \)
- \( \tau_2 \)
- \( \tau_3 \)
- \( \tau_4 \)
- \( \tau_5 \)

Interval 3:
- \( \tau_1 \)
- \( \tau_2 \)
- \( \tau_3 \)
- \( \tau_4 \)
- \( \tau_5 \)

Interval 4:
- \( \tau_1 \)
- \( \tau_2 \)
- \( \tau_3 \)
- \( \tau_4 \)
- \( \tau_5 \)

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 28 29 30 31 32 33 34 35 36 37 38 39
Error Recovery Prior to Final Interval

- Task priority
- Time
- DMA load
- DMA unload
- Other core's slot

Under analysis

Interval 1
Interval 2
Interval 3
Interval 4
Interval 5

Error Recovery Prior to Final Interval
Error Recovery Prior to Final Interval

\begin{itemize}
  \item \textbf{Interval 1:} DMA unload
  \item \textbf{Interval 2:} DMA load
  \item \textbf{Interval 3:} other core's slot
  \item \textbf{Interval 4:} under analysis
  \item \textbf{Interval 5:} task priority
\end{itemize}

\begin{itemize}
  \item \textbf{τ₁}: Error Recovery Prior to Final Interval
  \item \textbf{τ₂}: under analysis
  \item \textbf{τ₃}: task priority
  \item \textbf{τ₄}: DMA load
  \item \textbf{τ₅}: DMA unload
\end{itemize}
Error Recovery Prior to Final Interval

- **τ₁**: Interval 1
- **τ₂**: Interval 2
- **τ₃**: Interval 3
- **τ₄**: Interval 4
- **τ₅**: Interval 5

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

### Timeline

- **Interval 1**: τ₁
- **Interval 2**: τ₂
- **Interval 3**: τ₃
- **Interval 4**: τ₄
- **Interval 5**: τ₅

- Time markers from 0 to 39
Error Recovery Prior to Final Interval

![Diagram showing task priority and DMA operations over time intervals.]

- **Task Priority:** $	au_1, 	au_2, 	au_3, 	au_4, 	au_5$
- **DMA Operations:** DMA load, DMA unload
- **Other Core's Slot:** X
- **Time Intervals:** Interval 1 to Interval F
- **Error Recovery:** E
- **Under Analysis:** Core under analysis

**Legend:**
- DMA load → ▲
- DMA unload → ▼
- Other core's slot → X

---

**Intervals Overview:**
- **Interval 1**: [0, 5]
- **Interval 2**: [6, 10]
- **Interval 3**: [11, 15]
- **Interval 4**: [16, 20]
- **Interval 5**: [21, 25]
- **Interval F**: [26, 30]

---

**Note:**
- DMA unload and DMA load operations indicated within time intervals.
- Core slot status shown at specific intervals.
Error Recovery Prior to Final Interval

Just adds one more interval during H – Interval 3

- Interval 1
- Interval 2
- Interval 3
- Interval 4
- Interval 5
- Interval F

Under analysis:

- Task 1
- Task 2
- Task 3
- Task 4
- Task 5

DMA load
DMA unload
Other core’s slot
Error Recovery Prior to Final Interval

Just adds one more interval during H – Interval 3
Error Recovery Prior to Final Interval
Response Time of Task Under Analysis
Response Time of Task Under Analysis
Response Time of Task Under Analysis

Error Recovery Prior to Final Interval

\[ B + H_{rec} + F \]
Error Recovery In Final Interval

- DMA load
- DMA unload
- other core's slot
Error Recovery In Final Interval

- Task: DMA unload
- Priority: DMA load
- Other core’s slot: X

- Time: τ₁, τ₂, τ₄, τ₅
- Priority: low, high

- DMA load: ↑
- DMA unload: ↓
- Other core’s slot: X

- Graph showing error recovery in final interval with time from 0 to 39.
Error Recovery In Final Interval

- **DMA load**
- **DMA unload**
- **other core's slot**

<table>
<thead>
<tr>
<th>Time</th>
<th>low priority</th>
<th>high priority</th>
<th>under analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>21</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>23</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>24</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>34</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>35</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>36</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>37</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>38</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>39</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **τ₁**
- **τ₂**
- **τ₃**
- **τ₄**
- **τ₅**

- **Error Recovery In Final Interval**
Error Recovery In Final Interval

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

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

- Error Recovery
  - DMA load
  - DMA unload
  - Other core's slot
Error Recovery In Final Interval

- Task priority
- DMA unload
- DMA load
- Other core's slot
- Time

- $\tau_1$, $\tau_2$, $\tau_3$, $\tau_4$, $\tau_5$
- High priority
- Under analysis
- Low priority

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

- 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 28 29 30 31 32 33 34 35 36 37 38 39
Error Recovery In Final Interval

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

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

- Error recovery in final interval
  - Low priority
  - High priority
  - Under analysis

- DMA load
- DMA unload
- Other core's slot
Error Recovery In Final Interval

- High priority tasks: $\tau_1, \tau_2$
- Under analysis tasks: $\tau_3$
- Low priority tasks: $\tau_4, \tau_5$

Intervals:
- Interval 1: 0-5
- Interval 2: 6-10

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

Time:
- 0 to 39

Legend:
- $\uparrow$ DMA load
- $\downarrow$ DMA unload
- $\times$ other core's slot
Error Recovery In Final Interval

- **Task Priority:**
  - High priority: \( \tau_1 \)
  - Under analysis: \( \tau_2 \)
  - Low priority: \( \tau_3 \)

- **DMA Load/Unload:**
  - DMA load: \( \uparrow \)
  - DMA unload: \( \downarrow \)

- **Other Core's Slot:**
  - \( X \)

- **Interval:**
  - Interval 1
  - Interval 2

- **Time:**
  - 0 to 39

- **Error Recovery In Final Interval**

---

**Legend:**
- DMA load
- DMA unload
- Other core's slot
Error Recovery In Final Interval

- Interval 1

- Interval 2

- DMA load
- DMA unload
- other core’s slot

- τ1
- τ2
- τ3
- τ4
- τ5

- high priority
- under analysis
- low priority

- low priority

- task priority

- time
Error Recovery In Final Interval

- Task priority: DMA unload, DMA load, other core's slot
- Priority: high, under analysis, low
- Time intervals:
  - Interval 1
  - Interval 2
- Error recovery in final interval
- DMA load and DMA unload indicators
- Other core's slot indicators
Error Recovery In Final Interval

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

- Priority:
  - Low priority
  - Under analysis
  - High priority

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

- Recovery:
  - Error Recovery in Final Interval

- Noted:
  - DMA load
  - DMA unload
  - Other core's slot
Error Recovery In Final Interval

- Task priority: DMA unload and DMA load
- Under analysis: other core's slot

Diagram showing intervals and task priority:
- Interval 1: τ₁
- Interval 2: τ₂
- Interval 3: τ₃
- Interval 4: τ₄

- Time scale: 0 to 39

Symbols:
- DMA load: ↑
- DMA unload: ↓
- Other core's slot: X
Error Recovery In Final Interval

- Task priority
  - High priority
  - Under analysis
  - Low priority
- Task
  - $\tau_1$
  - $\tau_2$
  - $\tau_3$
  - $\tau_4$
  - $\tau_5$

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

- Interval 1
- Interval 2
- Interval 3
- Interval 4
- Interval F

- 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 28 29 30 31 32 33 34 35 36 37 38 39
Error Recovery In Final Interval

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

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

- Time:
  - Interval 1
  - Interval 2
  - Interval 3
  - Interval 4
  - Interval F

- Error Recovery:
  - Interval 4
  - Interval F
Error Recovery In Final Interval

- Task priority:
  - DMA load
  - DMA unload
- Priority:
  - High
  - Low
- Under analysis

### Tasks

- **τ₁**: Interval 1
- **τ₂**: Interval 2
- **τ₃**: Interval 3
- **τ₄**: Interval 4
- **τ₅**: Interval F

### Errors
- Interval 1: τ₁
- Interval 2: τ₂
- Interval 3: τ₃
- Interval 4: τ₄
- Interval F: τ₅

### Time

- Time intervals: 0 to 39
- Error recovery in final interval
Error Recovery In Final Interval

- **Task Priority**: DMA unload, DMA load, other core's slot
- **Priority Levels**: High, Under Analysis, Low
- **Intervals**: Interval 1, Interval 2, Interval 3, Interval 4, Interval F
- **Error Recovery**: In Final Interval

Diagram shows the timeline with tasks and their priority levels.
Error Recovery In Final Interval

- **Task Priority**: DMA unload, DMA load, other core's slot
- **Priority Levels**: Low, High
- **Interval Markers**: Interval 1, Interval 2, Interval 3, Interval 4, Interval F
- **Error Recovery**: Occurs in Final Interval
Error Recovery In Final Interval

- Task priority
  - high priority
  - under analysis
  - low priority
  - task priority

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

Interval 1
Interval 2
Interval 3
Interval 4
Interval F

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

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 28 29 30 31 32 33 34 35 36 37 38 39

time
Error Recovery In Final Interval

- **Task Priority**: DMA unload, DMA load, other core's slot
- **Interval**: 1, 2, 3, 4, F
- **Sub-Interval**: 1
- **Error Recovery**: E, R
- **Time**: 0 to 39
- **Markers**: DMA load (↑), DMA unload (↓), other core's slot (X)
Error Recovery In Final Interval
Error Recovery In Final Interval

- **τ₁, τ₂, τ₃, τ₄, τ₅**: Tasks with different priorities.
- **Interval 1 to Interval F**: Different intervals for task scheduling.
- **DMA load**: Indicates DMA load events.
- **DMA unload**: Indicates DMA unload events.
- **X**: Marks other core's slot.
- **Sub Interval 1, Sub Interval U**: Subintervals for detailed analysis.
- **E, R**: Indicate error and recovery stages.
- **time**: X-axis indicating time progression.
Error Recovery In Final Interval

- \( \tau_1 \) to \( \tau_5 \): Tasks under analysis
- High priority tasks
- DMA load and unload
- Other core's slot

Time intervals:
- Interval 1
- Interval 2
- Interval 3
- Interval 4
- Interval F

Error Recovery In Final Interval
Error Recovery In Final Interval

- Task priorities: DMA unload, DMA load, other core's slot
- Time intervals: Interval 1, Interval 2, Interval 3, Interval 4, Interval F
- Subintervals: Sub Interval 1, Sub Interval U
- Error recovery in final interval
Error Recovery In Final Interval

- **Task Priority:**
  - DMA unload
  - DMA load
  - Other core's slot

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

- **Sub Intervals:**
  - Sub Interval 1
  - Sub Interval U
  - Sub Interval 2

- **Error Recovery:**
  - E
  - R

- **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 28 29 30 31 32 33 34 35 36 37 38 39

- **Task Priority Levels:**
  - High priority
  - Under analysis
  - Low priority
Error Recovery In Final Interval

Interval $F_{rec}$

- $\tau_1$
  - Sub Interval 1
- $\tau_2$
  - Sub Interval U
  - Sub Interval 2
- $\tau_3$
- $\tau_4$
- $\tau_5$

- [X] DMA load
- [↓] DMA unload
- [X] other core's slot

Task priority:
- high priority
- under analysis
- low priority

Sub Interval U

Sub Interval 1

Sub Interval 2

Time:
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37

Error Recovery In Final Interval
Error Recovery In Final Interval

Interval $F_{rec}$

Sub Interval 1
Sub Interval U
Sub Interval 2

τ₁
τ₂
τ₃
τ₄
τ₅

E
R

Final Interval ($F_{rec}$)

DMA load
DMA unload
other core’s slot

18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37

low priority
under analysis
high priority

task priority
Error Recovery In Final Interval

Final Interval ($F_{rec}$) = SubInterval 1 + SubInterval 2 + SubInterval 3 – Overlapped Time
Error Recovery In Final Interval

Final Interval \((F_{\text{rec}})\)

\[ F_{\text{rec}} = \text{SubInterval 1} + \text{SubInterval 2} + \text{SubInterval 3} - \text{Overlapped Time} \]

Where:

\[ \text{SubInterval 1} = \max(C_i + 5\sigma, 7\sigma) \]

\[ \text{SubInterval U} = \max(C_u, 4\sigma) \]

\[ \text{SubInterval 2} = \max(C_i + 5\sigma, 7\sigma) \]

\[ \text{Overlapped Time} = 2\sigma \]
Error Recovery In Final Interval
Response Time of Task
Under Analysis
Response Time of Task Under Analysis
Response Time of Task Under Analysis

\[ = B + H + Frec \]
Max between

Response Time of Task Under Analysis
Max between

\[ \max \ = B + H_{REC} + F \]
Max between $B + Hrec + F$ and
Max between

\[ B + H_{rec} + F \]

and

\[ B + H + F_{rec} \]
1. EEMBC Benchmark
1. EEMBC Benchmark

2. Fake faults were injected.
1. EEMBC Benchmark

2. Fake faults were injected.

3. Overhead of the recovery was measured.
Acceptable trade-off for increased reliability

Instance of fault-tolerant co-scheduling

OS data structures are not protected

Results - Schedulability

![Graph showing schedulability vs utilization with three lines: blue, green, and red, indicating Contention, Our with no error recovery, and Our with error recovery.](image)
For task set with lower periods recovery cost dominates.

For longer periods recovery overhead is negligible.
RTOS with predictability and reliability was presented.

Predictability is ensured by running from local memories.

Reliability is ensured by redundancy and error detection.
RTOS with predictability and reliability was presented.

Predictability is ensured by running from local memories.

Reliability is ensured by redundancy and error detection.
RTOS with predictability and reliability was presented.

Predictability is ensured by running from local memories.

Reliability is ensured by redundancy and error detection.
Conclusion and Future Work

RTOS with predictability and reliability was presented.

Predictability is ensured by running from local memories.

Reliability is ensured by redundancy and error detection.
RTOS with predictability and reliability was presented.

Predictability is ensured by running from local memories.

Reliability is ensured by redundancy and error detection.
Conclusion and Future Work

RTOS with predictability and reliability was presented.

Predictability is ensured by running from local memories.

Reliability is ensured by redundancy and error detection.

Extend For Multiple Errors
Extend For More Cores
Faults in OS
Thanks

rtabish@illinois.edu