Timed Processors and WCET-Aware Code Management for Mixed-Criticality Systems

RePP 2014 Workshop, Grenoble, France
April 6, 2014

David Broman*
broman@eecs.berkeley.edu
UC Berkeley, USA
and
Linköping University, Sweden

* This talk highlights key aspects of two papers that will appear in RTAS 2014 (April 15-17, Berlin), authored by the following persons:

Michael Zimmer
David Broman
Chris Shafer
Edward A. Lee

Yooseong Kim
David Broman
Jian Cai
Aviral Shrivastava

Agenda

Part I
Mixed-Criticality Systems

Part II
The FlexPRET Processor Platform

Part III
WCET-Aware SPM Management
Part I
Mixed-Criticality Systems

Part II
The FlexPRET Processor Platform

Part III
WCET-Aware SPM Management

Cyber-Physical Systems (CPS)

Physical system (the plant)

Cyber system: Computation (embedded) + Networking

Sensors

Actuators

Automotive
Process Industry and Industrial Automation
Aerospace

broman@eecs.berkeley.edu
Modern Systems with Many Processor Platforms

Modern aircraft have many computer controlled systems
- Engine control
- Electric power control
- Radar system
- Navigation system
- Flight control
- Environmental control system
etc...

Modern cars have many ECU (Electronic Control Units)
- Airbag control
- Door control
- Electric power steering control
- Power train control
- Speed control
- Battery management.
etc.. Over 80 ECUs in a high-end model (Albert and Jones, 2010)

### Mixed-Criticality Systems

**Issues with too many processors**
- High cost
- Space and weight
- Energy consumption

**Required for Safety**
- Spatial isolation between tasks
- Temporal isolation between tasks (necessary to meet deadlines)

**Federated Approach**
Each processor has its own task

**Consolidate into fewer processors**
**Mixed-Criticality Systems**

**Part I**
Mixed-Criticality Systems

**Part II**
The FlexPRET Processor Platform

**Part III**
WCET-Aware SPM Management

---

**Issues with too many processors**
- High cost
- Space and weight
- Energy consumption

**Federated Approach**
Each processor has its own task

**Required for Safety**
- Spatial isolation between tasks
- Temporal isolation between tasks (necessary to meet deadlines)

...but such safety requirements are only needed for highly critical tasks

**Mixed-Criticality Challenge**
Reconcile the conflicting requirements of:
- Partitioning (for safety)
- Sharing (for efficient resource usage)
(Burns & Davis, 2013)

---

**Possible Approaches...**

**Real-time Operating System (RTOS) Using Software Scheduling**
- Efficient resource usage
- Standard hardware
  - Requires verification and certification of the RTOS
  - Hard to do timing analysis (caches, pipeline, preemption of tasks)

**Multi-core with Task Partitioning**
- Temporal and spatial isolation if no resource sharing
- Standard hardware
  - Resources are typically shared (cache coherence problem)
  - WCET analysis for multi-core is very difficult
  - Low resource utilization (cores do nothing after tasks finished computations)
Our solution

**FlexPRET Softcore**
- Fine-grained Multithreaded Processor Platform (thread interleaved) implemented on an FPGA
- Flexible schedule (1 to 8 active threads) and scheduling frequency (1, 1/2, 2/3, 1/4, 1/8 etc.)
- Hard real-time threads (HRTT) with predictable timing behavior
  - Thread-interleaved pipeline (no pipeline hazards)
  - **Scratchpad memory instead of cache**
- Soft real-time threads (SRTT) with cycle stealing from HRTT

**WCET-Aware Scratchpad Memory (SPM) Management**
- Automatic DMA transfer of code to SPM
- Optimal mapping for minimizing WCET

---

**Part I**
Mixed-Criticality Systems

**Part II**
The FlexPRET Processor Platform

**Part III**
WCET-Aware SPM Management

---

Related Work

**Software Scheduling for Mixed Criticality**
- Reservation-based partitioning, ARNIC 653
- First priority-based MC (Vestal, 2007)
- Sporadic task scheduling (Baruha and Vestal, 2008)
- Slack scheduling (Niz et al. 2009)
- Review of MC area, 168 references (Burns & David, 2013)

**WCET Analysis**
- WCET-aware compiler (Falk & Lukuciejewski, 2010)
- Detection of loop and infeasible paths (Gustafsson et al., 2006)
- Cache analysis (Ferdinand & Wilhelm, 1999)
- WCET Survey (Wilhelm et al., 2008)

**Scratchpad Memory Management**
- Average case SPM methods for SMM (Bai et al, 2013; Jung et al., 2010; Pabalkar et al. 2008; Baker et al., 2010)
- Static SPM WCET methods (Keinaorge 2008, Platzar 2012)
- SPM management at basic block level (Puaut & Pais, 2007)

**Predictable and Multithreaded Processors**
- PRET idea (Edwards and Lee, 2007)
- PTARM (Liu et al., 2012)
- Patmos (Schoeberl et al., 2011)
- JOP (Schoeberl, 2008)
- XMOS X1 (May, 2009)
- MERASA, MC on multicore (Ungerer, 2010)

Several EU projects related to Mixed-Criticality: MultiPARTES, Recomp, CERTAINTY, Proxima,...
Part II
The FlexPRET Processor Platform

Single threaded processor

- FlexPRET can execute 1 to 8 hardware threads concurrently
- 5-stage pipeline implemented in Chisel (HDL embedded in Scala)
- Synthesized on a Xilinx Virtex-5 FPGA (for prototyping)
- RISCV ISA

<table>
<thead>
<tr>
<th>TID</th>
<th>Addr.</th>
<th>Inst.</th>
<th>Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>1 2 3 4 5 6 7 8</td>
</tr>
<tr>
<td>0</td>
<td>0x00</td>
<td>BR 0x0C</td>
<td>F D E M W</td>
</tr>
<tr>
<td>0</td>
<td>0x04</td>
<td>I</td>
<td>F D - - -</td>
</tr>
<tr>
<td>0</td>
<td>0x08</td>
<td>I</td>
<td>F - - - -</td>
</tr>
<tr>
<td>0</td>
<td>0x0C</td>
<td>I</td>
<td>F D E M W</td>
</tr>
</tbody>
</table>

This example: one thread (thread ID 0)
Branches to address 0x0C
Pipeline is flushed - 2 cycles are wasted

Unpredictability: 1 cycle if branch-not-taken and 3 cycles if taken
Thread-Interleaving for Predictability

<table>
<thead>
<tr>
<th>TID</th>
<th>Addr.</th>
<th>Inst.</th>
<th>Cycle</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0x00</td>
<td>BR</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>0x30</td>
<td>I</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>0x60</td>
<td>I</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>0x90</td>
<td>I</td>
<td>F</td>
</tr>
<tr>
<td>0</td>
<td>0x0C</td>
<td>I</td>
<td>F</td>
</tr>
</tbody>
</table>

In this example, we have 4 interleaved threads.

Thread 0 is not scheduled until after the branch address is calculated. No wasted cycles.

No pipeline hazards (no wasted cycles) if 3 or more scheduled threads in a row.

Flexible Scheduling with Cycle Stealing

- FlexPRET allow arbitrary interleaving
- Soft real-time threads (SRTT) can steal cycles from hard real-time threads (HRTT)

Example execution (read from up to down, left to right)
C level programming using real-time

- Work-in-progress of a LLVM based WCET-aware compiler
- Currently using a GCC port for RISC-V when compiling programs with C inline assembly macros. See period loop example below:

```c
int h, l;   // High and low 32-bit values
get_time(h, l); // Current time in nanoseconds
while(1){   // Repeat control loop forever
    add_ms(h, l, 10); // Add 10 milliseconds
    exception_on_expire(h, l, missed_deadline_handler);
    compute_task();   // Sense, compute, and actuate
    deactivate_exception(); // Deadline met
    delay_until(h, l); // Delay until next period
}
```

Part I
Mixed-Criticality Systems

Part II
The FlexPRET Processor Platform

Part III
WCET-Aware SPM Management

Mixed-Criticality Avionics Case Study

Task set structure derived from an avionics system at Honeywell (Vestal, 2007)
- Tasks from Maldarden
- 21 tasks
- 8 hardware threads.
- 4 criticality levels A-D.

A on separate HRTT
B on one HRTT (with Rate monotonic)
C and D SRTT with cycle stealing (EDF)

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Thread Mode</th>
<th>Tc, Dc (ms)</th>
<th>E1,1 (×10^5)</th>
<th>E1,1/2 (×10^3)</th>
<th>E1,1/3 (×10^0)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_A1</td>
<td>HA</td>
<td>25</td>
<td>1.10</td>
<td>1.00</td>
<td>0.95</td>
</tr>
<tr>
<td>T_A2</td>
<td>HA</td>
<td>50</td>
<td>1.80</td>
<td>1.64</td>
<td>1.55</td>
</tr>
<tr>
<td>T_A3</td>
<td>HA</td>
<td>100</td>
<td>2.00</td>
<td>1.82</td>
<td>1.72</td>
</tr>
<tr>
<td>T_A4</td>
<td>HA</td>
<td>200</td>
<td>5.30</td>
<td>4.83</td>
<td>4.56</td>
</tr>
<tr>
<td>T_B1</td>
<td>HA</td>
<td>25</td>
<td>1.40</td>
<td>1.27</td>
<td>1.20</td>
</tr>
<tr>
<td>T_B2</td>
<td>HA</td>
<td>50</td>
<td>3.90</td>
<td>3.54</td>
<td>3.34</td>
</tr>
<tr>
<td>T_B3</td>
<td>HA</td>
<td>50</td>
<td>2.80</td>
<td>2.54</td>
<td>2.40</td>
</tr>
<tr>
<td>T_B4</td>
<td>HA</td>
<td>50</td>
<td>1.40</td>
<td>1.28</td>
<td>1.21</td>
</tr>
<tr>
<td>T_B5</td>
<td>HA</td>
<td>50</td>
<td>3.70</td>
<td>3.37</td>
<td>3.19</td>
</tr>
<tr>
<td>T_B6</td>
<td>HA</td>
<td>100</td>
<td>1.80</td>
<td>1.64</td>
<td>1.55</td>
</tr>
<tr>
<td>T_B7</td>
<td>HA</td>
<td>200</td>
<td>8.50</td>
<td>7.75</td>
<td>7.32</td>
</tr>
<tr>
<td>T_C1</td>
<td>SA</td>
<td>50</td>
<td>1.90</td>
<td>1.77</td>
<td>1.63</td>
</tr>
<tr>
<td>T_D1</td>
<td>SA</td>
<td>50</td>
<td>5.40</td>
<td>5.03</td>
<td>4.65</td>
</tr>
<tr>
<td>T_D2</td>
<td>SA</td>
<td>200</td>
<td>2.40</td>
<td>2.33</td>
<td>2.28</td>
</tr>
<tr>
<td>T_D3</td>
<td>SA</td>
<td>50</td>
<td>1.30</td>
<td>1.26</td>
<td>1.23</td>
</tr>
<tr>
<td>T_D4</td>
<td>SA</td>
<td>200</td>
<td>1.50</td>
<td>1.45</td>
<td>1.42</td>
</tr>
<tr>
<td>T_D5</td>
<td>SA</td>
<td>50</td>
<td>2.30</td>
<td>2.14</td>
<td>1.98</td>
</tr>
<tr>
<td>T_D6</td>
<td>SA</td>
<td>100</td>
<td>4.80</td>
<td>4.65</td>
<td>4.30</td>
</tr>
<tr>
<td>T_D7</td>
<td>SA</td>
<td>200</td>
<td>13.00</td>
<td>12.70</td>
<td>12.44</td>
</tr>
<tr>
<td>T_D8</td>
<td>SA</td>
<td>100</td>
<td>0.60</td>
<td>0.57</td>
<td>0.56</td>
</tr>
<tr>
<td>T_D9</td>
<td>SA</td>
<td>50</td>
<td>2.40</td>
<td>2.33</td>
<td>2.28</td>
</tr>
</tbody>
</table>

Part I
Mixed-Criticality Systems

Part II
The FlexPRET Processor Platform

Part III
WCET-Aware SPM Management
Part III

WCET-Aware

SPM Management

Software Managed Multicores

In FlexPRET, HRTT can only access Scratchpad memory (SPM) directly.

Problem: How can we dynamically load code from the main memory to SPM such that WCET is minimized?

Traditional use of SPM. Static allocation (partitioning) and direct access to main memory.

Software Managed Multicore (SMM)
Only access to SPM. Need DMA.

Examples:
- Cell processor
- FlexPRET
Main Idea

Task 1: Given a function-to-region mapping, compute WCET
Task 2: Find an optimal mapping that minimizes WCET

Contribution:
- Formalized an optimal solution using ILP
- Developed a scalable, but sub-optimal heuristic

![Diagram of function-to-region mapping]

Part I: Mixed-Criticality Systems
Part II: The FlexPRET Processor Platform
Part III: WCET-Aware SPM Management

Benchmarks and Limitations

Benchmark on Malardalen benchmarks and MiBench

- ILP finds solution quickly, except for one benchmark. Note: sub-optimal ILP still best.
- Compare with SPM heuristics designed for average case.

Conclusion: Finds optimal solution, but cannot handle recursive programs.
Conclusions

FlexPRET

Software Management Multicore (SMM) are architectures where the processor only can access SPM.

Developed an optimal SPM mapping solution that minimizes WCET.
Conclusions

The accepted (camera-ready) versions of the RTAS papers are available here: http://www.eecs.berkeley.edu/~broman/


Thank you for listening!