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 Shaver
Edward A. Lee

Yooseong Kim
David Broman
Jian Cai
Aviral Shrivastava

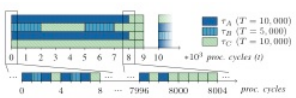
Agenda

Part I

Mixed-Criticality Systems

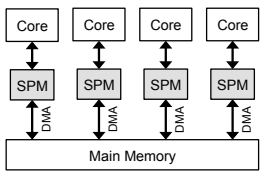
Part II

The FlexPRET Processor Platform



Part III

WCET-Aware SPM Management



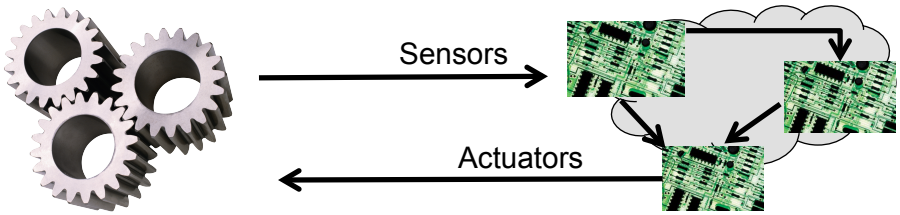
Part I
Mixed-Criticality Systems

Cyber-Physical Systems (CPS)

Automotive

Process Industry and Industrial Automation

Aerospace



Physical system (the plant) **Cyber system**: Computation (embedded) + Networking

Modern Systems with Many Processor Platforms

broman@eecs.berkeley.edu

Aerospace

Modern aircraft have many computer controlled systems

- Engine control
- Electric power control
- Radar system
- Navigation system
- Flight control
- Environmental control system

etc…

Automotive

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

broman@eecs.berkeley.edu

Issues with too many processors

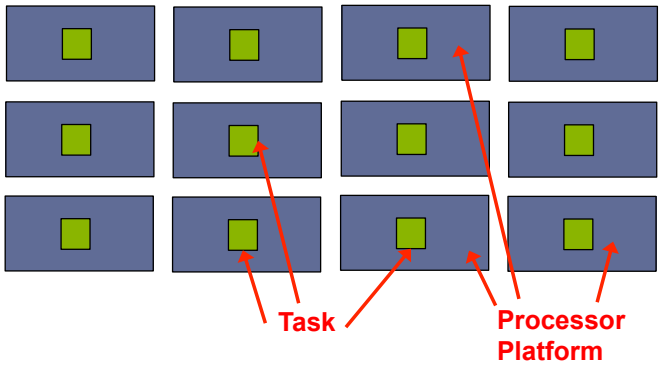
- High cost
- Space and weight
- Energy consumption

Required for Safety

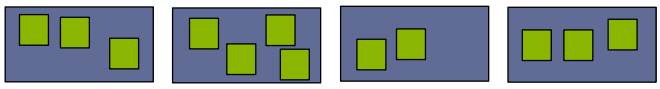
- Spatial isolation between tasks
- Temporal isolation between tasks (necessary to meet deadlines)

Federated Approach

Each processor has its own task



Consolidate into fewer processors



Mixed-Criticality Systems

broman@eecs.berkeley.edu

Issues with too many processors

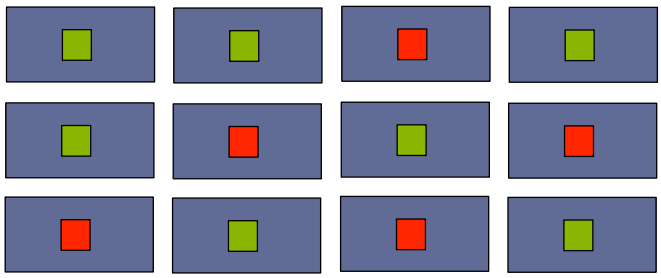
- High cost
- Space and weight
- Energy consumption

Required for Safety

- Spatial isolation between tasks
- Temporal isolation between tasks (necessary to meet deadlines)

Federated Approach

Each processor has its own task



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

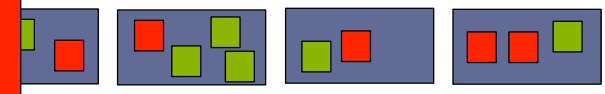
Mixed-Criticality Challenge

Reconcile the conflicting requirements of:

- Partitioning (for safety)
- Sharing (for efficient resource usage)

(Burns & Davis, 2013)

olidate into fewer processors



Possible Approaches…

broman@eecs.berkeley.edu

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)

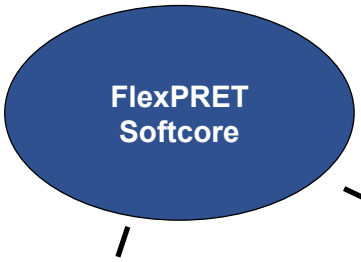
Multi-core with Task Partitioning

+ 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

broman@eecs.berkeley.edu



Fine-grained Multithreaded Processor Platform (thread interleaved) implemented on an FPGA

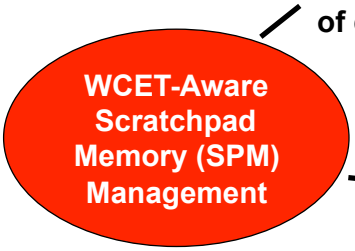
Flexible schedule (1 to 8 active threads) and scheduling frequency (1, 1/2, 2/3, 1/4, 1/8 etc.)

Hard real-time threads (HRTT) with predictable timing behavior

- **Thread-interleaved pipleine (no pipeline hazards)**
- **Scratchpad memory instead of cache**

Soft real-time threads (SRTT) with cycle stealing from HRTT

Automatic DMA transfer of code to SPM



Optimal mapping for minimizing WCET

Related Work

broman@eecs.berkeley.edu

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)

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)

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)

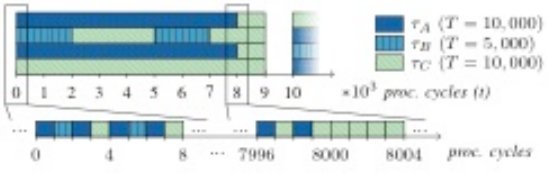
Predictable and Multithreaded Processors

- 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**

5-stage pipeline. We do not compute the branch address until the end of the execution cycle.

TID	Addr.	Inst.	Cycle							
			1	2	3	4	5	6	7	8
0	0x00	BR 0x0C	F	D	E	M	W			
0	0x04	I		F	D	-	-	-		
0	0x08	I			F	-	-	-	-	
0	0x0C	I				F	D	E	M	W

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

broman@eecs.berkeley.edu

TID	Addr.	Inst.	Cycle							
			1	2	3	4	5	6	7	8
0	0x00	BR 0x0C	F	D	E	M	W			
1	0x30	I		F	D	E	M	W		
2	0x60	I			F	D	E	M	W	
3	0x90	I				F	D	E	M	W
0	0x0C	I					F	D	E	M

In this example, we have 4 interleaved threads.

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

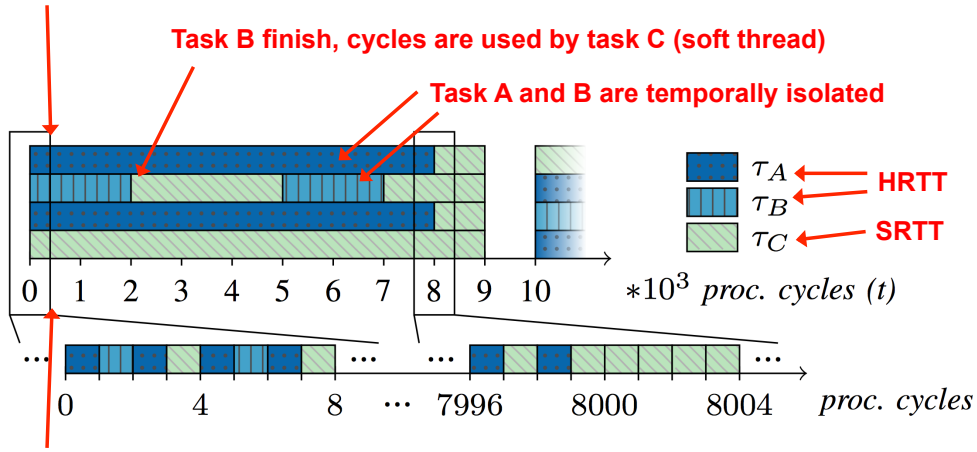
No pipeline hazards (no wasted cycles) if 3 or more scheduled treads in a row.

Flexible Scheduling with Cycle Stealing

broman@eecs.berkeley.edu

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

Task A (hard) frequency 2/4 = 1/2
Task B (hard) frequency 1/4
Task C (soft) frequency 1/4 + cycle stealing



Example execution
(read from up to down, left to right)

C level programming using real-time

broman@eecs.berkeley.edu

- **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:**

1-2: Get time in nano seconds (64 bits)

5: Add en exception handler (immediate detection of missed deadline)

```
1 int h,l;                          // High and low 32-bit values
2 get_time(h,l);                    // Current time in nanoseconds
3 while(1){                         // Repeat control loop forever
4   add_ms(h,l,10);                 // Add 10 milliseconds
5   exception_on_expire(h,l,missed_deadline_handler);
6   compute_task();                 // Sense, compute, and actuate
7   deactivate_exception(); // Deadline met
8   delay_until(h,l); // Delay until next period
9 }
```

6: Compute

7-8: Deactivate and delay (force lower bound)

NOTE: The delay until (DU) instruction is used for cycle stealing

Mixed-Criticality Avionics Case Study

broman@eecs.berkeley.edu

Task set structure derived from a avionics system at Honeywell (Vestal, 2007)

- **Tasks from Malardalen**
- **21 tasks**
- **8 hardware threads.**
- **4 criticality levels A-D.**

Periodic tasks

WCET cycles for different scheduling frequences

Task	Thread ID	Thread Mode	T_i, D_i (ms)	$E_{i,1}$ $(*10^5)$	$E_{i,1/2}$ $(*10^5)$	$E_{i,1/3+}$ $(*10^5)$
τ_{A1}	0	HA	25	1.10	1.00	0.95
τ_{A2}	1	HA	50	1.80	1.64	1.55
τ_{A3}	2	HA	100	2.00	1.82	1.72
τ_{A4}	3	HA	200	5.30	4.83	4.56
τ_{B1}	4	HA	25	1.40	1.27	1.20
τ_{B2}	4	HA	50	3.90	3.54	3.34
τ_{B3}	4	HA	50	2.80	2.54	2.40
τ_{B4}	5	HA	50	1.40	1.28	1.21
τ_{B5}	5	HA	50	3.70	3.37	3.19
τ_{B6}	5	HA	100	1.80	1.64	1.55
τ_{B7}	5	HA	200	8.50	7.75	7.32
τ_{C1}	6	SA	50	1.90	1.77	1.63
τ_{D1}	6	SA	50	5.40	5.03	4.65
τ_{D2}	6	SA	200	2.40	2.33	2.28
τ_{D3}	6	SA	50	1.30	1.26	1.23
τ_{D4}	6	SA	200	1.50	1.45	1.42
τ_{D5}	7	SA	25	2.30	2.14	1.98
τ_{D6}	7	SA	100	4.80	4.65	4.30
τ_{D7}	7	SA	200	13.00	12.70	12.44
τ_{D8}	7	SA	100	0.60	0.57	0.56
τ_{D9}	7	SA	50	2.40	2.33	2.28

(a) The task set

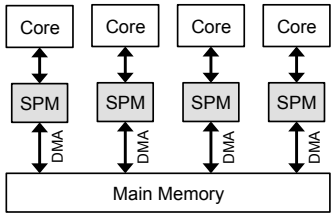
A on separate HRTT

B on one HRTT (with Rate monotonic)

C and D SRTT with cycle stealing (EDF)

Part III

WCET-Aware SPM Management

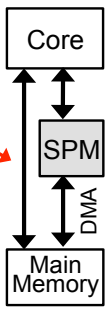


Software Managed Multicores

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

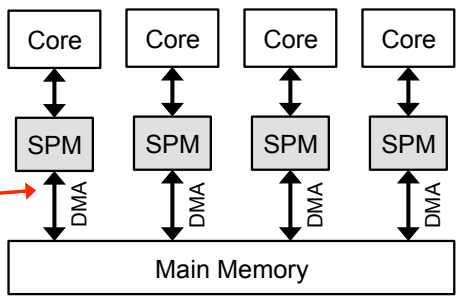
Problem: How can we dynamically load code from the main memory to SPM such that WCET is minimized?

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



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

Examples:
- **Cell processor**
- **FlexPRET**



Main Idea

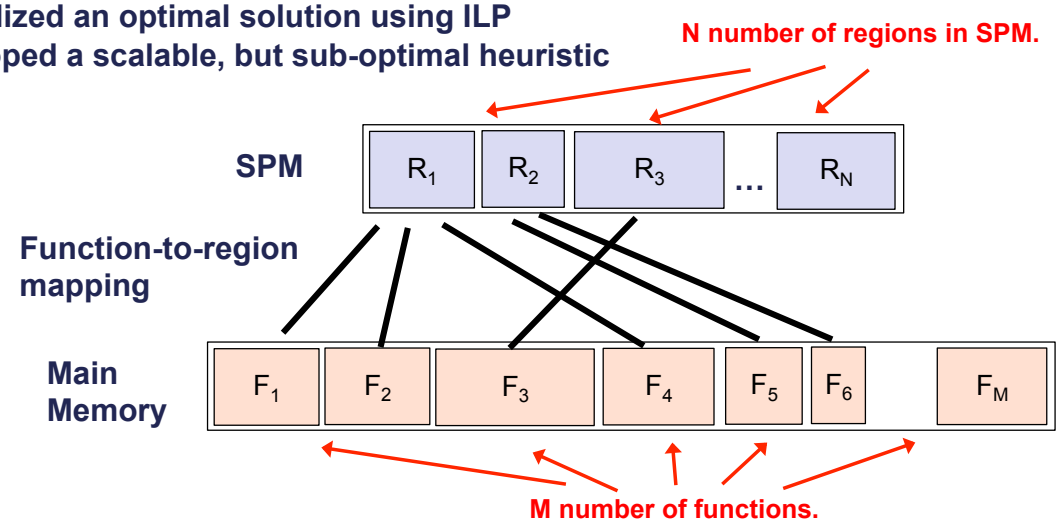
broman@eecs.berkeley.edu

Task1: Given a function-to-region mapping, compute WCET

Task2: Find an optimal mapping the minimizes WCET

Contribution:

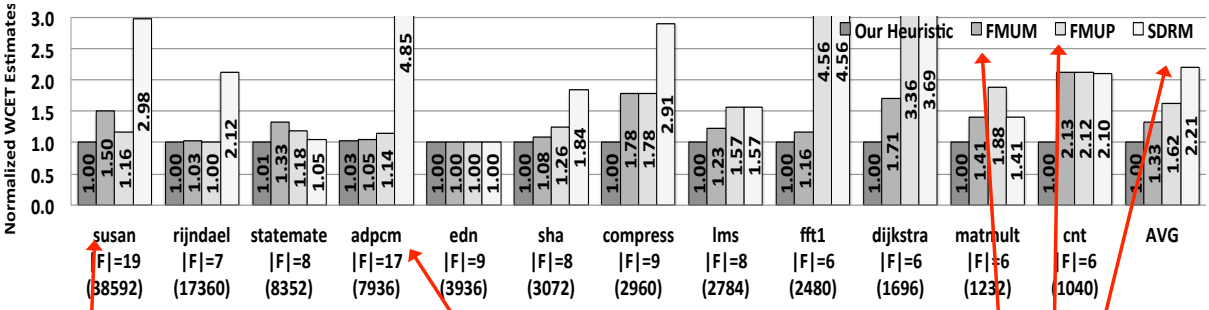
- **Formalized an optimal solution using ILP**
- **Developed a scalable, but sub-optimal heuristic**



Benchmarks and Limitations

broman@eecs.berkeley.edu

Benchmark on Malardalen benchmarks and MiBench



(b) Normalized WCET Estimates when the SPM size is 75% of the code size

Normalized compare to optimal solution.

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.

broman@eecs.berkeley.edu

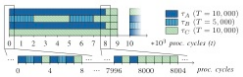
Conclusions

Part I
Mixed-Criticality Systems

Part II
The FlexPRET Processor Platform

Part III
WCET-Aware SPM Management

Conclusions



FlexPRET

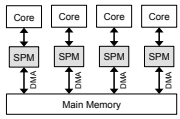
Gives timing predictability for HRTT using thread-interleaved pipeline and SPM

Flexible thread scheduling for up to 8 threads.
Scheduling in hardware.

Improving utilization by cycle stealing (from HRTT to SRTT).
Uses timing instructions for cycle stealing.

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

Developed an optimal SPM mapping solution that minimizes WCET.

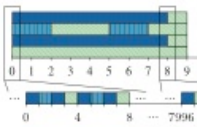


WCET-Aware SPM Mapping

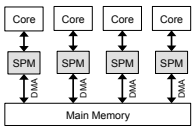
Conclusions

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

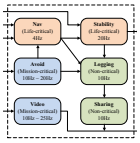
.



Michael Zimmer, David Broman, Chris Shaver, and Edward A. Lee.
FlexPRET: A Processor Platform for Mixed-Criticality Systems.
Proceedings of the 20th IEEE Real-Time and Embedded Technology and Application Symposium (RTAS), Berlin, Germany, April 15-17, 2014.



Yooseong Kim, David Broman, Jian Cai, and Aviral Shrivastaval**.**
WCET-Aware Dynamic Code Management on Scratchpads for Software-Managed Multicores. Proceedings of the 20th IEEE Real-Time and Embedded Technology and Application Symposium (RTAS), Berlin, Germany, April 15-17, 2014.



Eugene Yip, Matthew Kuo, Partha S Roop, and David Broman.
Relaxing the Synchronous Approach for Mixed-Criticality Systems.
Proceedings of the 20th IEEE Real-Time and Embedded Technology and Application Symposium (RTAS), Berlin, Germany, April 15-17, 2014.

Thank you for listening!