Time-Aware Correct-By-Construction Systems Design

SCS Seminar, December 4, 2014

David Broman
Associate Professor, KTH Royal Institute of Technology
Assistant Research Engineer, University of California, Berkeley

Agenda

Part I
Time-Aware Systems Design: a Vision

Part II
Predictable Processors for Mixed-Criticality Systems
What is a Time-Aware System?

**Time-aware systems** are systems where time or timing affects the correctness of the system behavior.

- **Real-time systems** are time-aware. For instance, execution of tasks must finish within certain deadlines.
- **Simulation systems** can be time-aware, but are not necessarily real-time.
- **Cyber-Physical Systems** are time-aware and real-time. Emphasis on networks and the interaction between cyber and physical.
- **Distributed systems** can be time-aware that are not CPS.
Part I
Time-Aware Systems Design: a Vision

Part II
Predictable Processors for Mixed-Criticality Systems

David Broman
dbro@kth.se

Time-Aware Systems - Examples

Cyber-Physical Systems (CPS)

Aircraft

Automotive

Process Industry and Industrial Automation

Time-Aware Simulation Systems

Physical simulations
(Simulink, Modelica, etc.)

Time-Aware Distributed Systems

Time-stamped distributed systems
(E.g. Google Spanner)

Time-Aware Systems Design

\[ J_1 \omega_2 = M_5 - M_1 \]
\[ J_2 \omega_1 = M_6 - M_3 \]
\[ \omega_2 = -\omega_1 \omega_2 \]
\[ M_5 = -r^{-1} M_2 \]

Equation-based model

Various models of computation (MoC)

Modeling

Simulation with timing properties

System

Physical system (the plant)

Cyber system: Computation (embedded) + Networking

Part I
Time-Aware Systems Design: a Vision

Part II
Predictable Processors for Mixed-Criticality Systems

David Broman
dbro@kth.se
**Time-Aware Systems Design**

\[
\begin{align*}
J_1 \omega_1 &= M_2 - M_1 \\
J_2 \omega_2 &= M_3 - M_2 \\
\omega_1 &= -\frac{1}{J_2} \omega_2 \\
M_1 &= -r^{-1}M_2
\end{align*}
\]

Simulation with timing properties

**Model fidelity problem**

"Ensuring that the model accurately represents the real system"

**Challenge:**
Compile/synthesize the model's cyber part, such that the simulated model and the behavior of the real system coincide.

The main challenge is to guarantee correct timing behavior.

---

**What is our goal?**

"Everything should be made as simple as possible, but not simpler"
attributed to Albert Einstein

**Execution time should be as short as possible, but not shorter**

No point in making the execution time shorter, as long as the deadline is met.

**Objective:**
Minimize area, memory, energy.

**Challenge:**
Still guarantee to meet all timing constraints.

---

David Broman
dbro@kth.se

Part I
Time-Aware Systems Design: a Vision

Part II
Predictable Processors for Mixed-Criticality Systems
A Story…

Fly-by-wire technology controlled by software.

Safety critical ➔
Rigorous validation and certification

Success?

They have to purchase and store microprocessors for at least 50 years production and maintenance…

Why?

Apparently, the software does not specify the behaviour that has been validated and certified!

Programming Model and Time

Timing is not part of the software semantics
Correct execution of programs (e.g., in C, C++, C#, Java, Scala, Haskell, OCaml) has nothing to do with how long time things takes to execute.

Traditional Approach

Our Objective

Make time an abstraction within the programming model

Timing is independent of the hardware platform (within certain constraints)
Part I
Time-Aware Systems
Design: a Vision

Part II
Predictable Processors for Mixed-Criticality Systems

David Broman
dbro@kth.se

Time-Aware Tool Chain Vision

**Modeling Languages**
- Simulink/Stateflow (Mathworks)
- Modelica (Modelica Associations)
- Ptolemy II (Eker et al., 2003)
- Modelyze (Broman and Siek, 2012)
- Giotto and E machine (Henzinger et al., 2003)

**Programming Languages**
- Real-time Concurrent C (Gehani and Ramamritham, 1991)
- Real-Time Euclid (Klingerman & Stoyenko, 1986)

**Assembly Languages**
- The assembly languages for today's processors lack the notion of time
- PRET Machines at UC Berkeley (see part II)

Difficult to compute WCET (e.g., determine loop bounds and infeasible paths)

Work-in-progress: C extended with timing constructs

Timed C

PRET ISA
Time-Aware Tool Chain Vision

**Modeling Languages**
- Simulink/Stateflow (Mathworks)
- Modelica (Modelica Associations)
- Ptolemy II (Eker et al., 2003)
- Modelyze (Broman and Siek, 2012)
- Giotto and E machine (Henzinger et al., 2003)

**Programming Languages**
- Work-in-progress: C extended with timing constructs

**Assembly Languages**
- PRET

**Time-Aware Compilation**

- Pretil
  - Expose timing constructs
  - Abstracting away memory hierarchy (scratchpad, DRAM etc.)

**Other (non PRET)**
- ISA

**Part I**
- Time-Aware Systems Design: a Vision

**Part II**
- Predictable Processors for Mixed-Criticality Systems

David Broman
dbro@kth.se
**Research Objective:** Develop *methodologies, algorithms, and a time-aware tool chain* that change the way we develop these kind of systems using a *correct-by-construction* approach.

**Modeling/Program Language**

- **Area 1:** Programming Languages and APIs with timing constrains
  - Modelyze
  - Timed C
  - Functional Mockup Interface (FMI)

- **Area 2:** Time-aware compilation/synthesis
  - LLVM-based time-aware compiler
  - WCET analysis

- **Area 3:** Predictable Architectures and Clock Synchronization
  - PRET processors
  - Clock Synchronization

**Part I**

*Time-Aware Systems Design: a Vision*

**Part II**

*Predictable Processors for Mixed-Criticality Systems*

*This part 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*
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)

Part I
Time-Aware Systems
Design: a Vision

Part II
Predictable Processors for Mixed-Criticality Systems

David Broman
dbro@kth.se

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

David Broman
dbro@kth.se

Part I
Time-Aware Systems
Design: a Vision

Part II
Predictable Processors for Mixed-Criticality Systems
Mixed-Criticality Systems

Part I
Time-Aware Systems
Design: a Vision

Part II
Predictable Processors for
Mixed-Criticality Systems

David Broman
dbro@kth.se

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)

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

Federated Approach
Each processor has its own task

Consolidate into fewer processors

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

Our solution

FlexPRET Softcore

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

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

WCET-Aware Scratchpad Memory (SPM) Management

Automatic DMA transfer of code to SPM
Optimal mapping for minimizing WCET

Part I
Time-Aware Systems
Design: a Vision

Part II
Predictable Processors for Mixed-Criticality Systems

David Broman
dbro@kth.se
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,…

Flexible Scheduling with Cycle Stealing

- 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)

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

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

```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
}
```

6: Compute

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

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

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
WCET-Aware Scratchpad Allocation: main idea

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

More info on this topic

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.
Conclusions

Some key take away points:

- **Time-aware systems** are systems where time or timing affects the correctness of the system behavior.

- **Cyber-physical systems** (CPS) are Time-Aware, but systems without physical plants can also be time-aware (e.g., distributed time-stamped systems)

- **Overall objective**: Develop a new methodology, algorithms, and a tool chain that are time-aware and use a correct-by-construction approach.

- **Mixed-criticality systems** can be designed using predictable processors.

Thanks for listening!