What is mixed criticality?

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

This talk focuses on the time and timing aspects of the problem

Mixed Time-Critical Systems

Other aspects are equally important (hardware failures, network aspects etc.), but are not considered here.
Viewpoints on the MCS timing aspect

Viewpoint I
The Implementation View

- **Software Scheduling**
  Vestal’s model (and variants thereof) with different WCET numbers for different criticality levels.

- **Hardware Scheduling**
  For instance, the FlexPRET approach (Zimmer et. al 2014) with predictable and less predictable hardware threads.

Viewpoint II
The Specification View

- **A Task Model with Bounded Frequencies**
  Yip et al. (2014) on relaxed the synchronous approach for MSC.

- **Programming with Time**
  Express timing constraints and fault handling explicitly in a programming language.

---

**Hardware Scheduling with FlexPRET**

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

**Note:** Not limited to 8 tasks. Can schedule several tasks on the same hardware thread using software scheduling.

**Open Source:**
https://github.com/pretis/flexpret

Viewpoints on the MCS timing aspect

Viewpoint I
The Implementation View

- Software Scheduling
  Vestal’s model (and variants thereof) with different WCET numbers for different criticality levels.
- Hardware Scheduling
  For instance, the FlexPRET approach (Zimmer et. al 2014) with predictable and less predictable hardware threads.

Viewpoint II
The Specification View

- A Task Model with Bounded Frequencies
  Yip et al. (2014) on relaxed the synchronous approach for MSC.
- Programming with Time
  Express timing constraints and fault handling explicitly in a programming language.

A Task Model With Bounded Frequencies

Each periodic task has two frequency parameters: $f_{\text{max}}$ and $f_{\text{min}}$.

- Life Critical Tasks
  $f_{\text{max}} = f_{\text{min}}$.
- Mission Critical Tasks
  $f_{\text{max}} > f_{\text{min}}$.
- Non-Critical Tasks
  $f_{\text{max}}$ is the goal. $f_{\text{min}} = 0$

Example: Unmanned Aerial Vehicle (UAV)

Note:
The task model does not say anything about the implementation technique or WCETs for specific platforms.

Eugene, Kuo, Roop, and Broman. "Relaxing the Synchronous Approach for Mixed-Criticality Systems" (RTAS 2014)
Viewpoints on the MCS timing aspect

**Viewpoint I**

**The Implementation View**

- **Software Scheduling**
  Vestal's model (and variants thereof) with different WCET numbers for different criticality levels.

- **Hardware Scheduling**
  For instance, the FlexPRET approach (Zimmer et al. 2014) with predictable and less predictable hardware threads.

**Viewpoint II**

**The Specification View**

- **A Task Model with Bounded Frequencies**
  Yip et al. (2014) on relaxed the synchronous approach for MSC.

- **Programming with Time**
  Express timing constraints and fault handling explicitly in a programming language.

---

**Programming with Time**

**Motivation**

- **Timing Specification**: Be able to describe different task models within one framework

- **Formal**: To have an unambiguous formal semantics with precise meaning

- **Fault handling**: Be able to express precise run-time behaviors when e.g. deadlines are missed.

**Some related work**

- Giotto by Henzinger et al. (2001)
- Ptides by Eidson et al. (2012)
- Timing constraint logic by Lisper and Nordlander (2012)
- Synchronous approach for MSC by Cohen et al. (2015)
A Timed Lambda Calculus (unpublished work)

Syntax

Variables $x, y \in X$

Constants $c \in C$

Time $t \in \mathbb{N} \cup \infty$

Expressions $e ::= x | \lambda x e | e | c | \text{overrun} | \text{time} | \text{within \{t_1 \to t_2\} do \{e\} else \{e\}}$

Values $v ::= \lambda x e | e$

Frames $F ::= \emptyset | v \oplus | \text{within \{t_1 \to t_2\} do \{e\} else \{e\}}$

Dynamic Semantics

$\delta(c, v, s, t) = (s', t', d)$\quad $\beta(e) = (s', t', d)$\quad $\gamma(e) = (s', t', d)$

within $t_1 \to t_2$ do $e$ else $c | s, t, D \to e | v | s, t$

within $t_1 \to t_2$ do overrun else $c | x, s, t, D \to \{x\} | v | s, t$

within $t_1 \to t_2$ do $e$ else $c | x, s, t, D \to \{x\} | v | s, t$

Computation to be done within the bound.

Fault handling if a deadline is missed.

In this case, specifies the timing bounds for releases.

Constructs can be nested.

Construction can be put within loops or have conditions.
Conclusions

Some key take away points:

- **Implementation view of MCS**
  - Software Scheduling
  - Hardware Scheduling

- **Specification view of MCS**
  - Bounded Frequencies Task Model
  - Programming with Time

Thanks for listening!

| David Broman  
dbro@kth.se | Part I | The Implementation  
|             | View   | Part II  
|             |        | The Specification  
|             |        | View        |