Tools for Real-Time and Embedded Systems
An Integration View

Tommaso Cucinotta
Real-Time Systems Laboratory (ReTiS)
Scuola Superiore Sant'Anna, Pisa (Italy)
Talk Outline

- Introduction
- Selected Tools for RTES at the ReTiS
  - Scheduler for soft real-time VMs running on Linux/KVM
  - MetaSim / RTSim
  - ARSim
  - MCoreSim
- Selected Tools for RTES at Evidence s.r.l.
  - Scheduler for real-time embedded applications on Linux
  - RT-Druid
  - Erika
- Integration Efforts in Automotive (INTERESTED)
- Questions
Introduction
RTES Design & Development Challenges

- Compliance of Design, Architecture, Implementation with Requirements
- Functional as well as non-functional correctness
  - Behaviour, timeliness, deadlines
  - Reliability
  - Guarantees of correctness (under any possible run-time condition)
- Composability of software components
  - Keeping under control timing interferences
- Mapping on multi-core and heterogeneous hardware
- Exploration of design space
- ...

Introduction
RTES Design & Development Challenges

- Compliance of Design, Architecture, Implementation with Requirements
- Functional as well as non-functional correctness
  → Behaviour, timeliness, deadlines
  → Reliability
  → Guarantees of correctness (under any possible run-time condition)
- Composability of software components
  → Keeping under control timing interferences
- Mapping on multi-core and heterogeneous hardware
- …

Appropriate Tools & Methodologies are needed
V-Model development flow

User requirements → Functional specs → Functional design → Architecture selection → Module design → Coding → Module testing → Integration testing → System verification → System validation
The Tools Babylonia

Problems

- Heterogeneous models
- Interoperability issues (syntax)
- Integrability issues (semantics)
The Tools Babylonia

Problems

- Heterogeneous models
- Interoperability issues (syntax)
- Integrability issues (semantics)

But let's add some more tools before going on!
Tools for RTES at the ReTiS
Features at a glance

- **Resource Reservations**
  - EDF-based scheduling *(hard CBS)*

- **Hierarchical scheduling**
  - Multiple tasks attached to same reservation
  - POSIX Fixed Priority scheduling inside each reservation

- **Multi-processor** reservations
  - Partitioned scheduling for improved efficiency
  - Migration of tasks among CPUs

- **Simple admission control**
Replace real-time throttling

Tight integration in Linux kernel
- Modification to the Linux RT scheduler

Reuse as many Linux features as possible
- Management of task hierarchies and scheduling parameters via **cgroups**
- POSIX compatibility and API

Efficient for SMP
- Independent runqueues
Real-Time systems SIMulator (RTSIM)

What is RTSIM?

- A collection of reusable C++ components for simulating real-time (control) systems

Features

- Task Model
  - Periodic, Sporadic, Aperiodic
  - Period, WCET, Deadline,...
- Multi-Core Support
  - Global/Partitioned Scheduling
- Scheduling Algorithms
  - RR, FP, EDF
  - Servers (SS, PollS., CBS, GRUB)
- Shared Resources
  - Priority Inheritance (MBWI wip...)
- Random Task Generators
- XML-based System Specification
- Qt-based GUI (optional)
- Open-source
- Platforms: Linux, Windows, Mac OS-X
RTSIM Architecture

- **MetaSim**: basic simulation engine
  - Events, random variables, tracing, stats

- **RTLib**: architectural model
  - Tasks, kernel, resources, networks

- **CtrlLib**: functional model
  - Computation units, inputs/outputs

- Numerical package: plant simulation
  - Numerical integration, based on Octave

Simulator
**RTSIM Example**

**Inverted Pendulum**

- Two control loops: *angle* and *position*
- Video camera for sampling the *position*
- Potentiometer for sampling the *angle*
RTSIM Example
Inverted Pendulum
MCoreSim Features

Simulates tile-based many-core systems
- Based on OMNeT++

Current features (extensible):
- 2D Mesh and Torus
- Network Interface with flow-control for reliable comms
- Direction-Oriented Routing
- MMU accessing both Local Store and Remote Memory

Software stack as sequence of basic operations
- Load
- Store
- Core to core messages
- Internal core computations

Outputs timing information (latency, throughput figures)
Why MCoreSim

Other Multi-Core Simulators

- COTSon
  -> AMD SimNow
- QEmu
- PhoenixSim
- NIRGAM & noxim (based on SystemC)

No simulator covers future NoC with many cores and no shared memory or incoherent memory

MCoreSim approach

- Higher abstraction level (simulation performance benefits)
- Focus on impact of NoC on software performance (latency, throughput)
- Complements Clash for simulation purposes
MCoreSim Example

Preliminary validation on the Intel SCC 48-core, tile-based platform
Tools for RTES at Evidence s.r.l.
Features at a glance

- New scheduling class for the Linux kernel
- Implements deadline-based resource reservations
  - grants scheduling guarantees to individual tasks
- Provides temporal isolation among tasks
- Model: budget, deadline, period
- Multiple variants (e.g., work-conserving)
Performance and high efficiency

- Advanced real-time scheduling algorithms
- RAM: Stack sharing to limit RAM consumption
- Flash: very low footprint (1-4 KB)

Portable API based on automotive standard (OSEK/VDX)

Supports a wide range of microcontrollers

- Including multicore systems
Supported Microcontrollers

- Microchip dsPIC
- Atmel AVR
- ARM7TDMI (Samsung KS32C50100, ST STA2051, UniBO MPARM)
- Tricore 2
- PPC e200 (MPC5674F (e200z7), MPC5668 (e200z6))
- Microchip PIC32
- Ensilica esiRISC
- Altera NIOS II (with multi-core support!)
- Lattice Mico32
Development environment for Erika Enterprise

Compliance with automotive standard

- OSEK OIL

Schedulability analysis plugin

Template applications support

Easy development

- Multicore support
- Support for hardware debugging
- Based on the well-known Eclipse IDE
Fast Prototyping with Scilab/Scicos
in collaboration with INRIA (FR) and Supsi Lugano (CH)

Scicos functional modeling

Simulation

INRIA/SUPSI Code generator

Hardware In the Loop

USB Connection

Same Behavior!

HW + Erika Enterprise
Licensing

ERIKA Enterprise License
- GPL + Linking Exception: possibility to statically link a proprietary application with the source code
  → http://en.wikipedia.org/wiki/GPL_linking_exception

RT-Druid IDE License
- Currently a demo license
- Will be released soon under the EPL open-source license
Integration Efforts in the Automotive Domain
Several applications (an example is fuel injection control) are developed with the following characteristics:

- High cost-sensitiveness
- Faster CPU often not an option, limited availability of memory
- Large number of functional subsystems (~200) interacting, exchanging several hundreds of communication signals
- System is multirate and characterized by tight timing constraints
- Utilization is very high (>90%) and mode changes are required

Model-based design using Simulink (SR) models

Need to plan transition to AUTOSAR-based development
Motivation: automotive systems
sample requirements

Need for

- Supporting the design of the task and resource model starting from system-level models to Simulink and AUTOSAR models
  - Enforce semantics preservation (partial orders, comm flows)
  - Optimize wrt performance metrics within timing constraints
- Algorithms that optimize memory use while enforcing deadlines
  - Use of preemption thresholds
- Model and control mode changes

Critical issues (just to mention a few)

- Lack of timing/behavior info in AUTOSAR models
- Lack of a formal timed event model
- Mapping of the functional model to the platform & task model
  - Mainly manual in the current practice
A Typical Flow: Heterogeneous Models

+ separation between the functional model and the architecture model

**System-level Functional design**

**Component model(s)**

**Architecture model(s)**

**Function-to-Architecture (deployment) model(s)**

**Behavioral model(s)**

**Task model(s)**

**Code implementation**

**SR models (Simulink)**

**AUTOSAR 4.0**

**Simulation of system-level control functions**

**UML/SvsML ADL**

**Coding**

**Module design**

**Architecture selection**

**Functional design**
Gaps in the flow

- Lack of support for the definition of the task and resource model

- Insufficient support for the specification of timing constraints and attributes

- Insufficient support for the guiding the definition of the HW architecture and the placement of components

Functional design

System-level Functional design

Component model(s)

Architecture model(s)

Function-to-Architecture (deployment) model(s)

Behavioral model(s)

Task model(s)

Code implementation

UML/SysML

ADL

4.0 ?

AUTOSAR

4.0 ?

SR models (Simulink)

- Insufficient support for the specification of timing constraints and attributes
- Insufficient support for the guiding the definition of the HW architecture and the placement of components

Tommaso Cucinotta – Real Time Systems Laboratory (ReTiS) – Scuola Superiore Sant'Anna
Frictions in the integration

- Issues of more integration and consistency and semantics preservation

System-level Functional design
Component model(s)
Architecture model(s)
Function-to-Architecture (deployment) model(s)
Module design
Behavioral model(s)
Task model(s)
Coding
Code implementation

UML/SysML
ADL
AUTOSAR
SR models (Simulink)

- 4.0 ?
Heterogeneous Models in INTERESTED

Impact: fill the gap at system-level design

DOORS

User Requirements

SysML/Papyrus

Functional specs

System verification

System validation

Module testing

Domain

Impact: reduction of timing faults, simplified unit test

AUTOSAR

Functional design

RTW

Architecture selection

AT

Component/Module design

Targetlink

Coding

RT-Druid

Integration testing

Simulink

System verification

RTW generator

Module testing

AiT

Impact: fill the gap at system-level design

Tommaso Cucinotta – Real Time Systems Laboratory (ReTiS) – Scuola Superiore Sant'Anna
SystemDesk is a tool from dSPACE for developing complex system architectures. SystemDesk is able to export AUTOSAR XML files containing the system description for further analysis in RT-Druid.

Artop is an implementation of common base functionality for AUTOSAR development tools. Artop can be used as an enhanced editor for preparing and validating AUTOSAR XML descriptions to be imported in RT-Druid.

Papyrus is an integrated environment for graphical editing UML, SysML and MARTE. Papyrus is able to export RT-Druid RTD files containing the system architecture.

RT-Druid

- import/export AUTOSAR XML files produced by SystemDesk and Artop;
- perform schedulability analysis;
- configure the ERIKA Enterprise RTOS based on the model using OSEK OIL or AUTOSAR XML;
- provide sensitivity analysis for understanding application timing bottlenecks;
- import timing annotations from AbsInt aiT using the XML Timing Cookies (XTC) standard;
- import timing annotations using trace measurements from Lauterbach Trace32.

Application source code

Trace32 is used to code execution time using the microcontroller trace port.
Conclusions
Conclusions

Tools and Standards for RTES Design and Development

- Heterogeneous models, syntax and semantics
  → In particular, in the automotive domain
- Need for more integration & interoperability
- Issues partially addressed by the INTERESTED EU Project
Related Resources

ReTiS Website
- http://retis.sssup.it

RTSIM Website
- http://rtsim.sssup.it/

ARSim Website
- http://home.gna.org/arsim/

Evidence Website
- http://www.evidence.eu.com/

Erika Enterprise and RT-Druid Website
- http://erika.tuxfamily.org/

INTERESTED Website
- http://www.interested-ip.eu/

Automotive SPIN Italia
Thanks for your attention

Final questions?

http://retis.sssup.it/people/tommaso