Model-Based Design, Analysis and Synthesis for Multi-Core and TSP Avionics Targets

Jérôme Hugues *, Christophe Honvault†, Claire Pagetti*
* ISAE - France † ESA - The Netherlands ‡ONERA - France

***Abstract*—Multi-core, and Time and Space Partitionnong systems are two emerging paradigms for architecting avionics systems. They impose new steps in the development process: capturing configuration attributes, analysing their correctness, or guaranteeing performance. In this context, model-based techniques provide a framework to design, analyse and synthesize these systems while automating much steps.**

In this paper, we report on a set of extenstions of TASTE to support multi-core and TSP systems. We first present the key architectural elements of these systems, and then detail how these have been support as part of the generation toolchain. We then present experiments realized on two case studies and two hardware targets, both provided with the XtratuM hypervisor.

I. Introduction

A. Context

One major output of the FP6 ASSERT project [1], [2] was the TASTE (The ASSERT Set of Tools for Engineering) tool suite. This open-source tool chain, dedicated to the development of embedded real-time systems, particularly addresses the modelling and deployment of distributed systems composed of heterogeneous software and hardware components. From such a model, TASTE provides automation of tedious and error-prone validation and integration tasks.

Since the former project, multi-core processors have emerged as good candidates for the space domain. In particular qualified and hardened processors have been developed, such as the dual core processor GR712RC [3] already available and the quad core processor GR740 [4] under development.

On top of these hardware, qualified executive layers, such as RTOS (real-time operating systems) or hypervisors, must be developed. Currently, no such qualified layer is yet available even though several studies are on going, among which we could mention RTEMS (EDISOFT RTEMS 4.8), XtratuM [5], pikeOS (http://www.pikeos.com/) or LVCUGEN [6].

B. Moving towards TASTE multi-core

During a one-year project funded by ESA, we evaluated potential extensions of TASTE to support multi-core in some pre-defined configurations. More precisely, we decided to focus on IMA (Integrated Modular Avionics) / TSP (Time and Space Partitioning) set up.

For that, we extended the TASTE environment with fundamental principles and that lead to the definition of design pattern definitions. To validate the approach, we considered two use cases and implemented them first manually with XtratuM on the ZYNQ board [7] and the LEON processors. And then we compared those implementation with the generated ones from TASTE tool chain.

This project focused on minimal extensions to support multi-core and TSP. We applied it for TASTE, but it can serve as a general approach for other frameworks such as the Space Component Model [8], ECOA [9] or others as they all share similar separation of concerns approach between high-level description of components, their combination and finally their deployment on top of an executive and hardware platform.

In the following, we present both TASTE mono-core and TSP principles in section II. In section III, we detailed the extensions of TASTE, both in terms of modelling with patterns and of code generation. The experimental part is described in section IV.

II. Starting points

The purpose of the project was to provide some multi-core extension to TASTE. The targetted TRL was very low due to the duration (1 year) and the allocated budget effort.

A. TASTE

TASTE features TASTE aims at automating the software development process of space-critical application. It supports the following system operational requirements: limited resources (memory, processor);real-time constraints (deadlines); applications of very different natures (control laws, resource management, protocols, fault detection); communication with hardware (sensors, actuators, FPGA); heterogeneous hardware (e.g. processors with different endianness); distribution over several physically independent platforms; may run autonomously for years; may not be physically accessible for maintenance (satellites).

Hence, the quality of the generated code, along with the capacity to validate the system early are of prime interest. The TASTE process is shown in Figure 1.

Step 1: functional architecture The philosophy is to let the user only focus on his functional code, letting him write it in the language of his choice, may it be a modelling language or a low-level implementation language. To achieve this, TASTE relies on the AADL [10] and ASN.1 [11] text-based modelling languages that give sufficient power of expression to capture all the essential elements of a system that are required to generate the tasks, communication middleware and glue around the user functional code.

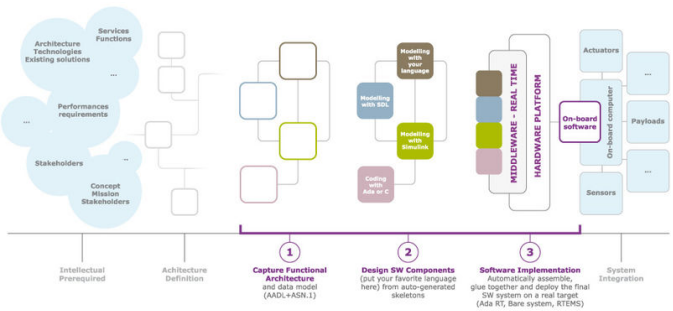

Figure 1: TASTE overview.

Step 2: internal function code Once a set of carefully selected system properties has been captured using these two languages, the core of the system's subcomponents can be developed using C, Ada, SDL, SCADE, Simulink, or VHDL.

Step 3: automatic code generation TASTE tools are responsible for putting everything together, including drivers and communication means and ensuring that the execution at runtime is compliant with the specification of the system real-time constraints. Without any major overhead in the code, TASTE will produce binaries that can be directly executed on several supported targets: native Linux, Real-time Linux (Xenomai), RTEMS, and Ada bare-board targets.

TASTE benefits TASTE evolved since 2007, it now supports the integration of several input languages for the functional part, multiple targets and enables both massive code generation, but also scheduling analysis, simulation capabilities. These capabilities have been validated through ESA-funded studies, but also partners.

B. TASTE TSP and multicore extensions

In the study, partners investigated various strategies to leverage configurations with multiple processing units such as multi-core systems or multi-processor systems. We particularly analysed requirements associated to AMP (Asymmetric multiprocessing), SMP (symmetric multiprocessing) and TSP settings, most definitions of which can be found in the book by Hennessy and Patterson [12].

Note: in the following, we restrict our abstract to the TSP paradigm because the space domain has defined TSP-based building blocks, such as XtratuM and LVCUGEN. The full paper would also cover the SMP case and RTEMS.

C. TSP principles and XtratuM hypervisor

a) TSP principles: Space domain refers as time and space partitioning while avionic domain refers to IMA (Integrated Modular Avionics). IMA proposes an integrated architecture with application software portable across an assembly of common hardware modules. It is partially captured in document DO-297 [13]. IMA is also used as a generic term within Airbus to denote a line of CPU, network and OS together. In the scope of this project, we will restrict IMA to RTOS kernels with time and space isolation capabilities. The later are made available thanks to new generation of CPU architectures that support both time (round-robin scheduling strategy) and space (memory segregation) strategies.

Compliant IMA platforms follow thee IMA guidance document, with the extension for Multi-core Platform. It mandates that the platform provides robust resource and time partitioning not only between software applications hosted on the same core, but also between applications hosted on different cores or between applications hosted on several cores.

Time-wise, AMP was the first proposed technology in the 60's, followed by SMP and then IMA in the 90's. In terms of adoption for safety-critical systems, or space system, AMP using multiple physical processors has been supported in real-time operating systems like RTEMS for many years. The introduction of SMP or IMA type of platforms is more recent, and largely delayed due to inherent difficulty to ensure strict determinism, and similar levels of safety assurance.

b) Quick overview of XtratuM: XtratuM is a TSP real-time hypervisor developed by Fentiss, a spin-off from the University of Valencia. In its current version, XtratuM supports Leon3 mono-core and Leon3 multi-core (GR712RC) as well as LEON4 multi-core. XtratuM is still being developed through various R&D projects (CNES, ESA, Thales Alenia Space and Airbus Defence and Space). XtratuM is able to host various guest OS (Linux, RTEMS, LithOS) and comes with a builtin support of the XtratuM Abstraction Layer (XAL) which enables the development of baremetal like partitions.

As a TSP kernel XtratuM has builtin support for spatial and temporal isolation of the software partitions with scheduling capabilities inspired from ARINC 653. XtratuM has support for Ethernet, SpaceWire and MIL-STD1553 devices.

Definition 1: An application is defined as a set of periodic or sporadic tasks $app = \{\tau_i = (C_i, T_i)\}$ where C_i is the WCET (Worst Case Execution Time) and T_i is the period or minimal inter-arrival time. The deadline is equal to the period.

An application is mapped as a set of *partitions* and a partition is defined by one or multiple *slots*, each with a start time and a length. Inside a slot, several tasks can be executed. Both the partitions slots and the schedule of tasks inside a slot are computed off-line, for instance with Xoncrete [14] the mapping and scheduling tool provided with XtratuM. This off-line information is called *plan* in the XtratuM terminology.

Definition 2 (Plan): A plan consists of:

- a *major frame* (MAF), representing the hyperperiod of the schedule, *MAF_length*;
- a set of slots sl_i distributed over the cores and the MAF. A slot is defined as $sl_i = ([s_i, e_i], n_i)$ where s_i is the start time, e_i is the end time and n_i is the number of core where the slot is allocated;
- a mapping of the jobs of critical applications in the slots. Jobs are unrolled on the MAF and we know for all job $\tau_{i,j}$ in which slot sl_k it belongs to. We know moreover in which order are executed the jobs inside a slot;
- a mapping of best-effort applications in the slots. For instance, app_i is executed in the slots $sl_{j_1}, \ldots, sl_{j_p}$.

III. Patterns

IV. Application to two use cases

A. ROSACE

The ROSACE– Research Open-Source Avionics and Control Engineering – has been developed as a collaboration between ONERA, ISAE and Polytech Montréal and its initial specification has been published in [**?**]. Although of modest size, this controller is representative of real avionics applications by introducing typical characteristics such as a data-flow design or complex multi-periodic execution patterns.

a) Specification: The application is composed of

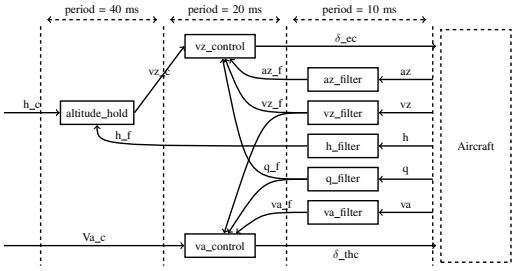

Figure 2: ROSACE architecture

b) Implementation in XtratuM: We chose the base time unit to be in milliseconds, the period of some functions is of 5ms and thus there could be at most 5 partition slots in the first 5ms. We compared two XtratuM implementations. The first was uni-processor for which wes defined 3 partitions (P0, P1 and P2) communicating via sampling channel. In the second, we made a dual core schedule with 5 partitions.

B. GCU case study

The second case study is extracted from the French national project Spacify. More precisely, we rely on the CNES case study detailed in [] that models the Payload Data Management System (GCU – Gestionnaire de Charge Utile in French) of a satellite. The purpose of the system is to apply commands from the ground to move in a given mode and to confirm to the ground that requests have been correctly applied.

a) Specification:

b) Implementation in XtratuM:

C. Experiments

References

[1] M. Perrotin, E. Conquet, P. Dissaux, T. Tsiodras, and J. Hugues, "The TASTE toolset: turning human designed heterogeneous systems into computer built homogeneous software," in *Proceedings of the 5th Conference on Embedded Real Time Software and Systems (ERTS'10)*, 2010.

[2] C. H. Julien Delange and J. Windsor, "Model-based engineering approach for system architecture exploration," in *Proceedings of the 6th Conference on Embedded Real Time Software and Systems (ERTS'12)*, 2012.

[3] COBHAM, "GR712RC Dual-Core LEON3-FT SPARC V8 Processor," 2016, http://www.gaisler.com/doc/gr712rc-datasheet.pdf.

[4] ——, "GR740 Quad Core LEON4 SPARC V8 Processor," 2017, http://www.gaisler.com/doc/gr740/GR740-UM-DS.pdf.

[5] M. Masmano, I. Ripoll, A. Crespo, J. Metge, and P. Arberet, "Xtratum: An open source hypervisor for TSP embedded systems in aerospace," in *DASIA 2009. DAta Systems In Aerospace.*, May. Istanbul 2009.

[6] J. Galizzi, J.-J. Metge, P. Arberet, E. Morand, F. Vigeant, A. Crespo, M. Masmano, J. Coronel, I. Ripoll, V. Brocal, F. Roubert, C. Scuri, V. Tedesco, and N. Thomasson, "LVCUGEN (TSP-based solution) and first porting feedback," in *Proceedings of the 6th Conference on Embedded Real Time Software and Systems (ERTS'12)*, 2012.

[7] Xilinx, "Zynq-7000 All Programmable SoC ZC702 Evaluation Kit," 201, https://www.xilinx.com/support/documentation/boards_and_kits/zc702_zvik/xtp310-zc702-quickstart.pdf.

[8] M. Panunzio and T. Vardanega, "A component-based process with separation of concerns for the development of embedded real-time software systems," *Journal of Systems and Software*, vol. 96, pp. 105 – 121, 2014. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121214001381

[9] J. Fenn, T. Cornilleau, Y. Oakshott, and A. Britto, "A pragmatic approach to capturing safety and security relevant information for reusable european component oriented architecture software components," in *9th IET International Conference on System Safety and Cyber Security (2014)*, Oct 2014, pp. 1–6.

[10] SAE, *Architecture Analysis & Design Language v2.1 (AS5506B)*. SAE, sep 2012.

[11] "REC. X680-X.683, ISO/IEC: Abstract Syntax Notation (ASN.1)," ITU-T, Tech. Rep., 2002.

[12] J. L. Hennessy and D. A. Patterson, *Computer Architecture, Fifth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., 1990.

[13] Radio Technical Commission for Aeronautics (RTCA) and EURopean Organisation for Civil Aviation Equipment (EUROCAE), *DO-297: Software, Electronic, Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations*, Std.

[14] V. Brocal, M. Masmano, I. Ripoll, A. Crespo, P. Balbastre, and J.-J. Metge, "Xoncrete," in *Proceedings of the 5th Conference on Embedded Real Time Software and Systems (ERTS'10)*, 2010, http://www.fentiss.com/documents/xoncrete_overview.pdf.