HW/SW Co-Design Framework for Mixed-Criticality Embedded Systems considering Xtratum-based SW Partitions

1st Vittoriano Muttillo
University of L'Aquila
L'Aquila, Italy
vittoriano.muttillo@graduate.univaq.it

2nd Luigi Pomante
University of L'Aquila
L'Aquila, Italy
luigi.pomante@univaq.it

3rd Patricia Balbastre
Universitat Politècnica de València
Valencia, Spain
pbalbastre@ai2.upv.es

4th José Simó
Universitat Politècnica de València
Valencia, Spain
jsimo@ai2.upv.es

5th Alfons Crespo
Universitat Politècnica de València
Valencia, Spain
acrespo@ai2.upv.es

***Abstract*—Heterogeneous parallel devices are becoming widely diffused in the embedded systems application field since they allow to improve time performances and other orthogonal metrics (e.g., cost, power, size, etc.) at the same time. In such a context, the introduction of safety requirements, as dictated by the relevant standards (i.e., DO-178 B/C and RTCA/DO-254 in airborne systems, ARINC 653 for avionics software, ISO-26262 in automotive domain, etc.) while considering shared resources on a heterogeneous parallel HW platform, adds further challenges to industrial and academic research. This kind of platforms that execute tasks with different levels of criticality are commonly called mixed-criticality embedded systems. So, the main problem in their management is to ensure that low criticality tasks do not interfere with high criticality ones. The final goal is to allow several applications to interact and coexist on the same platform. For this, the exploitation of virtualization technologies (i.e., hypervisors) allows to guarantee isolation and to satisfy certification requirements but introduces scheduling overhead and new HW/SW partitioning challenges. In such a scenario, this work focuses on a framework for modeling, analysis, and validation of mixed-criticality and real-time systems based on an existing "Model-Based Electronic System Level HW/SW Co-Design" methodology. The main contribution of this work is the integration of the considered framework with Xamber tool in order to provide systems implementations by exploiting a design space exploration able to consider Xtratum-based SW partitions.**

***Index Terms*—HW/SW Co-Design, Heterogeneous Parallel Systems, Design Space Exploration, Mixed-Criticality, Hypervisor**

I. Introduction

In the last thirty years, there has been an exponential increase in the diffusion and evolution of *Embedded Information and Communication Technologies* (Embedded ICT). As a consequence, the presence of embedded systems in everyday life is constant and, often, almost invisible. Embedded systems have common characteristics like the periodic execution of a single application (or a very limited set of applications) and the reactivity to external triggers possibly in Real-Time (RT). More in general, other than the main *Functional Requirements* (FR), it is possible to identify several *Non-Functional* ones (NFR) normally relevant in the embedded systems domain (e.g., cost, size, power/energy, etc.) and well known in advance.

During system development, the exploitation of proper HW/SW technologies normally allows to satisfy all the design requirements. However, it is not simple to identify the ones to be used, given the great number of heterogeneous available HW/SW components and the strong interdependence among the involved tasks. Furthermore, nowadays it is also possible to implement different functionality on a single chip to reduce manufacturing cost and design time. Such systems can include several heterogeneous processors (i.e., by following the classification provided in [1]: *General-Purpose*, GPP; *Application Specific*, ASP; *Single Purpose*, SPP), memories, and a set of interconnection links among them. In this context, the correct choice of the processor technologies (mainly as GPP/ASP that execute SW vs SPP that implement specific functionality directly in HW) plays an essential role in the design activity and HW/SW Co-Design methodologies with related *Electronic Design Automation* (EDA) tools, are of fundamental support for the designers. Unfortunately, there are no fully engineered general methodologies defined for this purpose and often the best option is still to refer to designers experience.

In such a context, an additional challenge is the recent switch from single-processor/core to (heterogeneous) parallel HW/SW platforms used to execute embedded applications with different levels of criticality (i.e., *Mixed-Criticality Embedded Systems*, MCESs). The main problem in the management of a MCES is to ensure that low criticality applications do not interfere with high criticality ones. This type of systems can be found in many domains such as aerospace (e.g., Integrated Modular Avionic, IMA [2]) and automotive industry [3]. Critical and non-critical applications can be further divided by identifying different criticality classes. The goal is always to allow these applications to interact and coexist on the same platform, but a proper management of such *Mixed Criticality* (MC) systems becomes a very complex task that poses several challenges also from the implementation point of view [4]. The basis for integrating various critical applications are the isolation mechanisms that allow to enforce temporal and spatial separation [5]. As an example, according to this approach, embedded applications with different levels of criticality can be allocated on different "partitions" by exploiting *Hypervisors* (HPVs) technologies, which can be verified and validated in isolation (e.g., PikeOS [6], Xtratum [7]). Another approach is to allocate them on different HW components. The identification of the best solution is not always possible so heuristics methods are needed to support MCESs designers. At *Electronic System Level* (ESL) of abstraction, there are very few works that introduce *Mixed-Criticality* (MC) issues directly into a HW/SW co-design flow.

So, according with the scenario described above, this work extends an existing open-source SystemC-based HW/SW Co-Design Environment for Heterogeneous Parallel Systems ([8], [9]) by introducing RT and MC requirements as additional non-functional constraints. More in detail, the focus is on: the extension of the *Design Space Exploration* (DSE) approach to take into consideration MC constraints; the introduction of the "SW partition" concept as provided by HPV technologies; the integration of Xamber external tool for HPV configuration and (semi)automatic code generation.

The remainder of the paper is organized as follows: Section II presents related works that consider mixed-critical requirements into the whole design flow. Section III describes the reference HW/SW co-design framework. Then, Section IV analyzes experimental results. Finally, Section V closes the paper with some conclusions and future works.

II. HW/SW Co-Design for Mixed-Criticality Applications

A remarkable number of research works have focused on ESL HW/SW Co-Design of *Dedicated Heterogeneous Multi-Processor Systems* (D-HMPS). As presented in [10], the HW/SW Co-Design has a long history of more than 30 years. The main problem is related to (automatically) find a solution (in terms of final platform implementations) able to consider at the same time different FR/NFR starting from an ESL of abstraction, in a HW/SW Co-Design context. Since this work focuses particularly on RT and MC requirements, in the following, the most similar works in the literature are analyzed and compared with the proposed one.

In this scenario, AUTOFOCUS3 [11] proposes a model-based development process introducing safety-oriented constraints associated to computing components. The tool assigns the levels of criticality to application tasks and computing resources, avoiding the allocation of high-criticality tasks to low-criticality resources. AUTOFOCUS considers a generic System "Logical Architecture", that is intrinsically non behavioural. Moreover, the safety constraints are related to SIL classification, meanwhile it avoids allocation of higher-criticality level tasks on different lower-criticality level resources, but they do not consider different criticality level tasks interference.

CONTREP [12] is a framework supporting UML/MARTE based modeling, analysis and design of Cyber-Physical Systems (CPS), also with MC constraints. It is based on the CONTREX UML/MARTE modeling methodology [13] with some SysML features integration and considers safety constraints into the different design activities. CONTREP allows to convert MARTE models into ForSyDe-SystemC simulatable models for a formal functional validation. CONTREP offers schedulability analysis producing models that can be processed by the MAST schedulability analysis tool [14]. CONTREP enables embedded software synthesis for heterogeneous multi-core platforms using eSSYN features [15]. The CONTREP framework allows the user to select a specific exploration tool for DSE (i.e., *Multicube Explorer* [16]). CONTREP is then combined with simulation-based tools (i.e., VIPPE [17]) to perform timing and power analysis. To support MC constraints, CONTREP applies a minor extension to MARTE profile, adding a criticality attribute to a NonFunctionalProperties subprofile as an integer to denote an abstract criticality level. CONTREP offers also MC-aware schedulability analysis and architectural mapping validation. Note that DSE does not consider MC requirements, while the CONTREP modeling methodology offers the possibility to check MC constraints fulfillment, and CONTREP does not consider HPV technologies into the design flow.

ForSyDe (Formal System Design) models [23] have been enabled as a design entry to an analytical DSE tool, called DeSyDe [19]. DeSyDe is a modular tool which provides a DSE for bare-metal applications, finding implementations for a set of tasks on a shared *Multi-processor System-on-chip* (MPSoC) starting from synchronous dataflow graphs (SDFGs) and a predictable model for target platform. DeSyDe also offers support for MC, in the sense that constraints on performance and cost metrics can be hard for some applications, and they will be implemented on specific safety resources, while other applications are provided with best-effort service on the remaining resources [24]. In that work, the concept of Mixed-Criticality System (MCS) is not-well established since it refers only to the complexity in implementing such MCS on specific (safety) resources.

The work in [20] [25] presents a design methodology, called OSSS (Oldenburg System Synthesis Subset), for MPSoC. The OSSS design entry point is the *Application Layer* (AL), the platform model is a *Virtual Target Architecture Layer* (VTAL), while a manual mapping between application tasks and system components allows to simulate system behaviour to check input requirements. Using FOSSY (Functional Oldenburg System SYnthesiser) tool [26], it is possible to synthesize a specific target platform in a separate step using AL and VTAL models. In [27] [28] the authors extend the methodology introducing MCS constraints, presenting the OSSS-MC system methodology. In these works OSSS-MC partitions the

TABLE I
CLASSIFICATION OF DIFFERENT ESL METHODOLOGY APPROACHES (CONSIDERING MC ISSUES).

	Specification		**Specification**	**Implementation**			**Decision Making**		**Refinement**	
ESL Approach	Appl.^{1}	Arch.^{2}	**Language**	MoS^3	MoP^4	**DSE**^{5}	Comp.^{6}	Comm.^{7}	Comp.^{6}	Comm.^{7}
AUTOFOCUS3 [11]	PN	HeMPES	Custom	Component	TAPM	●	●	○	●	-
CONTREP [18]	UML, SDF	HeMPES	MARTE & SysML & SystemC	TLM	TAPM	●	●	○	●	○
DeSyDe [19]	SDF	HeMPSoC	XML	-	-	●	●	○	●	○
OSSS-MC [20]	OSSS/MC	HeMPSoC	SystemC	TLM	T/ISAPM	-	●	-	●	-
MultiPARTES [21]	UML	HoMPES	MARTE	TLM	TAPM	●	●	-	●	-
This work [22]	CSP	HeMPES	SystemC	TLM	T/ISAPM	●	●	●	●	○

^{1} Appl.: Application Model; ^{2} Arch.: Architecture Model; ^{3} MoS: Model of Structure; ^{4} MoP: Model of Performance; ^{5} DSE: Design Space Exploration;
^{6} Comp.: Computation; ^{7} Comm.: Communication; CP: Constraint Programming; PN: Process Network; CSP: communicating sequential processes; UML: Unified Modeling Language; SDF: Synchronous Dataflow;
He/HoMPS: Heterogeneous/Homogeneous Multi-processor Systems; He/HoMPSoC: Heterogeneous/Homogeneous Multi-processor System on chip; He/HoMPES: Heterogeneous/Homogeneous Multi-processor Embedded System;
TLM: Transaction-Level Model; TAPM: Task Accurate Performance Model; ISAPM: Instruction Set Accurate Performance Model; CAPM: Cycle-Accurate Performance Model;

system behaviour into tasks and shared objects, clustered using criticality levels, defining a criticality-dependant end-to-end execution time and a criticality-dependant behaviour regarding system mode of execution respect to criticality levels.

A work that considers HPV and a methodology to identify a set of HPV-based SW partitions is MultiPARTES [21]. It relies on *Model Driven Engineering* (MDE) toolset and offers HPV partitions identification (i.e., Xtratum software partitions) and application allocation. However, MultiPARTES considers only a fixed multi-core architecture, managing HPV partitions only in a homogeneous multi-processor platform.

Starting from this list of methodologies considering MC requirements, it is possible to classify them updating the table described in [29] with some modifications and new tools. Table I presents a classification of this different ESL methodologies in terms of application and platform specification, DSE support and refinement activities. The specification column specifies the application models, in terms of Model of Computation (MoC) and meta-models, and the corresponding specification languages, and the platform architecture, in terms of heterogeneous/homogeneous multi-processor ones. From an implementation point of view, the Model of Structure (MoS) represents the system architecture and structure. MoS may be a netlist with a semantics limited to describing component connectivity or a *Transaction Level Model* (TLM) that tries to abstract as-much-as-possible architectural concepts. In order to estimate performances (in terms of timing, power/energy, cost/area etc.), a Model of Performance (MoP) is defined as a model where each individual elements is associated to a quality numbers respect to specific given implementations. The granularity is a measurements of the accuracy associated to each implemented elements in different solutions. It may be cycle-accurate (using RTL representations), instruction-set-accurate (using also Instruction-Set Simulators, ISS), or task-accurate (with estimations at a high level of abstraction). This quality numbers are used by the DSE step to find different implementation alternatives. Finally, starting from the specification, the synthesis is driven by decision making (into the DSE step) and system refinements (in terms of ESL synthesis activities) with both computational and communication elements. In the Table I, all the previous classification criteria are taken into account, where a full circle implies that the ESL aspect is fully supported, while an open circle means a partial support (and a partial automation). It is worth noting that this work fully support all the ESL synthesis criteria presents in the Table I, while future works will try to complete the missing features in terms of communication refinements and also external tools integration, to make advanced comparisons and to validate the final methodology.

III. REFERENCE HW/SW CO-DESIGN FRAMEWORK

In the context of MCESs, this work exploits an existing open-source ESL HW/SW co-design framework ([8], [9]), and extends it by introducing the possibility to consider also MC and RT requirements (the extended framework is called HEPSYCODE-MC: *HW/SW Co-Design of Heterogeneous Parallel Dedicated Systems with Real-Time and Mixed-Criticality Constraints*).

While the general methodology has been described in [30] [31], this work proposes a specialization of the reference framework, in the context of MegaM@Rt2 [32] European Project, in order to define a DSE methodology able to take into account MCES based on Xtratum HPV [7]. In particular, the work focuses on agnostic models for partitioned MCES into multi-core systems and on generation of automatic projects/code of partitions, using model transformation between Xamber tool [33] and HEPSYCODE-MC approach (Fig. 1).

Xamber is a graphical configuration tool adapted to assist the user through completion of the configuration of partitioned systems, and provides an interface for capturing and editing the elements that are part of the system. Xamber generates the configuration file needed by a HPV (i.e., Xtratum) to execute the system. Meanwhile, XtratuM [7] is a bare metal HPV supporting paravirtualization for multiple architectures. XtratuM natively supports SPARC architecture and LEON processors.

Starting from HEPSYCODE-MC methodology (and related tools) and Xamber tool, an integration step has been performed in order to check overlapping functionality and to exploit HEPSYCODE-MC framework functionality.

The list of activities involves different modeling and design adaptation in the HEPSYCODE-MC HW/SW Co-Design Flow in order to introduce HPV technologies in the DSE step, by considering a System-Level RT MoC based on Communicating Sequential Processes (CSP), modified with some for-

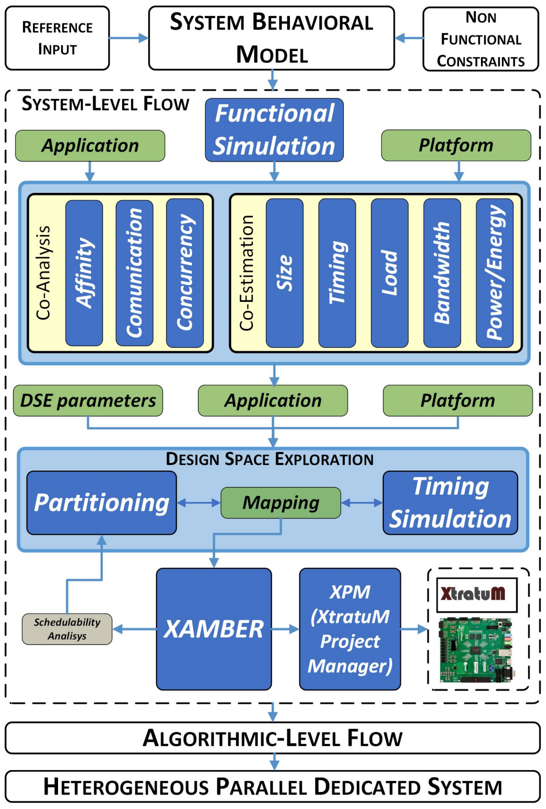

Fig. 1. HEPSYCODE-MC - Xamber Integration.

mal communication constraints with respect to unidirectional point-to-point blocking channels that allow tasks communication in a deterministic network model.

Starting from a CSP System Behavioural Model (CSP-SBM), representing an executable model of the application behavior, splitting processes into pieces of code that represent tasks in the RT domain (creating the so called Process Interaction Model, PIM), it is possible to transform the initial CSP application model into the final Process to Task Graph Model (PTM), conform to the most used RT standards, as presented in [34]. After these assumptions and related transformation activities, the integration between HEPSYCODE-MC and Xamber has been realized with a methodology change in the HEPSYCODE-MC framework, as shown in Fig. 1.

The rest of the paper describes the integration activity in details.

A. HML Specification

The reference System-Level modelling language in HEPSYCODE-MC is the Hepsy Modeling Languages (HML), where the application is described by a process network connected via synchronous channels. In the HEPSYCODE-MC environment, the application described via HML is transformed into a System Behaviour Model (SBM). The SBM is a Communicating Sequential Process (CSP-based) executable Model of Computation (MoC) of the system behavior that explicitly defines also a model of communication among processes using unidirectional point-to-point blocking channels for data exchange. An example of HML application is shown in Fig. 2.

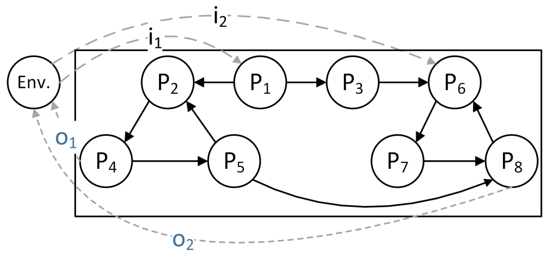

Fig. 2. HEPSYCODE-MC HML example.

The reference language in HEPSYCODE-MC is the SystemC, a C++ class library able to capture and define system specification. The SBM is implemented by SystemC modules and threads. Starting from the SBM code and following the CSP-to-RT adaptation step described in [34], it is possible to transform the CSP (concurrent process network model, not suitable for modeling RT scenarios) into a task-graph DAG model.

B. Metrics Evaluation

After the modeling step, several metrics evaluations and estimations have been performed and the execution time associated to each task has been estimated by means of HEPSIM [35].

C. Design Space Exploration

The DSE step is able to find a solution, in terms of HPV-based SW partitions in a heterogeneous multi-processor parallel scenario. In order to transform HEPSYCODE-MC input models into Xamber projects, it is needed to fix some parameters:

1) Only single core scenarios (with LEON3 processor cores) is permitted (Xamber support only single-core scenario at the moment);
2) No other basic HW components (in terms of extra processors connected into a heterogeneous distributed scenarios) are considered in the DSE step;
3) To consider a safety-critical scenario, a criticality level is assigned to different processes respect to their functionality;
4) A maximum number of 4 Xtratum SW partitions are allowed in the DSE.

D. HEPSYCODE-MC - Xamber Project Transformation

The final integration between HEPSYCODE-MC and Xamber has been realized with a methodology change in the HEPSYCODE-MC framework, as shown in Fig. 1. Using a transformation between XML schemas, the Partitioning solution, saved in a XML exchange file, has been translated into a Xamber compliant project, and schedulability analysis has been performed in order to find the best Hyperplan for the initial task set, setting Xamber project parameters (in terms of tasks, processors, partitions, RT parameters and so on) from HEPSYCODE-MC Co-Analysis, Co-Estimation and Partitioning steps.

The application model, the platform model, the partition model and the mapping among these entities have been transformed into a Xamber compliant project, using Java Architecture for XML Binding (JAXB) technology. JAXB is a software framework that allows Java developers to map Java classes to XML representations using marshal transformation.

All the Xamber parameters are derived from the Hepsycode framework (i.e., processes execution time and partitions allocation, End-To-End-Flow representation, IPC channels partitions) . This transformation allows also to use the Contrex tool that performs schedulability analysis and find the best hyper-plan for the reference application. After this activity, Xamber produces the Xtratum Configuration file for SparcV8 architectures (file .xmc) and it is possible to perform 2 different activities: (1) simulate the solution into the HEPSYCODE-MC HPV Simulator engine (with hierarchical scheduling feature); (2) check execution time (to check different HPV behavior respect to the specific use case) implementing the proposed input application.

E. HEPSIM Hierarchical Scheduling

In order to to simulate HPV timing behaviors, in the context of this work, HEPSIM has been extended by implementing a hierarchical scheduler, i.e., a second-level scheduler (i.e., *HEPSCHED2*, 2 Levels HEPSYCODE-MC SCHEDuler) with respect to the standard SystemC one. HEPSCHED2 has been implemented as a SystemC *SC_MODULE* containing a dedicated HEPSCHED2 instance for each instance of processor composing the system. Each HEPSCHED2 instance is implemented as a *SC_THREAD*. The implementation of different analysis mechanisms and scheduling policies in HEPSIM is based on the instrumentation of code by means of macros and their interaction with the *HEPSCHED2*. Macro *S* is inserted as a prefix to the SystemC statements composing the SBM to support the handshake mechanism for the scheduling of processes as shown in Listing 1.

Listing 1. HEPSCHED2 Full Handshake: Interactions with macro S

```
/* Macro S */
#define S(X) \
    pSystemManager->Increase(X); \
    if(!pSystemManager->checkSPP(X)) \
        wait(pSchedulingManager->schedule[X]); \
............
/* HEPSCHED2 */
if(ready[ps.id]==true){
    schedule[ps.id].notify(SC_ZERO_TIME);
    wait(release[ps.id]);
}
............
/* Macro S */
wait(pSystemManager->upSimTime(X)); \
    if(!pSystemManager->checkSPP(X)) \
        pSchedulingManager->release[X].notify(SC_ZERO_TIME);
#endif
............
/* The handle goes to HEPSCHED2 */
```

When control passes from *S* to the HEPSCHED2 instance (i.e., a *SC_THREAD*) associated to the processor that executes the process and support HPV technologies, a further handshake between processor and the related partition (a sort of *Partition Manager*) has been implemented. This mechanism lasts as long as the duration of the time slice associated to each partitions. To avoid partition overrun, HEPSCHED2 controls if the time needed for the execution of the next ready process statement exceeds the time bound (i.e., the partition time slice) associated to the considered partition. HEPSCHED2 is also able to define a specific *Partition Hyperplan*, as suggested by the DSE step. Then, after the handshake among the HEPSCHED2 and the macro *S*, the control came back to the SystemC scheduler.

IV. Experimental Results

This section presents some results regarding the simulation and the implementation of a specific use case. The reference use case taken into account is shown in Fig. 3, where the FirFirGCD application presented in [31] has been changed to match RT DAG representation.

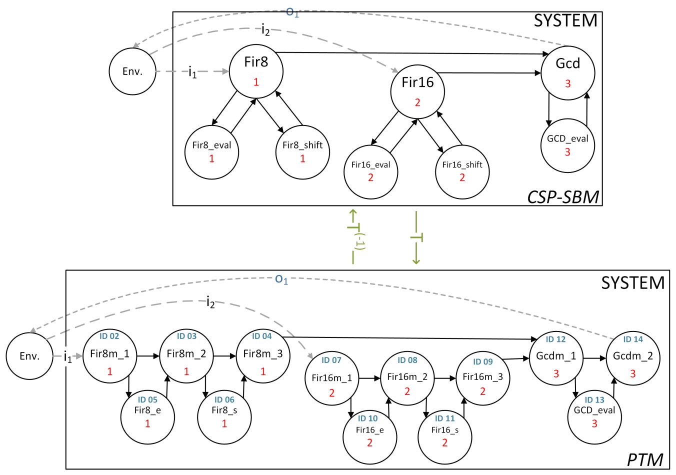

Fig. 3. CSP-SBM to *Process to Task Graph Model* (PTM) example transformation.

In this example, the initial CSP processes are divided into different tasks, by following to the transformation pattern defined in [34]. This transformation is driven by the CSP MoC. In Fig. 3, i_1 and i_2 are system inputs, o_1 is the system output, while the red number under the name of each process represents the criticality level that has been associated to processes (the value has been assigned depending on the number of communicating channels and interactions among different processes in order to verify the proposed methodology). The resulting processes inherit the criticality levels associated to the corresponding CSP-SBM processes.

The process communication matrix (the number of bits exchanged among the different processes) is shown in Table II.

Considering the example model in Fig. 3, some metric results has been evaluated by means of timing simulation activities, as described in [35]. During the Load Estimation

TABLE II
PROCESS COMMUNICATION.

Process ID	2	3	4	5	6	7	8	9	10	11	12	13	14
2	0	80	0	1630	0	0	0	0	0	0	0	0	0
3	0	0	80	0	720	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	80	0	0
5	0	190	0	0	0	0	0	0	0	0	0	0	0
6	0	0	640	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	80	0	2990	0	0	0	0
8	0	0	0	0	0	0	0	80	0	1360	0	0	0
9	0	0	0	0	0	0	0	0	0	0	80	0	0
10	0	0	0	0	0	0	190	0	0	0	0	0	0
11	0	0	0	0	0	0	0	1280	0	0	0	0	0
12	0	0	0	0	0	0	0	0	0	0	0	160	0
13	0	0	0	0	0	0	0	0	0	0	0	0	80
14	0	0	0	0	0	0	0	0	0	0	0	0	0

step, the ”Free Running Time” (FRT, the application execution time when all the processes are allocated on the same BB instance) has been estimated for the BB with LEON3 PU, and the values is equal to 0.0138695 s. The ”Free Running Loads” (FRLs, the load evaluated for each process when all the processes are allocated on the same BB instance) and the average execution time for each process (called Average-Case Response Time, ACRT) are presented in Table III.

TABLE III
PROCESS FREE RUNNING LOAD AND AVERAGE-CASE RESPONSE TIME.

Process ID	FRL	ACRT
2	0.032503	263 μ s
3	0.0301813	243 μ s
4	0.0198998	175 μ s
5	0.162515	1085 μ s
6	0.119067	850 μ s
7	0.0298497	263 μ s
8	0.0262014	234 μ s
9	0.0172465	175 μ s
10	0.256044	2050 μ s
11	0.179761	1554 μ s
12	0.030513	293 μ s
13	0.0859007	243 μ s
14	0.0102816	117 μ s

The FRLs have been calculated using the HEPSIM timing simulator. It is worth noting that such information are useful in order to find better tasks allocation among HPV software partitions (also considering tasks workload, concurrency, ACRT parameters, and MC requirements). Considering single core scenarios (because Xamber do not support multi-core architectures), only communication and criticality level are used to allocate and bind tasks on different partitions, then a schedulability analysis allows to refine the allocation in order to verify the correct behavior of tasks execution, considering also several system overhead (e.g., HPV context switch, tasks context switch, IPC). Moreover, HEPSYCODE-MC DSE produces one solution represented in Table IV.

This solution has been transformed into Xamber compliant project, as shown in Fig 4. Finally, the hyper-plan generated by Contrex tool is presented in Listing 2

Listing 2. FirFriGCD Contrex Hyper-plan

TABLE IV
DSE WITH HPV-BASED SW PARTITIONS.

	Cost	**Partition**				**PAM1**
Iteration	**Function**	**1**	**2**	**3**	**4**	**Execution Time**
0	1.5*	9,11,12,14	3,5,7,8,10,13	2,4,6	-	0.007046 s
22	0.516461	12,13,14	2,3,4,5,6	7,8,10,11	9	0.322346 s
25	0.50823	7,8,9,10,11	3,5,6	2,4	12,13,14	0.534798 s
35	0.502418	7,8,9,10,11	5	12,13,14	2,3,4,6	3.17861 s
36	0.5	2,3,4,5,6	12,13,14	7,8,9,11	10	3.63263 s
100	0.5	2,3,4,5,6	12,13,14	7,8,9,11	10	33.5264 s

* Unfeasible Solution (Minimum Cost function $\geq$ 1 [31])

```
<SystemDescription xmlns="http://www.xtratum.org/xm-3.x"
version="1.0.0" name="FirFirGCD">
<HwDescription><ProcessorTable><Processor id="0" freq="75MHz">
<CyclicPlanTable>
<Plan name="Plan_Auto" id="0" majorFrame="16.116ms">
<Slot id="0" start="0ms" duration="6.167ms" partId="1"/>
<Slot id="1" start="6.167ms" duration="0.543ms" partId="3"/>
<Slot id="2" start="6.710ms" duration="4.120ms" partId="4"/>
<Slot id="3" start="10.830ms" duration="3.953ms" partId="3"/>
<Slot id="4" start="14.783ms" duration="1.333ms" partId="2"/>
</Plan>
</CyclicPlanTable>
</Processor></ProcessorTable></HwDescription>
```

Using this hyper-plan configuration into the HEPSIM simulator, the output in Table V has been produced, where each lap represents the execution time of one instance of the hyper-plan (in seconds). It is worth noting that the HEPSIM simulation follows the hyper-plan defined in the Xamber configuration tool (for each one of the 10 input triggers) without timing errors.

For the system implementation, the LEON3 General Purpose processor (GPP) has been considered. LEON3 is a 32-bit synthesizable soft-processor that is compatible with SPARC V8 architecture: it has a seven-stage pipeline and Harvard architecture, uses separate instruction and data caches and supports multiprocessor configurations in Symmetric Multiprocessor (SMP) mode. It represents a soft-processor for aerospace applications. The single-core reference implementation is shown in Fig. 5. The development board is the Xilinx ML605 Virtex-6 FPGA with 512 MB RAM.

Starting from GRLIB, a VHDL library of IP cores for designing a complete system on chip centered around the LEON3 processor, the LEON3 processor has been customized with a system clock of 75 MHz per core and the following characteristics:

- 1 Cobham Gaisler LEON3 SPARC V8 Processor connected with AHB shared bus;
- 8 register windows;
- GRFPU High-Performance Floating-Point Unit;
- 2*8 KiB instruction caches, with 32 bytes per line with Least-Recently-Used (LRU) replacement algorithm;
- 2*4 KiB data caches, with 16 bytes per line with LRU replacement algorithm.

The final software partitioned system (suggested by the DSE activity) uses the Xtratum services to implement the FIR-FIR-GCD use case. All the processes are implemented as a bare-metal application into the partitions, where communication is allowed using sampling channels. Fig. 6 presents the comparison between the execution time on the real target (LEON3

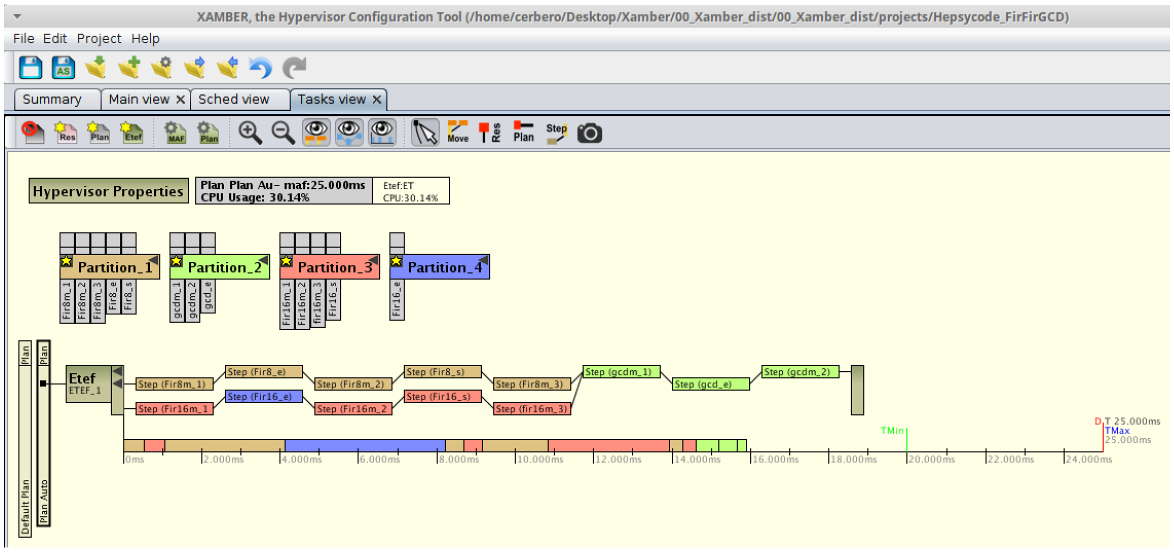

Fig. 4. HEPSYCODE-MC - Xamber transformation (Task View).

TABLE V
HEPSIM HPV Timing Simulation.

	HPV Hyper-Plan LAPS (s)									
Partition	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
1	0.006170	0.022297	0.038433	0.054567	0.070703	0.086835	0.102967	0.119103	0.135239	0.151373
3	0.006715	0.022841	0.038977	0.055113	0.071249	0.087381	0.103513	0.119649	0.135785	0.151919
4	0.010837	0.026966	0.043102	0.059238	0.075374	0.091506	0.107638	0.123774	0.139911	0.156044
3	0.014792	0.030924	0.047059	0.063195	0.079331	0.095463	0.111595	0.127731	0.143867	0.160001
2	0.016125	0.032262	0.048396	0.064532	0.080664	0.096796	0.112932	0.129068	0.145202	0.160336
Display	0.015047	0.031184	0.047354	0.063491	0.079736	0.095898	0.111890	0.128026	0.144232	0.159366
Tot. LAP Time	0.016125	0.032262	0.048396	0.064532	0.080664	0.096796	0.112932	0.129068	0.145202	0.160336

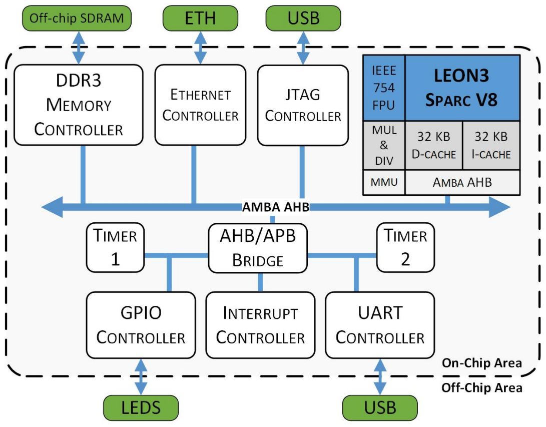

Fig. 5. Single-core LEON3 Hardware reference implementation.

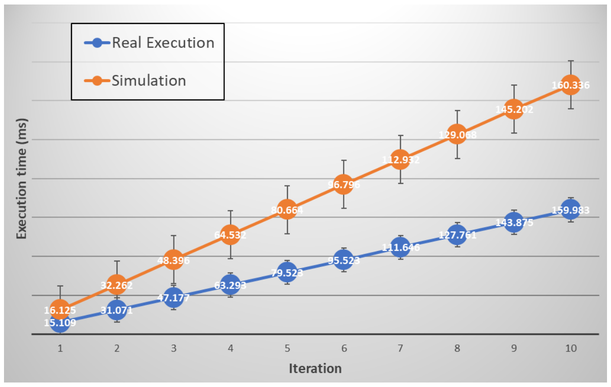

Fig. 6. HPV Simulation (HEPSIM) vs Real Execution (on a Single-core LEON3).

single-core) and the simulation made with HEPSIM. The final average error estimation is under 2 %, so the simulator is able to evaluate HPV timing behavior with a very limited error.

V. Conclusion and Future Work

This work has presented an ESL HW/SW Co-Design approach able to take into account mixed-criticality and real-time constraints. The presented methodology, design flow and framework are able to drive the designer from the input specification to the final implementation solution, while offering timing simulation capabilities, DSE activities with the support of analysis tool, integrating this approach with external tools (in this scenario Xamber [33], but other tools are under evaluation). Despite of the obtained results, a lot of works should be made in future in order to consider the multi-core scenario [8] while introducing schedulability and real-time analysis, introduce fixed WCET values (taken from external tools) to be used also in the DSE step to improve allocation and binding of processes/tasks, and to improve performance, integrate other external tools to enhance HEPSYCODE-MC functionality (i.e., art2kitekt [36], CHESS [37], CODEO [38]), and improve the hierarchical scheduling implementation considering Inter Partition Communication (IPC) overheads by means of benchmarking activities.

ACKNOWLEDGMENT

This work has been partially supported by the ECSEL RIA 2016 MegaM@Rt2, AQUAS and ECSEL RIA 2017 FitOptivis projects.

REFERENCES

[1] T. G. F. Vahid, *Embedded system design: A unified hard-ware/software approach.* Wiley, 1999.

[2] P. J. Prisaznuk, "Integrated modular avionics," in *Proceedings of the IEEE 1992 National Aerospace and Electronics Conference, NAECON 1992*, pp. 39–45 vol.1, May 1992.

[3] R. I. D. A. Burns, "Mixed criticality systems - a review," in *Research report, University of York*, 2015.

[4] S. Baruah, H. Li, and L. Stougie, "Towards the design of certifiable mixed-criticality systems," in *2010 16th IEEE Real-Time and Embedded Technology and Applications Symposium*, pp. 13–22, April 2010.

[5] R. Pellizzoni, P. Meredith, M. Y. Nam, M. Sun, M. Caccamo, and L. Sha, "Handling mixed-criticality in soc-based real-time embedded systems," in *Proceedings of the Seventh ACM International Conference on Embedded Software*, EMSOFT '09, (New York, NY, USA), pp. 235–244, ACM, 2009.

[6] *PikeOS Hypervisor*, 2018 (accessed: 31.03.2019). https://www.sysgo.com/products/pikeos-hypervisor/.

[7] M. Masmano, I. Ripoll, and A. Crespo, "Xtratum: a hypervisor for safety critical embedded systems," in *11th Real Time Linux Workshop*, 2012.

[8] L. Pomante, "Hw/sw co-design of dedicated heterogeneous parallel systems: an extended design space exploration approach," *IET Computers & Digital Techniques*, vol. 7, pp. 246–254, Nov 2013.

[9] L. Pomante, "System-level design space exploration for dedicated heterogeneous multi-processor systems," in *ASAP 2011 - 22nd IEEE International Conference on Application-specific Systems, Architectures and Processors*, pp. 79–86, Sept 2011.

[10] J. Teich, "Hardware/software codesign: The past, the present, and predicting the future," *Proceedings of the IEEE*, vol. 100, pp. 1411–1430, May 2012.

[11] S. Voss, J. Eder, and F. Hölzl, "Design space exploration and its visualization in autofocus3," in *Software Engineering*, 2014.

[12] F. Herrera, P. Peñil, and E. Villar, "A model-based, single-source approach to design-space exploration and synthesis of mixed-criticality systems," in *Proceedings of the 18th International Workshop on Software and Compilers for Embedded Systems*, SCOPES '15, (New York, NY, USA), pp. 88–91, ACM, 2015.

[13] *Contrex project*, 2018 (accessed: 31.03.2019). https://contrex.offis.de/home/.

[14] *MAST - Modeling and Analysis Suite for Real-Time Applications*, 2018 (accessed: 31.03.2019). https://mast.unican.es/.

[15] *eSSYN - Embedded software synthesis*, 2018 (accessed: 31.03.2019). http://essyn.com/.

[16] V. Zaccaria, G. Palermo, F. Castro, C. Silvano, and G. Mariani, "Multicube explorer: An open source framework for design space exploration of chip multi-processors," in *23th International Conference on Architecture of Computing Systems 2010*, pp. 1–7, Feb 2010.

[17] *VIPPE: Virtual Platform Parallel Performance Evaluation*, 2018 (accessed: 31.03.2019). http://vippe.teisa.unican.es/.

[18] F. Mallet, E. Villar, and F. Herrera, *MARTE for CPS and CPSoS*, pp. 81–108. Singapore: Springer Singapore, 2017.

[19] *DeSyDe: Design space exploration for System Design*, 2018 (accessed: 31.03.2019). https://github.com/forsyde/DeSyDe.

[20] A. Schallenberg, W. Nebel, A. Herrholz, P. A. Hartmann, and F. Oppenheimer, "Osss+r: A framework for application level modelling and synthesis of reconfigurable systems," in *2009 Design, Automation Test in Europe Conference Exhibition*, pp. 970–975, April 2009.

[21] S. Trujillo, A. Crespo, and A. Alonso, "Multipartes: Multicore virtualization for mixed-criticality systems," in *2013 Euromicro Conference on Digital System Design*, pp. 260–265, Sept 2013.

[22] *Hepsycode: A System-Level Methodology for HW/SW Co-Design of Heterogeneous Parallel Dedicated Systems*, 2018 (accessed: 31.03.2019). http://www.hepsycode.com.

[23] S.-H. Attarzadeh-Niaki, E. Altinel, M. Koedam, A. Molnos, I. Sander, and K. Goossens, *A Composable and Predictable MPSoC Design Flow for Multiple Real-Time Applications*, pp. 157–174. Springer, 12 2017.

[24] K. Rosvall, N. Khalilzad, G. Ungureanu, and I. Sander, "Throughput propagation in constraint-based design space exploration for mixed-criticality systems," in *Proceedings of the 9th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*, RAPIDO '17, (New York, NY, USA), pp. 4:1–4:8, ACM, 2017.

[25] K. Grttner, P. A. Hartmann, P. Reinkemeier, F. Oppenheimer, and W. Nebel, "Challenges of multi- and many-core architectures for electronic system-level design," in *2011 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation*, pp. 331–338, July 2011.

[26] *FOSSY: Functional Oldenburg System SYnthesiser*, 2018 (accessed: 31.03.2019). http://system-synthesis.org/synthesis/home.

[27] P. Ittershagen, K. Gruttner, and W. Nebel, "Mixed-criticality system modelling with dynamic execution mode switching," in *2015 Forum on Specification and Design Languages (FDL)*, pp. 1–6, Sept 2015.

[28] P. Ittershagen, K. Grüttner, and W. Nebel, "An integration flow for mixed-critical embedded systems on a flexible time-triggered platform," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 23, pp. 51:1–51:25, May 2018.

[29] A. Gerstlauer, C. Haubelt, A. D. Pimentel, T. P. Stefanov, D. D. Gajski, and J. Teich, "Electronic system-level synthesis methodologies," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, pp. 1517–1530, Oct 2009.

[30] V. Muttillo and G. Valente, "Injecting hypervisor-based software partitions into design space exploration activities considering mixed-criticality requirements," in *6th EUROMICRO/IEEE Workshop on Embedded and Cyber-Physical Systems*, ECYPS 2018, IEEE, 2018.

[31] V. Muttillo, G. Valente, and L. Pomante, "Design space exploration for mixed-criticality embedded systems considering hypervisor-based sw partitions," in *Euromicro Conference on Digital System Design (DSD 2018)*, DSD '18, 2018. Best Poster Award.

[32] W. Afzal, H. Bruneliere, D. D. Ruscio, A. Sadovykh, S. Mazzini, E. Cariou, D. Truscan, J. Cabot, A. Gmez, J. Gorroogoitia, L. Pomante, and P. Smrz, "The megam@rt2 ecsel project: Megamodelling at runtime scalable model-based framework for continuous development and runtime validation of complex systems," *Microprocessors and Microsystems*, vol. 61, pp. 86 – 95, 2018.

[33] *Xamber*, 2018 (accessed: 31.03.2019). http://www.fentiss.com/en/products/xamber.html.

[34] V. Muttillo, G. Valente, D. Ciambrone, V. Stoico, and L. Pomante, "Hepsycode-rt: A real-time extension for an esl hw/sw co-design methodology," in *Proceedings of the Rapido'18 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*, RAPIDO '18, (New York, NY, USA), pp. 6:1–6:6, ACM, 2018.

[35] V. Muttillo, G. Valente, D. Ciambrone, and L. Pomante, "Hepsim: an esl hw/sw co-simulator/analysis tool for heterogeneous parallel embedded systems," in *6th EUROMICRO/IEEE Workshop on Embedded and Cyber-Physical Systems*, ECYPS 2018, IEEE, 2018. Best Paper Award.

[36] H. Isakovic, R. Grosu, D. Ratasich, J. Kadlec, Z. Pohl, S. Kerrison, K. Georgiou, K. Eder, N. Druml, L. Tadros, F. Christensen, E. Wheatley, B. Farkas, R. Meyer, and M. Berekovic, "A survey of hardware technologies for mixed-critical integration explored in the project emc2," in *Computer Safety, Reliability, and Security* (S. Tonetta, E. Schoitsch, and F. Bitsch, eds.), (Cham), pp. 127–140, Springer International Publishing, 2017.

[37] *CHESS: Composition with guarantees for High-integrity Embedded Software components assembly*, 2018 (accessed: 31.03.2019). http://www.en.intecs.it/page/chess.

[38] *PikeOS Hypervisor Eclipse based CODEO*, 2018 (accessed: 31.03.2019). https://www.sysgo.com/products/pikeos-hypervisor/eclipse-based-codeo/.