A Model-Based ESL HW/SW Co-Design Framework for Mixed-Criticality Systems

F. Federici, V. Muttillo, L. Pomante, P. Serri, G. Valente
Università Degli Studi Dell'Aquila - Center of Excellence DEWS
Via Giovanni Gronchi, 18, 67100, L'Aquila, Italy
{vittoriano.muttillo, paolo.serri, giacomo.valente}@graduate.univaq.it
{fabio.federici, luigi.pomante}@univaq.it

Abstract — **In the last few years, model-based design techniques have provided a set of design environments which facilitate the designers work. Engineers can find and correct errors, analyze performances and early validate a system, by minimizing the temporal and financial impact of system modification. Moreover mixed-criticality systems are becoming of great interest in embedded system area but there is a lack of model-based tools in their development. In such a scenario, this work focuses on the development of a framework for modeling, analysis and validation of mixed critical systems, through the exploitation of a "Model-Based Electronic System Level (ESL) HW/SW Co-Design" methodology, refined to use estimates, metrics and simulations able to consider mixed-criticality and real-time requirements. The final goal is an extension of an existing HW/SW co-design methodology in order to support real-time mixed-criticality embedded systems development in industrial and commercial contexts.**

Keywords — electronic design automation; hardware/software co-design; mixed-criticality systems

I. Introduction

Despite the widespread diffusion of embedded systems, a well-defined and general *Electronic-System Level* (ESL) design flow is still missing. The main problems in embedded systems design are to perform an accurate modeling of functional and non functional requirements, and then to check their satisfaction before the final implementation steps. Designers commonly use system-level models (e.g. block diagrams, UML, SystemC, etc.) to gain a complete understanding of the problem and to evaluate the quality of different software to hardware mappings by simulating the related system behavior. In such a context, proper software tools are fundamental to support designers to reduce costs and the overall complexity of systems implementation. Unfortunately, there is no mature general methodology for this purpose and the design process often relies on empirical criteria and qualitative experience-based assessments. Furthermore, the use of multi-core and many-core platforms for mixed-critical embedded systems allows increasing performances but introduces isolation problems in a shared resources scenario that could be very complex to manage.

The remainder of the paper is organized as follows: Section II provides an overview of mixed-criticality systems, Section III describes the proposed HW/SW co-design framework, whereas Section IV discusses a solution to adapt the adopted model of computation (MoC), based on Communicating Sequential Processes (CSP) to the real-time and mixed-critical world. Finally, Section V reports some conclusive considerations and presents future work activities.

II. BACKGROUND

A critical industrial challenge is to integrate multiple applications with different criticality-levels on a single computing platform, both efficiently (in terms of costs) and correctly (to preserve the proper execution). These platforms, usually referred as Mixed-Critical Systems [1], concurrently run applications for which failures may cause risks and danger for people, large losses of money or extensive environmental damages, and others applications for which the effects of a malfunction are normally tolerable and manifest themselves primarily as a Quality of Service (QoS) decay. In this case, the system is designed in order to guarantee that the less critical applications are unable to disturb high critical ones (the most expensive to be designed and validated). The use of multi/many-core embedded platforms allow to significantly improve the integration and performance of this particular systems, but a related relevant problem is to ensure an adequate management of shared resources.

III. PROPOSED FRAMEWORK

In the context of mixed-criticality systems design, this work proposes a specific framework, based on a proper extension of an existing HW/SW co-design methodology [2], that introduce the possibility to specify real-time and mixed-criticality requirements in the set of the non-functional ones.

The system behavior modeling is based on the CSP MoC [3], that allows modeling system behavior as a network of processes communicating through unidirectional synchronous (i.e. rendezvous-based) channels. The following example shows a scenario with three main CSP subsystems:

- Stimulus: single instance process activation
- System: System Behavior Model (SBM)
- Display: output feedback for offline analysis

So, the developed system is modeled by a set of processes and internal or external channels, as shown in Fig. 1.

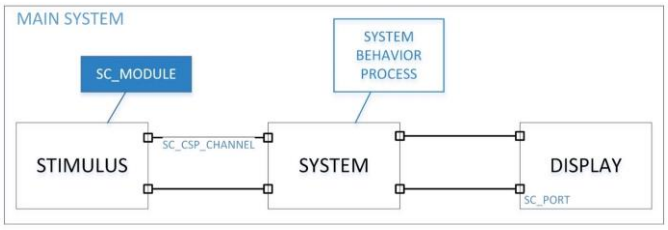

Fig. 1. SystemC CSP Model

The reference ESL HW/SW co-design flow is shown in Fig. 2. The entry point is the *System Behavior Model* (SBM) based on the CSP MoC. In order to list and describe the basic HW elements available to automatically build the final HW architecture, a proper *Technologies Library* (TL) provides a characterization of available processors, memories and interconnection links. The first step of the proposed co-design flow is the *Functional Simulation* where SBM is simulated to check its correctness with respect to some *Reference Inputs*. If SBM is not correct (i.e. wrong outputs or critical conditions such as e.g. deadlocks) it should be properly modified and simulated again. The next step aims at extracting as much as possible information about the system by analyzing the SBM while considering the provided TL. This step is supported by Co-Analysis and Co-Estimation activities.

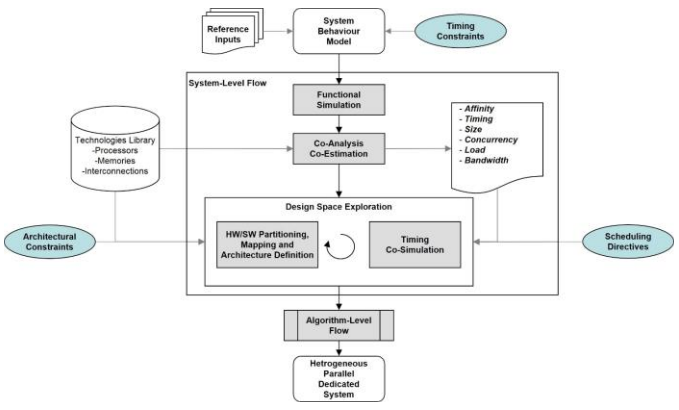

Fig. 2. The reference co-design flow

Finally, the reference co-design flow reaches the Design Space Exploration (DSE) step. It includes two iterative activities:

- "*HW/SW Partitioning, Mapping and Architecture Definition*", based on a genetic algorithm that allows to explore the design space looking for feasible mapping/architecture items suitable to satisfy imposed constraints
- "*Timing Co-Simulation*", that considers suggested mapping/architecture items to actually check for constraints satisfaction.

If the suggested mapping/architecture does not meet defined constraints, the designer should perform a new round of design space exploration by changing some exploration parameters, by modifying the starting SBM, by enriching the TL with new elements, or by relaxing some constraints. When the mapping/architecture item proposed by the DSE step is acceptable, it is possible to proceed with system implementation (i.e. Algorithm Level Flow). For this purpose, the SW-mapped processes are typically transformed in C code, with the optional support of an embedded real-time OS, whereas the HW-mapped ones are transformed in synthesizable HDL code or implemented by means of existing COTS component depending on the final system architecture. This step is fully based on existing commercial algorithm-level methodologies and tools that are out of the scope of this work.

IV. CSP FOR MIXED_CRITICAL REAL-TIME APPLICATIONS

The CSP-like notation adopted in the reference HW/SW co-design methodology doesn't match very well with real-time (RT) constraints (in fact, it actually consider only *Time-To-Completion* constraints) and mutual exclusion issues on shared resources accesses. Therefore, this work proposes an extension/transformation of the CSP-like notation in order to overcome these problems. The idea is to identify a set of items classified as follows:

- *s* (statement): C statement with C++/SystemC data types;
- *p* (process) or *job* (J): consists of a set of *s* divided into two sections (*init* and never-ending *while*) encapsulated in a SystemC SC_THREAD;
- *t* (task): set of competing and/or cooperating *p* or J (with communication rule implemented through CSP-like SC_CSP_CHANNEL) encapsulated in a SystemC SC_MODULE with a given criticality;
- *k* (container): subsystem composed of one or more t with a given criticality;
- *a* (application): application (or system) composed of one or more k (mixed-criticality system);

With these particular items, it is possible to model the system behavior by means of a set of tasks as shown in (1), where n is the number of task instance or processes (jobs).

$$t_i = \{J_{i,1}, J_{i,2}, \dots, J_{i,n}\} \equiv \{p_{i,1}, p_{i,2}, \dots, p_{i,n}\} \qquad (1)$$

In this way, classical real-time parameters (tasks period, deadlines, computational time and so on) can be estimated during the simulation step, mapped on the CSP model, and checked with respect to the ones entered by the designer.

The next step to consider is in which way to model relation constraints. In fact, in some applications, processing cannot be executed in an arbitrary manner, but must comply with precedence relation constraints defined at design stage. These relations are usually described by means of *Directed Acyclic Graph* (DAG). However, in general, the CSP-based modeling approach never create DAG. For this purpose, there are two possible solutions:

- put all the *s* of the *while* section in a *super_s* so that the while section of each *p* becomes a DAG that is repeated (a)periodically;

- force the designer to write *p* as a DAG (by separating *init* and *while* into two *p*) and to work on (a)periodical sequence of *p* in *t* .

Finally, preemption has been introduced by means of points of preemption included in *p*, with a *wait()* in the scheduling policy simulation step. In this phase mutual exclusion constraints on shared resources have not been considered.

In such a scenario, an example model is shown in Fig. 3. The simple starting example is an application represented by one *k*, composed of eight *t*. Each *t* is composed of n_i *p* (*J*), and *s*, where n_i is the number of processes of the task t_i.

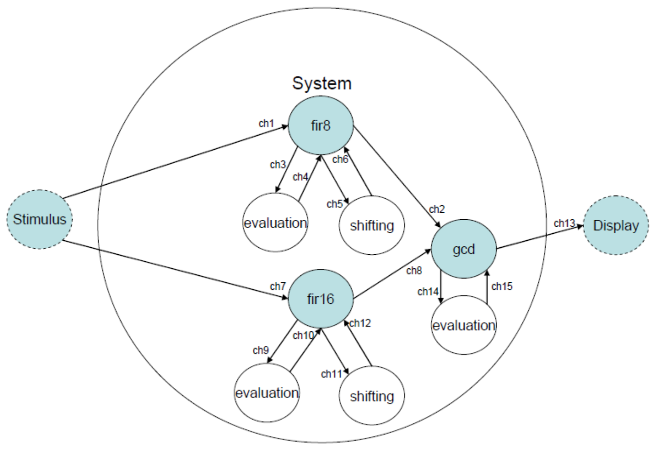

Fig. 3. CSP representing the SBM

The work aims to model different scheduling policies, in order to allow an extend DSE step to provide suggestions also about such policies, and to use the proposed framework for the convergence between CSP-like and RT model (i.e. given a CSP model not conforming to the classical RT “template”, to obtain a proper compliant transformation), also introducing different levels of criticality.

Finally, to offer an integrated IDE for embedded systems electronic designers with real-time and mixed criticality requirements, the GUI implementation exploits the *Eclipse Modeling Framework* (EMF) plug-ins and MDE technologies, which can be used to model a system and to generated code or other outputs. *Sirius* technology is used to allow to create custom graphical modeling workbenches by leveraging the Eclipse modeling tool, whereas *Acceleo Object Management Group* (OMG) *Meta-Object Facility* (MOF) *Model to Text Language* (MTL) is used for model transformation from CSP to SystemC.

V. CONCLUSION AND FUTURE WORKS

This work has proposed an extended and innovative ESL *Electronic Design Automation* (EDA) methodology (and related tools) supporting the development of Mixed-Criticality Embedded Systems. For this, after defined a CSP to RT model transformation, the next step is to further enhance the DSE step to suggest to the designer how to manage different criticality levels of applications, components, and tasks, by means of relevant available technologies (e.g. hypervisors, physical partitioning, etc.). The final result will be a methodology able to support mixed-criticality systems developments by suggesting both the platform and mapping solutions for the specific mixed-criticality application.

ACKNOWLEDGMENT

This work has been supported by the Artemis-JU AIPP 2013 EMC2 (GA 621429) project.

REFERENCES

[1] Burns, A., & Davis, R. (2013). Mixed criticality systems-a review. Department of Computer Science, University of York, Tech. Rep.

[2] Pomante, L. (2011, September). System-level design space exploration for dedicated heterogeneous multi-processor systems. In Application-Specific Systems, Architectures and Processors (ASAP), 2011 IEEE International Conference on (pp. 79-86). IEEE.

[3] Hoare, C. A. R. (1978). Communicating sequential processes (pp. 413-443). Springer New York.