An Efficient Performance-Driven Approach for HW/SW Co-Design

(Vision Paper)

Daniele Di Pompeo
DISIM
Via Vetoio, 1
L'Aquila, Italy
daniele.dipompeo@graduate.univaq.it

Emilio Incerto
GSSI
Viale F. Crispi, 7
L'Aquila, Italy
emilio.incerto@gssi.infn.it

Vittoriano Muttillo
DEWS
Via Vetoio, 1
L'Aquila, Italy
vittoriano.muttillo@graduate.univaq.it

Luigi Pomante
DEWS
Via Vetoio, 1
L'Aquila, Italy
luigi.pomante@univaq.it

Giacomo Valente
DEWS
Via Vetoio, 1
L'Aquila, Italy
giacomo.valente@graduate.univaq.it

ABSTRACT

Nowadays embedded systems are powerful and everywhere. They implement complex functionality relying on a huge set of different hardware and software (HW/SW) architectures. In order to reduce their development effort, HW/SW Co-Design techniques are used during the entire development cycle. These techniques aim at helping designers to define a feasible hardware and software partitioning for the system in such a way that functional and non-functional requirements are fulfilled. In this context Design Space Exploration is a challenging activity since a huge number of different implementation alternatives need to be evaluated.

By exploiting some intrinsic properties of the embedded system domain, in this paper we propose our vision for a novel Performance-Driven HW/SW Co-Design methodology. It combines: (i) the "design for verifiability" concept, suitable to model the system behaviour avoiding the state space explosion problem, and (ii) model-driven techniques to address performance issues. For this goal, we introduce the concepts underlying: (i) a novel formal modeling language and, (ii) a performance-driven verification/transformation chain.

CCS Concepts

•**Hardware** → **Emerging technologies;** •**Software and its engineering** → *Correctness; Formal methods; Software performance;*

Keywords

HW/SW Co-Design, Performance awareness, Model-Driven Engineering, Design for verifiability

1. INTRODUCTION

Nowadays embedded systems are powerful and everywhere, as they are widely used in industries for different kinds of applications (e.g., aircraft autopilots, telecommunication products, medical devices) with different criticality and requirements. As a consequence, the design of such systems is becoming an increasingly complex process since several implementation alternatives with different hardware constraints (e.g., CPU speed, memory limitation, battery life) need to be considered.

In such a context, heterogeneous multi-processor architectures have been introduced with the aim of developing heterogeneous multi-processor systems (HMPS), which are exploited for a wide range of application domains (e.g. [3, 4]). In particular, they are often used to implement digital electronic systems with an application-specific HW/SW architecture aimed at satisfying functional and non-functional requirements.

In order to cope with the growing complexity of those systems, it is fundamental to exploit a design flow that allows an entry point at a high level of abstraction. In view of these considerations, HW/SW Co-Design has been recognized as the mainstream technology for the rapid development of complex HMPS [17]. Basically it aims at helping designers in choosing the right hardware and software partition for the system functionality in such a way that the requirements are fulfilled.

Unfortunately, to the best of our knowledge, despite the proliferation of works in this research field, none of them addresses the problem of providing an automated and efficient Co-Design methodology, in that they usually require several user interactions with considerable timing overhead. In this paper we propose our vision of a novel multi-staged approach that combines the *design for verifiability* concept [6] with *performance model-driven* techniques suitable to reduce the complexity of the Design Space Exploration (DSE) process. We intend to automatically and efficiently derive an optimal hardware and software partitioning for the system's capabilities by improving an existent DSE approach [14]

The main idea underling our proposal is that, by exploiting some properties of the embedded system domain (such as the real-time scheduling and the few number of user interactions), we aim to build a novel modeling language whose formal semantic is equipped only with those constructs leading to a compact state space representation. Then, by leveraging on it, we intend to realize a model-driven verification/transformation chain where, at each step, the initial high-level system behavioral specification is refined in a more concrete version that is verified against user-defined properties. The goal of the step-wise verification is to identify the HW/SW configurations that do not meet functional and non-functional requirements, so as to reduce the size of the feasible solutions set that is evaluated during the design space exploration phase of our approach. By doing so, we aim to reduce both the user intervention and the timing overhead in the design and implementation activities of complex and heterogeneous multi-processor embedded systems. Fig 1 depicts the conceptual structure of our vision.

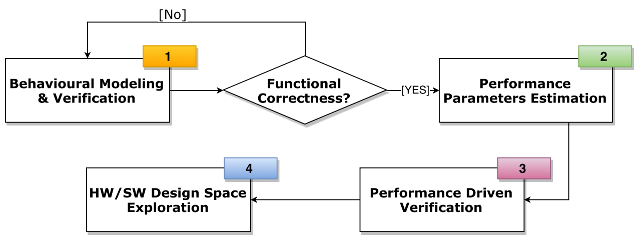
[No]
1
Behavioural Modeling & Verification
Functional Correctness?
[YES]
2
Performance Parameters Estimation
3
Performance Driven Verification
4
HW/SW Design Space Exploration

Figure 1: Overview of the reference Co-Design approach

In the *Behavioral Modeling & Verification* stage the high-level behavior representation of the system is modeled. As described before, the syntax and semantic of the envisaged modeling language, i.e. HW/SW Modeling Language (HML), are based on the "design for verifiability" concept that enables an efficient functional verification. Goal of this stage is to identify and eliminate all the system configurations for which functional requirements are certainly not fulfilled, hence useless to be considered in the following stages of the process. In the *Performance Parameters Estimation* stage the initial behavioral specification is simulated in order to estimate some performance attributes. Then, the outcome of this stage is used as the input for the *Performance Driven Verification* one in which a performance evaluation of the system is conducted, aimed at generating a set of constraints that identify the architectural elements that have not to be considered in the last step. Finally in the *HW/SW Design Space Exploration* stage, by exploiting and improving the existed DSE approach defined in [14], an optimal HW/SW partitioning of the system is identified on the basis of by the architectural constraints derived in the previous steps.

2. THE VISION OF PERFORMANCE-DRIVEN CO-DESIGN

In the following we provide a detailed description of the blocks introduced in Figure 1 that have been expanded as in Figure 2. We use square boxes for denoting artifacts (i.e., models and constraints) and the elliptic boxes for actions (i.e., transformations and verification).

Behavioral Modeling and Verification

The starting point of this stage is a *Behavioral Model* representing the behavioral view of the system written in HML. It can be seen as an extension of the well-known Communication Sequential Processes (CSP) [8] formalism equipped with a novel semantic suitable to associate a compact state space representation to each HML model, hence enabling an efficient functional verification. Our vision, at this stage, is to exploit some intrinsic properties of the embedded systems domains (e.g., real time scheduler or synchronous communication channels) for reducing the non-determinism and the parallelism of the model. Moreover, it is worth to notice how a reduced complexity allows to move the assessment of some critical functional properties, such as input-output relation, from the testing to the verification stage. Once the HML model is defined, it can be used as an input for a Model to Model (M2M) transformation whose outcome is a new model (i.e., Verifiable Model) on which the user-defined functional requirements can be efficiently verified using off the shelf model checkers ([7, 9, 13]).

Performance Parameters Estimation

Goal of this stage is to estimate the timing information for the system's behaviors so as to produce a quantitative annotated version of the original behavioral specification. For doing so, the behavioral model is given as input to a Model to Text (M2T) transformation aimed to generate a SystemC [2] platform-independent executable representation. Our vision here is to make an early lightweight estimation of the performance of each behavioral block by focusing only on their execution time. To this goal, we map each SystemC subsystem on elements of a Technology Library (TL) representing basic hardware components, whose assembly can model different execution platforms could be composed. In particular, it contains all the available processors, memories and interconnection links that can be used for generating the HW platform on which the system will be executed. As a result, the simulation of this technology-dependent representation enables to build a performance-profile for each of the HML process, later used to drive the Design Space Exploration aimed at timing requirements fulfillment.

Our vision here relies on the assumption that the timing estimation produced in this stage can be considered as a kind of *best case scenario*, since the estimated performance will be worse with an increased simulation accuracy (e.g., scheduling and communication overhead, interaction with sensors and peripherals, etc.). This is the key factor that makes the simulation results as necessary conditions with respect to the timing requirements fulfillment, thus allowing us to eliminate all basic hardware components that certainly lead to unfeasible system configurations.

Performance Driven Verification

The objective of this stage is to explore the possible system HW/SW partitions and mappings, in order to identify those that surely do not fulfill timing requirements. For doing that, both the Quantitative Annotated Model and Technology Library are given as inputs to a M2M transformation aimed to produce a *Performance Super Model* (PSM) that,

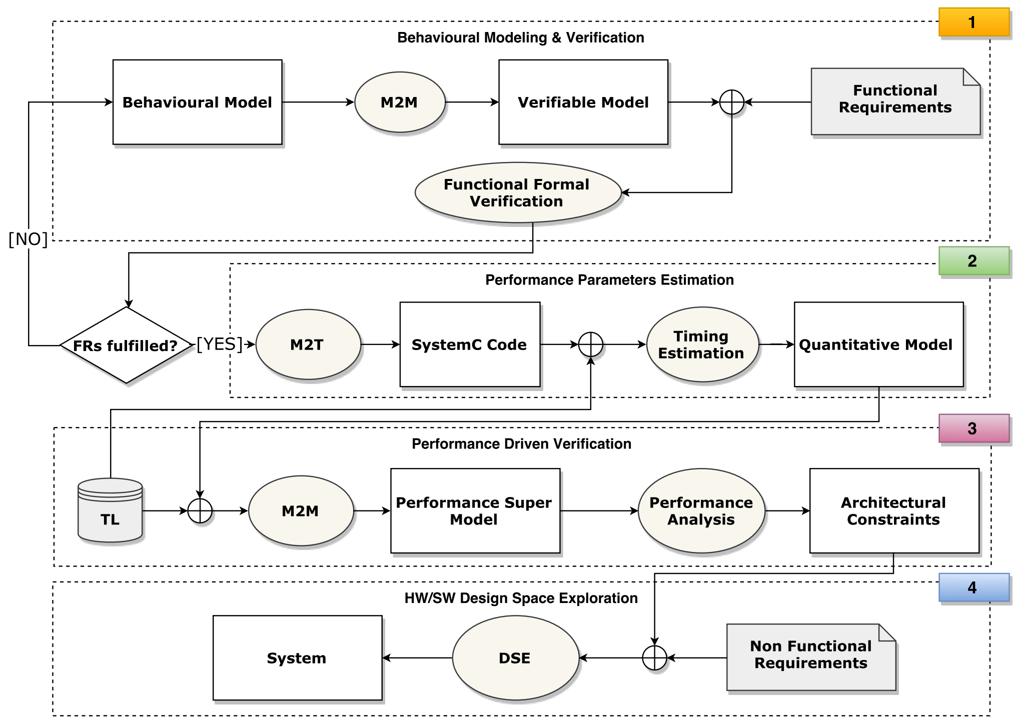
1
Behavioural Modeling & Verification
Behavioural Model
M2M
Verifiable Model
Functional Requirements
Functional Formal Verification
[NO]
2
Performance Parameters Estimation
FRs fulfilled?
[YES]
M2T
SystemC Code
Timing Estimation
Quantitative Model
3
Performance Driven Verification
TL
M2M
Performance Super Model
Performance Analysis
Architectural Constraints
4
HW/SW Design Space Exploration
System
DSE
Non Functional Requirements

Figure 2: The reference Co-Design flow

similarly to what presented in [12], is suitable to encode all the system variability (i.e., software to hardware mapping) in one single model. Our vision here, analogously to what presented in [10], is to exploit the Satisfiability Modulo Theory (SMT) for encoding both PSM and the chosen performance model (e.g., Queueing Networks, Petri Nets, Markov Chains) in such a way that the exploration process can be conducted in a performance-driven fashion. Basically, the idea of the Performance Analysis phase is to ask an SMT solver for the architectural elements that do not belong to any correct system configuration (i.e., satisfying the performance requirements), that are thus worthless to be considered in the Design Space Exploration step. Then, the result of this analysis is encoded in a set of Architectural Constraints used as input for the last stage of the proposed approach.

HW/SW Design Space Exploration

The last stage of this vision is the *HW/SW Design Space Exploration*, in which we ntend to exploit the power of genetic algorithms for searching the optimal HW/SW mapping and partitioning driven by non-functional requirements. Our *DSE* approach builds on the idea presented in [14], which enables the optimization of the system with respect to a richer set of non-functional requirements than those used in the previous stage (e.g., production costs, load-balancing of the whole system, and communication overhead).

We want to emphasize that the key feature of our approach is to make a multi-staged filtering in order to reduce the solution space. Hence, we automatically generate an optimal set of solutions (see *System* in Figure 2), which the designer can refer to in order to find look her solution.

3. RELATED WORK

In the past years, a remarkable number of research projects focused on the system-level HW/SW Co-Design embedded systems following the ”Y” structure [11]. To the best of our knowledge, none of them fully addresses the problem of both ”automatically suggest an HW/SW partitioning” and ”map the partitioned entities into an automatically defined heterogeneous multi-processor architecture”. Beside this we provide the opportunity to perform an early formal functional verification. In this section we consider only those works that respect the ”Y” structure.

The Essyn project [15] requires the definition of three submodels. The first UML/MARTE sub-model is specifically used to design software applications; the second one is used to represent hardware platforms, and both are connected by a third model, namely Platform Specific Model, which defines the mapping among software applications and hardware platforms. With respect to it, our vision provides an higher level of abstraction of the system behaviour, and it intend to provide an automated generation of optimal mapping among functionality and basic hardware components.

Another related project is Contrex [5]. In order to allow both a system synthesis and an analysis, with respect to non-functional requirements (e.g., temperature), it provides an environment suite, based on Papyrus[1], that is augmented with a specific extension for DSE. Our methodology is quite different, because it aims to suggest, other than a HW/SW partitioning and mapping, an hardware platform by composing basic elements from a Technology Library.

Art2kitekt [16] is aimed at modeling and analyzing systems with respect to both hardware and software representation and, additionally, it provides set of integrity checks and DSE algorithms that help to perform offline real-time

[1]https://eclipse.org/papyrus/

and schedulability analysis. In our approach the designer would not be required to provide as input an explicit mapping between hardware and software.

Finally, it is worth to cited Intel CoFluent [1]. It is a representative SystemC-based commercial product, as a promising system-level modeling and simulation environment. With respect to our methodology it requires, by exploiting the hardware configuration given as input, a manual mapping among hardware and behavioral models.

4. CONCLUSIONS

Nowadays embedded systems are increasingly powerful and complex, hence novel development paradigms are required. Unfortunately, despite the proliferation of research works in this area, no efficient and fully automated approaches have been proposed yet.

In order to face such challenge, in this paper we presented our vision for an efficient performance-driven HW/SW Co-Design methodology. It mitigates the complexity of the design space exploration process by relying on the design for verifiability concept and on model-driven techniques for removing performance issues. In particular, we provided the high level architecture of our approach for building an efficiently automated flow, with the potential of requiring few user interactions and introducing a small timing overhead.

As future work, our research agenda includes the following tasks: (i) choosing the right trade-off between the expressiveness of the envisaged modeling language and the efficiency of its semantic, we plan to infer it by analyzing several real-world embedded systems applications; (ii) identifying the suitable granularity (e.g., varying from the instruction level to process level) of the timing simulation for the estimation of performance parameters; (iii) choosing the most appropriate modeling formalism for encoding a Performance Super Model and Architectural Constraints, and the proper analysis technique; (iv) improving the used algorithm for the HW/SW Design Space Exploration stage.

The big challenge that appears in our vision is the integration among the four stages. In fact, we intend to use a multi-representation, which might bring to the incompatibility among the outcome of a stage and the input of the subsequent one.

Finally we plan to validate the proposed approach on HW/SW Co-Design case studies coming from real industrial applications.

5. REFERENCES

[1] Intel co-fluent. http://www.intel.com/content/www/us/en/cofluent/overview.html. Accessed: 2016-25-11.

[2] Systemc. http://www.accellera.org. Accessed: 2016-25-11.

[3] L. H. Crockett, R. A. Elliot, M. A. Enderwitz, and R. W. Stewart. *The Zynq Book: Embedded Processing with the Arm Cortex-A9 on the Xilinx Zynq-7000 All Programmable Soc*. Strathclyde Academic Media, 2014.

[4] P. Cumming. The ti omap^{TM}platform approach to soc. In *Winning the SOC Revolution*, pages 97–118. Springer, 2003.

[5] E. Ebeid, F. Fummi, and D. Quaglia. Model-driven design of network aspects of distributed embedded systems. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 34(4):603–614, 2015.

[6] J. F. Groote, T. W. D. M. Kouters, and A. Osaiweran. Specification guidelines to avoid the state space explosion problem. *Softw. Test., Verif. Reliab.*, 25(1):4–33, 2015.

[7] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. Prism: A tool for automatic verification of probabilistic systems. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 441–444. Springer, 2006.

[8] C. A. R. Hoare et al. *Communicating sequential processes*, volume 178. Prentice-hall Englewood Cliffs, 1985.

[9] G. J. Holzmann. *The SPIN model checker: Primer and reference manual*, volume 1003. Addison-Wesley Reading, 2004.

[10] E. Incerto, M. Tribastone, and C. Trubiani. Symbolic performance adaptation. In *Proceedings of the 11th International Workshop on Software Engineering for Adaptive and Self-Managing Systems*, pages 140–150. ACM, 2016.

[11] K. Keutzer, S. Malik, A. Newton, J. Rabaey, and A. S. Vincentelli. System level design: Orthogonalization of concerns and platform-based designh. *IEEE Trans. Comput.-Aided Des. Integ. Circ. Syst.*, 19(12):1523–1543, 2000.

[12] M. Kowal, M. Tschaikowski, M. Tribastone, and I. Schaefer. Scaling size and parameter spaces in variability-aware software performance models (t). In *Automated Software Engineering (ASE), 2015 30th IEEE/ACM International Conference on*, pages 407–417. IEEE, 2015.

[13] K. G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer (STTT)*, 1(1):134–152, 1997.

[14] L. Pomante. Hw/sw co-design of dedicated heterogeneous parallel systems: an extended design space exploration approach. *IET Computers Digital Techniques*, 7(6):246–254, November 2013.

[15] H. Posadas, P. Peñil, A. Nicolás, and E. Villar. Automatic synthesis of communication and concurrency for exploring component-based system implementations considering uml channel semantics. *Journal of Systems Architecture*, 61(8):341–360, 2015.

[16] W. Weber, A. Hoess, J. van Deventer, F. Oppenheimer, R. Ernst, A. Kostrzewa, P. Dorè, T. Goubier, H. Isakovic, N. Druml, et al. The emc2 project on embedded microcontrollers technical progress after two years. In *Eur. Conf. on Digit. Syst. Des. (DSD)*, pages 524–531, 2016.

[17] W. Wolf. A decade of hardware/software codesign. *Computer*, 36(4):38–43, 2003.