HEPSYCODE-RT: a Real-Time Extension for an ESL HW/SW Co-Design Methodology

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

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

Daniele Ciambrone
Center of Excellence DEWS
Via Vetoio, 1
L'Aquila, Italy
daniele.ciambrone@student.univaq.it

Vincenzo Stoico
Center of Excellence DEWS
Via Vetoio, 1
L'Aquila, Italy
vincenzo.stoico@student.univaq.it

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

ABSTRACT

This work focuses on the definition of a methodology for handling embedded real-time applications, starting from an existing HW/SW co-design methodology able to support the design of dedicated heterogeneous parallel systems. The state-of-the-art related to similar tools and methodologies is presented and the reference framework with the proposed extension to the real-time world is introduced. A case study is then described to show a design space exploration able to consider such an extension.

CCS CONCEPTS

• **Hardware → Software tools for EDA**; • **Hardware → Modeling and parameter extraction**; *Timing Simulation*; *DSE;*

KEYWORDS

HW/SW Co-Design, Heterogeneous Parallel Systems, DSE, real-time systems

1 INTRODUCTION

During the last years, the spread and importance of embedded systems are increasing but it is still not yet possible to completely engineer their system-level design flow. Designers commonly adopt one or more system-level models (e.g. block diagrams, UML, *SystemC*, etc.) to have a complete problem view, to perform a check on HW/SW resources allocation and to validate the design by simulating the system behavior. In this scenario, SW tools to support designers to reduce costs and overall complexity of systems development are even more of fundamental importance. Unfortunately, there are no fully engineered general methodologies defined for this purpose and often the best option is still to refer to experienced designer indications to take advantage of empirical criteria and qualitative assessments. For example, systems based on heterogeneous multi-processor architectures (*Heterogeneous Multi-Processor Systems*, HMPS) have been recently exploited for a wide range of application domains, especially in the *System-on-Chip* (SoC) form factor (e.g. [2]). In particular, such architectures are often used to implement *Dedicated Systems* (DS), i.e. digital electronic systems with an application-specific HW/SW architecture designed to satisfy a priori known application with F/NF requirements. In such a case, they are called *Dedicated Heterogeneous Multi-Processor Systems* (D-HMPS). D-HMPS are so complex that the adopted *HW/SW Co-Design Methodology* plays a major role in determining the success of a product. The situation is even worse if the considered system is a hard/soft real-time one. In such a case, time constraints are normally defined considering the worst possible case (hard) or an average situation (soft).

In such a context, this work focuses on the definition of a HW/SW co-design methodology and the development of a related prototypal tool to improve the design time of embedded real-time applications. Specifically, the whole framework drives the designer from an *Electronic System-Level* (ESL) behavioral model, with related NF requirements, including real-time ones, to the final HW/SW implementation, considering specific HW technologies, scheduling policies and *Inter-Process Communication* (IPC) mechanisms. The remainder of the paper is organized as follows: *Section II* provides an overview of HW/SW co-design tools related to embedded real-time computing systems. *Section III* describes the reference HW/SW Co-Design methodology, whereas *Section IV* discusses the extension to adapt it to the real-time world. *Section V* presents a case study that shows a design space exploration able to consider such an extension. Finally, *Section VI* reports some conclusive considerations and presents future works.

2 HW/SW CO-DESIGN OF REAL-TIME EMBEDDED SYSTEMS

A remarkable number of research works have focused on the system-level HW/SW co-design of D-HMPS [3]. In such a context, the most critical steps are always related to the *System Specification* and the *Design Space Exploration* (DSE) activities

[4]. The main differences between the approaches are related to the amount of information and actions explicitly requested to the designer and influenced by his experience. In particular, many approaches (especially those based on the *Y-Chart* principle [5]) explicitly require as an input the HW architecture to be considered for mapping purposes. Other works [6] aims to the problem of designing embedded real-time systems starting from the input/output constraints on the final implementation. Offline schedulability and feasibility analysis involve different research works [7][8], with respect to the correct algorithms that can guarantee optimality depending on the load parameters. To analyze the behavior of a system, many tools have been developed to evaluate/estimate timing parameters, to validate scheduling and to perform simulations. In such a domain, the work presented in [9] starts from three sub-models, considering a model for SW application (*Platform Independent Model*) on one side and a platform (*Platform Description Model*) on the other side, and both models are connected by a *Platform Specific Model* that defines the mapping of SW into HW. By exploiting a specific extension for DSE and performance evaluation [10], in order to consider non-functional properties such as real-time, power, temperature, reliability constraints and so on, the tool offers different simulation and estimation outputs that drive the designer from the system-level model to the final implementation.

With respect to works that heavily relies on *Model of Computations* (*MoC*) theory, *ForSyDe* (*Formal System Design*) [11] is a methodology for modeling and design heterogeneous embedded and cyber-physical systems. The starting application is modeled by a network of processes interconnected by signals. Then, the model is refined by different design transformations into a target implementation language.

An interesting academic tool is *SynDEx* [12], a system level EDA tool based on the *Algorithm-Architecture Adequation* (*AAA*) methodology intended to find implementation solution, under real-time constraints, for embedded applications onto multi-component HW/SW architectures.

Finally, to have a look also to a SystemC-based commercial product, it is worth noting to cite *Intel CoFluent* [13] as a promising system-level modeling and simulation environment. Other than the model of the system behavior, it explicitly requires a manual modeling of both the hardware architecture and the mapping.

So, at the best of our knowledge, there are very few system-level HW/SW co-design methodologies that try to fully address the problem of both "automatically suggest an HW/SW partitioning of the system specification" and "map the partitioned entities onto an automatically defined heterogeneous multi-processor architecture" while considering also real-time constraints.

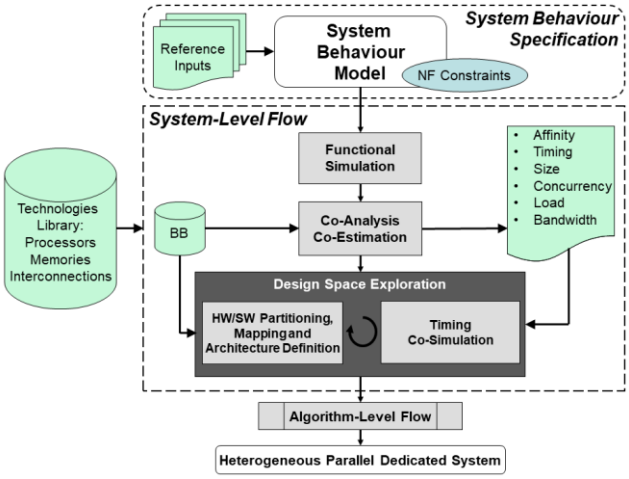

Figure 1: Reference HW/SW Co-Design Flow.

3 HW/SW CO-DESIGN FRAMEWORK

In the context of embedded real-time systems design, this work starts from a specific framework (called *HEPSYCODE*: *HW/SW Co-Design of Heterogeneous Parallel Dedicated Systems*) [14], based on an existing *System-Level HW/SW Co-Design Methodology* [15][19][21], and introduces the possibility to specify real-time requirements in the set of non-functional ones (the new framework is so called *HEPSYCODE-RT*). The main items composing such a methodology and its extension are discussed in the next paragraphs, while the reference ESL HW/SW Co-Design Flow is shown in Fig. 1.

3.1 Modeling Language

The system behavior modeling language introduced in *HEPSYCODE-RT*, named *HML* (*HEPSY Modeling Language*) [17], is based on the *Communicating Sequential Processes* (*CSP*) *MoC* [16]. It allows modeling the behavior of the system as a network of processes communicating through unidirectional synchronous channels. By means of HML it is possible to specify the *System Behavior Model* (*SBM*), an executable model of the system behavior, a set of *Non–Functional constraints* (*NFC*) and a set of *Reference Inputs* (*RI*) to be used for simulation-based activities. It is worth noting that another *HEPSYCODE* extension able to exploit more formal approaches is currently under development [17].

In particular *SBM = {PS, CH}* is a CSP-based executable model of the system behavior that explicitly defines communication among processes (*PS*) using unidirectional point-to-point blocking channels (*CH*) for data exchange. $PS = \{ps_1, ps_2, .. , ps_n\}$ is a set of concurrent processes that communicate with each others exclusively by means of channels and use only local variables. Each process is described by means of a sequence of statements by using a suitable modeling language. Each process can have a priority *p*: 1 (lower) to 100 (higher) imposed by the designer. The concept of statement has to be fixed once selected a

proper specification/modeling language. Languages suitable to describe CSP are *SystemC* (chosen for this work), *OCCAM*, *Handel-C*, *ADA* and so on. More abstract languages are *UML*, *SysML*, *Simulink* and so on. $CH = \{ch_1, ch_2, .. , ch_n\}$ is a set of channels where each channel is characterized by source and destination processes, and some details (i.e. *size*, *type*) about transferred data. Each channel can have also a priority *p*: 1 (lower) to 100 (higher) imposed by the designer.

RI: $\{(i_1, o_1), \ldots, (i_n, o_n)\}$ is a set of inputs (possibly timed), representative as much as possible of typical operating conditions of the system, and related expected outputs to be used for analysis and simulation-based validation.

The *Non–Functional Constraints* (*NFC*) are composed of *Timing Constraints (TC)*, *Architectural Constraints (AC)* and *Scheduling Directives*. Two different *TC* can be considered by the designer: *Time-to-Completion* (*TTC*), unique and related to the whole SBM, is the time available to complete the SBM execution from the first input trigger to the complete output generation; *Time-to-Reaction* (*TTR*) is a set of real-time constraints related to the time available for the execution of leaf CSP processes (i.e. the time available to execute the statements inside an input/output pair that delimits the CPS process main body, see Fig. 2). Different leaf processes can have different associated TTR. This real-time constraints are not strictly related to classical RT requirements, but impose a timing bound to the execution of some specific processes. Both TTC and TTR constraints shall be satisfied by each element of RI.

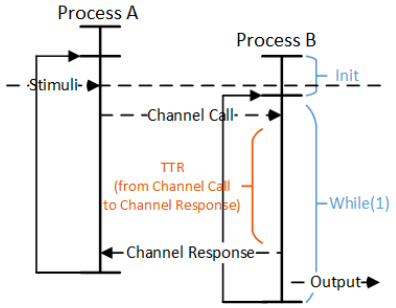

Figure 2: Time-To-Reaction Constrain.

The Architectural Constraints (AC) are related to the *Target Form Factor* (TFF) as *On-chip* (ASIC, FPGA) or *On-Board* (PCB) and to the *Target Template Architecture* (TTA) as type of available *Basic Blocks* (BB), min/max number of possible BB instances, min/max number of available *Interconnections* instances and/or total available area (or an equivalent metric). Finally, the *Scheduling Directives* (SD) specify the available scheduling policies. At the moment they are *First-Come First-Served* (FCFS) and *Fixed Priority* (FP) preemptive scheduling.

3.2 Technologies Library and Basic Blocks

The target HW architectures is composed of different basic HW components. These components are collected into a *Technologies Library* (*TL*). *TL* can be considered as a generic "database" that provides the characterization of the available technologies. In particular $TL = \{PU, MU, EIL\}$, where $PU = \{pu_1, pu_2, .. , pu_p\}$ is a set of *Processing Units*, $MU = \{mu_1, mu_2, .. , mu_m\}$ is a set of *Memory Units* and $EIL = \{il_1, il_2, .. , il_c\}$ is a set of *External Interconnection Links*. However, the detailed characterizations are dependent on *TFF*. The main differences are related to the different attributes (or different meaning of the same attribute) needed to characterize processing units, local memories, and interconnections. This work considers only *TL* for *PCB* where each PU that executes SW shall be a separate discrete *Commercial Off-The-Shelf* (*COTS*) *Integrated Circuit* (*IC*) mounted on a board.

PU elements are divided into two main groups: the ones that perform processing by means of the execution of some *Instruction Set Architecture* (ISA), called *SW-PU*, and the ones that perform processing without relying on an *ISA*, called *HW-PU*. Each pu_i in *PU* for *PCB* is characterized by a *Name*, a *Processor Type*, *Capacity* (SW-PU: max allowed *load*; HW-PU: available resources as number of equivalent-gates, *LUT*, Cell, etc.), *ISA* (only for SW-PU), *Frequency*, *Context Switch Overhead* (only for *SW-PU*), a *statement-level performance metric* (like CC4CS [18] or equivalent ones), and a *unit cost* (€). With respect to *Processor Type*, *PU* elements are further classified in three classes [1]: *General-Purpose Processors* (SW-PU: GPP); *Application-Specific Processors* (SW-PU: ASP) targeted to particular application domains (e.g. *Digital Signal Processors*, DSP); *Single-Purpose Processor* (HW-PU: SPP; realized by means of ASIC/FPGA.

MU elements are divided in two main classes: *Volatile Memory Units* (*VLMU*) and *Non-Volatile Memory Units* (*NVLMU*), with a main parameter related to capacity (i.e. bytes).

EIL elements are characterized by some parameters related to *bandwidth*, *number of connectable items* and *concurrency properties*.

The designer will use such components to build a set of *Basic Blocks* (BB). So, $BB = \{bb_1, bb_2, .. , bb_b\}$ is the set of BB available during DSE step to automatically define the HW architecture. A generic *BB* is composed of a set of *PU*, a set of *MU* and a *Communication Unit* (CU). *CU* represents the set of *EIL* that can be managed by a BB. BB internal architecture is dependent on *TFF* and *TTA*. In particular, each BB element can be generally composed of 1 or more *PU* elements, some *MU* elements and 1 *CU* element. *BB* elements are the ones effectively taken as input by the system-level flow for analysis, estimations and DSE steps. So, the target HW architecture can be seen as a set of *BB* elements interconnected by means of one or more *EIL* elements. The type of available BB are automatically defined by the selected *TTA*.

This work currently focuses only on: *Homogeneous Multi-Processor System with Distributed Memory* where each *BB* element is composed of only 1 *PU* element (homogenous among *BB* elements), some local *MU* elements and 1 *CU* element; *Heterogeneous Multi-Processor System with Distributed Memory* where each *BB* element is composed of only 1 *PU* element (heterogeneous among BB elements), some local *MU* elements and 1 *CU* element.

3.3 ESL HW/SW Co-Design Flow

The first step of the adopted co-design flow is the *Functional Simulation* where *SBM* is simulated to check its correctness with respect to *RI*. Then, the next step aims at extracting as much as possible information about the system by analyzing the *SBM* while considering the available *BB*. This step is supported by *Co-Analysis* and *Co-Estimation* activities to evaluate/estimate several metrics related to the *BB* involved in the design flow.

Co-Analysis performs evaluation of two metrics. The first one is called *Affinity* [19]. The *Affinity* $A = \{[a_1, a_2, .. a_n] \mid a_i = [A(GPP_i), A(DSP_i), A(SPP_i)]\}$ of a process ps_i is a triplet of values in the interval [0,1] that provides a quantification of the matching between the structural and functional features of the functionality implemented by a process and the architectural features of each of *GPP*, *DSP*, and *SPP*. The second metric evaluated during *Co-analysis* is related to *Concurrency*. The *concurrency* $CN(PS, CH) = [CN_{PS}, CN_{CH}]$ is expressed by the set of processes *PS* and channels *CH* pairs that could be potentially concurrently "working", where $CN_{PS} = \{ [ps_i, ps_j] : ps_i \wedge ps_j$ *could be potentially executed concurrently}* and $CN_{CH} = \{ [ch_i, ch_j] : ch_i \wedge ch_j$ *could potentially transfer data concurrently}*. *CN* is evaluated by means of a functional simulation with respect to *RI*.

Co-estimation performs two kinds of estimations: *Static Estimations* of *Timing* and *Size, and Dynamic Estimations* of *Load* and *Bandwidth*. The *Timing* metric is the number of clock cycles needed to execute each statement *j* of each process ps_i by means of each processor *k* in the available BB, with *k=1..n*. The goal is to estimate how many clock cycles are needed by a specific BB to execute the implementation of a specific statement (e.g. [18] and [20] presents two possible approaches).

Size is a set of estimations for each statement of each process with respect to each available processor. It is related to bytes or area/resources metrics depending on SW or HW implementations. *L* is the *Load* (i.e. the *processor utilization percentage*) that each process would impose to each *not-SPP* processor to satisfy imposed TTC/TTR timing constraints (see *Section IV*). Finally the *Bandwidth* (B) is the number of bits sent/received over each channel (i.e. bits exchanged by communicating processes pairs in PS) during an interval of time equal to TTC.

After this steps, the reference co-design flow reaches the DSE step (as shown in Figure 3). 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 timing constraints satisfaction. 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*).

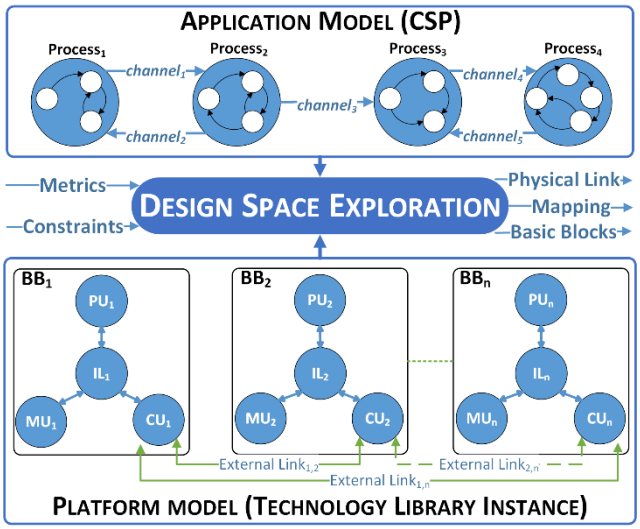

Figure 3: Design Space Exploration Framework.

4 HEPSYCODE-RT: PROPOSED EXTENSION

With respect to NF requirements, this work provides an extension that allow the methodology to better consider architectural and timing constraints. Related to the *SBM* model, it is now possible to identify two classes of CSP processes: *classical CSP process* and *real-time CSP processes*. In the current version, the last ones shall be leaf processes and their body (i.e. a never-ending loop) shall start with a channel read and end with a channel write towards the same process. To such input/output pair will be referred the *TTR* constraint. Moreover, in such a context, a *CSP to Task Model* transformation has been defined to allow considering classical real-time world notations. Such a transformation involves concepts related to both processes and channels.

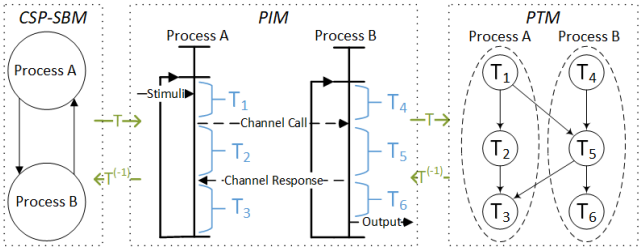

Figure 4: Time-To-Reaction Constrain.

The general transformation is shown in Figure 4. In this example the *CSP SBM* model is first expanded in a *Process Interaction Model* (*PIM*), where the processes A and B are split into different pieces of code, delimited by channel calls. The final transformation starts from the *PIM* model and associates the single pieces of code to specific tasks in the classical task representation models (i.e. *Process-Task Model*, *PTM*). At this time, the designer should write a SBM avoiding cycles to match the classical real-time DAG representation of tasks.

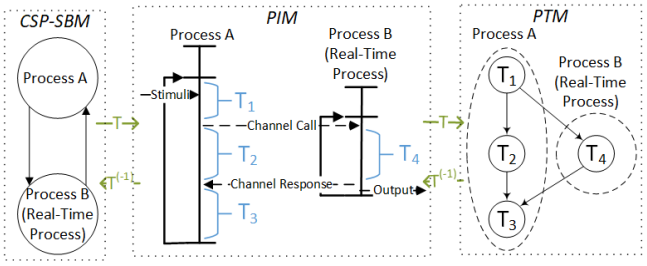

Figure 5: Time-To-Reaction Constrain.

With respect to the real-time CSP processes, the actual transformation is the one shown in Figure 5. With this specific kind of representation it is possible to consider concurrently timing constraints related to the whole SBM (TTC) and real-time constraints related to the reaction of specific processes (TTR) while considering periodic leaf processes as periodic ones.

With this assumption, it is possible to adapt the *Load Estimation* step to consider such real-time constraints. In particular, the load can be defined in two different ways.

The *Load* L_i that each **non real-time process** ps_i would impose to each *non-SPP* processor *s* to satisfy TTC. L_i is estimated by allocating all the *n* processes to a single-instance of each software processor *s* ($pu_j \subseteq \{[pu_1, .. , pu_s]\}$ *with* $s \leq$ *Total Number of* pu_j) and performing some simulations. Three parameters have to be computed: FRT_j *(Free Running Time), i.e.* the total application simulation time on processor pu_j; t_i, the simulated time for each process ps_i in a loop on processor pu_j; *N*, the number of simulation loops. Starting from this estimated parameters, the *Free Running Load* FRL_i is calculated by the equation:

$$FRL_i = \frac{(t_i * N)}{FRT_j} \quad (1)$$

where FRT_j/N is the average period of each processes on processor pu_j. By imposing that the execution time shall be equal to *TTC*, it is possible to evaluate the Load L_i that processes ps_i would impose to the *SW* processor to satisfy *TTC* itself. In fact, setting FRT_j equal to *TTC*, for each process/processor pair, such as:

$$\text{TTC} = x_j * FRT_j \qquad with\ 0 \leq x_j \leq 1 \quad (2)$$

The value of estimated Load L_i that the system imposes to processor pu_j to satisfy *TTC* is:

$$\text{L}_i = \frac{(t_i * N)}{TTC} = \frac{(t_i * N)}{FRT_j} * \frac{FRT_j}{TTC} = \frac{FRL_j}{x_j} \quad (3)$$

The Load L_i that each **real-time process** ps_i would impose to each *s* software processor to satisfy input real-time constrain TTR_i is directly set equal to:

$$L_i = \frac{t_i}{TTR_i} \quad (4)$$

TTR_i is the real-time constraint related to the process ps_i. In this way it is possible to consider two different situations: *Hard real-time process*, if $t_i < TTR_i$, the constraints are fulfilled and it is possible to consider the value L_i as an input to the DSE step; *Soft real-time process*, if $t_i < (TTR_i + \delta(t))$, then constraints could be considered as soft real-time ones.

Then, thanks to all the estimated TTC/TTR loads, it is possible to perform DSE step in order to fulfill also RT constraints. Moreover, an additional architectural constraint deriving from TTR is that non-SPP processors executing real-time processes have to adopt a scheduling policy suitable for real-time scheduling (e.g. *fixed-priority preemptive scheduling*). Finally, the effect of such scheduling policy shall be considered during the timing co-simulations performed to validate the proposed solutions.

5 CASE STUDY

This section presents a simple case study used to show the effects of the proposed real-time extension to HEPSYCODE.

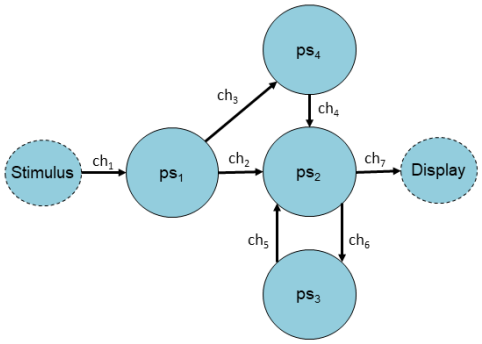

Figure 6: CSP MoC Example

The reference SBM is shown in Figure 6, where the processes $PS = \{ps_1, .. , ps_4\}$, with priority of $\{ps_1, ps_2, ps_4\}$ equal to each other and priority of ps_3 higher than the others, exchange data using the channels $CH = \{ch_1, .. , ch_7\}$. In this scenario there are three non real-time processes $\{ps_1, ps_2, ps_4\}$ and one process $\{ps_3\}$ with real-time constraint equal to TTR_3. The whole SBM is also subject to a TTC. So, for a given processor pu_j, the load parameters for the four processes are: $L_{1,j}=t_{1,j}/([x_j*FTR_j]/N)$, $L_{2,j}=t_{2,j}/([x_j*FTR_j]/N)$, $L_{4,j}= t_{4,j}/([x_j*FTR_j] /N)$, $L_{3,j} = t_{3,j}/TTR_3$. After the *Co-analysis* and *Co-estimation* steps, by considering such loads, the DSE step should be able to suggest a partition/mapping item able to fulfill both TTC and TTR_3 constraints. Several DSE have been performed considering different TTC/TTR pairs. In the considered use case the available BBs are: bb_1 (SW-PU): 20 MHz 8-bit 8051 CISC core with 128 byte of Internal RAM, 64K of internal ROM, without cache and external memory (cost 10); bb_2 (SW-PU): 150 MHz 32-bit LEON3 soft-processor with 2*4 KiB L1 caches, RAM size of 4096 KiB and a ROM of 2048 KiB (cost 50); bb_3 (HW-PU): 300 MHz Altera Stratix V (cost 300).

For each BB is allowed maximum 1 instance and they are supposed to communicate by means of a shared bus. Moreover, each SW-PU uses a *Fixed Priority* preemptive scheduling algorithm. Results shown in Table 1 figure out as the *DSE* step with real-time extension is able to satisfy *TTC/TTR* constraints, at

least with respect to timing simulations. In particular, by setting *TTR* and decreasing *TTC*, *DSE* suggests solutions that fulfil the timing requirements most of the time (two not satisfactory suggestions are underlined in Table 1). Decreasing the *TTR*, the *DSE* suggests to allocate the real-time process on pu_j that fulfil the constraints. It is worth noting that, if the *TTR* is very strict, the only valid mapping involve the use of a more expensive FPGA.

Table 1: Results from the DSE on the Use Case Example

Allocation	Simulated Time (ms)	ps_3 (ms)	TTC (ms)	TTR (ms)
All on bb_1	794,88	8,10	*600*	10
All on bb_2	650,66	5,54	*600*	10
ps_1, ps_2, ps_3 on bb_1 ps_4 on bb_2,	590,80	8,10	*600*	10
ps_3 on bb_1 ps_1, ps_4 on bb_2 ps_2 on bb_3	264,89	8,10	*400*	10
ps_1, ps_3 on bb_1 ps_4 on bb_2 ps_2 on bb_3	298,88	8,10	*300*	10
ps_4 on bb_1 ps_1, ps_2, ps_3 on bb_3	201,48	0,009	*200*	10
ps_3 on bb_1 ps_4 on bb_2 ps_1, ps_2 on bb_3	145,67	8,10	*200*	10
ps_3 on bb_1 ps_1, ps_2, ps_4 on bb_3	81,04	8,10	*100*	10
ps_1, ps_2 on bb_1 ps_3, ps_4 on bb_2,	562,85	5,54	*600*	7
ps_1, pu_4 on bb_1 ps_2, ps_3 on bb_2	462,58	5,54	*400*	7
ps_1 on bb_1 ps_3, ps_4 on bb_2 ps_2 on bb_3	220,80	5,54	*400*	7
ps_1, on bb_1 ps_3 on bb_2 ps_2, ps_4 on bb_3	206,99	5,54	*300*	7
ps_3 on bb_2 ps_1, ps_2, ps_4 on bb_3	55,55	5,55	*200*	7
ps_1, ps_4 on bb_1 ps_2 on bb_2 ps_3 on bb_3	428,87	0,009	*600*	4
ps_4 on bb_1 ps_1, ps_2 on bb_2 ps_3 on bb_3	337,56	0,009	*400*	4
ps_4 on bb_1 ps_1 on bb_2 ps_2, ps_3 on bb_3	214,80	0,009	*300*	4
ps_4 on bb_2 ps_1, ps_2, ps_3 on bb_3	137,62	0,009	*200*	4
All on bb_3	0,22	0,009	*100*	4

4 CONCLUSIONS

This work has proposed an extended Electronic Design Automation (EDA) methodology (and related tools) in the ESL domain supporting the development of Real-time Embedded Systems. The final result is a methodology able to support real-time systems developments by suggesting both the platform and mapping solutions for the specific application. Future works will involve the introduction of other parameters associated to PU such as Power (peak power [W] or other metrics) and Energy. Others analysis, use cases and tests will be done in future, but starting from this preliminary results it is easy to note that the DSE step with load estimation and real-time extension seem to be quite effective with respect to execution times estimated by simulation. Validation on the final HW/SW implementation must be done in future to reduce errors at design time.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Vahid, F. and Givargis, T. Embedded System Design: A Unified Hardware/Software Introduction. John Wiley & Sons, NY, USA, 2001.
[2] Xilinx Zynq7000, http://www.xilinx.com.
[3] Jia, Z. J., Bautista, T., Núñez, A. Pimentel, A. D. and Thompson, M. A system-level infrastructure for multidimensional MP-SoC design space co-exploration. In *ACM Trans. Embedd. Comput.* Syst. 13, 1s, Article 27 (December 2013), 26 pages, 2013.
[4] Teich, J. Hardware/Software Codesign: The Past, the Present, and Predicting the Future. Proceedings of the IEEE, vol. 100, no. Special Centennial Issue, pp. 1411-1430, 2012.
[5] Keutzer, K., Malik, S., Newton, A., Rabaey, J., and Sangiovanni-Vincentelli, A. System level design: Orthogonalization of concerns and platform-based design. IEEE Trans. Comput.-Aided Des. Integ. Circ. Syst. 19, 12, 1523–1543, 2000.
[6] Lee, E. A. and Seshia, S. A. Introduction to Embedded Systems, a Cyber-Physical Systems approach. In MIT Press, Second Edition, 2015.
[7] Real, J. and Crespo, A. Mode Change Protocols for Real-Time Systems: A Survey and a New Proposal. In Journal Real-Time Systems, 26, 2, 161-19, 2004.
[8] Buttazzo, G. Hard Real-Time Computing Systems - Predictable Scheduling Algorithms and Applications. In Springer, 3rd edition, 2011.
[9] Posadas, H., Penil, P., Nicolas, A. and Villar., E. Automatic synthesis of communication and concurrency for exploring component-based system implementations considering uml channel semantics. In Journal of Systems Architecture, 61, 8, 341-360, 2015.
[10] Contrex - Design of embedded mixed-criticality CONTRol systems under consideration of EXtra-functional properties. https://contrex.offis.de.
[11] Rosvall, K. and Sander. I. A constraint-based design space exploration framework for real-time applications on MPSoCs. In Design, Automation and Test in Europe, Dresden, Germany. 2014.
[12] Yu, H., Ma, Y., Gautier, T., Besnard, L., Talpin, J.P., Le Guernic, P. and Sorel., Y. Exploring system architectures in aadl via polychrony and syndex. In Frontiers of Computer Science Journal, 7, 5, 627-649, 2013.
[13] Intel cofluent. http://www.intel.com.
[14] Hepsycode: A System-Level Methodology for HW/SW Co-Design of Heterogeneous Parallel Dedicated Systems, www.hepsycode.com.
[15] Pomante, L. System-level design space exploration for dedicated heterogeneous multi-processor systems. In Conf. on Appl. Syst., 79-86, 2011.
[16] Hoare, C. A. R. Communicating sequential processes. In Springer, New York, NY, 413-443, 1978.
[17] Di Pompeo, D., Incerto, E., Muttillo, V., Pomante, L. and Valente, G. An Efficient Performance-Driven Approach for HW/SW Co-Design. In Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering (ICPE '17), ACM, New York, NY, USA, 323-326, 2017.
[18] Stoico, V., Muttillo, V., Valente, G., Pomante, L. and D'Antonio, F. CC4CS: A Unifying Statement-Level Performance Metric for HW/SW Technologies, In Eur. Conf. on Digit. Syst. (DSD), 2017.
[19] Pomante, L., Sciuto, D., Salice, F., Fornaciari, W. and Brandolese, C. Affinity-Driven System Design Exploration for Heterogeneous Multiprocessor SoC, In IEEE Trans. on Comp., 55, 5, 2006.
[20] Allara, A., Brandolese, C., Fornaciari, W., Salice, F. and Sciuto, D. System-level performance estimation strategy for sw and hw, In Proc. Int. Conf. on Comp., Austin, TX, 48-53, 1998.
[21] Pomante, L. HW/SW Co-Design of Dedicated Heterogeneous Parallel Systems: an Extended Design Space Exploration Approach. In IET Computers & Digital Technique, 2013.