Criticality-driven Design Space Exploration for Mixed-Criticality Heterogeneous Parallel Embedded Systems

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

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

ABSTRACT

Heterogeneous platforms are becoming widely diffused in the embedded system area, mainly because of the opportunities to increase application execution performance and, at the same time, to optimize other orthogonal metrics. In such a context, the introduction of mixed-criticality constraints, while considering heterogenous parallel architectures, creates new challenges to industrial and academic research. The main design issue is related to a Design Space Exploration (DSE) approach able to cope with mixed-criticality constraints that typically limits the number of feasible solutions. So, this work[1] focuses on DSE for embedded systems based on heterogeneous parallel architectures and subjected to mixed-criticality constraints. In particular, it presents a criticality-driven evolutionary approach integrated into a reference Electronic System Level HW/SW Co-Design flow to support the designer of mixed-criticality embedded systems.

CCS CONCEPTS

• **Hardware → Software tools for EDA**; • **Hardware → Modeling and parameter extraction**; *Design Space Exploration; Safety Assurance Level; Mixed-Criticality Systems;*

KEYWORDS

HW/SW Co-Design, Heterogeneous Parallel Systems, Design Space Exploration, Mixed-Criticality Systems

1 INTRODUCTION

In recent years, there has been a growing trend for switching from single-processor/core to (heterogeneous) multi-processor/core (i.e. parallel) platforms to execute embedded applications with different levels of criticality (i.e. *Mixed-Criticality Embedded Systems*, MCESs). In case of single-processor/core MCESs, it is crucial to ensure temporal isolation between tasks. In fact, such MCES can be viewed as systems with *Time Division Multiple Access* (TDMA), in which resources are assigned to each task in different time slots. In case of parallel MCESs, different embedded applications run in parallel on different processors competing for the access to shared resources, using different communication and synchronization mechanisms.

The main problem in the management of a MCES is to ensure that low criticality embedded applications do not interfere with high criticality ones. This type of systems can be found in many domains such as aerospace [1] or automotive industry [2]. The basis for integrating mixed-criticality applications on a single embedded platform are all the mechanisms that allow to create multiple partitions with a strict temporal and/or spatial isolation [4]. According to this approach, embedded applications with different criticality levels can be allocated on different partitions. In such a context, the purpose of this work is to present a *Design Space Exploration* (DSE) step, integrated into an *Electronic System-Level* (ESL) *HW/SW Co-Design* framework, to support the development of heterogeneous parallel MCES. 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 adopted design flow, while Section IV presents the main features of the proposed DSE approach. Then, Section V analyzes experimental results. Finally, Section VI closes the paper with some conclusions and future works description.

2 Design Space Exploration for Safety Critical Applications

In the last few years, a growing trend in the embedded systems domain is to run multiple embedded applications with different levels of criticality on a shared hardware platform. The criticality of an application is an indication of the required level of safety and security (i.e. assurance). After Vestal mixed-criticality paper [5], that first analyzed mixed-criticality system with focus on real-time performances, a series of research papers have been published [6], with no standard problem formulation with respect to the assurance level and the real-time task model [7][8][9]. In such a context, the most critical development steps are related to the *System Specification* and the *Design Space Exploration* activities [10] and the main differences among the various works in the literature are mainly related to the different amount of information and actions that explicitly rely on the designer experience. For example, AUTOFOCUS3 [11] proposes a model-

PARMA-DITAM '18, January 23, 2018, Manchester, United Kingdom

ACM ISBN 978-1-4503-6444-7/18/01…$15.00
https://doi.org/10.1145/3183767.3183782

based development process at different levels of abstraction 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. Another work, called CONTREP (CONTREX Eclipse plug-in, [12]) is a framework supporting UML/MARTE based modeling, analysis and design of mixed-criticality embedded systems. It is based on the CONTREX UML/MARTE modeling methodology [13] and considers safety constraints into the different design activities, integrating external tool like Multicube Explorer [14] for the DSE step. The work in [15] proposes a *combined-DSE* flow for design of time-critical systems. Starting from a *joint analytical and simulation-based* (JSA) DSE phase, while relying on constraint programming and worst-case estimations, it filters the design space and get a set of safe solutions. Finally, DeSyDe [16] provides a DSE tool for bare-metal applications, finding implementations for a set of tasks on a shared multi-processor platform starting from synchronous dataflow graphs (SDFGs), used to describe the application, and a predictable model for target platform.

So, at the best of our knowledge, there are few works that introduce mixed-criticality issues directly into a HW/SW co-design flow. In this context, this work proposes a DSE approach that is able to consider mixed-criticality issues into the development of heterogeneous parallel MCES. The main differences among the proposed approach and the previous works are related to the system behavior model, that is based on a CSP-like (*Communicating Sequential Processes*) *Model of Computation* (MoC) that allows to perform several analysis and estimations. Then, the proposed approach is able to suggest a criticality-aware HW/SW partitioning/mapping by means of an evolutionary approach.

3 Reference HW/SW Co-Design Framework

In the context of MCES, this work adopts a specific framework (*HEPSYCODE*: *HW/SW Co-Design of Heterogeneous Parallel Dedicated Systems*) [17], based on an existing *Electronic System-Level HW/SW Co-Design Methodology* [18], and introduces the possibility to specify mixed-criticality requirements. The framework is shown in Figure 1.

3.1 Modeling Language

The system behavior modeling language, named HML (*HEPSY Modeling Language*) [19], is based on the well-known *CSP MoC* [20]. 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* (NFCs) and a set of *Reference Inputs* (RI) to be used for simulation-based activities.

Definition 1. SBM = {PS, CH} is a CSP-based executable model of the system behavior that explicitly defines also a model of communication among processes (PS) using unidirectional point-to-point blocking channels (CH) for data exchange. In this work, the language used to model the SBM is *SystemC*.

Definition 2. PS = {ps_1, ps_2, .. , ps_n} is a set of *n* concurrent processes that communicate exclusively by means of channels and use only local variables. Each process has a criticality level $C(ps_i)$: 0 (lower) to *max* (higher) imposed by the designer depending on the safety standard related to the specific application domain [3].

Definition 3. CH = {ch_1, ch_2, .. , ch_c} is a set of *c* logical channels where each channel is characterized by source and destination processes, and some details (i.e. *size*, *type*) about transferred data.

Definition 4. RI: {(i_1 ,o_1), …, (i_i ,o_i)} is a set of inputs (possibly timed), representative as much as possible of typical system operating conditions of the system, and related expected outputs, to be used for analysis and simulation-based validation.

NFCs are composed of *Timing Constraints* (TCs), *Architectural Constraints* (ACs) and *Scheduling Directives* (SDs). In this work, the TC expressed by the designer is the *Time-to-Completion* (TTC) one. It is the time available to complete the SBM execution from the first input trigger to the complete output generation. ACs are related to the *Target Form Factor* (TFF) as *System On-chip* (SoC: ASIC or FPGA) or *System On-Board* (SoB: PCB) and to the *Target Template Architecture* (TTA) depending on the available *Basic Blocks* (BBs). Finally, SDs specify the available scheduling policies. At the moment, they are only *First-Came First-Served* and *Fixed Priority* preemption ones [21].

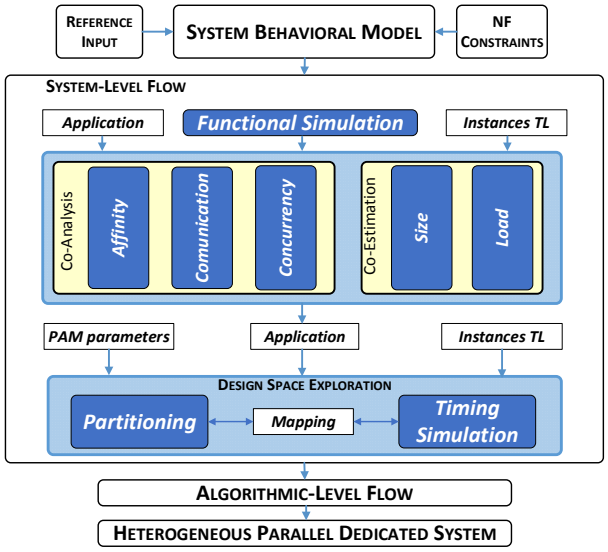

Figure 1: Reference HW/SW Co-Design Framework.

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.

Definition 5. TL = {PU, MU, EIL} is the Technology Library where PU = {pu_1, pu_2, .. , pu_p} is a set of *p Processing Units*, MU = {mu_1, mu_2, .. , mu_m} is a set of *m Memory Units* and IL = {il_1, il_2, .. , il_l} is a set of *Interconnection Links*.

However, the detailed characterizations are dependent on TFF. The main differences are related to the different attributes needed to characterize PU, MU, and IL. This work considers only TL for *SOB* where each PU that executes SW shall be a discrete *Commercial Off-The-Shelf* (COTS) *Integrated Circuit* (IC) mounted on a board.

PU elements are then 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 [22] or equivalent ones), and a unit cost (€). With respect to *Processor Type*, PU elements are further classified in three classes: *General-Purpose Processors* (SW-PU: GPP); *Application-Specific Processors* (SW-PU: ASP) targeted to tasks related to a particular application domain (e.g. *Digital Signal Processors,* DSP); *Single-Purpose Processor* (HW-PU: SPP; realized by means of ASIC or 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). IL elements are characterized by some parameters related to bandwidth, number of connectable items and concurrency properties.

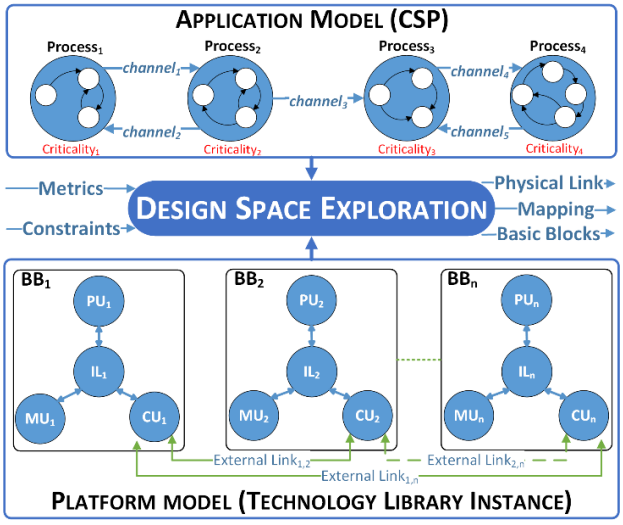

Figure 2: Design Space Exploration Approach.

The designer uses such components to build a set of *Basic Blocks* (BB) (*Instances TL*).

Definition 6. BB = {bb_1, bb_2, .. , bb_b} is the set of b Basic Blocks 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 IL 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. The target HW architecture can be seen as a set of BB elements interconnected by means of one or more IL elements. The type of available BB is automatically defined by the selected TTA. This work focuses on *Heterogeneous Multi-Processor System with Distributed Memory* where each BB element is composed of only 1 PU element (possibly heterogeneous among BB elements), some local MU elements and 1 CU element. It is worth noting that the reference methodology is able to consider other TTA, but the current prototypal tools fully support only the one listed above [23].

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 (*Application*) while considering the available BB (*Instances TL*). 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 *Affinity* [24], *Concurrency* and *Communication* metrics. *Co-estimation* performs a *Static Estimation* of *Size*, and a *Dynamic Estimation* of *Load*. After these steps, the reference co-design flow reaches the DSE step (as shown in Figure 1 and Figure 2). Starting from *Application*, *Instances TL* and *PAM parameters*, 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 (*Mapping*) items to actually check for timing constraints satisfaction.

4 Design Space Exploration Approach

The proposed DSE is based on a *Genetic Algorithm* (GA) used to optimize a multi-objective cost function that quantifies the quality of each individual of the GA population, as listed below:

$$CF_{i,j} = f_{i,j}(X_{1,j}, X_{2,j}, .., X_{k,j}) \quad \forall\, i = 1..I, j = 1..P \tag{1}$$

$$\begin{aligned} \mathrm{CF}_{i,j} = {} & \omega_{TDA} X_{TDA_j} + \omega_{EP} X_{EP_j} + \omega_{NTCC} X_{NTCC_j} \\ & + \omega_L X_{L_j} + \omega_C X_{C_j} + \omega_S X_{S_j} + \omega_{CRIT} X_{CRIT_j} \end{aligned} \tag{2}$$

$\mathrm{CF}_{i,j}$ is the cost function evaluated at iteration i for each individual j, I is the maximum number of iterations of the search algorithm and P is the size of population at iteration i. X_k represents the value of metric k for each individual, while ω_k is the weight associated to each metric. The rest of this paragraph defines the metrics and the methods used to evaluate them. For such a purpose, the instance of an individual IND_j is defined as a vector where the index represents processes and the value represents BB instances, for example:

$$IND_j = \langle a_0 | a_1 | \cdots | a_i | \cdots | a_n \rangle \text{ with i} = ps_i \in \mathrm{PS}, a_i \in \mathrm{BB} \tag{3}$$

The first metric considered is the ***Affinity Index***. The *Affinity* $A_i = \{[a_1, a_2, .. a_n] \mid a_i = [A(GPP_i), A(DSP_i), A(SPP_i)]\}$ of a process ps_i is a triplet of values in the interval [0,1] that provides a

quantification of the matching among the structural and functional features of the functionality implemented by a process and the architectural features of each one of the following processor types: *GPP, DSP, SPP*. Higher the *Affinity* value, more suitable the corresponding processor type. Starting from this definition, for each individual IND_j, it is possible to evaluate the *Total Degree of Affinity (TDA) Index* as:

$$X_{TDA_j} = 1 - \frac{\sum_{i=1}^{n} a_i}{n} \tag{4}$$

The second metric is related to ***Process Concurrency Index***. It is based on a *Concurrency Matrix*:

$$CON_{PS} = \begin{bmatrix} con_{1,1} & con_{1,2} & \cdots & con_{1,n} \\ con_{2,1} & con_{2,2} & \cdots & con_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ con_{n,1} & con_{n,2} & \cdots & con_{n,n} \end{bmatrix} \tag{5}$$

CON_{PS} provides information about how much processes pairs can be potentially concurrently "working", where $CON_{PS} = \{con_{i,z} \neq 0 : ps_i \wedge ps_z$ can be potentially executed concurrently$\}$. So, for each individual IND_j, it is possible to define the *Exploited Parallelism (EP)*:

$$X_{EP_j} = \frac{\sum_{i=1}^{n}\sum_{z=1}^{n} EICP_{i,z}}{maxEP} \tag{6}$$

$$EICP_{i,z} = \begin{cases} con_{i,z} & if\ ps_i \in pu_x \wedge ps_z \in pu_y \wedge pu_x \neq pu_y \\ 0 & otherwise \end{cases} \tag{7}$$

$$maxEP = \sum_{i=1}^{n}\sum_{z=1}^{n} con_{i,z} \tag{8}$$

EICP stands for *Exploited Inter Cluster Parallelism*, that indicates how much an individual can exploit the potential concurrency.

The third metric is the ***Process Communication Index***. It is based on a *Communication Matrix*:

$$CM_{PS} = \begin{bmatrix} cm_{1,1} & cm_{1,2} & \cdots & cm_{1,n} \\ cm_{2,1} & cm_{2,2} & \cdots & cm_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ cm_{n,1} & cm_{n,2} & \cdots & cm_{n,n} \end{bmatrix} \tag{9}$$

CM_{PS} is expressed by the number of bits sent/received over each channel. So, for each individual IND_j, it is possible to define the *Normalized Total Communication Cost* (NTCC) *Index* as:

$$X_{NTCC_j} = \frac{\sum_{i=1}^{n}\sum_{z=1}^{n} ICCC_{i,z}}{maxNTCC} \tag{10}$$

$$ICCC_{i,z} = \begin{cases} cm_{i,z} & if\ ps_i \in pu_x \wedge ps_z \in pu_y \wedge pu_x \neq pu_y \\ 0 & otherwise \end{cases} \tag{11}$$

$$maxNTCC = \sum_{i=1}^{n}\sum_{z=1}^{n} cm_{i,z} \tag{12}$$

ICCC stands for *Inter Cluster Communication Cost*, that is the cost associated to process communication if processes are allocated on different processors.

The fourth metric is the ***Load Index***. The *Load* L_i is the load that each ps_i would impose to each *non-SPP* processor (used in at least one BB) to satisfy TTC. L_i is estimated by allocating all the *n* processes to a single-instance of each software processor and performing some simulations. Three parameters have to be computed: FRT_z *(Free Running Time), i.e.* the total application simulated time on processor pu_z; t_i, the simulated time for each process ps_i on processor pu_z; N_i, the number of executions. Starting from these estimated parameters, the *Free Running Load* FRL_i is calculated by the equation:

$$FRL_i = \frac{(t_i * N_i)}{FRT_z} \quad \forall i = 1..n \tag{13}$$

where FRT_z/N_i is the average period of each processes on processor pu_z. By imposing that the simulated 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_z equal to *TTC*, for each process/processor pair, such as:

$$\mathrm{TTC} = x_z * FRT_z \qquad with\ 0 \leq x_z \leq 1 \tag{14}$$

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

$$L_i = \frac{(t_i * N_i)}{TTC} = \frac{(t_i * N_i)}{FRT_z} * \frac{FRT_z}{TTC} = \frac{FRL_i}{x_z} \quad \forall i = 1..n \tag{15}$$

From a *DSE* perspective, by considering the sum of the Load L_i of all the processes allocated to a *GPP/ASP*, it is possible to check if the total imposed Load is acceptable. So, it is possible to define the *Load Index* as:

$$X_{L_j} = 1 - \frac{\sum_{i=1}^{s} L_i}{s} \quad with\ s = size(PU) - size(HW\_PU) \tag{16}$$

$$L_i = \begin{cases} \dfrac{FRL_i}{x_z} & if\ TTC \leq FRT_z \\ FRL_i & if\ TTC > FRT_z \end{cases} \tag{17}$$

The fifth metric is the ***Cost Index***. This is a metric related to the monetary cost C_i associated to each bb_i considered in the specific IND_j (considering PU, MU and CU):

$$X_{C_j} = \frac{\sum_{i=1}^{d} C_i}{maxCOST} \quad with\ d = (\#\ bb_i\ used\ in\ IND_j) \tag{18}$$

$$\mathrm{maxCOST} = \mathrm{size(BB)} * \max(C_i) \tag{19}$$

The sixth metric is the ***Size Index***. *Size* is a set of estimations for each statement of each process with respect to each available processor. It is related to number of bytes or area/resources metrics depending on SW or HW implementations:

$$X_{s_j} = X_{SW} + X_{HW} \tag{20}$$

$$X_{SW} = \frac{\sum_{i=1}^{s}(RAM_i + ROM_i) - maxSIZE\_SW}{mazSIZE\_SW} \tag{21}$$

$$X_{HW} = \frac{\sum_{i=1}^{h}(eqG_i) - maxSIZE\_HW}{mazSIZE\_HW} \tag{22}$$

RAM_i and ROM_i are the size value of each process ps_i allocated on SW processor pu_x. eqG_i is the equivalent gate value associated to each process ps_z allocated on HW processor pu_y.

The final metric, specifically introduced in this paper, is the ***Criticality Index***, related to the criticality level associated to each ps_i such as:

$$X_{CRIT} = \begin{cases} 0\ if\ C(ps_i) - C(ps_j) = 0 \wedge\ ps_i \in pu_x \wedge ps_j \in pu_y \wedge pu_x = pu_y \\ 1\ if\ C(ps_i) - C(ps_j) > 0 \wedge\ ps_i \in pu_x \wedge ps_j \in pu_y \wedge pu_x = pu_y \end{cases} \tag{23}$$

The goal behind this metric is to avoid having processes with different criticality levels on the same (shared) processor/core resource. If the constraint is not satisfied, the index value becomes 1, so the final cost function has a higher value if an individual doesn't satisfy criticality constraint.

4.1 DSE with mixed-criticality constraints

Considering the criticality level associated to each process, this work proposes different methods to manage mixed-criticality constraints to avoid interferences derived from damages or software errors and bugs. The main idea is to drive the DSE to avoid having processes with different criticality levels allocated on the same (shared) processor/core. For this, it is exploited the previously defined *Criticality Index*. Moreover, it is also possible to exploit the possibility to constrain the initial population of the GA to have only feasible individuals, and/or to constrain the crossover and mutation steps to make the population evolving only with feasible individuals (with respect to criticality constraints) avoiding generation of unfeasible solutions [25].

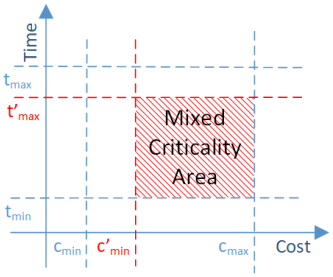

Figure 3: Design space representation with MC requirements.

5 Validation

This section presents some results related to the DSE step. Fig. 3 shows a design space where mixed-criticality constraints are introduced into classical DSE methods. Limiting the processes allocation taking into account mixed-criticality has two main effects: to increase the minimum cost value and to decrease the maximum execution time, because the number of BBs instances will not be less than the number of criticality levels (Figure 3).

Table 1: DSE Parameter Settings

Parameters	Nr.	Values
BBs	≤ 8	2 8051, 2 DSPIC, 2 LEON3, 2 Spartan3an, 2 Virtex-7
App. processes	8	CSP processes
App. Channels	15	CSP channels
GA Selection	1	Random
GA Crossover (C)	1	One-Point
C probability (pc)	1	0.3
GA Mutation (M)	1	Random
M probability (pm)	1	0.1
Survival Selection (S)	1	Fitness-Based
S probability (ps)	1	0.15
Search Iteration (I)	40	-
Initial Population Size (P)	1000	Number of Starting individuals

Table 1 shows the parameters setting related to DSE performed in this section. In the validation step, the available BBs are: **bb1** (SW_PU, GPP): 16 MHz 8-bit 8051 CISC core with 128 byte of Internal RAM, 64KB of internal ROM (cost 10); **bb2** (SW_PU, DSP): 16 MHz 16-bit PIC24 core with 14KB of internal ROM and 1KB of internal RAM (cost 20); **bb3** (SW-PU, GPP): 150 MHz 32-bit LEON3 soft-processor with 2*4 KB L1 caches, RAM size of 4096 KB and a ROM of 2048 KB (cost 100); **bb4** (HW-PU, ASP): 50 MHz Spartan3an (cost 400). **bb5** (HW_PU, ASP): 250 MHz Virtex-7 (cost 900). Considering *AC*, the maximum number of instances for each bb_i is 2, the maximum number of instances of bb_i considered into the DSE is equal to the number of processes (8) and bb_i are supposed to communicate by means of a shared bus.

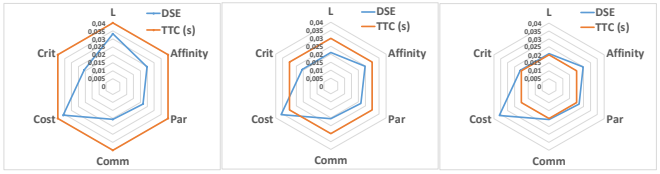

Figure 4: Simulated time with respect to different weights.

A first analysis involves the impact of the different metrics (i.e. indexes) with respect to the simulated time of the final system. Fig. 4 shows some radar charts where each angle represents the simulated time related to DSE solutions while setting one weight equal to 1 and the other ones to zero. The only metrics TTC-dependent is the load L, so the charts present different values for the *Load Index* when changing requested TTC. It is possible to see that the only metrics that badly drives the DSE with respect to timing performance is the *Cost Index* metrics, as could be expected if your goal is only to limit the cost. The other metrics find a sub-optimal solution every time, considering a decreasing TTC. In the adopted case study, the lower bound in the simulated time is driven by the arrival time of the input triggers (i.e. 20 inputs each one every 1 ms), so the simulated time is hovering around 20 ms.

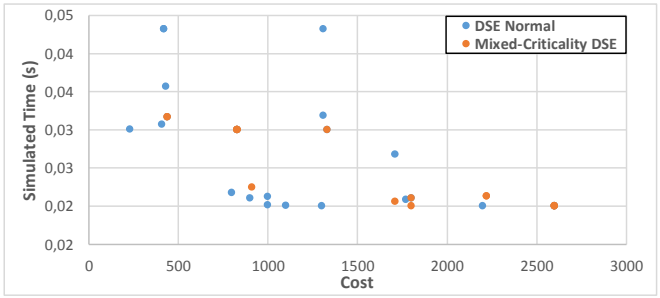

Figure 5: DSE small-set result.

After that, Figure 5 shows a subset of solutions suggested by the DSE while considering different weights and TTCs, with and without MC constraints. As expected, the Pareto set with no MC constraints (blue points more to the left) have solutions with a lower cost with respect to solutions with MC constraints (orange points, as shown in Figure 5). Considering the best solutions, the advantages of this DSE step is to directly identify individuals optimizing different metrics at the same time. Starting from the *Worst Case TTC* (0,06710 s) equal to the simulated time evaluated by means of a timing co-simulation performed allocating all the processes on the slowest available processor (in this case the one in bb_1), the DSE suggests a set of architecture/mapping pairs able to provide TTCs equal, respectively, to 0.90*WCTTC (0,06039 s), 0.75*WCTTC (0,05033 s), 0.5*WCTTC (0,03355 s), 0.40*WCTTC (0,02684 s) and 0.25*WCTTC (0,01678 s, it is

worth noting that such a value is under the lower-bound, so the final simulated time is still above 0.02 s). Given the previous set of TTC constraints, the DSE provides the results shown in Figure 6. The results are always under the blue *Requested Time* line, with costs that increase with the decreasing simulated time (i.e. from right to left). When considering also MC constraints, the DSE suggests solutions that are under the green *Estimated Time* line but with higher costs. In fact, MC applications are more expensive in terms of resources and the final solution space is reduced.

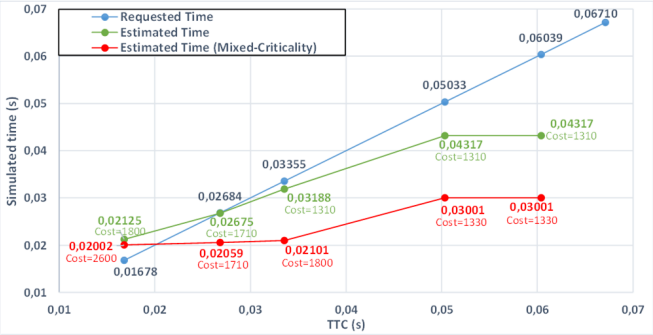

Figure 6: DSE step results

All the steps, prior to DSE one, have been executed in a few minutes (on a high-end notebook). It is worth noting that this is a one-time effort, while the time for the DSE step depends on designer experience and number of considered constraints. A more exhaustive analysis involves more time (Figure 5) with respect to one that directly suggest a possible partitioning/mapping item able to satisfy all the constraints. Just to propose an example, by exploiting an Intel Core i7-6700HQ 2.60 GHz CPU, 16 GB RAM and 64-bit Xubuntu 16.04 operating systems, the chart in Figure 6 has been achieved in 5 min, considering also MC constraints, meanwhile Figure 5 has been achieved in about 15 min because timing simulation has been the most expensive step.

6 CONCLUSIONS

This work has proposed a criticality-driven design space exploration for mixed-criticality heterogeneous parallel embedded systems. By introducing the *Criticality Index* into an evolutionary algorithm, the DSE is able to suggest solutions that fulfill constraints avoiding allocating applications with different levels of criticality on the same shared resource. Results show that mixed-criticality solutions are typically more expensive, and that this work helps to partition in a fast way processes into a heterogeneous parallel platform. Future works involve the introduction into the DSE step also the concept of SW partitions in order to allow modeling also hypervisors technologies.

ACKNOWLEDGMENTS

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

REFERENCES

[1] Prisaznuk, P. J.: Integrated Modular Avionics. In: Proceedings of the IEEE National Aerospace and Electronics Conference (NAECON), 1992.

[2] ISO 26262 - Road vehicles a Functional safety. Geneva, Switzerland, 2011.

[3] Baruah, S., Li, H., Stougie, L.: Towards the Design of Certifiable Mixed-criticality Systems, In: Proceedings of the 16th IEEE RealTime and Embedded Technology and Applications Symposium (RTAS), 2010, pp. 13-22

[4] Pellizzoni, R., Meredith, P., Nam, M. Y., Sun, M., Caccamo, M., Sha, L.: Handling mixed-criticality in SoC-based real-time embedded systems, In: Proceedings Of the 7th ACM international conference on Embedded software (EMSOFT), 2009.

[5] Vestal., S.: Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance. In: 28th IEEE International Real-Time Systems Symposium (RTSS), 2007, pp. 239-243.

[6] Burns, A., Davis, R. I.: Mixed Criticality Systems - A Review. In: Research report, University of York, 2014.

[7] Baruah, S., Li, H., Stougie, L.: Towards the Design of Certifiable Mixed-criticality Systems, In: Proceedings of the 16th IEEE Real Time and Embedded Technology and Applications Symposium (RTAS), 2010, pp. 13-22.

[8] Esper, A., Nelissen, G., and Nélis, V., Tovar, E.: How realistic is the mixed-criticality real-time system model?. In: Proceedings of the 23rd International Conference on Real Time and Networks Systems (RTNS), 2015, pp. 139-148.

[9] Barhorst, J., Belote, T., Binns, P., Hoffman, J., Paunicka, J., Sarathy, P., Stanfill, J. S. P., Stuart, D., Urzi, R.: A research agenda for mixed-criticality systems. In: Workshop on Mixed Criticality (CPS Week), 2009.

[10] J. Teich, "Hardware/Software Codesign: The Past, the Present, and Predicting the Future," in Proceedings of the IEEE, vol. 100, no. Special Centennial Issue, 2012, pp. 1411-1430.

[11] Voss, S., Eder, J., Hölzl, F. Design Space Exploration and its Visualization in AUTOFOCUS3. In: Software Engineering (Workshops), 2014, pp. 57-66.

[12] Herrera, F., Pablo, P., and Eugenio, V. "A model-based, single-source approach to design-space exploration and synthesis of mixed-criticality systems." Proceedings of the 18th International Workshop on Software and Compilers for Embedded Systems. ACM, 2015.

[13] Contrex project, https://contrex.offis.de/home/

[14] 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," 23th International Conference on Architecture of Computing Systems 2010, Hannover, Germany, 2010, pp. 1-7.

[15] Herrera, F. and Sander, I. Combining analytical and simulation-based design space exploration for time-critical systems. Proceedings of the 2013 Forum on specification and Design Languages (FDL), Paris, France, 2013, pp. 1-8.

[16] Rosvall, K., Khalilzad, N., Ungureanu, G. and Sander., I. 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). ACM, New York, NY, USA, 2017.

[17] Hepsycode: A System-Level Methodology for HW/SW Co-Design of Heterogeneous Parallel Dedicated Systems, www.hepsycode.com, Accessed 13/09/2017.

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

[19] D. Di Pompeo, E. Incerto, V. Muttillo, L. Pomante, G. Valente. 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, 2017, pp. 323-326.

[20] Hoare, C. A. R. Communicating sequential processes. Springer, New York, NY, 1978, pp. 413-443.

[21] 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 10th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools (RAPIDO '18). ACM, New York, NY, USA, 2018.

[22] V. Stoico, V. Muttillo, G. Valente, F. D'Antonio. "CC4CS: A Unifying Statement-Level Performance Metric for HW/SW Technologies", Euromicro Conference on Digital Systems Design (DSD) - WIP Session, 2017.

[23] L. Pomante. HW/SW Co-Design of Dedicated Heterogeneous Parallel Systems: an Extended Design Space Exploration Approach. IET Computers & Digital Techniques, vol. 7, no. 6, pp. 246-254, Nov. 2013.

[24] L. Pomante, D. Sciuto, F. Salice, W. Fornaciari, C. Brandolese. Affinity-Driven System Design Exploration for Heterogeneous Multiprocessor SoC. IEEE Transactions on Computers, vol. 55, no. 5, May 2006.

[25] V. Muttillo, G. Valente and L. Pomante. Criticality-aware Design Space Exploration for Mixed-Criticality Embedded Systems. In Proceedings of the 9th ACM/SPEC on International Conference on Performance Engineering (ICPE '18). 2018. Accepted.