Model-Based HW/SW Co-Design Methodology for UAV Systems Development

Vittoriano Muttillo, Vincenzo Stoico
Center of Excellence DEWS
University of L'Aquila
L'Aquila, Italy
vittoriano.muttillo@univaq.it, vincenzo.stoico@graduate.univaq.it

***Abstract*—Nowadays, Cyber-Physical Systems development became the most critical activity for industries, especially in unmanned aircraft systems (UAS), while its main significant component is the unmanned aerial vehicle (UAV). The design of such kind of aircraft requires the use of well-established methodologies and tools. In this context, this paper presents the usage of a model-based HW/SW co-design methodology to develop UAVs in an European project scenario. The UAV hardware platform is made of heterogeneous multicore execution nodes, which makes the design exploration space rather complex to explore. The reference approach is illustrated, while an unmanned aircraft system, taken from AQUAS European project, is used to show the usefulness of the proposed method. Obtained results justify more effort in such a direction as currently done in the proposed methodology has lots of room for improvements, and it is currently used in several ongoing European projects.**

***Index Terms*—embedded system design, HW/SW Co-Design, UAV, design space exploration, timing performance**

I. Introduction

Nowadays, there has been an exponential increase in the exploitation of Cyber-Physical Systems (CPS) in everyday life [1], [2]. Their development became the most critical activity for industries, especially in unmanned aircraft systems (UAS), used in several domains (e.g., precision agriculture, parcel delivery). The main component here is the unmanned aerial vehicles (UAV) [3]. UAVs are expressly efficient to perform a wide variety of tasks (e.g., autonomous control, surveillance, navigation, image processing), and their development is strictly related to the design of complex embedded systems. In this way, it is possible to have a modular heterogeneous system capable of processing several workload types [4]. Heterogeneity involves both Hardware (HW) and Software (SW) and focuses on integrating off-the-shelf components. This approach is an excellent solution to optimize different design metrics. For example, the designer may choose a component configuration to reach an optimal trade-off between size, performance, and costs. However, the continuous demand for high-performance systems led system designers in the adoption of heterogeneous components often on the same integrated circuit (i.e., System on Chip [5], [6]). This technology seems to be the best solution to reach a positive trade-off in terms of design metrics. Systems based on heterogeneous multi-processor architectures (*Heterogeneous Multi-Processor Systems*, HMPS) have been recently exploited for a wide range of application domains (e.g., avionics [7], aerospace [8]). Due to their complexity, the adopted design methodology plays a crucial role in settling the product's quality. However, selecting a suitable implementation is problematic due to the significant number of heterogeneous HW/SW components available on the market.

A model-based approach can be helpful to cope with these challenges. A model is an abstraction of the system, highlighting its essential characteristics. Modeling is a widespread technique, as witnessed by the incredible amount of work in this field. Many works converge in using an abstract representation of the system that is gradually refined in the subsequent steps. Furthermore, models may be built following formal semantics that allows the verification of system functional and non-functional correctness. However, very few HW/SW co-design flows exploit models to support designers during HW/SW partitioning and mapping the system specification into a candidate heterogeneous multi-processor architecture.

In such a context, this work presents the use of an existing HW/SW co-design methodology, called HEPSYCODE [9], to develop UAV platforms inside an industrial UAS real scenario, provided within the AQUAS European project [10]. Several different embedded computing boards have been evaluated, while application models, performance metrics, and Design Space Exploration (DSE) lead the designers to suitable (in terms of safety-oriented partitioning) and powerful (in terms of timing) solutions for the considered scenarios. Results have been reported in AQUAS deliverables, and future enhancement will be exploited in the ongoing COMP4DRONES European Project [11]. Synchronous (rendezvous based) process network Model of Computation (MoC) has been considered as a starting point for the subsequent System-Level HW/SW Co-Design Flow. Simultaneously, heuristic (evolutionary-based) algorithms and SystemC simulations contributed to a correct model-aware hardware boards comparison, introducing timing estimation techniques for first preliminary analysis and platform selection.

II. Related Works

The adoption of modeling languages and tools for CPSs design has experienced significant growth during the last years. This tendency has led to several studies exploiting model-based design for CPSs [12]. This section describes existing design approaches comparable to the proposed one (i.e, HEPSYCODE): Ptolemy II, Metro II, Metronomy, and ForSyDe.

Ptolemy II [13] is a framework for codesign of CPSs developed in the Industrial Cyber-Physical Systems Center (iCyPhy) at the University of California, Berkeley. The framework

follows an actor-based design approach [14] in which actors represent system components. Actors interaction is specified using a model of computation (MoC). Unlike HEPSYCODE, a model contains diverse MoCs identified by Ptolemy directors. A director manages a region of the model called domain. Directors may describe system behavior at different levels of abstraction and combined to form a hierarchy. The hierarchical model serves for the co-simulation of multiple MoCs at the same time. For example, the communication between a domain following a synchronous dataflow (SDF) director and a second one governed by a continuous-time (CT) might be managed by a third director implementing a discrete event (DF) director [15]. Furthermore, DSE activities are limited to a fixed number of selected platform models [16].

Metro II [17] obeys the same principle as Ptolemy supporting several MoCs in a single model. A Metro II model comprises three views of the system: Function, Mapping, and Architecture. An architecture represents system services and their costs. Instead, system functionalities are described as a set of concurrent executing sequential processes. Their interaction is performed through an object called media. Communication is encapsulated in a media instance, and it is detached from processes specification. HEPSYCODE provides a similar separation of concerns employing ports as interfaces for processes that communicate throughout channels. Mapping serves to map the function model to the architectural one. Finally, both Metro II and HEPSYCODE offer functional verification translating the system model into a SystemC model.

Ptolemy and Metro II benefits are exploited in the Metronomy project [18]. Metronomy uses a CoSimDirector to co-simulate the functional model of Ptolemy with the architectural model of Metro II. A timing contract provides a set of assumptions concerning the timing model of the functional and architectural model. Metronomy performs an evolutionary Design Space Exploration (DSE) to output a suitable system implementation. Indeed, the DSE in Metronomy is executed as a multiobjective optimization problem guided by timing checkers. This approach is similar to the one employed by HEPSYCODE. Indeed, HEPSYCODE adopts Timing Co-Simulation to verify the suitability of a solution proposed by the genetic algorithm that drives the DSE. At the end of this process, HEPSYCODE is able to suggest a candidate HW/SW implementation of the system specification. This seems to be a novelty for the HW/SW Co-Design domain.

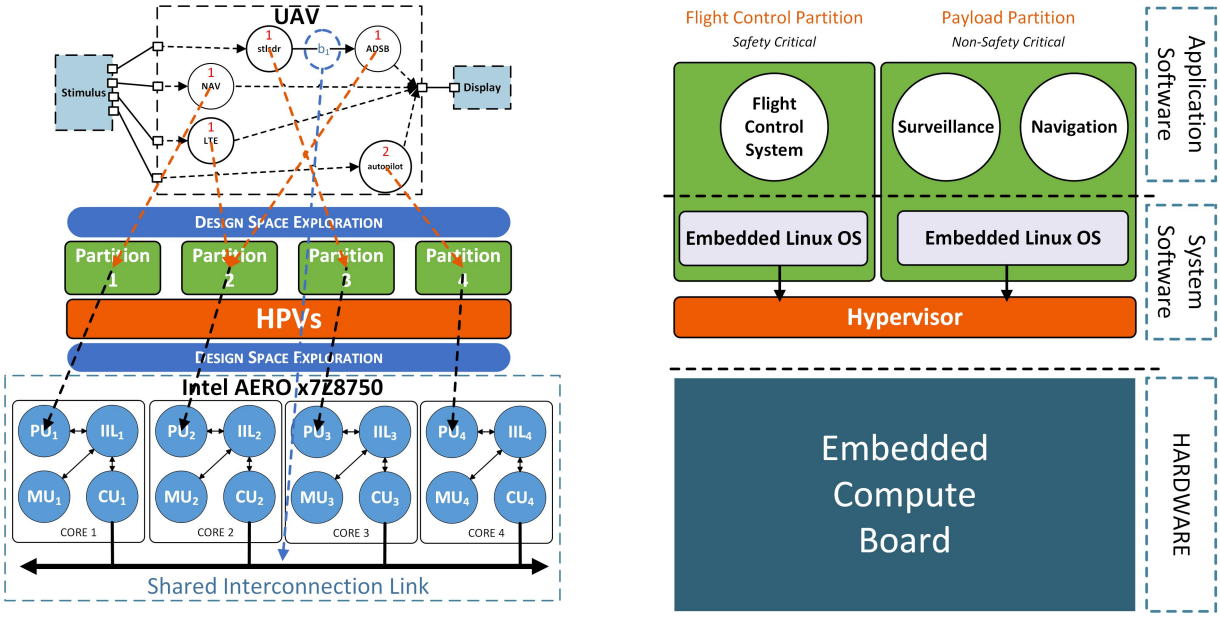
Finally, ForSyDe [19] differs from the above-described methodologies since it uses functional programming for system specification. A system is described as a set of processes executing concurrently. The processes communicate by reading and writing information carried by signals. ForSyDe shares the same features as Ptolemy II and Metro II allowing the specification of multiple MoCs in a single model. The initial model is refined through several steps to obtain a lower-level model synthesizable to an VHDL implementation on FPGA hardware platforms. The transformation includes rules to preserve the functional behavior and non-functional properties of the system. The remainder of this work will present the use of HEPSYCODE methodology in the AQUAS European project scenario. In contrast, future work will analyze and compare several other approaches with further comparative analysis.

III. HEPSYCODE Approach

In the context of Mixed-Criticality Embedded Systems (MCESs) [20], HEPSYCODE adopts a specific Electronic System Level (ESL) HW/SW co-design flow to manage in a right manner Mixed-Criticality (MC), and Real-Time (RT) requirements [21]. During the proposed work, HEPSYCODE has been integrated with another external tool, called CHESS [22], by re-using results from ECSEL RIA 2016 737494 MegaM@Rt2 European Project [23]. The main work focuses on the interoperability between the tools exploiting an automatic transformation pattern between CHESS and HEPSYCODE meta-models using the CERBERO Interoperability Framework (CIF) [24]. This work has been proposed as an initial collaboration pattern among several European project partners to define a uniform interoperability and transformation pattern for several MDE methodologies and tools based on custom meta-models and standard ones (e.g., UML/MARTE).

After the transformation between CHESS (UML/MARTE contract-based) and HEPSYCODE models, the central part of the HEPSYCODE methodology is related to the System Description, the Metrics Evaluation and Estimation, and the Design Space Exploration (DSE) activities that drive designers to find an implementation that fulfills input requirements. The presented combined analysis considers the UAV system partitioning alternatives in processes allocation, binding, and mapping the proposed virtualized environment in this context. The final step is the partition allocation on the different considered cores present in the final board, as shown in Fig. 1(a). The UAV software platform has been modeled inside the HEPSYCODE environment as a series of blocks and modules composing the whole system. Two different design space explorations will find the best allocation among the different HW system components. Several assumptions have to be considered: the UAV can be viewed as a classical mixed-critical embedded platform with constraints on tasks (or processes) allocation. The application with a higher criticality level will not be affected by tasks with a lower criticality.

The first step of the co-design flow is the Functional Simulation where the System Behaviour Mode (SBM) is simulated to check its correctness w.r.t. some test-benches. Test-benches are of critical importance since they have to be as much as possible representative of the possible system's operating conditions. Such a simulation allows timed inputs to be taken into account (i.e., there is a concept of simulated time), but it does not consider the time needed to execute the statements composing the processes; in other words, statements (both computation and communication) are executed in 0 simulated time. HEPSYCODE offers an environment and a set of tools that can also extract information about system behaviour at different abstraction levels. This step aims at extracting as much information as possible about the system by analysing the SBM (i.e., the Application Model) while considering the available HW architectural components (Platform Model) and the use of hypervisors (i.e., Partition Model). Firstly, the tool



(a) Proposed Design Space Exploration

(b) AQUAS Reference UAV Architecture

Fig. 1. HEPSYCODE Reference HW/SW Co-Design Approach.

inspects the potential concurrency. Concurrency is expressed by the set of processes and channels pairs that could be potentially working concurrently and could potentially transfer data concurrently. Concurrency is evaluated by means of the functional simulation. Their evaluation relies on using the HEPSYCODE Simulator (HEPSIM) software, where concurrency has been calculated counting all active processes and channel pairs every time communication occurs. DSE activity uses these values to find better allocations and solutions.

The next activity performed is "Load" metric estimation. The "Load" is the processor utilization percentage that each process would impose on each processor to satisfy imposed timing constraints. The "Load" is estimated by simulating a system where all the processes are allocated onto a single instance of each processor core while considering the need to satisfy imposed timing constraints. If the load is under a pre-fixed upper bound [25], the load upper bound is on the order of $\simeq 70\%$) for a given processor-process pair, this implies that the processor potentially could satisfy such constraints. Simulations exploit previous timing estimations where statements are executed in an estimated time.

After metrics evaluation, the co-design flow reaches the Design Space exploration step. Starting mainly from the Application Model, Partition Model, and Platform Model, it includes two iterative activities: (1) "Search Methods", that perform HW/SW partitioning, architecture definition and mapping using an evolutionary algorithm where the design space is explored looking for feasible architecture/mapping items suitable to satisfy imposed constraints; (2) "Timing Co-Simulation", that considers suggested mapping/architecture items to actually check for timing constraints satisfaction.

The "Search Methods" is split into two main phases: Partitioning, Architecture Definition, and Mapping Phase 1 and 2 (PAM1 and PAM2). PAM1 provides the partial HW/SW architecture (with the number and type of needed processors), the partitioning between HW and SW components, and the mapping between processes and basic hardware components. PAM2 provides the final HW/SW architecture ready to be implemented. PAM2 also finds the number of needed interconnection links (physical links) with a specific topology graph.

Finally, after the search method activity, the "Timing Co-Simulation" helps to check input constraints and to evaluate and compare different solutions applying Pareto analysis, using the HEPSIM (HEPSYCODE SIMulator [26]). The next section presents the experimental results related to a real UAV scenario.

IV. Experimental Results

The reference use case has been taken from the Aggregated Quality Assurance for Systems (AQUAS [27]) European project, i.e., Use Case 1 - Air Traffic Management (ATM) [28]. The main idea was to take as input CHESS (UML/MARTE) ATM models and adapt them to HEPSYCODE modelling approach, using CIF framework [23]. Fig. 1(b) shows the reference AQUAS UAV target architecture, that uses a single multi-core embedded computing board to safely run both flight control, surveillance, and navigation tasks. The initial idea was to consider only two partitions, one flight control partition for safety critical tasks (i.e., the UAV autopilot, which is responsible, among other tasks, for controlling motors to adjust UAV position, heading and speed, keep drone stability, etc.), and one payload partition for non-safety critical tasks (i.e., the navigation and surveillance modules). The ATM UAV model and the HEPSYCODE workspace are shown in Fig. 2. The autopilot process is the safety critical task designated for single partition allocation. The autopilot task periodically receives the sensors' data and the orders from the control station. Based on them, it generates the output signals for actuation. An essential element is the model of the physical

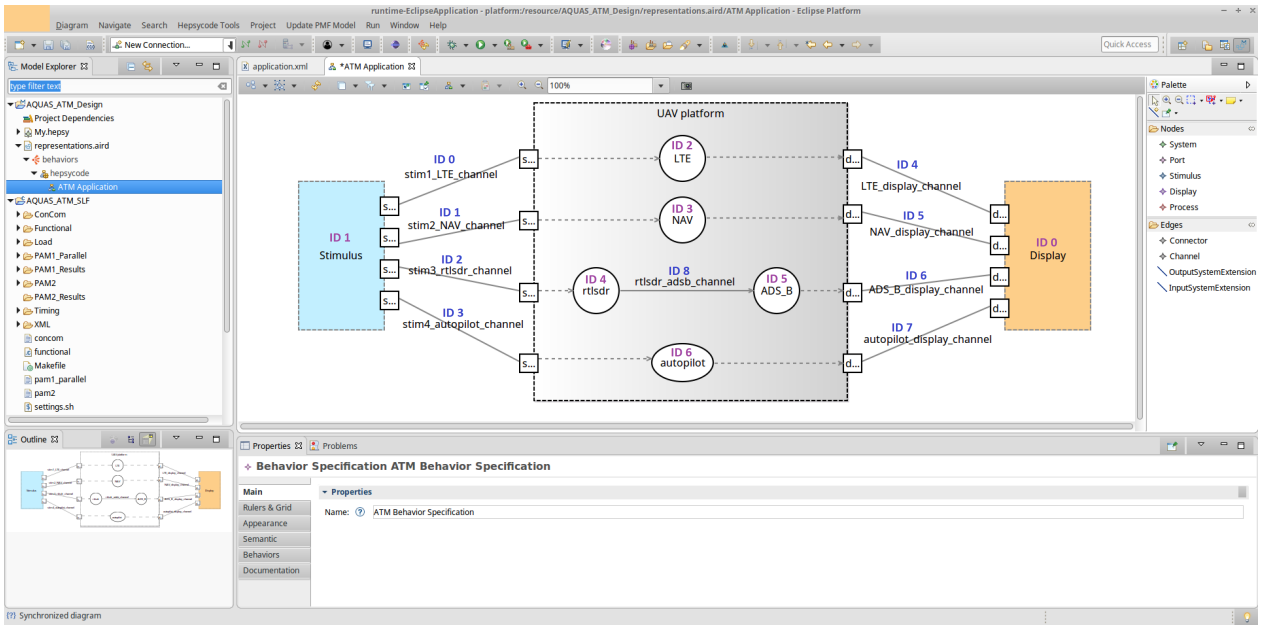


Fig. 2. HEPSYCODE workspace and ATM modelling Activity

environment in which the UAV operates. This model has to simulate the aerodynamics of the UAV. Either autopilot code, ground station control data, and environment test bench have been provided to us by AQUAS ATM use case owners. The other tasks are not mission critical, so they can share resources and compete to access CPU and memory locations. The main reference model is a synchronous process network Model of Computation (MoC) represented by threads inside SystemC modules, where tasks and processes have the same meaning in HEPSYCODE workflow. Starting from Fig. 2, the system model is translated into a SystemC executable functional model by means of Model Driven Engineering (MDE) techniques. As said before, for each UAV process/task, also defined at CHESS process level, a single-source SystemC code has been generated. The main tasks' behaviour have been extracted from source code offered by Integrasys [29] during the design and verification activities. It is worth noting that not all the functionalities have been implemented by Integrasys since the goal of the use case was the design methodology and not the final system realization (i.e., reduced task functionalities). The design methodology considers combined analysis to analyse the impact of performance, safety, and security into a co-engineering evaluation approach, integrated within the AQUAS Product Life-Cycle Co-Engineering method [27]..

After Model-to-Model transformation (using MDE Eclipse tools, i.e., Sirius [30], and Xtext [31]), the SystemC model has been generated, where the correct computation can be checked. The designer can refine the code at system-level while focusing on other possible metrics that can better drive the designer choice during the whole Product Life-Cycle development flow. The next activity involved the metrics evaluation and estimation using SystemC simulation, based on the considered synchronous process network model. Processes and channels concurrency values have been calculated, starting from AQUAS

TABLE I
WORKLOAD ESTIMATION (USED FOR DSE STEPS) WITHOUT OPERATING SYSTEM OVERHEAD (SINGLE-CORE LEON3 ON VIRTEX-7)

Process	Sim. ET	Load Metric
LTE	71.1068 ms	11.4566%
NAV	290.361 ms	46.7823%
Rtlsdr	91.0938 ms	14.6768%
ADS_B	112.907 ms	18.1914%
Autopilot	55.1954 ms	8.8929%
Tot.	620.664 ms	100%

ATM use case. In this work, we do not consider communication or links among system components since the primary purpose is system enhancement in performance and safety constraints. The next metric considered is the "Load" one while the board's description and model have been added into the workspace, described by XML schemas and files.

Several selected hardware architectures have been considered as possible alternatives for the ATM use case. We chose to reduce the number of possible boards to 4: (1) Intel **AERO**, that mounts an Intel **ATOM** x7-Z8750 64 bit processor with 4 cores/4 threads, 2.56GHz burst, 2M Cache, 4 GB RAM, and an Altera MAX 10 FPGA; (2) The **Raspberry** Pi 3, that mounts a quad Core 1.2GHz **ARM** Cortex-A53, and 1GB RAM; (3) The **ZedBoard**, that mounts a Zynq-7000 AP SoC XC7Z020-1CLG484 with a dual-core **ARM** Cortex-A9 MPCore with CoreSight at 866 MHz, L1 Cache 32 KB Instruction, L1 Cache 32 KB data per processor, L2 Cache 512 KB, On-Chip Memory 256 KB, 512 Mbyte DDR3, and an Artix-7 FPGA; (4) **Virtex**-7 FPGA mounted on the VC707 board. The VC707 board provides features common to many embedded processing systems, including a LVDS 200 MHz oscillator (U51) and

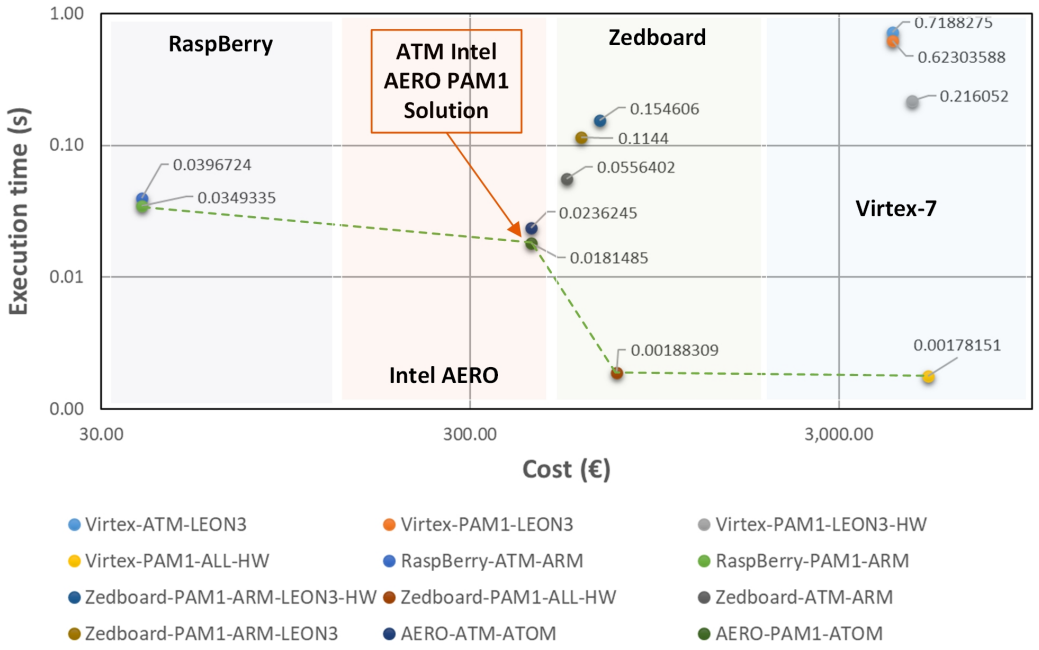


Fig. 3. Design Space Exploration Alternatives w.r.t. total (average) execution time and cost (i.e., monetary and design cost).

TABLE II
HEPSIM TIMING SIMULATION (MULTI-CORE LEON3 ON VIRTEX-7)

Process	Sim. ET ATM	Sim. ET PAM1
LTE	64.4138 ms	64.4138 ms
NAV	289.409 ms	291.633 ms
Rtlsdr	185.78 ms	78.2138 ms
ADS_B	123.974 ms	133.497 ms
Autopilot	55.2507 ms	55.2507 ms
Tot.	718.8275 ms	623.03588 ms

DDR3 SODIMM memory.

We have also decided to synthesize inside the FPGAs even a 2-core multi-processor system with the **LEON3** [32], a 32-bit RISC soft-microprocessor with seven-stage pipeline, and a 75 MHz system clock. It is possible to implement tasks on FPGAs in software (running on LEON3 processors) and/or hardware (e.g., tasks described in VHDL code) in this way (e.g., tasks on ZedBoard can be executed in SW on ARM and LEON3 cores, or HW on FPGA [33]). Table I presents the Simulated HEPSIM Execution Time (Sim. ET) and the load metric evaluation for the ATM UAV application running on LEON3 single-core synthesized on Virtex-7 FPGA. The following DSE approach uses all these metrics.

The reference ATM UAV allocation, chosen by the AQUAS partners as a reference allocation and binding solution, as shown in Fig. 1(b), considers the allocation of the autopilot on a single-core partition while allocating the remainder on another hypervisor partition on the same core. From a DSE point of view, this solution may not be the best feasible one; the idea was to compare alternatives and motivate the choice of this partitioning plan for the AQUAS use case, maybe changing the proposed allocation. Fig. 1(a) presents the solution found after one run of the search method step, considering the Intel ATOM board. The DSE has been constrained with respect to the hypervisors feature (i.e., partitions), while each partition can be allocated only on one core, avoiding the possibility to have two or more partitions on the same shared resources, and to guarantee spatial and temporal isolation for the different mixed-critical tasks set. The number of feasible solutions, and the average cost function values, do not change enough during the evolutionary algorithm runs, while the reduced number of cores and processes is reflected on the total population size and average cost functions values.

Table II presents results related to the allocation and mapping found during the search method run and the timing simulations extracted from the HEPSIM tool [26] considering LEON3 on Virtex-7 scenario. Finally, Fig. 3 compares the solutions obtained from the design space exploration considering all the boards. The different acronyms refer to regarded boards (Virtex-7, Raspberry, AERO, Zedboard), mapping solutions (ATM stands for AQUAS reference UAV architecture, PAM1 stands for HEPSYCODE DSE result), and processor technologies (ARM, LEON3, ATOM, and HW synthesized in VHDL code). From this graph, it seems that Intel AERO is one of the best solutions in terms of performance and cost. Simultaneously, a new allocation and partitioning plan concerning process allocation on partitions and cores have to be used to improve the whole system's performance.

V. CONCLUSION AND FUTURE WORKS

This work presented an HW/SW Co-Design approach integrated inside the AQUAS Co-Engineering method (i.e., considering safety/security/performance requirements at different In-

teraction Points [27]). The proposed approach helps design and simulating UAV systems in a cyber-physical environment while considering several orthogonal non-functional constraints. The results allow the AQUAS partners to evaluate several embedded computing board alternatives, while the final selected board was the better solution found during the co-design flow. Future works will consider the possibility of using an ARM multi-core system (e.g., raspberry boards) instead of the selected Intel board to validate the presented approach. Furthermore, FPGA technologies will be a possible alternative to implementing HW processes and tasks to accelerate the system computation and reduce workload. Designers can consider all these aspects as likely UAV's future improvements.

ACKNOWLEDGMENT

This work has been partially supported by the ECSEL RIA 2016 737475 AQUAS and ECSEL-JU 2018 826610 COMP4DRONES projects.

REFERENCES

[1] P. Derler, E. A. Lee, and A. Sangiovanni Vincentelli, "Modeling cyber–physical systems," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 13–28, 2012.

[2] Y. Z. Lun, A. D'Innocenzo, F. Smarra, I. Malavolta, and M. D. D. Benedetto, "State of the art of cyber-physical systems security: An automatic control perspective," *Journal of Systems and Software*, vol. 149, pp. 174 – 216, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0164121218302681

[3] G. Hoffmann, H. Huang, S. Waslander, and C. Tomlin, *Quadrotor Helicopter Flight Dynamics and Control: Theory and Experiment*. AIAA ARC, 2012.

[4] *Avionics Application Software Standard Interface: ARINC Specification 653P1-3, Required Services*, Accessed: 01.04.2019., aeronautical Radio, Inc. 2010-11-15. Retrieved 2013-10-20.

[5] Xilinx. (2020, nov) Xilinx zynq7000. Xilinx Inc. [Online]. Available: http://www.xilinx.com/

[6] G. Valente, T. Di Mascio, G. D'Andrea, and L. Pomante, "Dynamic partial reconfiguration profitability for real-time systems," *IEEE Embedded Systems Letters*, pp. 1–1, 2020.

[7] E. Villar, J. Merino, H. Posadas, R. Henia, and L. Rioux, "Mega-modeling of complex, distributed, heterogeneous cps systems," *Microprocessors and Microsystems*, vol. 78, p. 103244, 2020.

[8] V. Muttillo, L. Tiberi, L. Pomante, and P. Serri, "Benchmarking analysis and characterization of hypervisors for space multicore systems," *Journal of Aerospace Information Systems*, vol. 16, no. 11, pp. 500–511, 2019.

[9] L. Pomante, V. Muttillo, M. Santic, and P. Serri, "Systemc-based electronic system-level design space exploration environment for dedicated heterogeneous multi-processor systems," *Microprocessors and Microsystems*, vol. 72, 2020.

[10] M. W. Christian Fuss. (2018, nov) Deliverable d4.1 - report on co-engineering process support. AQUAS. [Online]. Available: http://aquas-project.eu/wp-content/uploads/2019/09/D4.1.pdf

[11] M. Hussein, R. Nouacer, Y. Ouhammou, E. Villar, F. Corradi, C. Tieri, and R. Castiñeira, "Key enabling technologies for drones," in *2020 23rd Euromicro Conference on Digital System Design (DSD)*. Kranj, Slovenia, Slovenia: IEEE, 2020, pp. 489–496.

[12] S. K. Khaitan and J. D. McCalley, "Design Techniques and Applications of Cyberphysical Systems: A Survey," *IEEE Systems Journal*, vol. 9, no. 2, pp. 350–365, Jun. 2015, conference Name: IEEE Systems Journal.

[13] J. Eker, J. W. Janneck, E. A. Lee, Jie Liu, Xiaojun Liu, J. Ludvig, S. Neuendorffer, S. Sachs, and Yuhong Xiong, "Taming heterogeneity - the ptolemy approach," *Proceedings of the IEEE*, vol. 91, no. 1, pp. 127–144, 2003.

[14] E. A. Lee, S. Neuendorffer, and M. J. Wirthlin, "Actor-Oriented Design of Embedded Hardware and Software Systems," *Journal of Circuits, Systems and Computers*, vol. 12, no. 03, pp. 231–260, Jun. 2003, publisher: World Scientific Publishing Co.

[15] Q. Zhu and A. Sangiovanni-Vincentelli, "Codesign methodologies and tools for cyber–physical systems," *Proceedings of the IEEE*, vol. 106, no. 9, pp. 1484–1500, 2018.

[16] H. Kim, L. Guo, E. A. Lee, and A. Sangiovanni-Vincentelli, "A tool integration approach for architectural exploration of aircraft electric power systems," in *2013 IEEE 1st International Conference on Cyber-Physical Systems, Networks, and Applications (CPSNA)*, 2013, pp. 38–43.

[17] A. Davare, D. Densmore, T. Meyerowitz, A. Pinto, A. Sangiovanni-Vincentelli, G. Yang, H. Zeng, and Q. Zhu, "A next-generation design framework for platform-based design," in *DVCon 2007*, February 2007. [Online]. Available: http://chess.eecs.berkeley.edu/pubs/228.html

[18] L. Guo, Q. Zhu, P. Nuzzo, R. Passerone, A. Sangiovanni-Vincentelli, and E. Lee, "Metronomy: A function-architecture co-simulation framework for timing verification of cyber-physical systems," *2014 International Conference on Hardware/Software Codesign and System Synthesis, CODES+ISSS 2014*, 10 2014.

[19] G. Ungureanu, T. Sundström, A. Åhlander, I. Sander, and I. Söderquist, "Design of sensor signal processing with ForSyDe: Modeling, validation and synthesis," KTH Royal Institute of Tehnology, Tech. Rep., 2019. [Online]. Available: https://forsyde.github.io/docs/aesa-radar/

[20] A. Burns and R. Davis. (2019, nov) Mixed criticality systems - a review. University of York. [Online]. Available: https://www-users.cs.york.ac.uk/burns/review.pdf

[21] V. Muttillo, G. Valente, and L. Pomante, "Design space exploration for mixed-criticality embedded systems considering hypervisor-based sw partitions," in *2018 21st Euromicro Conference on Digital System Design (DSD)*, 2018, pp. 740–744.

[22] INTECS. (2008, nov) Chess: Composition with guarantees for high-integrity embedded software components assembly. CHESS Consortium. [Online]. Available: http://rcl.dsi.unifi.it/projects/chess/chess

[23] Consortium. (2019, nov) D5.3: Megam@rt integrated framework - final version. MegaM@rt2 Consortium. [Online]. Available: https://megamart2-ecsel.eu/wp-content/uploads/2020/05/D5.3-MegaM@Rt-Integrated-Framework-final-version.pdf

[24] M. van den Baar and J. Oliveira. (2017, nov) Cerbero interoperability framework. CERBERO Consortium. [Online]. Available: https://www.cerbero-h2020.eu/toolchain/cif/

[25] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, p. 46–61, Jan. 1973.

[26] D. Ciambrone, V. Muttillo, L. Pomante, and G. Valente, "Hepsim: An esl hw/sw co-simulator/analysis tool for heterogeneous parallel embedded systems," in *2018 7th Mediterranean Conference on Embedded Computing (MECO)*, 2018, pp. 1–6.

[27] L. Pomante, V. Muttillo, B. Křena, T. Vojnar, F. Veljković, P. Magnin, M. Matschnig, B. Fischer, J. Martinez, and T. Gruber, "The aquas ecsel project aggregated quality assurance for systems: Co-engineering inside and across the product life cycle," *Microprocessors and Microsystems*, vol. 69, pp. 54–67, 2019.

[28] Consortium. (2016, nov) Use cases. AQUAS. [Online]. Available: https://aquas-project.eu/use-cases/

[29] *Integrasys*, 2020 (accessed: 24.11.2020). [Online]. Available: https://www.integrasys-space.com/

[30] *Sirius - The easiest way to get your own Modeling Tool*, 2020 (accessed: 24.11.2020). [Online]. Available: https://www.eclipse.org/sirius/

[31] *Xtext - Language Engineering Made Easy!*, 2020 (accessed: 24.11.2020). [Online]. Available: https://www.eclipse.org/Xtext/

[32] V. Muttillo, G. Valente, F. Federici, L. Pomante, M. Faccio, C. Tieri, and S. Ferri, "A design methodology for soft-core platforms on fpga with smp linux, openmp support, and distributed hardware profiling system," *EURASIP Journal on Embedded Systems*, vol. 1, no. 15, pp. 1–14, 2016.

[33] G. D'Andrea, T. Di Mascio, and G. Valente, "Self-adaptive loop for cpss: is the dynamic partial reconfiguration profitable?" in *2019 8th Mediterranean Conference on Embedded Computing (MECO)*, 2019, pp. 1–5.