Information and Communication Technologies (ICT) Programme

Project N°: H2020-ICT-2016-1-732105

D5.2: *CERBERO Framework Components*

Lead Beneficiary: UniCA

Workpackage: WP5

Date: 06/06/2019

Distribution - Confidentiality: [Public]

Abstract: This deliverable describes all the CERBERO framework components, identifying all the parts composing the cross-layer model-based structure for design, optimization, verification and deployment of complex cyber-physical systems and systems of systems. The document presents separately all the components/tools, starting from their motivations and already provided features, going to the extensions envisioned in order to attain CERBERO objectives and to accomplish use case requirements.

Disclaimer

The CERBERO Consortium is the following:

Num.	**Beneficiary name**	**Acronym**	**Country**
1 (Coord.)	IBM Israel – Science and Technology LTD	IBM	IL
2	Università degli Studi di Sassari	UniSS	IT
3	Thales Alenia Space Espana, SA	TASE	ES
4	Università degli Studi di Cagliari	UniCA	IT
5	Institut National des Sciences Appliquees de Rennes	INSA	FR
6	Universidad Politecnica de Madrid	UPM	ES
7	Università della Svizzera italiana	USI	CH
8	Abinsula SRL	AI	IT
9	Ambiesense LTD	AS	UK
10	Nederlandse Organisatie Voor Toegepast Natuurwetenschappelijk Ondeerzoek TNO	TNO	NL
11	Science and Technology	S&T	NL
12	Centro Ricerche FIAT	CRF	IT

For the CERBERO Consortium, please see the http://cerbero-h2020.eu web-site.

Document Authors

The following list of authors reflects the major contribution to the writing of the document.

Name(s)	**Organization Acronym**
Carlo Sau	UniCA
Tiziana Fanni	UniCA
Francesca Palumbo	UniSS
Pablo Muñoz	S&T
Antoine Movran	INSA
Michael Masin	IBM
Alfonso Rodriguez	UPM
Rafael Zamacola	UPM
Daniel Madroñal	UPM
Eduardo Juarez	UPM
Raquel Lazcano	UPM
Luca Pulina	UniSS
Simone Vuotto	UniSS
Joost Adriaanse	TNO

Document Revision History

Date	**Ver.**	**Contributor (Beneficiary)**	**Summary of main changes**
07/05/2019	0.1	UniCA	Initial ToC
27/05/2019	0.2	UniCA, UniSS	MDC related contributions
29/05/2019	0.3	S&T	MECA related contributions
03/06/2019	0.4	UniCA, INSA	PREESM/SPIDER related contributions, section 1 draft, section 3 draft
05/06/2019	0.5	IBM	AOW related contributions
05/06/2019	0.6	INSA	Updated PREESM/SPIDER related contributions

06/06/2019	0.7	UPM, UniSS, TNO	ARTICo^3, IMPRESS, PAPIFY, SAGE, DynAA related contributions
08/06/2019	0.8	UniCA	Final draft ready for internal review
19/06/2019	0.9	UniCA	General actions required by the internal review
30/06/2019	1.0	ALL	Partners actions required by the internal review, integration and refinements

1. Executive Summary

This document is an update of D5.6, related to the progress of the main components/tools of the CERBERO framework at M30. The CERBRRO framework is a design environment for Cyber-Physical Systems (CPSs) based on cross-layer modelling and on an advanced adaptivity support. As for D5.6, each framework component/tool will be described separately to provide:

- a short summary of the starting point at the beginning of the project, coming from D5.6;
- a description of the new features and extensions developed during the project (from M15 to M30);
- a discussion on the possible deviations of those features and extensions from the original plan at the beginning of the project (D5.6);
- a description of what these new features and extensions are bringing to the same CERBERO framework and of how they will be useful for the Use Cases (UCs) of the project.

By the description of the progress of each component/tool it should be clear how components/tools have been extended and/or modified and how they will contribute to meet the CERBERO objectives and the use cases needs.

1.1. Structure of Document

The document is focused on the progress of the main component/tools composing the CERBERO framework. Section 2 provides firstly an overview of the framework, of its views and abstraction/application layers. Then, Sections from 2.1 to 2.10 provide a detailed discussion of the new features and extensions of each component/tool, as well as the resulting benefits for the framework and for the CERBERO use cases. Moreover, as resuming useful information for the framework, available websites, download links, tutorials and important references are listed for each tool/component. To conclude, Section 3 summarises the final supported features of the CERBERO framework components/tools, as well as their positioning within the CERBERO UCs and abstraction/application layers.

Since lots of acronyms are used in the document, as occurred for D5.6, a dedicated introductory section, Section 1.3, is provided to keep the content of the document more readable.

1.2. Related Documents

The CERBERO deliverables related to this document are:

- D3.2 – Models of Computation
    - D5.6 deals with CERBERO framework components which are often based on specific models of computations that are described in D3.2, such as PiSDF for PREESM.

- D4.1 – Multilayer Adaptation Strategies
  - Some components/tools extensions mentioned in D5.2 are actively involved in the CERBERO multilayer adaptation strategies and, to avoid repetitions, their detailed description, provided in D4.1, is only referenced in D5.2.
- D4.4 – Self-Adaptation Manager
  - Several tools/components presented in D5.2 provide support for the CERBERO self-adaptation infrastructure described in D4.4, i.e. run-time monitoring offered by PAPIFY.
- D5.1 – CERBERO Holistic Methodologies & Integration Interfaces (Final)
  - Some ongoing works on new features and extensions of the CERBERO framework components exposed in D5.2 are also involved in components/tools integration. Being overlapped between the corresponding tasks, they will be completed during integration activities.
- D5.5 – CERBERO Holistic Methodologies & Integration Interfaces (Ver. 2)
  - New features and extensions of the CERBERO framework components exposed in D5.2 are related to component/tool isolated evolution, but often it has been required to mention also components/tools integration to motivate or better describe a standalone modification.
- D5.6 – CERBERO Framework Components
  - This is the previous (M15) version of this deliverable, describing in detail the CERBERO framework components/tools at M15 as well as their involvement in the project and their relevance for the UCs. This deliverable should be read according to its previous version since to avoid repetitions lots of references to D5.6 are adopted.
- D5.7 – CERBERO Framework Demo
  - In D5.7 the integration of the tools/components presented in D5.2 is discussed according to their involvement in the CERBERO framework demonstrators.

1.3. List of Acronyms

In this section the most recurrent acronyms of the document are remarked:

- API – Application Programming Interface
- CPS – Cyber-Physical System
- DSE – Design Space Exploration
- DPR – Dynamic and Partial Reconfiguration
- HDL – Hardware Description Language
- HLS – High Level Synthesis
- HW – Hardware
- KPI – Key Performance Indicator
- MoC – Model of Computation
- MoA – Model of Architecture
- OM – Ocean Monitoring

- PiSDF – Parameterized and interfaced Synchronous DataFlow
- PMC – Performance Monitoring Counter
- PE – Planetary Exploration
- ST – Smart Travelling
- SW – Software
- UC – Use Case

2. The CERBERO framework components

With respect to M15, the CERBERO framework has evolved under several aspects, so that also the way its overview is given has changed. Design-time and run-time views of the CERBERO framework have been identified, as it can be noticed in Figure 1, which depicts the design-time one, and Figure 2, which represents the run-time one.

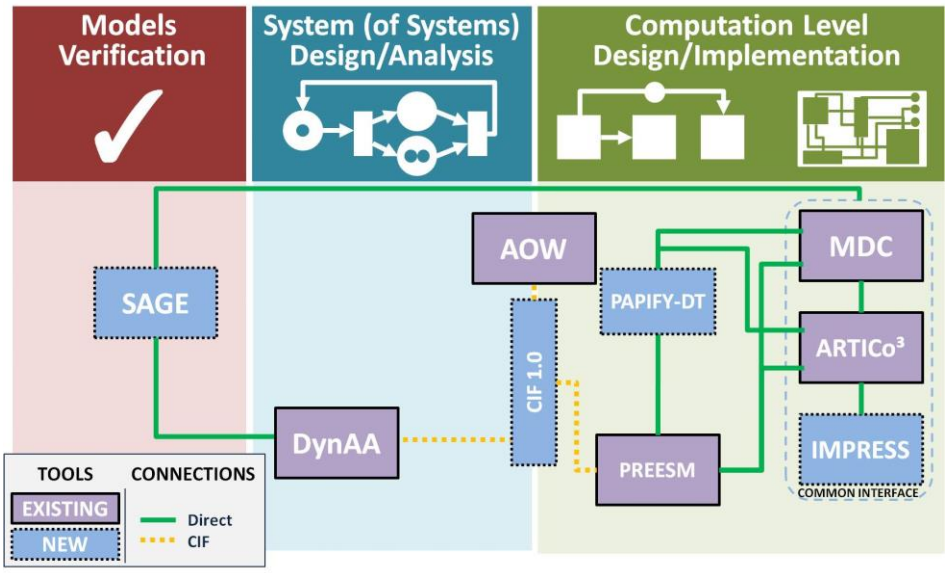


Figure 1 – CERBERO framework components overview: design time.

Components/tools are still spread among different levels of abstraction/application, which have been slightly re-organized. We go from models verification with SAGE (previously named VT), passing through system (of systems) level with DynAA, MECA and AOW, up to the computation level with PREESM, SPIDER, PAPIFY, IMPRESS (previously referred to as JIT HW), ARTICo³ and MDC. The advancement and status of the tools/components will be discussed individually in Sections from 2.1 to 2.10. Connections between framework tools/components are direct (tool-to-tool connections) or they leverage on the CERBERO Intermediate Format (CIF) just released in its 1.0 version (for more details please refer to D5.5).

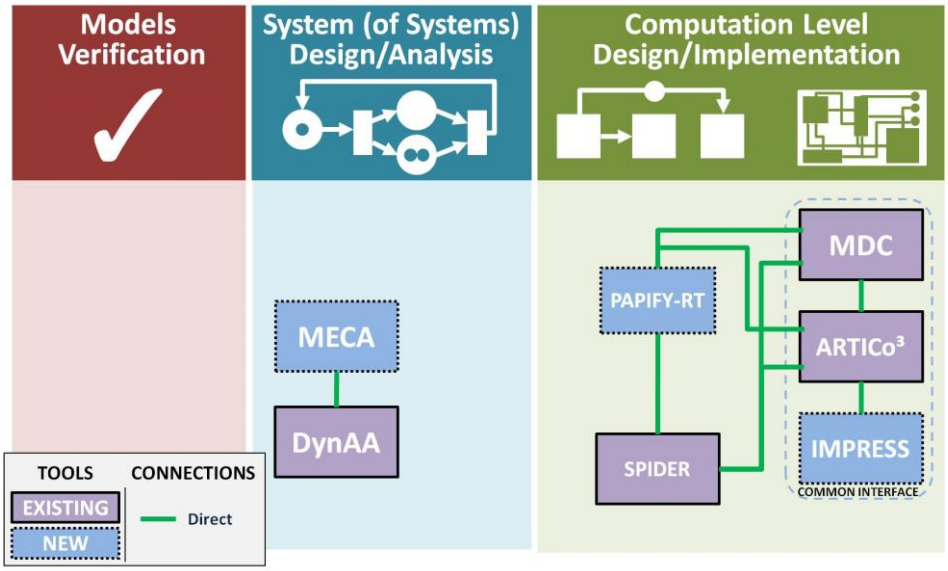


Figure 2 – CERBERO framework components overview: run time.

2.1. MECA

a) Starting Point Summary

For the Smart Travelling (ST) demonstrator at M15 MECA supported the minimum functionality for carrying the scenario execution, monitoring user and environmental (reserved charging pole out of order) adaptation triggers. Also, the connection with DynAA was set and a frontend for the demonstrator was deployed.

b) New Features and Extensions

After M15 MECA has been extended with the following functionalities required for the final ST demonstrator:

- New support is added to enable intermediate waypoints in the route planner.
- The advice module is updated to provide more user-specific information about the routes ranking.
- Temporal allocation of the charging stop is now used as part of the route ranking. This implies an analysis of the facilities near to the charging poles regarding to the expected arrival time. For instance, stopping during lunch time implies that MECA will try to propose routes in which the charging stop is close to a restaurant.
- New monitors have been implemented to enable adaptation for different purposes and situations: (a) driver is no longer following the route; (b) detecting overconsumption with respect to the expected (by DynAA) battery consumption and; (c) re-planning when the driver includes a new stop in the current itinerary.

- A new car simulator is implemented for automated testing. This simulator allows to assess different conditions such as low battery level, driver tiredness or out of route positions for triggering different adaptation behaviors.

c) Differences with Respect to the Plan

At the beginning, MECA was designed as a monitoring rule-based engine. However, a re-implementation was made to follow an execution-based approach. This means that, instead of using a formal language for the definition of monitoring rules, the new approach followed is to use a Planning Domain Definition Language (PDDL), a first order logic language. The planner, according to a domain and problem defined in PDDL, is then in charge of deciding what are the required actions to execute and which monitors shall be activated to control the execution. Each module deployed can be controlled and monitored by the MECA execution engine as a result of the automated planner. On the one hand, this simplifies the development of modules (for the ST demonstrator DynAA module, route planner, car interface and charging pole infrastructure are present) as each module has to be built upon a predefined schema. On the other hand, the planner expresses the capabilities and constraints of the system in a high level language that is a standard in the planning community, actually allowing to use different planners inside MECA.

From the point of view of integration with other tools in the ST Use Case (UC), originally it was planned to connect MECA with DynAA and the SCANeR simulation suite. However, the second interface will not be deployed as for the UC it has been decided to use an HMI (implemented by AI) that will be integrated with SCANeR. Thus, MECA will interact with this new HMI in order to provide the information about routes and retrieve the car and driver data.

d) Relevance with respect to the CERBERO framework and to UCs

The new developments inside the MECA tool will enable a better monitoring of the ST UC by defining extra monitoring possibilities for common situations during a car trip. In this regard, these extensions rely on the information exchange which has been designed considering not only interoperability between CERBERO tools but a more general approach using open standards (like the schema used for the route planner based on the OSRM service) and adding extra information required to deal with the ST UC.

Moreover, the modular implementation of MECA and the modules deployment allowed us to adopt an incremental implementation approach, making it possible to have an easy transition between the expected interface with SCANeR to the new interface with the AI HMI. In particular, this has been done by re-using part of the itinerary schema used with DynAA and by defining a new one (to carry car and driver information) to exchange information with the HMI, implementing a Rest server inside MECA.

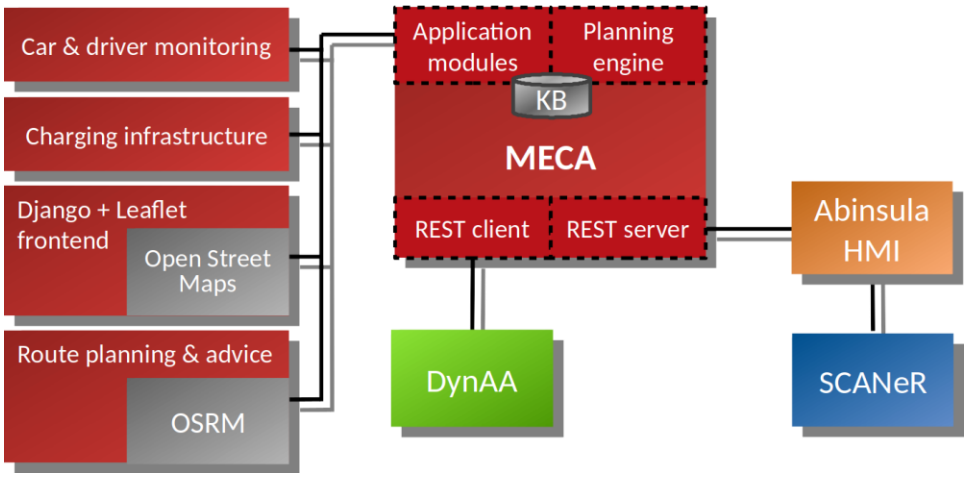


Figure 3 - Final MECA tool and modules and interfaces with other CERBERO tools for the ST UC.

As the interconnection point between different tools inside the ST UC, MECA is not only in charge of performing the route planning and ranking but it has also to monitor different tools and the user to trigger CPSoS adaptation based on the current context and the expected system status. The final architecture of the MECA tool within the ST UC is depicted in Figure 3.

e) Available Resources

Website:

- http://stcorp.nl/highlights/decision-support-systems/

Download:

- not available

Tutorials:

- Not available

Relevant Publications:

- [Bosse 2017]
- [Breebaart 2009]
- [Breebaart 2008]
- [Meerincx 2008]

2.2. SAGE

a) Starting Point Summary

At M15, the SAGE Verification Suite (previously called Verification Tool, VT, in D5.6) implemented a first prototype of two tools in a very early development stage (they were part of VT without even having a dedicated name):

- ReqV: a tool to formalize requirements into a formal specification and to perform consistency checking;
- HyDRA: a tool to support the controller synthesis over discrete domains and accounting for a limited form of temporal specification.

Available in SAGE there is also a SoftWare (SW) library (SpecPro) aiming at supporting the verification tasks involved in the SAGE tools.

b) New Features and Extensions

ReqV

At M30, the user interface of ReqV has been improved and new functionalities added. In particular:

- a new wizard to help the user while writing requirements following the correct syntax has been implemented;
- new features to manage, filter and select requirements have been added in the requirements table interface;
- the capability to extract a minimal subset of inconsistent requirements (also called MUC extraction) with a new performing algorithm has been added (this feature helps the user while searching problems in the specification).

HyDRA

Concerning HyDRA, its development since M15 has focused on the support for controller synthesis in the context of hybrid systems, thus developing the capabilities of modeling and reasoning for CPSs with continuous dynamics.

In particular, Hydra has been extended with the following functionalities:

- extension of the input language, initially supporting classical planning only, with the specification of the continuous dynamics of the systems. The System dynamics are specified as partial differential equations dependent on the control input, allowing its compilation to hybrid automata.
- support for controller synthesis on hybrid systems. Hydra develops an integrated approach mixing discrete satisfiability and numeric optimization for the synthesis goal-oriented controllers.

SpecPro and test generation

The ReqV code has been modularized and the key algorithms have been encapsulated into a Java library called SpecPro (v0.3.0). SpecPro is accessible though Application Programming Interfaces (APIs) and with a command line interface. In addition to PSPs, it also accepts Linear Temporal Logic (LTL) as an input language.

SpecPro also contains a beta implementation of a new testing generation technique that, starting from a LTL specification, produces sequences of assignments that can be applied

on the system under test to prove its validity with the initial requirements set. It will be a starting point for future extensions beyond CERBERO.

c) Differences with Respect to the Plan

In addition to what has been planned, we provided a preliminary implementation of an automated test generation algorithm in the SpecPro library, in order to offer to the system designer a way to better assess the designed model with respect to the requirements.

d) Relevance with respect to the CERBERO framework and to UCs

The features implemented in ReqV ease the definition and verification of functional requirements in the early stage of the design process. The same requirements can then be used at the end of the development cycle to validate the implementation through the test suite generated with the new proposed methodology.

Considering HyDRA, it can be used to verify the capabilities of a CPS currently being modeled to support a variety of scenarios. Its model-centric approach enables the reuse of the same model as the ones developed in other components and in particular in ReqV for the specification of formal requirements that the system must respect. It provides a testing facility where HyDRA can be used to prove that the modeled system is capable of handling a particular scenario, where the proof takes the form of an automatically synthesized controller. As such, it is to be used as an additional testing facility in conjunction with ReqV.

e) Available Resources

Website:

- http://www.sagelab.it/hydra/ (HyDRA)
- http://www.sagelab.it/reqv/ (ReqV)

Source code and documentation:

- https://gitlab.sagelab.it/sage/hydra (HyDRA)
- https://gitlab.sagelab.it/sage/ReqV (ReqV)
- https://gitlab.sagelab.it/sage/SpecPro (SpecPro)

Tutorials:

- https://youtu.be/2WKSxh64Z2k (ReqV video tutorial)

Relevant Publications:

- [Bit-Monnot 2019] (HyDRA)
- [Bit-Monnot 2019b] (HyDRA)
- [Narizzano 2018] (ReqV, SpecPro)
- [Narizzano 2019] (ReqV, SpecPro)
- [Vuotto 2019] (ReqV)
- [Vuotto 2019b] (SpecPro)

2.3. DynAA

a) Starting Point Summary

At the starting point of M15 the version of DynAA was able to provide simulations that could be used for prediction calculation of itineraries. At this purpose, the existing simulation and search space functionalities of DynAA were used. To allow more flexibility and to further adapt the DynAA tool, the Matlab based parts of the core were re-implemented using Java in the first year of CERBERO.

b) New Features and Extensions

One of the new features included in DynAA after M15 has been the parallel processing functionality. This will enable DynAA to execute simulations in parallel instances and collect the results centrally after completion of the simulations. The parallelization function will especially be relevant in case large simulations need to be executed and minimizing the response time is important. This is also the case for predictions in the ST UC, where the driver needs near real time feedback on best itineraries to drive. In case number of itineraries to calculate will require too much processing time, DynAA would be able to parallelize the computation on multiple HardWare (HW) and thereby reduce the overall response time (see Figure 4).

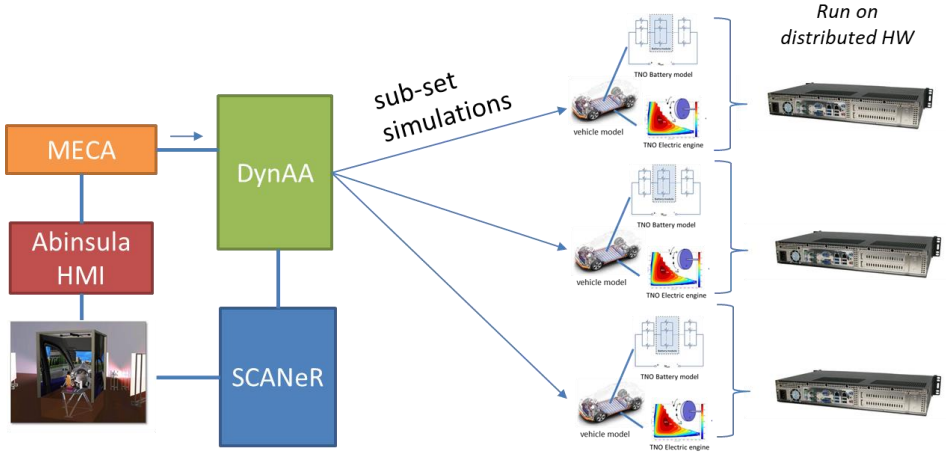


Figure 4 – Parallel execution of DynAA simulations.

Also a feature for system in the loop simulation has been added. It is now possible to run DynAA simulations in parallel with the SCANeR driving simulator. In this way, the battery and motor models are much more flexible: the hard code linked as library interface with SCANeR has now been replaced with a running DynAA instance adopting the message

interface of SCANeR (see Figure 5). So that, CRF will be able to manipulate the running models without directly communicating with SCANeR.

Finally the simulation runs in DynAA have been adapted to provide new information used by MECA to compare battery state of charge during the trip with predicted values.

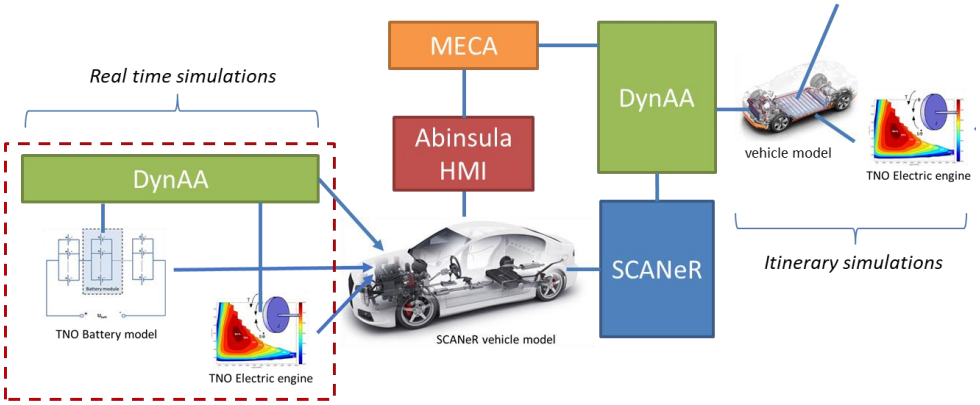


Figure 5 – Real time system in the loop simulation via DynAA.

c) Differences with Respect to the Plan

In CERBERO, we have explored the possibility of parallelizing the Design Space Exploration (DSE) used in DynAA. In this approach, the space to be explored (usually a large set of parameters) is divided in smaller segments, and simulations for each segment are executed in parallel. That, combined with smart space exploration methods, will boost the DSE capabilities and allows it to be used also in run-time environments. The planned activities in this sense have been all completed.

In CERBERO, system in the loop simulation capabilities have been developed for DynAA in order to run simulations in real time. Such feature will make DynAA a more valuable tool during the design of CPSs, especially when simulations of components under development have to be carried out with parts of the system that already exist. The new way of interfacing towards the SCANeR environment via DynAA system in the loop will also give CRF much more flexibility to manipulate the simulation during execution. Also in this case the planned activities have been concludes successfully.

d) Relevance with respect to the CERBERO framework and to UCs

Being one of the system level design tools, DynAA is adopted to model, analyze, and simulate aspects of the CPSs under design in early stages of the development. DynAA makes it possible to have an early evaluation of the system Key Performance Indicators (KPIs), thus helping decision making process.

Smart Travelling: The main involvement of DynAA in CERBERO UCs is on the ST one, where a smart route planning is implemented in strict interaction with the internal state of the car and the battery charging grid infrastructure. Here DynAA contributes to:

- Run-time solution space exploration: the core application of ST UC is a predictive model to estimate energy consumption to complete a route, also considering re-charging stops. In this context, DynAA mainly deal with simulations to predict

energy consumption and battery lifetime KPIs. The new parallelism features implemented on DynAA after M15 should let it possible to reduce the time needed for predictions.

- System in the loop simulations: also the electric power train of the car is going to be simulated in the ST UC. Thanks to the new system in the loop mechanism introduced in DynAA, models of battery and the electric engine can be simulated while running the rest of the system, providing a run-time, direct interaction with signals that come from the car.

Ocean Monitoring: DynAA will also be involved in the Ocean Monitoring (OM) UC, in order to:

- simulate complex camera models to optimize, interacting with AOW, different system KPIs (such as response time, image quality or throughput);
- predictive models to estimate how long the battery will last during mission in order to trigger adaptation accordingly.

Thanks to the new DynAA features developed after M15, these tasks can be implemented faster, leveraging on simulations parallelization, and at run time, leveraging on system in the loop capabilities.

e) Available Resources

Website:

- on-going (not yet available).

Download:

- on-going (not yet available).

Tutorials:

- https://hackmd.io/s/BkSVvHGjZ#
- https://github.com/DynAASim/dynaa-tutorial-compressandsend.git

Relevant Publications:

- [Filho 2013]

2.4. AOW

a) Starting Point Summary

AOW is developed in two main directions: building Separated Continuous-time Linear Programming (SCLP) optimization engine, and serving as a backend optimization tool in CERBERO toolchain. At M15 we had a Matlab code implementing SCLP algorithm with limited scalability of only few controls and states. At M15 we had also identified opportunity for applying AOW for backend optimization in PREESM and MDC tools.

b) New Features and Extensions

Different new features and extensions have been brought on AOW:

- building SCLP optimization engine: the first Python-based version has been developed with improved scalability of a factor of 100-1000x to the base-line Matlab version. For a class of problems, it provides 10-100x compared to the state of the art optimization engines, such as Cplex [Cplex]. This version will be open sourced on Q3 2019 with Apache 2.0 license and a paper will be submitted to a leading Operations Research journal at Q3 2019.
- We have started development of the theory for Robust SCLP; a paper is expected to be submitted to a leading Operations Research journal at Q4 2019.
- Initial HW/SW co-design algebra has been developed using Mixed Integer Linear Programming (MILP) and Linear Programming (LP) models with a successful PoC to allow AOW usage for the backend DSE engine in PREESM. A paper will be submitted to a leading design automation conference or journal at Q3 2019.
- Initial algebra for HW adaptation related to DSE in MDC. Connection through CIF has been enabled, and a PoC is in progress at M30. The integration between AOW and MDC will be reported in final WP5 integration deliverable (D5.1) due at M32). Depending on PoC results it is highly likely that a paper will be submitted at Q4 2019

c) Differences with Respect to the Plan

Based on the CERBERO needs and development status, instead of developing theory and PoC for Mixed Integer SCLP, AOW has been extended to support HW adaptation. The rest is according to the plan described in D5.6.

d) Relevance with respect to the CERBERO framework and to UCs

AOW integrated with PREESM and MDC will support UCs according to their participation for HW/SW co-design and HW adaptation, respectively. Usage of open-sourced SCLP engine for optimization problems in ST and OM UCs is under evaluation.

e) Available Resources

Website:

- on-going (https://github.com/sclp-solvers/sclp-solver/)

Download:

- on-going (not yet available).

Tutorials:

- on-going (not yet available).

Relevant Publications:

- [Shindin-Weiss, 2018]
- [Shindin-Weiss, 2015]
- [Shindin-Weiss, 2014]
- [Weiss, 2008]

2.5. PREESM

a) Starting Point Summary

PREESM at M15 was at version v2.9.0. PREESM was supporting the Parameterized and interfaced Synchronous DataFlow (PiSDF) Model of Computation (MoC), and generating scheduled code for static PiSDF. The code generator for dynamic PiSDF was also functional for the run-time as it was at M15.

b) New Features and Extensions

At M30, PREESM has received 32 releases (including one major) and it is at version v3.7.0. Numerous issues have been solved, and contributions from external partners (UPM) have been successfully integrated. The changes impact on the robustness of the tool, as a long run maintenance work and including documentation effort, but also model evolutions (MoC, Model of Architecture (MoA), scenario) and integration effort with other tools. The contributions noteworthy in the context of CERBERO, excluding integration ones, are summarized as follows:

- add support for delay initialization;
- add support for periodic actors, in the context of real-time scheduling;
- partial support for distributed memory communications in SLAM;
- complete rewrite of the instrumentation library;

- improve user friendliness through various fixes and additions in UI, error reporting and logging, documentation (including tutorials) and algorithm code;
- improve tool robustness through numerous fixes;
- improve support for MacOS and Windows;

Complete list of changes is available in the release notes of the PREESM project[1]. Document additions are available on the PREESM website[2].

c) Differences with Respect to the Plan

Out of the 4 expected extensions, 3 are integration plans with partners tools. They are the direct connections with (1) lower level tools (MDC, PAPIFY, etc.); (2) higher level tools; and AOW for multi-objective optimization. They are discussed in detail in D5.5.

The last one refers to the MoC extensions presented in D3.5. The extensions are:

- delay initialization (section 5.1 of D3.5);
- non-functional parameters (section 5.2 of D3.5);
- moldable parameters (section 5.3 of D3.5).

Among the presented envisioned MoC extensions, the delay initialization has been successfully implemented and led to a publication [Arrestier 2018]. The support for periodic actors directly fits under the non-functional parameter support within the MoC. The moldable parameters, have no current implementation. Since this contribution was not considered as essential for supporting the CERBERO use cases, it has been assigned a lower priority. Nevertheless, its development is still planned for future months in the PREESM roadmap, in parallel and possibly after of CERBERO's timeline.

d) Relevance with respect to the CERBERO framework and to UCs

A lot of changes and new features impact the robustness of the tool and its user friendliness. Although it does not contribute scientifically to the project, the tool visibility is greatly increased thanks to them. Moreover, the documentation and maintenance efforts have been indirectly easing the integration with partner tools.

The elucidation of the delay initialization semantic is of great relevance for the PE UC. Indeed, the delay initialization solves a conceptual hole of the PiSDF MoC that prevented the proper reading of the robotic arm initial state.

Periodic actors are tailored for real time constraints. Therefore, they are of complete relevance for all UCs requiring tasks to be executed in a strict periodic timeline. Such periodic tasks could include the monitoring of the car and/or driver in the ST UC, or the update of the sensors values for the robotic arm of PE UC. Indeed, the schedulability of applications designed with periodic actors can be easily checked according to given real time constraints.

[1] https://github.com/preesm/preesm/blob/v3.7.0/release_notes.md
[2] https://preesm.github.io/docs/devdoc/ and https://preesm.github.io/docs/workflowtasksref/

e) Available Resources

Website:

- http://preesm.org

Download:

- https://preesm.github.io/get/

Tutorials:

- https://preesm.github.io/tutos/

Relevant Publications:

- [Arrestier2018]
- [Pelcat 2014]

2.6. SPIDER

a) Starting Point Summary

SPIDER at M15 was at version v0.6.1, and had not received any new release for the past 2 years. The run-time at this time was supporting the same dynamic PiSDF that PREESM was supporting at M15, without any of the extensions presented in Section 2.5. On top of that, the run-time was not aware of architecture heterogeneity.

b) New Features and Extensions

At M30, SPIDER has received 9 releases and is at version v1.4.0. Numerous fixes and new features from INSA and UPM have been integrated. As for PREESM, many changes impact robustness and were done in order to integrate SPIDER with partner tools (mostly PAPIFY). Algorithms and run-time platform have been re-written to support asynchronous communication, resulting in performance improvements. Among all the changes, the most noteworthy for the CERBERO project are the following:

- add support for delay in hierarchical graphs;
- add support for delay initialization;
- add support for heterogeneous HW, with the cooperation of UNISS and UPM;
- add new online greedy scheduler;
- re-write algorithms (BRV, scheduler, communication router) for better performance;
- improve tool robustness through numerous fixes and code cleaning;
- improve support for MacOS and Windows.

Complete list of changes is available in the release notes of SPIDER [3].

[3] https://github.com/preesm/spider/blob/master/release_notes.md

c) Differences with Respect to the Plan

Out of the 5 expected extensions, 3 are integration plans with partner tools. They are (1) connection with PAPIFY; (2) connection with MDC and ARTICo³; (3) and connection with CIF. PAPIFY and MDC connections have been described in D5.5, and further details on other integration efforts will be included in D5.1.

Envisioned in D5.6, the support for real-time extension, that is the periodic actors added in PREESM PiSDF MoC, is not yet provided by SPIDER at M30. This feature has been identified as non-essential to CERBERO UCs since the writing of D5.6, so that development and integration of other SPIDER features have been prioritized.

d) Relevance with respect to the CERBERO framework and to UCs

The improved support for Mac OS and Windows, as well as the improved tool robustness, do not directly contribute the scientific production of the CERBERO project. However, they should be mentioned since they contributed to boost the tool visibility and ease partner tool integration.

The extensions on the PiSDF MoC, when integrated in the SPIDER runtime, are as relevant as they are for PREESM. Such is, for instance, the elucidation of delays initialization.

The support for heterogeneous HW in SPIDER eases deployment of dynamic PiSDF applications on the heterogeneous HW platforms of partners, such ARTICo³ and MDC-compliant platforms.

Finally, the re-written algorithms reduce the run-time overhead of SPIDER. This speeds up the reconfiguration time, reduces energy footprint, and frees computing resources faster. On top of better overall performance, the new algorithms avoid re-configuring the system at every iteration of the graph. This is of complete relevance for all dynamic applications needing fast response time to environment change, or with limited compute/energy resources.

e) Available Resources

Website:

- http://preesm.org

Download:

- https://preesm.github.io/get/

Tutorials:

- https://preesm.github.io/tutos/

Relevant Publications:

- [Arrestier2018]
- [Heulot 2014]

2.7. PAPIFY/PAPIFY VIEWER

a) Starting Point Summary

PAPIFY and PAPIFY-VIEWER are tools aiming at easing the code instrumentation and the visualization of Performance Monitoring Counters (PMCs) at run-time. This tool, at M15, was able to instrument applications on homogeneous target platforms, i.e., only one type of Processing Units (PUs) (SW cores or HW accelerators) could be monitored.

b) New Features and Extensions

To support run-time monitoring for heterogeneous architectures, PAPIFY has been extended to include monitoring information for PUs of different nature. By doing so, the user is able to set up a monitoring for a specific actor and, on run-time, PAPIFY will select the suitable monitoring configuration depending on the resource that performs the execution.

Additionally, the library supporting PAPIFY, the so-called *eventLib* library, has been optimized in two different ways:

- Firstly, the memory footprint of the library has been minimized to be able to use it when the architecture has tight memory constraints.
- Secondly, the library has been modified so as to be thread-safe. This requirement appears due to the possibility of having several threads running on the same PU. Specifically, these threads would need to share information for the monitoring configuration but, on the other hand, the monitoring information must be isolated depending on the thread executing each actor.

c) Differences with Respect to the Plan

With respect to the planned activities:

- A methodology to estimate energy KPI models based on the monitoring information retrieved by PAPIFY has been developed. Specifically, it has been developed using as a target platform a many-core architecture called MPPA-256-N [Mppa]. This methodology will be applied to extract the energy consumption model for the ARM cores that will be used in the PE UC.

d) Relevance with respect to the CERBERO framework and to UCs

The relevance of PAPIFY and PAPIFY VIEWER with respect to CERBERO framework and UCs has nothing changed since M15, so that please refer uniquely to D5.6 in this case.

e) Available Resources

Website:

- https://gitlab.citsem.upm.es/papify/papify

Download:

- https://gitlab.citsem.upm.es/papify/papify

Tutorials:

- https://preesm.github.io/tutos/papify/

Relevant Publications:

- [Madroñal 2018]

2.8. IMPRESS

a) Starting Point Summary

At M15, a first version of IMPRESS [Zamacola 2018] had been developed. IMPRESS, previously referred to as JIT HW in D5.6, is a reconfiguration tool for Just-In-Time (JIT) HW composition. It extends the reconfiguration capabilities provided by Vivado and better adapts them to the reconfiguration needs of CERBERO reconfigurable fabrics. The following features were included in the previous version of IMPRESS: module relocation, sub-clock reconfiguration, direct reconfigurable-to-reconfigurable communications, flexible virtual architectures and a decoupled implementation of the static and reconfigurable systems.

b) New Features and Extensions

IMPRESS

During M15 to M30 IMPRESS [Zamacola 2018] has been stabilized and extended with the following features:

- *Design-time support for look-up table (LUT) reconfiguration:* One of the main disadvantages of DPR compared to virtual reconfiguration is its elevated reconfiguration times. However, it is possible to fine-tune a circuit in a fast way using DPR. This can be done by reconfiguring specific elements of the FPGA (e.g., look-up tables (LUTs), flip-flop contents or net paths) instead of reconfiguring an entire area of the FPGA. IMPRESS has been extended to support the reconfiguration of LUTs.
  LUT-based functional units (FUs), constants and multiplexers are included as parameterized Hardware Description Language (HDL) components that the users can instantiate in their design to adapt the behavior of the FU, change a hardwired parameter or change the datapath of a module.

The minimum reconfiguration element of a Xilinx FPGA is a frame which spans a whole clock region column. Therefore, when reconfiguring one LUT all the LUTs of the same column are also reconfigured. In order to improve reconfiguration times, IMPRESS automatically includes placement constraints to stack reconfigurable LUTs in the same column, so that all the LUTs are reconfigured at the same time.

- *Run-time support for LUT reconfiguration:* In order to speed-up the reconfiguration of LUT-based components, a specialized reconfiguration engine optimized for fast reconfiguration of LUTs has been designed. The reconfiguration engine is implemented in the FPGA and is connected to the processor using an AXI-lite protocol. To initiate the reconfiguration, the processor sends the column to reconfigure, the element type, and a compressed partial bitstream that is then decompressed by the reconfiguration engine. The reconfiguration engine performance is increased by avoiding a read back step and by receiving a compressed partial bitstream from the processor (i.e., to reduce the communication overhead).

- *Run-time support for composing at run-time 2D architectures overlaid in a grid-based virtual architecture:* IMPRESS has also been extended with a library that contains APIs to dynamically build 2D architectures (i.e., overlays) with modules that contain LUT-based components. The first step to build an overlay is to define a library of reconfigurable modules that have been previously implemented with IMPRESS. Then the user may use reconfiguration to stitch together at run-time different modules to compose a custom overlay. Once the overlay is built, it is possible to reconfigure the LUT-based components to quickly adapt the overlay configuration to new conditions. Figure 6 shows the main components of this process. For a more in-depth explanation of how this process work please refer to D4.1.

In order to show the capabilities of IMPRESS for generating overlays at run-time, two different applications have been developed. The first one uses an evolutionary approach for generating a circuit, while the second one uses a deterministic approach to build accelerators from SW descriptions.

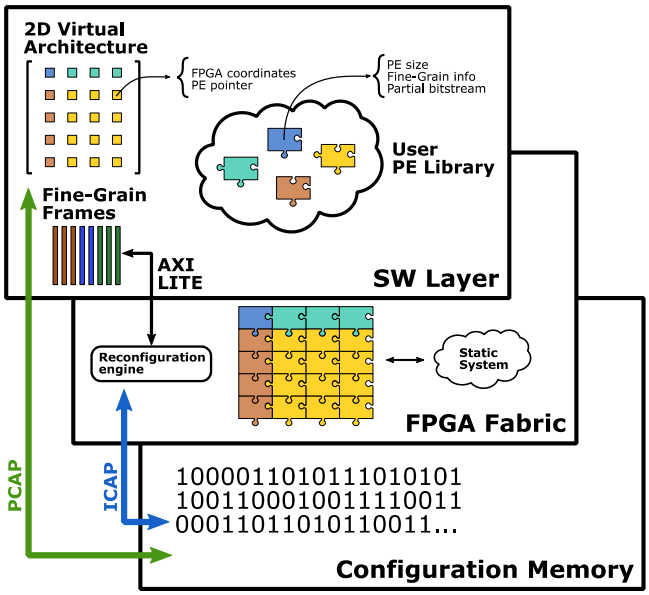


Figure 6 - Run-time support for composing at run-time 2D architectures.

Iterative/evolutionary just-in-time HW composition

From M15 to M30, the evolutionary just-in-time HW composition approach has been developed. A block-based neural network (BbNN) has been selected as the architecture to implement the evolutionary approach. This architecture benefits from the possibilities provided by IMPRESS tool. In contrast to deep neural networks, where each layer of cells is fully connected to all the cells of the previous layer, a BbNN is composed of blocks that are connected to some of their neighbors. This makes BbNNs very well suited to be composed using partial reconfiguration. Moreover, while deep neural networks have a fixed structure of cells, in a BnNN each block of the network can adopt a different configuration. Each configuration can implement a different number of cells (1 to 3 cells) and each of those cells can be connected with one of their block neighbors. Therefore, the network is trained not only to find the bias and weight values of each cell, but also the configuration and connectivity of each block.

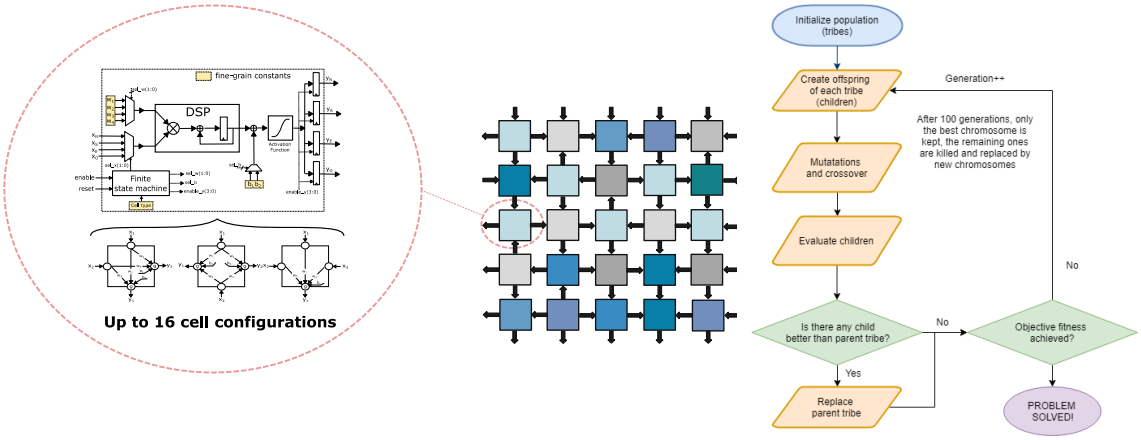
As said before, the BbNN is built using the reconfiguration capabilities provided by IMPRESS. Each block can be stitched together using partial reconfiguration. This allows changing the size of the BbNN at run-time, scaling its size when needed. Each block of the BbNN contains LUT-based constants that define the cell configuration and the weight and bias parameters of each cell of the block. That way module reconfiguration is only carried out when it is needed to change the size of the network, while LUT-based reconfiguration can be used to quickly adapt the parameters of the network. Figure 7.a shows how the BbNN has been implemented.

The evolutionary algorithm shown in Figure 7.b has been implemented to train the BbNN. The training phase needs to evaluate multiple BbNN configurations which makes the fast LUT-based reconfiguration really useful in this application. One of the main advantages to train the BbNN with an evolutionary algorithm is that the architecture used for the training and the inference phase is the same, making it possible to train and use the BbNN in the

same device. This allows life-long learning application where the CPS needs to adapt itself to new requirements and thus needs to be trained multiple times in its lifetime.

Deterministic just-in-time HW composition

The objective of the deterministic just-in-time HW composition is to generate accelerators from dataflow graph (DFG) descriptions obtained from SW specifications, by stitching together different modules previously implemented. This way, it is possible to generate a library of pre-implemented modules that can then be combined on the fly to compose different accelerators. A first version of this application has been developed that shows how IMPRESS can be used for either building a 2D overlay using coarse dynamic partial reconfiguration or fine-tune an existing overlay by using LUT-based dynamic partial reconfiguration.



a) b)

Figure 7 - a) BbNN structure b) evolutionary algorithm.

The deterministic JIT HW composition application can be divided into three different sections. The first one is obtaining a data flow graph from a SW description. The idea is to convert the source code into LLVM [Lattner 2004] intermediate representation and from there obtaining a data flow graph. The CGRA-ME [Chin 2017] tool already allows to obtain a DFG from a for loop. This tool has been extended to convert if instructions to select statements (which can be implemented with multiplexers) and to obtain acyclic data flow graphs.

The second step is to define an overlay structure and map the DFG into the overlay. The overlay that has been selected to implement the DFG is based on the one proposed in [Capalija 2014]. The mapping and routing of the DFG into the overlay is carried out using the CGRA-ME tool.

The third step is to reconfigure the overlay using the mapping and routing results obtained in the previous step. IMPRESS coarse-grain reconfiguration can be used to change the functional units of the overlay while the LUT-based reconfiguration can be used to change the routing and constants of the overlay. Right now, this step needs to be performed manually by the user and there is no connection between the processor and the overlay.

However, despite these limitations, this first version of the application shows how IMPRESS can be used for deterministic just-in-time composition of HW.

c) Differences with Respect to the Plan

The main difference with respect to the original plan is that a new application for JIT HW composition based on an evolutionary approach has been developed using IMPRESS. Moreover, this application has evolved over the course of the project. In the first deliverables, the idea was to generate an overlay and use an evolutionary algorithm to change the functional units of the overlay. However, the final implementation uses a BbNN that has fixed functional units, but that can be parameterized to change its structure and computing parameters. This allows faster training as only the LUT-based components are reconfigured instead of reconfiguring whole functional units.

d) Relevance with respect to the CERBERO framework and to UCs

IMPRESS, the low-level reconfiguration tool, is going to be integrated with ARTICo3 to improve its reconfiguration capabilities adding features as bitstream relocation, using flexible virtual architectures and decoupling the design of the static design and reconfigurable accelerators.

As IMPRESS and the JIT HW composition applications have been developed from scratch in CERBERO they are not as mature as other tools from the project. Therefore, they will not be integrated in the UCs and they will instead be validated using an application as a PoC. The iterative approach will be used to generate an adaptable controller of a CPS system that consists on a simulated cart pole obtained from the open AI gym toolkit [Gym]. The demonstrator will show how the BbNN can be used for lifelong learning applications. Once the BbNN finds a valid controller for the cart pole, the dynamic of the system can be changed (e.g., the size of the pole changes) if the previous controller is no longer valid, the device can re-train itself to find a new valid controller.

e) Available Resources

Website:

- on-going (not yet available).

Download:

- on-going (not yet available).

Tutorials:

- on-going (not yet available).

Relevant publications:

- [Zamacola 2018]
- [Zamacola 2019]

2.9. ARTICo³

a) Starting Point Summary

ARTICo^3 is a reconfigurable multi-accelerator processing architecture that uses DPR in FPGAs to offload data-parallel computations into a configurable number of HW accelerators. A script-based toolchain complements the architecture and automates the generation of ARTICo^3-based systems, whereas a run-time library provides a transparent way of managing both DPR and scalable parallelism.

b) New Features and Extensions

The main extensions that have been achieved from M15 to M30 in the ARTICo^3 framework can be summarized as follows:

- changes in the run-time library to support shared memory communications between high-level and heterogeneous (i.e. HW/SW) dataflow actors;
- changes in the accelerator templates and in the run-time library to support coarse-grain and LUT-based reconfiguration inside each HW accelerator.

ARTICo^3 uses shared memory buffers between user applications and the HW accelerators placed in the FPGA. The execution model used before M15 assumed a traditional processor/coprocessor approach, where applications running in the host processor explicitly allocated a shared buffer using the ARTICo^3 run-time API. To enable support for a dataflow-oriented execution model, where memory buffers can be pre-allocated by other entities in the system, the run-time API has been extended to either allocate a shared memory buffer or to directly use an already existing one. At M15, the ARTICo^3 architecture used DPR as the only reconfiguration mechanism. Moreover, it was used to change large regions in the FPGA fabric (i.e. the reconfigurable slots where the HW accelerators are placed). Two additional reconfiguration mechanisms have been added to the ARTICo^3 framework: register-based coarse-grain reconfiguration and LUT-based DPR-enabled reconfiguration. To support the former, the structure of the register banks in the ARTICo^3 kernel wrapper has been modified, and the run-time library has been extended with read/write functions to access those registers. To support the latter, on the other hand, an additional architectural template with a systolic array within an ARTICo^3 kernel wrapper has been developed, where a new reconfiguration engine is used to change the LUT contents in each processing element, thus altering the functionality of the whole HW accelerator.

c) Differences with Respect to the Plan

Regarding the original planning of activities:

- The back-ends for either high- and low-level dataflow descriptions are already available in the ARTICo^3 framework. The automated design flow from high-level dataflow descriptions down to reconfigurable HW accelerators (e.g., using High Level Synthesis (HLS) from C code) has not been implemented yet. However,

transparent usage of input code is available for developers (e.g., the input specification in C code adopted in PREESM can be also used as input for HLS-based generation of ARTICo3 accelerators, even if both tools are used independently).

- The enhanced register banks available in the ARTICo^3 kernels represents, together with the already existing monitoring infrastructure (i.e., PAPI component and architecture-level monitoring counters), the foundation of a unified SW/HW (self-)monitoring infrastructure for ARTICo^3 HW accelerators. The final version of the PAPI component for the multi-accelerator architecture is under development and will be finished at the end of the project. It will be described more in detail in D5.1 since it is part of the integration activities between PAPIFY and hardware acceleration tools, ARTICo^3 and MDC.
- Intra-accelerator reconfiguration was originally conceived to be implemented in a register-based coarse-grain approach. The new accelerator template and run-time extensions have enabled this type of reconfiguration. In addition, and in an attempt to lay the initial foundation for JIT-HW composition within ARTICo^3 accelerators, an originally unplanned reconfiguration mechanism (i.e., LUT-based) has also been implemented.

d) Relevance with respect to the CERBERO framework and to UCs

At M15, the ARTICo^3 framework provided standalone adaptation fabrics and engines, and the interface for users to implement their own manager (i.e., the ARTICo^3 run-time library). The extensions developed from M15 to M30 enable the cross-layer integration of technologies and tools within the CERBERO framework:

- adaptation Fabric/Engine: seamless integration with register-based coarse-grain reconfiguration (i.e., MDC) and LUT-based fine-grain reconfiguration (i.e., JIT-HW Composition);
- adaptation Manager: structural integration with external entities that make high-level decisions on resource distribution and allocation (i.e., SPIDER).

On the UC side, the adaptivity surrounding ARTICo^3 can be used not only to provide functional (i.e., algorithmic) but also non-functional adaptation (i.e., energy efficiency, fault tolerance). These features increase the potential applicability of ARTICo^3 in sensitive scenarios such as the one depicted in the PE UC, where stringent requirements (e.g., changing performance needs, different radiation levels) are common.

e) Available Resources

Website:

- on-going (not yet available).

Download:

- on-going (not yet available).

Tutorials:

- on-going (not yet available).

Relevant publications:

- [Rodriguez 2018]
- [Rodriguez 2016]
- [Ortiz 2018]
- [Rodriguez 2019]

2.10. MDC

a) Starting Point Summary

The Multi-Dataflow Composer (MDC) is a tool capable of generating Coarse-Grain Reconfigurable (CGR) HW datapaths starting from a set of applications described by dataflow models. Moreover, it offers additional features like structural profiling, to optimize the combination of the dataflow; dynamic power manager, to minimize power consumption; co-processor generator, to deliver fast and easy system integration.

b) New Features and Extensions

To provide self-containing HW, MDC requires the HW code corresponding to the dataflow actors and the HW communication protocol adopted by them. At M15 it was already possible to derive the HW code of such actors through any source, either manual or HLS powered, and to let MDC integrate it through a generalized XML protocol specification (see D4.4). Such generalized protocol specification was not linked with some of the additional features of the tool, such as the co-processor generator. In order to reach full automation, from applications specification to the ready-to-use CGR Xilinx IPs, it is now possible to specify on the XML HW communication protocol the mapping of each signal on a FIFO-like protocol used by the IP wrapper (see Figure 8). In such a way, the internal CGR datapath will successfully communicate with the wrapper and the rest of the system, definitely closing the design automation path.

```
<wrapper>
    <comm_signals>
        <signal id="0" channel="data" size="variable" mapping="data"></signal>
        <signal id="1" channel="rd" size="1" mapping="pop"></signal>
        <signal id="2" channel="wr" size="1" mapping="push"></signal>
        <signal id="3" channel="empty" size="1" mapping="empty"></signal>
        <signal id="4" channel="full" size="1" mapping="full"></signal>
    </comm_signals>
</wrapper>
```

Figure 8 – Generalized HW communication protocol for linking the CGR datapath with the accelerator Xilinx IP wrapper.

The co-processor generator has been updated and improved going from M15 to M30. Firstly, it now supports Vivado environment (EDK was supported before) and its powerful block design editor. Moreover, besides the level of coupling with the host processor, that is the communication link to be used between memory mapped (AMBA AXI4-full) and stream (AMBA AXI4-stream), additional knobs are available to the user: the host processor can be selected between a MicroBlaze soft-core or an ARM hard-core, while it is also possible to specify if DMA engines have to be used for data moving or not. The HW code, the Xilinx IP and block design support files, and the accelerator drivers are generated taking into account the specific choices made by the user in terms of coupling, host processor and DMA usage. Finally, in order to go one step further in design automation, the co-processor generator has also been made capable of generating scripts to automate the Xilinx Vivado projects building for the accelerator IP and for the integrated system involving host processor, communication infrastructure and accelerator. An overview of the new co-processor generator is depicted in Figure 9.

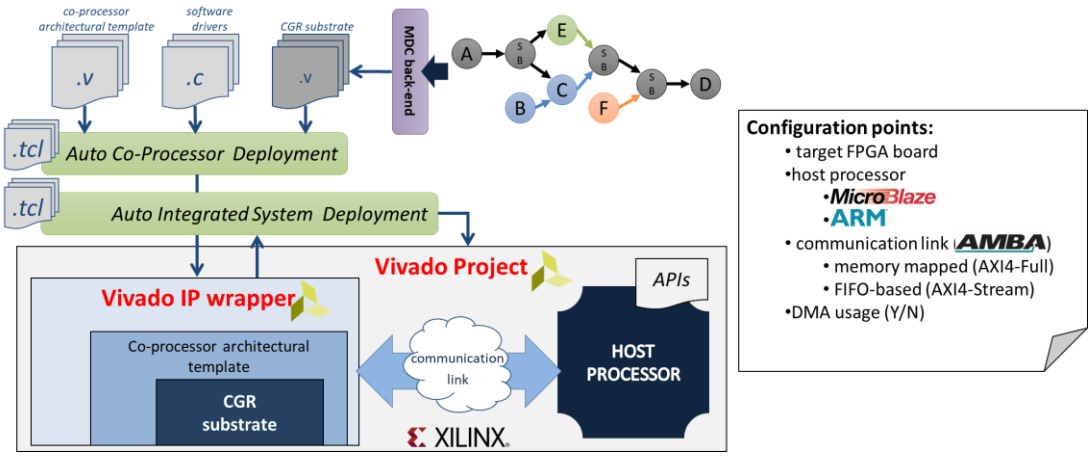


Figure 9 – New version of the MDC coprocessor generator.

MDC HW code generation has been also modified to support accelerator-level monitoring insertion (see D5.5). If required by the user, some registers in the resulting accelerator will be reserved to keep trace of important metrics during execution, such as the execution time (clock cycles), the number of input tokens and the number of output tokens. At M30 however, low-level monitoring, that is monitoring internal to the CGR datapath, has not been integrated within the code generation flow.

c) Differences with Respect to the Plan

With respect to the planned activities:

- Full automation from dataflow to HW has been successfully reached through the integration between MDC and the CAPH HLS engine. Moreover, thanks to the HW communication protocol generalization, it is now possible to use MDC with any other HLS engine, if the user models actors in the HLS engine input specification format, and dataflow graphs with the MDC supported one, that is XDF (RVC-CAL).

- HW/SW partitioning leveraging on PREESM has not been refined, since MDC is not yet supporting PiSDF models. In order to let PREESM take decisions, a model of the accelerators, and before that, a way of modelling architectures in an homogenous way, was needed. We have been working on defining a MoA allowing early stage latency estimation. The MoA has been derived for multi-core architectures and the path to derive it also for heterogeneous ones, where MDC-compliant CGR accelerators are used, is on-going (see Figure 10). In any case this cannot be considered a deviation of MDC plan, since integration related activities will be completed by M32 and we do not expect further changes at MDC level.

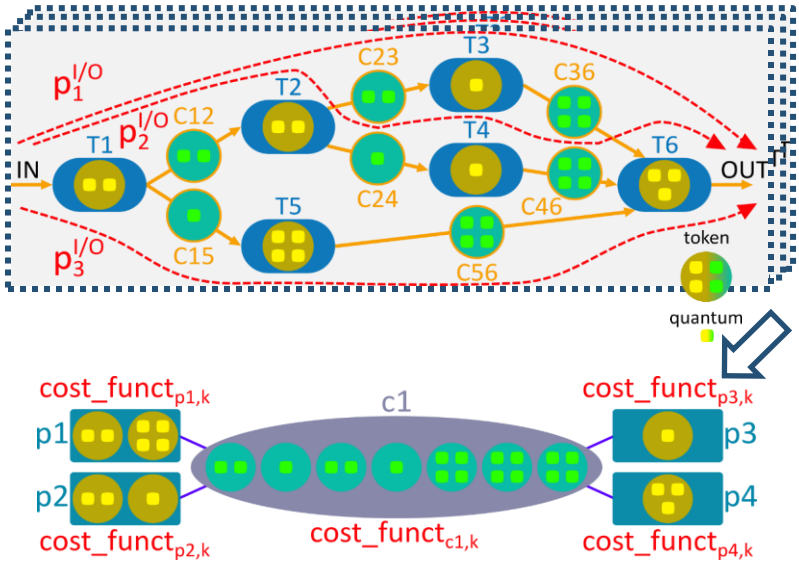


Figure 10 – MoA for latency metric to be used with MDC-compliant CG reconfigurable accelerators.

- Instrumenting the MDC generated CGR accelerators with run-time monitors according to the PAPI approach used by PAPIFY and PAPIFY VIEWER has been done manually at both accelerator- and low-level (see D5.5), and a PoC has been provided to effectively monitor data during execution. From the automation point of view only accelerator-level monitor instrumentation has been automated in MDC, low-level one will be finalized as part of integration activities before completion of WP5 at M32.
- The co-processor generator update and extension was not initially foreseen in the plan but it has been necessary due to the Xilinx tools (Vivado and block design alignment) and devices (host processor and DMA improvement) evolution.

d) Relevance with respect to the CERBERO framework and to UCs

The MDC co-processor generator update made the tool ready for the integration with ARTICo^3 (D5.7). This update, together with the MDC full automation, made MDC close to be ready for the HW/SW co-design and, in turn, for the integration with PREESM, SPIDER and the whole design time and run-time adaptation loop. In fact, being fully automated and providing advanced system integration, MDC features can be adopted at design time by PREESM in order to identify which parts of the application(s) can be more

successfully accelerated in HW; then, at run-time, SPIDER can dynamically manage the MDC generated CGR accelerator by changing its configuration or by switching it off if only-SW execution is preferable at a certain point (e.g. if faults occur on HW due to radiations in the PE UC). An overview of the resulting PREESM/SPIDER and MDC integrated system is depicted in Figure 11.

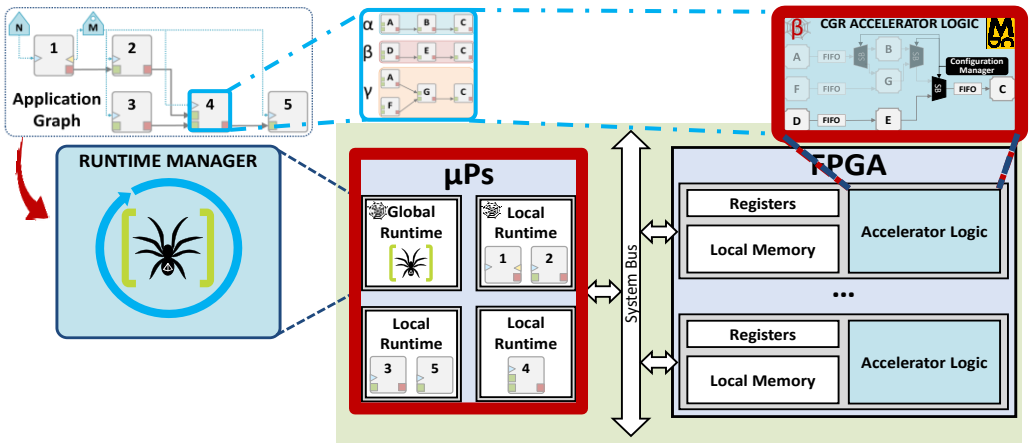


Figure 11 – Overview of the final HW/SW system with the integration of PREESM/SPIDER and MDC.

In January 2019 we successfully integrated and demonstrated dynamic tasks re-mapping among HW and SW using SPIDER running on the FPGA. By completing the work on the MoA and using the information gathered from PAPIFY through the HW monitors supported by MDC, the adaptation loop will be completely closed. We expect to deliver a small demonstrator of the closed loop as part of the final project demonstration activities using the PE as reference scenario.

e) Available Resources

Website:

- http://sites.unica.it/rpct/

Download:

- on-going (not yet available)

Tutorials:

- https://www.youtube.com/watch?v=_cyYFJCDR3U&list=PLql1YxTzHalZztJPu7wn0uzAYbr81QTpH

Relevant Publications:

- [Rubattu 2018]
- [Sau 2017]
- [Palumbo 2017]
- [Sau 2016]
- [Sau 2015]

3. Conclusions

This document provided an overview of the progress related to the main components/tools of the CERBERO framework. The evolution of each component/tool during the project has been described in detail, together with the impact of such evolution on the framework itself and on the CERBERO UCs. New unforeseen developments and deviations with respect to the original activities have also been highlighted. Please note that none of the deviations is on critical activities that may prevent successful demonstration of technologies. More mature tools/components, and their integration, are expected to be demonstrated at M36 within UCs (e.g. the whole computational-level self-adaptation framework, involving PREESM, SPIDER, PAPIFY, ARTICo^3 and MDC, is going to be adopted for a demonstrator in the PE UC), while less mature ones will be demonstrated stand-alone (as in the IMPRESS case).

Table 1 Relevant features provided by the CERBERO framework components/tools.

	Modelling	**Optimization**	**HW/SW Design**	**Run-time Support**	**Incremental / Fast Prototyping**	**In Loop Simulation**	**Open Source**
MECA	X			X	X		
SAGE	x				X		X
DynAA	X	X				X	
AOW	X	X	X				X
PREESM	X	X	X		X		X
SPIDER		X		X			X
PAPIFY			X	X			X
IMPRESS		X	X	X			
ARTICo³		X	X	X	X		
MDC		X	X	X	X		

As occurred for D5.6, some resuming tables are provided in order to summarize the current state of the CERBERO framework components/tools. In Table 1 each tool is cross-mapped with respect to the features that are most important for the CERBERO objectives (modelling, optimization, HW/SW design, run-time support, rapid prototyping, in loop simulation and open source). Please note that SAGE (previously VT in D5.6) and IMPRESS (previously JIT HW) are components/tools started from scratch during CERBERO.

Table 2 provides a classification of the CERBERO components/tools in terms of abstraction/application layer (models verification, system (of systems) design/analysis and computation level design/implementation), design-time or run-time usage and cross-

linking them to the CERBERO UCs: Smart Travelling (ST), Planetary Exploration (PE) and Ocean Monitoring (OM). Please note that this Table is different from the corresponding one in D5.6 since it provides additional information, such as the design- or run-time nature of the component/tool, and the level of abstraction has been changed in application layer, that is more descriptive of the component/tool functionality and is aligned to the new overview of the CERBERO framework depicted in Figure 1 and Figure 2.

Table 2 Level of abstraction on which each tool/component works and related relevance with respect to the CERBERO UCs.

	Application Layer	**Design-Time (DT) or Run-Time (RT)**	**UC**		
			ST	**PE**	**OM**
MECA	System (of Systems) Design/Analysis	RT	X		
SAGE	Models Verification	DT	X	X	X
DynAA	System (of Systems) Design/Analysis	DT and RT	X		X
AOW	System (of Systems) Design/Analysis Computation Level Design/Implementation	DT	X	X	X
PREESM	Computation Level Design/Implementation	DT		X	X
SPIDER	Computation Level Design/Implementation	RT		X	X
PAPIFY	Computation Level Design/Implementation	DT and RT		X	
IMPRESS	Computation Level Design/Implementation	DT and RT		X	
ARTICo³	Computation Level Design/Implementation	DT and RT		X	
MDC	Computation Level Design/Implementation	DT and RT		X	

The two resuming tables provide an overview of the CERBERO framework status and how the different components/tools composing it covers the features required to meet the CERBERO project objectives, how they are spread throughout different system layers, in which phases they operate and on which UCs they will be employed. Please note that the CERBERO framework and the involved components/tools are actively contributing to the meeting of the CERBERO operational objectives according to the big challenges faced by the project. As depicted in Table 3, CERBERO framework is involved in the addressing of all the three CERBERO challenges and in six of the related operational objective. The framework as a whole or some of the involved components/tools are specifically contributing to the different objectives and challenges.

Table 3 Contribution of the CERBERO framework contribution to the meeting of the CERBERO operational objectives for each challenge of the project.

CHALLENGE	Operational Objective	CERBERO Framework Contribution
CH1: To drastically reduce energy consumption and improve safety, security and system performance, while guaranteeing both functional and non-functional requirements by a holistic model-based and cross-layer engineering approach	**CH1.2** Provide a comprehensive framework, customizable upon the UC needs, extending and making interoperable a large set of tools.	The CERBERO framework as a whole is addressing such an objective, being used in the different UCs and making 10 different tools interoperable.
	CH1.3 Reduce by 30% the energy consumed by a fully CERBERO compliant CPS or CPSoS, while maintaining its performance.	Different framework components/tools are contributing to meet this objective: the self-adaptivity loop on both systems (of systems) and computational level is capable of adapt according to energy and performance KPIs.
CH2: To reduce time-to-market, development efforts and, in turn the cost of ownership, of CPS and CPSoS.	**CH2.1** Reduce DSE by an order of magnitude.	DSE is enabled by different framework components/tools, such as DynAA, AOW and PREESM. By acting at different abstraction levels, they guarantee fast and smart DSE for complex CPS and CPSoS.
	CH2.2 Reduce by 50% the design efforts required to build a CPS of a given performance.	Design effort is mainly reduced by automation, optimization and fast prototyping features. Such features are all supported, complementary, by the CERBERO framework components/tools.
	CH2.3 Reduce by 50% cost of maintenance.	Design effort is reduced by tools reducing design effort (if new design phases are required) and by run time support, provided by MECA, DynAA, SPIDER, PAPIFY, ARTICo^3, IMPRESS and MDC.
CH3 To proactively contribute to standardization activities and open-innovation initiatives and to influence specific industrial standards	**CH3.1** Provide a fully marketable version of the CERBERO modelling and design environment.	Most of the tools are going to be released open source and they have been or are going to be provided with documentation and training material. Such aspects, together with the production of advertising material for fairs and events of the CPS field, make them almost ready to a market evaluation.

4. References

[Arrestier 2018] Florian Arrestier et al., *Delays and states in dataflow models of computation,* International Conference On Embedded Computer Systems: Architectures, Modeling And Simulation, 2018.

[Bit-Monnot 2019] A. Bit-Monnot et al., *Cyber-Physical Planning: Deliberation for Hybrid Systems with a Continuous Numeric State*, International Conference on Automated Planning and Scheduling, 2019.

[Bit-Monnot 2019b] A. Bit-Monnot et al., *SMT-based Planning for Robots in Smart Factories*, International Conference on Industrial, Engineering & Other Applications of Applied Intelligent Systems, 2019.

[Bosse 2017] T. Bosse et al., *Developing ePartners for Human-Robot Teams in Space based on Ontologies and Formal Abstraction Hierarchies*, International Journal of Agent-Oriented Software Engineering, vol. 5, issue 4, 2017.

[Breebaart 2008] L. Breebaart et al., *The MECA Project – Using an OWL/RDF Knowledge Base to ensure Data Portability for Space Mission Operations*, NASA-ESA Workshop on Product Data Exchange, 2008.

[Breebaart 2009] L. Breebaart et al., *The MECA Project: Ontology-based Data Portability for Space Missions*, IEEE International Conference on Space Mission Challenges for Information Technology, 2009.

[Capalija 2014] D. Capalija and T. S. Abdelrahman, *Tile-based bottom-up compilation of custom mesh-of-functional-units FPGA overlays*, International Conference on Field Programmable Logic and Applications, 2014.

[Chin 2017] S.A. Chin et al., *CGRA-ME: A unified framework for CGRA modelling and exploration,* International Conference on Application-specific Systems, Architectures and Processors, 2017.

[Cplex] Online: www.ibm.com/software/commerce/optimization/cplex-optimizer/

[Filho 2013] J. Oliveira de Filho, Z. Papp, R. Djapic, and J. Oostveen, "Model-based Design of Self-adapting Networked Signal Processing Systems," 2013 IEEE 7th International Conference on Self-Adaptive and Self-Organizing Systems, pp. 41–50, Sep. 2013.

[Gym] Online: gym.openai.com/

[Heulot 2014] J. Heulot et al., *SPIDER: A Synchronous Parameterized and Interfaced Dataflow-Based RTOS for Multicore DSPs*, European Embedded Design in Education and Research Conference, 2014.

[Lattner 2004] C. Lattner et al., *Llvm: A compilation framework for lifelong program analysis & transformation,* International Symposium on

Code Generation and Optimization: Feedback-directed and Runtime Optimization, 2004.

[Madroñal 2018] D. Madroñal, et al., *Automatic Instrumentation of Dataflow Applications using PAPI*, ACM International Conference on Computing Frontiers, 2018.

[Meerincx 2008] M. Neerincx et al., *The Mission Execution Crew Assistant: Improving Human-Machine Team Resilience for Long Duration Missions*, International Astronautical Congress, 2008.

[Mppa] www.kalrayinc.com/portfolio/processors/

[Narizzano 2018] M. Narizzano et al, *Consistency of property specification patterns with boolean and constrained numerical signals*, NASA Formal Methods Symposium, 2018.

[Narizzano 2019] M. Narizzano et al., *Property specification patterns at work: verification and inconsistency explanation*, Innovations in Systems and Software Engineering, pp. 1-17, 2019.

[Ortiz 2018] A. Ortiz et al., *A Runtime-Scalable and Hardware-Accelerated Approach to On-Board Linear Unmixing of Hyperspectral Images*. Remote Sens, vol. 10, 2018.

[Palumbo 2017] F. Palumbo et al., *Power-Awarness in Coarse-Grained Reconfigurable Multi-Functional Architectures: a Dataflow Based Strategy*, Journal of Signal Processing Systems, vol. 87, issue 1, pp. 81-106, 2017.

[Pelcat 2014] M. Pelcat et al., *PREESM: A Dataflow-Based Rapid Prototyping Framework for Simplifying Multicore DSP Programming*, European Embedded Design in Education and Research Conference, 2014.

[Rodriguez 2015] A. Rodríguez et al., *Execution modeling in self-aware FPGA-based architectures for efficient resource management*, International Symposium on Reconfigurable Communication-centric Systems-on-Chip, 2015.

[Rodriguez 2018] A. Rodríguez et al., *FPGA-Based High-Performance Embedded Systems for Adaptive Edge Computing in Cyber-Physical Systems: The ARTICo3 Framework*, Sensors, 2018.

[Rodriguez 2019] A. Rodríguez et al., *Scalable Hardware-Based On-Board Processing for Run-Time Adaptive Lossless Hyperspectral Compression*, IEEE Access, vol. 7, pp. 10644-10652, 2019.

[Rubattu 2018] C. Rubattu et al., *Dataflow-Functional High-Level Synthesis for Coarse-Grained Reconfigurable Accelerators*, IEEE Embedded System Letters, preprint, 2018.

[Sau 2015] C. Sau et al., *Reconfigurable coprocessors synthesis in the MPEG-RVC domain*, International Conference on ReConFigurable Computing and FPGAs, 2015.

[Sau 2016] C. Sau et al., *Automated Design Flow for Multi-Functional Dataflow-Based Platforms*, Journal of Signal Processing Systems, 2016.

[Sau 2017] C. Sau et al., *Challenging the Best HEVC Fractional Pixel FPGA Interpolators With Reconfigurable and Multi-frequency Approximate Computing*, IEEE Embedded System Letters, vol. 9, issue 3, pp. 65-68, 2018.

[Shindin-Weiss 2018] E. Shindin, G. Weiss, *A simplex-type algorithm for continuous linear programs with constant coefficients*, Mathematical Programming (12), 2018,

[Shindin-Weiss 2015] E. Shindin, G. Weiss, *Structure of Solutions for Continuous Linear Programs with Constant Coefficients*, SIAM J on Optimization, 25, pp. 1276-1297, 2015

[Shindin-Weiss 2014] E. Shindin, G. Weiss, *Symmetric Strong Duality for a Class of Continuous Linear Program with Constant Coefficients*, SIAM J on Optimization, 24(3), pp 1102-1121, 2014

[Vuotto 2019] S. Vuotto et al., *Automatic Consistency Checking of Requirements with ReqV*, IEEE Conference on Software Testing, Validation and Verification, 2019.

[Vuotto 2019b] S. Vuotto et al., *Automata Based Test Generation with SpecPro*, International Workshop on Requirements Engineering and Testing, 2019.

[Weiss 2008] G. Weiss, *A simplex based algorithm to solve separated continuous linear programs*, Mathematical Programming, Ser. A115, pp 151–198, 2008

[Zamacola 2018] R. Zamacola, et al., *IMPRESS: Automated Tool for the Implementation of Highly Flexible Partial Reconfigurable Systems with Xilinx Vivado*, International Conference on ReConFigurable Computing and FPGAs, 2018.

[Zamacola 2019] R. Zamacola et al., *Automated Tool and Runtime Support for Fine-Grain Reconfiguration in Highly Flexible Reconfigurable Systems*, IEEE International Symposium on Field-Programmable Custom Computing, 2019.