Information and Communication Technologies (ICT) Programme

Project Nº: H2020-ICT-2016-1-732105

D4.1: CERBERO Multi-Layer Adaptation Strategies (Final version)

Lead Beneficiary: UPM
Workpackage: WP4
Date: April 2019
Distribution - Confidentiality: [Public]

Abstract:

This document contains technical information on the different strategies to support adaptivity at hardware-, software- and sensor-level in CERBERO-compliant systems. It follows an incremental approach, showing a summary of the results reported in D4.3 (M15), and the new results and achievements since then.

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
Alfonso Rodríguez, Leonardo Suriano, Rafael Zamacola, Andrés Otero and Eduardo de la Torre	UPM
Tiziana Fanni	UNICA
Francesca Palumbo	UNISS
Florian Arrestier, Maxime Pelcat and Karol Desnos	INSA
Francesco Regazzoni	USI
Coen Van Leeuven, Joost Adriaanse, Julio de Oliveira Filho	TNO,
Leszek Kaliciak, Hans Myrhaug, Stuart Watt, Ayse Goker	Ambiesense
Pablo Sánchez de Rojas	TASE
Michael Masin	IBM

Document Revision History

Date	Ver.	Contributor (Beneficiary)	Summary of main changes
14/02/2019	0.1	UPM	Initial draft and ToC
04/03/2019	0.2	UNICA	Integrated contribution on Multi-grain adaptivity
04/04/2019	0.3	INSA	Integrated contribution on SW adaptation
12/04/2019	1.0	All	First version issued with all contributions included
16/04/2019	1.1	USI contrib. on security Bibliography update, First review round	Section on security added, INSA UNISS (F. Palumbo)
24/04/2019	1.2	Revisions on the overall document and minor corrections	UNISS (F. Palumbo), UNICA (T.Fanni), UPM (E. de la Torre)
25/04/2019	1.2	Minor updates after revisions	AS
26/04/2019	1.3	Final revision	UPM

30/04/2019	1.4	Preparation for submission	IBM
07/05/2019	1.5	Incorporation of missing contribution from INSA on Section 5.2.2	INSA

Table of contents

1. Executive Summary 7
1.1. Structure of Document 7
1.2. Relation with CERBERO Requirements 7
1.3. Related Documents 8
2. Overall Adaptation Strategies 9
2.1. The CERBERO Adaptation Components 9
2.2. The CERBERO Adaptation Components 10
3. Hardware Adaptation 11
3.1. State of the Art: Hardware Adaptation outside CERBERO 11
3.2. CEBERO Advances in Hardware Adaptation 12
3.2.1. ARTICo3 – MDC integrated adaptation 12
3.2.2. JIT composition of HW 15
3.3. Adaptation Strategies for security and reliability 19
4. Software Adaptation 21
4.1. State of the Art: Software Adaptation outside CERBERO 21
4.2. Challenges 21
4.3. State of the Art 22
4.4. CERBERO Advances in Software Adaptation 24
5. Sensor Adaptation 26
5.1. State of the Art: Sensor Adaptation outside CERBERO 26
5.2. CERBERO Advances in Sensor Adaptation 27
6. Adaptation Strategies in Use Cases 30
6.1. Adaptation Strategies in Planetary Exploration UC 30
6.2. Adaptation Strategies in Ocean Monitoring UC 31
6.3. Adaptation Strategies in Smart Travelling UC 31
7. References 34

1. Executive Summary

This document complements D4.3 – CERBERO Multi-Layer Adaptation Strategies (Ver. 1), where the overall adaptation scenario for the CERBERO project, as well as its main components are presented. Adaptation is addressed in three different fabrics: hardware-based processing, software-based processing and sensor layer. Moreover, and coupled with each of these fabrics, there is a set of adaptation engines that are in charge of enforcing all adaptation decisions made by the CERBERO Self-Adaptation Manager. This is achieved by actively changing the working point of the computing/sensor fabrics.

The approach followed in this document revises the State of the Art to complement the original prospections, and thoroughly documents all extensions and new developments made on each of the adaptation fabrics/engines.

Please, note that, for complete understanding, this document is to be read after D4.3 to get the whole picture of the various adaptation strategies proposed and developed in the project. Nevertheless, this D4.1 document is written to be as complete as possible.

1.1. Structure of Document

This document is meant to be an extended and updated version of D4.3, highlighting recent updates in both the State of the Art and CERBERO developments. Section 2 presents the updated components in the CERBERO adaptation loop, Section 3 focuses on hardware adaptation techniques, Section 4 focuses on software adaptation techniques, and Section 5 focuses on sensor-based adaptation techniques. The document finishes with a revision of the use of the proposed adaptation strategies on the use cases in Section 6.

1.2. Relation with CERBERO Requirements

Deliverable D2.2 of the CERBERO project defines a list of CERBERO Technical Requirements (CTRs) the project should achieve. The CERBERO adaptation strategy, and its related components and techniques, described in this document contribute to the fulfilment of the mentioned requirements in the following various aspects. The table below shows the means and the components achieved to fulfil these core requirements. The updates with respect to the equivalent table in D4.3 are shown with <u>underlined text</u>.

CTR id	CTR Description	Link with the D5.6 document on CERBERO framework components
0001	CERBERO framework SHOULD increase the level of abstraction at least by one for HW/SW co-design and for System Level Design.	The support provided by PREESM for the abstraction of SW and HW tasks, the capability of SPIDER to decide, at runtime, task migration between fabrics, the unified PAPI access scheme to monitors for HW and SW are the key contributions to this requirement.
0003	CERBERO framework SHOULD provide incremental prototyping capabilities for HW/SW co-design.	Incremental prototyping capabilities are envisioned at the tools/components level: • MDC will be provided with an enhanced HLS support; • DPR features will be improved thanks to JIT HW implementation and composition tool (<u>IMPRESS</u>);

		• the runtime monitoring of ARTICo³, JIT HW and MDC reconfigurable hardware accelerators will be enabled thanks to PAPIFY integration; • PREESM, SPIDER and PAPI will be used to drive ARTICo³, JIT HW and MDC prototyping features.
0006	CERBERO framework SHOULD ensure energy efficient and dependable HW/SW co-design using cross-layer runtime adaptation of reconfigurable HW.	Energy is a main KPI, and it is addressed in most of the techniques described in this deliverable, including all HW fabric types, SW agents, and sensor infrastructure and physical layer. Monitors and adaptation techniques for energy efficiency and dependability are foreseen in the three kinds of adaptation fabrics, also.
0009	CERBERO SHALL develop integration methodology and framework.	The adaptation infrastructure and the associated tools are part of the CERBERO framework.
0014	CERBERO WP and task leaders SHALL organize scheduled face to face and remote meetings.	WP4 periodic management meetings have been organised in order to track progress, deviations and risks. Adaptation-specific seminars have been produced externally (CPS Week and Summer School) and internally (Haifa GA).
0016	CERBERO tools SHOULD be tested vs state-of-the-art	Section 6 in this deliverable contains information about the use of the various components and technologies in the three use cases. Updated state of the art is included.
0018	CERBERO technology providers SHALL prepare face to face or online tutorials / education for use case engineers.	Tutorials on HW and SW adaptation have been prepared for the Summer School 2017 and CPS Week 2018. Academic engineers have received these courses in order to have feedback. CERBERO adaptation strategy was the key topic of a keynote in Summer School 2018.
0019	CERBERO technology providers SHALL coordinate technical support for their tools with use case engineers.	A preliminary version of some of WP4-related tools (PREESM, ARTICo3, MDC) has been delivered for the Summer School 2017. Use case integration meetings have been periodically achieved during the last months,
0020	CERBERO framework SHALL provide methodology and tools for development of adaptive applications.	This deliverable provides information about the components (mainly) and tools (partially) involved in the adaptivity support.

1.3. Related Documents

- D4.3 – CERBERO Multi-Layer Runtime Adaptation Strategies (Ver. 1): the initial version of this deliverable. It contains detailed technical information on the different strategies to support adaptivity in CERBERO-compliant systems. D4.1 extends D4.3 to cover new developments/features and revised state of the art.
- D4.2 – CERBERO Self-Adaptation Manager (Final version): The CERBERO Self-Adaptation Manager uses the strategies presented in D4.1 to orchestrate adaptation. Hence, D4.1 is an input for D4.2.
- D5.2 – CERBERO Framework Components (Final version): all the specific adaptation techniques and components described in D4.1 are bounded by specific framework components (detailed in D5.2).

2. Overall Adaptation Strategies

2.1. The CERBERO Adaptation Components

Figure 1 shows an updated view of the multi-layer adaptation scheme identified in CERBERO as the background for building all dependencies and interactions between elements. If compared with the previous figure, contributions are centred on the adaptation to CPSoS layer, where, now, two adaptation loops (CPS and CPSoS) can be identified, and to show the possible interactions with the Physical part.

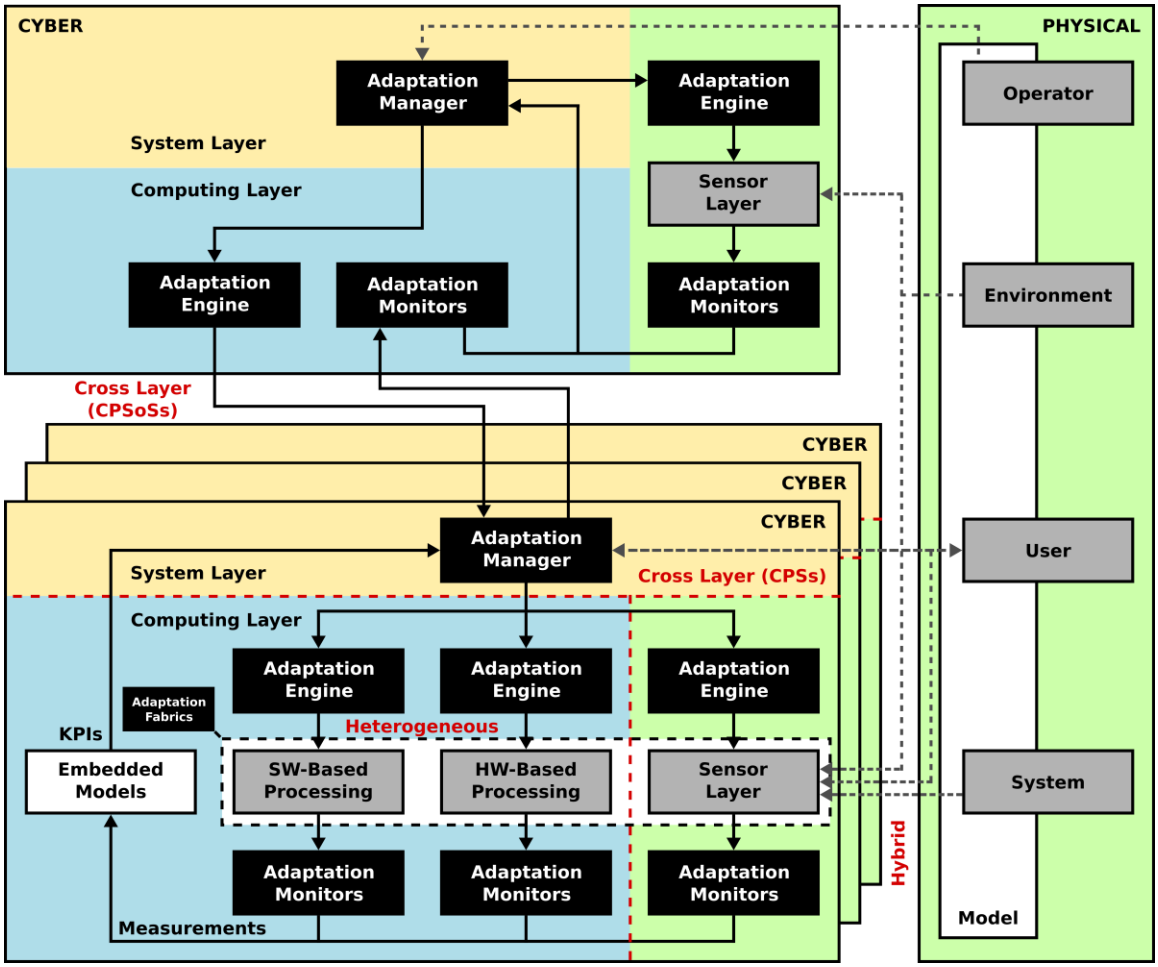

Figure 1 – CERBERO (Self-)Adaptation Infrastructure

As it can be seen, the identification of the elements that comprise the adaptation loops at both CPSoS and CPS level have been defined comprehensively and using the same terminology, since we are having the intention of being able to integrate and generalise as much as possible any type if adaptation, for any type of system, for all elements that compose the loop and for any layer. In other words, there will always be, no matter of the considered abstraction level, some elements that can be mapped into one the categories:

- **Adaptation Fabrics**, which hold all the computing and sensing resources.
- **Adaptation Monitors:** they comprise hardware and software components to track the state of the *Adaptation Fabrics and of the Physical subsystem.*
- **Adaptation Engines** are the components that drive the changes in the *adaptation Fabrics*, according to decisions taken from the adaptation manager.
- **Adaptation Manager:** are the entities with runtime decision-making capabilities. They uses the information provided by the *Adaptation Monitors, made comprehensive and perhaps simplified and estimated via a model,* to decide whether to trigger adaptation or not.

When putting together the adaptation loops at CPSoS and CPS level, it must be pointed out that:

- Every CPS subsystem, considered as a single entity, acts as an adaptation fabric for the CPSoS adaptation loop level. Therefore, individual CPSs should respond to adaptation triggers produced by the adaptation engine at the higher level.
- The adaptation managers of single CPSs should be capable of processing the information obtained from their own monitors in order to provide simplified/merged/normalised/forecasted KPI information to the higher layer.
- KPI information should be obtained by aggregation or fusion of the info obtained from individual lower layer subsystems as well as from the physical subsystem.

2.2. The CERBERO Adaptation Components

The following table identifies some of the basic components addressed in CERBERO within this double-loop adaptation scheme.

Layer / Comp.	CPS layer					CPSoS layer	
Adaptation fabric	HW fabrics			SW fabric	Sensor	SW	Sensor
	ARTICo3	MDC	JIT				
Adaptation engine	DPR block	CGR	DPR block + FGR	Spider	Weights in Fusion Model		
Adaptation manager	Spider		Spider+EA	Spider		DynAA	
Adaptation monitors	Papify	Papify	Papify	Papify	MECA		MECA (user)

3. Hardware Adaptation

his section addresses updates in the state of the art technologies due to a continuous technology monitoring effort of the involved partners, as well as updates in the developments achieved within the HW-based fabrics, namely, the combined granularity achieved by combining ARTICo3 and MDC architecture types, advances in the Just-In-Time HW composition techniques for adaptive and deterministic HW creation, and a discussion on security and reliability related issues.

3.1. State of the Art: Hardware Adaptation outside CERBERO

Hardware adaptation is highly studied in literature, at different levels of granularity, and D4.3 provided an overview of the studies presented at the State of the Art. This section provides an update with the solutions presented in 2018 related to hardware adaptation and CERBERO topics.

Le Lann at al. [Le-Lann'18] proposed Argen, an integrated toolflow for Overlay-centric System-on-chip. The presented overlay is based on a homogenous regular array of cells composed of several logic elements containing look-up tables and registers. The Argen toolchain takes care of all the steps of the design flow: architectural and application design, application mapping and synthesis, as well as final binary code generation, for both the overlay and the application.

Liu et al. [Liu'18] proposed a hybrid-grained reconfigurable architecture (HReA) to process 13-Dwarfs computation [Asanovic'06] (a dwarf is an algorithmic method that captures a pattern of computation and communication). HReA combines a 32-bit Coarse-Grained Reconfigurable (CGR) datapath with a 1-bit Fine-Grain Reconfigurable (FGR) datapath to accommodate co-existence of multiple computing granularities in 13-Dwarfs. The two datapaths with different granularities can interact with each other in an arithmetic logic unit.

Fuchs et al. [Fuchs'18] presented a tiled multiprocessor system-on-a-chip (MPSoC) design able to provide fault detection, isolation, and recovery (FDIR) for very small spacecraft. They exploit a multi-stage fault tolerant approach that implements forward error correction and utilizes coarse-grain lockstep of weakly coupled cores to generate a distributed majority decision across tiles. FPGA reconfiguration is exploited to recover from upsets in tile logic, and cover permanent faults using alternative configuration variants. If too few healthy tiles are available due to accumulation of permanent faults, re-allocation of resources to high-criticality applications, by sacrificing performance of lower-criticality threads, is performed.

The research conducted on hardware adaptation outside CERBERO either consider only one kind of adaptivity or mesh-based arrays. Both Le Lann at al. [Le-Lann'18] and Liu et al. [Liu'18] exploits a hardware-to-application approach in which the application is mapped on an existing regular array of homogenous processing elements, and reconfiguration is based only on virtual reconfiguration (multiplexing resources in time by means of multiplexers). Furthermore, none of them address repair-oriented adaptivity. While Fuchs et al. presented a reconfigurable architecture to provide fault tolerance, but

they do not address any other type of adaptivity, neither they provide hardware accelerations.

3.2. CERBERO Advances in Hardware Adaptation

Hw-Adaptation in CERBERO is supported by the ARTICo^3, MDC and Just-In-Time (JIT) composable HW overlays. ARTICo^3 and MDC support DPR (Dynamic and Partial Reconfiguration) at block-level and CGR (Coarse-Grain Reconfiguration) respectively. The combination of ARTICo^3 and MDC offer a new level of flexibility implementing a multi-grain heterogenous reconfigurable fabric.

MDC accelerators have been embodied into ARTICo^3 wrappers and set as new ARTICo3 compatible HW accelerators that may be configured to obtain HW scalability (ARTICo^3 feature), dynamic fault tolerance (ARTICo^3), and fast functional and non-functional adaptive ity (MDC feature) with HW reuse (MDC feature). The availability of previous results before CERBERO in ARTICo^{3}and MDC, together with the work achieved in T4.3, allowed a successful integration of the tools and the development of a consistent methodology to support the combined granularities from both approaches.

Just-In-Time composition, on the other side, was identified since the very beginning as a less mature approach, with associated risks and challenges, that invited at the moment of writing the CERBERO proposal to go for a longer-term integrated architecture and toolflow. In few words, JIT composable HW offers yet another degree of reconfiguration (we call it fine-grain), and requires the definition of, on one side, new tools to support it (IMPRESS, [Zamacola'18]) and, on the other side, two design approaches to make designs on top of them:

- A deterministic circuit design method, based on intermediate representations and mapping into HW blocks.
- An iterative approach based on training/evolutionary methods on which the circuit is obtained by composition of blocks whose global functionality is verified against an objective function to be optimized in order to obtained a circuit that matches as much as possible the expected behavior for the system at hand.

ARTICo3 – MDC is reported in section 3.2.1, whereas the advances in JIT composition (mainly on the iterative approach) are shown on section 3.2.2.

3.2.1. ARTICo3 – MDC integrated adaptation

With respect to the State of the Art, the hardware adaptivity provided in CERBERO is based on the multi-grain reconfiguration, given by the combination of ARTICo^3 framework and MDC tool. Practically speaking, in terms of architecture, depicted at the bottom Figure 2, MDC compliant accelerators (as those reported in D4.3) are wrapped within ARTICo^3 slots (described in D4.3 as well). This type of integration delivers the best of both CGR and DPR approaches into an adaptive multi-grain heterogenous reconfigurable fabric, which can meet the changing of functional, non-functional and repair-oriented requirements of CPS designs. Moreover, with respect to the above-mentioned works in literature, the CERBERO approach is not simply meant to provide a novel architecture,

rather it aims at offering support the design of the different parts of the system, their deployment and runtime management [Fanni'18] [Rodríguez'18].

The hardware generation flow starts from high-level dataflow descriptions of the configurations/behaviors to be implemented in the configurable logic, and the integrated toolchain derives the corresponding CGR HDL computational kernel, properly wrapping it with the glue logic necessary to serve as an ARTICo^3 DPR reconfigurable partition. Both reconfiguration mechanisms are transparently managed by the user code running in a host processor. With respect to the standalone MDC and ARTICo^3 flows, an adaptation step (*Kernel Adapter*) is needed to make the MDC-generated kernels compliant with kernels expected by ARTICo^3 *Wrapper Automation* step.

Figure 2 illustrates the whole MDC-ARTICo^3 design flow, through a Step-by-Step example that considers three input dataflow networks. (1) Firstly, MDC merges the user-defined dataflow specifications and generates the CGR computing core as described in D5.6. (2) Then, the generated mm-TIL is modified by the Kernel Adapter which delivers an HDL ARTICo3-compliant CGR kernel. (3) Finally, the ARTICo3 framework processes the input HDL CGR kernel to implement the whole reconfigurable processing system (see D5.6 for details on ARTICo^3 framework). (4) The bottom part of Figure 2 depicts an example of multi-grain reconfiguration. In this example the CGR approach offered by MDC is exploited for low-power fast-switching of functionality, while the DPR supported by ARTICo^3 is exploited for changing the number of slots working in parallel to increase the throughput or provide fault tolerance.

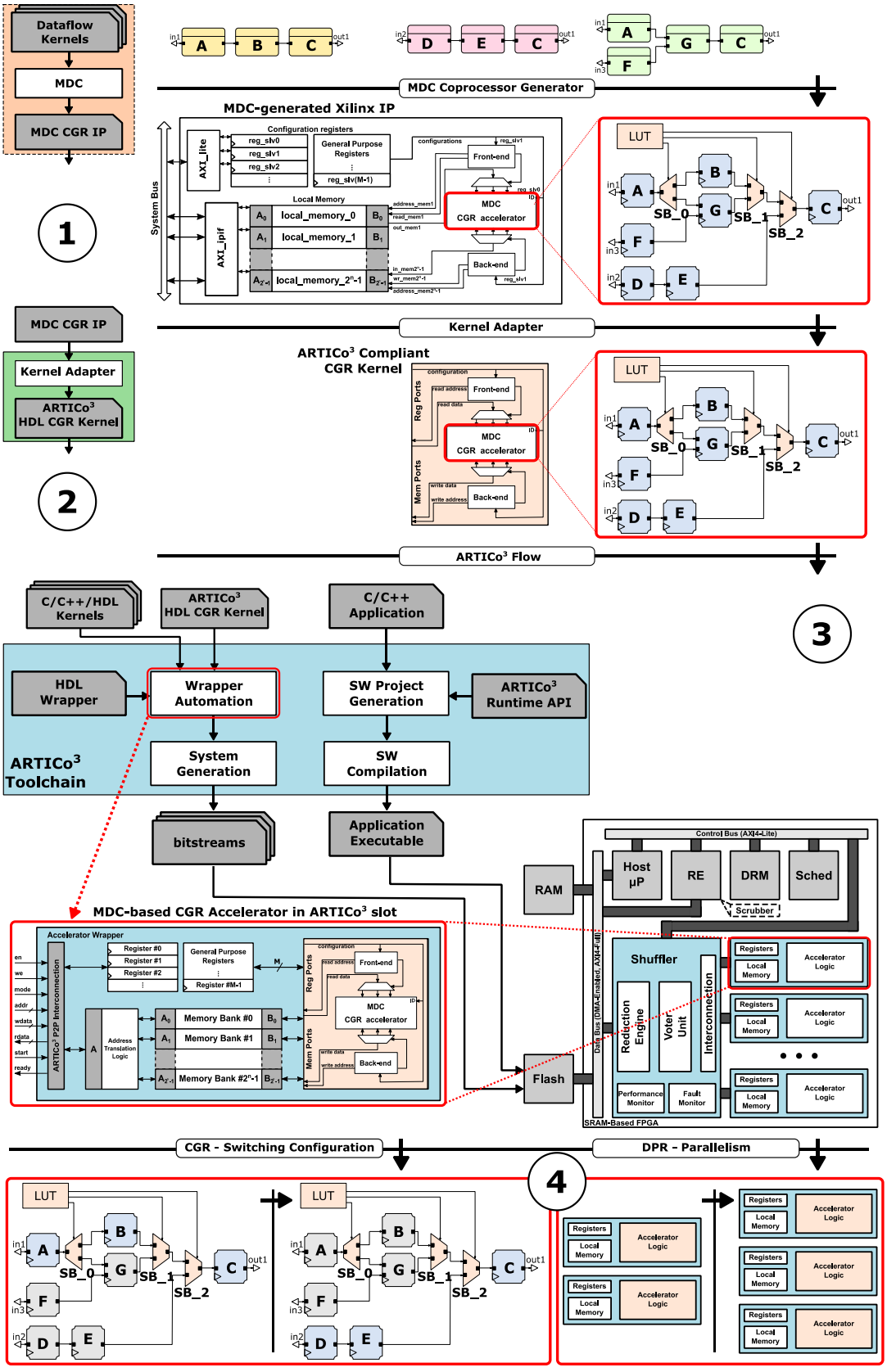

Figure 2 - Multi-grain design flow and adaptation.

3.2.2. JIT composition of HW

Adaptation fabric

JIT hardware composition refers to the ability to implement, at runtime, hardware accelerators on FPGAs without a pre-synthesized design. Hardware accelerators are mapped onto an overlay, a 2D highly regular mesh-type architecture composed by multiple processing elements (PEs) placed on top of an FPGA. Overlays can be fixed, (i.e., they have a predefined PE composition that cannot be changed) or can be reconfigurable (i.e., PEs can be reconfigured to change their functionality).

As explained in deliverable 4.3, two different methods to compose accelerators were envisaged. A deterministic approach where a software algorithm is converted to a dataflow graph that is mapped to the overlay, and an iterative/evolutionary approach where the accelerator is composed using an evolutionary heuristic to find an overlay configuration that solves a given problem (e.g., how to control a cart-pole system).

Traditional reconfigurable overlays have a predefined virtualization where specific regions of the FPGA are reserved to allocate different PEs. Therefore, in these overlays even if the PEs are reconfigurable, the overlay structure (i.e., mesh location, communication between PEs, PE shapes) is fixed. For this method, we propose the use of a run-time composable overlay. We propose to create just one reconfigurable region (which could be later on embodied into an ARTICo^3 slot) in the FPGA that defines the communication with the static system (i.e., the rest of the design that is not reconfigured over time). This reconfigurable region is then loaded with PEs with different shapes. These PEs interface with other modules through their borders, thus it is possible place PEs that connect to their neighbour PEs. This solution allows to build multiple different overlays specialized for different application domains without being constrained to a fixed overlay architecture. This overlay composition makes it highly recommended to use relocatable partial bitstreams (PBS) so that one PE can be allocated in compatible FPGA regions (i.e., regions that share the same resource footprint). Moreover, these overlays can use modules spanning less than a clock region, therefore it is necessary to allow a finer granularity by allowing to stack several modules in one clock region.

Currently, it is not possible to build run-time composable overlays with commercial tools. Therefore, the first step to build these architectures has been to create IMPRESS [Zamacola'18] (IMplementation of Partial REconfigurable SystemS) a TCL-based reconfigurable tool which incorporates the following features: decouples the implementation of the reconfigurable and the static system, allows RM to RM communication, enables hierarchical reconfiguration, RM reallocation, and makes it possible to stack multiple RPs in the same vertical columns, within the same clock region. All these features enable just-in-time overlay composition.

There is, however, one big limitation of these run-time composable mesh-type overlays. This is how to access inner PEs (that do not have a direct connection with the static system) in the mesh to change hardwired parameters, datapath or even functionality in a fast way without having to reconfigure the entire PE. To tackle this problem IMPRESS have been incremented with LUT-based reconfiguration. Several LUT-based parameterized components have been created, so that the user can instantiate them in their HDL (hardware description language) PEs. The available LUT-based components are: a constant, multiplexer and a functional unit that can implement several functions. Once the user

instantiates them in their PEs, IMPRESS is in charge of packing all the LUTs in the minimum number of columns. These components can be reconfigured at run-time in a fast way as only the LUT truth tables need to be reconfigured while the rest of the PE is left unaltered.

The first overlay built using just-in-time HW composition tools is a block-based neural network (BBNN). BBNNs are highly regular 2D architectures in where each PE can contain 1 one to 3 neurons of the network. Differently to classical neural networks the structure of the network is not fixed as each cell can adopt multiple configurations (i.e., it can be connected with different blocks and use a variable number of cells). Therefore, training of these neural networks works not only to obtain a set of weights and bias for each cell but also a specific network configuration. BBNNs can be considered as a specific case of JIT hardware composition based on the iterative/evolutionary approach. In this case instead of changing between different processing elements, we reconfigure each cell parameters (i.e., weights, bias and cell type).

The proposed BBNN shown in Figure 3 is based on the proposals in [2], however the cell structure has been modified to reduce resource usage. To that end, only one DSP is used for MAC (multiply and accumulation) operation. There is one FSM (finite state machine) that controls the operation of the cell which has a latency of seven cycles. The neural network is constructed using JIT in time HW composition by composing the BBNN cells in an initially empty reconfigurable region. The configurable parameters of the cell (i.e., weights, bias and cell type) are implemented with reconfigurable LUT-based components.

The proposed BBNN is trained using an evolutionary algorithm that tries different solutions until finding a valid one. The main benefit of using an evolutionary algorithm instead of gradient-based algorithm is that online training is possible as we use the same processing array for the training and inference phases. The BBNN has been used to solve control problems. The cart-pole and mountain car python simulation provided by the openAI framework have been used to show the effectiveness of the proposed solution.

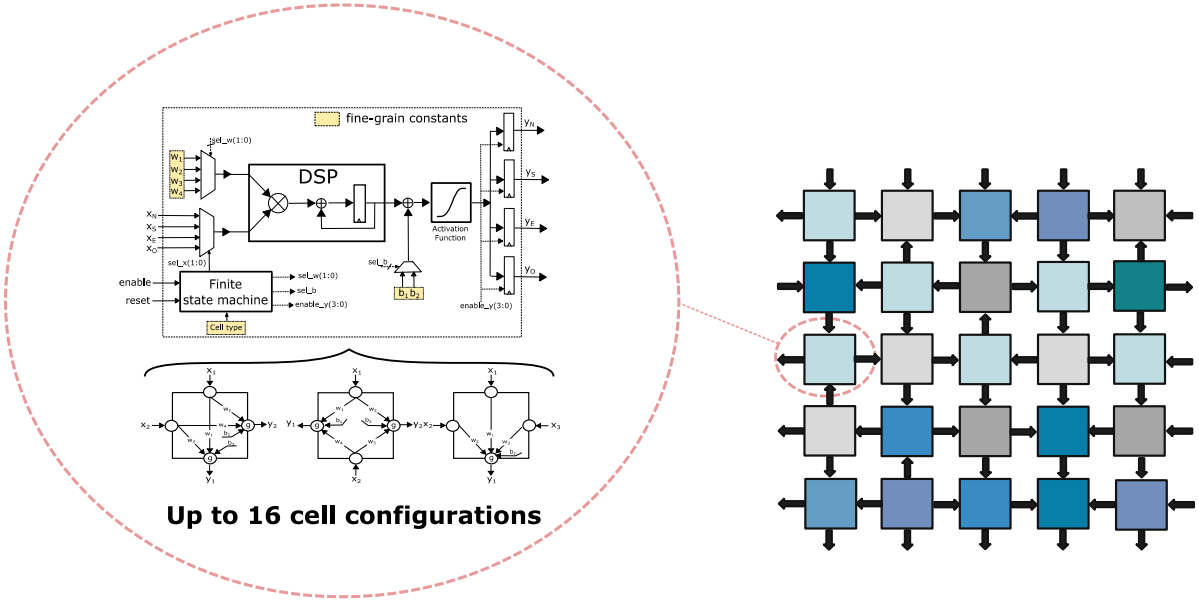

Figure 3 - block-based neural network structure

Regarding the deterministic just-in-time HW composition approach we can distinguish three steps: dataflow graph extraction, overlay design and lastly the place & route of the dataflow graph onto the overlay. The first step, the dataflow graph extraction, has been realized in two steps. First, from a SW description the LLVM intermediate representation (IR) is obtained and then the dataflow graph is extracted. There are still improvements to achieve in this step like support loop unrolling and if to multiplexer conversions in LLVM IR code. The overlay design and place & route steps will be tackled in the following months.

Fine-Grain Adaptation engine

This subsection addresses the new type of reconfiguration granularity that complements the block-based one (used in ARTICo^3)

IMPRESS not only provides support for generating the static system and reconfigurable processing elements (PEs). It also provides run-time support (adaptation engine) to compose overlays at run-time. This adaptation engine is made of one software layer that provides an API to compose overlay at run-time and two reconfiguration engines (REs). A reconfiguration engine is a component in charge of reconfiguring the FPGA configuration memory to modify the circuit implementation. The proposed adaptation engine is composed of two different REs. The first one is a generic SW-based RE that is in charge of changing entire reconfigurable PEs using the PCAP reconfiguration port while the second one is a specialized RE implemented in HW that can rapidly reconfigure LUT-based components using the ICAP reconfiguration port.

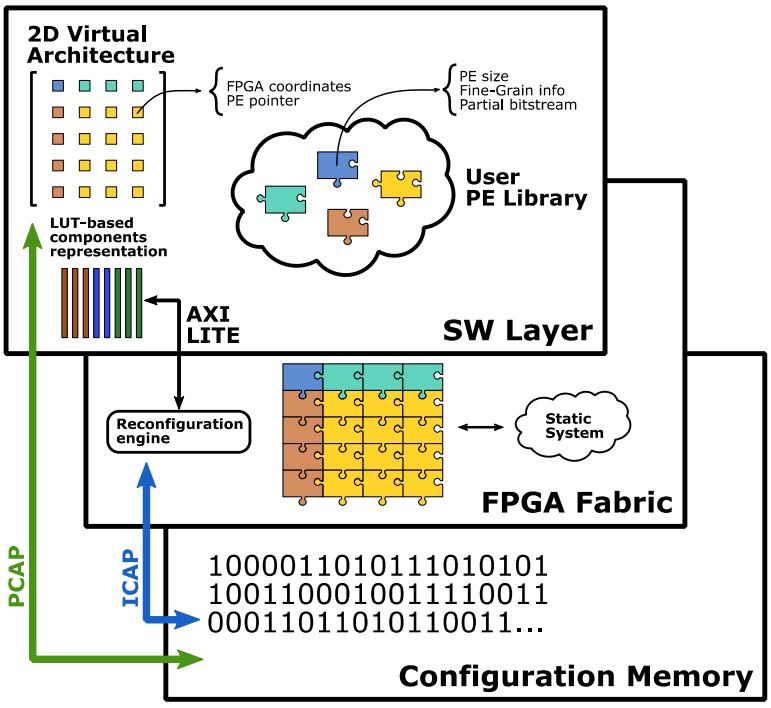

Figure 4 - Adaptation engine for JIT HW composition

Figure 4 shows an overview of the adaptation engine. The SW layer is composed of three main components:

- A PE library: this library is made by the user at design time and contains all the different elements that are available for composing different overlays at run-time. Each element of the library contains the information on PE size, partial bitstream and reconfigurable LUT-based components used inside the PE. This library can be visualized as the pieces (PEs) that can be used to make different puzzles (overlays).
- 2D virtual architecture: this matrix variable is an internal representation of the overlay implemented in the FPGA. Each element has information on the PE location and a pointer to a specific PE from the PE library. Once an element of the matrix is updated with a new PE, the new PE partial bitstream is sent to the generic SW-based reconfiguration engine. Therefore, the internal matrix representation always resembles the implemented overlay. It is the responsibility of the user to ensure that the processing elements have compatible interfaces with the neighbour PEs.
- LUT-based component representation: these variables represent the configuration of LUT-based components. As explained before, LUT-based components are packed in columns at design time by IMPRESS. Therefore, each of these variables represent one column filled with LUT-based components. Once the user changes a specific component of a specific processing element of the overlay, automatically the adaptation engine searches for the component location and modifies the affected internal representation of the column. The user can initiate the fast LUT reconfiguration by sending to the specialized HW-based RE each of the columns that have been modified using an AXI lite interface.

Overlay composition procedure

The first step is to design the static system, which must contain all the elements that will remain unaltered during the system operation. It must also include, at least, one region reserved to implement the overlay at run-time. This region is initially empty and only includes dummy logic instantiated by IMPRESS that interfaces with the static system.

Then, it is necessary to design all the processing elements (PEs) that can be used for overlay composition. The PE interface is defined selecting the borders that can be used (e.g., the north side). It is responsibility of the user to create PEs that have compatible interfaces. Many overlays are just composed with one PE that is replicated over a lattice, in this case it is really easy to ensure a PE design with an interface that can be replicated ensuring a correct communication.

The previous two steps generate a full bitstream of the static system and a partial bitstream for each PE. The user needs to fill the PE library explained in the adaptation engine section with the information of each PE, including the information of every LUT-based component that the PE integrates. Then the partial bitstreams are stored in an SD card and the static bitstream is loaded into the FPGA.

Then the user can modify the matrix variable “2D virtual architecture” shown in Figure 4. Each element of this matrix contain a location of the FPGA and pointer to a PE element from the PE library. Once the user modifies an element, the SW-based reconfiguration engine is used to update the configuration memory adding the selected processing element

to the specified position. This process is repeated until all the overlay processing elements have been reconfigured.

Once the overlay has been placed it is possible to tune LUT-based constants, multiplexers and functional units that are part of the processing elements. The user just has to select a specific processing element and the component to change and it new value. Automatically the tool updates the internal frame representation of the LUT-based components updating it with the new contents. Once the user has changed all the LUT-based components it can send the new representation to the specialized reconfiguration engine that changes these components using the ICAP port.

3.3. Adaptation Strategies for security and reliability

In this section we discuss how to address the security needs within CERBERO, and adaptation and reliability are used in this context. Our security needs consist in guaranteeing, in a reliable way, the confidentiality of the information and in providing authentication of the message. Both requirements can be fulfilled by using authenticated encryption algorithms. Several authenticated encryption algorithms, addressing different goals and needs, have been proposed in the past, especially within the CAESAR contest [Caesar'17]. The choice of the most suitable algorithm should be dictated by the needed level of security, the required performance, and by the target power and energy consumption. However, this choice is not always possible, since it is often imposed by a standard to which a specific communication link must be complaint with. This is our case, were the authenticated encryption should be provided using the AES algorithm in CRT mode [Dworkin'01], supporting a key size of at least 128 bits, and the authenticated encryption has to be provided by AES used in GCM [Dworkin'07] mode, with a MAC of size 128 bit. Despite the restriction on the algorithm type, designers are still free to decide the specific implementation strategies that are most suitable for optimizing other parameters, such as performance, reliability, and area occupation.

Addressing these needs, we designed modular hardware accelerators and software routines implementing the required encryption and authenticated encryption algorithm, capable of providing reliability thanks to the appropriate error detection and correction methodology. The parameters of the algorithm and the level of reliability can be adapted at run time. Adaptation, in the context of security, is often considered synonymous of crypto-agility, which, informally, is often defined as the capability of quickly switching between different cryptography primitives and algorithms. In our context, we use adaptation to rapidly offer the possibility of changing parameters of the algorithm under execution. In our case, changing the parameter of the algorithm mainly means to be able to configure, at run time, the desired length of the key. In AES changing the level of the key requires also to change the number of rounds, and to modify the key unrolling function. These changes are handled automatically by the architecture.

Our accelerators have to operate reliably, thus they need to provide support for detection and correction of errors. In security error detection and correction are important for two reasons. The first, is to guarantee the resistance of the algorithm to accidental faults. The second is that, thanks to error detection and correction schema, it is possible to mitigate the so called fault attacks [4] (these attacks aim at gain sensitive information from a device by inducing it into an erroneous state and by analyzing the difference between the correct and

the wrong execution). Several fault detection and correction have been proposed for AES, which is our target algorithm, and for block ciphers in general. Proposes schema range from a simple parity bit, added at a granularity of a byte [Bertoni'05], to more complex codes capable of providing also correction [Regazzoni'12].

We augmented our accelerators with different algorithm for detection and correction of errors. This support for reliability is modular and can be activated at run time. Depending on the desired level of reliability, our design can provide simple error detection capability using a single bit of parity or much more complex correction capability using an advanced hamming code. The selection of the specific reliability mechanism is driven by a trigger that can come from an internal monitor (in case complete insulation of the cryptographic module is required) or can come from an external monitor.

Despite designed for the AES algorithm, the proposed modular approach is sufficiently generic for being applied to any block ciphers. To demonstrate this fact, we used the same methodology to realize a modular and adaptable accelerator for lightweight algorithms. We focused on lightweight algorithm as second case of study because they are used in resource constrained designs (as several CPSs are) and because they will be soon standardized. In the next months, the designed architecture will be demonstrated within the CERBERO use case.

4. Software Adaptation

4.1. State of the Art: Software Adaptation outside CERBERO

In this section, we first present the main challenges addressed on software adaptation within the CERBERO project. Then, we present existing techniques and frameworks in the context of reconfigurable dataflow MoCs, and the approach chosen within the CERBERO project to tackle the problem.

4.2. Challenges

As explained in Deliverable D3.5 on *Models of Computation*, the CERBERO project develops tools and methods that rely on precise Models of Computation and Models of Architecture in order to foster reuse and tool reusability. As for software reconfigurability, we consider a reconfigurable dataflow MoC such as the PiSDF MoC [Desnos'13] or the SPDF MoC [Fradet'12] for their capacity to represent a complex datapath with limited and partly predictable control. A review of State of the Art on system reconfiguration in a broader context can be found in D4.3 on the CERBERO Self-Adaptation Manager. This section concentrates on parameterized dataflow-based software reconfiguration.

In the considered context, reconfigurable means that the application graph may evolve at runtime with changes in both data rates and in the graph topology itself. The software platform management system needs to evolve accordingly and exploit the resources of the underlying parallel execution platform. Using a reconfigurable dataflow MoC, a full static analysis of an application is generally not possible at compile time and needs to be handled at runtime. For instance, the degree of parallelism or amount of exchanged data are likely to change at runtime. However, when parameters remain static and application topology is stable for a long time, runtime management should exploit this conservation of properties to reduce management complexity and cost.

SPDF and PiSDF MoCs lend themselves well to quasi-static scheduling [Bhattacharyya '00], deriving at compile time a set of pre-computed schedules, removing a part of the runtime overhead. Conversely, when dealing with dynamic behavior such as graph reconfiguration, a first challenge is to perform graph analysis and scheduling of the application with an overhead as low as possible with regard to the application execution time. Ideally, the time allowed for those analyses should always be negligible when compared to effective computation time.

A second challenge is on the memory footprint of the runtime manager. Analysis techniques need to store application state information that is used only for analysis and decision purpose.

In embedded architectures such as the Kalray MPPA manycore processor [De Dinechin '13], memory consumption is a major concern. The MPPA processor is exemplary of the current challenges faced by software reconfiguration. Its architecture features 16 clusters of 16 VLIW processing cores. Each of the clusters has a local memory of 2MB and, although it has access to a larger external shared-memory, reading and writing to this external memory is excessively expensive and should be avoided as much as possible. In this context, storing additional information for analysis purpose in the runtime manager

can result to more frequent accesses to the shared-memory and thus in a strong downgrade of overall software performance. In the CPS context of the Cerbero project, all the considered energy efficient software processing architectures share these concerns, i.e. parallelism exploitation and difficult resources scheduling.

We now present a State of the Art of dataflow-based runtimes and approaches for reconfigurable dataflow-based self-adaptation. We show that most State of the Art methods and all available software for software adaptation from dataflow MoCs use a directed acyclic graph intermediate representation that can raise problems in terms of data storage and management time. A major objective of the CERBERO project software adaptation strategy will be to free the runtime system from this intermediate representation.

4.3. State of the Art

HI-HTGS (HMBE-Integrated-HTGS) [Wu'18] is a design tool that aims at automating analysis and optimizations of Windowed Synchronous DataFlow (WSDF) graphs [Keinert '06]. HI-HTGS provides a lock-free and race-condition-free scheduler that dynamically adapts to changes in actor execution times and copes with nondeterministic characteristics of a thread-based execution. HI-HTGS works in two distinct phases: a compile time phase and a run-time phase. During the compile time phase, HI-HTGS builds a Single-Rate Directed Acyclic Graph (srDAG) representation of the WSDF user graph and performs various analysis that will serve during the run-time phase.

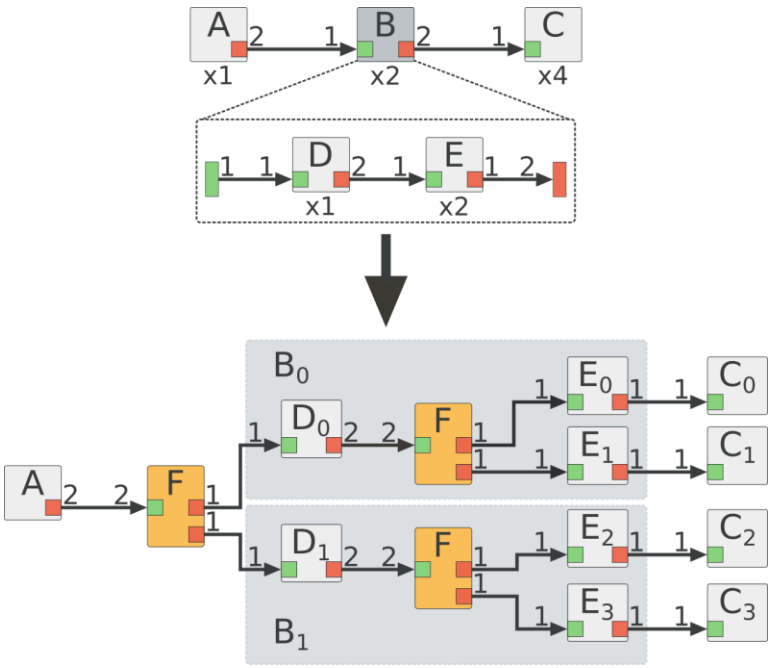

Figure 5 - Transformation of a hierarchical PiSDF graph into an srDAG

An srDAG, also called Acyclic Precedence Expansion Graph (APEG) [Lee 1987] is a specialization of an SDF graph that does not contain cycles and all the data rates on edges are unitary (i.e. carrying one data token). Figure 5 shows the transformation of a PiSDF

graph, upper part of the figure, to the equivalent srDAG, lower part of the figure. Under each PiSDF actor of are indicated the repetition value relatively to their containing graph. Actors D and E have repetition vector values of 1 and 2, respectively, within 1 iteration of actor B but a total repetition value of 2 and 4, respectively. For more details on dataflow representation, one may refer to Deliverable D3.5 on *Models of Computation*.

The srDAG is useful because it exposes the available application concurrency. The transformation to an intermediate srDAG is however very costly, as it creates one vertex in the intermediate application graph for each execution (firing) of the corresponding task (actor). At run-time, HI-HTGS uses the built srDAG and additional metadata of the compile time phase to perform dynamic scheduling on a multi-core processor. However, due to its compile time construction of the srDAG intermediate representation, HI-HTGS can only handle static applications with constant data rates.

SPIDER [Heulot'14] is a runtime manager designed for the execution of reconfigurable PiSDF applications on heterogeneous Multi-Processor System on Chip (MPSoC) platforms. SPIDER manages a PiSDF graph high-level description of the application as input. Due to the reconfigurable nature of the PiSDF MoC, SPIDER derives an SRDAG at runtime and performs graph optimizations, mapping and scheduling of the application at runtime, as opposed to HI-HTGS [Wu'18]. The transformation to SRDAG may take non negligible time on reconfigurable applications with high-level of task and data parallelism and with low complexity computation kernel.

The OpenVX language and standard [Rainey 2014] is a graph-based Application Programming Interface (API) proposed by the Khronos group for developing and deploying computer vision applications on embedded platforms. OpenVX uses a dataflow MoC which enables programmers to design their application in an architecture agnostic fashion while exposing high-level optimization and parallelism opportunities. The MoC used by the OpenVX standard is an SRDAG specialization of the Synchronous Dataflow (SDF) MoC [Lee'1987]. srDAGs, when used to manually model the application, are less expressive and more restrictive than SDF graphs but allow for global high-level optimization. This choice is motivated in OpenVX by the fact that each node is supposed be a coarse-grain computer vision, or deep-learning computation kernel, representing thousands or millions of instructions. Task/Actor granularity is an essential concern when building dataflow-based software adaptation, as it influences both programming languages and intermediate representations.

In SDF graphs, data-parallelism comes implicitly from the property of the graph to have non-unitary data rates, allowing for each computation kernel to be further parallelized because they do not require the full data to fire. As a consequence of its srDAG representation with unitary tokens, the OpenVX standard mostly relies on task parallelism.

Other runtimes exist, such as StarPU [Augonnet'11] or XKaapi [Gautier'13], that manipulate task graphs. Similarly to OpenVX, StarPU and XKaapi use a DAG dataflow model to schedule the different tasks. The main difference between these runtime systems lies in the used scheduling policies. While StarPU uses a Heterogeneous Earliest-Finish-Time (HEFT) scheduling [Topcuoglu'02] algorithm, XKaapi uses a work-stealing based scheduling .

StarPU and XKaapi mainly focus on High Performance Computing (HPC) applications ran on heterogeneous architectures composed of networked multi-core CPUs and GPUs

whereas the OpenVX main focus is on embedded computer vision applications. It is important to note that contrary to OpenVX, StarPU schedules the application graph at the same time the graph is constructed, thus limiting its vision of the full application for resources allocation decisions and making greedy algorithms compulsory.

To limit the explosion of nodes in the SRDAG transformation, clustering of the original SDF graph is proposed in [Pino 1995], where four clustering criteria are identified. These clustering criteria provide sufficient conditions for checking the introduction of deadlocks in resulting clustered graph. Pino et al. then propose a hierarchical scheduling algorithm and shows that clustered SDF graph result in faster scheduling with very low impact on the obtained makespan compared to scheduling the full SRDAG. Using a MoC that is hierarchical and compositional by nature, as in the Interfaced Based Synchronous Dataflow (IBSDF) MoC [Piat'09] or the PiSDF MoC removes the need of the clustering step and the hierarchical scheduling algorithm may be used directly.

Another approach to avoid the full-expansion of the srDAG is called the vectorization of the SDF graph [Ritz 2993]. In the work of Ritz et al., optimal vectorization of the SDF graph is achieved by multiplying the rates of the original graph by integers resulting in less invocation of the actors of the SDF graph. A Partial Expansion Graph (PEG) formulation [Zaki'17] provides a framework in which vectorization of actors is integrated efficiently for multiprocessor scheduling context. Zaki et al. use Particle Swarm Optimization (PSO) to find and adjust the amount of expansion, or vectorization, of the actors of the graph. In the CERBERO adopted approach, fast heuristics are developed based on the topological structure of the graph in order to achieve vectorization of actors at the scheduling level.

Schedule-Extended SDF graphs [Damavandpeyma'13] constitute another class of SDF graphs that aims at providing a more compact representation for throughput analysis and buffer sizing than srDAGs. Damavandpeyma et al. show that encompassing scheduling information directly into the original SDF graph significantly reduces time for iterative throughput and buffer sizes analysis. Additionally, authors show that srDAG representations may lead to overestimated buffer sizes compared to applying same technique on schedule-extended SDF graphs. Authors also mention that construction time of the srDAG is very low compared to the analysis time. Although this is true in the context of static analysis at compile time, the same assumption can not be made when construction of srDAG is performed at runtime.

All existing works presented in this section show that using an srDAG transformation for scheduling and analysis of dataflow graphs is the most classical approach. srDAG offers a complete exposure of task and data parallelism available in the application. However, most of the presented work use static dataflow MoCs and srDAG computation time is neglected, as it can be computed at compile time. In the context of a reconfigurable MoC such as the PiSDF MoC, embedded runtimes need to compute srDAG on-the-fly and this transformation may have a significant impact on application performance, especially in the context of embedded platforms.

4.4. CERBERO Advances in Software Adaptation

Building the srDAG from an SDF or hierarchically modeled application to feed runtime management does not always provide the best performance. The resulting srDAG graph often contains more parallelism than what can actually be exploited by the targeted

architecture. Moreover, the exponential growth of the SRDAG with respect to the original SDF graph increases the complexity of scheduling algorithms for MPSoC platforms.

In this context, the CERBERO software adaptation method and tooling aim at building a srDAG-free adaptation method and confront it to the srDAG-based State of the Art runtime management strategy in terms of resources and time behavior. A tunable numerical model of the SR-DAG is introduced which allows for faster scheduling, less overhead and more flexibility. This model is compared to current approach on various real case computer vision and machine learning applications. This work is currently under submission.

5. Sensor Adaptation

Here we consider sensor adaptation based on fusion of information from multiple sensors. Specifically, using sensor fusion to enable adaptivity. Information from multiple sources is combined into a unified space in order to paint a more accurate picture of observed phenomena and make better decisions.

5.1. State of the Art: Sensor Adaptation outside CERBERO

The state of the art in sensor fusion and adaptation is usually based on methods presented in D4.3, such as:

- Central limit theorem
- Kalman filter
- Bayesian networks
- Dempster-Shafer
- Convolutional neural network

Here we present a few recent advances in state-of-the-art in sensor fusion and adaptation in the context of unmanned vehicles and robotics.

Manzanilla et al. [Manzanilla'2019] used extended Kalman filter (EKF) to fuse visual information with data from an inertial measurement unit, in order to recover the scale of the map and improve the pose estimation. A proportional integral derivative controller with compensation of the restoring forces was proposed to accomplish trajectory tracking, where a pressure sensor and a magnetometer provided feedback for depth control and yaw, respectively, while the remaining states were provided by the EKF.

Research presented in [Devassykutty'2018] addresses an unconventional approach for localizing underwater robots through sensor fusion and inverted Long Baseline (LBL) method. The output of the localization algorithm is combined with Inertial Navigation System (INS) readings by an unscented Kalman Filter.

Valenti et al. [Valenti'18] presents a combination of techniques enabling vision-based autonomous Unmanned Aerial Vehicle (UAV) systems. Various computer vision processing algorithms are used to exploit visual information to simultaneously perceive obstacles and refine localization in a GPS-denied environment. An omni-directional stereo-vision based setup is used to build a 3D representation of the surroundings.

The use of multiple visual sensors, which produce images of a scene taken at different viewpoints requires representing them first in the common plane. This process, referred to as image rectification, is needed before fusing the information from different sources.

Current approaches to image rectification are mainly based on keypoints matching for un-calibrated cameras, and chessboard rectification for the calibrated case. These methods do not always guarantee good results and the rectification process often needs to be repeated. This in turn makes the process time consuming and to some degree manual.

Ciurea et al. [Ciurea'16] proposed a method to automatically detect and correct for errors in geometric calibration in camera arrays. They exploit the redundancy of a camera array system to recover from the variation of calibrated parameters.

A new rectification algorithm was proposed for uncalibrated stereo images by Ko et al. [Ko'17]. It adopts a generalized homography model to reduce the rectification error and incorporates several practical geometric distortions in the cost function as regularization terms, which prevent severe perspective distortions in rectified images.

5.2. CERBERO Advances in Sensor Adaptation

5.2.1 Adaptive Rectification for Visual Sensors

Regarding CERBERO and advances in sensor adaptation, the current work focuses on adaptive image rectification method based on the keypoints matching approach.

The goal is to adaptively find more and better keypoints matches to continuously improve the rectification results over time. This would allow for the self-correction process in the case when the viewpoints of the camera sensors change. This could be due to the physical stress or damage the sensors may sustain as a result of a rough landing on Mars surface, or an electric car or underwater vehicle collision with another object.

The proposed approach is as follows. We assume that the scene is changing so the new keypoints can be found over a certain time period.

(1) Keypoints detection. Detect a number of characteristic points in the images (keypoints) using a detector, e.g. Harris corner detector.
(2) Keypoints description. Areas around keypoints can be described with one of descriptors e.g. Scale Invariant Feature Transform descriptor.
(3) Keypoints matching. Search for the corresponding patches can be performed by similarity/dissimilarity measurement between descriptors e.g. Euclidean, Manhattan, cosine of the angle.
(4) Removal of outliers. Apply Random Sample Consensus method to remove outliers.
(5) Transformation to common plane. The displacement vector for each feature can be used to find the transformation to the common plane.
(6) Average disparity and geometric distortions as a measure of rectification quality. Calculate the average pixel-wise disparity and magnitude of image distortions (dissimilarity between rectified and original image) as a measure of the quality of image rectification.
(7) Select matched keypoints which minimize disparity and the magnitude of geometric distortions. Choose a minimum number of matches needed for perspective transformation based on the similarity measurement.
(8) Detect new keypoints and adaptively improve the rectification results by repeating the procedure until the desired rectification quality.

The KPI measured by performance monitor is the average disparity between the images produced by the camera sensors, and the magnitude of the geometric distortions which is the dissimilarity between rectified and original image. If the KPI is above a predefined threshold believed to be the desired disparity and magnitude of distortions the adaptation manager triggers adaptation.

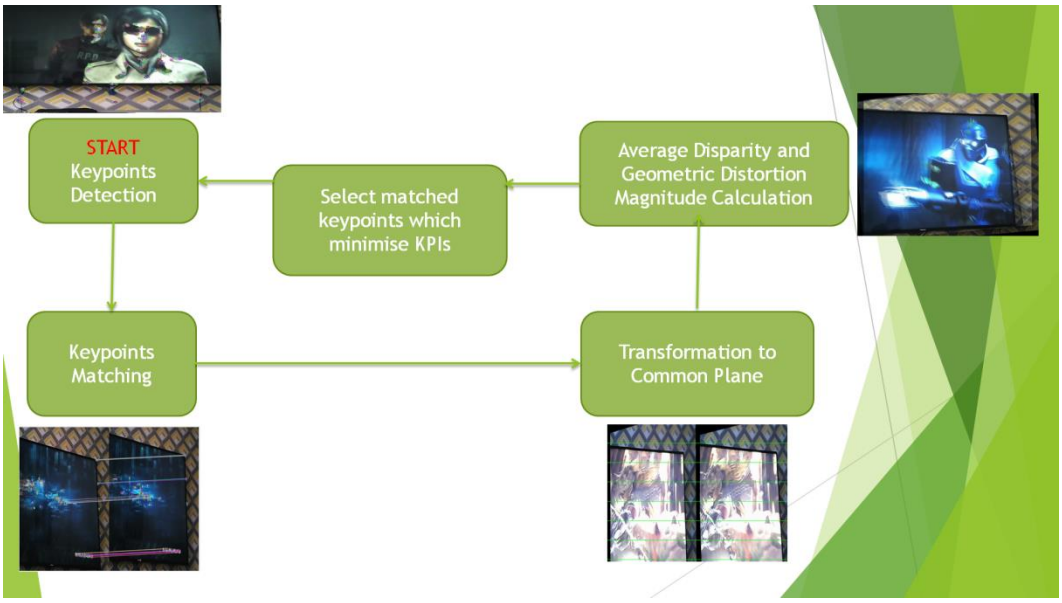

Figure 6 – New Adaptive Image Rectification

5.2.2 Energy-optimized Sensor-based Adaptation

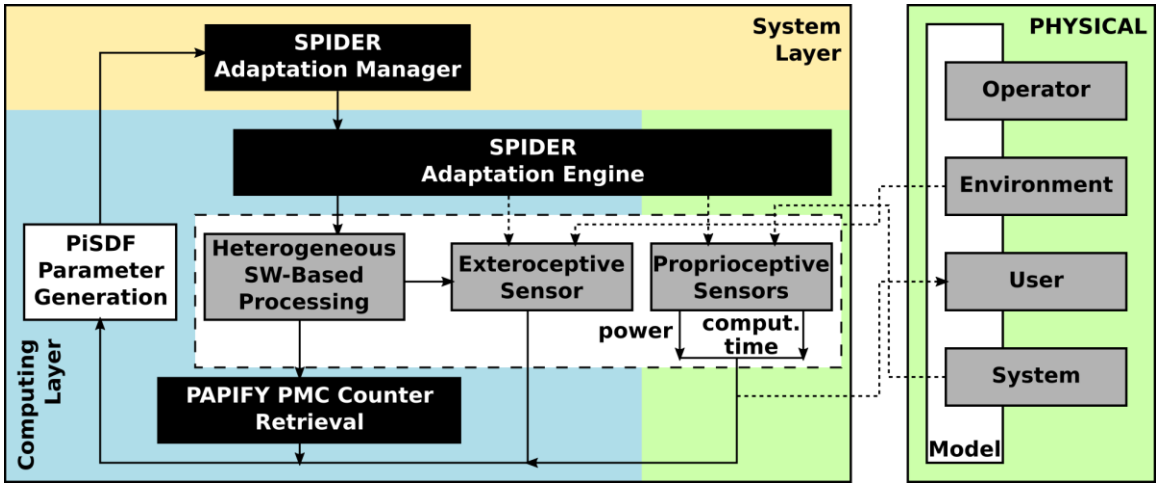

Figure 7 – Sensor-Based Adaptation of Heterogeneous SW

Experiments have been conducted with PREESM and SPIDER tools, together with the PAPIFY tool, to demonstrate energy-optimized sensor-based adaptive software on a heterogeneous platform. The experimental setup, based on the CERBERO software adaptation strategy, is illustrated in Figure 7. The SPIDER adaptation manager, handling the parallel actors composing a PiSDF modeled application, is fed with constantly modified parameter values. These parameters affect the amount of processing load of the system, as well as the instantaneous parallelism. The SPIDER adaptation manager triggers the SPIDER adaptation engine that updates memory allocation for communications and processing elements allocation for computation. The managed application, displayed in white with dashed borders, itself pilots two types of sensors:

Exteroceptive sensors, retrieving the environment state,

Proprioceptive sensors, retrieving the system state.

In current experiments, a camera is used as an exteroceptive sensor and power and time sensors are used as proprioceptive sensors. These informations are fed back to the SPIDER adaptation manager for influencing heterogeneous SW management.

The next table shows experimental results on the energy consumption measured per frame for an image filter running on an 8-core ARM big.LITTLE architecture managed by SPIDER. Two SPIDER cases are considered: with or without platform heterogeneity knowledge. SPIDER can also reconfigure between two application configurations. Having knowledge on the platform heterogeneity during re-mapping is demonstrated on the example to save between 16% and 40% of energy, while SPIDER manages to reduce energy when filtering effort is reduced. Code, tutorials and documentation for this experimentation are available on preesm.org. A video detailing the results is available at: http://youtu.be/a9WIucWfjkU. This setup has been demonstrated in our University Booth at DATE 2019.

Algorithm	Computational energy with platform heterogeneity information	Computational energy without platform heterogeneity information
Full image filter	2.35 J/frame	> 4 J/frame
Reduced image filter	1.92 J/frame	2.3 J/frame

Table – Energy Consumption of an Image Filtering Managed by SPIDER on an 8-core Heterogeneous Processor.

6. Adaptation Strategies in Use Cases

This section addresses the main adaptation characteristics that will be included in the different use cases of the project.

6.1. Adaptation Strategies in Planetary Exploration UC

The Planetary Exploration use case focuses on hardware adaptation strategies to ensure the success of the robotic exploration mission. In this uncertain environment, different flavours of functional and architectural reconfiguration can be triggered.

- **Error detection and correction**: the computing fabric will be able to recover from system malfunctions produced by radiation failures. When incorrect behavior is detected in one of the accelerators, this slot is reconfigured by ARTICo^3 in order to return to its correct operation. Functional fault injection will be used to test this feature, as explained on D2.1.

- **Adaptive degree of ruggedization**: depending on an estimation of the current radiation rate, ARTICo^3 is capable of implementing different redundancy techniques (DMR, TMR) or even transferring critical execution tasks to software if the stringent conditions of the mission require it.

- **Optimization of computing consumption**: diverse approaches of the motion planning algorithm are being taken into account. The ARTICo^3/MDC toolchain will generate a CGR architecture with the available implementations of the accelerator to enable fast-switching between the different power-demanding algorithms, depending on the current operating consumption or the battery levels of the spacecraft.

- **Minimizing power consumption in the actuators**: Besides the power consumption optimization of the computing platform, the energy consumed by the actuator joints must be minimized as well. This will be done by generating a set of trajectories from different random seeds for each interpolation step, and selecting the most power efficient one from the output of an energy model of the arm.

- **Adaptation to physical environment**: the robotic arm will be equipped with a time-of-flight proximity sensor based on infrared laser in order to enable unknown obstacle detection. When the arm gets too close to a physical object, the sensor reading will be below the defined threshold, and the selected trajectory will modified in order to avoid the obstacle and safely reach the final position.

In addition to enabling the description of both the algorithm and architecture, the PREESM tool provides automatic code instrumentation capabilities thanks to its integration with PAPIFY. The hardware and software KPIs measured from the performance monitors are fed to the Adaptation Manager, which evaluates the system itself and its environment and takes decisions, closing the adaptation loop. The scheduling of the system is performed by SPIDER at runtime in order to dynamically

manage the available hardware and software resources dedicated to the execution of the threads.

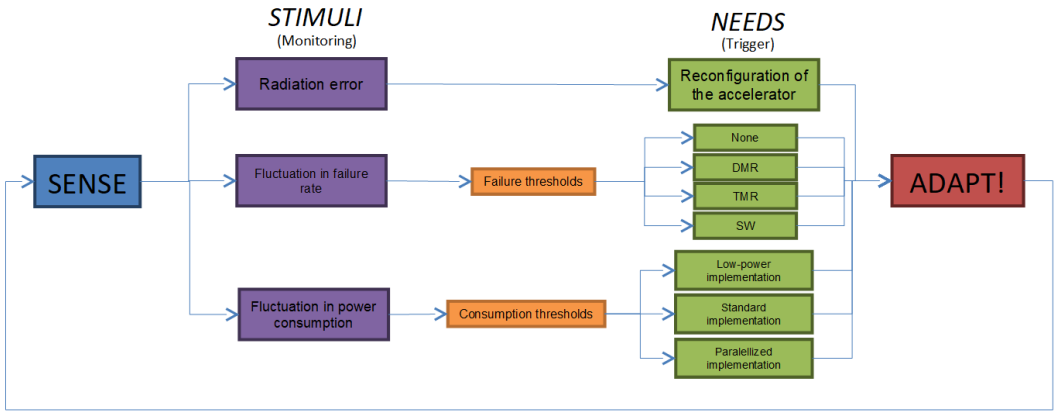

Figure 8 - ***Simplified Planetary Exploration adaptation diagram***

6.2. Adaptation Strategies in Ocean Monitoring UC

Ocean Monitoring employs various adaptation strategies in order to improve its performance. The adaptation approaches follow the CERBERO adaptation loop with monitors measuring KPIs and adaptation manager taking decisions about when to adapt. OM represents user commanded, environment triggered, and context aware adaptation. The following adaptation approaches are present in OM use case:

- Context aware adaptation when pairs of lenses are chosen for different purposes. Different camera functionalities use different fusion strategies.
- Environment triggered adaptation when image brightness is automatically adapted to changing lighting conditions.
- User commanded adaptation when the user changes the level of image enhancement to improve his/her situational awareness or the capabilities of computer vision methods.
- Adaptive Hybrid Image Retrieval Model where the strength between each query and its corresponding context is measured and used to dynamically adjust the weights associated with visual and textual context features.
- Adaptation of the image rectification approach to progressively improve the rectification results. This is an ongoing work.

6.3. Adaptation Strategies in Smart Travelling UC

In the Smart Travelling use case the following adaptation strategies will be applied:

- **Self-adaptation through the decision support function:** The decision support function in the Smart Travelling use case acts as the Self-Adaptation Manager, responsible for any required adaptation relevant for the driver. Like a software

agent the manager will follow the 3T model and follow the sense, plan and act paradigm using triggers from the car and its environment.

- **Time synchronisation:** In order for DynAA to work as a system in the loop, adaptation are included to connect and synchronise real time sensors data to the simulations run inside DynAA.
- **Fusion of time-series sensor data:** In order to analyse the results of the simulation runs, pre-processed data from all the sensors and simulation modules are fused to ingle synchronized time-series data set using the data fusion tool.
- **Parallel processing:** to keep response times of optimization processes within reasonable levels, a set of parallel simulation environments were set up to spread the computational load.

In order to provide immediate location feedback to the driver, an additional Human Machine Interface (HMI) was added to the demonstrator which will directly receive triggers from the SCANeR simulator to minimize delays. The triggers will be used to plot the car on a map, show information like battery status and visually track the executed route. The received triggers will be forwarded from the HMI to MECA for status monitoring of the car. MECA will act as the adaptation manager and use the HMI to instruct the driver to follow or adapt the route based on status of the environment, the car or the driver him-/herself. The new HMI will also be used to communicate suggested adaptations from MECA to the driver and receive input from the driver on choices and specific preferences.

To detect the status of the driver (tiredness level) special sensors are added to the simulator capable of monitoring eye and eye lid movement, which indicates the tiredness of the driver. MECA will use signals from this sensors to determine appropriate actions, like warn the driver or suggest rest or charging at next station.

Cyber-physical systems typically require time synchronization methods in order for the ICT (simulated) component of the system to keep in line with the physical component. A *plugin* was developed for the DynAA simulation tool that enables this synchronization in a fashion similar to some co-simulation strategies [Nicolescu'07][Gomes'17] (and in fact was tested with exactly this application). This plugin provides both an input as well as output SPI. Developers using this SPI have to provide certain interfaces for their sensors/actuators which then subsequently can be embedded in a DynAA model. When implementing these time-synchronous methods we categorize them into:

	Input to simulation	Output from simulation
Cyber is faster than physical	Hold simulation until relevant input is available	Buffer output until real world caught up with simulation
Physical is faster than cyber	Buffer input until simulation caught up with real world	**Problems are here**

The first three categories are not problematic (assuming the simulation can be paused), since a simple solution exists in either pausing the simulation or buffering the input/output. These solutions are provided to the user in the form of some helper functions that implement the buffering and/or halting when necessary. However the real world cannot be paused, so in the final category where the simulation has some output, but is lagging behind the physical world a problem arises. Here we provide two possible ways out, and leave the choice open for the modeler: either the actuator throws an exception, leading to a failed simulation meaning that the modeler should simplify his model or run the experiment on more potent hardware. The other solution is to accept the lag, which under some conditions may be acceptable, for instance when the output is only signaling to the physical world by means of a flashing LED, or a display on a monitor. There is potentially a third column, in which there is output to the physical world, which eventually leads to input. In this situation accepting the delayed output is probably erratic, so one has to make sure the simulation outpaces the real world.

7. References

[Augonnet'11] Augonnet, C., Thibault, S., Namyst, R., & Wacrenier, P. A. (2011). StarPU: a unified platform for task scheduling on heterogeneous multicore architectures. Concurrency and Computation: Practice and Experience, 23(2), 187-198.

[Bhattacharyya'00] Bhattacharya, B., & Bhattacharyya, S. S. (2000). Quasi-static scheduling of reconfigurable dataflow graphs for DSP systems. In Proceedings 11th International Workshop on Rapid System Prototyping. RSP 2000. Shortening the Path from Specification to Prototype (Cat. No. PR00668) (pp. 84-89). IEEE.

[Bertoni'05] Guido Bertoni, Luca Breveglieri, Israel Koren, Paolo Maistri, Vincenzo Piuri:
A Parity Code Based Fault Detection for an Implementation of the Advanced Encryption Standard. DFT 2002: 51-59

[Caesar'17] CAESAR competition, https://competitions.cr.yp.to/index.html

[CERBERO'17] http://www.cerbero-h2020.eu

[Ciurea'16] Ciurea, F., Lelescu, D., Chatterjee, P. and Venkataraman, K., 2016. Adaptive Geometric Calibration Correction for Camera Array. *Electronic Imaging*, *2016*(13), pp.1-6.

[Damavandpeyma'13] Damavandpeyma, M., Stuijk, S., Basten, T., Geilen, M., & Corporaal, H. (2013). Schedule-extended synchronous dataflow graphs. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 32(10), 1495-1508.

[Devassykutty'18] Edwin Devassykutty and Gunnar Brink. 2018. Evaluation of High Precision Localization Approach for a Fleet of Unmanned Deep Ocean Vehicles. In *Proceedings of the 2nd International Symposium on Computer Science and Intelligent Control* (ISCSIC '18). ACM, New York, NY, USA, Article 35, 6 pages.

[De Dinechin'13] De Dinechin, B. D., de Massas, P. G., Lager, G., Léger, C., Orgogozo, B., Reybert, J., & Strudel, T. (2013). A distributed run-time environment for the kalray mppa®-256 integrated manycore processor. Procedia Computer Science, 18, 1654-1663.

[Desnos'13] Desnos, K., Pelcat, M., Nezan, J. F., Bhattacharyya, S. S., & Aridhi, S. (2013, July). Pimm: Parameterized and interfaced dataflow meta-model for mpsocs runtime reconfiguration. In 2013 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS) (pp. 41-48). IEEE.

[Dworkin'01] Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, https://csrc.nist.gov/publications/detail/sp/800-38d/final

Dworkin'07] Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, https://csrc.nist.gov/publications/detail/sp/800-38d/final

[Fanni'18] Tiziana Fanni, Alfonso Rodríguez, Carlo Sau, Leonardo Suriano, Francesca Palumbo, Luigi Raffo and Eduardo de la Torre, "Multi-Grain Reconfiguration for Advanced Adaptivity in Cyber-Physical Systems". 2018 International Conference on ReConFigurable Computing and FPGAs (ReConFig'18). December 2018. doi: 10.1109/RECONFIG.2018.8641705

[Fradet 2012] Fradet, P., Girault, A., & Poplavko, P. (2012, March). SPDF: A schedulable parametric data-flow MoC. In Proceedings of the Conference on Design, Automation and Test in Europe (pp. 769-774). EDA Consortium.

[Fuchs'18] C. M. Fuchs, N. M. Murillo, A. Plaat, E. van der Kouwe and P. Wang, "Towards Affordable Fault-Tolerant Nanosatellite Computing with Commodity Hardware," *2018 IEEE 27th Asian Test Symposium (ATS)*, Hefei, 2018, pp. 127-132. doi: 10.1109/ATS.2018.00034

[Gautier'13] Gautier, T., Lima, J. V., Maillard, N., & Raffin, B. (2013, May). Xkaapi: A runtime system for data-flow task programming on heterogeneous architectures. In 2013 IEEE 27th International Symposium on Parallel and Distributed Processing (pp. 1299-1308). IEEE.

[Gomes'17] Gomes, C., Thule, C., Broman, D., Larsen, P. G., & Vangheluwe, H. (2017). Co-simulation: State of the art. *arXiv preprint arXiv:1702.00686*.

[Heulot'14] Heulot, J., Pelcat, M., Desnos, K., Nezan, J. F., & Aridhi, S. (2014, September). Spider: A synchronous parameterized and interfaced dataflow-based rtos for multicore dsps. In 2014 6th European Embedded Design in Education and Research Conference (EDERC) (pp. 167-171). IEEE.

[Keinert 2006] Keinert, J., Haubelt, C., & Teich, J. (2006, May). Modeling and analysis of windowed synchronous algorithms. In 2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings (Vol. 3, pp. III-III). IEEE.

[Ko'17] Ko, H., Shim, H.S., Choi, O. and Kuo, C.C.J., 2017. Robust uncalibrated stereo rectification with constrained geometric distortions (USR-CGD). *Image and Vision Computing*, *60*, pp.98-114.

[Lee 1987] Lee, E. A., & Messerschmitt, D. G. (1987). Synchronous data flow. Proceedings of the IEEE, 75(9), 1235-1245.

[Le-Lann'18] J.C. Le Lann, T. Bollengier, M. Najem and L. Lagadec, "An Integrated Toolchain for Overlay-centric System-on-chip," *2018 13th International Symposium on Reconfigurable Communication-centric Systems-on-Chip (ReCoSoC)*, Lille, 2018, pp. 1-8. doi: 10.1109/ReCoSoC.2018.8449388

[Liu'18] L. Liu, Z. Li, C. Yang, C. Deng, S. Yin, and S.Wei. "HReA: An energy-efficient embedded dynamically reconfigurable fabric for 13-dwarfs processing". *IEEE Transactions on Circuits and Systems II: Express Briefs, 65(3):381–385, March 2018*.

[Manzanilla'19] A. Manzanilla, S. Reyes, M. Garcia, D. Mercado and R. Lozano, "Autonomous Navigation for Unmanned Underwater Vehicles: Real-Time Experiments Using Computer Vision," in *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1351-1356, April 2019

[Merchant'10] S. G. Merchant and G. D. Peterson, "Evolvable block-based neural network design for applications in dynamic environments," VLSI Des., vol. 2010, 2010.

[Nicolescu'07] Nicolescu, G., Boucheneb, H., Gheorghe, L., & Bouchhima, F. (2007). Methodology for efficient design of continuous/discrete-events co-simulation tools. *High Level Simulation Languages and Applications-HLSLA. SCS, San Diego, CA*, 172-179.

[Piat'09] Piat, J., Bhattacharyya, S. S., & Raulet, M. (2009, October). Interface-based hierarchy for synchronous data-flow graphs. In 2009 IEEE Workshop on Signal Processing Systems (pp. 145-150). IEEE.

[Pino'95] Pino, J. L., Bhattacharyya, S. S., & Lee, E. A. (1995). A hierarchical multiprocessor scheduling framework for synchronous dataflow graphs.

Electronics Research Laboratory, College of Engineering, University of California.

[Rainey 2014] Rainey, E., Villarreal, J., Dedeoglu, G., Pulli, K., Lepley, T., & Brill, F. (2014). Addressing system-level optimization with OpenVX graphs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 644-649).

[Regazzoni'12] Francesco Regazzoni, Luca Breveglieri, Paolo Ienne, and Israel Koren. Interaction between fault attack countermeasures and the resistance against power analysis attacks. In Marc Joye and Michael Tunstall, editors, Fault Analysis in Cryptography, chapter 15, pages 257–72. Springer, 2012

[Ritz'93] Ritz, S., Pankert, M., Zivojinovic, V., & Meyr, H. (1993, October). Optimum vectorization of scalable synchronous dataflow graphs. In Proceedings of International Conference on Application Specific Array Processors (ASAP'93) (pp. 285-296). IEEE.

[Rodríguez'18] Alfonso Rodríguez and Tiziana FANNI, *DEMO: "*Multi-Grain Adaptivity in Cyber-Physical Systems". *Special Session on Energy Efficient Cyber Physical Systems held at the 30th International Conference on Microelectronics (ICM'18). December 2018. Proceedings in press.*

[Topcuoglu'02] Topcuoglu, H., Hariri, S., & Wu, M. Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE transactions on parallel and distributed systems, 13(3), 260-274.

[Valenti'18] F. Valenti, D. Giaquinto, L. Musto, A. Zinelli, M. Bertozzi and A. Broggi, "Enabling Computer Vision-Based Autonomous Navigation for Unmanned Aerial Vehicles in Cluttered GPS-Denied Environments," *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, Maui, HI, 2018, pp. 3886-3891.

[Wu'18] Wu, J., Blattner, T., Keyrouz, W., & Bhattacharyya, S. S. (2018, March). A design tool for high performance image processing on multicore platforms. In 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE) (pp. 1304-1309). IEEE.

[Zaki'17] Zaki, G. F., Plishker, W., Bhattacharyya, S. S., & Fruth, F. (2017). Implementation, scheduling, and adaptation of partial expansion graphs on multicore platforms. Journal of Signal Processing Systems, 87(1), 107-125.

[Zamacola'18] R. Zamacola, A. G. Martínez, J. Mora, A. Otero, and E. D. La Torre, "IMPRESS : Automated Tool for the Implementation of Highly Flexible Partial Reconfigurable Systems with Xilinx Vivado," 2018.