ARGO

WCET-Aware Parallelization of Model-Based Applications
for Heterogeneous Parallel Systems

H2020-ICT-2015

Project Number: 688131

Deliverable D6.1

D6.1 ARGO State of The Art

Editors:	Pierre-Aimé Agnel
Authors:	M. Bednara (IIS), U. Durak (DLR), C. David (SCI), PA Agnel (SCI), S. Derrien (UR1), D. Hardy (UR1), A. Kritikakou (UR1), I. Puaut (UR1), N. Voros (TWG), M. Katsimpris (TWG), P. Alefragis (TWG), G. Theodoridis (TWG), S. Reder (KIT), L. Masing (KIT), H. Bucher (KIT), J. Becker (KIT), G. Rauwerda (RS), K. Sunesen (RS)
Version:	v1.00
Status:	FINAL
Dissemination level:	Public (PU)
Filename:	D6.1_ARGO_StateOfTheArt.docx

ARGO Consortium

Karlsruhe Institute of Technology	DE
Scilab Enterprises	FR
Recore Systems B.V.	NL
Université de Rennes I	FR
Technological Educational Institute of Western Greece	GR
AbsInt Angewandte Informatik GmbH	DE
Deutsches Zentrum für Luft- und Raumfahrt	DE
Fraunhofer IIS	DE

Document revision history

Version	Based on	Date	Author	Comments / Changes
V1.00		13.04.2016	Pierre-Aimé Agnel	Initial version

Table of Contents

1. Abstract

This document summarizes the State of The Art for project European H2020 project ARGO.

The ARGO toolchain presented in Figure 1 is a cross-layer programming approach for heterogeneous and manycore systems in a model based workflow. The objective is to design, implement and deploy hard real-time applications on multi-core targets through parallel code generation with top-notch Worst Case Execution Time (WCET) analysis in a programming environment that will guarantee efficiency and productivity.

The model-based design environment allows engineers to design a system from a high-level point of view. Code generation and code transformations are performed with a strong objective of keeping the code base predictable or warning the user as early as possible of possible problems in WCET estimation in the current design. The targeted architecture, defined with an Architecture Description Language, and specific low level transformations ensure parallelization with WCET constraints as tight as possible.

Constant feedback is provided to the user and the possibility to select at each step the transformations to consider and perform in an interactive manner enables a semi-automatic, guided process. The models are enriched with the results of the code generation and the real time constraints analysis, thus tracing and controlling the results of an iteration of the process for early verification and validation

This state of the art document is designed to show, at each step of the process, how the the project consortium, with a strong academic and industrial background, benefit from the existing breakthroughs in – and how it will contribute to – the WCET analysis and code generation for heterogeneous multi-core architectures in a model based design workflow.

In order to cover the state of the art and the state of the practice, the sections are designed to relate to each step of the process from the high level requirements to the low level implementations and optimizations performed. First we explore, with the illustration of the project use cases, the current expectations and practices from the end-user perspective. Then the existing model based design environments are analysed with regard to the conception of hard real-time parallel and distributed applications deployed on multi-core architectures. The third section focuses on predictability enhancement and parallel code generation keeping the WCET constraints while providing improvements on the performance of executed code. The section explores both the coarse grain and the fine grain enhancement techniques with regard to the machine code to produce. Finally, the last section focuses on the hardware specific issues with regard to data placement, communication and input/output.

The sections are thus presented from the high level (functional requirements) to the low level (hardware and network layers) as follows:

Section 2: Presentation of the Use Cases

The ARGO uses cases are presented first to emphasize on the real-time objectives of the project. They present the current approach in generating code for multi-processor architectures from the end-user perspective and they encompass the full tool-chain in the functional aspect.

Section 3: Model Based Design to Parallel Code Generation Toolchain

This section presents the high level tools to model systems and how the current practices cope with the link between the functional models and the multi-core code generation with WCET constraints.

Section 4: Worst Case Execution Time Toolchain

The various approaches to generate code for a multi-processor target, code annotation and modification to improve its predictability are presented in this section. The innovation brought by the ARGO project in comparison with the state-of-the-art approaches is also examined in this section.

Section 5: Memory and communication optimization in time-predictable multi-core architectures

This section presents the current considerations for I/Os and memory management in WCET analysis of application for multi-core architecture.

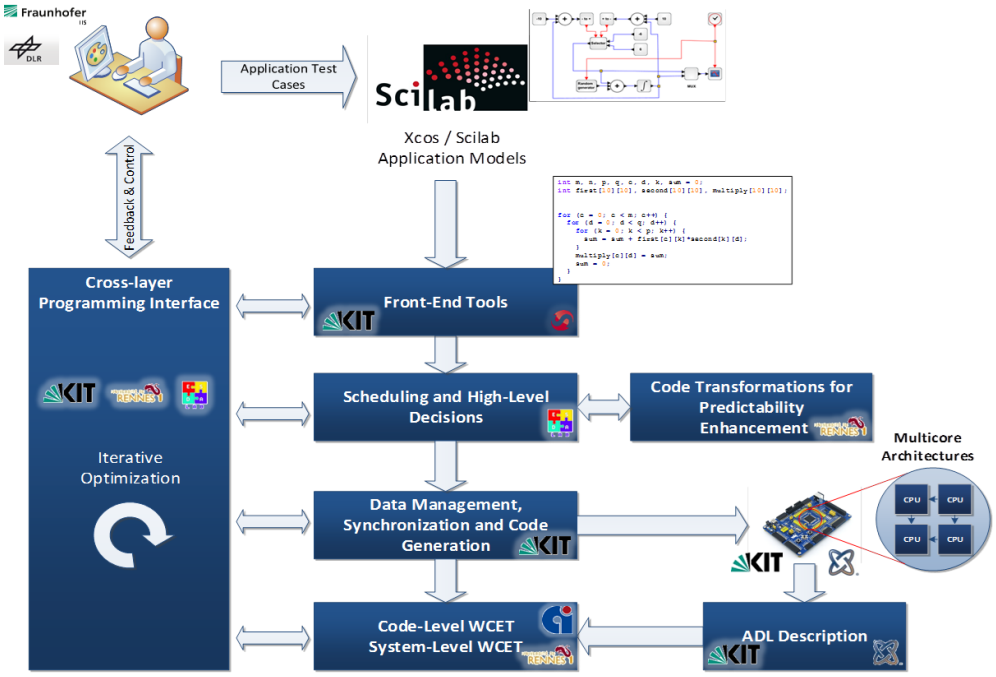

Figure 1: ARGO tool flow

2. Presentation of the Use Cases (Fraunhofer IIS)

The following description of the use-cases shows the importance of the WCET analysis across the different phases of conception until the implementation of a system.

2.1 Polarization Image Processing

The task of Fraunhofer IIS within the ARGOARGO project is to provide an application for the ARGOARGO toolchain that is to be used as a test case. This application will be a specialized image processing system for image data originating from a novel polarization image sensor developed at Fraunhofer IIS [1] Polarization image data is significantly different from 'traditional' (i.e. color) image data and requires widely different – and significantly more computation intensive - processing operations.

Amongst others, polarization imaging is used in the area of industrial applications e.g. for quality control in glass production [2] and carbon fiber treatment [3]. The industrial requirements on the quality control are extremely strict and no error in the classification of the produced glass is tolerated. . It is important to notice that a polarization camera must be considered as a measuring instrument rather than a source for visual images. This leads to significantly higher performance requirements than in a conventional camera system. Moreover, a new version of the polarization sensor with a much higher resolution is planned for the near future. Since each frame must be processed in the predefined time frame, we have hard real-time constraints here.

In this section, we summarize how the polarization image processing system is currently implemented and point out how we expect to benefit from the ARGOARGO work flow.

For providing a basic impression of Fraunhofer IIS use case current algorithm, we start with a sketch of polarization image processing.

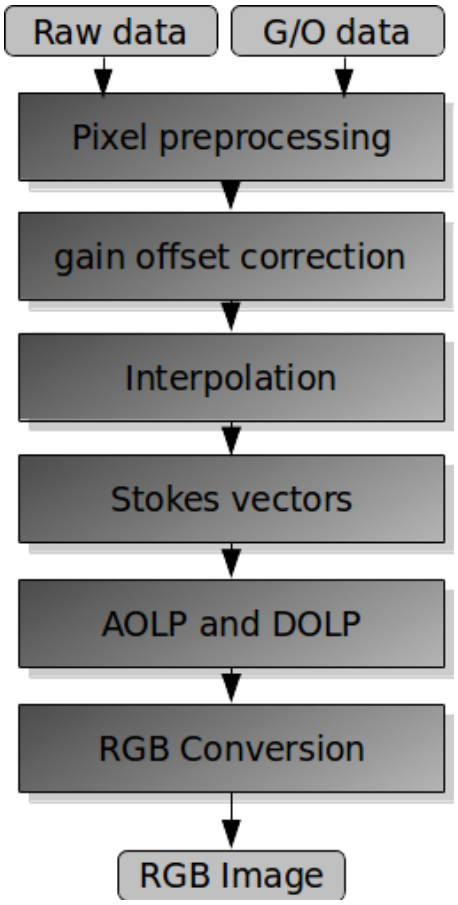

Figure 2: Polarization image processing

The sensor *raw data* first is *preprocessed* in a very basic way. Actually, this is just a pixel reordering. Afterwards, a *gain/offset correction* is performed on each pixel to equalize sensitivity and linearity inhomogeneities. For this purpose, additional gain/offset data is required (G/O input in Figure 2).

Since each pixel only provides a part of the polarization information of the incoming photons, the unavailable information is *interpolated* from the surrounding pixels (similar to Bayer pattern interpolation on color image sensors). From the interpolated pixel values we now compute the *Stokes vector*, which is a vector that provides the complete polarization information of each pixel. By appropriate transformations, the Stokes vectors are converted into the degree of linear polarization (*DOLP*) and angle of linear polarization (*AOLP*). These parameters are usually the starting point for any further application dependent processing (which is not shown here). For demonstration purposes, we convert AOLP and DOLP into a *RGB* color image that can be used for visualizing polarization effect.

It should be noted that this is an outline of our current algorithm where we do not have any dependencies between runtime and data. In the future there will be more complex and data dependent functions such that the WCET aspect of the ARGO project will become more relevant.

Currently there are three different realizations of this polarization image workflow:

- A single core C++ implementation on a PC with Windows operating system;
- An experimental implementation on an embedded image processing platform;

- OpenCL (***Open C****omputing* ***L****anguage*) and IPP (*Integrated Performance Primitives*) based implementations on a PC using a graphics adapter.

These implementations will be characterized in the following subsections.

2.1.1 Single Core C++ Implementation on x86 PC

The design and development of the correction algorithm has been performed with MATLAB R2015b using the Image Processing Toolbox for visualization, reading and writing images as well as for analyzing interim results. First, we set up an offline simulation for single frames and afterwards the algorithm is verified for continuous streaming with data directly being captured from the polarization camera by utilizing MATLAB's Image Acquisition Toolbox.

As shown in Figure 2, the algorithm is composed of multiple steps which have to be executed in a well-defined order. To simplify the development process a straightforward the implementation consiting in three different steps as follows:

- each of the algorithmic steps is covered by a single MATLAB function;
- every function is a frame transformation of an input image X to an output image X';
- the functions are executed sequentially.

This setup enabled us to analyze and improve the algorithm step by step. Once the development has been finished the MATLAB sources are manually ported to C++. In this initial phase no improvements were made concerning performance aspects. The Code was compiled using Microsoft Visual Studio 2015 with optimization for speed (-O2 compilation flag).

Neither the MATLAB simulation nor the C++ implementation are currently currently capable to operate in real-time.

2.1.2 Experimental Embedded Image Processing Platform

Currently the main algorithm is performed on PC based platforms, but for the future it is planned to integrate more functionality into the polarization camera itself. A first step towards this integration was to implement the complete image processing pipeline on an experimental platform called VEMPIRE (Versatile Embedded Platform for Image Recognition) that was designed as evaluation system at Fraunhofer IIS. The purpose of this step was to get an estimation of the processing capacity required to achieve hard real time behavior on an embedded system.

VEMPIRE mainly comprises a mid-sized FPGA and a multi-core digital signal processor. A small portion of the sensor data pre-processing is done in the FPGA, but the main part is implemented in DSP software. We do not use any parallelizing compiler like OpenMP (which was not available for this target platform), so all parallelization and core communication was designed by hand, which is an very time-consuming and error-prone task.

The embedded implementation of the polarization image processing pipeline is performed roughly in the following steps:

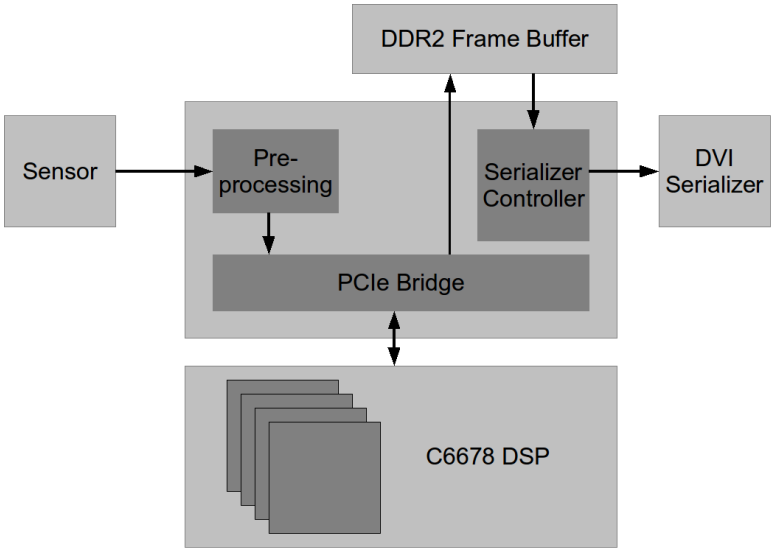

Figure 3 Architecture of the embedded image processing system VEMPIRE

- Compiling a plain C implementation for the embedded target platform. The C code was based on the PC version but amended with some architectural modifications, especially w.r.t. to interfaces. The result was an operational system with an extremely poor performance.
- The next task was to optimize the code in order to achieve a better performance. The optimization techniques applied at this step can be categorized as follows:
  - *Algorithmic optimizations:* How can the algorithm itself be optimized while ignoring technical aspects of the underlying hardware platform? Subject to optimization are usually minimization of memory accesses, simplification of arithmetic operations, reduction of loop iterations, and generally degrading the algorithmic complexity if possible (e.g. the degree of polynomial complexity);
  - *Platform specific optimizations:* Here the details of the target architecture are taken into account, so a deep knowledge of the underlying platform and the real time operating system is required. These techniques focus on cache optimization, loop unrolling, usage of machine intrinsics or other specialized co-processor blocks, utilization of the maximum operand width, instruction level parallelization in SIMD-architectures, and aspects of scheduling and memory management;
  - *Multi-core parallelization:* This aspect is covered below.

By applying the above mentioned optimization strategies, we could increase the performance of the embedded implementation by a factor of 8 approximately while still running on a single core. It should be noted that particularly the platform specific optimizations require substantial modifications of the original source code. This leads to a highly optimized code at the cost of platform-dependency, poor portability, and a much higher effort for maintenance and documentation.

- Finally, the algorithm was parallelized for the multi-core DSP. Multi-core parallelization without support from a parallelizing compiler requires deep modifications of the source code. The mapping of tasks and processes to particular DSP cores must be specified explicitly as well as the communication and synchronization between the cores.

The image processing functions operate mainly on a local pixel proximity; there are (so far) no global operators. This allows for exploiting the inherent data parallelism here. The polarization image is subdivided into sections of approximately identical size, and each section is mapped to a DSP core. This way, inter-core communication is minimized, since all cores perform the most of the time identical functions on different data sets. For future image processing tasks, this may not hold anymore. For example, the blob detection usually requires an irregular scan pattern of the whole image and an intensive data-dependent core communication.

We could reduce the effort for parallelization by developing a software framework that provides a communication infrastructure that can be reused for all those image processing tasks that require communication only at the beginning and the end of a processing function, which is often the case. Moreover, the software framework defines a scheme for distributing jobs to the DSP cores and manages the core synchronization.

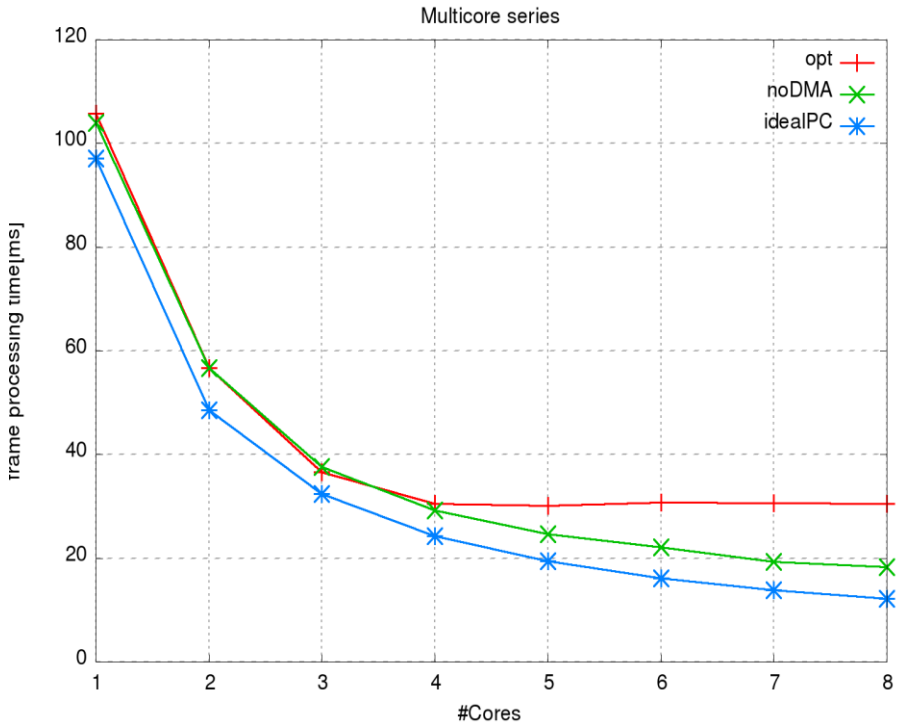

Figure 4 Impact of multi-core parallelization on the overall performance

As shown in Figure 4, we gain a considerable speedup of the system by parallelization. As one can see from the red curve, there is no more speedup when increasing the number of cores to more than 4 due to sensor bandwidth limitations. So there is ample room for bigger sensors or future processing functions. The green curve shows the speedup under the (theoretical) assumption that there is no effective bandwidth limitation in the sensor. The blue curve is an estimation of the speedup using multiple cores on a PC, where only the single core implementation was really implemented.

The details about the embedded implementation are described in [4].

2.1.3 OpenCL and IPP Based Implementation on x86 PC

Also for the PC implementation we are currently examining some architecture dependent optimizations. Compared to the plain C++ code, we could attain a significant performance speedup of about 2 by using Intel® Integrated Performance Primitives (Intel® IPP). This was achieved by identifying the parts of the algorithm taking the most computation time (using a profiling tool) and then replacing loops over the whole image with Intel IPP commands. Intel IPP makes use of special processor instructions (SSE SIMD instructions) and Intel multicore architecture.

In the current version, IPP does not employ any resources of the graphics adapter but only of the main CPU cores. In order to gain a further speedup, we have ported appropriate parts of the algorithm to OpenCL and expected to obtain a speedup by executing loops over the image pixels by the parallel processing units of the GPU. Unfortunately it turned out, that the speedup gained by parallelization is obscured by the interface latency while transferring image data from and to the shader cores. This may be overcome by using a more powerful graphics adapter, since the one we currently used comprises of only 10 shader cores and a slow interface.

2.1.4 Expected benefits from the ARGOARGO project

Software development for real time image processing often starts with a functional model written in a simulation language like Scilab or MATLAB. After finishing the algorithm development, the code is (as in our case) manually ported into C or C++ language providing a first executable that can run on the target architecture but may suffer from a low performance and usually fails matching any real time constraints. As we have seen from the above described PC and DSP multi-core implementations, now an expensive optimization process must follow. At the end we have complex software that is highly optimized for a single target architecture but has almost nothing in common with the original functional model.

The ARGO workflow as planned will provide a way to maintain the functional model during the whole development process but nevertheless end up with a target-optimized executable that matches all real time constraints. Instead of switching the programming language in between on hand-optimize the code, all parallelization and optimization is done automatically but with user interaction and control.

From our experience in the past with the development of real time image processing software, we expect a reduction in the magnitude of months for the whole software design process.

2.2 Enhanced Ground Proximity Warning System (DLR)

The task of DLR within the ARGO project is to provide application test cases for the ARGO toolchain. These applications will be from the flight systems domain and will aim at demonstrating the capabilities of the toolset for avionic systems. The application test cases will be an Enhanced Ground Proximity Warning System (EGPWS) and the Wake Encounter Advisory and Avoidance System (WEAA). The WEAA is a flight system concept of DLR for tactical small scale evasion from wake vortices to avoid hazardous wake encounters. Therefore, it is not suitable for a state of the art analysis. An EGPWS, however, is a readily available system in current aircraft. It provides alerts and warnings for obstacles and terrain

along the flight path utilizing high resolution terrain databases, Global Positioning System and other sensors. EGPWS is a current name that is used for Terrain Awareness and Warning Systems (TAWS) which aim to prevent controlled flight into the terrain. There are various TAWS options available in the market for various platforms in various configurations. Examples may include EGPSW from Honeywell [5], T2CAS from ACSS [6],LANDMARK™ from L3 [7] and TAWS from Universal Avionics [8]. A brief comparison of these systems and more can be found in [9]

In this section, we will first introduce the EGPWS based on the capabilities of the Honeywell MK V system [10] [11] on DLR's Advanced Technologies Research Aircraft (ATRA) A320 aircraft. This capability set will be the baseline for the application test case for ARGO. Then, the section will provide a discussion on the state-of-the art with regards to parallelization and multi-core architectures. There is no data available in open literature that provides accurate information about the development practices and hardware architecture of neither the EGPWS from Honeywell, nor other TAWS available in the market. However, common industry practices in avionics are applicable to these products. Thereby, we will provide a short review of efforts that target parallelization of avionics applications in multi-core architectures.

2.2.1 EGPWS

The core feature set of EGPWS is to create visual and aural warnings between 30ft to 2450 ft Above Ground Level (AGL) in order to avoid controlled flight into the terrain. These warnings are categorized in 5 modes:

Mode 1: Excessive Descent Rate provides alerts for excessive decent rates for all phases of flight.

Mode 2: Excessive Terrain Closure Rate provides alerts to protect the aircraft from impacting the ground when terrain is rising rapidly with respect to the aircraft.

Mode 3: Altitude Loss After Take-off provides alerts when significant altitude loss is detected after take-off or low altitude go around.

Mode 4: Unsafe Terrain Clearance provides alerts when there exists no sufficient terrain clearance regarding the phase of the flight, aircraft configuration and speed.

Mode 5: Excessive Deviation Below Glideslope provides alerts when the aircraft descends below the glideslope.

Additionally, it provides some enhanced functions based on the worldwide terrain database. These functions are:

Terrain Awareness Display (TAD): It provides an image of the surrounding terrain represented in various colors on Navigation Display.

Terrain Clearance Floor (TCF): It provides a low terrain warning during landing, and thus enhances the basic functions by providing alerts for the descent below a predefined "Terrain Clearance Floor" disregarding the aircraft configuration.

The trend in avionics architectures is shifting towards more central computing platforms which are categorized as IMA [12]. Rather than decentralized and dedicated computing cards, in IMA, multiple applications utilize the same computing card [13]. The operating system then allows the operation of independent application software in partitions, thus the safety requirements are addressed. Partitions are defined as isolated execution environments with separate sets of resources that guarantee resource availability and timing. EGPWS are also following this trend. While some of the systems like the EGPWS MK V from

Honeywell whose capabilities are introduced above follow federated avionics architecture and provide TAWS capabilities as a stand-alone Line Replaceable Unit (LRU), other systems like T2CAS from ACSS or Aircraft Environmental Surveillance System (AESS) from Honeywell [14] follow the IMA approach and provide TAWS with Traffic Collision Avoidance System (TCAS) or weather radar functionality. They share the central computing platform with other avionic applications in a partitioned architecture. Furthermore there are some recent efforts that target parallelization and utilization of multi-core architectures in IMA. The next section will introduce a review of these efforts by mostly concentrating on the projects that are related to ARGO. Moreover, one of these recent efforts reports an approach that targets parallelization and multi-core architectures for TAWS for helicopters.

2.2.2 Parallelization in Flight Systems Development using Multi-Core Architectures

This section will provide pointers to recent research projects and publications from these projects that endeavor to achieve parallelization in flight systems development using multi-core architectures. RECAMP, which was funded by the ARTEMIS research grant between 2010 and 2013, addressed the certification of mixed-criticality systems with applications also in avionic architectures. Aligned with that, the ARAMIS research project was funded by the German Federal Ministry for Education and Research between 2012 and 2015 in order to attack the utilization of multi-core architectures in safety-critical automobile, train and aircraft systems. In their work that is supported by these grants, Nowatsch and Paulitsch from EADS Innovation Works examined the utilization of multi-core systems in partitioned environments like IMA for running applications of different safety-criticality [15]. In 2013, Karray and Paulitsch from EADS Innovation Works with Koppenhöfer and Geiger from CASSIDIAN presented the non-functional requirements for the application of multi-core architecture that is developed within the scope of the ACROSS project which is also funded by the ARTEMIS grant for a degraded vision landing system for a helicopter. In HiPEAC Conference 2015, Koppenhöfer and Geiger presented Hybrid Avionics Integrated Architecture Demonstrator which was developed within the scope of an ongoing ARTEMIS project called EMC2 [16]. In their presentation, they presented a Helicopter Terrain Awareness and Warning System (HTAWS) as a sample application of their demonstrator. They aim at providing a comprehensive, map based overview of a helicopter´s surroundings to prevent avoidable collision with ground or obstacles.

In parallel with these projects, Agrou and colleagues from THALES presented design principles of predictable and efficient multi-core systems to meet embedded computer requirements in avionics [17]. In 2014, Löfwenmark from Saab Aeronautics and Nadjm-Tehrani from Linköping University presented challenges and described research directions to address guaranteeing determinism for avionic applications running on multiple cores and interacting through shared memory [18].

While these efforts reported initial results of parallelization in flight systems development using multi-core architectures, they do concentrate on the applicability regarding the safety constraints of the avionics domain. Nevertheless, there is no reported effort that attacks the development methodology and tries to provide an avionics application development approach for parallel multicore-core architectures.

3. Model Based Design to Parallel Code Generation Toolchain

This section focuses on the tools used to manage the functional requirements to code generation taking into account the targeted parallel architecture and time-constraint requirements.

The traditional top-down approach to system design is to use specialized software to first have a complete hardware / software architecture out of any defined algorithm logic. This approach can by performed using different methodologies and different modeling language ; for instance SysML and AADL can be used to describe tasks and parallelisms. VERTAF/Multi Core [19] is a practical methodology and toolset to real world system design re-using high-level parallel semantics on visual entities.

The usage of VERTAF/MC let the system designers focus on task representation and a top-down approach that will eventually lead to a more robust systems as it will expose only high-level parallel construct (such as tasks, state machines and classes) and will let the tools manage the mapping to the hardware. The tools can either perform static verification or dynamic checks to fully manage thedefined parallelisms.

Unfortunately using such a methodology does not open the door to potential parallel performance gains as it usually disconnects system design and algorithms. Our current approach in ARGO is to center the flow on a high level language like Scilab and relying on the tools to extract possible parallelism, measure performance (through WCET estimation) and map directly to the hardware. The major and central point is that all the tools and users will interact ; getting and pushing back information on a shared container.

On a classical development cycle, the end-user will produce high-level Scilab algorithm ; emit sequential C code from it ; estimate functions WCET ; map functions to tasks and generate the code for a target. The important point is that this is might not be a top-down approach but an iterative one where each tool can be used independently.

SysCOLA [20] combines system modelling done in the COLA language and virtual prototyping done with SystemC The system functionality is designed in three steps, an abstract system modelling phase where the application model is mapped to an abstract platform, a virtual prototyping phase where the exploration of mapping of application tasks to the actual architecture is rendered where it is possible, and finally a system realization phase where the validated tasks are ported to the real platform for execution.

In the abstract system modelling phase, the COLA language is designed to convert the functional requirements to a functional model, rendering a model-level simulation and formal verification if possible. Units of functional models are then clustered into mode clusters (managing the model state transitions) and working clusters (realising the behaviour for each of the system state). The communications between clusters in this phase are defined only as logical communication. This creates a separation of concern between the actual communication studied in a later design phase (virtual prototyping) while allowing a first topological placement exploration design to be performed. Platform component as ECUs, network interfaces, sensors or actuators are only abstractly defined to create the topology and run a first placement and scheduling algorithm based solely on the parameters defined for these abstract components and clustered functional models.

The virtual prototyping phase is used to refine the virtual platform abstraction. Communications and interaction between components are derived from chosen SystemC implementation of the abstract components. This enables a timing simulation that can refine

the abstract system figures, thus creating an iterative process between the virtual prototyping phase and the abstract system modelling phase. An annotation system of the generated code can introduce the delays due to the RTOS scheduling or network overheads determined during simulation.

SysCOLA differs from ARGO in the sense that the mapping of the functional behaviours to the target platform nodes of computation is defined at very high level. WCET estimations are derived from the used components and no explorations of the parallelism inside the function clusters are studied.

RAPITA Systems LTD presented a white paper of technical works, developed in ARTEMIS 2011 project CRAFTERS [21], on integrating WCET information into Simulink models in [22]. They show how information on the WCET can be integrated in the simulation model and be shown to engineers to detect critical path and features in terms of WCET contribution. The similar idea is present in the ARGO project with the possibility to re-inject information and annotations from the code parallelization and optimization with WCET back-annotations into the Scilab Xcos simulation models.

The tool-chain described in the European FP7 project ParMERESA [23] differs from the ARGO tool-chain at the high level entry point. ParMERESA offers the possibility to analyse legacy code as presented in [24] but does not tackle the implication on the model-based design tools. The approach they present is however very sound in the use of predictable code structures and annotations of sequential source code to meet the requirements of predictability for hard real time constrained applications.

Interactions between tools for transmission of information related to WCET have been studied in the European FP7 project ALL-TIMES [25]. Interface formats defined in this project to enrich and annotate code and traces show that there is a great industrial focus on interactions between tools. This is particularly important for the ARGO tool-chain as WCET estimates for parallel code derived from functional models targeting specific platform that include many different layers interactions. Also, the Multi Core Association (MCA)[1] has established working groups to support tool interoperability, such as TIWG[2] and .SHIM[3]. The goal of the Software-Hardware Interface for Multi-Many-Core (SHIM) working group is to define an architecture description standard from the software design perspective to provide a common interface that abstracts the hardware properties that are critical to enable multicore tools. The Tools Infrastructure Working Group (TIWG) promotes the interaction of existing and new tools to support migration of legacy sequential applications to multicore platforms and to support development of new parallel software.

[1] http://www.multicore-association.org
[2] http://www.multicore-association.org/workgroup/tiwg.php
[3] http://www.multicore-association.org/workgroup/shim.php

4. Worst Case Execution Time Toolchain

4.1 WCET Estimation techniques for many-cores

This section provides the state-of-the-art in research performed recently on WCET estimation for many-cores. Section 4.1.1 provides a general overview of the collaborative projects in Europe closely related to ARGO. Section 4.1.2 provides more detailed individual research results produced in academia, on WCET estimation for many-cores, and on WCET-oriented parallelization.

4.1.1 Related collaborative projects in Europe

Some collaborative projects in Europe have partially addressed the issue of WCET estimation for parallel applications running on many-cores. This section omits the projects that do not address WCET estimation at all, such as:

- RECOMP (http://atcproyectos.ugr.es/recomp) focusing on certification;
- RESPECTED (http://anr-respected.laas.fr/index en.html) focusing on simulation and implementation of scheduling algorithms;
- CERTAINTY (http://www.certainty-project.eu/), mainly focusing on certification, SAFURE (https://safure.eu/), focusing on security;
- MULTIPARTES (http://www.multipartes.eu/), focusing on execution over partitioned multi-core embedded platforms;
- CONTREX (https://contrex.o s.de/home/), focusing on mixed critical control over multi-core platforms;
- DREAMS (http://www.uni-siegen.de/dreams/home/), focusing on virtualization technologies and certification methods for security and safety.

The remaining of this section describes in details the relevant projects:

- P-Socrates (http://www.p-socrates.eu/, Parallel Software Framework for Time-Critical Many-core Systems, 2013-2016) is a European project. It aims at exploiting the huge performance opportunities of the most advanced many-core processors for high-performance and real-time applications, while ensuring a predictable performance and maintaining (or even reducing) development costs. The purpose is to develop design framework (from the conceptual design of the functionality to the physical implementation of the system) to facilitate the deployment of standardized parallel architectures. In contrast to ARGO, P-Socrates does not aim at defining parallelization techniques for WCET improvement, it targets commercial-off -the-shelf (COTS) hardware instead of defining predictable many-core architectures, and does not target hard real-time applications.
- ParMERASA (http://www.parmerasa.eu/, Multi-Core Execution of Parallelized Hard Real-Time Applications Supporting Analyzability, 2011-2014) is a FP7 project. ParMERASA is the follow up of MERASA focusing on parallel high-critical applications. The proposed solutions provide guarantee of probabilistic WCET for parallel application and the proposed system scales up to 64 cores. ParMERASA includes the design of a predictable multi-core architecture. Parallelization techniques are also studied in ParMERASA, however, in contrast to ARGO, only coarse-grain (task-level) parallelism is considered, and the parallelization process is not fully automated.

- PROXIMA (http://www.proxima-project.eu/, Probabilistic real-time control of mixed-criticality multi-core and many-core systems, 2013-2016) is an FP7 project. It is based on the results of PROARTIS focusing on deliberately adding randomness to the timing behavior to provide novel probabilistic timing analysis methods. It defines hardware and software design guidelines to benefit from randomization and proposes a new analysis model exploiting randomization capabilities of new architectural designs for effective timing analysis of new high performance hardware features and more complex software systems. In contrast to ARGO, PROXIMA does not aim at defining parallelization techniques for WCET improvement.
- ACROSS (http://www.fortiss.org/en/research/projects/across/, 2010-2013) is funded by IST, FP7, and ARTEMIS. It aims at a multi-processor system-on-chip (MPSoC) suitable for safety-critical applications, which realizes a cross-domain ARTEMIS reference architecture based on the blueprint developed in the GENESYS project, and it develops a flexible tool-supported model-based design methodology. Similarly to ARGO, the application is analyzed in two parts, i.e. standalone code that does not use any ACROSS specific features and the code that employs them, such as TTNoC based communication. In contrast to ARGO, ACROSS does not aim at defining and automating parallelization techniques for WCET improvement.
- T-CREST (http://www.t-crest.org/, time-predictable multi-core architecture for embedded systems, 2007-2013) is developing a time-predictable system that will simplify the safety argument with respect to maximum execution time while striving to double performance for 4 cores and to be 4 times faster for 16 cores than a standard processor of the same technology (e.g. FPGA). The goal of the T-CREST system will be to lower costs for safety relevant applications, reducing system complexity and at the same time achieving faster time-predictable execution. Within T-CREST a compiler with WCET-oriented optimizations was designed. However, unlike ARGO, the optimizations do not include WCET-guided parallelization.
- Capacités (capacites.minalogic.net/en/, Parallel Computing for Time and Safety Critical Applications, 2014 - 2018) has a French government funding. The project's objective is to develop a platform based on many-core architectures, and to demonstrate the relevance of these many-core architectures (and more specifically the Kalray many-core) for several industrial applications. The Kalray MPPA many-core architecture is currently the only one able to meet the needs of embedded systems simultaneously requiring high performance, lower power consumption, and the ability to meet the requirements of critical systems (low latency I/O, deterministic processing times, and dependability). Unlike ARGO, Capacités does not aim at defining WCET-guided parallelization.

4.1.2 Research results on WCET estimation for many-core architectures

Originally, WCET estimation methods were designed to target sequential code running on single core architectures (see [26] for a survey). For the scope of safety-critical systems, methods using static analysis are recommended to make sure strict upper bounds of WCETs are produced (identify the longest execution path in the program as well as the worst-case state of all architectural components). ARGO partners have a long and strong experience in techniques and tools for WCET estimation for single core architectures, in particular in the analysis of caches [27], [28], [29], the analysis of caches hierarchies [30], [31], [32] and the control of memory hierarchy such as cache locking [26] and control of scratchpad memories [33].

Several challenges exist in the domain of WCET estimation. At the hardware level, the main challenge raised by many-core architectures is the sharing of hardware resources between cores (busses, NoCs, caches, memory controller, memory banks, etc), as surveyed in [34]. At the software-level, the challenge is to have sufficient information on the behavior of applications (communications, synchronizations) to avoid producing overestimated WCET results. ARGO will address several of the aforementioned challenges.

4.1.2.1 WCET estimation for independent tasks

The first results on WCET estimation for many-cores were focusing on scenarios in which independent tasks are running on different cores competing for the architecture shared resources. The main challenge here was to derive a worst-case estimate of interferences caused by the accesses to shared resources. These seminal works assume that the task-to-core mapping is known prior to the analysis. In addition, no assumption is made on how tasks are scheduled on each core.

Research has also been performed for shared buses. In particular, precise methods have been proposed for TDMA buses [35] and for several other bus arbitration policies [36].

Regarding shared caches, Yan et al. [37] analyzed the maximum interference suffered by one task when sharing the L2 cache with concurrent tasks. Subsequent results attempted to reduce the amount of interferences by using software-controlled bypass of the L2 cache [38] or by taking advantage of some knowledge on the application behavior [39]. Tools, such as the one described in [40], integrate this type of interference analyses.

4.1.2.2 WCET estimation for parallel applications

Assuming that the application consists of independent tasks often results in pessimistic WCET estimates. The reason is that it is then very difficult (or even impossible) to predict the exact time when concurrent accesses to resources occur. In order to overcome this problem, several recent research works consider more elaborate application models (such as task graphs). In these models, task interactions are explicit, and more information about accesses to shared resources exist and it can, therefore, be used to obtain less pessimistic estimates.

For example, Ozaktas et al. [41], [42] describe WCET calculation techniques for parallel applications that are based on the fork-join task model, with barriers and locks as synchronization tools. Their work focuses on providing a tight estimates of the processors idle time during synchronizations. In contrast to ARGO, the aforementioned research does not address the sharing of architectural resources.

Similarly, Potop-Butucaru et al. [43] also propose a WCET calculation method for applications modeled as task graphs. The originality in their approach lies in their ability to consider the amount of cache reuse between tasks allocated to the same core. In this work, task-to-core mapping is assumed known. In contrast, in ARGO, task mapping is part of the process of WCET optimization to enable more opportunities for WCET improvement.

4.1.2.3 Joint WCET estimation and task mapping/scheduling

All the aforementioned techniques assume that the task-to-core mapping is already defined. This is a significant shortcoming, as the choice of the task mapping itself can potentially help to the significant reduction of the WCET estimates.

The ARGO project aims at addressing this shortcoming by jointly considering WCET estimation and task mapping and scheduling. More specifically, the goal is to enable a form of iterative design space exploration to obtain the best possible WCET estimate given a platform and an application model described as a task graph. Many studies on static scheduling of task graphs on many cores have been produced in the past [44], [45], [46], [47]. However, they all assume WCET constant and known. In ARGO, a holistic cross-layer iterative optimization process involving both task mapping and WCET-aware optimizations will be defined to take benefit of as many sources of WCET optimizations as possible.

4.1.2.4 Parallelization for WCET improvements

As mentioned earlier, the main issue for WCET estimation on multi-core is dealing with shared resources such as the NoC and/or external/shared memory. In all the joint WCET approaches discussed in the previous paragraph, the application task graph is considered as an input to the problem. This choice is natural for a program consisting of independent tasks/applications and/or for manually parallelized programs. However, the problem is slightly different when using an automatic parallelizing compiler as proposed in ARGO. Thanks to complex program transformations, such a tool has the ability to derive many different functionally equivalent task-graphs from a single input application [48]. This multiplicity of tasks graphs is useful to explore solutions with different parallelism/locality trade-off [49] before entering the scheduling/mapping process. In our context, we instead want to use this diversity to explore alternative trade-off between parallelism and shared resource conflicts.

This exploration is possible thanks to the availability of local/private scratchpad memory on each processor. Such memory can be used to cache data and/or intermediate results that are used during computations, and therefore reduce the number of accesses to shared resources. Although automatic scratchpad management has been widely studied in the context of hard-real time systems [33], [50], most previous work on the topic either focused on the instruction cache [51], [52] and/or did not considered complex array access patterns as those that can be found in compute intensive image processing and/or control algorithms. Our goal in the ARGO project is to address these shortcomings by building on previous research work on communication synthesis and storage optimization targeting compute intensive loop nests kernels. These topics originate from the high performance computing and optimizing compiler communities, but have however never been applied to hard real-time systems and WCET estimation. We are particularly interested in recent results in intra and inter array storage optimization [53], [54], [55] and on multi-banks memory conflict management [56], which may help us reducing the impact of shares resources on the tightness of the WCET results.

4.2 WCET-aware optimizations for heterogeneous multi-core architectures

Most of today's high-performance processors are multicores, not only in the desktops and servers but also in the embedded systems market. Though the increased overall computational power is beneficial to the average-case application, multi-cores pose a fundamental problem for safety-critical real-time applications. Since some of the hardware components in a multi-core system are shared between cores, tasks that execute on different cores may interfere with each other during accesses to shared components. This breaks the isolation between tasks and makes their Worst-Case Execution Time (WCET) harder or even

impossible to predict. WCET estimation methods for a multi-core architecture must take into account not only the program execution paths and underlined architecture (as addressed in WCET estimation methods for uni-core architecture [57]) but also the resource contentions (e.g., bus contention and shared memory contention [58]). Moreover, when analyzing the timing behavior of parallel applications, these methods also have to consider the application's properties (e.g., multitasking, inter-task communication and synchronization).

The problem of mapping and scheduling on heterogeneous multi-core architectures has been addressed only for the average case execution time of the application's tasks. To the best of our knowledge, there does not exist any approaches that take into account heterogeneous multi-core architecture with main objective the WCET optimization. All the approaches have as objective to minimize the overall completion time (makespan) of the application, which is represented as a directed acyclic graph (DAG), where the nodes represent application tasks and edges represent data dependencies between the tasks. Many solution methods are based on metaheuristics, such as genetic algorithms [59] [60] [61] [62] or evolutionary algorithms [63] [64], while others are based on a variety of heuristic categories, such as list scheduling [65] [66] [67] [68] or duplication-based search [69] [70].Moreover, recently, a hybrid approach has been developed based on integer linear programming and constraint programming whose objective is to provide the optimal solution for the case of a heterogeneous platform [71] [72] [73]. However, it does not consider the communication cost between the tasks and has to be extended in order to capture the impact of the data transfer. In the following paragraphs the most state-of-the-art works, which have been done in the field of WCET-aware optimization for multi-core architectures, are presented and discussed.

In [74], an approach for performing worst-case execution timing analysis and system scheduling for real-time applications implemented on multiprocessor System-on-Chip architectures is proposed. The emphasis of this work is on the bus scheduling policy and its optimization, which is of huge importance for the performance of such a predictable multiprocessor application. Bus scheduling for real-time systems has been previously considered only in the context of inter-task communication ignoring however the problem of interference caused due to cache misses [75] [76]. The issues addressed in this paper add a completely new dimension to the problem, by the dependency of WCETs on the bus schedule and by the much finer granularity at which bus transfers, due to cache misses, has to be considered.

In more details, in this work, a TDMA-based bus access policy is considered and the actual bus access schedule is determined at design time to be enforced during the execution of the application. Moreover, the bus access schedule is taken into consideration during WCET estimation that is integrated together with the determination of bus access schedule in the system level scheduling process, which presented in this paper.

Initially, in the proposed algorithm, which based in a list scheduling technique, the system level static cyclic scheduling takes place. Then, the task scheduling of an inner optimization loop is applied in which the close to optimal bus segment schedule with respect to the active tasks must be determined. In order to find the best candidate bus schedule, an integrated WCET estimation is applied together with a proper cost function, determining the global delay of the system and bus schedule. The proposed algorithm terminates until the system static scheduling is determined and the appropriate close to optimal bus schedule is found. Finally, the experimental results which consist of a set of benchmarks and a real-world example prove the efficiency of the proposed method.

In [77], a technique for optimizing the average case and the worst case simultaneously, allowing for a good average-case execution time while still keeping the worst case as small as possible, is introduced. The authors of this paper have previously proposed a novel

technique to achieve predictability on multiprocessor systems by doing WCET analysis and scheduling simultaneously [74]. The main contribution of this work is the combination of the average-case and the worst-case execution time, and it was the first time it has ever been done within the context of achieving predictability.

Initially, the same procedure for deriving the system and bus schedule as in [74] is followed. The results which produced from the previous step together with a designer-specified limit on the maximum allowed WCET and the memory access histogram data for the tasks are taken as input parameters in a proposed optimization approach. The outputs of the proposed algorithm are the final bus and task schedule that are optimized for both WCET and ACET and the average-case task schedule length. As the average-case and worst-case execution timing analysis with respect to several bus schedule candidates is a time-consuming process, a novel cost function, which identifies which parts of the bus schedule could be modified in order to decrease ACET without initially considering the effects on WCET, is proposed. The introduced function is based on heuristics, which create a new bus schedule by modifying some regions of the initial provided bus schedule. The main purpose is to improve the ratio between average-case improvement and worst-case extension, with respect to global delay. The framework which presented in this work is evaluated by using an extensive set of generated C programs. The programs were constructed with respect to randomized task graphs consisting of between 20 and 200 tasks, mapped on 2 to 8 processors.

In [78], a cache aware task mapping approach to minimize the Worst-Case Response Time (WCRT) of concurrent tasks, is proposed. Caches are modeled through abstract interpretation and an ILP formulation approach is employed for task mapping. Both the cache conflicts in the L2 cache and the workload balance are considered in this approach. The proposed shared cache aware task mapping framework consists of three phases: intra-task cache modeling, task mapping with shared cache modeling and iterative WCRT computation. As regards the intra-cache analysis, it utilizes the multi-level non-inclusive cache analysis, which proposed in [79]. Furthermore, for the WCRT estimation, a proper framework, which models shared cache conflicts in multi-cores [80], is used. Experimental results with both synthetic task graphs and real-world benchmarks show that the particular approach returns the best task mapping most of the time. Also, it is more efficient in terms of runtime compared to an exhaustive enumeration approach, which is able at producing an optimal solution. Finally, the aforementioned approach achieves significant reduction in WCRT compared to traditional approach agnostic to shared cache conflicts and solely focusing on load balancing.

In [81], a heuristical framework that optimizes the system schedule and the bus schedule for a given task set and a given number of processor cores is presented. The focus is on the interference caused by the shared buses and the optimization goal is to minimize the WCET. The framework is modular in the sense of defining an interface for heuristics that select the task to be executed next on a particular processor core. The proposed algorithm is parametric in the task and bus selection heuristic. It integrates the construction of the system schedule and the construction of the bus schedule by alternately adding a task to the system schedule and building a part of the bus schedule. The system which assumed in this paper consists of a simple multi-core processor model with a shared bus arbitrated according to a TDMA policy. A main drawback of this work is that their system model assumes a single execution time per task (ignoring possible bus blocking effects); as a result the execution time of the task set and the overall WCET of the task set coincide for the assumed system model.

In [82] two approaches for optimizing the resource usage in a temporally partitioned multi-core system are examined and it is shown that these techniques can reduce the WCET by

more than 30% on average, leading to better schedulability and higher system utilization. In this paper, two novel optimizations that can significantly improve the WCET but also the average-case execution time (ACET) of programs running on timing-predictable multi-cores are presented. The system model which assumed in this work consists of *n* highly predictable ARM7TDMI cores. Each of the cores has access to local scratchpad memories and a shared RAM. The shared memory is accessed via a shared bus, which is responsible for arbitrating requests of the cores. The first proposed approach is an evolutionary optimization which consists of a multi-objective evolutionary search algorithm. The evolutionary algorithm automatically determines a range of well-suited schedules (TDMA, FAIR, PRION, PD) for an application and enables users to choose a solution which balances WCET, ACET and utilization according to their needs. The presented experimental results confirm the efficiency of the first proposed approach and it can be seen that WCET can be improved more than 30% using the appropriate bus schedule policy. The second proposed approach is a multi-core WCET-aware instruction scheduling which re-structures the input programs to increase their performance. It consists of two novel priority assignment heuristics that are tailored towards the optimization of the WCET of tasks running on time-predictable multi-cores. Moreover, the combination of the evolutionary bus optimization from the first proposed approach and the instruction scheduler from the second approach is tested. The instruction scheduler was invoked for every generated individual to also find solutions, which are only accessible through a combination of bus and instruction scheduling. The above combination leads to a combined average WCET reduction of to up to 33%.

In [83] a static scheduling solution for an isolated parallel application running on a many-core architecture is introduced. The proposed scheduler not only respects dependence constraints between tasks (i.e., communications and synchronizations) but also takes into account the effect of local caches on tasks WCETs. It is developed an optimal integer linear programming model for solving the task scheduling problem, whose objective is to minimize the WCET of the parallel application. The proposed scheduling approach is a partitioned non-preemptive scheduling approach: tasks are not allowed to be migrated and preempted, which prevents the system from suffering from hard-to predict migration and preemption costs. The system model which assumed in this work consists of many-core architecture with homogeneous cores and each core has a private cache memory.

In order to account for the variability of task's WCETs due to private caches, two cases has been considered in this work depending on whether the task that runs on the core is first or it runs after another task. It is assumed that the WCET of a task that runs after another task will be less than a task which runs first in the core. Although an optimal solution could be found through by the proposed ILP formulation of the task mapping/scheduling problem, the level of abstraction which included in the cache analysis could lead to underestimation of the overall WCET. For the performance evaluation of the proposed scheduler, the WCET values obtained by the proposing scheduling method, a random scheduling method, and scheduling method without taking into account the effect of private caches are compared. Finally, based on the experimental results it is shown that the proposed scheduling approach generates schedules that lead to the smallest estimated WCET in comparison to the results from the other implemented methodologies.

Although cache memories provide a solution to the problem of the large off-chip memory latency, they introduce some major problems. Firstly, cache memory consumes a significant amount of energy due to its tag store and associated circuitry. Secondly, cache memory makes it difficult to compute the WCET of a program. For this reason scratchpad memories are introduced which are an attractive alternative to cache memories due to their lower energy consumption and higher predictability of program execution. In the following paragraphs are presented some state-of-the-art works which have been done in the field of WCET-aware optimization for multi-core architectures with scratchpad memories.

In [84] a compile-time scratchpad allocation framework for multi-processor platforms, where the processors share on-chip scratchpad space and external memory is accessed through a shared bus, was developed. The hardware platform, which assumed in this work, is a multi-processor architecture which contains multiple processing elements (PEs) on a chip. Each PE owns a private scratchpad memory and with respect to a specific PE, the SPMs of other PEs are referred to as remote SPMs. A PE has dedicated access to its private SPM with minimum latency. A PE can also access a remote SPM through the crossbar connecting the processors. Access to a remote SPM is relatively slower than accessing private SPM but much faster than accessing the off-chip memory. In this work, it is assumed that the latency to access remote SPM is bounded by a small constant (since the on-chip links generally operate on high bandwidth, this is a reasonable assumption). This kind of architecture essentially creates a virtually shared scratchpad memory space (VS-SPM) among all the PEs.

The allocation method which described in this paper considers the waiting time for bus access while deciding which memory blocks to load into the shared scratchpad memory space. Incorporating the bus schedule into the scratchpad allocation method leads to a global optimization of an application, as compared to employing local scratchpad allocation schemes in individual processors which locally optimize the per-processor execution time. The prime novelty in this work is the incorporation of the bus schedule into the multi-processor scratchpad allocation scheme. The proposed allocation framework exploits the shared scratchpad space available in MPSoCs and considers variable lifetimes to efficiently utilize the available shared scratchpad space. As evidenced by the experiments, the proposed scratchpad allocation scheme is able to significantly reduce the WCET of real-life embedded applications. The efficacy, sensitivity, and efficiency of the proposed memory allocation scheme is evaluated on two real-world embedded applications - an application controlling an Unmanned Aerial Vehicle (UAV), and a (fragment of) an in-orbit spacecraft software.

In [85] two novel WCET-aware dynamic SPM code management techniques for Software Managed Multi-core (SMM) architectures are presented. In SMM architectures, each core can only access its scratchpad memory (SPM); any access to main memory is done explicitly by DMA instructions. As a consequence, dynamic code management techniques are essential for loading program code from the main memory to SPM. Current state-of-the-art dynamic code management techniques for SMM architectures are, however, optimized for average-case execution time, not worst-case execution time (WCET), which is vital for hard real-time systems. The first proposed technique can find the optimal mapping solution for WCET and is based on integer linear programming (ILP). Because the number of function-to-region mapping choices grows exponentially with the number of functions in the program, this technique does not scale to large programs. Thus, a polynomial-time WCET-aware heuristic algorithm that is scalable is also presented, but finds a sub-optimal mapping. Experimental results with benchmarks from Malardalen WCET [86] suite and MiBench suite [87] show that the ILP solution which described in this paper can reduce the WCET estimates up to 80% compared to previous techniques. Furthermore, the proposed heuristics can, for most benchmarks, find the same optimal mappings within one second on a 2GHz dual core machine.

	ILP	Heuristics	L1 Cache	L2 Cache	SPM	Bus Schedule	WCET,ACET
[74]	✓	✓	✓			✓	WCET
[77]		✓	✓			✓	WCET & ACET
[78]	✓		✓	✓			WCET
[81]		✓				✓	WCET
[82]		✓	✓			✓	WCET
[83]	✓		✓				WCET
[84]		✓			✓		WCET
[85]	✓				✓		WCET

Table 1. Summarized features for each referenced paper.

In the above table the included features for each paper which described in this document are summarized. The columns consist of the features which are distinct in each work and respectively the rows consist of the described papers. The first and the second column refer to the solution method which will be either ILP or heuristics. The second, third and fourth columns refer to the on-chip memory system which used to tackle with the large off-chip memory latencies. Thus, these columns refer to L1 caches, L2 caches or scratchpad memories. The sixth column refers to the consideration of bus schedule in the optimization and finally the seventh columns refer to the optimization objective which will be the WCET, the ACET or the combination of the previous two.

5. Memory and communication optimization in time-predictable multi-core architectures

Enabling time-predictability for both hardware and software is substantial for any kind of embedded system with hard real-time constraints. On the hardware side, the introduction of time-predictability becomes much more challenging when multicore processors instead of single-cores come into play. The main reason for that is the *interference* between the tasks running concurrently on different processor cores. Such interference can occur when multiple threads/cores try to access a shared resource like a shared bus or memory segment at the same time [88]. Dependent on the resource's architecture and arbitration scheme, such concurrent accesses can dramatically increase the worst case access time, leading to a high WCET for all operations which might possibly access shared resources.

Additionally, in many cases, it is practically impossible to determine all possible concurrent accesses on shared resources in a static code-analysis. On the one hand such an analysis would require detailed knowledge on all tasks/threads running on the multi-core processor. This limits the possibilities to develop software components/tasks independently because changes on one component will affect the WCET behaviour whole multi-core system and all other tasks [88]. On the other hand, the occurrence of interference on multicore processors does not only depend on the worst case behaviour of the tasks. The relative time offsets between two tasks can vary significantly dependent on the branches that are actually taken in the code of each thread. Hence, it is very difficult to statically determine each point in time where interference can possibly occur, since every possible combination of taken branches needs to be considered. For more complex applications, this makes a static analysis of interference almost impossible. In the worst case it would then be necessary to assume permanent presence of thread interference which results in significantly higher WCETs.

Given this problems, it can be reasonable to use WCET optimized multicore architectures instead of standard Commercial Off-The-Shelf (COTS) multicore architectures which are often optimized for average case performance using branch-prediction, out of order-pipelines, etc. [89]. WCET aware multicore architectures on the opposite, try to maximize the worst case speedup compared to the WCET of an equivalent single-core processor. This is especially important given the presence of thread interference, because overly conservative WCET estimations on COTS multicore architectures can otherwise dramatically reduce or even negate WCET speedups [90].

Several related works such as ARAMiS [91], parMERASA [92], [90], [93] and [94] have been investigating multicore WCET analysis methods and WCET optimized multicore hardware-architectures. This resulted in different approaches for hardware, software and WCET analysis optimizations targeting hard real-time systems. The optimizations mainly aim to I) minimize the worst-case amount of interferences between cores/threads, II) try to minimize the worst case delay for concurrent requests on shared resources and III) improve multicore WCET analysis methods to predict possible thread interferences. An approach for I) is to use appropriate spatial and temporal partitioning concepts for the software to bound the amount of concurrent shared resource accesses. This can be complemented by runtime mechanisms like those in [95] which are used to enforce guaranteed capacities for shared resources. On the hardware side, distributed memory architectures can help to isolate the concurrent thread and separate address spaces in order to limit thread interference. In order to opt for II), the worst case delays for shared resource accesses can be reduced using WCET optimized hardware architectures like [88]. This is achieved by sophisticated memory hierarchies (including private and shared caches) as well as strongly predictable bus and/or network on

chip architectures with appropriate arbitration mechanisms. Concerning point III), in [95] interference-sensitive WCET analysis methods have been proposed to analyse resource usage and estimate the worst case interference delays.

ARGO focusses mainly on Network on Chip (NoC) based multi-core processor systems. Therefore, the state of the art in time predictable and WCET optimized NoC architectures is summarized separately in the following section

5.1 NoC predictability

In recent times, continuously increasing transistor densities enabled the integration of large numbers of IP blocks on a single chip. Multi-processor System on Chip (MPSoC) architectures originating from this development have been in the market for few years. Most of the available Systems on Chip (SoC) utilize buses and point-to-point connections for on chip communication. Non-scalability due to centralized arbitration mechanisms limits the use of such interconnect systems as communication infrastructures of next generation many-core SoCs with hundreds of processing elements. Furthermore, WCET analysis can be tricky unless very inefficient methods like TDMA are used. Networks on Chip (NoC) offer better scalability by deploying distributed arbitration schemes for on-chip communication in many-core systems [96]. Modern NoC based multi-core architectures have already proven their effectiveness [97, 98, 99]. Intel's Single-chip Cloud Computer"(SCC) is one example for an existing NoC based computer architecture [100]. It is a multi-core architecture that consists of 24 tiles connected through a mesh network, with two Pentium cores per tile. In future many-core architectures, new strategies need to be devised for efficiently utilizing such novel interconnect systems especially in the scenarios of dynamically varying workloads and communication patterns. Invasive computing [101, 102] proposes a novel computing paradigm to facilitate application level parallelism in many-core architectures. Its main idea and novelty is to support resource-aware programming from application as well as architectural aspects. An invasive application can dynamically adapt its resource requirements on the status of the underlying hardware by the use of hardware monitoring information. It can exclusively reserve computation, communication and memory resources in a phase called invasion. The ideas of invasive computing require architectural changes. The iNoC as communication infrastructure needs to provide efficient support for reservation and release of communication resources to give the processes of an invasive application the possibility of predictable and fast communication among themselves. The reservation of communication resources by applications in an iNoC is realized through configurable guaranteed service connections, reserving virtual channels (VC) along its path. These connections, once established, are fully predictable and thus WCET conform.

Providing Quality of Service (QoS) guarantees for on-Chip communication in Networks on Chip in general is a well-researched topic. Millberg et al. present the Nostrum NoC in which guaranteed service (GS) support is augmented in addition to non-predictable best effort (BE) traffic delivery [103]. A concept called Looped Containers is applied to give service guarantees. Though the VCs used for the Looped Containers are set up semi-statically it has limited run-time flexibility. Goossens et al. present the NoC architecture Aethereal which offers both guaranteed service and best effort traffic support [104]. GS support requires resource reservation to provide worst case bandwidth guarantees with bounded latency. BE traffic exploits the NoC capacity which is not utilized by GS traffic. Nostrum and Aethereal use Time-division multiplexing (TDM) based global synchronization, which leads to unused time slots and therefore reduces the overall throughput. Bolotin et al. present QNoC that provides QoS traffic guarantees with four service levels of different priorities [105]. Malek et

al. proposed resilience improvements for service-aware NoCs [106].Preemptive scheduling is used to schedule traffic corresponding to the respective service level. In such a scheduling scheme, the traffic of higher priority always limits the one with lower priority. Virtual Channel (VC) reservation is investigated in [107]. Source routing is used to calculate the routes centralized. This strategy is not applicable for scalable NoCs due to the increasing complexity of centralized routing. In [108] VC reservation for throughput and latency guarantees is investigated. BE traffic is prioritized to minimize the impact of GS transmissions. Bjerregaard and Sparso propose a clock-less NoC named MANGO, which offers GS and BE support through two separately implemented routers [109]. The GS routers forward data streams on statically programmed point-to-point connections. Whereas the BE routers are responsible for connection-less packet based traffic. The allocation of resources to BE and GS traffic can only be done statically. The global TDM based designs that are referred above, suffer from overall throughput reduction by inefficient virtual channel utilization. This is due to the reason that the same time slots need to be available in all routers of a requested connection.

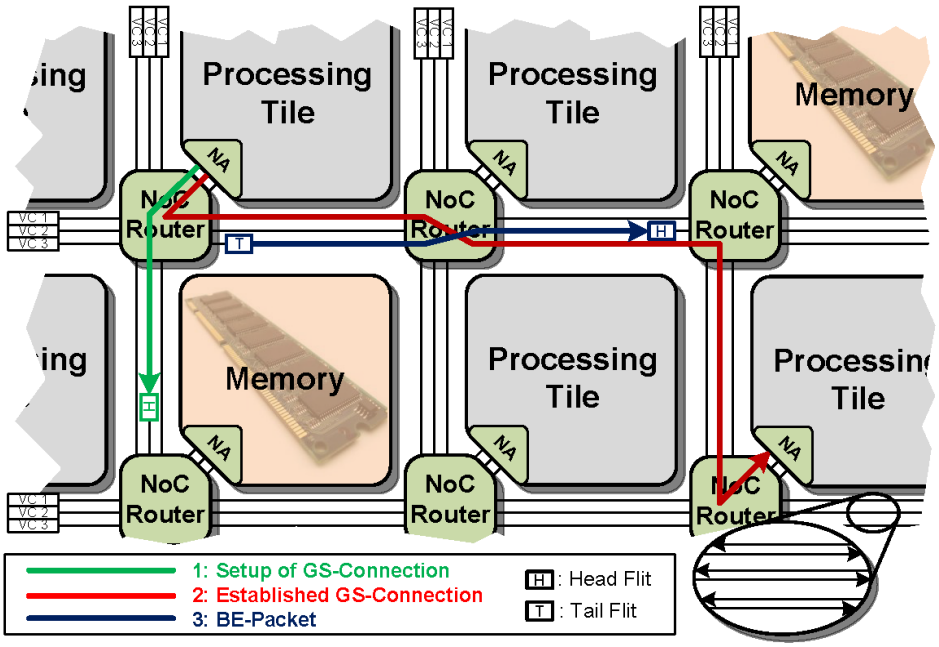

Figure 5 Hardware configuration for the NoC

6. Conclusion

Worst Case Execution Time analysis techniques managing parallel code on heterogeneous architectures is a very hot research topic as proven by the abundant literature covered in this document. Applying these techniques in model based design environments and toolchains is crucial for a better use of the current and future generations of hardware to their full potential.

Industrial players have focused their efforts in the code generation from the models for hard real-time applications on single core processors, sometimes preferring to invest in more recent multicore architectures and using single cores to maintain the predictability of the generated code. Some more modern approaches include dividing the high level process in independent tasks that can be mapped on a single core. But using the full possibilities offered by multi-core architectures is still rare and difficult considering the available products and techniques.

ARGO provides an exploration of the newest techniques available in WCET analysis on recent architectures keeping in mind the requirements and constraint of industrial players. This exploration of the possibilities offered by heterogeneous, multi-core architecture is done at every step and every level of the design of application. It encompasses all aspects and tackles all difficulties previously encountered in the domain while keeping in mind the requirements of end-users with regard to the process of building a system.

As a consortium, we are confident that our joint expertise, our academic and industrial backgrounds and our commitment to produce quality results will significantly reduce the gap between the breakthroughs in WCET analysis for parallel code generation and their diffusion to the industry.

ARGO has set sail; you can follow our adventure on http://www.argo-project.eu/.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 688131.

7. References

[1] N. Weber, J. Ernst, S. Junger, H. Neubauer, W. Tschekalinskij and N. Wervaal, "Nanostructured Optical Filters in CMOS for Multispectral Polarization and Image Sensors," in *Microelectronic Systems*, H. Hrsg. A., E. G. and H. R., Eds., Berlin, Springer-Verlag, 2012.

[2] Arne Nowak, Jürgen Ernst and Friedrich Günther, "Inline Residual Stress Measurement in Glass Production," *Glass Worldwide 58,* January 2015.

[3] Michael Schöberl, Koray Kasnakli and Arne Nowak, "Measuring Strand Orientation in Carbon Fiber Reinforced Plastic (CFRP) with Polarization," *to appear in Proceedings of the 19th World Conference on Non-Destructive Testing (WCNDT),* June 2016.

[4] Marcus Bednara and Katarzyna Chuchacz-Kowalczyk, “Real time polarization sensor image processing on an embedded FPGA/multi-core DSP system,” *Proc. SPIE 9506, Optical Sensors 2015,* 5 May 2015.

[5] Honeywell, "Terrain and Traffic Awareness, March 2016," *https://aerospace.honeywell.com/en/product-listing/terrain-and-traffic-awareness,* 2016.

[6] ACSS, "T2CAS," *Retrieved March 16, 2016 from http://www.acss.com/products/t2cas/,* 2016.

[7] LANDMARK, "Terrain Awareness & Warning Systems," *Retrieved March 16, 2016 from https://www.l-3avionics.com/products/landmark/,* L3. 2016.

[8] Universal Avionics, "TAWS Terrain Awareness and Warning System," *Retrieved March 16, 2016 from http://www.uasc.com/home/shop/avionics/taws,* 2016.

[9] Dale Smith, "Traffic Alert Collision Avoidance Systems," *Pilot's Guide to Avionics 2005, TAS Buyer's Guide. Retrieved March 16, 2016 from https://www.aea.net/PilotsGuide/pdf/05-06_Archive/TAWSPG05.pdf,* 2005.

[10] Honeywell, "Continued Protection," *Retrieved March 16, 2016 from https://aerospace.honeywell.com/en/products/safety-and-connectivity/mark-v-egpws ,* 2016.

[11] Honeywell, "MK V and MK VII EGPWS Pilot's Guide," *Retrieved March 16, 2016 from https://www51.honeywell.com/aero/common/documents/egpws-documents/Operation-documents/060-4241-000.pdf,* 2013.

[12] Paul J. Prisaznuk, "Integrated modular avionics.," *In Proceedings of the IEEE*

1992 National Aerospace and Electronics Conference. IEEE, 39-45., 1992.

[13] Christopher B. Watkins and Randy Walter, "Transitioning from federated avionics architectures to integrated modular avionics," *In IEEE/AIAA 26th Digital Avionics Systems Conference. IEEE, pp. 2-A.,* 2007.

[14] Honeywell, "Integrated Surveillance Performance," *Retrieved March 16, 2016 from https://aerospace.honeywell.com/en/products/safety-and-connectivity/aircraft-environmental-surveillance-system,* 2016.

[15] Jan Nowotsch and Michael Paulitsch, "Leveraging multi-core computing architectures in avionics," *In 2012 Ninth European Dependable Computing Conference (EDCC). IEEE, 132-143.,* 2012.

[16] Bernd Koppenhoefer and Dietmar Geiger, "EMC2 Use Case:Hybrid Avionics Integrated Architecture Demonstrator," *HiPEAC, Workshop EMC² Retrieved March 16, 2016 from http://www.artemis-emc2.eu/fileadmin/user_upload/Publications/2015_HiPEAC/EMC2_Bernd_Koppenhoefer_Airbus.pdf,* 2015.

[17] Hicham Agrou, Pascal Sainrat, Marc Gatti and Patrice Toillon, "Mastering the behavior of multi-core systems to match avionics requirements," *2012 IEEE/AIAA 31st Digital Avionics Systems Conference (DASC), 6E5-1.,* 2012.

[18] Andreas Lofwenmark and Simin Nadjm-Tehrani, "Challenges in future avionic systems on multi-core platforms," *In International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE, 115-119.,* 2014.

[19] C.-S. Lin, C.-H. Lu , S.-W. Lin, Y.-R. Chen and P.-A. Hsiung, "VERTAF/Multi-Core: A SysML-Based Application Framework for Multi-Core Embedded Software Development," *Journal of Computer Science and Technology, 2011,V26(3): 448-462.*

[20] Z. Wang, H. Andreas, W. Haberl and M. Wechs, "SysCOLA: a framework for co-development of automotive software and system platform," *In Proceeding DAC '09 Proceedings of the 46th Annual Design Automation Conference,* 2009.

[21] "CRAFTERS Project," [Online]. Available: http://www.crafters-project.org/home.

[22] R. Systems, "Measuring MATLAB Simulink models worst case execution time with RapiTime," *from https://www.rapitasystems.com/system/files/TechnicalNote_Measuring_MATLAB_Simulink_models_worst_case_execution_time_with_RapiTime.pdf.*

[23] "EUROPEAN FP7 Project ParMERESA," march 2016. [Online]. Available: http://www.parmerasa.eu/.

[24] R. Jahr, M. Frieb, M. Gerdes, T. Ungerer, A. Hugl and H. Regler, "Paving the Way for Multi-cores in Industrial Hard Real-time Control Applications," in *In Work in Progress Session of 9th IEEE International Symposium on Industrial Embedded*

Systems (SIES), Pisa, Italy, June 18–20, 2014.

[25] "EUROPEAN FP7 project ALL-TIMES website," March 2016. [Online]. Available: http://www.mrtc.mdh.se/projects/all-times/index.html.

[26] R. Wilhelm, "The worst-case execution-time problem; overview of methods and survey of tools," *ACM Trans. Embed. Comput. Syst., 7(3):36:1-36:53, May 2008.*

[27] Christian Ferdinand and Reinhard Wilhelm , " Efficient and precise cache behavior prediction for real-time systems," *Real-Time Systems, 17(2-3):131-181, 1999 .*

[28] Christian Ferdinand , Reinhold Heckmann , Marc Langenbach , Florian Martin , Michael Schmidt , Henrik Theiling , Stephan Thesing and Reinhard Wilhelm , " Reliable and precise WCET determination for a real-life processor," *In Embedded Software, pages 469-485. Springer, 2001 .*

[29] Reinhard Wilhelm , Daniel Grund , Jan Reineke , Marc Schlickling , Markus Pister and Christian Ferdinand , " Memory hierarchies, pipelines, and buses for future architectures in time-critical embedded systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 28(7):966, 2009 .*

[30] Damien Hardy and Isabelle Puaut , " WCET analysis of multi-level non-inclusive set-associative instruction caches," *In Proceedings of the 2008 Real-Time Systems Symposium, RTSS '08, pages 456-466, 2008 .*

[31] Damien Hardy and Isabelle Puaut , " WCET analysis of instruction cache hierarchies," *J. Syst. Archit., 57(7):677-694, August 2011 .*

[32] Benjamin Lesage , Damien Hardy and Isabelle Puaut , " WCET analysis of multi-level set-associative data caches," *In 9th International Workshop on Worst-Case Execution Time Analysis, 2009 .*

[33] Jean-Francois Deverge and Isabelle Puaut , " WCET-directed dynamic scratchpad memory allocation of data," *In Real-Time Systems, 2007. ECRTS'07. 19th Euromicro Conference on, pages 179{190. IEEE, 2007 .*

[34] G. Fernandez , J. Abella , E. Quinones , C. Rochange , T. Vardanega and F. J. Cazorla , " Contention in multicore hardware shared resources: Understanding of the state of the art," *In 14th Workshop on Worst-case execution time analysis (WCET), 2014 .*

[35] Timon Kelter , Heiko Falk , Peter Marwedel , Sudipta Chattopadhyay and Abhik Roychoudhury , " Bus-aware multicore WCET analysis through tdma of set bounds," *In 23rd Euromicro Conference on Real-Time Systems (ECRTS), pages 3-12. IEEE, 2011 .*

[36] Timon Kelter , Tim Harde , Peter Marwedel and Heiko Falk , " Evaluation of resource arbitration methods for multi-core real-time systems," *In OASIcs-OpenAccess Series in Informatics, volume 30. Schloss Dagstuhl-Leibniz-Zentrum*

fuer Informatik, 2013 .

[37] J. Yan, "WCET analysis for multi-core processors with shared L2 instruction caches," *In Proceedings of the 2008 IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS '08, pages 80-89, Washington, DC, USA, 2008. IEEE Computer Society.*

[38] Damien Hardy , Thomas Piquet and Isabelle Puaut , " Using bypass to tighten WCET estimates for multi-core processors with shared instruction caches," *In Real-Time Systems Symposium, 2009, RTSS 2009. 30th IEEE, pages 68-77. IEEE, 2009 .*

[39] Yan Li , Vivy Suhendra , Yun Liang and Tulika Mitra , " Timing analysis of concurrent programs running on shared cache multi-cores," *In Real-Time Systems Symposium, 2009, RTSS 2009. 30th IEEE, pages 57-67. IEEE, 2009 .*

[40] Sudipta Chattopadhyay , Lee Kee Chong , Abhik Roychoudhury , Timon Kelter , Peter Marwedel and Heiko Falk , " A united wcet analysis framework for multicore platforms," *ACM Trans. Embed. Comput. Syst., 13(4s):124:1-124:29, April 2014 .*

[41] H. Ozaktas , C. Rochange and P. Sainrat , " Automatic WCET analysis of real-time parallel applications," *In 13rd Workshop on Worst-case execution time analysis (WCET), 2013 .*

[42] " Minimizing the cost of synchronizations in the WCET of real-time parallel programs," *In Workshop on software and compiler for embedded systems (SCOPES), 2014 .*

[43] D. Potop-Butucaru and I. Puaut , " Integrated worst-case execution time estimation of multicore applications," *In 13rd Workshop on worst-case execution time analysis, 2013 .*

[44] T. Carle, "Static mapping of real-time applications onto massively parallel processor arrays," *In Application of Concurrency to System Design (ACSD), 2014 14th International Conference on, pages 112-121. IEEE, 2014.*

[45] Thierry Grandpierre and Yves Sorel , " From algorithm and architecture specifications to automatic generation of distributed real-time executives: a seamless flow of graphs transformations," *In Formal Methods and Models for Co-Design, 2003. MEMOCODE'03. Proceedings. First ACM and IEEE International Conference on, pages 123-132. IEEE, 2003 .*

[46] Pranav Tendulkar , Peter Poplavko , Ioannis Galanommatis and Oded Maler , " Many-core scheduling of data parallel applications using smt solvers," *In Digital System Design (DSD), 2014 17th Euromicro Conference on, pages 615-622. IEEE, 2014 .*

[47] Yu-Kwong Kwok and Ishfaq Ahmad , " Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Comput. Surv., 31(4):406-471,*

December 1999 .

[48] H. Nikolov , T. Stefanov and E. Deprettere , " Systematic and automated multiprocessor system design, programming, and implementation," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 27(3):542-555, March 2008 .*

[49] Louis-Noël Pouchet , Uday Bondhugula , Cédric Bastoul , Albert Cohen , J. Ramanujam and P. Sadayappan , " Combined iterative and model-driven optimization in an automatic parallelization framework," *In Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, SC'10, pages 1-11, Washington, DC, USA, 2010. IEEE Computer Society .*

[50] Lian Li , Lin Gao and Jingling Xue , " Memory coloring: A compiler approach for scratchpad memory management," *In Proceedings of the 14th International Conference on Parallel Architectures and Com-pilation Techniques, PACT '05, pages 329-338, Washington, DC, USA, 2005. IEEE Computer Society .*

[51] Isabelle Puaut , "WCET-centric software-controlled instruction caches for hard real-time systems," *In Real-Time Systems, 2006. 18th Euromicro Conference on, page 10 IEEE.*

[52] Heiko Falk and Jan C Kleinsorge , " Optimal static WCET-aware scratchpad allocation of program code," *In Proceedings of the 46th Annual Design Automation Conference, pages 732-737. ACM, 2009 .*

[53] Christophe Alias, Fabrice Baray and Alain Darte, "Bee+cl@k: An implementation of lattice-based array contraction in the source-to-source translator rose," *In Proceedings of the 2007 ACM SIGPLAN/SIGBED Conference on Languages, Compilers, and Tools for Embedded Systems, LCTES '07, pages 73-82, New York NY, USA, 2007.*

[54] Somashekaracharya G. Bhaskaracharya , Uday Bondhugula and Albert Cohen , " Smo: An integrated approach to intra-array and inter-array storage optimization," *In Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, pages 526-538 New York, NY, USA, 2016 .*

[55] Eddy De Greef , Francky Catthoor and Hugo De Man , " Array placement for storage size reduction in embedded multimedia systems," *In Application-Specific Systems, Architectures and Processors, 1997. Proceedings., IEEE International Conference on, pages 66-75. IEEE, 1997 .*

[56] A. Cilardo, "Improving multibank memory access parallelism with lattice-based partitioning," *ACM Trans. Archit. Code Optim., 11(4):45:1-45:25, January 2015.*

[57] R. Wilhelm et al , " The Worst-Case Execution Time Problem & Overview of Methods and Survey of Tools," *ACM Trans. Embed. Comput. Syst., vol. 7, no. 3, pp. 1-52, 2008 .*

[58] G. Fernandez et al , " Contention in Multicore Hardware Shared Resources: Understanding of the State of the Art," *no. Wcet, pp. 31-42, 2014 .*

[59] E. S. H. Hou , N. Ansari and H. Ren , " A Genetic Algorithm for Multiprocessor Scheduling," *IEEE Trans. P, vol. 5, no. 2, pp. 113-120, 1994 .*

[60] M. I. Daoud and N. Kharma , " A hybrid heuristic - genetic algorithm for task scheduling in heterogeneous processor networks," *J. Parallel Distrib. Comput., vol. 71, no. 11, pp. 1518-1531, 2011 .*

[61] Y. Kang and D. Zhang , " A Hybrid Genetic Scheduling Algorithm to Heterogeneous Distributed System," *Appl. Math., vol. 3, no. 7, pp. 750-754, 2012 .*

[62] Y. Xu , K. Li , J. Hu and K. Li , " A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues," *Inf. Sci. (Ny)., vol. 270, pp. 255-287, 2014 .*

[63] C. Reuter , M. Schwiegershausen and P. Pirsch , " Heterogeneous Multiprocessor Scheduling and Allocation using Evolutionary Algorithms," *in Proceedings IEEE International Conference on Application-Specific Systems, Architectures and Processors, 1997, pp. 294-303 .*

[64] S. Nesmachnow and E. Alba , " Heterogeneous computing scheduling with evolutionary algorithms," *Soft Comput., vol. 15, no. 4, pp. 685-701, 2010 .*

[65] H. Topcuoglu , S. Hariri and Min-You Wu , " Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing," *IEEE Trans. Parallel Distrib. Syst., vol. 13, no. 3, pp. 260-274, Mar. 2002 .*

[66] L. F. Bittencourt , R. Sakellariou and E. R. M. Madeira , " DAG Scheduling Using a Lookahead Variant of the Heterogeneous Earliest Finish Time Algorithm," *in 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing, 2010, pp. 27-34 .*

[67] S. C. Kim , S. Lee and J. Hahm , " Push-Pull: Deterministic Search-Based DAG Scheduling for Heterogeneous Cluster Systems," *IEEE Trans. Parallel Distrib. Syst., vol. 18, no. 11, pp. 1489-1502, Nov. 2007 .*

[68] H. Arabnejad and J. G. Barbosa , " List Scheduling Algorithm for Heterogeneous Systems by an Optimistic Cost Table," *IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 3, pp. 682-694, Mar. 2014 .*

[69] S. Ranaweera and D. P. Agrawal , " A task duplication based scheduling algorithm for heterogeneous systems," *in Proceedings 14th International Parallel and Distributed Processing Symposium. IPDPS 2000, 2000, pp. 445-450 .*

[70] M. Hosseinzadeh and H. S. Shahhoseini , " An effective duplication-based task-scheduling algorithm for heterogeneous systems," *Simulation, vol. 87, no. 12, pp.*

1067-1080, Dec. 2011 .

[71] A. Emeretlis , G. Theodoridis , P. Alefragis and N. Voros , " A Hybrid ILP-CP Model for Mapping Directed Acyclic Task Graphs to Multicore Architectures," *in 2014 IEEE International Parallel & Distributed Processing Symposium Workshops, 2014, pp. 176-182 .*

[72] A. Emeretlis , G. Theodoridis , P. Alefragis and N. Voros , " Mapping DAGs on Heterogeneous Platforms Using Logic-Based Benders Decompostion," *in 2015 IEEE Computer Society Annual Symposium on VLSI, 2015, pp. 119-124 .*

[73] A. Emeretlis , G. Theodoridis , P. Alefragis and N. Voros , " A Logic-Based Benders Decomposition Approach for Mapping Applications on Heterogeneous Multicore Platforms," *ACM Trans. Embed. Comput. Syst., vol. 15, no. 1, pp. 1-28, Feb. 2016 .*

[74] J. Rosén , A. Andrei , P. Eles and Z. Peng , " Bus access optimization for predictable implementation of real-time applications on multiprocessor systems-on-chip," *Proc. - Real-Time Syst. Symp., pp. 49-60, 2007 .*

[75] H. Kopetz , " Real-Time Systems, Design Principles for Distributed Embedded Applications 1997".

[76] P. Pop , P. Eles , Z. Peng and T. Pop , " Analysis and optimization of distributed real-time embedded systems," *ACM Trans. Des. Autom. Electron. Syst., vol. 11, no. 3, p. 625, 2006 .*

[77] J. Rosén et al , " Bus access design for combined worst and average case execution time optimization of predictable real-time applications on multiprocessor systems-on-chip," *Real-Time Technol. Appl. - Proc., pp. 291-301, 2011 .*

[78] H. Ding , Y. Liang and T. Mitra , " Shared cache aware task mapping for WCRT minimization," *Proc. Asia South Pacific Des. Autom. Conf. ASP-DAC, vol. 2, pp. 735-740, 2013 .*

[79] D. Hardy and I. Puaut , " WCET analysis of multi-level non-inclusive set-associative instruction caches," *Proc. - Real-Time Syst. Symp., pp. 456-466, 2008 .*

[80] Y. Li , V. Suhendra , Y. Liang , T. Mitra and A. Roychoudhury , " Timing Analysis of Concurrent Programs Running on Shared Cache Multi-Cores," *2009 30th IEEE Real-Time Syst. Symp., pp. 57-67, 2009 .*

[81] M. John and M. Jacobs , " A Framework for the Optimization of the WCET of Programs on Multi-Core Processors," *Proc. 8th Jr. Res. Work. Real-Time Comput., pp. 2-5, 2014 .*

[82] T. Kelter , H. Borghorst and P. Marwedel , " WCET-aware scheduling optimizations for multi-core real-time systems," *Proc. - Int. Conf. Embed. Comput.*

Syst. Archit. Model. Simulation, SAMOS 2014, pp. 67-74, 2014 .

[83] V. A. Nguyen , D. Hardy and I. Puaut , " Scheduling of parallel applications on many-core architectures with caches?: bridging the gap between WCET analysis and schedulability analysis, 2015".

[84] S. Chattopadhyay and A. Roychoudhury , " Static bus schedule aware scratchpad allocation in multiprocessors," *ACM SIGPLAN Not., vol. 47, no. 5, p. 11, 2012 .*

[85] Y. Kim , D. Broman , J. Cai and A. Shrivastaval , " WCET-aware dynamic code management on scratchpads for Software-Managed Multicores," *2014 IEEE 19th Real-Time Embed. Technol. Appl. Symp., pp. 179-188, 2014 .*

[86] The Mälardalen WCET research group , “WCET Project / Benchmarks,” [Online]. Available: http://www.mrtc.mdh.se/projects/wcet/benchmarks.html.

[87] University of Michigan, "MiBench Suite webpage," [Online]. Available: http://vhosts.eecs.umich.edu/mibench/.

[88] M. Paolieri, E. Quiñones, F. J. Cazorla, G. Bernat and M. Valero, "Hardware support for WCET analysis of hard real-time multicore systems," in *ACM SIGARCH Computer Architecture News*, 2009.

[89] R. Kirner and P. Puschner, "Obstacles in worst-case execution time analysis," in *Object Oriented Real-Time Distributed Computing (ISORC), 2008 11th IEEE International Symposium on*, 2008.

[90] M. Schoeberl, S. Abbaspour, B. Akesson, N. Audsley, R. Capasso, J. Garside, K. Goossens, S. Goossens, S. Hansen, R. Heckmann, S. Hepp, B. Huber, A. Jordan, E. Kasapaki, J. Knoop, Y. Li, D. Prokesch, W. Puffitsch, P. Puschner, A. Rocha, Cláudio Silva, Jens Sparsø and A. Tocchi, "T-CREST: Time-predictable multi-core architecture for embedded systems," *Journal of Systems Architecture,* vol. 61, no. 9, pp. 449-471, 2015.

[91] "Project ARAMiS," [Online]. Available: http://www.projekt-aramis.de.

[92] T. Ungerer, C. Bradatsch, M. Frieb, F. Kluge, J. Mische, A. Stegmeier, R. Jahr, M. Gerdes, P. Zaykov, L. Matusova and others, "Experiences and Results of Parallelisation of Industrial Hard Real-time Applications for the parMERASA Multi-core," in *Submitted to the 3nd Workshop on High-performance and Real-time Embedded Systems (HiRES 2015), Amsterdam, the Netherlands*, 2015.

[93] M. Zimmer, D. Broman, C. Shaver and E. A. Lee, "FlexPRET: A processor platform for mixed-criticality systems," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2014 IEEE 20th*, 2014.

[94] M. Fernández, R. Gioiosa, E. Quiñones, L. Fossati, M. Zulianello and F. J. Cazorla, "Assessing the suitability of the NGMP multi-core processor in the space domain," in *Proceedings of the tenth ACM international conference on Embedded*

software, 2012.

[95] J. Nowotsch, M. Paulitsch, D. Buhler, H. Theiling, S. Wegener and M. Schmidt, "Multi-core interference-sensitive wcet analysis leveraging runtime resource capacity enforcement," in *Real-Time Systems (ECRTS), 2014 26th Euromicro Conference on*, 2014.

[96] L. Benini and G. D. Micheli , " Networks on chips: a new SoC paradigm," *Computer, 2002 .*

[97] A. Agarwal , " The tile processor: A 64-core multicore for embedded processing," *In HPEC, 2007 .*

[98] G. Smit , E. Schuler , J. Becker , J. Quevremont and W. Brugger , " Overview of the 4S project," *In ISSOC, 2005 .*

[99] N. Voros , A. Rosti and M. Hubner , " Dynamic System Reconfiguration in Heterogeneous Platforms: The MORPHEUS Approach," *Springer, 2009 .*

[100] J. Howard , S. Dighe and Y. H. et. al , " A 48-Core IA-32 message-passing processor with DVFS in 45nm CMOS," *In ISSCC, 2010 .*

[101] J. Teich , " Invasive Algorithms and Architectures," *it -Information Technology, 2008 .*

[102] J. Teich , J. Henkel and H. et. al , " Invasive Computing: An Overview," *In M. Hubner and J. Becker, editors, Multiprocessor System-on-Chip: Hardware Design and Tool Integration. Springer, 2011 .*

[103] R. T. Mikael Millberg , Erland Nilsson and A. Jantsch , " Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip," *In DATE, 2004 .*

[104] K. Goossens , J. Dielissen and A. Radulescu , " The real network on chip: Concepts, architectures, and implementations," *IEEE Design & Test, 2005 .*

[105] E. Bolotin , I. Cidon , R. Ginosar and A. Kolodny , " QNoC: QoS architecture and design process for network on chip," *Journal of Systems Architecture, 2004 .*

[106] A. Malek, I. Sourdis, S. Tzilis, Y. He and G. Rauwerda, “RQNoC: A Resilient Quality-of-Service Network-on-Chip with Service Redirection,” *ACM Transactions on Embedded Computing Systems (TECS), Volume 15, Issue 2, March 2016.*

[107] Kavaldjiev and N. et. al , " Providing QoS guarantees in a NoC by virtual channel reservation," *In Bertels and K. et. al., editors, Recon_gurable Computing: Architectures and Applications. 2006 .*

[108] J. Diemer , R. Ernst , M. Kauschke and E_cient , " throughput-guarantees for latency-sensitive networks-on-chip," *In ASP-DAC, 2010 .*

[109] T. Bjerregaard and J. Sparso , " A router architecture for Connection-Oriented service guarantees in the MANGO clockless Network-on-Chip," *In DATE, 2005 .*

[110] Hichem Karray , Michael Paulitsch , Bernd Koppenhoefer and Dietmar Geiger, "Design and implementation of a degraded vision landing aid application on a multicore processor architecture for safety-critical application," *In 16th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC). IEEE , pp. 1-8.,* 2013.

[111] D. Kissler , F. Hannig , A. Kupriyanov and J. Teich , " A Highly Parameterizable Parallel Processor Array Architecture," *In FPT, 2006 .*

[112] Z. Guz , I. Walter , E. Bolotin , I. Cidon , R. Ginosar and A. Kolodny , " Network delays and link capacities in application-speci_c wormhole nocs," *In VLSI Design, 2007 .*

[113] E. L. de Souza Carvalho , N. L. Calazans and F. G. Moraes , " Dynamic task mapping for MPSoCs," *IEEE Design & Test of Computers, 2010 .*

[114] W. Dally , " Virtual-channel low control," *Parallel and Distributed Systems, 1992 .*

List of Figures

List of Tables

Glossary of Terms

AOLP	Angle Of Linear Polarization
ACET	Average Case Execution Time
ADL	Architecture Description Language
AESS	Aircraft Environmental Surveillance System
CPU	Central Processing Unit
DOLP	Degree Of Linear Polarization
DSL	Domain-Specific Language
DSP	Digital Signal Processor
ECU	Electronic Control Unit
FPGA	Field Programmable Gate Array
EGPWS	Enhanced Ground Proximity Warning System
GPU	Graphics Processing Unit
HDL	Hardware Description Language
ILP	Instruction Level Pipelining
IMA	Integrated Modular Avionics
IPP	Integrated Performance Primitives
MBD	Model-Based Design
MCA	Multicore Association
MPSoC	Multiprocessor System on Chip
NoC	Network on Chip
LRU	Line Replaceable Unit
PE	Processing Element
RAM	Random Access Memory
SHIM	Software Hardware Interface for Multi-Many-Core
SIMD	Single Instruction Multiple Data

SoC	System-on-Chip
SSE	Streaming SIMD Extensions
SPM	ScratchPad Memory
TAD	Terrain Awareness Display
TCF	Terrain Clearance Floor
TDMA	Time Division Multiple Access
TAWS	Terrain Awareness and Warning Systems
TIWG	Tool Infrastructure Working Group
VEMPIRE	Versatile Embedded Platform for Image Recognition
WCET	Worst Case Execution Time
WEAA	Wake Encounter Advisory and Avoidance System