A Model-driven development framework for highly Parallel and EneRgy-Efficient computation supporting multi-criteria optimisation

D3.1 Multi-criteria optimization requirements

Version 1.1

Documentation Information

Contract Number	871669
Project Website	www.ampere-euproject.eu
Contractual Deadline	30.06.2020 (delivery on 30.09.2020 due to COVID-19 situation)
Dissemination Level	PU
Nature	R
Author	Luis Miguel Pinho (ISEP), Björn Forsberg (ETHZ), Alessandro Biondi (SSSA), Tommaso Cucinotta (SSSA), Sara Royuela (BSC)
Contributors	Luis Nogueira (ISEP), Thomas Benz (ETHZ)
Reviewer	THALIT
Keywords	Non-functional requirements, multi-criteria optimization, energy models, timing and schedulability analysis, resilience methods

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 871669.

Change Log

Version	Description Change
V0.1	First draft
V0.5	Inputs from ISEP, SSSA, ETHZ
V0.6	Consolidation of inputs from ISEP, SSSA, ETHZ
V0.7	Review of THALIT
V0.8	Revised version and inputs from BSC
V1.0	First complete release
V1.1	Final revision

Table of Contents

1 Executive Summary

This deliverable covers the work done during the first phase of the project within WP3. The deliverable spans 9 months' work (including 3 extra months with respect of the Grant Agreement [1] due to the COVID19 situation), and provides results of the work done in Task 3.1 "Multi-criteria optimisation requirements specification" for milestone 1 (MS1).

The document provides the analyses and consolidation of the requirements related to the multi-criteria optimization associated to the non-functional constraints considered in AMPERE (i.e., energy-efficiency, time-criticality and fault tolerance), and describes the initial techniques that will be considered in the scope of the development of the work package for later milestones and tasks (i.e., energy models, timing and schedulability analysis and software and hardware resilient methods) and that better fit parallel execution.

The target at MS1 is the evaluation of energy models, time and schedulability and resilient techniques that better fit parallel execution. The first milestone of Task 3.1 has been carried out successfully, and all objectives of MS1 have been reached and documented in this deliverable.

2 Introduction

AMPERE addresses the challenge of fully exploiting the benefits of performance demanding emerging applications (such as artificial intelligence or big data analytics), which can only be met on parallel platforms composed of different heterogeneous computing resources, whilst guaranteeing the energy efficiency, real-time response and resiliency non-functional requirements, required by cyber-physical applications.

The goal of WP3 is thus to investigate and provide a set of analyses, which are able to perform a multi-criteria optimization at development phase, guiding the model-driven to programming model transformation and ensuring that non-functional constraints are fulfilled, and devise execution models and methods to guarantee their fulfilment at run-time, considering the underlying runtimes and platforms.

In particular, and in what respects to energy-efficiency, WP3 will (i) investigate the energy consumption components present in the selected parallel platform(s) and how are impacted by the different power management knobs and run-time decisions; (ii) devise methods to extract information on (1) workload specification, (2) non-functional constraints included in the parallel programming model and (3) hardware platform characteristics impacting on energy; and (iii) develop energy-aware execution models based on previous information that push the selected parallel platform(s) introspection capabilities beyond what is currently feasible.

With respect to time predictability, WP3 will (i)investigate and develop predictable execution models of the selected parallel platforms to simplify timing and schedulability analysis, including optimizations in the placement of functionality into cores, offloading operations, DPR operations and computation on accelerators; and (ii) develop timing and schedulability analyses for the proposed scheduling algorithms and heterogeneous execution models developed in WP4.

With respect to fault tolerance, WP3 will improve the system's fault tolerance, considering reliability and availability aspects resulting in improved system's dependability.

This document provides the initial baseline for the work of WP3, in particular the requirements related to the multi-criteria optimization associated to the non-functional constraints considered in AMPERE (i.e., energy-efficiency, time-criticality and fault tolerance). Requirements are listed one-by-one together with the following attributes:

- ID: the requirement identifier.
- Topic: the main system the requirement is applied to. Requirements have been analysed and consolidated in the following topics: Timing, Energy, Communication, Security
- Subtopic: the category of the requirement.
- Name: the friendly name of the requirement
- Description: the body of the requirement, with the associated metrics.
- Means for verification: the way this requirement will be evaluated within the project.
- Type: the type of requirement defined according to the MoSCoW Model:
    - MUST HAVE (M): Defines a requirement that has to be satisfied for final solution to be acceptable. It is mandatory.
    - SHOULD HAVE (S): This is a high-priority requirement that should be included if possible.

    - COULD HAVE (C): This is a desirable or nice-to-have requirement, but the solution will still be accepted if the functionality is not included.
    - WOULD LIKE (W): This represents a requirement that stakeholders would like to have but have agreed will not be implemented within the scope of this project
- Implementer: the WP responsible of the requirement capture within the project
- Source: Indicates where this requirement comes from.

The document also puts together the initial techniques that will be considered in the scope of the development of the work package for later milestones and tasks (i.e., energy models, timing and schedulability analysis and software and hardware resilient methods). These models will consider predictable high performance as the main goal, and will be based on techniques that better fit parallel execution, addressing the baseline requirements.

The relation of this deliverable with other WP is shown in the next Table 1, considering deliverables and tasks that are involved with the studies performed for completing this deliverable.

Table 1. Relationship between D3.1 and other deliverables.

Deliverable	Leader	Task	Description
D1.1	THALIT	T1.1	System models requirements specification and use case selection
D2.1	BSC	T2.1	Model transformations requirements
D4.1	SSSA	T4.1	Run-time requirement specification
D5.1	SYSGO	T5.1	Reference parallel heterogeneous hardware selection

2.1 Document structure

This document is organized in 7 sections:

- Section 1 provides an Executive Summary of the document.
- Section 2 introduces briefly the context and gives a main view of the structure of the document.
- Section 3 gives a general overview of the constraints related to energy-efficiency, and the initial energy models and optimization strategies to be considered within AMPERE.
- Section 4 gives a general overview of the constraints related to time-criticality, and the initial timing and scheduling analysis to be considered within AMPERE.
- Section 5 gives a general overview of the constraints related to fault-tolerance, and the initial hardware and software resilient methods to be considered within AMPERE.
- Section 7 provides a summary and conclusion of the document.

3 Energy Models

This section summarizes the requirements related to energy-efficency, collected from the work in WP1 (Task T1.1 “System model requirement specification and use case definition” [2]) as well as from WP2 (Task 2.1 “Model transformation requirements specification” [3]), WP4 (Task 4.1 “Run-time requirement specification” [4]) and the platforms from WP5 (Task 5.1 “Platform Selection” [5]).

To design an energy-efficient AMPERE eco-system, we need to track energy usage during the execution of an AMPERE system, and the data gathered about energy consumption is analysed during the offline optimization stage.

The energy consumption of software components running on complex modern CPUs that are in the focus of the AMPERE high-performance embedded application use-cases, cannot simply be modelled using simplistic approaches relying solely on knowledge of the configuration of the DVFS tunables of the platform (frequency of the CPU cores), accompanied by the workload/idle ratio over each CPU core. Indeed, during the execution of a workload that at a high-level is keeping a CPU core continuously busy processing, the energy consumption may vary in non-negligible ways depending on what parts of the CPU pipeline are actively engaged vs stay idle, due to the instructions actually in execution, and how long the cores stall waiting for data from the main memory due to cache misses, for example. This can be highlighted with a simple experiment, as reported in the figure below.

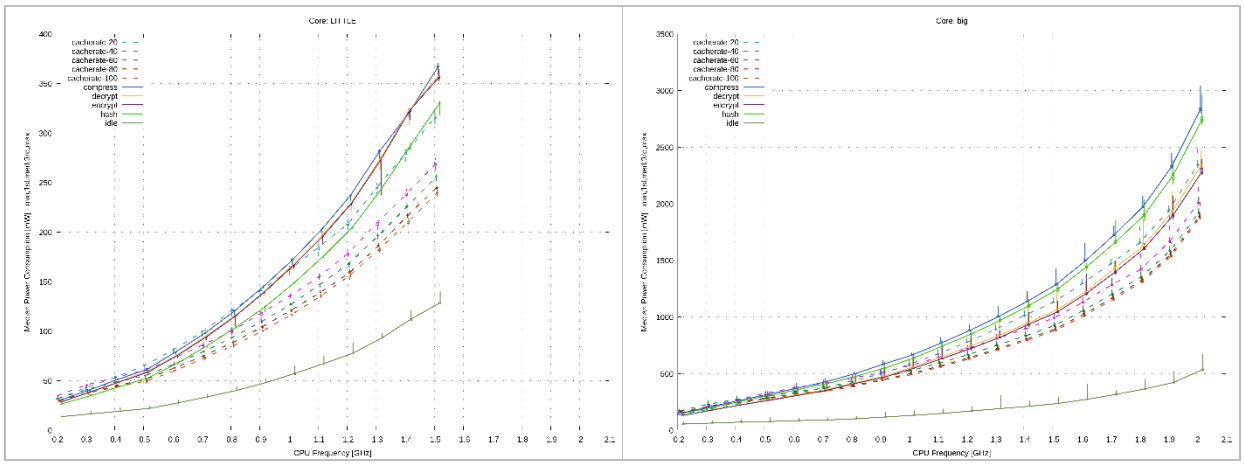

Figure 1. Execution of benchmarks in the LITTLE and big cores of ARM big.LITTLE ODROID-XU3 board.

We have run a number of common CPU-intensive application benchmarks on one of the LITTLE and big cores of an ARM big.LITTLE ODROID-XU3 board. Albeit not being the UltraScale+ board selected for the project, this board possesses interesting non-SMP features in its architecture, constituting an excellent challenge for investigating soft real-time guarantees vs energy efficiency trade-offs. On the left and right subplots, we can see what power consumption (on the Y axis) corresponds to each of the possible OPPs/frequencies (on the X axis) configurable for the LITTLE and big island, respectively. As evident, the curves corresponding to various application workloads (continuous lines), as well to synthetic data-intensive computational hogs imposing approximate percentages of cache misses (dashed lines), obtain different power consumptions at equal OPP configurations. This underlines the importance of tracking, modelling and estimating the energy

consumption of applications. Clearly, a real application may dynamically switch among different types of computations, making the power consumption estimation task even more challenging, and calling for on-line solutions that can adaptively refine initially rough estimations monitoring the applications at run-time.

At run-time, the energy usage of the system may be tracked by estimating the energy usage with a linear model, which is trained during the offline stage. The overall picture is presented in Figure 2. The platform specific countable hardware and software events (A) and the state of the software-tunable energy-relevant parameters, such as DVFS configuration (B) are used as input for the offline-trained model (C), which delivers an energy prediction (D) – i.e., an estimation of the current usage – which is used at runtime to impact the software-tunable parameters (B).

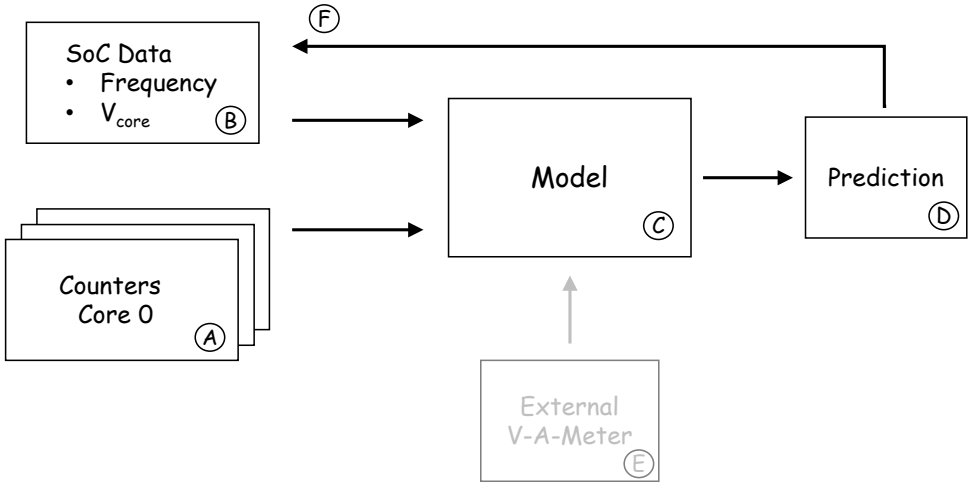

Figure 2. A schematic overview of the interaction of the energy consumption model.

3.1 Model Rationale

The energy usage of the system is given by the power usage and the time it is spent. Fundamentally, the power used by the system consists of a static (leakage) component, and a dynamic component. The dynamic component depends on the activity in the hardware, e.g., as highlighted with alphas in Figure 3. Taking these two components into account, the power usage of the system can be estimated as shown below. Each signal alpha is assigned a weight w that represents its contribution to the system power use, which are summed up with the static power for each signal.

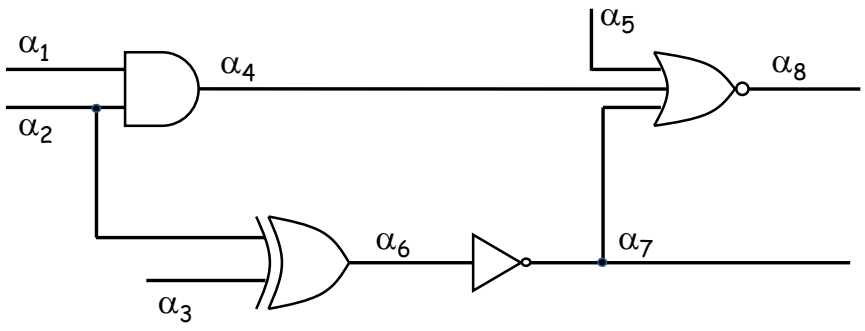

Figure 3. Signaling activity represented by different alphas in a mockup-system.

$$P_{Est} = \sum \alpha_i * w_i + P_{stat} \quad (1)$$

However, it is unfeasible to analyse the system at this granularity. First, because this low-level information is typically not available for the commercial platforms used in the system, and secondly that the time to estimate would make the multi-criteria optimization pass of the AMPERE framework unbearably slow. These two aspects mainly affect the offline phase of the energy-efficiency optimization. Furthermore, it is not possible to track the system at this granularity at runtime, making the monitoring of individual components during execution impossible.

For this reason, as is commonly done, the system is abstracted from individual signals into larger functional units. At this point, the power estimation can still be done in a similar way, using the below equation. In this case, a new weight w is assigned to each functional unit alpha (as opposed to the previous signals), depending on its contribution to the system power usage. Importantly, while the equations look similar, the abstraction into functional units alpha vastly reduces the number of terms in the summation, making it feasible to fit a linear model to obtain the weights (offline), and run inference on the model at runtime. In particular the latter becomes feasible in terms of the low overhead runtime-monitoring (KPI3.2) only at this level. For the offline training (fitting) of the model, abstracting the system into larger functional units has one additional advantage. While information about individual signals in commercial platforms are not available, information about functional units, e.g., ALU, SIMD engine, etc., are readily available.

$$P_{Est} = \sum \alpha_i * \bar{w}_i + P_{stat} \quad (2)$$

The final form of the model is given by the below equation. It assumes that power is linear in functional units, as well as that all relevant functional units can be tracked.

$$P_{Est} = \sum_{i=0}^{NumCores} \left[s_i * \left(P_{gated} - P_{leakage} \right) + P_{leakage} + \sum_{j=0}^{NumCounters} (1 - s_i) * \alpha_j * v^2 * f * C \right] \quad (3)$$

Based on this information, the functional units of the selected platforms, the Xilinx UltraScale+ and the NVIDIA Xavier (see D5.1) are identified. Both commercial platforms used within the project feature a multi-core ARMv8 CPU as host processor. These processors can be abstracted into multiple functional units, e.g., ALU, SIMD engine, etc. To track the activity within each functional unit, the performance counters can be used [6]. The activity in the ARM cores can be tracked by using the ARM Performance Monitoring Unit Version 3 (PMUv3). The PMUv3 provides the infrastructure to count a number (3 on Xavier, 6 on US+) of hardware events (e.g., bus access, completed cycles, etc.). These events are primarily intended for performance monitoring purposes, but provide access to information about the functional units of the processor necessary for energy/power estimation.

ID	**REQ-ENE-01**
Topic	Energy Efficiency
Subtopic	Energy Modeling
Name	Access to Hardware Events for Energy Modeling
Description	The AMPERE eco-system must support the tracking of hardware event counters, and ensure that counter access does not conflict between different parts of the AMPERE runtime.
Means for verification	Demonstrators
Type	M
Implementer(s)	WP3
Source	Platform

If there is competition for different parts of the runtime to use the performance monitoring unit, it may be worth exploring if, during the offline optimization phase, changes in the program execution pattern (e.g., memory vs compute bound segments) can be identified, and hooks to the energy optimization runtime can be injected into the code. Performing such on-demand adjustments to the software-controlled mechanisms for energy management may also serve to reduce the runtime overhead.

3.2 On the importance of representative counters

The first step is to identify which counters that correlate well with the energy usage of the system. By tracking these counters, the behaviour of the most relevant functional units are implicitly identified. For this purpose we use a technique similar to [7]

. An example for two benchmarks and a set of counters is shown in Figure 4.

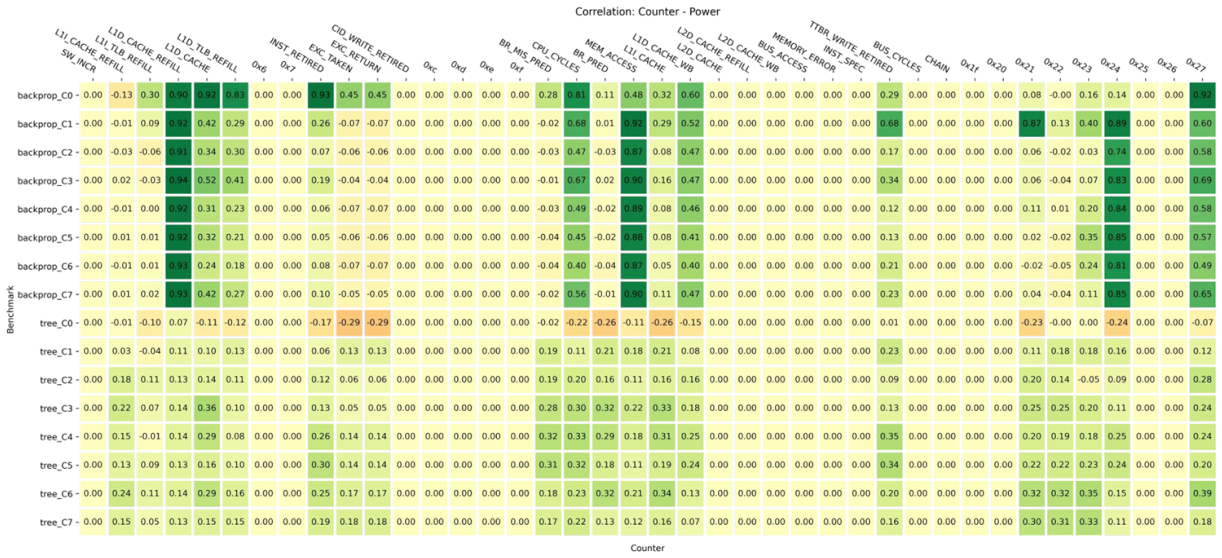

Figure 4. Correlation of hardware events (from performance counters) to energy usage of the NVIDIA Xavier.

Figure 4 shows one row for each out of the eight cores, for two benchmarks. Each column represents a counter. Counters with a high positive correlation to power usage are highlighted in green, counters with a high negative correlation are highlighted in red, and all other counters with low correlation are highlighted in bright yellow. As can be seen in the figure, there are a number of counters with high correlation to energy, which we include in the model. In particular, this includes the L1 data cache refill event, the number of memory accesses, the completed cycles, and the unnamed counter 0x27, which is triggered with SIMD-heavy computation is executed on the Xavier.

How important the selection of relevant counters is can be seen in the following two traces, shown in Figure 5 and Figure 6. The figure shows in orange the externally measured power usage of the system, while the blue line shows the estimated power based on the model. The trace shows the execution of several Rodinia benchmarks [8]. In Figure 5, the model does not include the SIMD event counter, and as can be seen, especially between the 5000 and 6000 ms marks, the estimated power does not follow the actual power usage very well. If instead the model is retrained with the

SIMD event counter included, as shown in Figure 6, the estimated power follows the true value more closely.

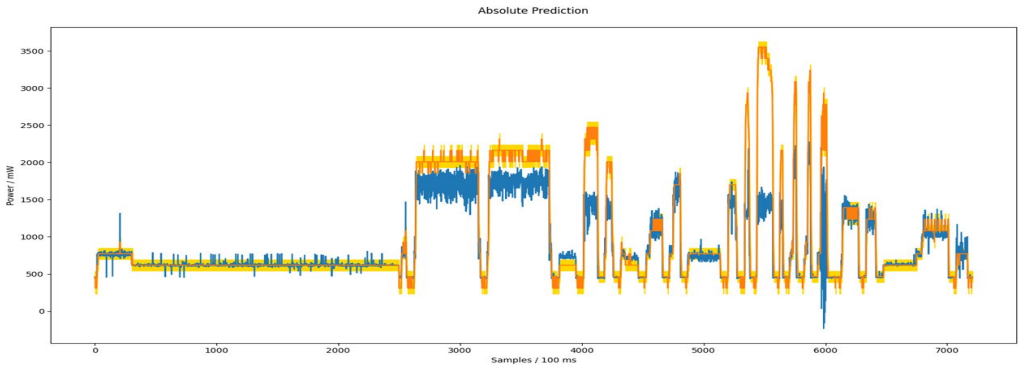

Figure 5. The estimated energy use (blue) and the ground truth (orange line, yellow measurement error) without including the SIMD unit.

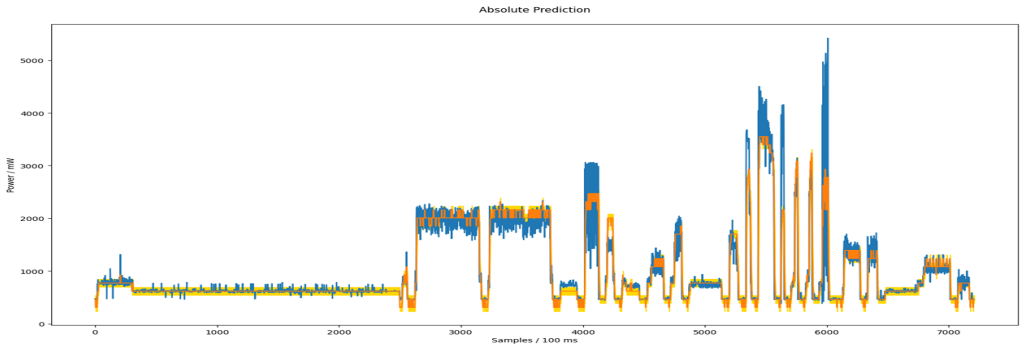

Figure 6. The estimated energy use (blue) and the ground truth (Orange line, yellow measurement error), including the SIMD unit.

At this point, it is tempting to imagine that the system can be well-modelled by devising synthetic workloads that stress each individual functional unit independently. For this purpose, we devise synthetic workloads ALU, FLOAT, MEM, BRANCH, in single- and multi-threaded configurations, which we use to train the model. However, as can be seen in Figure 7, under these circumstances the estimated power deviates significantly from the actual power. This behaviour is specific to the more complex processors in the commercial platforms, but works well for simpler processing units, as the RISC-V-based processing elements (PE) in HERO[1], using only synthetic workloads. This indicates that a combination of these approaches can potentially be used to cover all units of all platforms within the project.

[1] HERO, the Heterogeneous Research Platform, is a programmable many-core accelerator system developed at ETHZ. It consists of a 64-bit RISC-V host processor and one or more clusters of simple RISC-V cores (processing elements, PE) for parallel processing. This system can be used, as per the WP5 description, as a research platform within the project to explore solutions in hardware that are not available in the commercial platforms. For more information, please see https://arxiv.org/pdf/1712.06497.pdf

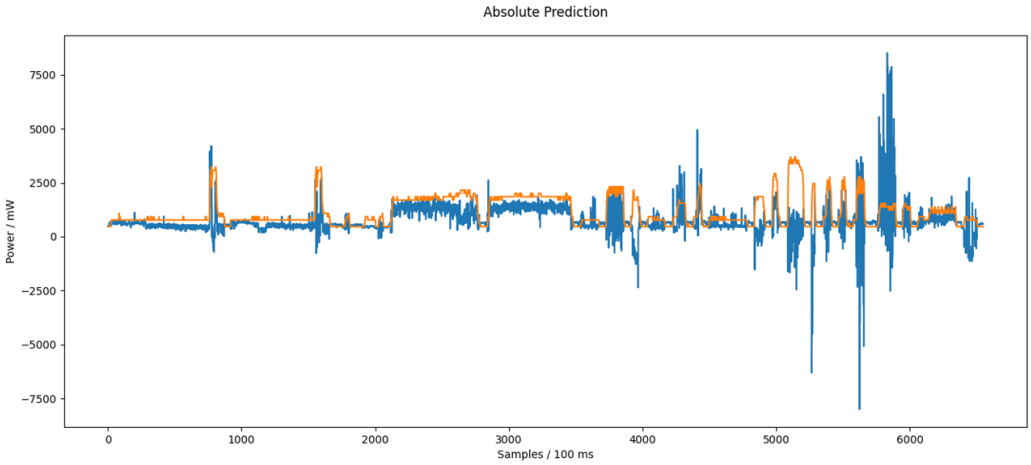

Figure 7. The bad energy estimation (blue) compared to the ground truth (Orange) when training the model with synthetic benchmarks.

An issue that was up until now not discussed, but that is also relevant to achieve the good estimates previously presented is the power gating of the processor.

When the processor is power gated, it ideally uses no dynamic or leakage power, and therefore greatly impacts the overall energy usage of the system. Unfortunately, in the case of the Xavier processors, gating cannot be controlled by software, and as such must be included in the model to achieve a good estimate.

Thankfully, this information can be derived from the cycle counter of the PMU. In Figure 8, the cycle counter for a benchmark execution is shown on the Y axis, as measured at each point in time (X axis). During this execution, the operating point has been fixed, i.e., the frequency remains constant, yet, as shown in the Figure, there is large variance in the number of recorded active cycles. At the points in time where few cycles are recorded, the processor has been power gated. In the shown example, this happens automatically and corresponds to the phases of the executing program when a significant portion of time is spent waiting on data from memory. Thus, even if the system is configured for a fixed operating point, low-level hardware features within the platform is constantly optimizing the energy usage, turning of unused components in real-time and without explicit management from software.

ID	**REQ-ENE-02**
Topic	Energy Efficiency
Subtopic	Energy Modeling
Name	Access to Active Cycle Counter
Description	The AMPERE eco-system must support the tracking of the Active Cycle Counter, to ensure software-awareness of hardware controlled events, such as power gating.
Means for verification	Demonstrators
Type	M
Implementer(s)	WP3
Source	Platform

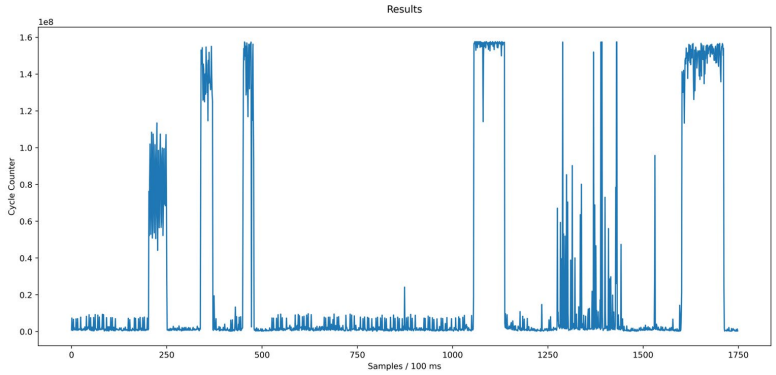

Figure 8 The cycle counter for a sequence of Rodinia benchmarks, while keeping the processor executing at a fixed frequency.

By correctly tracking the effects of power gating in the energy estimates using the active cycles counter from the PMU, the energy usage of the system can be well estimated, as shown in Figure 9. The figure shows the data from the board's current/voltage monitor as an orange line, the quantization error associated with it as a yellow band, and the estimation shown as a blue line. On average, the estimated energy usage (blue) gives the same result as the current/voltage monitor, while at each individual point in time the average deviation is within +/-15%.

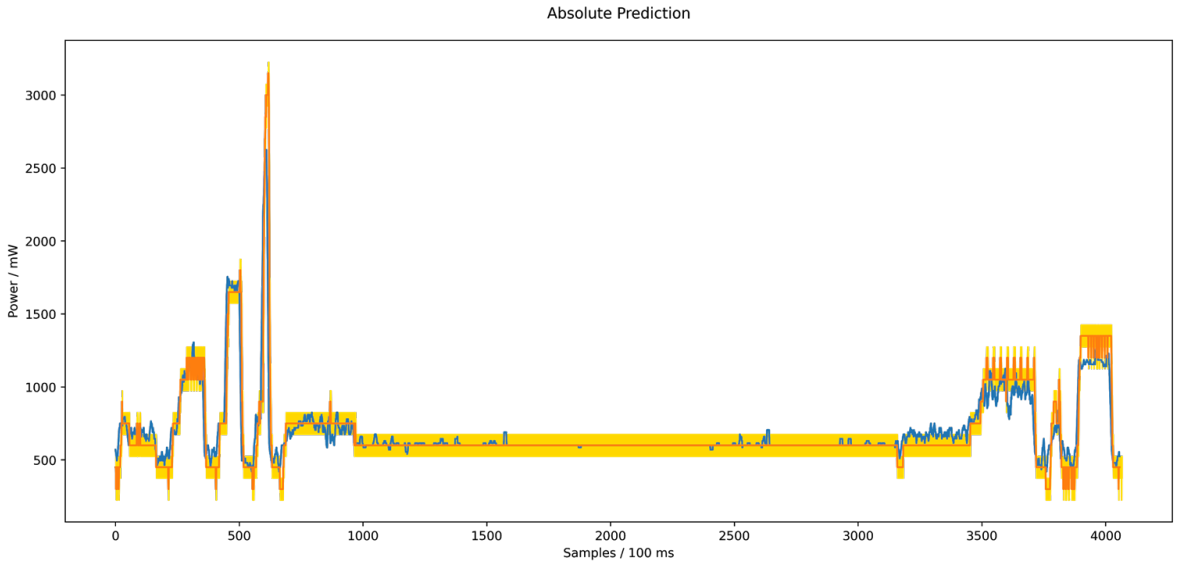

Figure 9. Well-estimated energy usage (blue) compared to ground truth (Orange, yellow measurement error), when Accounting for clock gating through monitoring the active cycle counter.

3.3 Impact to Offline Optimization Stage

The training of the runtime model requires representative training data. Our goal is to provide a pre-trained model that does not require re-training of this model when adopting the system, however, the ability to do that should be present in the AMPERE eco-system. This is in line with the envisioned profiling and adaptation "backloops" in the AMPERE multi-criteria optimization design flow, as shown in Figure 10.

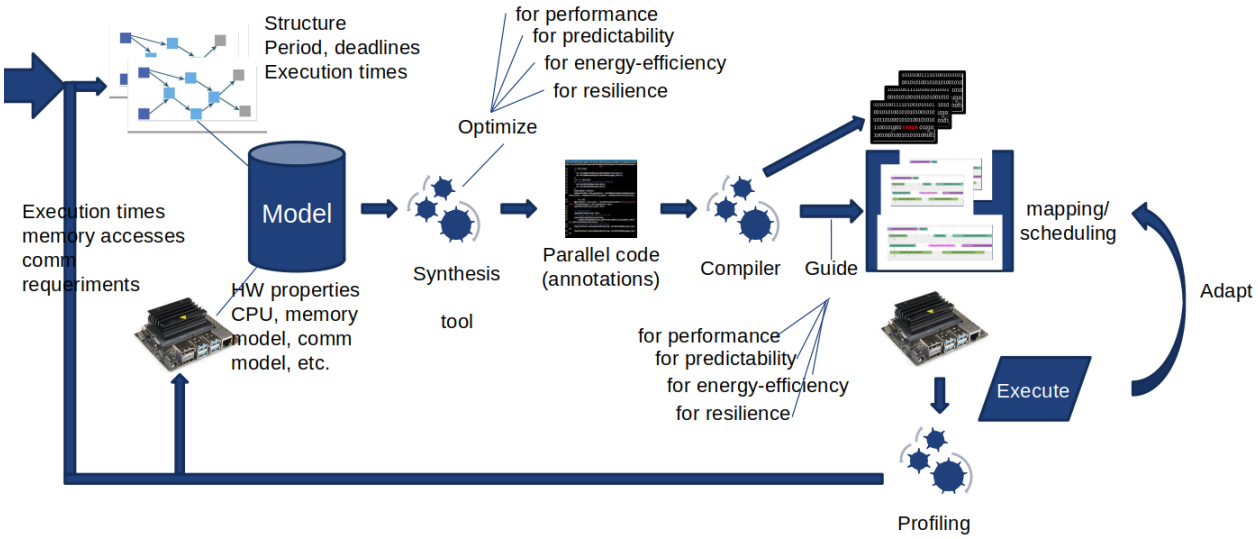

Figure 10. The envisioned flow for the multi-critiera optimization phase.

ID	**REQ-ENE-03**
Topic	Energy Efficiency
Subtopic	Multi-criteria Optimization Pipeline
Name	Profiling and Adaptation Mechanisms for Multi-Criteria Optimization
Description	The AMPERE multi-criteria optimization pipeline must establish means for profiling and subsequent adaptations during the optimization phase.
Means for verification	Test
Type	M
Implementer(s)	WP3
Source	Programming Model, Platform

To perform initial energy-efficiency optimization, there must be an offline model for the initial energy estimation. While the presented model is able to provide very accurate estimations of the energy use, it currently relies on information that is only available online, i.e., the impact of hardware-controlled clock gating, as well as the activity in different hardware units. As part of the offline optimization stage, an important aspect is to determine how the heterogeneity of the system can be best used, and as such, it must be possible to understand what the energy requirements are for different workloads on the different processing devices (e.g., CPU, GPU, …) within the system. For this purpose, a simplified model can be used that assumes the worst-case clock-gating behaviour (always on), or a heuristic model that encompasses information about computational patterns that trigger gating behaviour, e.g., memory intensive code regions. This simplifies the input data to the model, but requires more details of the energy impact of different code constructs and instructions to be included in the platform model. As a starting point, a model such as [9] can be used. By including separate energy-performance pairs for each DVFS operating point in the platform model, a safe minimum frequency, i.e., minimum energy-level, can be assigned during the offline optimization stage. By including this information for all processing devices, heterogeneity of the system can be explored from an energy-efficiency perspective in the offline optimization phase.

ID	REQ-ENE-04	
Topic	Energy Efficiency	
Subtopic	Platform Model	
Name	Specification of Hardware Energy Characteristics in Platform Model	
Description		The AMPERE platform models used for the optimization pipeline must include energy-characteristics for the supported platforms, to allow offline estimation of energy requirements for different code constructs. This ensures the possibility for energy analysis without having to resort to profiling.
Means for verification		Inspection
Type		M
Implementer(s)		WP3
Source		Platform, Programming Model

It should therefore be the goal of the offline optimization stage to provide safe upper bounds for the energy-usage of the system, which can be improved with runtime information. As part of the work in Task 4.2, ETHZ will explore techniques that allow tighter bounds to be generated from offline information (system model + code) with respect to online data (hardware events counted). This allows the employment of techniques such as (energy) slack reclamation and borrowing mechanisms [10, 11, 12] to be used to further improve the energy efficiency at runtime, based on the online optimization. During the offline stage, we envision energy budget information to be encoded for each task, such that the AMPERE runtime can further optimize during execution.

ID	REQ-ENE-05	
Topic	Energy Efficiency	
Subtopic	Modelling Framework	
Name	Encoding of Non-Functional Energy Requirements in Application Models	
Description		Executable runnables and generated code must, where applicable, have an energy budget per activation assigned to them. This ensures that a) analysis can verify that non-functional requirements are upheld, and b) enable further optimization at runtime by reclaiming unused energy budgets.
Means for verification		Demonstrators
Type		M
Implementer(s)		WP3
Source		Platform

4 Timing and Schedulability Analysis

This section summarizes the requirements related to timing and schedulability analysis, collected from the work in WP1 (Task T1.1 "System model requirement specification and use case definition" [2]) as well as from WP2 (Task 2.1 "Model transformation requirements specification" [3]), WP4 (Task 4.1 "Run-time requirement specification" [4]) and the platforms from WP5 (Task 5.1 "Platform Selection" [5]).

4.1 Time-criticality requirements

The use cases considered by AMPERE (railway obstacle detection and avoidance system and automotive intelligent predictive cruise control), are representative of cyber-physical systems, where the interactions between the computing systems and the environment are made through control loops, which have clear semantics of a sense-compute-actuate chain. Due to the critical nature of some of these control loops, the AMPERE eco-system must guarantee that the response time of the critical chains is within specified bounds, imposed by the requirements of the external controlled physical system.

ID	**REQ-TIM-01**
Topic	Timing and Schedulability Analysis
Subtopic	Control loops
Name	Types of control loops
Description	The AMPERE eco-system must support control loops which consist of event-chains and/or computation graphs, with end-to-end response time requirements.
Means for verification	Use case demonstrators
Type	M
Implementer(s)	WP2,WP3,WP4
Source	Use cases, programming models

ID	**REQ-TIM-02**
Topic	Timing and Schedulability Analysis
Subtopic	Control loops
Name	Real-time requirements
Description	The AMPERE eco-system must support applications with hard real-time requirements.
Means for verification	Use case demonstrators
Type	M
Implementer(s)	WP2, WP3, WP4
Source	Use cases

4.2 Models of computation

At the same time, AMPERE considers a Model Driven Engineering (MDE) approach, where the non-functional requirements are mapped in higher-level modelling frameworks (AMPERE considers AMALTHEA and CAPELLA) and transformed into the underlying parallel programming model

(OpenMP in AMPERE). This transformation needs to take into consideration the timing properties of applications, and guarantee the fulfilment of the related requirements, under different models of computation [13, 14, 15, 16], approaches which are going to be developed in task 4.3.

ID	REQ-TIM-03
Topic	Timing and Schedulability Analysis
Subtopic	Schedulability Analysis
Name	Graph Analysis
Description	The AMPERE eco-system must support schedulability analysis of application graphs/chains.
Means for verification	Use case demonstrators
Type	M
Implementer(s)	WP3
Source	Use cases, Programming model

ID	REQ-TIM-04
Topic	Timing and Schedulability Analysis
Subtopic	Schedulability Analysis
Name	Publish-subscribe Analysis
Description	The AMPERE eco-system must support schedulability analysis of application components interacting using communication and synchronization primitives with publish-subscribe semantics.
Means for verification	Use case demonstrators
Type	M
Implementer(s)	WP3
Source	Use cases, Programming model

4.3 Runtime impact

From the runtime perspective, AMPERE considers a hierarchical approach [17], where the computing platform can be abstracted to applications and operating systems, through a hypervisor, providing isolation between different applications executing in the same computing node. Therefore, analysis tools need to include the concept of multiple levels of mapping and scheduling, which need to be also considered in the run-time mechanisms.

ID	REQ-TIM-05
Topic	Timing and Schedulability Analysis
Subtopic	Schedulability Analysis
Name	Mixed-criticality scheduling
Description	The AMPERE eco-system should support multi-core schedulability analysis of hierarchical schedulers, with mixed-criticality constraints.
Means for verification	Use case demonstrators
Type	M
Implementer(s)	WP3
Source	Use cases

Moreover, in order to guarantee that there is enough computational capacity available for critical tasks, the AMPERE eco-system needs to support reservation-based approaches [18] and runtime regulation mechanisms (e.g. MemGuard [19]).

ID	REQ-TIM-06
Topic	Timing and Schedulability Analysis
Subtopic	Schedulability Analysis
Name	Analysis of reservation/regulation schedulers
Description	The AMPERE eco-system must support schedulability analysis of reservation-based schedulers and SW-based regulation.
Means for verification	Use case demonstrators
Type	M
Implementer(s)	WP3
Source	Use cases, Runtime

4.4 Execution time

Real-time schedulability tests for multi-core systems assume that the worst-case execution time (WCET) for the computation (code fragments) is given [20]. Therefore, in order to be able to derive the schedulability analysis of the applications, and provide the required guarantees, AMPERE must provide the appropriate mechanisms and analysis for determining execution time.

ID	REQ-TIM-07
Topic	Timing and Schedulability Analysis
Subtopic	Worst-case execution time
Name	Worst-case execution time analysis
Description	The AMPERE eco-system should support worst-case execution time (WCET) analysis.
Means for verification	Inspection
Type	S
Implementer(s)	WP3
Source	Use cases

Most timing analysis methodologies focus on determining the worst-case execution time (WCET) of a program fragment, in order to use this value to determine a safe upper bound on the execution time. Relying on the existence of an accurate model of the timing behaviour of the underlying hardware is challenging, particularly when considering parallel and heterogeneous platforms [21]. Task 4.3 will consider approaches, that integrate both the analysis of the application structure with measurement-based techniques and determine the execution time of those paths by executing the application on the target hardware platform (an approach as in Figure 10) [22], as well as the use of regulation techniques, which reduce interference and therefore analysis pessimism [19].

ID	REQ-TIM-08
Topic	Timing and Schedulability Analysis
Subtopic	Worst-case execution time

Name	Worst-case execution time measurements
Description	The AMPERE eco-system must support execution time profiling with performance counters (see REQ-ENE-01).
Means for verification	Inspection
Type	M
Implementer(s)	WP5
Source	Use cases

4.5 Analysis of HW acceleration

AMPERE will use high-performance parallel platforms with heterogeneous components (CPU, GPU, and FPGA), particularly Off-The-Shelf, which challenge both the extraction of timing information as well as the knowledge of the details of the hardware contention mechanisms. The AMPERE eco-system must therefore consider both static and dynamic allocation of accelerators, with co-scheduling approaches to reduce interference [23], more dynamic and measurement-based approaches to execution time analysis, and offload-based schedulability analysis techniques [24].

ID	**REQ-TIM-09**
Topic	Timing and Schedulability Analysis
Subtopic	Analysis of HW acceleration
Name	Response-time analysis for accelerators
Description	The AMPERE eco-system must support response-time analysis for both dynamically-configured and static hardware accelerators deployed on GPUs and FPGA fabrics.
Means for verification	Test
Type	M
Implementer(s)	WP3
Source	Use cases, Platform

ID	**REQ-TIM-10**
Topic	Timing and Schedulability Analysis
Subtopic	Analysis of HW acceleration
Name	Analysis of multiple offloading schemes
Description	The AMPERE eco-system must support schedulability analysis of software tasks that issue both synchronous and asynchronous hardware acceleration requests.
Means for verification	Test
Type	M
Implementer(s)	WP3
Source	Use cases, Platform

ID	**REQ-TIM-11**
Topic	Timing and Schedulability Analysis
Subtopic	Analysis of HW acceleration
Name	Profiling of accelerators
Description	The AMPERE eco-system should support the profiling of GPU and FPGA hardware accelerators in terms of execution time, number of

	bus/memory transactions, and resource demand, as well as FPGA reconfiguration times
Means for verification	Test
Type	M
Implementer(s)	WP3
Source	Use cases, Platform

The optimization techniques used in AMPERE shall take into account timing constraints for FPGA-based acceleration requests via worst-case timing analysis. The analysis to be used in AMPERE will extend the one presented in [23], which works by studying the worst-case response times of SW-tasks issuing HW acceleration requests.

For the purpose of the analysis, it is crucial to distinguish between synchronous and asynchronous acceleration requests. In the former case, in [23] SW-tasks are modelled as segmented self-suspending tasks, I.e., sequential computations where execution phases are interleaved to suspension phases. The latter correspond to acceleration requests and their duration depend on the time taken complete an acceleration, which includes (i) the reconfiguration time, (ii) contention delays, and (iii) the actual execution time of the accelerator. The contention delays change depending on whether the FPGA reconfiguration interface (FRI) supports pre-emption or not.

Under the scheduling infrastructure described in [23], the following theorem can be proved (as a special case of Theorem 1 in [23]) to ensure predictable worst-case delays when requesting the execution of a HW-task under preemptive FRI management.

Consider an arbitrary acceleration request $\mathcal{R}_a$ related to HW-task τ_a^H issued by a SW-task τ_i. Let $s_k = s(\tau_a^H)$ be the FPGA slot to which τ_a^H is allocated to. Under preemptive FRI management, the maximum delay incurred by $\mathcal{R}_a$ is upper-bounded by

$$\overline{\Delta_a^P} = \sum_{\tau_j \neq \tau_i} \max_{\tau_b^H \in \mathcal{H}(\tau_j)} \{\Delta_b^{slot} + r_b\} \quad (4)$$

where

$$\Delta_b^{slot} = \begin{cases} C_b^H & \text{if } s(\tau_b^H) = s_k \\ 0 & \text{otherwise,} \end{cases} \quad (5)$$

- r_b denotes the worst-case reconfiguration time of HW-task τ_b^H,
- $\mathcal{H}(\tau_j)$ denotes the HW-tasks associated to SW-task τ_j, and
- C_b^H denotes the worst-case execution time of HW-task τ_b^H.

The term Δ_b^{slot} represents the interference experienced by τ_a^H due to slot contention originated by the execution of τ_b^H.

Under non-preemptive FRI management, a bound $\overline{\Delta_a^P}$ on the delay experienced by a HW-task is provided by Theorem 2 in [23]. Consider an arbitrary acceleration request $\mathcal{R}_a$ for HW-task τ_a^H issued by a SW-task τ_i. Let $s_k = s(\tau_a^H)$ be the slot to which τ_a^H is allocated to. Under non-preemptive FRI management, the maximum delay incurred by $\mathcal{R}_a$ is upper-bounded by

$$\overline{\Delta_a^{NP}} = \overline{\Delta_a^{P}} + NH_k^{max} \times r_k^{max} \quad (6)$$

where

$$NH_k^{max} = |\{\tau_b^H \in \Gamma^H : s(\tau_b^H) = s_k\}| \quad (7)$$

With Γ^H being the set of all HW-tasks, and

$$r_k^{max} = \max_{\tau_b^H \in \Gamma^H} \{r_b : s(\tau_b^H) \neq s_k\}. \quad (8)$$

The term NH_k^{max} represents the number of HW-tasks allocated to slot s_k. The scheduling infrastructure from [23] ensures that each of them can be directly blocked due to non-preemptable FPGA reconfiguration by at most r_k^{max} units of time, representing the largest reconfiguration time of the HW-tasks allocated on the other slots.

In the context of the AMPERE project we plan to extend this analysis framework to support both *parallel SW-tasks* and *asynchronous acceleration*.

To address parallel SW-tasks it is essential to avoid a direct transformation to segmented self-suspending tasks as, without extensions or third-party algorithm, they are capable of modeling sequential computations only. The delay analysis shall also be extended to cope with additional contention in serving acceleration requests caused by parallelism.

A new modeling and analysis approach is instead needed to cope with asynchronous acceleration. Indeed, when an asynchronous acceleration request is issued, the processor can serve computations and then later wait for the completion of the accelerated task. To address these scenarios, we are working on a new analysis framework based on a new task model focused on *event-dependent suspensions*. An example instance of the new model is sketched in the following figure.

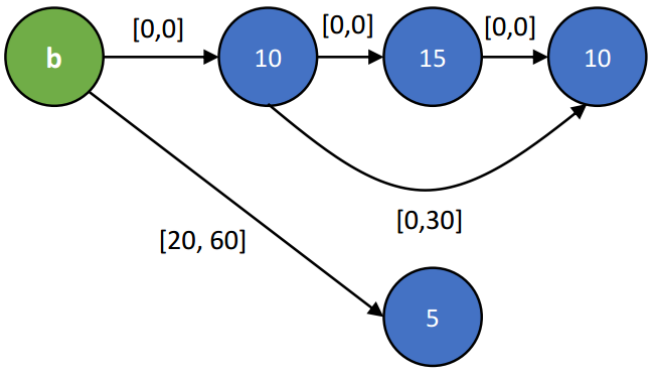

Figure 11. Example of event-dependent suspensions.

Here, SW-tasks are modelled via a directed acyclic graph whose nodes denote sequential computations to be executed on the same processor and edges denoted precedence constraints that are satisfied with a *variable but bounded delay*. The first node labelled with "**b**" is a special node that denotes the task release event. The variable delays that satisfy the precedence constraints can be used to model pending HW acceleration requests, while nodes that are not on the same path of the graph can be used to express computations on the processor that occur while a HW acceleration is pending.

4.6 Soft real-time and trade-offs

Classical hard real-time techniques aim at guaranteeing that no timing requirement can be violated at any time, usually because these techniques are employed in safety-critical control systems, like in automotive or aerospace. However, this requires expensive techniques for determining worst-case execution times and worst-case interference scenarios that may be very unlikely to happen. Yet, accounting for these worst-case conditions in the system analysis, design and run-time configuration, may lead to a significant amount of pessimism that may result into a relatively low saturation of the underlying physical resources, coupled with a relatively high energy consumption.

Soft real-time techniques [25], on the other hand, deal with systems that can tolerate infrequent timing requirement violations, as the adopted computation and control logics are designed to compensate, for example, the occurrence of deadline misses. The typical example is the one of real-time multimedia, virtual and augmented reality and anytime computing, in which deadline misses can be compensated for example with frame skipping or lowering the quality of the computed outputs, or proper trade-offs can be sought between quality of the computed output vs resource requirements. This enables trading predictability for efficiency with the adoption of design and profiling techniques that aim at identifying resource requirements and interference scenarios that are likely to occur up to a sufficiently high probability, avoiding the expensive computation of precise worst-case bounds (i.e., focusing on experimental maximum computing times, or high percentiles of the expected computing times distributions, as opposed to computing WCET bounds), and employing at run-time adaptive techniques that allow for dealing with those infrequent times when timing requirements are violated. For example, with reservation-based scheduling, a number of techniques exist [26, 27] for on-line adaptation of the allocated reserved resources (e.g., feedback-based control of the reserved budget), leading to a controlled maximum percentage of deadline misses, beyond which the system may exhibit a too unstable behaviour.

As high-performance computing platforms are more and more widespread in embedded real-time applications, it becomes increasingly important to support the coexistence on the same platform of hard real-time components, typically employing more easily analysable algorithms deployed in simpler software architectures, alongside with soft real-time components that may be characterized by more sophisticated computations using complex software stacks. Furthermore, the employment of soft real-time techniques allows for designing off-line analysis and on-line adaptive frameworks that enable trading timeliness of real-time applications for efficiency in resource usage and energy consumption within the platform.

ID	**REQ-TIM-12**
Topic	Timing and Schedulability Analysis
Subtopic	Soft Real-Time
Name	Coexistence of soft and hard real-time components
Description	The AMPERE eco-system must support the coexistence of hard and soft real-time components on the same platform, in a way that ensures the respect of the hard and soft timing requirements in place.
Means for verification	Test
Type	M
Implementer(s)	WP5
Source	Use cases, Platform

Thanks to the isolation guaranteed by REQ-TIM-12 just introduced, it will be possible to create spatial/temporal partitions of the system resources so that in AMPERE it will be possible to develop analysis techniques in WP3 that independently analyse the hard real-time and the soft real-time domains. Note that this requirement is related to REQ-TIM-05, but from the perspective of hard vs. soft real-time, and not application criticality.

ID	**REQ-TIM-13**
Topic	Timing and Schedulability Analysis
Subtopic	Soft Real-Time
Name	Probabilistic analysis of soft real-time applications
Description	The AMPERE eco-system should support probabilistic analysis of soft real-time applications, given a probabilistic characterization of the processing times and/or expectable interference terms.
Means for verification	Test
Type	S
Implementer(s)	WP3
Source	Use cases, Platform

As mentioned, in soft real-time systems it may be important to support on-line refinement and adaptation of the system parameters, as sensed at run-time while applications are running, and employing a feedback-based control logic that reviews dynamically the system configuration as needed. This can include prediction and estimation logic that refines knowledge on the resource requirements of the actively running application components, as well as control logic specific to each soft real-time application component to drive its associated future resource allocation, as well as a control logic to drive and adapt the energy tunable in the system according to given system-level goals. Such an approach can be built by extending techniques like [28], among others.

For this to be possible, the underlying platform needs to support proper sensors and actuation knobs (as noted in REQ-TIM-08).

ID	**REQ-TIM-14**
Topic	Timing and Schedulability Analysis
Subtopic	Soft Real-Time
Name	Per-entity resource consumption monitoring
Description	The AMPERE eco-system should support on-line monitoring of the resources consumption of individual real-time components
Means for verification	Test
Type	S
Implementer(s)	WP5
Source	Use cases, Platform

For particularly dynamic workloads, the resource consumption levels for real-time application components in the future may non-necessarily reflect exactly what has been recently measured, or anyway it may be necessary to develop specific techniques to better foresee/predict the future evolution of the workload, i.e., considering linear regression or percentile estimation techniques or relatively simple techniques with sufficiently low associated overheads [26].

ID	REQ-TIM-15
Topic	Timing and Schedulability Analysis
Subtopic	Soft Real-Time
Name	Per-entity resource consumption estimation and prediction
Description	The AMPERE eco-system could support on-line estimation and prediction of the expected resources consumption of individual real-time components in the future, exploiting information coming from the on-line monitoring of REQ-TIM-14.
Means for verification	Test
Type	C
Implementer(s)	WP5
Source	Use cases, Platform

ID	REQ-TIM-16
Topic	Timing and Schedulability Analysis
Subtopic	Soft Real-Time
Name	Adaptability in resource allocation
Description	The AMPERE eco-system should support the possibility to adapt dynamically the resources allocated to the soft real-time tasks
Means for verification	Test
Type	S
Implementer(s)	WP5
Source	Use cases, Platform

ID	REQ-TIM-17
Topic	Timing and Schedulability Analysis
Subtopic	Soft Real-Time
Name	Soft real-time controllers
Description	The AMPERE eco-system could include controllers employing logic to control dynamically the resource allocation depending on the monitored and/or foreseen resource consumption (REQ-TIM-14 and REQ-TIM-15), so to meet precise timeliness requirements for the application.
Means for verification	Test
Type	C
Implementer(s)	WP5
Source	Use cases, Platform

5 Software and Hardware Resilient Methods

This section summarizes the requirements related to resilience and fault-tolerance, collected from the work in WP1 (Task T1.1 "System model requirement specification and use case definition" [2]) as well as from WP2 (Task 2.1 "Model transformation requirements specification" [3]), WP4 (Task 4.1 "Run-time requirement specification" [4]) and the platforms from WP5 (Task 5.1 "Platform Selection" [5]).

Fault-tolerance and resiliency are aspects of the dependability of systems targeted at software and at hardware level. Furthermore, the different components of the software stack (e.g., parallel programming model, runtime, operating system, etc.) can include techniques to enhance these aspects of the system.

The remainder of the section introduces first the particularities of Cyber-Physical Systems (CPS) regarding resiliency; then, an overview of the most common hardware and software techniques towards fault tolerance is introduced; after that, fault tolerance at programming model level is shown for task-based models in general, and OpenMP in particular; finally, the approach for fault tolerance in AMPERE is explained, including particular requirements of the project and considered techniques.

5.1 Resiliency in CPS

CPSs in general, and the safety-critical components of CPSs in particular, need to keep delivering their functionality in the presence of run-time faults. The shrinking size of the components of the system coupled with the external noise and radiation (e.g., power supplies variations, lightning or alpha particles hitting the transistors of the processor) increases the vulnerability of the system to *transient faults* caused by a transistor's state flipping. The system is also vulnerable to *permanent faults* (e.g., short circuit) and *intermittent faults* (e.g., loose electrical connection).

AMPERE addresses two type of CPSs, as defined in the use cases targeted in the project: automotive and railway systems. In both scenarios, the heavy use of wireless communications opens the door to cyber-attacks, and thus jeopardizes the security of the system. As a consequence, the robustness and the recovery capabilities of the system are a paramount aspect of these systems.

MDE is a common approach for the development of CPSs. This design flow may prevent several faults by allowing verification and validation processes at a model level. Additionally, MDE allows the use of code synthesis tools to generate the final code to be deployed on the target platform. Unfortunately, faults cannot be completely prevented, and particularly safety and reliability are identified as non-functional requirements difficult to fulfil even in MDE approaches [29].

Several works target the enhancement of the expressiveness of modelling languages regarding dependability, including fault tolerance concepts. Relevant examples are: the work extending Simulink in order to support the specification of common fault-tolerance design patters, like *sparing*, *comparing* and *voting*, so the extended models can tolerate hardware faults [30]; the work extending the Architecture Analysis and Design Language (AADL) to assist dependability analysis at the architecture level 11; and the work extending UML/Marte with a dependability profile covering fault tolerance concepts [31]. However, these proposals require considerable amounts of

manual effort to extend the mentioned models. Furthermore, they consider traditional DMSL defining concurrency, but not including parallel programming.

5.2 Fault tolerance techniques: overview

Fault tolerance is mostly achieved by introducing *redundancy* in software or hardware [32]. There are several forms of redundancy. This section introduces the most relevant ones for achieving fault tolerant systems, organized as hardware and software techniques.

5.2.1 Hardware techniques

Hardware fault tolerance techniques can be divided into three groups:

- *Fault-masking*. This passive, or static, technique consists on hiding failures so the system can achieve fault tolerance without requiring any action. It is based on replicating resources and computation, and then using voting mechanisms to decide the correct result.
- *Reconfiguration*. This active, or dynamic, technique consists of four steps: (1) detection, i.e., recognize a fault has occurred; (2) location, i.e., determine where a fault has occurred; (3) containment, i.e., isolate a fault and prevent te effects of that fault from propagating through the system; and (4) recovery, i.e., regain operational status via reconfiguration (modifying the use of components of the system).
- Hybrid techniques. These techniques, combining static and dynamic approaches, entail a high cost, but also provide better evidence of fault-tolerance. An example is *self-purging redundancy*, where all units participate actively in the system and also have the capability to remove themselves from the system in the occurrence of faults.

5.2.2 Software techniques

The most common software techniques for fault-tolerant scheduling are the following:

- *Checkpointing*. This is a *backward error recovery* technique that consists on periodically saving the state of the system in a checkpoint; then, when a fault occurs, the system state is replaced by the last checkpoint and the execution continues.
- *Replication* consists on copying parts of an application; it can be *active replication* (a.k.a. *spatial* or *structural replication*), when multiple copies of the replicated part are executed in parallel, or *passive replication* (a.k.a. *temporal replication*), when a backup copy is run only if the original fails, so they never run in parallel. While spatial replication typically leads to increased *makespan*, i.e., the total length of the schedule, until the last processing unit has finished, temporal replication, on the other hand, adds overhead that might be unnecessary if no faults occur, as well as increase the *energy consumption* of the computation.

5.3 Fault tolerance in AMPERE

This section lists the requirements of the AMPERE project regarding fault tolerance, and describes the techniques envisioned in the project for addressing this non-functional requirement across the whole software ecosystem.

5.3.1 AMPERE requirements regarding fault tolerance

The requirements introduced by WP3 regarding fault tolerance are summarized as follows.

ID	REQ-FAU-01
Topic	Fault tolerance
Subtopic	Support at HW level
Name	Access to hardware failures
Description	The AMPERE eco-system must include fault tolerant architectures. It must also support techniques for fault detection in all the hardware components.
Means for verification	Inspection
Type	M
Implementer(s)	WP3,WP5
Source	Platform

ID	REQ-FAU-02
Topic	Fault tolerance
Subtopic	Support at programming model level
Name	Static analysis based on the taskgraph
Description	The AMPERE eco-system must provide the static analysis techniques for analysing the parallel execution from a programming model perspective to decide the best places to automatically introduce fault tolerance techniques like task redundancy and checkpointing.
Means for verification	Test
Type	M
Implementer(s)	WP3, WP2
Source	Programming model

ID	REQ-FAU-03
Topic	Fault tolerance
Subtopic	Support at runtime level
Name	Dynamic reconfiguration
Description	The AMPERE runtime system, including the parallel runtime, the OS and the hypervisor, should include mechanisms for automatic reconfiguration in the occurrence of a failure in some hardware component.
Means for verification	Test
Type	M
Implementer(s)	WP3, WP4
Source	Runtime

5.3.2 Fault tolerance in task-based parallel programming models

The parallelism exposed in parallel applications can often be decomposed into a set of tasks with a series of input and output constraints. This structure can be modelled as a taskgraph, where nodes are tasks, and edges are dependencies (or ordering constraints) between tasks.

Replication can be naïvely performed on a taskgraph by duplicating all tasks in the graph [33]. Then, scheduling mechanisms can be applied to (1) ensure only one instance, the original or the duplicated, and (2) decide the better scheduling of the taskgraph with duplicated tasks in order to minimize idle resources, and to eliminate overhead in fault-free. The decision of duplicating all tasks may have however a severe impact in the energy consumption of the application, and again scheduling mechanisms are needed for frequency scaling [34].

Some features of the taskgraph, are however interesting when considering software techniques for fault tolerance. The most relevant are the following:

- The depth of tasks, or *task height*. The height of a task is recursively computed as the maximum height of all its immediate successors +1. This information has been exploited to create partitions of the taskgraph (which, by definition, contains tasks independent among them) and schedule the tasks contained in each partition together with their duplicates in order to achieve fault tolerance in multiprocessor systems [35].
- The cost of task communications, or *communication weight*. This aspect describes the amount of data flowing through an edge of the graph, i.e., the amount of data produced by the predecessor task and consumed by the successor task. This information is further valuable for refining timing analysis and reducing the energy consumption [36].
- The cost of tasks, or *task weight*. The processing time of tasks on a processor may differ from those of the same tasks on another processor. The cost of tasks, together with the cost of the communications, has been previously used in proposed scheduling algorithms to prioritize tasks in fault-tolerant systems [37].
- The *synchronization points*. Synchronizations such as barriers and locks pose two important issues when considering fault-tolerant system: causality related dependencies and resource contention. Previous works analyse these issues in order to allow checkpointing tasks at any time, even when holding or waiting for locks and barriers [38].

OpenMP [39] is the parallel programming model considered in the AMPERE project for exploiting performance in parallel architecture by virtue of its many benefits: productivity, portability and heterogeneous support, among others. OpenMP allows describing the behaviour of a program as a taskgraph by means of the tasking model. In OpenMP, this representation is called *task dependency graph (TDG)*. Figure 12 shows a snippet (left) of an OpenMP application, adapted from the DAPHNE benchmark suite [40] to use OpenMP tasks instead of OpenMP worksharings (e.g., `for` loops), and the corresponding TDG (right).

```
… extractEuclideanClusters(…)
  #pragma omp parallel taskloop
  for (int i = 0; i < cloud_size; ++i)
  { … }
  #pragma omp parallel taskloop
  for (int i = 0; i < cloud_size; ++i)
  { … }
  omp_target_alloc(…);
  omp_target_memcpy(…);
  for (int i = 0; i < cloud_size; ++i) {
    #pragma omp target map(to: …) map(from: …)
    {
      #pragma omp teams distribute parallel for
      for (int i = 0; i < cloud_size; ++i) {
      { … }
    }
  }
  omp_target_free(…);
}


#pragma omp task
… sort ()
{…}

#pragma omp task
… color()
{…}
```

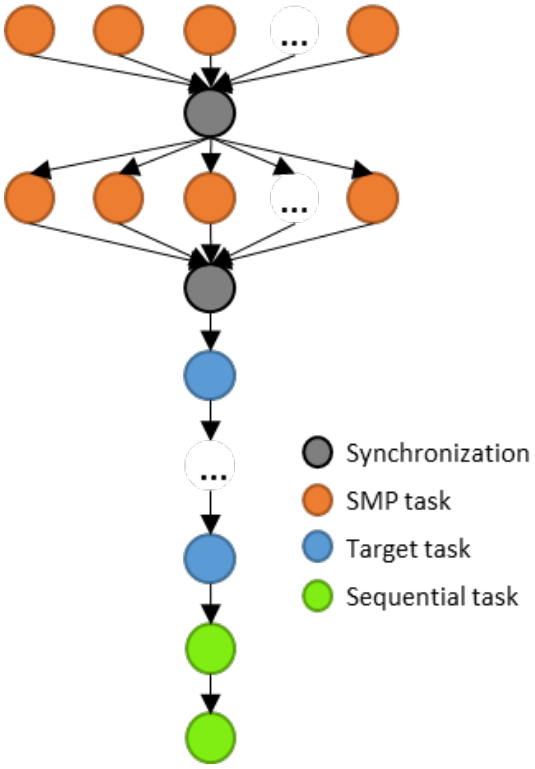

Figure 12. OpenMP Euclidean Cluster benchmark from the DAPHNE suite: code snippet (left) and TDG (right).

Although OpenMP focuses on exploiting performance on HPC systems, there are several aspects of the specification and some proposed runtime extensions that provide OpenMP frameworks with capabilities for fulfilling non-functional requirements such as fault tolerance:

- Features towards fault tolerance in the OpenMP specification.

  OpenMP provides cancellation constructs as a step for addressing fault tolerance. These are the `cancel` construct, that activates the cancellation of the innermost enclosing region, considering `parallel` and `taskgroup` regions, among others; and the cancellation point construct, that introduces a user-defined point at which the cancellation of the innermost enclosing region can be checked.

  Cancellations are however insufficient for providing a fault-tolerant system in the presence of critical tasks. There is already an extensive proposal towards fault tolerance based on *user-defined error handling*, i.e., mechanisms offer at a user-level for specifying a particular action when a failure occurs during the execution of a given computational unit [41]. The error model proposed for OpenMP includes three different features: (1) *constructs*, like the already supported `cancel` construct, to stop a given region; (2) *return codes*, suitable for exception-unaware languages; and (3) *callbacks*, suitable for exception-aware languages.

  Another interesting mechanism to support fault tolerance is based on the concept of *alternative task* [42]. This is a form of spatial replication that uses different implementations for replicating a task. OpenMP defines the `metadirective` and the `declare variant` directives. The former is a directive that can specify multiple directive variants that can be conditionally selected. The latter declares a specialized variant of a

function and specifies the context in which it is used. OmpSs has a feature similar to the `declare variant` directive, the implements clause that, attached to the `target` construct can be used to specify that the annotated task is an implementation of another task [43].

- Fault tolerant OpenMP runtime frameworks

  Application-Level Checkpointing (ALC) is a mechanism based on saving the application-level state (i.e., heap, global and local variables, and call stack). It is an alternative to the commonly used System-Level Checkpointing (SLC), based essentially in core-dump-style snapshots of the computational state of the machine, which is very machine and OS-dependant. There is a proposal based on ALC that aims at providing self-checkpointing. It requires user intervention to decide the suitable places for checkpointing [44]. This proposal has been further enhanced with compiler analysis and optimizations to reduce the amount of data checkpointed [45, 46].

  With the objective of reducing the overheads introduced by checkpointing techniques, and so increase the scalability of the system, redundant threads for parallel regions have been proposed [47]. This work uses spatial redundancy; more specifically it creates three replicas of each fault tolerant section, and run them in three different threads. Then, a comparison and vote step decides the final result. The redundancy is applied to all parallel regions of the OpenMP program, so the structure of the taskgraph is not exploited.

  Finally, there are also proposals for task level redundancy [48,49]. These works are based on the concept of *reliable task*, which defines a unit of computation that has the ability of detecting and recovering from a fault. For each reliable task in the system, three redundant tasks are created. At the end of their execution, a compare and vote step synchronizes the results providing the correct one. This method however does not ensure correctness if all replicated tasks mismatch their outputs. Results show a 95% success ratio and a maximum performance degradation of 1,8x.

In the AMPERE ecosystem, the domain-specific modelling language, the synthesis tools and compilers, and the multi-criteria optimization tools will communicate by means of meta models mainly representing a TDG. For this reason, the fault tolerance techniques that the AMPERE project will consider are based on the information that can be represented in the TDG, as well as on how this information can be exploited at a parallel programming model level, and respected across the whole runtime system. Overall, the research lines include:

- Use the structure of the TDG to determine the best places to apply fault tolerant mechanisms. As endorsed by previous research at a dynamic level, these points are:
  - Barriers and other synchronization constructs (e.g., taskwaits, taskgroups, etc.) are suitable points to perform automatic checkpointing because the amount of on-the-fly data can be negligible compared to those parts of the TDG exposing more parallelism.
  - The cost of tasks is also an important aspect to consider in an automatic task replication mechanism. In this regard, tasks with higher costs can be spatially replicated or provide alternatives, while tasks with lower costs can be temporally replicated.
  - The cost of communications is an important aspect to take into account for the scheduling of replicated tasks. For example, in the occurrence of a transient fault, a

replicated task could be scheduled in the same processor it failed in order to avoid the cost of moving huge amounts of data.

- Mix user-defined exception handling and alternative task techniques [50]. The proposal to extend OpenMP with *callbacks* can be combined with alternative implementations of a task, via the `declare variant` directive, in order to specify a workflow to be launched to deal with a failure in a specific task. This workflow can include CPU, GPU and FPGA versions of the same task.

6 Conclusion

This document presented the requirements related to the multi-criteria optimization associated to the non-functional constraints considered in AMPERE (energy-efficiency, time-criticality and fault tolerance), consolidated from the analysis of the project use cases. The description of the requirements includes the concrete criteria and metrics, as well as the means of verification, for the evaluation of the project results.

The document also identifies the initial techniques that will be considered in the scope of the development of the project, related to the considered energy models, the timing and schedulability analyses and the software and hardware resilient methods. These techniques will provide the required guarantees to the non-functional requirements of the targeted systems, whilst targeting high-performance parallel execution.

Acronyms and Abbreviations

- ALU – Arithmetic Logic Unit
- CPS – Cyber-Physical Systems
- CPU – Central Processing Unit
- DVFS – Dynamic Voltage and Frequency Scaling
- FPGA – Field Programmable Gate Array
- GPU – Graphics Processing Unit
- KPI – Key Performance Indicator
- MDE – Model Driven Engineering
- MS – Milestones
- PE – Processing Element
- PMU - Performance Monitoring Unit
- RISC-V - open instruction set architecture based on Reduced Instruction Set Computer (RISC)
- SIMD – Single Instruction Multiple Data
- WCET – Worst-Case Execution Time
- WP – Work Package

References

[1] Grant Agreement number: 871669 — AMPERE — H2020-ICT-2018-20/H2020-ICT-2019-2, October 2019

[2] AMPERE Consortium, Deliverable D1.1, “System models requirement and use case selection”, September 2020

[3] AMPERE Consortium, AMPERE Deliverable D2.1, “Model transformation requirements”, July 2020

[4] AMPERE Consortium, AMPERE Deliverable D4.1, “Run-time architecture”, July 2020

[5] AMPERE Consortium, AMPERE Deliverable D5.1, “Reference parallel heterogeneous hardware selection”, July 2020

[6] F. Pittino, F. Beneventi, A. Bartolini and L. Benini, "A Scalable Framework for Online Power Modelling of High-Performance Computing Nodes in Production," 2018 International Conference on High Performance Computing & Simulation (HPCS), Orleans, 2018, pp. 300-307, doi: 10.1109/HPCS.2018.00058.

[7] M. Witkowski, A. Oleksiak, T. Piontek, and J. Wglarz, “Practical power consumption estimation for real life hpc applications,” Future Generation Computer Systems, vol. 29, no. 1, pp. 208 – 217, 2013, including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures

[8] Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J. W., Lee, S. H., & Skadron, K. (2009, October). Rodinia: A benchmark suite for heterogeneous computing. In 2009 IEEE international symposium on workload characterization (IISWC) (pp. 44-54). Ieee.

[9] Balsini, A., Pannocchi, L., & Cucinotta, T. (2019). Modeling and simulation of power consumption and execution times for real-time tasks on embedded heterogeneous architectures. ACM SIGBED Review, 16(3), 51-56.

[10] Maurya, A.K., Modi, K., Kumar, V. et al. Energy-aware scheduling using slack reclamation for cluster systems. Cluster Comput 23, 911–923 (2020). https://doi.org/10.1007/s10586-019-02965-7

[11] Ramesh, P., & Ramachandraiah, U. (2018). Energy aware proportionate slack management scheduling for multiprocessor systems. Procedia computer science, 133, 855-863.

[12] Forsberg, B., Lampka, K., & Spiliopoulos, V. (2016, October). An online overclocking scheme for bursty real-time tasks and an evaluation of its thermal impact. In Proceedings of the 14th ACM/IEEE Symposium on Embedded Systems for Real-Time Multimedia (pp. 104-113).

[13] M. A. Serrano, A. Melani, M. Bertogna and E. Quinones, "Response-time analysis of DAG tasks under fixed priority scheduling with limited preemptions," 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, 2016, pp. 1066-1071.

[14] P. Burgio, M. Bertogna, A. Melani, E. Quinones, M. A Serrano, “Mapping, Scheduling, and Schedulability Analysis”. In Pinho, L, Quiñones, E, Bertogna, M, Marongiu, A, Nélis, V, Gai, P, Sancho, J, (Eds) "High-Performance and Time-Predictable Embedded Computing", Jul, 2018, DOI: 10.13052/rp-9788793609624.

[15] R. Ernst, S. Kuntz, S. Quinton, M. Simons. “The Logical Execution Time Paradigm: New Perspectives for Multicore Systems (Dagstuhl Seminar 18092)”. Dagstuhl Reports, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2018, 8, pp.122 - 149. ff10.4230/DagRep.8.2.122ff. ffhal01956964f

[16] Cui, J., Tian, C., Zhang, N. et al. Verifying schedulability of tasks in ROS-based systems. J Comb Optim 37, 901–920 (2019). https://doi.org/10.1007/s10878-018-0328-0

[17] G. Lipari and E. Bini, "A Framework for Hierarchical Scheduling on Multiprocessors: From Application Requirements to Run-Time Allocation," 2010 31st IEEE Real-Time Systems Symposium, San Diego, CA, 2010, pp. 249-258, doi: 10.1109/RTSS.2010.12.

[18] G. Lipari and E. Bini, "Resource partitioning among realtime Applications," in Proc. of Euromicro Conf. on Real-Time Systems (ECRTS'03), July 2003.

[19] H. Yun, G. Yao, R. Pellizzoni, M. Caccamo and L. Sha, "MemGuard: Memory bandwidth reservation system for efficient performance isolation in multi-core platforms," 2013 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS), Philadelphia, PA, 2013, pp. 55-64, doi: 10.1109/RTAS.2013.6531079.

[20] Robert I. Davis and Alan Burns. 2011. A survey of hard real-time scheduling for multiprocessor systems. ACM Comput. Surv. 43, 4, Article 35 (October 2011), 44 pages. DOI:https://doi.org/10.1145/1978802.1978814

[21] V. Nelis, P. M. Yomsi, and L. M. Pinho, “Methodologies for the WCET Analysis of Parallel Applications on Many-core Architectures,” in The Euromicro Conference on Digital System Design (DSD 2015), 2015.

[22] Nélis, V, Yomsi, P, Pinho, L, "Timing Analysis Methodology". In Pinho, L, Quiñones, E, Bertogna, M, Marongiu, A, Nélis, V, Gai, P, Sancho, J, (Eds) "High-Performance and Time-Predictable Embedded Computing", Jul, 2018, DOI: 10.13052/rp-9788793609624.

[23] A. Biondi, A. Balsini, M. Pagani, E. Rossi, M. Marinoni and G. Buttazzo, "A Framework for Supporting Real-Time Applications on Dynamic Reconfigurable FPGAs," 2016 IEEE Real-Time Systems Symposium (RTSS), Porto, 2016, pp. 1-12, doi: 10.1109/RTSS.2016.010.

[24] M. A. Serrano and E. Quiñones, "Response-Time Analysis of DAG Tasks Supporting Heterogeneous Computing," 2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC), San Francisco, CA, 2018, pp. 1-6, doi: 10.1109/DAC.2018.8465575.

[25] G. Buttazzo, G. Lipari, L. Abeni and M. Caccamo. “Soft Real-Time Systems – Predictability vs. Efficiency,” Springer, Boston, MA, 2005.

[26] L. Abeni, T. Cucinotta, G. Lipari, L. Marzario, L. Palopoli, "QoS Management through adaptive reservations," Real-Time Systems Journal, Vol. 29, Issue 2-3, March 2005, ISSN:0922-6443, Kluwer Academic.

[27] T. Cucinotta, F. Checconi, L. Abeni, L. Palopoli, "Self-tuning Schedulers for Legacy Real-Time Applications," in Proceedings of the 5th ACM European Conference on Computer Systems (EuroSys 2010), Paris, France, April 2010.

[28] T. Cucinotta, L. Palopoli, L. Abeni, D. Faggioli, G. Lipari, "On the integration of application level and resource level QoS control for real-time applications," IEEE Transactions on Industrial Informatics, Vol. 6, No. 4, November 2010.

[29] Ameller, D., Franch, X., Gómez, C., Martínez-Fernández, S., Araujo, J., Biffl, S., . . . Méndez, D. a. (2019). Dealing with non-functional requirements in model-driven development. Transactions on Software Engineering. IEEE.

[30] Ding, K., Morozov, A., & Janschek, K. (2018). More: Model-based redundancy for Simuklink. International Conference on Computer Safety, Reliability, and Security (pp. 250--264). Springer.

[31] Bernardi, S., Merseguer, J., & Petriu, D. C. (2011). A dependability profile within MARTE. Software & Systems Modeling, 313--336.

[32] Hu, T., Bertolotti, I. C., Navet, N., & Havet, L. (2020). Automated fault tolerance augmentation in model-driven engineering for CPS. Computer Standards \& Interfaces, 103424.

[33] Fechner, B., Honig, U., Keller, J., & Schiffmann, W. (2008). Fault-tolerant static scheduling for grids. International Symposium on Parallel and Distributed Processing (pp. 1--6). IEEE.

[34] Eitschberger, P., & Keller, J. (2013). Energy-efficient and fault-tolerant taskgraph scheduling for manycores and grids. European Conference on Parallel Processing (pp. 769--778). Springer.

[35] Hashimoto, K., Tsuchiya, T., & Kikuno, T. (2002). Effective scheduling of duplicated tasks for fault tolerance in multiprocessor systems. Transactions on Information and Systems, 525--534.

[36] Cichowski, P., & Keller, J. (2013). Efficient and fault-tolerant static scheduling for grids. International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (pp. 1439--1448). IEEE.

[37] Chen, C.-Y. (2015). Task scheduling for maximizing performance and reliability considering fault recovery in heterogeneous distributed systems. Transactions on Parallel and Distributed Systems, 521--532.

[38] Badrinath, R., & Morin, C. (2004). Locks and barriers in checkpointing and recovery. International Symposium on Cluster Computing and the Grid (pp. 459--466). IEEE.

[39] OpenMP ARB. (2018, November). OpenMPApplication Programming Interface. Retrieved from https://www.openmp.org/wp-content/uploads/OpenMP-API-Specification-5.0.pdf

[40] Sommer, L., Stock, F., Solis-Vasquez, L., & Koch, A. (2019). DAPHNE-An automotive benchmark suite for parallel programming models on embedded heterogeneous platforms: work-in-progress., (pp. 1--2).

[41] Wong, M., Klemm, M., Duran, A., Mattson, T., Haab, G., de Supinski, B. R., & Churbanov, A. (2010). Towards an error model for OpenMP. International Workshop on OpenMP (pp. 70--82). Springer.

[42] Hwang, S., & Kesselman, C. (2003). Grid workflow: a flexible failure handling framework for the grid. High Performance Distributed Computing (pp. 126--137). IEEE.

[43] Planas, J., Badia, R. M., Ayguade, E., & Labarta, J. (2013). Self-adaptive OmpSs tasks in heterogeneous environments. International Symposium on Parallel and Distributed Processing (pp. 138--149). IEEE.

[44] Bronevetsky, G., Marques, D., Pingali, K., Szwed, P., & Schulz, M. (2004). Application-level checkpointing for shared memory programs. ACM SIGPLAN Notices, 235--247.

[45] Bronevetsky, G. a. (2006). Experimental evaluation of application-level checkpointing for OpenMP programs. International Conference on Supercomputing, (pp. 2--13).

[46] Bronevetsky, G., Marques, D., Pingali, K., McKee, S., & Rugina, R. (2009). Compiler-enhanced incremental checkpointing for openmp applications. International Symposium on Parallel & Distributed Processing (pp. 1--12). IEEE.

[47] Fu, H., & Ding, Y. (2010). Using redundant threads for fault tolerance of OpenMP programs. International Conference on Information Science and Applications (pp. 1--8). IEEE.

[48] Tahan, O., & Shawky, M. (2012). Using dynamic task level redundancy for openmp fault tolerance. International Conference on Architecture of Computing Systems (pp. 25--36). Springer.

[49] Shawky, M., & Oussama, T. (2012). Using dynamic task level redundancy for openmp fault tolerance. International Conference on Architecture of Computing Systems (pp. 25--36). Springer.

[50] Hwang, S., & Kesselman, C. (2003). Grid Workflow:A Flexible Failure Handling Framework for the Grid. High Performance Distributed Computing (pp. 126--137). IEEE.