Integrating Multi-/Many-Cores in Avionics: Open Issues and Future Concepts

Anika Christmann, Adam Kostrzewa, Rolf Ernst
Institute of Computer and Network Engineering
Technische Universität Braunschweig
surname@ida.ing.tu-bs.de

Marius Rockschies, Martin Halle, Frank Thielecke
Institute of Aircraft Systems Engineering
Hamburg University of Technology
forename.surname@tuhh.de

Alexander Peuker, Alexander Kuzolap,
Meiko Steen, Peter Hecker
Institute of Flight Guidance
Technische Universität Braunschweig
forename_initial_letter.surname@tu-braunschweig.de

Kai-Frederik Nessitt, Selma Saidi
Chair of Embedded Systems
TU Dortmund University
forename.surname@tu-dortmund.de

***Abstract*—The demand on on-board computing power in the aerospace domain is increasing while multi-/many-core systems promise an overall performance enhancement. One of the main drivers is the accumulating set of functions in an avionic system, for example in the cockpit, flight navigation/guidance domain or image data processing. The upcoming obsolescence of single-core processors being replaced by multi-/many-core processors is addressed in recent research activities of the avionics community. Despite intensive research, there is still a lack of system design guidelines and experience how to transition existing applications from single- to multi-/many-core. This paper gives an overview on state-of-the-art methods, potential applications, and future concepts how to assess new avionics platform designs in a systematic way.**

***Index Terms*—multi-/many-core, safety-critical, avionic systems**

I. Introduction and Problem Statement

New visual options for the pilot and new cockpit philosophies will trend towards the single-pilot cockpit which requires higher computing power to compensate for the first officer. For flight navigation, new 3D databases and improved sensor fusion algorithms are being developed. Future and novel systems also require processing of image data for approach and auto landing scenarios. Multi-/many-core processors can deliver high processing throughput adequately and have been proven to be cost-effective. Since the aviation industry in its current form relies on single-core systems, computing resources and cost-efficiency are not optimal. An ideal solution would be to use commercial off-the-shelf (COTS) components.

In avionics, the use of multi-/many-core processors is difficult due to the requirements of certification and safety standards, e.g. DO-178C, DO-254, ARINC653. Simultaneous execution of different applications results in a wide range of interference that can occur due to accesses to shared resources. Since multi-/many-core processors promise high performance, there have been many research and development projects addressing this challenge in the last decade, e.g., parMERASA, PROXIMA, EMC2, PHYLOG. The *first part* of this paper gives an overview of these and briefly summarizes the main results in terms of methods, tools, and mechanisms to assure functional safety in multi-/many-core architectures.

The *second part* consists of the systems perspective on the required computing power, especially for mixed criticality systems. Multi-/many-core processors would open new possibilities for system applications to run computationally intensive algorithms. For example, Receiver Autonomous Integrity Monitoring algorithms that provide integrity throughout the navigation system would benefit significantly from the parallelization of their operations. In addition, increased performance in arithmetic calculations such as matrix multiplication, which are common in filter structures, are expected. Applications can also be found in less critical environments, e.g., when different iterations of cost functions are executed simultaneously, which are used in multi-criteria optimizations. Independent of the number of cores, the same certification rules apply. A solution must be found to ensure the same probabilities for Design-Assurance-Levels (DAL) as well as segregation and redundancy requirements which may limit the utilization of multiple cores.

In the *third part*, future concepts will be proposed to provide methods for designing a multi-/many-core avionics platform for civil aircraft. Presented is a two step-approach: 1.) Bottom-up with detailed analysis on processor level to achieve guarantees for the applications. This paper proposes a new method to ensure functional safety with respect to multi-/many-core COTS processors. 2.) A top-down approach addresses the management of the process-, system- and certification-requirements and supports understanding new design patterns that arise from new avionics platform architectures.

II. State-of-the-Art

The set of system functions running on an avionics platform is constantly growing in size and complexity. Today, multi-/many-core processors are commonly foreseen as a key enabling technology for managing this challenge. Through integration of multiple functions on a single computing device, they allow to meet tight requirements on size, weight and power consumption, as well as to increase the overall computing power.

However, running multiple functions concurrently on different processor cores does not prevent them from sharing other resources such as cache, memory, buses and I/O peripherals. The resulting interference may increase latency and jitter of operations and thus violates safety and reliability requirements if not handled properly. The same effects occur when a function with highly parallelized code blocks is executed. Therefore, the development of an avionics platform using multi-/many-core processors is subject to strict regulations and certification processes.

The certification requirements for hardware in avionics are described in DO-254 [30] and for single-core software development in DO-178C [31]. Both standards employ the DALs to categorize critical (e.g. flight control) and non-critical (e.g. passenger infotainment) functions. The benefits of multi-/many-core processors as well as the upcoming obsolescence of single-core processors led certification authorities to develop supplements in the last years. As DO-178C focuses solely on single-core processors, the certification authorities software team published a position paper CAST-32A [36] to address the impact of multi-/many-core processors on safety, performance, and integrity. Furthermore, ARINC653 Avionics Application Software Standard Interface Part 1 Supplement 5 adds service capabilities for multi-core processors since 2019 [4].

A. Multi-Core Architectures

In the following sections the challenges resulting from multi-/many-core processors with respect to aforementioned standards are described first. Next, dedicated mechanisms to meet these requirements are presented.

1) High Performance Architectures: The majority of multi-core COTS processors are designed to increase average-case performance of all integrated functions. In most cases, COTS processors consist of several processing cores with a hierarchy of private and shared caches. The processing cores are connected via Network-on-Chip transporting interrupt, cache coherency and memory messages. Other shared resources include I/O devices and memory peripherals. Functionally independent software functions can be executed simultaneously on different cores of COTS processors. However, accesses to shared resources lead to a wide range of interference effects [12] what makes the runtime behavior of individual software functions difficult to predict. Safety-critical functions are designed for the worst-case execution time (WCET) and must meet strict temporal requirements (e.g. deadlines, arrival jitter etc.). Exceeding the WCET can lead to a complete system failure resulting in loss of life. To compute a safe upper bound on execution time, it is necessary to identify all sources of interference that could cause runtime variations resulting from competing accesses from applications running on different cores. Semiconductor manufacturers do not publish all information about their products, such as the interconnect structure. As proven in related research [8], measurement techniques and stressing benchmarks to characterize the architecture are complex and do not always detect all sources of interference that cause runtime variation. Without this information, it is hardly possible to assure the WCET. This leads frequently to overestimation of WCET in order not to compromise functional safety. This decreases utilization and performance and squanders the majority of benefits of such architectures in the avionics domain, e.g. [27] [32].

2) Architectures with dedicated Safety Features: To tackle some of the previous mentioned challenges hardware manufacturers started to incorporate specific safety features in COTS platforms. As of Revision ARMv8.4-A ARM-based processors can be equipped with the Memory System Resource Partitioning and Monitoring (MPAM) [29] extension. This allows monitoring of memory traffic and observation of performance for allocating system resources such as cache capacity and memory bandwidth to different parts of the system, where they are needed to improve interference effects.

A different safety approach is presented in [26] where failure rates are reduced by incorporating Failure Mode Effect and Diagnostic Analysis (FMEDA) into Electronic Design Automation (EDA) tools. The system is divided into separate hierarchical levels and Failure Modes (FMs) are defined. The safety hierarchy is linked to the design hierarchy to associate FMs to the corresponding part of the design. The metric describing the probability of failure is evaluated and calculated based on hardware failure rates. With heuristics the FM distribution is evaluated and safety mechanisms are inserted into the design to cover the FMs.

In other cases, manufacturers include hardware features for specific tasks in a platform. For example, in the automotive domain, the heterogeneous S32V processor from NXP [33] is designed to comply to the safety requirement set dictated by ISO 26262. Although this example is primarily based on automotive use cases, these kinds of processors will likely be more prevalent in the avionics domain and could play a major role in the development for future avionic platforms incorporating multi-/many-core.

B. Avionic Research Projects

Over the last decade, there have been many research and development projects that have addressed the challenge of multi-core use in avionic systems. The following is a brief summary of some of the research projects and their approaches.

Real-Time Capable Designs: Safe and tight WCET estimation is difficult on multi-core COTS processors. Therefore, MERASA, parMERASA and PREDATOR focused on developing a multi-core hardware architecture that enables simplified WCET analysis. In order to minimize intertask interference, the MERASA architecture isolates the tasks at core level and introduces an interference-aware bus arbiter, a dynamically partitioned cache and an analyzable real-time memory controller [37]. However, shared memory and bus-based techniques limit the design to about four to eight cores.

In the successor project parMERASA, the objective was to increase the performance up to 64 cores. The focus was on parallelization for hard real-time applications and the usage of a scalable, timing analyzable interconnect instead of an interference-aware bus. Similar to MERASA, the PREDATOR architecture approach is based on analyzable caches, compiler-controlled memory management and simple cores with analyzable behavior [15].

Probabilistic Timing Analysis: Probabilistic timing analysis (PTA) requires that execution times of an application on multi-core processors can be modeled with independent and identically distributed random variables. In the PROARTIS project, the central hypothesis is that multi-core and software features enable a truly randomized timing behavior [23]. Under this assumption, the PTA can be used to verify that the probability of execution time outliers is negligible. For example, cache replacement policies do not meet the PTA requirements. Therefore, a random placement policy was proposed in PROARTIS [20]. The successor project PROXIMA also considers PTA implemented on FPGA based COTS technologies including disruptive effects at their non-PTA compliant periphery [11].

Architecture and Mechanism: Several projects focused on ensuring safety guarantees in Multiprocessor System-on-a-Chip through extensions that enable interference mitigation. In the RECOMP project partners analyzed the impact of sharing on-chip resources in a COTS architecture and the resulting interference that affects determinism, e.g. [21]. The results suggested possible approaches to mitigate the undesirable effects or limit their temporal impact. However, RECOMP demonstrated that there is no unified approach to address all of the problems outlined. This work continued in the ARAMIS I & II projects where partners focused on introducing additional phases into modern timing analysis techniques to capture resource utilization and calculate an interference delay, e.g. [28]. Subsequently, this was complemented by run-time monitoring and shaping to enforce timing guarantees. The results showed a reduction in multi-core WCET of up to 53% by limiting interference scenarios. However, the performance degradation was still significant. Therefore, as a step towards certification, Airbus identified in EMC2 the need for solutions that maintain the ARINC653 single-core application while scheduling additional safety-critical partitions on the other cores, e.g. [2]. It has also been shown that the static throttling of memory bandwidth can cause such partitions to slow down or leave little bandwidth available to other cores.

Certification: Avionic platforms must comply with multiple safety standards (DO-178C, DO-254, ARINC653, ...) which is one of the greatest challenges for certifying computing devices with multi-/many-core processors within these platforms. Multiple research efforts tried to find solutions for some of the biggest certification hurdles, e.g. the request for freedom of interference and timing predictability. Two prominent projects in this regard are CERTAINTY [14] and PHYLOG [10].

CERTAINTY focuses on temporal isolation and presents the Acquisition Execution Restitution (AER) programming model. Each task runs on a separate core to maximize performance but communication is done sequentially. This mitigates interference since buses and shared resources will not be used simultaneously. By using PikeOS as an operating system and the Kalray MPPA-256 processor, higher predictability was obtained by replacing cache hierarchies by non-standard DMA based memory transfer.

PHYLOG describes a model-based approach by identifying interference within a given system by modeling and benchmarking the hard- and software components with the Phylog Modeling Language (PML). These models are analyzed in terms of overlapping communication paths which hint at possible interference. The resulting information is used for constructing argumentative structures to fulfill certification requirements defined in the CAST-32A papers.

Currently no complete method for certifying avionics platforms using multi-/many-core processors capable to run mixed criticality system functions exists.

C. *Industrial Efforts*

Besides academic work, industrial actions are fundamental for progress in this domain. In the following two recent advancements in industrial research are highlighted.

Quantifying Interference Empirically: Similar to [27], Rapita Systems [38] developed a test methodology to measure interference empirically. At first, a *Platform Characterization* is conducted by running generic known applications (called RapiDaemons) in a scenario alone, followed by a stressing scenario, where the remaining cores intensively access the same shared resource. A comparison of the execution times of the two scenarios reveals how prone to interference the computing device is on this/these particular shared resource(s). This way it is possible to argue, that certain shared resources, on which no effect is seen, will not cause interference with other real applications. Secondly, a *Software Characterization* is conducted, in which the application of interest will be investigated. In a finger printing stage it will be determined which shared resources the application uses. Afterwards the application will be executed while other cores, using the RapiDaemons again, stress the target system application on exactly those resources. The idea is that such a setup constitutes the worst case of possible interference and thus justifies an upper bound on the WCET.

Progress in Multi-Core Real Time Operating Systems: The operating system has the key role of distributing resources to applications, while mitigating interference on shared resources. INTEGRITY-178 tuMP is one of interest because it has proven to mitigate interference sufficiently, since it successfully managed to receive certification in a multi-core module in regards to all CAST-32A objectives [17]. The interference mitigation approach chosen is a bandwidth allocation and monitoring functionality. This is a fine-grained control of shared resources like the chip-level interconnect and enforces a fixed bandwidth utilization of applications [34].

III. Upcoming Demands

In the following, two selected use-cases show the potential and hurdles for running system applications on multi-/many-core processors.

1) Use-Case 1: In the field of flight-guidance, satellite navigation has become one of the major players over the last decade. This is mainly based on global availability and the benefits of maintaining less ground infrastructure. However, as operations expand, safety issues that arise from the increased number of satellites available must be addressed.

Typically, safety is achieved by providing an adequate integrity estimation called Receiver Autonomous Integrity Monitoring (RAIM). RAIM can be implemented as different types of algorithms that operate in either the residual or position domain of a navigation solution.

A common strategy is to assume that one of the received satellites is sending hazardously misleading information (HMI), caused by an undetected failure. The task is to identify the faulty satellite and exclude it from the navigation solution. This can be accomplished by calculating different sub-position solutions, each of which discards a different satellite from the calculation. By comparing all sub-position solutions, one of them can be identified as fault free, since the faulty satellite was discarded.

A disadvantage of this algorithm is the high demand for computing power, which increases linearly with the number of satellites, as an additional position solution has to be calculated for each satellite. In addition, a high number of satellites is depleting the available integrity budget, assigned by the safety standards. The budget refers to the maximum allowed probability, that HMI occurs undetected.

The overall probability of HMI caused by a single satellite failure can be determined using equation (1), where n is the number of satellites and k the number of faulty satellites.

$$p(HMI)_{apriori} = \binom{n}{k} \cdot p_{sat}^{k} \cdot (1 - p_{sat})^{n-k} \tag{1}$$

Depending on the chosen a priori failure probability of a satellite ($p_{sat} \approx 1 - 5 \times 10^{-5}$) [16], the overall probability is compared with the required continuity probability 2×10^{-4} assigned by the ICAO [18]. Alternatively, a comparison can be made against the missed detection probability $2 - 8 \times 10^{-4}$ [24].

As the overall probability is strongly related to the number of satellites, it can be seen that the integrity budget is depleted within 20 satellites. In contrast, new multi-constellation concepts will feature up to 40 satellites in view, assuming approximately 10 satellites per constellation. It is therefore necessary to check for a multiple satellite failure ($N_{failure}$) rather than a single one to allow for an additional margin of integrity [9, 40].

This opens up a large number of permutations on the sub-positions solutions (N_{sub}) to be calculated and increases the required computing power exponentially as seen from equation (2).

$$N_{sub} = \sum_{k=1}^{N_{failure}} \binom{n}{k} \tag{2}$$

For example, in a multi-constellation scenario with 20 satellites, up to 210 sub-solutions must be computed to satisfy the two-failure assumption, which is driven by the additional integrity margin. In comparison, the single failure assumption currently used only requires 20 sub-solutions. Therefore, it is expected that the required computing power will increase by a factor of 10 to 20 compared with the current demand.

Fortunately, the computation of the sub-solutions can be completely parallelized as they do not depend on each other. This opens up the opportunity of using multi-/many-core- instead of single-core processors, especially for applications in real-time environments.

Another opportunity for using multi-/many-core systems lies within the flight control domain. More computing power would enable using new methods in control technology, which allows better system performance [35], robustness [39], fault-tolerance [22] of nonlinear systems, possibilities for online system identification up to adaptive control or Model Predictive Control (MPC)[1].

MPC uses discrete dynamic models of the plant to calculate the future dynamic behavior of the vehicles input signals. This enables the calculation of the optimal input depending on a chosen quality function and therefore an optimal output results. While the vehicles behavior is predictable within a certain time frame, usually only the input is used for the next time step and then the optimization is repeated. This requires considerable computing power, which is why MPC controllers are preferred in process engineering processes whose dynamics are slow enough to perform an optimization in each sampling step. Various parallel computing approaches using different technologies have been proposed to speed up the execution of MPC. An overview of this topic is provided by [1], in which the parallelization of the MPC calculation using multi-core processors (CPUs) and many-core processors (GPUs) is presented.

2) Use-Case 2: The aircraft's air conditioning system (ACS) as part of the Environmental Control System (ECS) is being examined as a further application with regard to multi-/many-core processors, whereby the ACS is not only responsible for temperature control and passengers comfort, but also for the pressurization and ventilation of the cabin. The following explanations are taken from [13, 3].

As the ACS ensures structural fuselage integrity by controlling the required pressure according to a pressurization flight profile, it can be assigned to the safety-critical systems. Since the flight profile is altitude-dependent and the number of passengers is required as an input value for the control, the ACS controller must communicate with the flight management system (FMS).

Although the aircraft types vary, the essential principles of an ACS are comparable for all aircraft and are briefly explained here with regard to the control. The system mixes hot and cold air to achieve the desired temperature which is set via a manual knob by the pilot and serves as a reference variable for the controller. The hot air is provided via the bleed air of the engines and the cold air is provided by two Air Conditioning Packs. Inside each pack the bleed air is cooled by Ram-air through one heat exchanger and an Air Cycle Machine (ACM) to deliver the coldest required temperature, which is controlled accordingly by adjusting valves. In addition, parts of the air from the cabin, cockpit and cargo is recycled and mixed with fresh air from the packs inside the mixer. Finally, hot bleed air is mixed with the cold air from the mixer for the individual zones in the aircraft via the trim air system. While the temperature is only achieved by mixing hot and cold air, the cabin pressurization is accomplished by controlling the amount of air that flows out of the cabin. This is done by using e.g. the outflow valve and the bleed air valve, which regulates the amount of air entering the system detected via the flow and pressure sensors.

The temperature is controlled by adjusting different valves inside various subsystems within the ACS. For example, the Temperature Control Valve is responsible for the temperature control inside the pack. In order to implement the temperature control, various state variables must be measured by sensors inside the system and compared with the reference values given by the controller forming the control loop.

There are several types of states that need to be controlled. One of the types are logic states used primarily in emergency systems for fault isolation, which are not used during regular operation. These are mostly controlled via logic manipulated variables that are rarely used and for this reason require low computing power, e.g. packing valves. Packing valves close automatically if there is insufficient pressure upstream or if the fire button of the associated engine is pressed and the corresponding cross-bleed valve is closed. The other type are continuous manipulated variables, which are determined by a controller. Controllers of aircraft critical systems are typically kept as simple as possible due to certification efforts. For this reason, Proportional-Integral (PI) controllers are used whenever possible. PI feedback- and feed-forward-controllers do not require high computing power. This and the fact that many feedback sensors and actuators across the aircraft need to be connected which results in a significant additional cable-weight if all applications are centralized. Therefore, ECS was deliberately chosen as an example to show that not all applications benefit from multi-/many-core processors yet.

IV. Future Concepts

The concept of Integrated Modular Avionics (IMA) already allows to run several system functions of different DAL on the same single-core computing device. But with respect to the number of processing cores, another degree of freedom on the computing device is added that must be managed. System functions depend on sensors and actuators that must be connected physically to the computing device and functions of different complexity running at various rates. Considering the spatial distribution of sensors and actuators of systems across the aircraft, moving functions from multiple single-cores to probably centralized multi-/many-core computing devices is challenging. There is no experience with platform architectures and how computing devices and functions are optimally allocated considering all these side-effects, e.g. constraints like requiring some functions being implemented on dissimilar hardware.

This problem is addressed within the Many-Core Avionics Design, Architecture, Modeling and Simulation (MC-ADAMS)[1] project. To approach the design problem, a detailed view on the computing device is conducted to ensure guarantees for the system design (bottom-up). Furthermore, system demands, especially from segregation/redundancy requirements, will define constraints that must be considered when allocating functions on the multi-/many-core platform (top-down). Design patterns that require multiple single-core computing devices due to safety, may also require multiple computing devices when using multi-/many-core processors.

In the following, the topic will be approached in two ways: Bottom-up and top-down which is explained in the following paragraphs.

A. Bottom-Up Approach

A detailed analysis of the behavior of applications on computing device level is carried out to achieve guarantees at the system level. This corresponds to most of the approaches discussed in section II. The basic procedure is to start with a detailed hardware analysis and provisionally determine potential roots of non-determinism. It is common to distinguish, for example, between the three phases of interference-identification, -analysis and -mitigation [19]. However, there are innovative and less restrictive ideas of ensuring guarantees of determinism and bounds on WCET. The idea of *Timing Diversity* is one currently being investigated in the scope of the MC-ADAMS project [25].

Hardware mechanisms for contemporary multi-core processors such as caching, pipelining and speculative execution were introduced to reduce the average execution time. However, the same mechanisms may have a negative impact on the worst-case behavior of the system. For instance, although cache memory improves an average execution time the sporadic cache misses may significantly increase execution times of certain code sections and endanger system safety. Timing diversity is an approach based on modular redundancy that mitigates this challenge. It executes the same function on multiple units with different hardware architectures. As timing errors should be sporadic (safety critical functions are usually well specified and tested due to certification), it may be assumed a timing error does not occur on two hardware platforms simultaneously if one can demonstrate statistical independence. In this case, one can set up a *single-timing-error* model, which means that one of two implementations always returns the result in time. The timing diversity approach is similar to the dual modular redundancy (DMR) for hardware errors, used e.g. in AFDX [7].

The setup considered in MC-ADAMS project is presented in Fig. 1. It consists of a commercial single-core ARINC653 front-end processor connected to two high-performance computing platforms at the back-end. The functions are forwarded

[1]MC-ADAMS website: https://tinyurl.com/mcadamsproject

to the back-end via the ARINC653 front-end and executed redundantly.

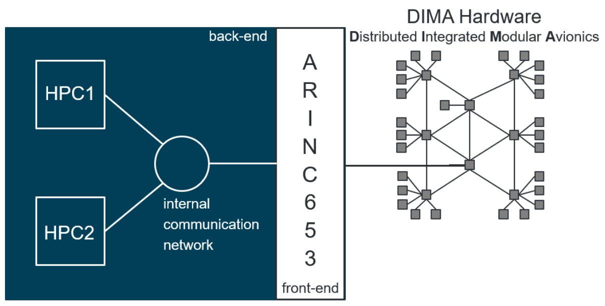
DIMA Hardware
Distributed Integrated Modular Avionics
back-end
HPC1
HPC2
internal communication network
A R I N C 6 5 3
front-end

Fig. 1. MC-ADAMS setup consists of a commercial ARINC653 front-end connected to two high performance computing platforms HPC1 and HPC2 via an internal communication network back-end

B. Top-Down Approach

The top-down approach addresses the avionic platform design and related management of the process-, system- and certification-requirements. In order to maximize the usage of multi-/many-core for avionic platforms, the platform should consist of an optimal set of computing devices for the various system functions. An optimal set minimizes costs or weight of certain required resources. The design of single-core avionic platforms is already a complex process due to the sheer number of applications and their relations in terms of constraints to each other. Therefore, model-based methods have been developed in the past to capture architecture requirements and design avionic platforms using domain specific models [6]. A promising domain specific model focusing on the mapping of e.g. tasks to devices and devices to installation locations as well as mapping of signals and I/O is the *Open Avionics Architecture Model* [5]. It does not support modeling multi-cores out of the box. Therefore, the underlying model shall be enhanced to support abstractions for cores as an own sub-device, resulting in a more fine grained description of the avionic architecture.

In order to design such an avionic architecture, characteristics from cores and operating system have to be collected. Those include assumptions on the reliability, provided computing power and other provided resources like I/O. The provided characteristics, the system requirements and the certification requirements for CS-25 aircraft enforce specific designs and architectures with necessary redundancy.

One example is the task assignment. On classic architectures using single-cores, tasks are assigned to devices. But when using multi-core devices, tasks will be assigned to cores. If enough resources are available, the application can be allocated to a device/core. This is a combinatorial optimization (bin packing) problem with additions of constraints for functional safety. These functional safety constraints can be for example: *application* A_1 *and* A_2 *need to be on different computing devices*. This might have a significant impact on the design process and will potentially limit the utilization of many-core systems in certain cases.

Another constraint is the thermal management. A higher core number dissipates more heat at the same clock frequency and may require cooling systems. In nowadays aircraft, cooling capacities are typically only available in the avionics bay.

Other design factors regard the number and type of used components. Nowadays, an avionics platform consists of computing devices and remote data concentrators that connect field buses to an aircraft data communication network. As the amount and length of connected wires contributes to weight, wires influence the topology and weight of the overall architecture significantly. This leads to the question: Do multi-/many-cores tend toward centralized or distributed architectures? Are new installation locations required that provide active cooling, for example in the center or back of an aircraft?

Fig. 2 points out the transition to architecture design with multi-/many-core hardware and underlines the density of applications usually being ARINC653 partitions.

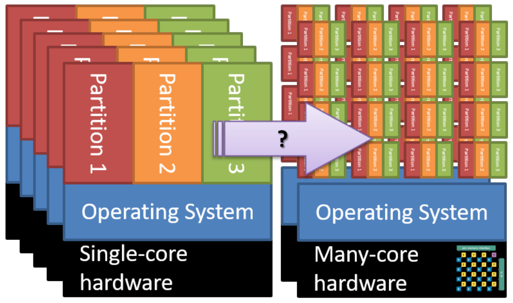

Fig. 2. Transition to architecture design with multi-/many-core hardware

C. Use Case and Assessment Approach

To investigate feasible multi-/many-core platform designs, MC-ADAMS starts with the architecture of an existing long-range aircraft with an IMA platform as reference. This will setup the number and location of sensors and actuators for the systems. On modern aircraft around 1000 system functions are running on IMA and this number will likely increase.

To avoid the need of detailed system functions, the MC-ADAMS project proposes a *template system application* based assessment approach. In a first step system applications are analyzed and grouped according to similar resource demands and algorithm complexity. Examples are: Matrix operations, controller types, logical operations or signal-processing. Each group will be represented by a template system application.

This approach enables generation and assessment of a realistic set of applications for the evaluation of novel multi-/many-core IMA architectures. Moreover, the approach is legitimate, since it captures and maintains typical characteristics of nowadays system applications on IMA architectures.

Therefore, it can be assumed that the template system applications with different characteristics like criticality, computing demand or type and number of required I/Os, well represent real applications. Furthermore, it can be assumed that design patterns (simplex, duplex, triplex, con-mon-voter...) can be expressed in a statistical manner based on known current system applications on IMA. The given component hardware topology defines application assignments and signal paths. This will relate template system applications and define logical and physical communication paths across the aircraft. The next step is to express the type and number of hardware items required for an avionics platform where these templates can be instantiated on. Signal flows and paths are also considered to obtain a realistic virtual avionics platform model.

To assess parameters like WCET on different n-core computing devices, the template applications will be investigated on a processor simulation to validate the design. Thus, it shall be possible to statistically examine the effect of different design options like computing device types (i.e. core count), number of applications, ratio between applications of different criticality, centralized vs. distributed, use of data concentrators etc. This will result in different design points that can be compared based on the architecture model derived and with respect to different cost functions, like size, weight, number of devices etc.

V. Summary and Outlook

In the process of integrating multi-/many-core computing devices into an avionics platform, various issues arise, with the biggest hurdle being the assurance of determinism. The state-of-the-art methods briefly summarized in section II highlight several approaches like probabilistic timing analysis, none of which are used in the avionics industry. The application specific demands for avionic systems like Satellite Navigation or Environmental Control System were explained in section III and show varying potentials with respect to the utilization of the technology. Based on the given situation, there is not yet a design method to integrate multi/many-core into avionics platforms, especially for IMA platforms.

Within the MC-ADAMS project, new concepts for a holistic integration approach of multi-/many-cores in the avionic domain will be explored. This incorporates all arising issues from the high level platform application and communication down to the individual processors and on-chip communication. The next step will be to evaluate the practical limit of the core number, respecting current certification and safety requirements. Therefore different device types will used and examined as well as the impact on the aircraft data network. That will be done using the template approach described in section IV.

Considering that information, new design methods applicable to multi-/many-core avionic systems will be developed. These will be based on an assessment of different design strategies using simulation models. In order to compensate for a realistic number of applications, further real system applications based on the experience of the project partners are collected and elaborated.

Acknowledgement

This work was funded by the German Federal Ministry of Economic Affairs and Energy (BMWi) within the MC-ADAMS project. Their support is greatly appreciated.

Supported by:

Federal Ministry for Economic Affairs and Energy

on the basis of a decision by the German Bundestag

References

[1] Karam Abughalieh and Shadi Alawneh. "A Survey of Parallel Implementations for Model Predictive Control". In: *IEEE Access* PP (Mar. 2019), pp. 1–1.

[2] Ankit Agrawal et al. "Contention-Aware Dynamic Memory Bandwidth Isolation with Predictability in COTS Multicores: An Avionics Case Study". In: *29th Euromicro Conference on Real-Time Systems (ECRTS)*. Ed. by Marko Bertogna. Vol. 76. Leibniz International Proceedings in Informatics (LIPIcs). Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017, 2:1–2:22. ISBN: 978-3-95977-037-8.

[3] Airbus. *Airbus A380 FLIGHT CREW OPERATING MANUAL*. 2005.

[4] Airlines Electronic Engineering Committee. *653P1-5 Avionics Application Software Standard Interface, Part 1, Required Services*. Ed. by SAE Industry Technologies Consortia. Dec. 23, 2019.

[5] Björn Annighöfer. "An open source domain-specific avionics system architecture model for the design phase and self-organizing avionics". In: *SAE International Journal of Advances and Current Practices in Mobility* 1.2019-01-1383 (2019), pp. 429–446.

[6] Björn Annighöfer. "Model-based Architecting and Optimization of Distributed Integrated Modular Avionics". Dissertation. Hamburg University of Technology, Mar. 2015. 312 pp. ISBN: 978-3-8440-3420-2.

[7] *ARINC Specification 664: Aircraft Data Network, Part 7*. 1. September 2009.

[8] Jingyi Bin et al. "Studying co-running avionic real-time applications on multi-core COTS architectures". In: *ERTS 2014*. 2014.

[9] Juan Blanch et al. "Advanced RAIM user algorithm description: integrity support message processing, fault detection, exclusion, and protection level calculation". In: *Proceedings of the 25th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS 2012)*. 2012, pp. 2828–2849.

[10] Frédéric Boniol et al. "PHYLOG certification methodology: A sane way to embed multi-core processors". In: *10th European Congress on Embedded Real Time Software and Systems (ERTS 2020)*. Toulouse, France, Jan. 2020.

[11] *Concepts — proxima Public Website. May 06, 2021. URL: http://proxima-project.eu/concepts*.

[12] Dakshina Dasari et al. "Identifying the sources of unpredictability in COTS-based multicore systems". In: *8th IEEE International Symposium on Industrial Embedded Systems (SIES)*. Piscataway, NJ: IEEE, 2013. ISBN: 9781479906581.

[13] M. Dechow and C.A.H. Nurcombe. *Aircraft Environmental Control Systems*. Ed. by Martin Hocking. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. ISBN: 978-3-540-31491-2.

[14] Guy Durrieu et al. "Predictable Flight Management System Implementation on a Multicore Processor". In: *7th Conference on Embedded Real Time Software and Systems (ERTS)*. 2014.

[15] Alexander A. Evstyugov-Babaev. *Reconciling Efficiency with Predictability: the PREDATOR Approach. May 06, 2021. URL: https://www.predator-project.eu/approach.htm*. 26.01.2011.

[16] *Global Positioning System Standard Positioning Service Performance Standard*. Tech. rep. Department of Defense, 2020.

[17] Green Hills Software. *Press Release: World's First Multicore Avionics Certification to CAST-32A Uses the INTEGRITY-178 tuMP Multicore RTOS*. Danske Bank. Mar. 17, 2021. URL: https://www.ghs.com/news/20210316_CAST32A_certification_integrity.html.

[18] ICAO. "Annex 10 to Convention on International Civil Aviation, Aeronautical Telecommunications". In: *Volume I, Radio Navigation Aids, Seventh Edition, ICAO Montreal* (2018).

[19] Xavier Jean, Laurence Mutuel, and Vincent Brindejonc. "Assurance methods for COTS multi-cores in avionics". In: *2016 IEEE/AIAA 35th Digital Avionics Systems Conference (DASC)*. IEEE. 2016, pp. 1–7.

[20] Leonidas Kosmidis et al. "Efficient Cache Designs for Probabilistically Analysable Real-Time Systems". In: *IEEE Transactions on Computers* 63.12 (2014), pp. 2998–3011. ISSN: 0018-9340.

[21] O. Kotaba et al. "Multicore in Real-Time Systems – Temporal Isolation Challenges due to Shared Resources". In: *DATE 2013*. 2013.

[22] Peng Lu et al. "Aircraft Fault-Tolerant Trajectory Control Using Incremental Nonlinear Dynamic Inversion". In: *Control Engineering Practice* 57 (Sept. 2016).

[23] *Main Objectives — PROARTIS Public Website. May 06, 2021. URL: http://www.proartis-project.eu/main-objectives*.

[24] Anais Martineau, Christophe Macabiau, and Mikael Mabilleau. "GNSS RAIM assumptions for vertically guided approaches". In: *Proceedings of the 22nd International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS 2009)*. 2009, pp. 2791–2803.

[25] Mischa Möstl et al. "Work-in-Progress: Timing Diversity as a Protective Mechanism". In: *2021 International Conference on Embedded Software (EMSOFT)*. 2021.

[26] Alessandra Nardi et al. "Design-For-Safety For Automotive IC Design: Challenges And Opportunities". In: *IEEE Custom Integrated Circuits Conference, CICC, Austin, TX, USA, April 14-17, 2019*. IEEE, 2019, pp. 1–8.

[27] J. Nowotsch and M. Paulitsch. "Leveraging Multicore Computing Architectures in Avionics". In: *Ninth European Dependable Computing Conference (EDCC)*. Piscataway, NJ: IEEE, 2012. ISBN: 9781467309387.

[28] Jan Nowotsch et al. "Multi-core Interference-Sensitive WCET Analysis Leveraging Runtime Resource Capacity Enforcement". In: *2014 26th Euromicro Conference on Real-Time Systems*. 2014, pp. 109–118.

[29] Falk Rehm et al. "The Road towards Predictable Automotive High-Performance Platforms". In: *Proceedings of DATE*. 2021.

[30] RTCA, Inc. *Design Assurance Guidance for Airborn Electronic Hardware: RTCA/DO-254*. 19. April 2000.

[31] RTCA, Inc. *Software Considerations in Airborn Systems and Equipment Certification: RTCA/DO-178C*. 13. Dezember 2011.

[32] Selma Saidi et al. "The shift to multicores in real-time and safety-critical systems". In: *2015 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*. 2015, pp. 220–229.

[33] Semiconductors, N.X.P. *S32V234 Data Sheet*. Tech. rep. S32V234 Rev. 9. Mar. 2020.

[34] Green Hills Software. *Optimal Multicore Processing for Safety-Critical Applications*. White Paper and Technote. 2020.

[35] Brian L. Stevens, Frank L. Lewis, and Eric N. Johnson. "Modern Design Techniques". In: *Aircraft Control and Simulation: Dynamics, Controls Design, and Autonomous Systems*. John Wiley & Sons, Ltd, 2015. Chap. 5, pp. 377–499. ISBN: 9781119174882.

[36] Will Struck TAD ANM-111. "Certification Authorities Software Team (CAST)". In: (2016).

[37] Theo Ungerer et al. "Merasa: Multicore Execution of Hard Real-Time Applications Supporting Analyzability". In: *IEEE Micro* 30.5 (2010), pp. 66–75. ISSN: 0272-1732.

[38] Steven H VanderLeest and Samuel R Thompson. "Measuring the Impact of Interference Channels on Multicore Avionics". In: *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*. IEEE. 2020, pp. 1–8.

[39] Greg Walker and Dave Allen. "X-35B STOVL Flight Control Law Design and Flying Qualities". In: *2002 Biennial International Powered Lift Conference and Exhibit*. American Institute of Aeronautics and Astronautics, June 2002.

[40] *Working Group C - ARAIM Technical Subgroup*. Tech. rep. EU-U.S. Cooperation on Satellite Navigation, Feb. 2016.