Observing the Impact of Multicore Execution Platform for TSP Systems Under Schedulability, Security and Safety Constraints

Ill-Ham Atchadam, L Lemarchand, Frank Singhoff, Hai Nam Tran

Observing the impact of multicore execution platform for TSP systems under schedulability, security and safety constraints

Ill-ham Atchadam[1], Laurent Lemarchand[1],
Frank Singhoff[1], and Hai Nam Tran[1]

Univ. Brest, Lab-STICC, CNRS, UMR 6285, F-29200 Brest, France
{firstname.lastname}@univ-brest.fr

Abstract. Avionic systems are integrating more and more functions to cope with the increasing number of features on modern aircrafts. These systems are subject to many requirements that have to be considered during their design. Time and Space Partitioning (TSP), which consists of isolating applications within partitions, is a well-known means to assign avionic applications to computing units according to security, schedulability, and safety constraints. Multicore execution platforms are becoming popular in avionic systems. In this paper, we propose to investigate the partitioning of avionic applications over such execution platforms while considering schedulability, security, and safety constraints. We propose a design space exploration approach using a multi-objective meta-heuristic, that provides trade-offs between schedulability and security while considering safety and multicore platforms with different numbers of cores. We illustrate how this meta-heuristic can investigate key parameters such as hardware shared resource overhead.

Keywords: Time and Space Partitioning · Scheduling · Safety · Security · Architecture Exploration · Multi-Objective Optimization · Multicore.

1 Introduction

SWaP (size, weight, and power) are great challenges in the avionic domain. Avionics systems were designed based on a federated approach [10] where each function had its own dedicated computing system [2]. To cope with the increasing number of functions embedded in an aircraft, the integrated avionic modular (IMA) [9] approach has been proposed to provide a pool of shared computing, communications, and I/O resources that is partitioned for use by multiple avionics functions [25]. IMA considers time and space partitioning (TSP) which guarantees time and space isolation between functions on the shared hardware resources. Space isolation is obtained by memory protection between partitions while time isolation is enforced by offline partitioned scheduling.

Avionic functions implemented as real-time tasks in TSP systems have stringent constraints on safety, security, and schedulability. Safety is enforced by both the isolation through partitioning to prevent fault propagation and by the active

redundancy, i.e. replications of tasks and partitions. Security requires that malicious entities cannot compromise the confidentiality of data exchanged between tasks. This can be achieved through encryption mechanisms. Implementation of safety and security measures must not affect system's schedulability as timing constraints must be respected to ensure the correct functionality of the system.

[***Problem statement***] A safe and secure TSP system implies additional tasks, partitions, and communications for safety, and additional overheads for encryption. This leads to a non-negligible increase in computation need, which may lead some tasks to miss their deadlines. Therefore it is important to propose approaches that can overcome the overheads due to security and safety requirements. This can be achieved by increasing the computing capacity of the system with multicore execution platforms. In this paper, we investigate how the use of multicore platforms may help in improving the safety and security of TSP systems while not jeopardizing their schedulability.

[***Contributions***] In this paper, we propose a Design Space Exploration (DSE) approach based on the Pareto Archived Evolutionary Strategy (PAES) meta-heuristic that provides trade-offs for multi-objective optimization problems (MOOP) between safety, security, and schedulability for TSP systems on multicore platforms. We explore the tasks to partitions assignment in TSP systems when communications are secured and tasks are replicated. We show that our approach allows designers to explore the gain in terms of schedulability while considering security and safety. To validate the approach, we conduct two experiments. We analyze the search space by varying the number of cores. A first experiment provides consistent results showing schedulability improvements when the number of cores is increased, which assesses the relevance of our DSE. A second experiment illustrates the interest of our DSE, the experiments investigate the impact of multicore hardware shared resources on schedulability.

The rest of this paper is structured as follows. Section 2 presents the background together with the system model and assumptions taken in our work. Section 3 describes our DSE approach. Section 4 shows the experiments conducted to evaluate the DSE approach. Section 5 discusses related work and positions our contribution. Finally, Section 6 concludes the paper.

2 Background and assumptions

In this section, we present system model and assumptions taken in our work. Then, we also present the context of security and safety that we consider.

2.1 System model and assumptions

In this paper, we consider a multicore TSP systems of m applications $(A_1, ..., A_m)$ where each application is a set of tasks. Systems considered are composed of a set n periodic tasks $(\tau_1, ..., \tau_n)$. We assume a multi-core architecture of d identical cores $(CO_1, ..., CO_d)$.

Each task τ_i is defined by a set of parameters (C_i, T_i, D_i, CI_i, CL_i, A_i, P_i, CO_i). C_i, called the capacity of the task τ_i, represents its worst-case execution time. A task is released every T_i unit of time and has a deadline at D_i. We assume that the initial request of all the tasks is at time 0. A task is characterized by a tolerance level CI_i (hard or soft) to meet its deadline. A task is classified based on a confidentiality level CL_i (Top-secret, Secret, Unclassified). A_i represents the application to which the task τ_i belongs. P_i characterizes the partition to which the task τ_i is assigned. A partition is characterized by an execution time duration. We assume that all the partitions have the same properties and are executed based on an offline cyclic scheduling with a fixed interval called *major time frame* (MAF). Finally, a task is assigned to a core CO_i and core migration is not allowed at runtime.

We assumed that tasks communicate with each other through intra-partitions, or inter-partition communications depending on their assigned partitions. Intra-partition communications are communications between tasks in the same partition while inter-partition communications are about tasks assigned to different partitions. These communication services are provided by an application programming interface (API) such as the one proposed by the ARINC653 standard [5]. Both intra and inter-communications introduce overheads on the tasks concerned by the communications (i.e. sending and receiving tasks). As shown in [17], these overheads depend on the size of the exchanged data.

An offline cyclic scheduling is fixed for the partitions. Partitions are executed cyclically on an interval time called major time frame (MAF). Tasks inside partitions are executed concurrently based on a given scheduling policy (i.e. fixed-priority scheduling).

Fig 1 shows an example of scheduling of a multicore system with four tasks, assigned to two partitions and two cores. We note that τ_1, τ_3 and τ_2 are respectively assigned to core CO_1 and CO_2. For tasks to partitions assignment, τ_1, τ_2, τ_3, τ_4 are respectively assigned to partitions P_1, and P_2. The same MAF is assumed for all cores. Then when a partition is activated, only its tasks are executed concurrently on the cores depending on the tasks to cores assignment. Cores that have no task in the activated partition are in idle mode. They are not used till the activation of a partition with tasks assigned to them. In this example, we assumed that there is a communication from τ_1 to τ_2 and another from τ_1 to τ_3. Then τ_2 has to wait for τ_1 completion time before being starting its execution. This explains why even if τ_1 and τ_2 are on different cores, and τ_2 is the only task on CO_2, τ_2 could not start at time 0. It has to wait for the completion of τ_1.

Finally, we also consider overheads introduced by the hardware shared resources (level-2 cache, bus, memory, etc) when multiple cores execute simultaneously [3]. In [14], it has been proven that they are non-negligible and have to be considered when performing the schedulability analysis of a multicore real-time system. This issue is part of the key point addressed by the CAST-32A. CAST-32A is a guidance for Avionics Multi-Core Processing that highlights some

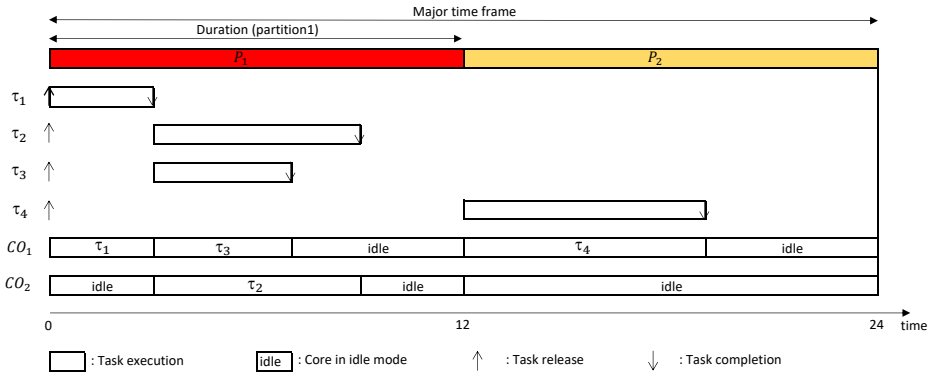

Fig. 1: Example of a multicore TSP system scheduling

parameters that may impact safety, security and performance. It considers that contention for shared resources has an impact on the execution time of tasks.

2.2 Security and safety

Security We assumed that attacks can be operated on intra-partition and inter-partition communications through attacks such as eavesdropping [26] and code injection [19] (e.g. a code injected by a malicious employee). They violate the confidentiality of a system, which normally restricts the access of data only to authorized entities.

A communication is said to be vulnerable if it violates security rules defined by a security model. In this work, we assume the Bell-La Padulla (BLP) [1] security model with regard to confidentiality vulnerabilities. It is based on the No read-up/no write-down principle. A communication from task τ_i to τ_j is considered as a confidentiality violation if $CL_i > CL_j$. In this communication, τ_i performs a write down and τ_j performs a read up, which violates the rules.

When a communication from task τ_i to τ_j is vulnerable, functions of a library implementing encryption and decryption are called. Then an encryption (resp. decryption) call is added to task τ_i (resp. τ_j) source code. Assuming the worst-case situation where the encryption key is set up at each release of a task, we also added a key set up function to both source code tasks. This process changes the capacities of tasks τ_i and τ_j as follows where C'_i and C'_j represent their new capacities. $C_{encryption\_function}$, $C_{decryption\_function}$, and $C_{encryption\_key\_function}$ represent respectively the execution times of the encryption, decryption, and key set up functions.

$$\begin{aligned} C'_i &\leftarrow C_i + C_{encryption\_function} + C_{encryption\_key\_function} \\ C'_j &\leftarrow C_j + C_{decryption\_function} + C_{encryption\_key\_function} \end{aligned} \tag{a}$$

Safety We consider safety problems induced by arbitrary failures [15]. It includes the detection that some messages are not sent or received, the detection of incorrect messages sent with errors, and the detection of extra sent messages. We assume the worst-case situation where the replication is applied to all the software components (tasks and partitions). Then with such safety constraints, each task and partition is implemented by three instances. We impose that two instances of the same task are not allowed to be placed on the same partition. This paper is not an answer to multicore plaforms with hardware single point failure; e.g. when cores are interconnected by a bus, the bus is a single point of failure, while it is not the case if cores are interconnected with a crossbar.

2.3 Multi-objective optimization

Multi-objective optimization problems (MOOP) [4] are characterized by multiple conflicting objectives to optimize: the optimization of one objective can deteriorate other objectives. Then it becomes difficult or sometimes impossible to build solutions that optimize simultaneously all the objectives. Thus, design space exploration is an alternative that helps to explore the space of solutions and propose a set of trade-offs between the objectives. The designer can then choose, between the trade-offs, the most appropriate solution to the specifications needed for his system.

The simplest approach is to investigate all the possible solutions in order to find the best trade-offs. However, for a large-scale problem, we can face a combinatorial explosion of the design space and the exhaustive search can become very time-consuming. Thus a multi-objective evolutionary algorithm (MOEA) [4] such as PAES [13] is an alternative to compute in fewer time solutions close to the best trade-offs. During the exploration, solutions are compared to each other using the Pareto dominance principle [13]. A solution $s1$ dominates a solution $s2$, if for all the objectives $s1$ is not worse than $s2$ and $s1$ is better than $s2$ for one objective at least. $s2$ is not a good trade-off and is discarded. Non dominated solutions found during the exploration constitute the Pareto set of solutions.

3 Design space exploration (DSE) approach for multicore TSP systems

In this section, we present a DSE approach that computes trade-offs between security and schedulability while considering safety constraints and resources constraints such as the number of cores and partitions. Since security, schedulability, and safety are conflicting requirements that lead to a multi-objective optimization problem, we adopt the PAES algorithm in our approach.

PAES is a meta heuristic framework. It starts with random solution(s) and transforms them using exploration operators, keeping the most interesting ones generation after generation. In order to customize PAES framework for isolving efficiently our problem, we need to identify (1) specific initial solutions, (2) constraints to perform the feasibility tests, (3) objective functions to optimize,

(4) mutation operators to generate new solutions. These components are detailed in the next section.

The entry point of the PAES being the initial solution, we proposed an initial solution adapted to our problem. During the design space exploration, new solutions are generated through mutation operations that consider tasks to partitions and cores assignments and communications security. Feasibility tests are performed according to the respect of security, schedulability, and safety constraints. Feasible solutions are evaluated according to objective functions based on schedulability and security analysis.

Initial solution We design the initial solution by resolving all confidentiality vulnerabilities in the system (as described in section 2.2) , placing all the initial tasks in the same partition running on a single core. Then we triplicate the tasks, the communications between tasks, and the partitions to ensure safety. We proceed with a schedulability analysis of this solution. If it is schedulable, there is no need to continue with the exploration: we consider this solution as an optimal solution since it is fully secured, schedulable and safe with the minimal number of cores. Otherwise, we add this initial solution to the archive.

Instead of starting the exploration with an archive containing one solution as specified in the original PAES algorithm, we fill the archive with several solutions. We made this choice to improve solution diversity and exploration of the design space. We fill the archive with solutions modeling various tasks to cores assignment and communications security. Then we added solutions with all tasks assigned to a single core and with all tasks of a partition assigned per core. For these solutions, we decided to resolve all or no security vulnerabilities.

Objective functions and constraints In the PAES meta-heuristic, the constraints are conditions that should always be respected. Otherwise, the concerned solutions are considered not valid. Objective functions are defined to tolerate some violations and should be optimized in order to propose the best trade-offs between conflicting objectives. We defined the constraints and the objective functions based on schedulability, safety, and security issues. In our model, tasks can be either hard deadline tasks or soft deadline tasks. As a constraint, a solution is considered invalid and is rejected if a hard deadline task misses its deadline. Missed deadlines are tolerated for soft deadline tasks.

– C1: No missed_deadlines for hard deadline tasks

Our first objective function is defined by the number of soft deadline tasks that missed their deadlines. This number is computed through a scheduling simulation of the solution. The function is noted below:

– F1: Minimize (number of missed soft deadlines)

Since we decide to investigate tasks to core assignment to evaluate their impact on the considered systems, our second objective function represents the number of cores used in a given solution:

– F2: Minimize (number of cores)

The problem depicted in this paper addresses the confidentiality of communications between tasks. We defined the constraints below for security vulnerabilities based on BLP rules for communicating tasks:

– C2: No data received by Unclassified task from Secret or Top-secret task

Each model that compromises one of these constraints is rejected. Otherwise, any communication violating the other BLP rules is tolerated. This allows the definition of the security objective function:

– F3: Minimize (number of tolerated confidentiality violations)

The equation F3 represents the number of tolerated communications that violate BLP rules.

Since we address safety issues by applying active redundancy, each task of our model is triplicated. By definition, this redundancy imposes that two instances of the same task should never be placed on the same partition. Then we assumed as safety constraints that every solution with two instances of a task placed on the same partition should be automatically rejected.

– C4: Two instances of the same task cannot be placed in the same partition

In order to find trade-offs for our MOOP, all the defined objective functions have to be minimized. Constraints and objective functions are computed with the Cheddar tool in which our DSE heuristic has been implemented [24].

Mutation operator Since PAES works with a neighborhood-based search, the design space is explored by mutating a solution to another nearby. We are interested in tasks to partitions assignment, tasks to cores assignment, and the security of communications between tasks.

The first mutation operator changes the tasks to partitions assignment of a solution. It is defined with two different options. The first option consists of moving all tasks of a randomly chosen application to a randomly chosen partition. The second option consists of moving a randomly chosen task to a randomly chosen partition.

The second operator is similar to the first one but changes tasks to cores assignment. Thus, the first option consists of moving all the tasks of a randomly chosen application to be executed on a randomly chosen core. The second option is operated by moving a randomly chosen task to be executed on a randomly chosen core.

Notice that the change of tasks to partitions or tasks to cores assignment has an impact on the schedulability of the solution.

The third operator concerns communications of the solution. It is realized by a random choice of a communication. If the communication presents security vulnerabilities, then we secure it by adding security functions. Otherwise, we remove the security functions and the communication becomes unsecure.

After each mutation operation, we conduct feasibility tests to check the respect of schedulability and security constraints. If the new solution generated by the mutation does not respect one of the constraints, it is rejected and another mutation operation is performed. Otherwise, if the solution respects all the

constraints, then schedulability and security analysis are performed to evaluate the objective functions of the solution.

In the next section, we propose to validate and illustrate our DSE approach through experiments.

4 Test cases and Evaluation

In these experiments, we evaluate our DSE approach with a case study with the objective of proposing a set of solutions representing good trade-offs between security and schedulability while considering safety and multicore executing platforms with different numbers of cores.

We highlight that our choices of tasks model, considered faults, and encryption algorithms are classic and from known benchmarks, but can be adapted.

Case study We use a case study composed of a set of two applications: a flight controller application ROSACE (Research Open-Source Avionics and Control Engineering) [20] and a digital signal processing application CFAR (Constant False Alarm Rate detection) [22]. ROSACE is a real-time benchmark composed of fifteen dependent and periodic tasks with the WCETs of tasks and their period taken from [20]. CFAR is a target detection application composed of four dependent tasks with the WCETs taken from the StreamIT benchmark profiled in [22]. We assume for ROSACE and CFAR, an average data size of 8 bytes.

We assumed that cores are identical and have the same predefined MAF. The partitions are identical with a duration of 1250 us. We also supposed that the tasks are periodic and their deadlines (soft or hard) and security levels are fixed independently by the designer as inputs.

Results of the experiment Considering the two applications, the initial system model is made of 19 tasks. With our safety assumptions, we triplicated partitions, tasks, and communications. This implies 57 tasks with at least 3 partitions for the DSE. By considering one of the additional initial solutions defined in 3 that runs each application per partition, we assume a DSE with a maximal number of 6 partitions. Then, we explore multiple solutions with 3, 4, 5, and 6 partitions since the safety imposes a minimum of 3 partitions.

We assumed that intra-partition (resp. inter-partition) communications are performed through blackboards (sampling ports). For their cost, we consider the execution times of the APEX calls SFPBench Benchmark proposed in [16]. Considering the data size of our case study, for blackboards (resp. sampling ports), it gives a cost of 0.76 us/0.32 us (resp. 4.24 us/5.04us) for read/write.

For confidentiality vulnerabilities securing, we used the blowfish encryption algorithm [23]. With a frequency of 1.2 GHz, we computed the time execution of security functions based on values provided by the crypto++ benchmark [8] and the data size of our applications. Then for both applications, the execution times of encryption, and encryption key refreshment are respectively 0.166 us,

and 88.83 us. We consider the decryption execution time equal to the encryption execution time.

About the shared hardware resource overheads, we only consider the interconnection overhead. We conduct the DSE first by considering the best case with negligible interconnection overhead. Second, we conduct another DSE by assuming the overhead percentage provided in [14]. It depends on the number of cores of the considered system. Then for a system with only one core, there is no interconnection overhead. For a system netween 2 and 4 cores (resp. between 5 and 8), the interconnection overhead on each task corresponds to 10% (resp. 13%) of its capacity. For systems with more than 8 cores, we assume a 26% overhead. Each DSE was performed for 20000 iterations which takes 12 hours.

The solution with minimum cores corresponds to the solution with all tasks assigned to a single core. It has a high number of missed deadlines (45 over 57 tasks). By increasing the number of cores to 57 cores (i.e. number of tasks), more tasks are able to meet their deadlines (e.g. from 45 to 0 missed deadlines when inter-core communication is considered negligible). This confirms the impact of multicore platforms on safe and secure TSP systems. This is explained by the fact that using more cores increases the computation capacity of the system.

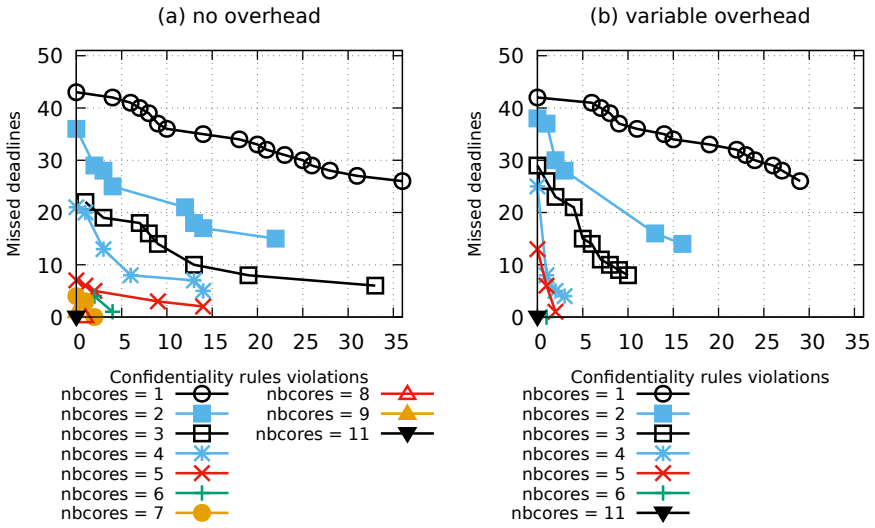

Fig. 2: Schedulability vs. confidentiality with inter-core communication overhead variation

Since these solutions are extreme, we propose to investigate the design search space, in order to find interesting trade-offs. The DSE proposes a set of 52 (resp. 40) different trade-offs with no interconnection overhead (with interconnection overhead). Fig. 2 shows the set of non-dominated solutions.

For the system model with negligible interconnection overhead, our DSE tool was able to decrease from 45 to 11 the number of cores required for a fully secured, safe, and no missed deadlines solution (Fig 2(a)). Our DSE is able to detect a minimal number of cores that corresponds to a fully secured, safe, and no missed deadlines solution. This solution considers a better grouping of tasks on the cores in order to propose a reduced number of cores while not allowing any task to miss its deadline. The tasks to cores assignment of this solution is so irregular that it will be very time-consuming and almost impossible to get manually such an assignment considering 57 tasks to assign to 11 cores. This shows the effectiveness of our DSE in proposing good trade-offs while considering schedulability, security, safety, and different numbers of cores.

The DSE with non-negligible interconnection overhead also proposes a solution that reduces to 11 the number of cores for a fully secured, safe, and no missed deadlines solution(Fig 2(b)). Contrary to the above-mentioned solution, it has a different tasks to cores assignment and used 4 partitions instead of 3 partitions. This can be explained by the fact that the above-mentioned solution updated with interconnection overhead was not able to keep no missed deadline. Then the DSE was able to explore different tasks to partitions and tasks to cores assignments to avoid missed deadlines without using more cores. Those solutions are not intuitive and motivate the use of our DSE approach.

As expected, we observe in the figures that the more the number of cores increases, the easier it becomes to ensure the security of safe TSP systems while minimizing the number of missed deadlines. This confirms the relevance of the proposed DSE.

The speedup relates to the number of cores can be compromised by a high shared hardware resource overhead. As shown on the graphs, trade-offs with no security vulnerabilities proposed by the DSE with non-negligible interconnection overhead have a number of missed deadlines greater than or equal to the equivalent in the DSE with negligible interconnection overhead. Let us consider the fully secure solutions with 5 cores. With no interconnection overhead, there are 7 missed deadlines (Fig 2(a)) while there are 13 missed deadlines when considering interconnection overhead (Fig 2(b)). This can also explain that with interconnection overhead, solutions with 6, 7, 8, and 9 cores are dominated by the other solutions and then rejected by the PAES. This illustrates that overhead related to shared hardware resources is a key parameter in the design of safe and secure multicore TSP systems.

No optimal result is known, since the search space size is in the order of 10^{146}. Thus an exhaustive search should be no tractable. Comparison could be made with other heuristic approaches only, without a guarantee on optimality.

5 Related work

In this section, we position our work by presenting different approaches that addressed the design of multicore platforms for TSP systems with schedulability, safety, and/or security constraints/objectives.

Many researchers have investigated TSP systems on multicore platforms. In [21], the authors depicted that multicore platforms can intervene in ensuring high-performance requirements. For this purpose, they identified some conditions such as privileging the intra-partition parallelism, which assumes the possibility of running parallel tasks of the same partition on different cores. In [7], the authors proposed the evolution of a TSP unicore system to a TSP multicore system while considering inter and intra-partition parallelism mechanisms. The former consists of activating simultaneously many partitions on different cores. The work in [12] proposed a similar idea towards the design and analysis of TSP multicore systems. The authors specifically focused on symmetric multiprocessing (SMP) architectures where each core has access to a common shared memory and I/O resources with a single operating system for all the cores. They defined patterns for SMP/TSP multicore systems with which they extended the Ocarina code generation tool.

Since safety and security are important requirements for TSP systems, several researchers showed interest in these domains. In [6], the authors proposed a survey for validation and certification of TSP multicore systems deployed on the Xtratum hypervisor [18]. For example, it highlights fault tolerance for safety and data protection for security. The authors of [11] addressed multicore platforms not specifically for TSP systems, but for real-time systems in general. They also addressed the systems' security vulnerabilities. Then the authors added security mechanisms such as a hash algorithm to their systems and then proposed a DSE to optimize their schedulability while exploring the security tasks to cores assignment possibilities.

The potential schedulability benefits of deploying TSP systems on multicore platforms have led to multiple researches on the design and analysis of such systems. Some have addressed their security and safety vulnerabilities. Few have studied the assignment of tasks to cores through a DSE for real-time systems in general. We propose a DSE approach for multicore TSP systems that investigates not only tasks to cores assignment but also tasks to partitions assignment and securing communications alternatives in order to find trade-offs. We also integrate safety constraints into our proposal. As far as we know, no work has proposed such a set of combinations.

6 Conclusion

In this paper, we investigate the impact of multicore platforms on safe and secure TSP systems by proposing an approach to explore their design space. Our DSE approach covers the different possibilities of tasks to partitions assignment, tasks to cores assignment, and securing communications, which is a combinatorial problem. Our approach is based on a meta-heuristic which proposes trade-offs between schedulability and security for a safe TSP system while considering different numbers of cores.

As expected, our approach shows that for a safe and secure TSP system with some missed deadlines, increasing the number of cores effectively helps to

optimize the system schedulability. Better solutions can also be obtained by moving some tasks from one partition to another or from a core to another. This first result confirms the relevance of our DSE.

To illustrate the interest of our approach, we test the DSE by considering shared hardware resources overhead existing in multicore platform. This overhead results from tasks on different cores accessing simultaneously the same hardware resources. It may increase considerably the required number of cores to keep a certain level of schedulability. Our experiments show that the shared hardware resources overhead, the number of cores, the number of partitions, tasks to partitions and cores assignments are key parameters in the design of multicore safe and secure TSP systems.

In this paper, the multicore hardware resource overheads introduced in the DSE was limited while it exists various sources and types of such overhead in multicore platforms (from the cache, interconnection or memory systems). In the future, we want to integrate in the DSE some of these overheads. We also intend to consider other security vulnerabilities such as those related to integrity or/and availability since we only consider confidentiality vulnerabilities.

Artefact All experiment data presented in this paper are available at `http://beru.univ-brest.fr/svn/CHEDDAR/trunk/artefacts/DECSOS22/`.
Programs and scripts written to produce these experimental data are available at `http://beru.univ-brest.fr/svn/CHEDDAR/trunk/src/framework_examples/architecture_exploration_tools/`.

References

1. Bell, D.E., La Padula, L.J.: Secure computer system: Unified exposition and multics interpretation. Tech. rep., MITRE CORP BEDFORD MA (1976)
2. Bieber, P., Boniol, F., Boyer, M., Noulard, E., Pagetti, C.: New challenges for future avionic architectures. AerospaceLab (4), p–1 (2012)
3. Chai, L., Gao, Q., Panda, D.K.: Understanding the impact of multi-core architecture in cluster computing: A case study with intel dual-core system. In: Seventh IEEE international symposium on cluster computing and the grid (CCGrid'07). pp. 471–478. IEEE (2007)
4. Coello, C.A.C., Lamont, G.B., Van Veldhuizen, D.A., et al.: Evolutionary algorithms for solving multi-objective problems, vol. 5. Springer (2007)
5. Committee, A.E.E.: Arinc 653: Avionics application software standard interface, supplement 1 (2003)
6. Coronel, J., Tsagkaropoulos, M., Mylonas, D., Balbastre, P., Kollias, V., Crespo, A.: Validation of securely partitioned systems over multicore architectures based on xtratum. In: Data systems in aerospace (DASIA), Proceedings on (2013)
7. Craveiro, J., Rufino, J., Singhoff, F.: Architecture, mechanisms and scheduling analysis tool for multicore time-and space-partitioned systems. ACM SIGBED Review **8**(3), 23–27 (2011)
8. Dai, W.: Crypto++ 5.6. 0 benchmarks. http://www. cryptopp. com/benchmarks. html (2009)

9. (Firme), R.: Integrated Modular Avionics (IMA) Development Guidance and Certification Considerations. RTCA (2005)
10. Garside, R., Pighetti, F.J.: Integrating modular avionics: A new role emerges. IEEE Aerospace and Electronic Systems Magazine **24**(3), 31–34 (2009)
11. Hasan, M., Mohan, S., Pellizzoni, R., Bobba, R.B.: A design-space exploration for allocating security tasks in multicore real-time systems. In: 2018 Design, Automation & Test in Europe Conference (DATE). pp. 225–230. IEEE (2018)
12. Hugues, J., Honvault, C., Pagetti, C.: Model-based design, analysis and synthesis for multi-core and tsp avionics targets (2018)
13. Knowles, J., Corne, D.: The pareto archived evolution strategy: A new baseline algorithm for pareto multiobjective optimisation. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406). vol. 1, pp. 98–105. IEEE (1999)
14. Kumar, R., Zyuban, V., Tullsen, D.M.: Interconnections in multi-core architectures: Understanding mechanisms, overheads and scaling. In: 32nd International Symposium on Computer Architecture (ISCA'05). pp. 408–419. IEEE (2005)
15. Lala, J.H., Harper, R.E.: Architectural principles for safety-critical real-time applications. Proceedings of the IEEE **82**(1), 25–40 (1994)
16. Gohring de Magalhaes, F., Torres Aurora Dugo, A., Lefoul, J.B., Nicolescu, G.: On the benchmarking of partitioned real-time systems. arXiv e-prints pp. arXiv–2007 (2020)
17. de Magalhaes, F.G., Dugo, A.T.A., Lefoul, J.B., Nicolescu, G.: On the benchmarking of partitioned real-time systems. arXiv preprint arXiv:2007.10794 (2020)
18. Masmano, M., Ripoll, I., Crespo, A., Metge, J.: Xtratum: a hypervisor for safety critical embedded systems. In: 11th Real-Time Linux Workshop. pp. 263–272. Citeseer (2009)
19. Mo, Y., Garone, E., Casavola, A., Sinopoli, B.: False data injection attacks against state estimation in wireless sensor networks. In: 49th IEEE Conference on Decision and Control (CDC). pp. 5967–5972. IEEE (2010)
20. Pagetti, C., Saussié, D., Gratia, R., Noulard, E., Siron, P.: The rosace case study: From simulink specification to multi/many-core execution. In: 2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS). pp. 309–318. IEEE (2014)
21. Patte, M., Lefftz, V., Zulianello, M., Crespo, A., Masmano, M., Coronel, J.: System impact of distributed multi core systems. Technical Report ESTEC Contract 4200023100 (2011)
22. Rouxel, B., Puaut, I.: Str2rts: Refactored streamit benchmarks into statically analyzable parallel benchmarks for wcet estimation & real-time scheduling. In: 17th International Workshop on Worst-Case Execution Time Analysis (WCET 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2017)
23. Schneier, B.: Description of a new variable-length key, 64-bit block cipher (blowfish). In: International Workshop on Fast Software Encryption. pp. 191–204. Springer (1993)
24. Singhoff, F., Legrand, J., Nana, L., Marcé, L.: Cheddar: a flexible real time scheduling framework. In: ACM SIGAda Ada Letters. vol. 24, pp. 1–8. ACM (2004)
25. Watkins, C.B., Walter, R.: Transitioning from federated avionics architectures to integrated modular avionics. In: 2007 IEEE/AIAA 26th Digital Avionics Systems Conference. pp. 2–A. IEEE (2007)
26. Zou, Y., Wang, G.: Intercept behavior analysis of industrial wireless sensor networks in the presence of eavesdropping attack. IEEE Transactions on Industrial Informatics **12**(2), 780–787 (2015)