RTOS for mixed criticality applications deployed on NoC-based COTS MPSoC

RTOS for Mixed Criticality applications deployed on NoC-based COTS MPSoC

Stefano Esposito, Serhiy Avramenko, Massimo Violante
DAUIN
Politecnico di Torino
Torino, Italy
{stefano.esposito, serhiy.avramenko, massimo.violante}@polito.it

***Abstract*—Network-on-chip-based multi-processor systems-on-chip are clearly the trend for next generation embedded applications. In order to enable use of such systems in a mixed-criticality context such as an avionic application, spatial and temporal partitioning should be provided between safety-critical tasks and non-safety-critical tasks. The contribution of this work is to propose a software-only partitioning scheme that exploits the routing algorithm used by the network-on-chip. The proposed approach implements the partitioning on traffic route level, overcoming the concept of strict segregation between critical and non-critical domains. As a further contribution this paper describes a software module we developed to implement the proposed partitioning scheme, thus enabling the use of commercial-off-the-shelf components in a safety critical context. The proposed module is intended to be added to a real-time operating system, to satisfy portability and certification requirements.**

Keywords—network-on-chip, mixed criticality, commercial-off-the-shelf, real-time operating system

I. Introduction

The new performance requirements of mission and safety critical system demand use of new architectures to be fulfilled. Even though technologies able to satisfy such performance requirements do exist on the market, there are other requirements that make their use still not widespread in some industries. In particular, the requirement of a high reliability is imperative for industries involved in development of safety-critical applications, like aerospace, automotive, railways and signaling, etc. Applications like the European Space Agency's global astrometric interferometer for astrophysics (GAIA) mission [1] could have benefitted from the amount of computing power available in multi- and many-core architectures. These architectures, could allow to integrate functionalities now deployed on several computing units into a single computer, provided that there are architectures able to withstand the harshness of the space environment.

Electronics systems used on board aircrafts – commonly referred to as avionics – are another example of the complexity and performance required to modern embedded systems. The most used design pattern for avionic systems is the *federated architecture* [2]. In this design pattern, each functionality or tightly coupled set of functionalities is deployed on a single on-board equipment (OBE) – i.e., a computer – isolated from all other OBEs, as far as memory and input/output are concerned. However, the federated architecture reached its limit with the increase in functionalities required of modern avionics system, hitting a wall in terms of space, weight, and power (SWaP) consumption. The integrated modular avionic (IMA) architecture [3] and the ARINC-653 application programming interface (API) standard [4] were developed to overcome such limitations, allowing sharing of OBEs among different set of functionalities in a time-division multiplexing (TDM) fashion. Use of multi- or many-core systems could help further increase the resource sharing among different applications, thus reducing SWaP. The lack of an established certifiable architecture forces the deployment of IMA-based systems on single-core processors, or forces designers to use multi-core processors as single-core processors [5][6].

This paper is focused on the current trend in multi-processor system-on-chip (MPSoC) architectures, which are relying more and more on a network-on-chip (NoC) interconnect to integrate increasing numbers of processing cores and peripherals on the same chip. The NoC interconnect was born to overcome limitations of traditional interconnects, like bus and crossbars [7]. Although NoC interconnects have a higher scalability than traditional interconnect, they introduce a further level of arbitration in the system architecture, thus increasing complexity in the analysis of such systems. Industries that must certify their products adherence to rigorous standards like those described in [8][9][10], have been so far unable to fully exploit the potential of NoC-based systems.

In this paper, we propose a method to consolidate mixed-criticality applications (MCAs) on commercial-off-the-shelf (COTS) NoC-based MPSoCs. This method is purely software and relies on the integration of a dedicated module in a real-time operating system (RTOS), which should be certified once and could be then reused for new products. The module is in charge of enforcing a spatial and temporal partitioning scheme that is conceptually derived from [11] and [12], although it has been adapted to the different underlying architecture.

This paper is organized as follows. Section II presents a brief survey of the literature on mixed-criticality applications on both bus-based and NoC-based MPSoCs; Section III describes the proposed spatial and temporal partitioning approach, and its implementation in a software module to be integrated in a RTOS; Section IV presents experimental evaluation of the

978-1-5386-1472-3/18/$31.00 ©2018 IEEE

proposed module; Section V draws some conclusions and outlines future works.

II. Mixed-Criticality Applications on MPSoCs

This section offers a brief survey of solutions for MCAs on bus-based COTS MPSoCs [13][14] and we identify what is likely to be the solutions for the NoC-based COTS MPSoC.

A. Spatial and temporal partitioning

An important concept for MCAs is the partitioning, defined as "appropriate hardware and software mechanisms to restore strong fault containment" [15]; in the literature, there are two types of partitioning:

1. *Spatial partitioning* is the separation of resources among different applications. It can be enforced through software [16][17].
2. *Temporal partitioning* must ensure that there is no effect on the execution time of a task when other tasks are running on the same MPSoC; however, applications deployed on MPSoCs must share core resources, like the interconnect. A precise analysis of the effect of interferences in such a system would require knowing details about the microarchitecture of the MPSoCs which are often not shared by the manufacturers.

B. Temporal-partitioning-specific solutions

Several works have been proposed on the topic of temporal partitioning [13]. The first class of solutions, time multiplexing solutions, use static scheduling to avoid interference among tasks [18][19]. This is implemented by partitioning applications in local phases – during which a core runs code using just local data – and memory access phases – during which the application commits produced data and reads data for the next computation phase. Other approaches may add stricter requirements to the local phase, like forbidding interrupt service routines (ISRs) execution [20]. However, all time multiplexing approaches introduce long idle times, lowering the overall utilization.

The second class of solutions, bounded interference solutions, are focused on a more precise worst-case execution time (WCET) analysis in the case of a multi-core system. The interference sensitive WCET (isWCET) [21][22], considers the number of cores that are accessing a shared resource at the same time to evaluate the access latency to be considered in the WCET estimation. The use of performance counter can also introduce fault tolerance based on on-line detection and recovery [23].

C. System scheduling, partitioning and other solutions

Besides partitioning, another key aspect of mixed-criticality systems is the system scheduling. The problem was first formulated in [24], which also introduced a first solution. The solution was proposed for single-core systems and was later extended for multi-core systems [25][26][27]. More recently, the focus moved from bus-based multi-core systems to NoC-based many-cores. The scheduling problem, in this context, moved from tasks to flows as in [28][29][30][31]. Partitioning schemes were also proposed for NoC-based systems, by extending network interfaces by adding a dedicated channel for high-criticality traffic, to implement a time-triggered scheme of communication [32]. This solution grants a bounded latency on high-criticality traffic by using a contention-free channel. The rest of the traffic is scheduled to happen without interfering with the high-criticality traffic. Different solutions are based on the concept of quality-of-service (QoS) [33][34], or use hardware-enforced segregation between a safety-critical domain and a non-safety-critical domain [35].

D. Future of mixed criticality on NoC-based COTS MPSoCs

Although there are many solutions proposed, most rely on a special hardware or NoC architecture to be implemented. However, it is likely that, as it was for bus-based COTS MPSoCs, also NoC-based COTS MPSoCs will be optimized for the average, generic workload use-case which is far from the requirements of safety-critical applications. In bus-based MPSoC, a set of solutions to work around the non-determinism of their behavior had to be developed; predicting that the same kind of solutions will be in the end needed for NoC-based COTS MPSoCs, this paper proposes a first step towards a complete solution.

III. Proposed Solution

The proposed solution solves the temporal and spatial partitioning, as described in Section II, resorting to software means only – without any dedicated hardware support needed. This section will describe the proposed partitioning approach and the underlying NoC model characteristics required for the application of such approach. The software module we developed to implement the proposed approach is then described in detail.

A. Proposed partitioning scheme

In order to explain the proposed approach and its implementation, we consider a simple NoC, featuring a single physical network. Nodes are addressed through a dedicated memory map. Each node has its own memory map, including both local and remote resources. Local resources are accessed as usual through the local bus, whereas remote resources are forwarded on the NoC through the local network interface (NI). The NI is a peripheral on the local bus. It receives requests from the local processing element (PE) according to the protocol defined on the local bus. The PE can then wait for a response (bus-transaction (BT) model) or it can continue its computations, pending an interrupt that signals that requested data is available at a predefined location on a local memory (direct memory access (DMA) model). The NI performs the following tasks:

1. Receives requests from the local PE to remote resources, transforms them in NoC packets, and injects packets inside the NoC.
2. Receives responses from remote resources and sends them to the local PE according to the BT model or to the DMA model.
3. Receives requests from remote nodes and forwards them on the local bus to the local PE. The local PE

should respond to such requests according to the BT model or to the DMA model.

The NI has a look-up table to translate each global resource's address from the local memory map to a set of coordinates that identify the position of that resource on the NoC.

The PE in this NoC model is either a passive slave or a processor running software which includes the module that implements the proposed approach. The proposed module acts as a driver for the NoC interconnect, supervising and eventually filtering software requests for the NI.

The partitioning scheme we propose consists in segregating the non-critical traffic outside the critical traffic paths. This approach overcomes the classic strict domain segregation scheme as it allows a node to be simultaneously traversed by both critical and non-critical traffic. The proposed scheme relies on the system design to be done so that non-critical traffic paths are constrained to never overlap any critical traffic path. To enforce the segregation of the non-critical traffic, the proposed approach consists in locally filtering all the non-critical traffic that would violate the no-overlap constraint – due to a software fault (bug) active in one or more non-critical applications. The critical traffic is not filtered because the design of critical applications is supposed to grant absence of bugs [10]. In detail, the idea behind the proposed approach is to exploit the knowledge of a deterministic routing algorithm used by the NoC; since deterministic routing algorithms assure that the path of a packet is completely specified by coordinates of source and target node pairs. Considering a NoC consisting of a single physical network (which is the simplest architecture possible), the following requirements must be satisfied:

1. the NoC should use a deterministic routing algorithm;
2. the routing algorithm should be known;
3. the topology of the NoC should be known.
4. the topology of the NoC should grant existence of at least one non-critical traffic route which does not overlap any critical traffic route; i.e., it has to exist a traffic route for the non-critical traffic which does not "physically overlap" any critical traffic route.

It is worth to notice that the proposed approach is not limited neither to XY routing, nor to mesh topology, nor to 2D topology. Instead it can be applied to any deterministic routing algorithm and any topology, as far as requirements listed above are satisfied.

B. Proposed Software Module

The proposed partitioning scheme has been implemented as a software module to be added to a real-time operating system (RTOS) running on a NoC-based MPSoC. Knowing both the routing algorithm and the topology of the network, the designer can configure the proposed software module to implement the proposed filtering approach. The proposed module oversees and manages communications that a given non-critical node intends to inject in the NoC. In particular, its main goal is to filter all the traffic flows starting from a non-critical node which would overlap with any path used by critical traffic. The former condition is detected by only considering the destination of a packet. The module is not present on the critical nodes thus it does not deteriorate their responsiveness. As the proposed module runs locally and it does not need any synchronization with other modules on the NoC, therefore it does not add any communication overhead. The critical region is the union of the paths traversed by the traffic generated by critical nodes, and it is routing algorithm dependent. In Fig. 1 there is an example of definition of a critical region in a 2D 4x4 mesh topology using an XY routing. As already described in the previous subsection, the proposed approach does not restricts a non-critical node located inside the critical region to produce non-critical traffic, thus overcoming the strict partitioning between critical and non-critical regions. The proposed module should be integrated in a certified RTOS, to be reusable in several applications without submitting it to a new certification process in each new application. To be certifiable, it should be developed according to recommendations contained in [10]. The level of assurance should be selected according to the nature of the application, however, to be as general as possible, the module should be certified at the maximum level (DAL-A). Considering the effort for certification of a DAL-A module and the need for a low performance overhead, the implemented algorithm should be as simple as possible.

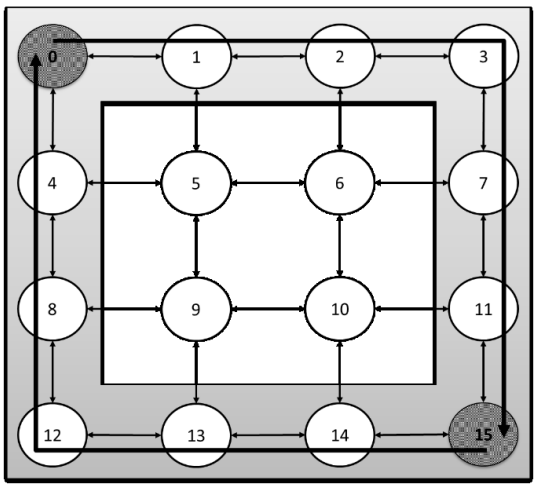

Fig. 1. 4x4 mesh topology with two critical nodes and the critical traffic paths (gray shade) defined by the XY routing path.

Algorithm 1 describes the core behavior of the proposed module. It implements the main functionality of traffic filtering according to a configured destinations map. The destination map is computed off-line according to the routing algorithm, the topology and the position of critical nodes. The rules to be considered when defining the destination map are:

1. Any traffic initiated by a critical node is critical traffic.
2. Any node which is destination of critical traffic is a critical node.
3. Nodes in a critical region can forward non-critical traffic if and only if it does not interfere with critical traffic, i.e. it does not use the same resources as critical traffic.
4. Non-critical nodes within the critical region can only communicate with nodes that are reachable through a

path that does not interfere with the critical traffic, as described in rule 3.

Algorithm 1. Traffic filter

```
Input: dst_addr, payload;
Data: dst_map

if dst_map[dst_addr] == OK_TRANS then
        forward packet to NI
else
        drop packet and return error code
end if;
```

Applying these rules to the node 1 of the NoC configuration in Fig. 1, yields the destination map in TABLE I. It can be observed that the node 1 can only communicate with three other nodes, due to limitations imposed by its position within the critical region and the XY routing. This problem can be solved by adding a redirection functionality besides the traffic filter described in Algorithm 1. This redirection functionality should act after the traffic filter, and should use a second field in the destination map, as showed in reported destination maps. Such field is used as described in Algorithm 2 which extends Algorithm 1.

Algorithm 2. Traffic filter with redirection

```
Input: dst_addr, payload;
Data: dst_map

if dst_map[dst_addr].OK_TRANS then
        forward packet to NI
else
        if dst_map[dst_addr].RDR > -1 then
                generate redirection packet
                forward packet to NI
        end if;
end if;
```

The redirection packet is a special packet to be forwarded to the node indicated in the destination map. The information on the original destination is contained in the payload of the redirection packet. When a node receives a packet, the software running on the node should check if the received packet is a redirection packet and forward it to the actual destination as described in Algorithm 3. Redirection is an optional feature which is only useful to improve reachability in a NoC. Using redirection has no impact on critical traffic, but greatly increases performance of non-critical nodes, allowing integration of additional applications.

IV. Experimental Evaluation

Experimental evaluation has been performed using a simulation environment considering NoC model using logic-based distributed routing (LBDR) [36] architecture. The considered NoC model is derived from the baseline NoC presented in [37] and has 2D mesh topology, based on one physical network (no virtual network is present). This is a good model for the purpose at hand, although real-life NoCs in COTS systems are more complex and uses different routing approaches. The LDBR router is configured to implement an XY routing algorithm, which is a fairly common, deadlock-free routing algorithm.

TABLE I. Destination map for node 1

DESTINATION	OK TO TRANSMIT	REDIRECTION
0	FALSE	-1
2	FALSE	5
3	FALSE	-1
4	FALSE	5
5	TRUE	-
6	FALSE	5
7	FALSE	5
8	FALSE	9
9	TRUE	-
10	FALSE	5
11	FALSE	9
12	FALSE	-1
13	TRUE	-
14	FALSE	5
15	FALSE	-1

Algorithm 3. Redirection packet check

```
Input: packet

if packet.redirection then
        traffic_filter(packet.actual_dst,
packet.actual_payload)
else
        forward to application software
end if
```

The experiments have been done considering the scenario described in Fig. 1. For simplicity sake, two critical nodes are considered. There are no shared resources between the critical and non-critical nodes, i.e. a critical node cannot be destination of non-critical traffic. Destinations of non-critical traffic were been selected at random. Experiments were performed considering several packet injection rates (PIRs), defined as the number of packets injected in the NoC at each clock cycle by any given node. For each considered PIR, 10,000 packets were injected by each node in the NoC.

The baseline scenario was first considered, i.e., the scenario described in Fig. 1 was simulated without the proposed partitioned scheme. In this scenario we also programmed the non-critical nodes to generate traffic overlapping critical traffic, simulating non-critical nodes affected by a bug. The experimental results, presented in TABLE II. exhibits the uncertainty in the communication latency among the two critical nodes (0 and 15), as the non-critical traffic introduces congestion on the critical traffic path.

Once implemented, the proposed software creates a reserved channel as shown in Fig. 1. This reserved channel has the effect to cancel any variance in the communication between the critical nodes, as shown by an additional experimental campaign, so the latency is always 305 ns. Besides cancelling the latency variance, thus allowing an easy and provable WCET estimation for the critical applications, the proposed approach, without the redirection module, has the side effect of significantly reduce the overall reachability. Here the reachability of a node i is defined as the number of nodes to which the i node is able to send a packet. This metric must not be confused with the connectivity, which is usually defined as number of nodes to which a given node is physically connected. In the reachability computation, critical nodes are not considered: by design they should only communicate with each other, since – by definition – any traffic

initiated by a critical node is critical, thus this communication would require definition of a new critical region.

TABLE II. CRITICAL TRAFFIC LATENCY (NS): BASELINE SETUP

PIR	MAX	MEAN	STD.DEV.
0.02	815	334.8	55.6
0.01	665	315.1	30.4
0.007	655	311.6	24.4

Reachability figures, without redirection module being implemented, are reported in TABLE III. The redirection extension is implemented with the purpose of incrementing the reachability for non-critical nodes, in order to have a more usable system while keeping warranties on the ability of the network of sustaining a critical traffic with deterministic latency. Implementation of the redirection feature, allows to obtain an almost perfect reachability, rising the average reachability to 10.3. It is worth to notice that with the redirection almost all non-critical nodes can reach all other non-critical nodes, while the reachability remains 0 for all the nodes that are only connected to nodes within the critical region. Such nodes cannot communicate with no other node and represent the cost of the proposed partitioning scheme. Considering the configuration of Fig. 1, such nodes are node 3 and node 12. The cost of this reachability improvement is some additional communication latency for non-critical nodes, due to the intervention of the software module during transmission.

A further use case is presented in Fig. 2. This configuration, features two separated non-critical regions. However, the proposed approach allows the two regions to communicate with each other (even without redirection) as it does not imply strict critical and non-critical domain partitioning. Without the redirection module, the average reachability in this configuration would be of 6.4, which is higher with respect to the scenario of Fig. 1, since the critical region is smaller and only the node 12 is unreachable. Using the redirection module, the reachability increase up to 12.1.

As it can be observed from the two considered scenarios, the position of the critical nodes has a huge impact on the non-critical nodes' metrics in term of reachability, and in particular the number of those nodes that are not able to communicate through the NoC. This suggests that a practical limitation of the proposed solution is the size critical region: a large critical region entails a large number of non-critical nodes unable to use the NoC. In practice this means either that the number of critical nodes must be much lower with respect to the number of non-critical nodes, or that the critical nodes should be mapped to reduce the size of the critical region.

V. CONCLUSIONS

This paper describes the use of software means to obtain a NoC configuration able to sustain an MCA. The main requirement in such applications is to provide temporal and special partitioning. As the critical applications have to fulfill the RT requirements, reducing or cancelling the uncertainty in communication latency for traffic belonging to a critical application is also a requirement for MCAs. In the scope of this paper we considered the case of an avionic application, with the relevant standard being the RTCA DO-178C [10] and recommendations contained in [5][6] and similar documents.

TABLE III. REACHABILITY FOR NON-CRITICAL NODES: NO REDIRECTION

NODE ID	REACHABLE NODES	REACHABILITY
1	5, 9, 13	3
2	6, 10, 14	3
3	-	0
4	1, 2, 5, 6, 7, 9, 10, 13, 14	9
5	1, 2, 4, 6, 7, 9, 10, 13, 14	9
6	1, 2, 4, 5, 7, 9, 10, 13, 14	9
7	1, 2, 4, 5, 6, 9, 10, 13, 14	9
8	1, 2, 5, 6, 9, 10, 11, 13, 14	9
9	1, 2, 5, 6, 8, 10, 11, 13, 14	9
10	1, 2, 5, 6, 8, 9, 11, 13, 14	9
11	1, 2, 5, 6, 8, 9, 10, 13, 14	9
12	-	0
13	1, 5, 9	3
14	2, 6, 10	3
AVERAGE	**N/A**	**6**

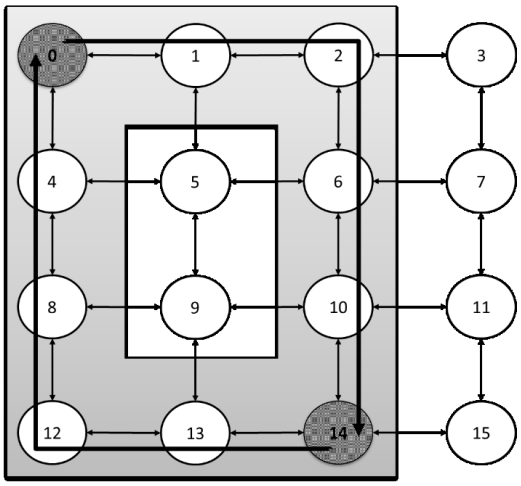

Fig. 2. 4x4 mesh NoC topology with critical nodes in 0 and 14.

The proposed partitioning scheme, based on deterministic routing, has been implemented as software module. The proposed module is considered to be part of an RTOS in order to optimize the certification effort. Being a module of an RTOS means that the module should be certified only once together with the containing RTOS and it could then be used in certifiable applications without any additional certification effort. The proposed module implements two functionalities:

1. traffic filter, to enforce an isolated critical route in which only traffic belonging to critical applications can flow;
2. redirection, to allow a non-critical node to reach more nodes, solving the limitations introduced by the traffic filter functionality. This can be considered as an optional functionality, providing a higher usability of the NoC.

It is conceivable that COTS systems would not be optimized for the MCA use-case, similarly to the case of bus-based COTS multi-core systems. Use of the proposed software module enables use of such systems in the MCA scenario. Experimental results show that the latency determinism is granted thanks to the traffic filter module, and that the redirection module does not introduce any overhead for the critical application, while granting a better usability of the NoC-based system.

The technical limitations of the proposed approach are described in detail in Section III (e.g., the routing algorithm used by the NoC to be both deterministic and known). The practical limitation to be specifically addressed in future work, is the high number of non-critical nodes to be arbitrarily mapped, as briefly addressed in Section IV.

REFERENCES

[1] S. Provost, M. Le Roy, B. Mamdy, G. Flandin, and T. Paulsen, "GAIA video processing embedded algorithms: Prototyping and validation activities," *Eur. Sp. Agency, (Special Publ. ESA SP*, no. SP-638, 2007.

[2] C. B. Watkins and R. Walter, "Transitioning from federated avionics architectures to Integrated Modular Avionics," *AIAA/IEEE Digit. Avion. Syst. Conf. - Proc.*, pp. 1–10

[3] P. J. Prisaznuk, "Integrated modular avionics," *Aerosp. Electron. Conf. 1992. NAECON 1992., Proc. IEEE 1992 Natl.*, pp. 39–45 vol.1, 1992.

[4] P. J. Prisaznuk, "Arinc 653 role in Integrated Modular avionics (IMA)," in *AIAA/IEEE Digital Avionics Systems Conference - Proceedings*, 2008.

[5] X. Jean, M. Gatti, G. Berthon, and M. Fumey, "MULCORS - Use of Multicore Processors in airborne systems," 2012.

[6] Position Paper CAST-32, Multi-core Processors, 2014.

[7] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference*, 2001, pp. 684–689

[8] Functional safety of electrical/electronic/programmable electronic safety-related systems, IEC-61508 Edition 2.0, http://www.iec.ch/functionalsafety/standards/page2.htm

[9] RTCA Inc., "DO-254 Design Assurance Guidance for Airborne Electronic Hardware", Issue date 4/19/2000.

[10] RTCA Inc., "DO-178c Software Considerations in Airborne Systems and Equipment Certification", Issue date 12/13/2011.

[11] S. Avramenko, S. Esposito, M. Violante, M. Sozzi, M. Traversone, M. Binello, and M. Terrone, "An Hybrid Architecture for Consolidating Mixed Criticality Applications on Multicore Systems," in *2015 IEEE 21st International On-Line Testing Symposium*, 2015, pp. 26–29

[12] S. Esposito, M. Violante, M. Sozzi, M. Terrone, and M. Traversone, "A novel method for online detection of faults affecting execution-time in multicore-based systems," *ACM Trans. Embed. Comput. Syst.*, vol. 16, no. 4, pp. 1–19

[13] M. Paulitsch, O. M. Duarte, H. Karray, K. Mueller, D. Muench, and J. Nowotsch, "Mixed-criticality embedded systems-A balance ensuring partitioning and performance," *Proc. - 18th Euromicro Conf. Digit. Syst. Des. DSD 2015*, pp. 453–461

[14] A. Burns and R. I. Davis, Mixed Criticality Systems - A Review, 7th edition. Univerisy of York, 2016.

[15] J. Rushby, "Partitioning in Avionics Architectures: Requirements, Mechanisms, and Assurance," NASA Langley Research Center, NASA CR-1999-209347

[16] S. Avramenko, S. Esposito, M. Violante, M. Sozzi, M. Traversone, M. Binello, and M. Terrone, "An Hybrid Architecture for Consolidating Mixed Criticality Applications on Multicore Systems," in *2015 IEEE 21st International On-Line Testing Symposium*, 2015, pp. 26–29

[17] S. Esposito and M. Violante, "On the Consolidation of Mixed Criticalities Applications on Multicore Architectures," *J. Electron. Test. Theory Appl.*, 2017, vol. 33, no. 65, pp. 65–76

[18] A. Schranzhofer, J.-J. Chen, and L. Thiele, "Timing predictability on multi-processor systems with shared resources," *Embed. Syst. Week-Workshop Reconciling Perform. with Predict.,* 2009, p. 89

[19] F. Boniol, H. Cassé, E. Noulard, and C. Pagetti, "Deterministic execution model on COTS hardware," *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7179 LNCS, pp. 98–110

[20] R. Pellizzoni, E. Betti, S. Bak, G. Yao, J. Criswell, M. Caccamo, and R. Kegley, "A predictable execution model for COTS-based embedded systems," *Real-Time Technol. Appl. - Proc.*, pp. 269–279

[21] J. Nowotsch, M. Paulitsch, D. Buhler, H. Theiling, S. Wegener, and M. Schmidt, "Multi-core interference-sensitive WCET analysis leveraging runtime resource capacity enforcement," *Proc. - Euromicro Conf. Real-Time Syst.*, pp. 109–118

[22] J. Nowotsch, M. Paulitsch, A. Henrichsen, W. Pongratz, and A. Schacht, "Monitoring and WCET analysis in COTS multi-core-SoC-based mixed-criticality systems," *Des. Autom. Test Eur. Conf. Exhib. (DATE), 2014*, pp. 1–5

[23] S. Esposito, M. Violante, M. Sozzi, M. Terrone, and M. Traversone, "A novel method for online detection of faults affecting execution-time in multicore-based systems," *ACM Trans. Embed. Comput. Syst.*, 2017, vol. 16, no. 4, pp. 1–19

[24] S. Vestal, "Preemptive scheduling of multi-criticality systems with varying degrees of execution time assurance," *Proc. - Real-Time Syst. Symp.*, pp. 239–243

[25] J. Anderson, S. Baruah, and B. Brandenburg, "Multicore operating-system support for mixed criticality," in *Workshop on Mixed Criticality: Roadmap to Evolving UAV Certification*, 2009.

[26] G. Giannopoulou, N. Stoimenov, P. Huang, and L. Thiele, "Scheduling of mixed-criticality applications on resource-sharing multicore systems," *2013 Proc. Int. Conf. Embed. Software, EMSOFT 2013*

[27] G. Giannopoulou, N. Stoimenov, P. Huang, and L. Thiele, "Mapping mixed-criticality applications on multi-core architectures," *Des. Autom. Test Eur. Conf. Exhib. (DATE),* 2014, pp. 1–6

[28] Q. Xiong, F. Wu, Z. Lu, and C. Xie, "Extending Real-Time Analysis for Wormhole NoCs," *IEEE Trans. Comput.*, vol. 66, no. 9, pp. 1532–1546

[29] A. Burns, J. Harbin, and L. S. Indrusiak, "A wormhole NoC protocol for mixed criticality systems," in *2014 IEEE Real-Time Systems Symposium*, 2014, pp. 184–195

[30] L. S. Indrusiak, J. Harbin, and A. Burns, "Average and Worst-Case Latency Improvements in Mixed-Criticality Wormhole Networks-on-Chip," in *Euromicro Conference on Real-Time Systems*, 2015, pp. 47–56

[31] G. Giannopoulou, N. Stoimenov, P. Huang, L. Thiele, and B. D. de Dinechin, "Mixed-criticality scheduling on cluster-based manycores with shared communication and storage resources," *Real-Time Syst.*, vol. 52, no. 4, pp. 399–449

[32] H. Ahmadian and R. Obermaisser, "Time-triggered extension layer for on-chip network interfaces in mixed-criticality systems," *Proc. - 18th Euromicro Conf. Digit. Syst. Des. DSD 2015*, pp. 693–699

[33] K. Goossens, J. Dielissen, and A. Radulescu, "Æthereal Network on Chip:Concepts, Architectures, and Implementations," *IEEE Des. Test Comput.*, vol. 22, no. 5, pp. 414–421.

[34] T. Marescaux and H. Corporaal, "Introducing the SuperGT network-on-chip," in *Design Automation Conference*, 2007, pp. 116–121.

[35] T. Hollstein, S. P. Azad, T. Kogge, and B. Niazmand, "Mixed-criticality NoC partitioning based on the NoCDepend dependability technique," in *10th International Symposium on Reconfigurable and Communication-centric Systems-on-Chip, ReCoSoC 2015.*

[36] J. Flich and J. Duarte, "Logic-Based Distributed Routing for NoCs," *IEEE Comput. Archit. Lett.*, vol. 7, no. 1, pp. 13–16.

[37] S. P. Azad, B. Niazmand, K. Janson, N. George, and A. S. Oyeniran, "From Online Fault Detection to Fault Management in Network-on-Chips : A Ground-Up Approach," in *Design and Diagnostics of Electronic Circuits & Systems (DDECS), 2017 IEEE 20th International Symposium on*