DAS: An Efficient NoC Router for Mixed-Criticality Real-Time Systems

Mourad Dridi*, Stéphane Rubini*, Mounir Lallali*, Martha Johanna Sepúlveda Flórez ‡, Frank Singhoff*, Jean-Philippe Diguet†
*Lab-STICC, CNRS, UMR6285, Univ. Bretagne Occidentale, 29200 Brest, France
†Lab-STICC, CNRS, UMR6285, Univ. Bretagne Sud, 56100 Lorient, France
‡ Institute for Security in Information Technology, Technical University of Munich, Germany
{mourad.dridi, stephane.rubini, mounir.lallali, frank.singhoff}@univ-brest.fr, johanna.sepulveda@tum.de, jean-philippe.diguet@univ-ubs.fr

***Abstract*—Mixed-Criticality Systems (MCS) are real-time systems characterized by two or more distinct levels of criticality. In MCS, it is imperative that high-critical flows meet their deadlines while low-critical flows can tolerate some delays. Sharing resources between flows in Network-On-Chip (NoC) can lead to different unpredictable latencies and subsequently complicate the implementation of MCS in many-core architectures. This paper proposes a new virtual channel router designed for MCS deployed over NoCs. The first objective of this router is to reduce the worst-case communication latency of high-critical flows. The second aim is to improve the network use rate and reduce the communication latency for low-critical flows. The proposed router, called DAS (Double Arbiter and Switching router), jointly uses *Wormhole* and *Store And Forward* techniques for low and high-critical flows respectively. Simulations with a cycle-accurate SystemC NoC simulator show that, with a 15% network use rate, the communication delay of high-critical flows is reduced by 80% while communication delay of low-critical flow is increased by 18% compared to usual solutions based on routers with multiple virtual channels.**

I. Introduction

Many-core architectures enable multiple software components to run at the same time on the same chip. These architectures also include a Network-On-Chips (NoC) to provide the required high communication bandwidth. So, the expected processing power allows designers to deploy many computation intensive functions. For example, in a drone application, designers will need to deploy a classical flight control software but also computation intensive functions such as image and video processing.

In the context of such systems, the criticality of these software components can be different. Burns and al. [1] define the criticality as *a designation of the level of assurance against failure needed for a system component*. This kind of systems, characterized by two or more distinct levels of criticality, is called Mixed Criticality Systems (MCS). In MCS, hard real-time and soft real-time applications share the same hardware platform and may exchange messages through a common communication infrastructure. Hard real-time applications such as longitudinal flight controller, have very stringent communication service requirements. It is imperative to meet deadlines otherwise the whole system might fail, leading to catastrophic results, like, loss of life or serious damage to the environment. In contrary, soft real-time applications such as video encoder, can tolerate some missed deadlines. So, in this paper, we will consider two criticality levels of communication flows: high-critical flows and low-critical flows.

This work and Cheddar (a GPL real-time scheduling analyzer) are supported by Brest Métropole, Ellidiss Technologies, CR de Bretagne, CG du Finistère and Campus France PESSOA programs number 27380SA and 37932TF.

We focus on the implementation of MCS on many-core systems. Network-On-Chips (NoCs) are mandatory in such architectures since they provide scalability, modularity, and communication parallelism. Applications with different levels of criticality exchange messages through the communication infrastructure. Sharing resources between flows can lead to different unpredictable latencies like direct interferences and indirect interferences [2]. Subsequently, it complicates the integration of MCS on NoCs.

Different types of NoC router architectures have been proposed in order to minimize delays of flows and to satisfy the timing requirements of hard and soft real-time applications. However, none of them can handle both high and low-critical flows. Virtual Channel (VC) routers reduce the latencies of flows and increase the network throughput [3]. However, VC routers are not suitable for MCS. While providing high throughput for low-critical flows, they lead to too pessimistic Worst Case Communication Time of high-critical flows [4].

In this paper, we introduce a new router called DAS router (Double Arbiter and Switching router), designed to efficiently run mixed-criticality applications on Network On-Chip. DAS router ensures the timing constraints for high-critical flows while limiting the bandwidth reservation for them. DAS router uses $N+1$ virtual channels: N VCs are devoted to the communication of the high-critical flows and the last VC to the low-critical flows. To enforce MCS requirements, DAS router implements automatic mode changes, two levels of preemption, two flow control techniques and two stages of arbitration.

In this work, we assume that high-critical flows are characterized by small packet sizes, while low-critical flows are characterized by larger packets. We also assume a task mapping which ensures that less than N critical flows share a given physical link in the NoC. The remainder of the paper is organized as follows. First, we provide an overview of MCS and VC routers. In section III, we detail the proposed router. Simulation results of our DAS router, compared to those of a VC router, are presented in section IV. Section V discusses related work. Section VI concludes the paper.

II. Background

In order to understand the solution we propose in sections III, this part presents the necessary background on mixed-criticality system and on NoC routers based on virtual channels.

A. Mixed-Criticality System (MCS)

Mixed-Criticality Systems are real-time systems characterized by two or more distinct levels of criticality. In a MCS, hard real-time

applications have very stringent communication service requirements. It is mandatory that all packets generated by a critical flow are delivered before or on their deadline even under the worst case scenarios, while soft real-time applications can tolerate some delays in the communication service. In this work, we will consider two criticality levels of communication flows: high-critical flows and low-critical flows.

B. Virtual Channel NoC Router

A NoC is a network of nodes that can be a processor, a memory, a peripheral or a cluster of nodes. The router is a key component in NoC. The parameters of the router are very important as they can modify the performance of the network. In this sequel, we give an overview of one particular type of router: the virtual channel router.

A virtual channel is an unidirectional logical connection between two nodes multiplexed with other virtual channels across the physical channel. The router is composed of input and output ports, a routing logic, a VC allocator, a switch allocator, and a crossbar [3].

VC allocator: The VC allocator assigns an available output channel to a new packet located in one of the input VC buffers. The allocation involves an arbitration between all packets requesting the same output channel. There exist several arbitration mechanisms. Round-robin and priority-based, are examples of these mechanisms. Round-robin arbiter gives the lowest priority to the last served request in the next arbitration while priority-based arbiter chooses one packet from many requests based on their priority [5].

Switch Allocator: It matches requests to output ports, and generates the required crossbar control signals. Moreover, the switching mode determines how a packet is allocated to buffers and channels and when it will receive service. For the *Store And Forward* mode (SAF), each switch waits for the full packet to arrive in a switch before sending it to the next router [6]. With SAF, buffer size and communication time depend on packet size. For the *Wormhole* mode, the packet is divided into a number of fixed size flits [7]. The packet is splited into a header flit, one or several body flits and a tail flit. The header flit stores the routing information and builds the route. As the header flit moves ahead along the selected path, the remaining flits follow in a pipeline way and possibly span a number of routers.

III. The proposed Router: Double arbiter and switching router (DAS Router)

The architecture of the DAS router is shown in Fig. 1. DAS router is composed of $N+1$ virtual channels, input and output arbitration units, a routing logic, a VC allocator, a switching allocator and a crossbar. It combines two switching modes: on each port, the router can use a wormhole or a Store-And-Forward (SAF) switching techniques depending on the criticality of the packets. In the sequel, we describe the use of $N+1$ virtual channels and we explain why we use SAF for high-critical flow and wormhole for low-critical flow. Then, we discuss the choice of N. Finally, we present the two stages of arbitration used in DAS router.

A. N+1 Virtual Channels

VCs 1 to N of the DAS router are dedicated to high-critical flows and VC $N+1$ is used for all low-critical flows. The N virtual channels of the high-critical flows use SAF switchings; Each is dedicated to only one given flow. The last VC dedicated to low-critical flows, is managed by a wormhole switching technique. VC $N+1$ can so be shared by several low-critical flows.

1) SAF for high-critical flows: High-critical flows are transmitted with a SAF policy and the preemption is managed at the packet level. In other words, a high-critical packet cannot be preempted by another flow. The main drawback of this policy is the input buffer cost that must be large enough to store the entire packet. We consider that this cost is limited in our context, since the analysis of real-life critical systems shows that high-critical flows are usually characterized by small packets size (e.g. sensor and control signals). For instance, in the ROSACE avionics benchmarks [8], it does not exceed 3 flits of 32 bits.

In a wormhole policy, a packet can be stored over multiple routers and then can occupy several physical links at a time, and consequently increases the potential congestion over the network. The links used by a packet are unavailable for other packets arriving into the router, and this additional blocking time makes difficult the computation of the WCCT [4]. On the contrary, using a SAF policy, each packet allocates only one link at a time and the congestion can be controlled without a prohibitive cost considering small high-critical packets.

SAF and packet-level preemption allow us to minimize the pessimistic degree of the computed WCCT. Moreover, by allocating one virtual channel to each high-critical flow, we can adapt the real-time scheduling analysis proposed in [2] to SAF policy and compute offline the WCCT for high-critical flows. As a result, SAF switching is intended to improve the schedulability of flows.

2) Wormhole for low-critical flows and Flit-level Preemption: Low-critical flows are transmitted with a wormhole policy. Wormhole is largely adopted in NoCs because it does not require large capacity buffers and, at the same time, it minimizes the communication latency.

We need to preempt a low-critical flow as soon as possible when a high-critical flow occurs in order to ensure predictable communication time and minimal interference delay for the later. So, the preemption is implemented at the flit level for the last virtual channel. In other words, high-critical flows, which use N first virtual channels, can preempt any low-critical flows at flit level.

3) Number of virtual channels: The number N of virtual channels allocated to high-critical flows depends on communication requirements of the software tasks, but also of the mapping of these tasks. On the one hand, for a fixed task mapping, we can choose N as the maximum of high-critical flows sharing the same link. We note that the higher the value of N, the larger the overhead of area for the NoC implementation. On the other hand, for a fixed N, we must choose a task mapping which allows us to have at most N high-critical flows sharing the same link. This kind of problems is similar to a Quadratic Assignment Problem (QAP) [9]. This is an NP-Hard optimization problem and there are several heuristics that can be used for defining such mapping as those described in [10] or [9]. These articles present multi-criteria heuristics to optimize the total communication volume and the number of virtual channels. This optimization issue is orthogonal to the scope of this paper.

B. The Two Stages of Arbitration

At each cycle, in the DAS router, only one virtual channel can advance from an input port, and only one virtual channel can be accepted by each output port. As shown in Fig. 1(b), DAS router implements input and output arbitration units in order to solve these problems. In the sequel, we describe these units.

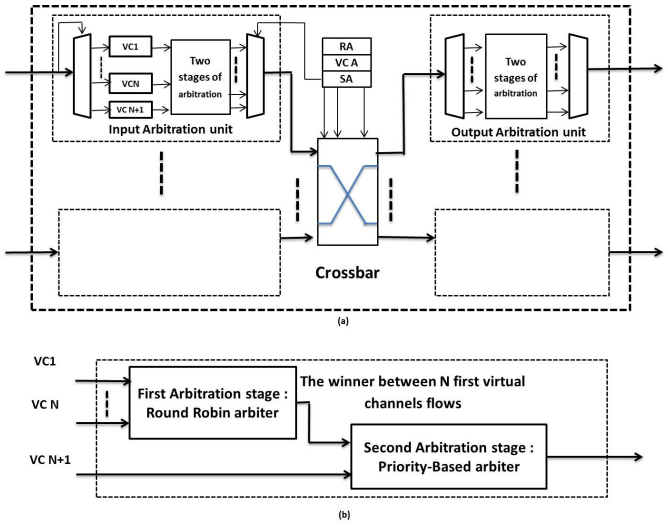

Fig. 1: DAS Router architecture: (a) Architecture, (b) Stages of arbitration

1) Input Arbitration Unit: Many virtual channels of the same input port can ask to advance to different or the same output port while the router can accept just one virtual channel from each input port at each cycle. The main task of input arbitration unit is to choose one virtual channel for each input port.

2) Output Arbitration Unit: On the other hand, many virtual channels of different input ports can ask to advance to the same output port while only one virtual channel can be accepted by each output port. The main task of output arbitration unit is to choose one virtual channel for each output port.

Input and output arbitration units are based on two stages of arbitration. The first stage is a round robin arbitration between the N first virtual channels while the second stage is a priority-based arbitration between the winner of the first stage and the last virtual channel.

The first stage has to be fair and gives equal chance between all the high-critical flows while, the second stage provides the flit-level preemption to high-critical flows. So, the winner of the first stage can preempt at flit-level the last virtual channel used by low-critical flows.

IV. Implementation and Evaluation

We have implemented the DAS router in the cycle accurate SystemC-TLM simulator SHOC [11]. SHOC provides all NoC components for the simulations of MPSoCs. It supports different types of traffic generators and consumers, and allows us to observe the traffic in the NoC. The evaluation consists of two experiments. First, we study the impact of resource sharing on high-critical flows in order to check the ability of DAS router to bound communication delays. Second, we evaluate the additional latency of low-critical flow caused by high-critical flows resource reservation.

The simulation results are established for two different NoCs. Both are 4*4 2D-mesh and use XY routing algorithm, but the first one is based on a traditional architecture of wormhole virtual channel router and the second one is the router we proposed in III.

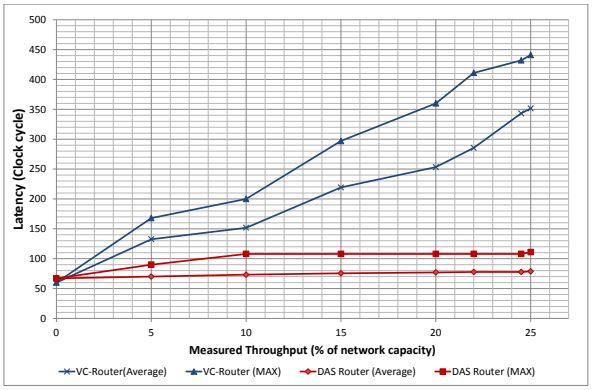

Fig. 2: Latency of high-critical flow with 3 physical links

A. High-critical flow latency

In this section, we evaluate the impact of the resource sharing on the communication delay of high-critical flows. In every evaluation, one high-critical flow is assigned to a randomly generated source and destination node. Then, we perform 100 simulations by increasing the use rate of the network. For each simulation, we generate low-critical flow sets which share some physical links with the high-critical flow. For this experiment, the size of high-critical flow is 2 flits while the low-critical flow's size is 8 flits. The release time and the period of each flow are randomly generated.

Fig. 2 shows the latency of the high-critical flow with different use rates of the network; The high-critical flow goes respectively through 3 physical links from its source to its destination node. As expected, the combination between SAF and virtual channels with flit-level preemption reduces significantly the latency of high-critical flows. For 15% of network use rate, the DAS router reduces by 80% the additional latency for high-critical flow using 3 links comparing to a VC router. Using DAS router, high-critical flows are less affected by the sharing of resources with low-critical flows.

B. Low-critical flow latency

In this section, we evaluate the latency overhead on low-critical flows due to high-critical flows resource reservation comparing to virtual channel routers. For each measure in this experiment, we generate randomly one low-critical flow crossing 4 hops. We do 100 simulations by increasing the number of high-critical flows and by decreasing the high-critical flow's period in order to increase the network use rate. Then, we measure the low-critical flow latency. The generated high-critical flow set shares the same links with the low-critical flows. For this experiment, high-critical flow's size is 2 flits while the low-critical flow's size is 8 flits. The release times of each flow are also randomly generated.

Fig. 3 shows the latency of low-critical flows with different use rates of the network. We can see that the preemption of low-critical flows and the system mode change used in the DAS router lead to larger latencies for low-critical flows compared to a virtual channel router. Notice that low-critical flows in MCS should tolerate some additional delays without damaging the integrity of the whole system.

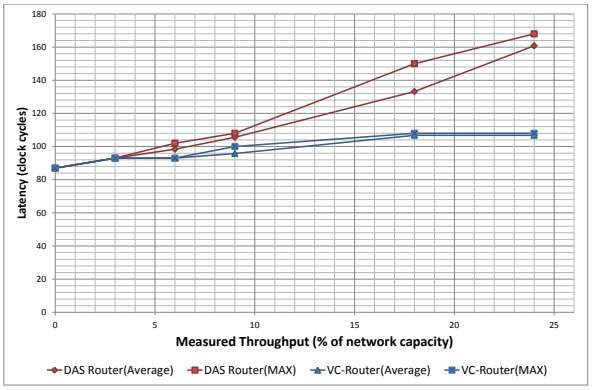

Fig. 3: Latency of low-critical flow with 4 physical links

V. Related Works

There is a growing interest in recent years to design NoCs for MCS. Such protocols aim to deal with the trade-off between resource sharing and separation of different criticality levels.

Tobuschat et al. have explicitly developed a NoC to support MCS. Their protocol, called IDAMC [12], ensures that high-critical flows meet their deadlines whilst maximizing the bandwidth given to the low-critical flow using back suction technique. WPMC [13] is another protocol applied to wormhole virtual channels NoCs, in order to integrate dual criticality systems. This protocol is improved in [14]. [15] presents a run-time configurable NoC which supports safety-critical traffic and best-effort traffic. It allows prioritizing best-effort traffic over critical traffic, while monitoring is used to change the priority during run-time.

[12], [13] and [15] use wormhole policy with flit-level priority preemption. All high-critical flows with the same priority share the same virtual channel. In addition, even high-critical flows can be pre-empted in flit-level by other highly priority flows. This configuration leads to a pessimistic worst case communication time which will limit the low-critical communications [4].

VI. Conclusion

In this paper, we propose DAS router. DAS router implements a new router architecture intended to deploy Mixed-Criticality Systems on NoCs. On the assumption that the critical communication traffic is composed of small packets, Store and Forward switching is used for high-critical flows in order to minimize the degree of pessimism on the estimation of the worst-case communication time. At the same time, wormhole policy remains for low-critical flows because this policy does not require large buffers and it also minimizes the average communication latency. The evaluation of this solution has shown a gain of 80% on latency of the critical flows. We synthesize our router with a 28nm SOI technology and show that the size overhead is limited of 2.5% compared to the solution based on virtual channel router.

Future work will include the study of mixed-criticality end-to-end response time analysis considering delays introduced by the NoC [16], the modeling and the verification of the protocol part of DAS-Router using the IF Language [17].

VII. Acknowledgments

This work and Cheddar[1] (a GPL real-time scheduling analyzer) are supported by Brest Métropole, Ellidiss Technologies, CR de Bretagne, CG du Finistère and Campus France PESSOA programs number 27380SA and 37932TF.

References

[1] A. Burns and R. Davis, "Mixed criticality systems-a review, 9^{th} ed." Department of Computer Science, University of York, Tech. Rep., Jan 2017, http://www-users.cs.york.ac.uk/burns/review.pdf.

[2] Z. Shi and A. Burns, "Real time communication analysis for on-chip networks with wormhole switching," in *Proceedings of the Second ACM/IEEE International Symposium on Networks-on-Chip (NOCS)*, Nov 2008, pp. 161–170.

[3] N. Kavaldjiev, G. J. M. Smit, and P. G. J. nsen, "A virtual channel router for on-chip networks," in *IEEE International SOC Conference, 2004. Proceedings.*, Sept 2004, pp. 289–293.

[4] L. S. Indrusiak, A. Burns, and B. Nikoli, "Analysis of buffering effects on hard real-time priority-preemptive wormhole networks," Tech. Rep. arXiv:1606.02942, Jun 2016, https://arxiv.org/abs/1606.02942.

[5] K. Jain, S. K. Singh, A. Majumder, and A. J. Mondai, "Problems encountered in various arbitration techniques used in noc router: A survey," in *2015 International Conference on Electronic Design, Computer Networks Automated Verification (EDCAV)*, Jan 2015, pp. 62–67.

[6] K. Jetly, "Experimental comparison of store-and-forward and wormhole NoC routers for FPGAs," Ph.D. dissertation, University of Windsor, Nov 2013.

[7] Z. Shi, "Real-time communication services for networks on chip," Ph.D. dissertation, University of York, Nov 2009.

[8] C. Pagetti, D. Saussié, R. Gratia, E. Noulard, and P. Siron, "The ROSACE case study: from simulink specification to multi/many-core execution," in *Proceedings of the* 20^{th} *International Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, April 2014, pp. 309–318.

[9] M. A. Al Faruque and J. Henkel, "Minimizing virtual channel buffer for routers in on-chip communication architectures," in *Proceedings of the Design Automation and Test Europe Conference (DATE)*, Mar 2008, pp. 1238–1243.

[10] M. A. Al Faruque and J. Henkel, "Transaction specific virtual channel allocation in QoS supported on-chip communication," in *Proceedings of the IEEE International Conf on Application-specific Systems, Architectures and Processors (ASAP)*, July 2007, pp. 48–53.

[11] M. Sepúlveda, M. Strum, and W. Chau, "Performance impact of QoSS (quality-of-security-service) inclusion for NoC-based systems," in *17th IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*, Oct 2009, pp. 12–14.

[12] S. Tobuschat, P. Axer, and R. Ernst, "IDAMC: A NoC for mixed criticality systems," in *19th International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*. IEEE, 2013, pp. 149–156.

[13] A. Burns, J. Harbin, and L. S. Indrusiak, "A wormhole NoC protocol for mixed criticality systems," in *Real-Time Systems Symposium (RTSS)*. IEEE, Dec 2014.

[14] L. S. Indrusiak, J. Harbin, and A. Burns, "Average and worst-case latency improvements in mixed-criticality wormhole networks-on-chip," in *Proceedings of the 27th Euromicro Conference on Real-Time Systems (ECRTS)*, July 2015, pp. 47–56.

[15] S. Tobuschat and R. Ernst, "Efficient latency guarantees for mixed-criticality networks-on-chip," in *Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 18-21 April 2017, pp. 43–49.

[1] http://beru.univ-brest.fr/~singhoff/cheddar/

[16] M. Dridi, S. Rubini, F. Singhoff, and J.-P. Diguet, "DTFM: a flexible model for schedulability analysis of real-time applications on noc-based architectures," in *4th IEEE International Workshop on Real-Time Computing and Distributed systems in Emerging Applications (REACTION)*, Nov 2016, pp. 43–49.

[17] M. Dridi, M. Lallali, S. Rubini, M. Sepúlveda, F. Singhoff, and J.-P. Diguet, "Modeling and validation of a mixed-criticality noc router using the if language," in *10th International Workshop on Network on Chip Architectures (NoCArc)*, Oct 2017.