DOI 10.12694/scpe.v20i3.1423
Scalable Computing: Practice and Experience ISSN 1895-1767
Volume 20, Number 3, pp. 495–509. http://www.scpe.org © 2019 SCPE

VIRTUAL CHANNEL AWARE SCHEDULING FOR REAL TIME DATA-FLOWS ON NETWORK ON-CHIP

MOHAMMED AMINE MEGHABBER[*], LAKHDAR LOUKIL[†], RICHARD OLEJNIK[‡], ABOU EL HASSAN BENYAMINA[§] AND ABDELKADER AROUI[¶]

Abstract. The increasing complexity of real-time applications presents a challenge to researchers and software designers. The tasks of these applications usually exchange many data-flows and often need to satisfy real-time constraints. Although the Network on-Chip (NoC) paradigm offers an underlying communication infrastructure that gives more hardware resources, tasks and data-flows cannot be performed before their deadlines. In recent works, preemptive wormhole switching with fixed priority has been introduced to meet real-time constraints of real-time applications. However, it suffers some bottleneck such as hardware requirement where none of these works takes account of the number of implemented Virtual Channels on the router. To alleviate this problem, we propose a novel scheduler for soft real-time data-flows application that takes into account the lack on resource in routers in term of Virtual Channels. Experimental results obtained on a benchmark of synthetic and soft real applications have shown the efficiency of our approach in term of real-time constraints satisfaction for data-flow traffics and hardware requirements.

Key words: Embedded Systems, Network on-Chip, Real Time systems, Router, Wormhole.

AMS subject classifications. 68M10, 68M14, 68M20

1. Introduction.

As the technology advances, the number of processors in a chip grows [1] due to the transistor integration progress. In the last three decades, the hardware (processors) evolution was exponential and the Moore law [2] which states that "the number of transistors on an integrated circuit doubled every year", has reached its physical limits due to heat dissipation and energy consumption that increased with the number of processors. This led nano-architecture engineers to propose a new system-on-chip (SoC) architecture called network-on-chip (NoC) [3].

Inspired from the classical network, Networks on-Chip (Fig. 1.1) have emerged as a new communication paradigm in systems on chip [4]. Unlike classical Multiprocessors System on-Chip (MPSoC) shared bus architectures [5] which do not allow scaling and behave unpredictable when connecting up to 10 processors [6], NoC systems offer high scalability and improved parallelism [4]. Currently, many systems on-chip manufacturers have developed commercial NoCs such as TeraFLOPS on-chip network developed by Intel [7] and TILEPro64 developed by Tilera [8].

On the other hand, embedded applications such as those found in aerospace domain [9] are increasingly complex and subject to strong energy consumption and real-time constraints that require efficient and high performance execution frameworks. Indeed, tasks of such applications are interdependent and usually exchange many messages during their execution on processing elements (PE), which requires an efficient underlying communication infrastructure like the one provided by a NoC. Furthermore, these dataflows are mostly subject to soft or hard real-time constraints on their period and their deadline, for example. In this conditions, dataflows must arrive at their destination nodes (Processing Element) in time and any exceeding deadlines can at best degrade the application performance and at worst have desastrous consequences.

Due to its small buffering requirement, high throughput and low latency, the wormhole switching technique [10] is the most common flow control mechanism in NoC routers (Fig. 1.2). In this switching technique, a dataflow is split into packets, and a packet is split into flits[1] (Fig. 1.3), where header flit embeds routing information and data flits contain user data. When a router receives a flit, it is first buffered in an input Virtual Channel (VC) [11] and then the arbiter function of the current router determines which of the candidate flits the one that will be routed to the next router. One of the most commonly used arbitration technique is round robin (RR) [12]. RR handles VC in circular order and ignores packet characteristics such as priority of a packet, which

[*]Computer Science Department, University of Oran 1 Ahmed Ben Bella, Algeria (meghabber.mohammed@edu.univ-oran1.dz).
[†]Computer Science Department, University of Oran 1 Ahmed Ben Bella, Algeria. (loukil.lakhdar@univ-oran1.dz).
[‡]CNRS, Univ. Lille, Centrale Lille, IMT Lille Douai, Lille, France (richard.olejnik@univ-lille1.fr).
[§]Computer Science Department, University of Oran 1 Ahmed Ben Bella, Algeria (abouelhassan.benyamina@univ-oran1.dz).
[¶]Satellite development Center, Algeria (aaroui@cds.asal.dz).
[1]FLow control unIT. The data unit processed by a router.

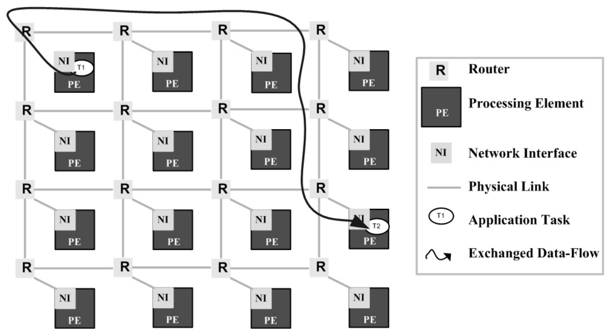

FIG. 1.1. *2D mesh network-on-chip.*
The figure shows 2D-mesh Network on chip topology where a set of processors are interconnected via routers. NI formats exchanged messages between application tasks to packets then flits.

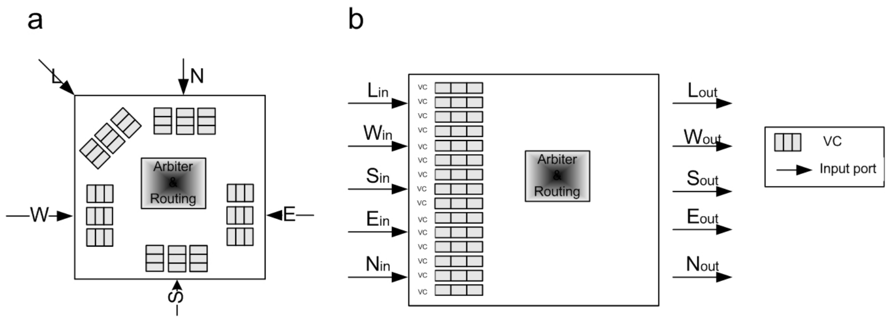

FIG. 1.2. *Interface views of a typical router: (a) Architectural view; (b) Functional view.*
The figure shows the router architecture. It includes five input and output port, arbiter and routing function. Each Input physical port includes a set of Virtual Channels (VC) connected to buffers. Each buffer has a limited amount of space for flits. For each cycle, the arbiter and routing fabric select one flit to be routed through the output port. The output port is a physical wire that transfers flits to next router.

makes it unsuitable for real-time applications where dataflows have priorities and must be routed according to their priority. To deal with prioritized dataflows, a wormhole switching with a fixed priority preemption has been proposed [13]. In this switching technique, Virtual Channels of an input port are labeled with a priority value and each new flit arriving at a router is assigned to the Virtual Channel labeled with the priority of the flit. This implies that the number of Virtual Channels per input port should be at least equal to the number of distinctive flit priorities. This constraint is not always feasible. On the one hand, nowaday real-time applications are more and more greedy and, on the other hand, it is technically not feasible to manufacture routers with a large number of VC. We notice, for instance, that the experimental Intel Single-chip Cloud Computer (SCC) NoC [7] implements 8 VC per input port.

In this work, we propose a general-purpose Virtual Channel allocation technique for wormhole switching with priority pre-emption which is able to schedule soft real-time applications by assigning a Virtual Channel to a flit of a given priority whatever the complexity of the application (number of priorities) and whatever the architecture of the router (number of VC per input port). This paper is structured as follows. Section 2 presents recent works related to real-time dataflows scheduling. Section 3 describes the application model targeted in this work. Section 4 depicts the VC allocation technique we propose. Then, We report and analyse simulation results performed on synthetic and real-life applications (Sect.5). Finally, we conclude and give some

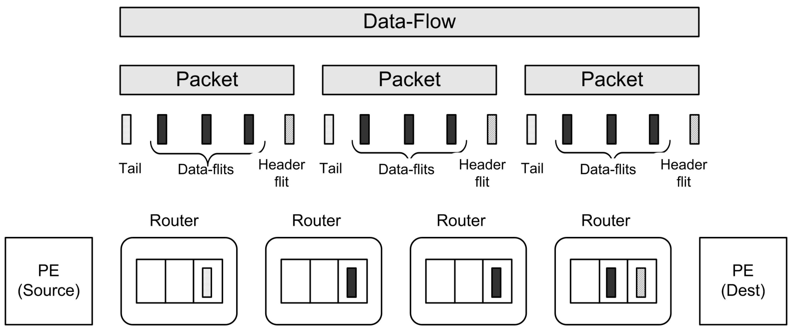

FIG. 1.3. *Packet formatting in wormhole switching.*
An exchanged message is split into packets, and a packet is split into flits. The flit represents the data unit processed by the router. During packet transmission, flits of the packet can be stored in multiple routers (buffers). If a contention occurs, just a flit will be blocked not the entire packet.

perspectives.

2. Related Work. During the last few years, real-time dataflows scheduling problem on NoC-based systems on chip has been widely studied. Various scheduling methods have been proposed in the literature where the objective is to achieve the real-time constraints of embedded applications using a minimum hardware resources. In this section, we will report the most important work related to this problem.

The preemptive wormhole switching with fixed priority was firstly proposed by Balakrishnan et al. [14] to address the problem of real-time communication in multiprocessor networks. This technique has subsequently been adopted by several researchers to estimate the worst case response time for wormhole switching. Hary et al. [15], in their paper, consider the links traversed by a dataflow as a single shared path and propose their analysis accordingly, while Kim et al. [16] attempt to construct a dependency graph for its response time analysis.

Shi et al. [17] have adapted the response time analysis in a fixed priority processor proposed in [18] for the analysis of the response time in the case of several dataflows. The authors consider that the NoC has as many Virtual Channels (VC) as dataflows, that the VC and dataflows have priorities and that any given priority dataflow is assigned to the VC that has the same priority. Such proposal presents scaling problems since it is technically unfeasible to have routers with a large number of VC. The implementation of a VC is indeed expensive in terms of manufacturing cost and logic area. Mello in [19] has stated that while the Hermes NoC with a single VC took 17% of the logic area of their hardware test-bed, the design with two and four VC took 32.61% and 75.41% of the logic area respectively. We notice that input ports of routers of a Intel TeraFLOPS NoC [7] developed by Intel has 8 Virtual Channels.

In order to improve their proposal and reduce hardware resources required to implement their scheduling strategy, She et al. in [20] suggest another scheduling algorithm that partitions dataflows into clusters and assigns a unique priority to dataflows of a cluster. This allows to reduce the number of Virtual Channels (since dataflows of a cluster will be assigned to a single Virtual Channel) and achieve real-time constraints but may give rise to unpredictable network latency and blocking problem. The latter can take place since a VC stores different packets shared the same priority and served in FIFO because the priority preemption is only available between VC and therefore a packet can be blocked by another packet with the same priority which holds a VC which can in turn block other packets, and so on.

In [21], authors propose a Dynamically Changing Virtual Channels (DCVC) scheduling technique where each packet still maintains its priority constant during an entire traversal of the NoC but may occupy different Virtual Channels within routers on its path. According to the authors, this considerably reduces the number of VC. DCVC however requires the header flits to hold the list of all Virtual Channels of the traversed path which will overload the routers and induce scalability issues. The same authors suggest in [22] a dynamic priority policy EDF (Earliest Deadline First) as arbiter in such a way to improve schedulability. The EDF scheduler allows jobs (task instance) to change their priority dynamically on runtime according to their deadline. A worst case time analysis has been performed but no implementation has been done. In their last work [23], Nikoli at al. propose a timing analysis for arbitrary VC size where they prove that bigger Virtual Channel buffers do not necessarily help tasks and data-flows to be performed before their deadlines.

Sudev et al. [24] who consider that a preemptive router is expensive in terms of implementation costs and energy consumption, adopt a real-time non-preemptive router with fixed priority. To process high priority dataflows, they propose a technique named PTF (Priority Forwarding and Tunneling) which consists in temporarily increase the priority of low-priority packets that prevent the timely transmission of high-priority packets and thus avoid the head of line (HoL) phenomenon.

In a later work [25], the authors propose an arbitration technique for non-preemptive routers. In their proposal, the router suspends the transmission of the current packet upon the arrival of a packet of higher priority. In that case, the router split the transmission by sending a tail flit followed by the construction of a new header that initiates a new arbitration request on that router. This would allow the higher priority packet to start transmission. Like that, they emulate the preemption.

Authors in [37] consider proposed methods in the literature as optimist, and they have improved their analysis through the distinction between downstream and upstream indirect interferences. However, they do not take into account the number of available VC. Afterwards, Giroudot at al. [38], have extended [37] to support the backpressure and flow serialization.

All works presented above propose solutions to schedule real-time data-flows and aim to achieve real time constraints. However, they all consider that the NoC has as many VC as dataflows. In practice, it is unfeasible to have routers with a large number of VC. Our challenge is to propose a lightweight algorithm which takes into account the hardware limitation in term of VC and schedules dataflows whatever the complexity of the application (number of priorities) and whatever the architecture of the router (number of VC per input port). In this paper, we propose a new Virtual Channel allocation and assignment technique, which reduces significantly the number of needed VC implemented in router. Otherwise, each dataflow could find a VC to be allocated even where there is not as many VC as dataflows. This technique relaxes hardware requirements in term of buffers (VC) and improve the real-time application schedulability.

3. Application Model.

An application is modeled by a finite set $\Omega = \{\delta_1, \delta_2, \cdots, \delta_n\}$ of n real-time dataflows δ_i. Each dataflow δ_i is characterized by the following parameters:

- S_i: Source node,
- $Dest_i$: Destination node,
- A_i: Arrival time
- L_i: Basic latency which is the maximum transmission time from the source node S_i to the destination node $Dest_i$. This parameter is estimated assuming no congestion in the network,
- p_i: Period (elapsed time between successive releases of packets of δ_i),
- D_i: Deadline which is the due time that the scheduling of the dataflow must not exceed,
- P_i: Priority.

The basic latency L_i can be evaluated using the following formula [26]:

$$L_i = H \times d_R + F \times d_T, \tag{3.1}$$

where :

- H is the number of hops from the source processing element to destination processing element.
- d_R is the time it takes for a router to switch a flit.
- d_T is the transfer time of a flit from a router to a neighboring router.
- F is the number of data flits in a packet. F is given by the following equation:

$$F = \frac{size(Packet) + A_d}{size(Flit)}, \tag{3.2}$$

where A_d is the size of the additional data (header and tail flits).

For a regular 2D-mesh NoC, the number of hops H is equal to :

$$H = |R_d^x - R_s^x| + |R_d^y - R_s^y|, \tag{3.3}$$

where:

- R_s^x and R_s^y are respectively the x and y coordinates of the source processing element,
- R_d^x and R_d^y are respectively the x and y coordinates of the destination processing element.

Hereafter, we will assume that d_R=d_T=1. So, Eq. 3.1 becomes as follows:

$$L_i = |R_d^x - R_s^x| + |R_d^y - R_s^y| + F. \tag{3.4}$$

4. Modulo-based VC Allocation and Assignment. As mentioned in Sect.2, most of the work dealing with scheduling real-time applications in NoC consider routers running a fixed priority preemptive wormhole switching to schedule real-time dataflows [16, 15, 27, 22]. In such routers, a newly arrived high priority header flit preempts a low priority flit being transmitted. Authors of these works generally assume that there are at least as many available VC per input port as priorities. Such assumption is technically unrealistic since the number of VC per input port in real-life NoCs is usually much more smaller than the number of priorities of real-time dataflows. For instance, Intel TeraFLOPS NoC [7] implements 8 buffers per input port. Moreover, hardware experience results [19, 28] show that implementing additional buffers is an expensive process.

In this work, we propose a general-purpose VC allocation and assignment strategy named *Modulo-based VC allocation and assignment technique* (designated hereafter by MVCAA). The qualifier "general-purpose" is used to say that MVCAA allows scheduling soft real-time dataflows whatever the number of VC in a router and whatever the size of real-time applications (the number of priorities).

Algorithm 1 depicts the pseudo-code of MVCAA. Overall, MVCAA proceeds in two steps. The first step (Sect.4.1) is the VC allocation and assignment that determines the VC that will host flits of each incoming dataflow δ_i of priority P_i. The second one (Sect.4.2) is the scheduling step that determines how the arbiter selects the VC to process.

Algorithm 1 Pseudo-code of modulo-based VC allocation and assignment (MVCAA).

```
Input: $AS$= {($\delta_1$,$P_1$), ($\delta_2$,$P_2$), $\cdots$, ($\delta_n$,$P_n$)} of $n$ couples ($Flit$, $Priority$)
        $VC$={$VC_1$, $VC_2$, $\cdots$, $VC_m$ } of $m$ Virtual Channels, where $m << n$
Output: $ALOC$= {($\delta_1$, $VC_{i1}$), ($\delta_2$,$VC_{i2}$), $\cdots$, ($\delta_n$,$VC_{im}$)} of couples ($Flit$, $Allocated\ Virtual\ Channel$)
for (Arriving flit $\delta_i$) do
    if ( $\delta_i$ is a header flit) then
        $BufferTag = P_i\%|\mathrm{VC}|$;
//Allocate $VC_{BufferTag}$ and assign $\delta_i$ to $VC_{BufferTag}$;
        $Allocate(\delta_i, VC_{BufferTag})$
//Update the $VC_{BufferTag}$ priority register;
        $UpdatePriorityRegister(VC_{BufferTag}, P_i)$;
    else//$\delta_i$ is a data flit, in which case assign it to $VC_{BufferTag}$;
        $Assign(\delta_i, VC_{BufferTag})$;
    end if
end for
```

4.1. Step 1: Assigning a VC to a dataflow. The architecture of a router considered in the present work is outlined in Fig.4.1. Each input port has a number of Virtual Channels and each Virtual Channel has a priority tag register (represented by a square on the VC) indicating the current priority of the VC (i. e. priority of flits the VC currently hosts). Thereby for each incoming dataflow δ_i of priority P_i, the Virtual Channel VC_j that would be allocated to that dataflow is determined using the following equation:

$$\forall \delta_i \in \Omega, \delta_i \text{ is assigned to } VC_j \text{ where } j = P_i\%(|VC|), \tag{4.1}$$

where:

- VC: the set of Virtual Channels of a physical input port of a router.
- Ω: the set of dataflows of a real-time application.

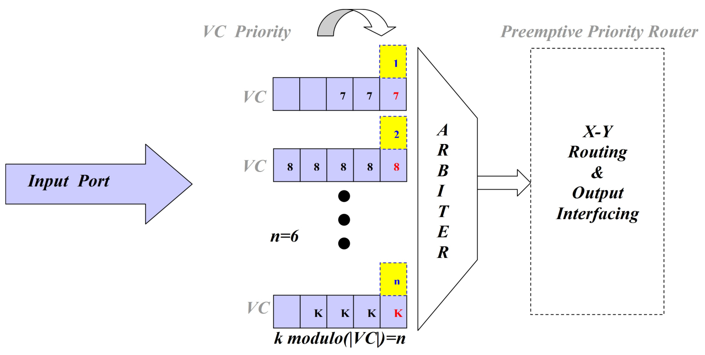

FIG. 4.1. *Dynamic VC priority for wormhole Router*

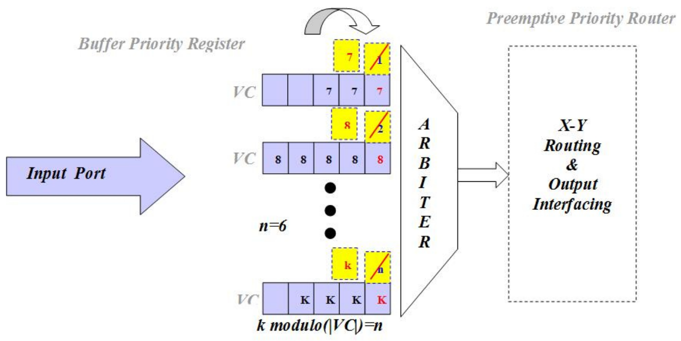

FIG. 4.2. *Updating VC tags.*

If VC_j is free, it is locked and allocated to the dataflow δ_i, and its priority tag register is set to the priority P_i of the dataflow δ_i (Fig. 4.2). The lock of VC_j will be released by the tail flit of the packet. Otherwise, the current dataflow δ_i is ignored and the next incoming dataflow is considered.

4.2. Step 2: Scheduling dataflows. At each cycle, the router assigns an output port to each incoming packet (stored in an input VC) according to its destination using a flow priority-based adaptive routing (see Sect.4.3). To this purpose, we implement a priority-based arbiter that scans input VC of the router input port to look for the highest priority packet. If the highest priority packet cannot be routed to the adjacent router for lack of buffer space in that router, the next highest priority packet is considered. To be aware of the buffers status of adjacent routers, a credit-based flow control [29] is performed. In credit-based flow control, a *credit counter* register associated to each output port indicates the credits of input buffers of the adjacent router in term of free slots. It sends a ready signal to the arbiter when the number of available buffer slots at the output buffer is greater than zero. When a header flit is selected to be routed to the next router, the input buffer of the adjacent router buffer is be locked. This lock will be released once the tail flit leaves the buffer.

4.3. Adaptive XY routing. XY routing is the most popular routing algorithm for 2D mesh or torus NoC topologies. It first routes a packet in X (horizontal) direction then in Y (vertical) direction. The asset of XY routing is that it never runs into deadlock or livelock [30]. Its disadvantage is that it produces a high load in the center of the network which can cause a hot spot.

For a better load balancing, we implemented a *half adaptive XY routing* [30] where the router first tries to route the flit in X direction and assigns a free VC if it exists. Otherwise, the router routes the flit in Y direction. If no VC is free in the two directions, the flit is blocked and the router serves the next dataflow. The pseudo-code of Priority based half adaptive XY routing is depicted in Alg.2.

Algorithm 2 Priority based half adaptive XY routing.

```
1: Input: Flit $\delta_i$ of priority $P_i$, source node $S_i$ and destination node $Dest_i$
2: Ouput: nextRouter, destVC
3: destVC = $P_i\%|VC|$;
4: nextRouter = xyRouting($S_i$, $Dest_i$);
5: if (vcAvailable(nextRouter, destVC)) then
6:     lock(nextRouter, destVC);
7:     moveFlit($\delta_i$, nextRouter, destVC);
8: else
9:     nextRouter = yxRouting($S_i$, $Dest_i$);
10:    if vcAvailable(nextRouter, destVC) then
11:        lock(nextRouter, destVC);
12:        moveFlit($\delta_i$, nextRouter, destVC);
13:    else
14:        nextRouter = -1;
15:        destVC = -1;
16:    end if
17: end if
```

5. Experimental results and analysis. In this section, we will report some experimental results carried out in this study. The objective of these experiments is twofold: the first is to test the performance of our proposal (i.e. MVCAA technique, see Sect.4) by comparing it with some state-of-the-art real-time scheduling methods for NoC-based mutiprocessor system-on chip, the second is to test the scalability of MVCAA when increasing the size of the NoC. The performance metric retained is the ratio of the number of dataflows that are missing their deadlines on the number of real-time dataflows injected into the NoC. We recall that a dataflow δ_i misses its deadline if its response time R_i is greater than its deadline D_i. The response time is given by the following expression:

$$R_i = L_i + B_i, \tag{5.1}$$

where L_i is the basic latency (see Sect.3) and B_i is the time during which the dataflow δ_i is blocked in a buffer by higher priority dataflows. In our case, an estimate of R_i is provided by the NoC simulator [31] and corresponds to the latency when the dataflow δ_i reaches its PE destination.

We have compared MVCAA with two state-of-the-art real-time scheduling methods for NoC-based mutiprocessor system-on chip: Priority Share Policy (PSP) of Shi and al. [20], and Dynamically Changing Virtual Channels (DCVC) of Nikolič and al. [21] where both aim to reduce the number of Virtual Channels for priority-preemptive NoCs.

Experiments have been carried out on the NoC in-house simulator published in [31]. The simulator implements a 2D-mesh NoC topology and a wormhole switching technique, uses adaptive XY routing (Alg.2) and the credit-based flow control. It allows the user to keep track in real-time of the communications and highlights routers that suffer from congestion. The input of the simulator is an `XML` file that describes the application dataflows and the NoC configuration. An application is a set of dataflows and each dartaflow is characterized by a deadline, a period, an arrival time, a basic latency, a type (header or data flit) and a priority. The NoC is described by its topology (2D-mesh), the arbitration technique (MCVAA, PSP, DCVC), the NoC size (number of routers in each dimension), the number of VC per input port and the VC size (number of slots per VC). Algorithm 3 depicts the simulator kernel. The simulation output is an `XML` file that reports, for each simulation cycle, buffer states of the different routers, traffics, latency and missed deadlines of different packets over the entire NoC. For each cycle and each router and according to the arbitration technique, a dataflow is identified and routed to the adjacent router using half adaptive XY routing.

Two categories of applications have been considered: synthetic and real-life applications. Synthetic real-time applications have been generated using the random generator proposed in [32]. Three real-life real-time

Algorithm 3 NoC simulator kernel [31].

```
1: Input: $AS= \{(\delta_i, P_i)/ \delta_i \in \Omega, P_i$: priority of dataflow $\delta_i\}$;
2:          NoC architecture.
3: Output: $NI\_ALOC=\{(\delta_i,NI_K)/ NI_K$: Network Interface$\}$
4: load(Application)
5: load(NoC architecture)
6: for cycle = 0 to numCycles do
7:     for each $NI$ in NoC do
8:         if ($NI$ has traffic to forward) then
9:             $moveTrafficFromNIToLocalRouter$ ;
10:        end if
11:    end for
12:    for Each Router $R$ in NoC do
13:        for Each Input port of $R$ do //Define output port for each incoming flit;
14:            $UpdateRouterRequests(Routing)$;
15:        end for
16:        for Each Output port of $R$ do
17:            $CheckCandidateFlitsPerOutputPort$;
18:            $RunArbitrationToSelectGrantedFlit$;
19:            $SwitchGrantedFlitToNextRouter$;
20:            $MVCAA(AS, VC, ALOC)$;
21:    end for
22:    end for
23: end for
```

Table 5.1
Experimental protocol.

NoC sizes	8×8 and 16×16 2D-mesh
Buffering	4, 5, and 8 VC per input port
Switching	Preemptive wormhole
Arbiter	Per priority
Flow control	Credit-based
Routing	Half Adaptive XY routing
Application sizes	10, 20, 30, 40, 50, 70, 80, 100 dataflows

applications have been selected: Video Object Plane Decoder (VOPD) [33, 34], MPEG4 [35], and an autonomous vehicle application (AutoV) [36]. Table 5.1 summarizes the experimental protocol adopted in our experiments.

5.1. Synthetic applications. Figures 5.1-5.6 give the percentage of missed deadlines for application sizes ranging from 10 to 100 dataflows and for VC numbers equal to 4, 5 and 8. 100 executions were performed for each configuration and the average value was considered. Figures 5.1-5.3 (8×8 2D mesh NoC) show that, for all configurations, `MVCAA` clearly outperforms `DCVC` and behaves almost like `PSP` in terms of the percentage of missed deadlines. However, for 16×16 2D mesh NoC, `MVCAA` outperforms both `PSP` and `DCVC` (see Figs.5.4-5.6) and the gap grows with the increase of the size of the application. This proves the robustness of our approach to increasing the size of the NoC. The performance degradation of `PSP` and `DCVC` can be explained by the fact that for `PSP` the packet blocking duration (see section 2) increases with the size of NoC (in term of the number of hops) and will have a negative impact on the number of missed dealines. Regarding `DCVC`, increasing the size of the NoC induces the increase of the number of hops and then of the size of `DCVC` packets which consequently increases the latency of the packet.

To study the impact of NoC architecture (Number of routers, number of VC per input port) on MVCAA schedulability of real-time dataflows, we considered 3 NoC sizes (7×7, 8×8 and 16×16) each with diferent

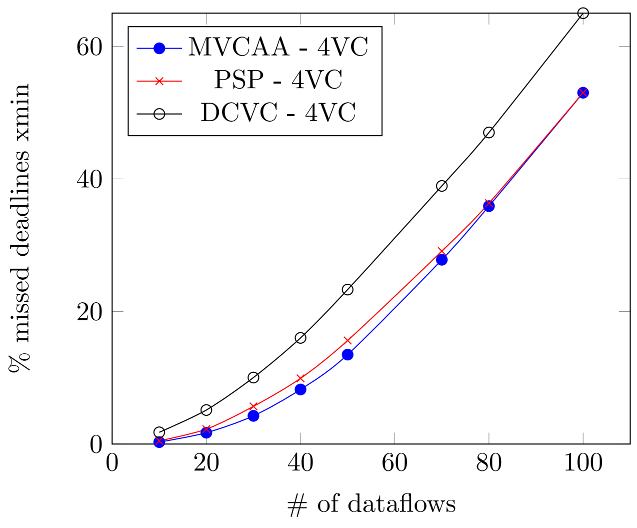

Fig. 5.1. *Percentage of missed deadlines on a 8×8 2D mesh NoC (4 VC).*

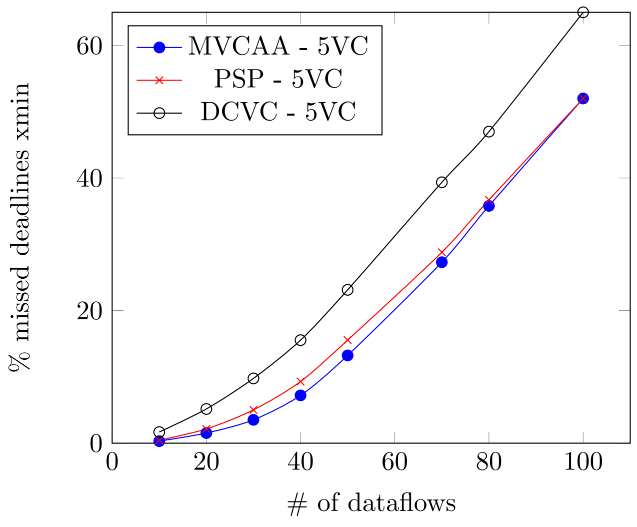

Fig. 5.2. *Percentage of missed deadlines on a 8×8 2D mesh NoC (5 VC).*

numbers of VC per input port (4, 5 and 8). Experimental results are illustrated by Figs. 5.7, 5.8 and 5.9. Several remarks can be drawn from these figures. The first one is that the increase in the number of VC while maintaining the architecture unchanged does not significantly impact the schedulability of real-time applications. This corroborates the fact that multiplying the number of VC in a router does not significantly improve the scheduling of dataflows, contrary to what is stated in some works as in [16, 17]. Consider the fact that the router always serves the highest priority packet, it seems that additional VC do not increase the application performance since all lowers priority packets have to wait for the transmission of the higher one. The other remark that we can extricate is that by slightly increasing the size of the NoC, one can gain much in quality of scheduling. We notice indeed that the 16×16 2D-mesh NoC reduces by more than 50% the number missed deadlines compared to the 8×8 NoC.

5.2. Real-life application. Three real-time embedded benchmarks are considered to evaluate the performance of the proposed `MVCAA`: Video Object Plane Decoder (`VOPD`) application [33, 34], `MPEG4` application [35]

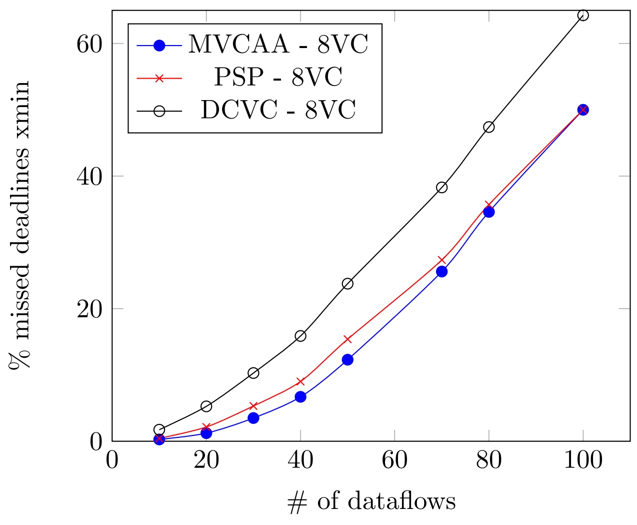

Fig. 5.3. *Percentage of missed deadlines on a 8×8 2D mesh NoC (8 VC).*

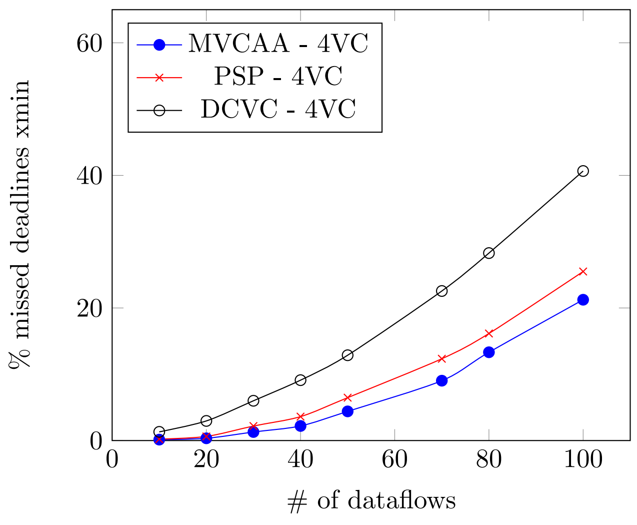

Fig. 5.4. *Percentage of missed deadlines on a 16×16 2D mesh NoC (4 VC).*

and an autonomous vehicle (`AutoV`) application [36].

`VOPD` application consists of 17 tasks that exchange data. Figure 5.10 depicts the application architecture, circles represent tasks and arcs represent exchanged data volume. Values on arcs represent the volume of data exchanged between tasks connected by the arc. Each task is characterized by three parameters: task period, capacity and deadline.

`MPEG4` consists of 20 tasks and 23 exchanged dataflows. Figure 5.11 gives the application task graph. Each task is characterized by its task period, its capacity and its deadline, the value on an arc represents the volume of data exchanged between tasks connected by the arc.

Autonomous vehicle application (`AutoV`) is characterized by large volume of dataflows (some dataflows may contain more than 38000 flits) and control loops with short messages. `AutoV` consists of 33 tasks and 38 dataflows.

We execute these applications on a 8×8 2D-mesh NoC having 8 VC per input port on different dataflow period instances. The results of the experiments illustrated in Fig. 5.12 show that `MVCAA` clearly outperforms

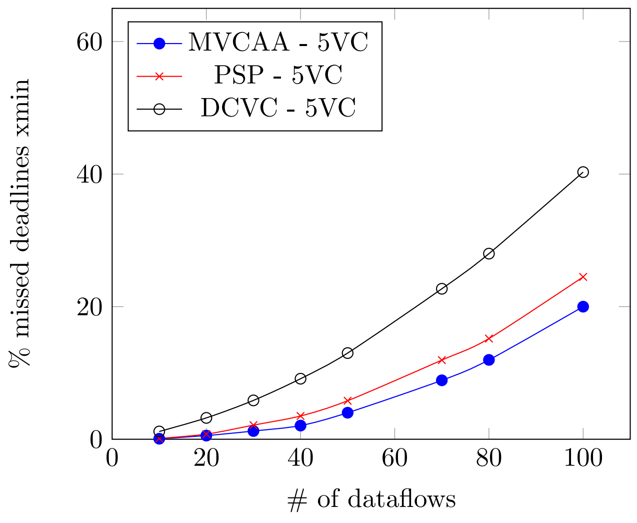

FIG. 5.5. *Percentage of missed deadlines on a 16×16 2D mesh NoC (5 VC).*

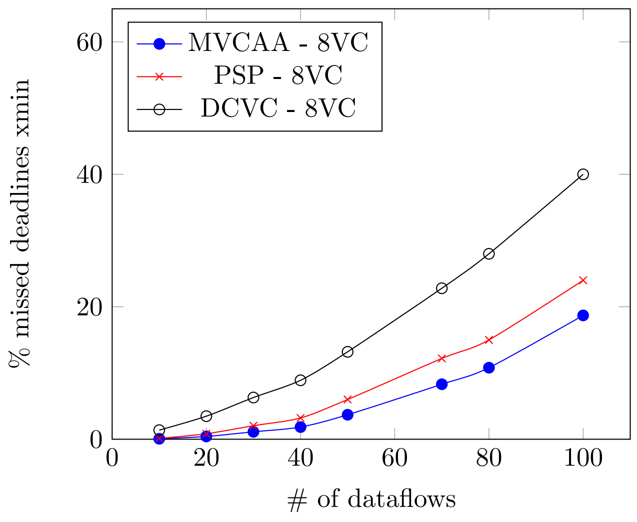

FIG. 5.6. *Percentage of missed deadlines on a 16×16 2D mesh NoC (8 VC).*

`PSP` and `DCVC` in terms of the percentage of dataflows that have saved their deadlines. The experimental results show that `MVCAA` achieves up to 19% for VOPD, 10% for `MPEG4`. For `AutoV` application, the improvement is 5% since the application exchanges huge and voluminous dataflows .

6. Conclusions. Networks-on-chip are an emerging technology and a promising communication paradigm offering more scalability and parallelism. Priority-based arbitration techniques have been proposed to schedule real-time applications. These techniques assume, however, that there are at least as many VC per router input port as there are dataflows. In this work, we have proposed a new priority-based arbitration technique named *Modulo-based VC Allocation and Assignment (`MVCAA`)* that takes into account the number of available VC in NoC and allows scheduling real-time dataflows regardless of the number of priorities and VC. Experiments have shown that `MVCAA` outperforms some state-of-the-art real-time scheduling methods.

As future works, we think that exploring new routing techniques, mapping and priority assignment algorithms would be more promising for real-time applications than implementing more VC within a router.

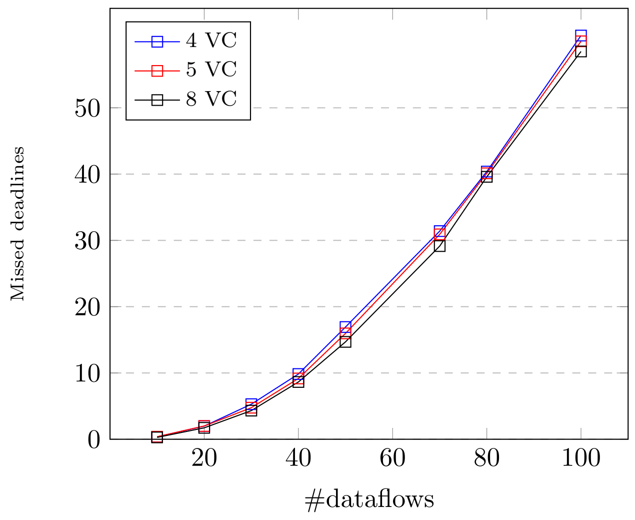

Fig. 5.7. *7×7 NoC.*

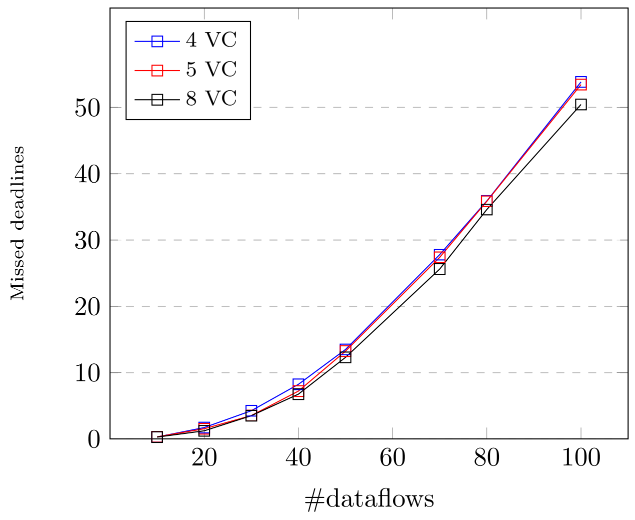

Fig. 5.8. *8×8 NoC.*

Acknowledgments. This work has been supported in part by the PHC TASSILI 14MDU917 project of the University of Oran 1, and by IRCICA,Université de Lille, USR 3380, F-59650 Villeneuve d'Ascq, France.

REFERENCES

[1] A. Jerraya, H. Tenhunen, and W. Wolf. *Guest Editors' Introduction: Multiprocessor Systems-on-Chips.* Computer, 38(7):36–40, July 2005.
[2] G. E. Moore. *Cramming more components onto integrated circuits.* In Mark D. Hill, Norman P. Jouppi, and Gurindar S. Sohi, editors, Readings in Computer Architecture, pages 56–59. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2000.
[3] L. Benini and G. De Micheli. *Networks on chips: a new SoC paradigm.* Computer, 35(1):70–78, January 2002.
[4] T. Bjerregaard and J. Sparso. *Implementation of guaranteed services in the MANGO clockless network-on-chip.* IEE Proceedings - Computers and Digital Techniques, 153(4):217–229, July 2006.
[5] S. Furber. *Future trends in SoC interconnect.* In 2005 IEEE VLSI-TSA International Symposium on VLSI Design, Automation and Test, 2005. (VLSI-TSA-DAT)., pages 295–298, Hsinchu, Taiwan, Taiwan, April 2005. IEEE.

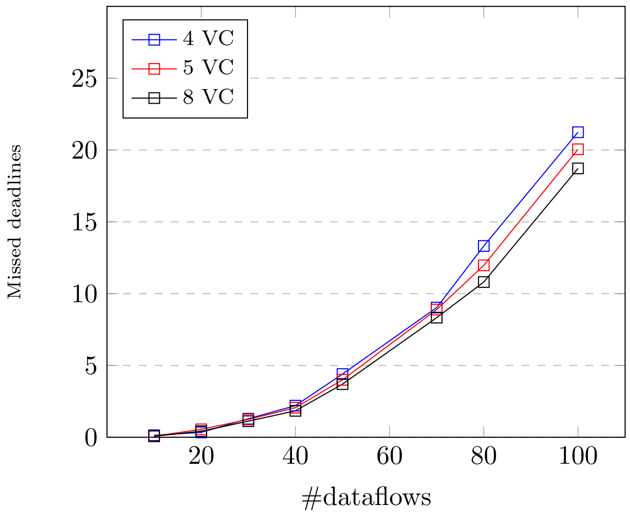

FIG. 5.9. *16×16 NoC.*

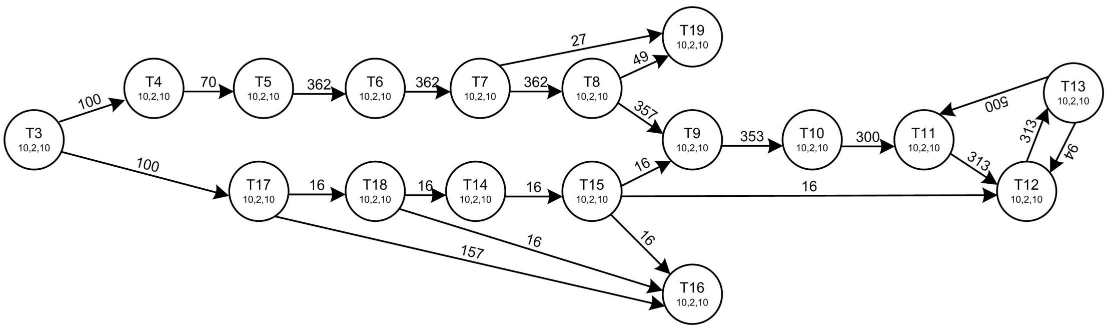

FIG. 5.10. *VOPD tasks graph.*

[6] JANTSCH. *Networks on Chip.* Springer US, New York, NY, USA, 2004.
[7] INTEL. *Single-Chip Cloud Computer (SCC),* 2010.
[8] CORPORATION TILERA. *Tile Processor Architecture Overview for the TILEPro Series,* 2013.
[9] F. BONIOL. *New Challenges for Future Avionic Architectures.* In Modeling Approaches and Algorithms for Advanced Computer Applications, Studies in Computational Intelligence, pages 1–1. Springer, Cham, Switzerland, 2013. doi: 10.1007/978-3-319-00560-7_1.
[10] L.M. NI AND P.K. MCKINLEY. *A survey of wormhole routing techniques in direct networks.* Computer, 26(2):62–76, February 1993.
[11] N. KAVALDJIEV, G. J. M. SMIT, AND P. G. JANSEN. *A virtual channel router for on-chip networks.* In IEEE International SOC Conference, 2004. Proceedings., pages 289–293, Santa Clara, CA, USA, USA, September 2004. IEEE.
[12] G. DIMITRAKOPOULOS, A. PSARRAS, AND I. SEITANIDIS. *Microarchitecture of Network-on-Chip Routers.* Springer-Verlag New York, New York, NY, USA, 1 edition, 2015.
[13] H. SONG, B. KWON, AND H. YOON. *Throttle and preempt: a new flow control for real-time communications in wormhole networks.* In Proceedings of the 1997 International Conference on Parallel Processing (Cat. No.97TB100162), pages 198–202, Bloomington, IL, USA,, August 1997. IEEE.
[14] S. BALAKRISHNAN AND F. OZGUNER. *A priority-driven flow control mechanism for real-time traffic in multiprocessor networks.* IEEE Transactions on Parallel and Distributed Systems, 9(7):664–678, July 1998.
[15] S. L. HARY AND F. OZGUNER. *Feasibility test for real-time communication using wormhole routing.* IEE Proceedings - Computers and Digital Techniques, 144(5):273–278, September 1997.
[16] B. KIM, J. KIM, S. HONG, AND S. LEE. *A real-time communication method for wormhole switching networks.* In 1998 International Conference on Parallel Processing, 1998. Proceedings, pages 527–534, Minneapolis, MN, August 1998. IEEE.
[17] 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 '08, pages 161–170, Washington, DC,

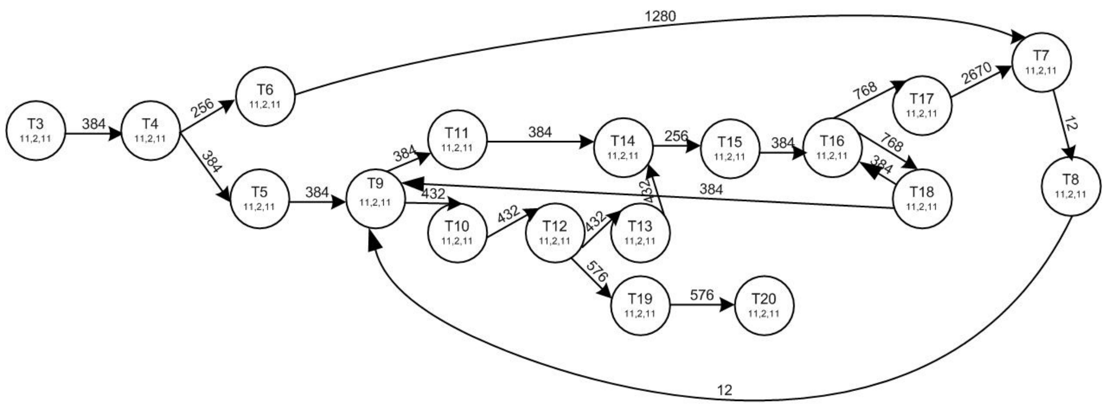

Fig. 5.11. *MPEG4 tasks graph.*

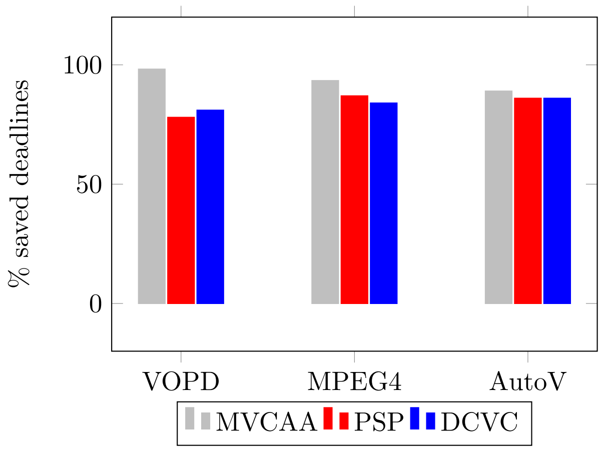

Fig. 5.12. *Percentage of saved deadlines on a 8×8 2D mesh NoC, 8 VC.*

USA, 2008. IEEE Computer Society.

[18] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings. *Applying new scheduling theory to static priority pre-emptive scheduling.* Software Engineering Journal, 8(5):284–292, September 1993.

[19] A. Mello, L. Tedesco, N. Calazans, and F. Moraes. *Virtual Channels in Networks on Chip: Implementation and Evaluation on Hermes NoC.* In 18th Symposium on Integrated Circuits and Systems Design, pages 178–183, Florianolpolis, Brazil, September 2005. IEEE.

[20] Z. Shi and A. Burns. *Real-Time Communication Analysis with a Priority Share Policy in On-Chip Networks.* In 2009 21st Euromicro Conference on Real-Time Systems, pages 3–12, Dublin, Ireland, Ireland, July 2009. IEEE.

[21] B. Nikolič, H. I. Ali, S. M. Petters, and L. M. Pinho. *Are Virtual Channels the Bottleneck of Priority-aware Wormhole-switched NoC-based Many-cores?* In Proceedings of the 21st International Conference on Real-Time Networks and Systems, RTNS '13, pages 13–22, Sophia-Antipolis, France, 2013. ACM.

[22] B. Nikolič and S. M. Petters. *EDF as an arbitration policy for wormhole-switched priority-preemptive NoCs #x2014; Myth or fact?* In 2014 International Conference on Embedded Software (EMSOFT), pages 1–10, Jaypee Greens, India, October 2014. IEEE.

[23] B. Nikolič, S. Tobuschat, L. Soares Indrusiak, R. Ernst and A. Burns. *Real-time analysis of priority-preemptive NoCs with arbitrary buffer sizes and router delays* In Real-Time System Journal , 55(1):63-105,2019(Springer)

[24] B. Sudev and L.S. Indrusiak. *PFT : A low overhead predictability enhancement technique for non-preemptive NoCs.* In 2013 IFIP/IEEE 21st International Conference on Very Large Scale Integration (VLSI-SoC), pages 314–317, Porto Alegre, Brazil, October 2013. IEEE.

[25] B. Sudev, , and L.S. Indrusiak. *Low overhead predictability enhancement in non-preemptive network-on-chip routers using Priority Forwarded Packet Splitting.* In 2014 9th International Symposium on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), pages 1–8, Montpellier, May 2014. IEEE.

[26] J. Duato, S. Yalamanchili, and N. Lionel. *Interconnection Networks: An Engineering Approach.* Morgan Kaufmann

Publishers Inc., San Francisco, CA, USA, 2002.

[27] Z. Shi, A. Burns, and L. S. Indrusiak. *Schedulability Analysis for Real Time On-Chip Communication with Wormhole Switching:.* International Journal of Embedded and Real-Time Communication Systems, 1(2):1–22, 2010.

[28] P. Kundu. *On-Die Interconnects for Next Generation.* In Workshop on On- and Off-Chip Interconnection Networks for Multicore Systems, California, 2006. Intel.

[29] T. Bjerregaard and S. Mahadevan. *A Survey of Research and Practices of Network-on-chip.* ACM Comput. Surv., 38(1):1–51, June 2006.

[30] M. Dehyadgari, M. Nickray, A. Afzali-kusha, and Z. Navabi. *Evaluation of pseudo adaptive xy routing using an object oriented model for noc.* In The 17th International Conference on Microelectronics., December 2005.

[31] M. A. Meghabber, A. Aroui, L. Loukil, A. El Hassan Benyamina, K. Benhaoua, and T. Djeradi. *A flexible network on-chip router for data-flow monitoring.* In 2017 5th IEEE International Conference on Electrical Engineering, pages 1–6. IEEE, October 2017.

[32] E. Bini. *Measuring the Performance of Schedulability Tests.* Real-Time Systems, 30(1-2):129–154, May 2005.

[33] S. J. Filho, A. Aguiar, F. G. de Magalhães, O. Longhi, and F. Hessel. *Task model suitable for dynamic load balancing of real-time applications in NoC-based MPSoCs.* In 2012 IEEE 30th International Conference on Computer Design (ICCD), pages 49–54, Montreal, QC, Canada, September 2012. IEEE.

[34] S. Murali and G. De Micheli. *Bandwidth-constrained mapping of cores onto NoC architectures.* In Automation and Test in Europe Conference and Exhibition Proceedings Design, volume 2, pages 896–901 Vol.2, Paris, France, February 2004. IEEE.

[35] D. Milojevic, L. Montperrus, and D. Verkest. *Power dissipation of the network-on-chip in a system-on-chip for MPEG-4 video encoding.* In 2007 IEEE Asian Solid-State Circuits Conference, pages 392–395, South Korea, November 2007. IEEE.

[36] S. Maatta, L. S. Indrusiak, L. Ost, L. Moller, J. Nurmi, M. Glesner, and F. Moraes. *Validation of executable application models mapped onto network-on-chip platforms.* In 2008 International Symposium on Industrial Embedded Systems, pages 118–125, France, June 2008. IEEE.

[37] Q. Xiong, F. Wu, Z. Lu, and C. Xie *Extending real-time analysis for wormhole nocs.* In IEEE Transactions on Computers, 66(09) pages 1532–1546, 2017. IEEE.

[38] Frederic Giroudot and Ahlem Mifdaoui *Buffer-Aware Worst-Case Timing Analysis of Wormhole NoCs Using Network Calculus* In 2018 IEEE Real-Time and Embedded Technology and Applications Symposium, Porto, Portugal, April 2018.

Edited by: Dana Petcu
Received: April 4, 2019
Accepted: June 5, 2019