
Proceedings Paper:
Burns, Alan orcid.org/0000-0001-5621-8816, Soares Indrusiak, Leandro orcid.org/0000-0002-9938-2920, Smirnov, N. et al. (1 more author) (2020) A Novel Flow Control Mechanism to Avoid Multi-Point Progressive Blocking in Hard Real-Time Priority-Preemptive NoCs. In: 26th IEEE Real-Time and Embedded Technology and Applications Symposium:Proceedings. 26th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 2020), 21-24 Apr 2020 IEEE , AUS .

A Novel Flow Control Mechanism to Avoid Multi-Point Progressive Blocking in Hard Real-Time Priority-Preemptive NoCs

A. Burns, L. S. Indrusiak, N. Smirnov, J. Harrison
Department of Computer Science,
University of York, United Kingdom

***Abstract*—The recently uncovered problem of multi-point progressive blocking (MPB) has significantly increased the complexity of schedulability analysis of priority-preemptive wormhole networks-on-chip. While state-of-the-art analysis is currently deemed safe, there is still significant inherent pessimism when it comes to considering backpressure issues caused by downstream indirect interference. In this paper, we attempt to simplify the problem by considering a novel flow control protocol that can avoid backpressure issues, enabling simpler schedulability analysis approaches to be used. Rather than construct the analysis to fit the protocol, we modify the protocol so that effective analysis applies. We describe the changes to a baseline wormhole router in order to implement the proposed protocol, and comment on the impact on hardware overheads. We also examine the number of routers that actually require these changes. Comparative analysis of FPGA implementations show that the hardware overheads of the proposed NoC router are comparable or lower than those of the baseline, while analytical comparison shows that the proposed approach can guarantee schedulability in up to 77% more cases.**

I. Introduction

Whenever any general resource control protocol is considered for use in the real-time domain it is necessary to develop analysis that provides safe upper bounds for worst-case behaviour. Unfortunately for many protocols that were not originally designed for real-time behaviour the necessary analysis can be both complex and inherently pessimistic; this is the case with the wormhole NoC protocol. Xiong *et al.* [33] have shown that wormhole NoCs with priority-preemptive arbitration can suffer from multi-point progressive blocking (MPB), which makes the calculation of packet latency upper bounds much harder than previously expected. Well cited works on this problem such as those by Shi and Burns [29] and by Kashif and Patel [16] did not consider MPB and were shown to be optimistic [34]. Subsequent works [34][13][24] were able to formulate latency upper bounds that are safe even under MPB scenarios, but those bounds are significantly higher than the ones obtained with previous (and now known to be unsafe) analyses. Such inflation of the upper bounds is not only caused by the MPB effect itself, but also by some pessimism that had to be introduced in order to formulate the problem in a way that it can be understood and solved.

Recent works have tried to address MPB, proposing novel concepts for link arbitration and flow control to avoid [22] or control backpressure [31]. While both concepts could be useful for avoiding MPB in general off-chip wormhole networks, the papers do not address implementation issues that can prevent their application to NoCs (for instance, the need for global wires).

In this paper, we propose a novel flow control protocol that completely prevents MPB effects, aiming to avoid inflated latency upper-bounds. In essence, we start with an effective and intuitive form of analysis [29] and derive a protocol that conforms to that analysis with minimum pessimism. To that end, we propose a new router architecture that uses the memory available in its local tile as temporary storage for incoming packets that cannot be immediately forwarded to their destinations (i.e. because their desired output port is used by another packet flow). We discuss in detail the proposed router design implementing the new flow control protocol, and evaluate its impact on real-time schedulability as well as its hardware overheads. Unavoidably, our solution poses challenges to the local tile memory management, which we discuss in this paper but leave a complete solution and respective implementation as future work.

The paper is organised as follows: we provide background to justify our approach (Section II), describe our contribution, argue that the proposed changes to the wormhole protocol are sufficient to eliminate MPB effects (Section III) and quantify the improvements on schedulability of real-time applications by comparing the proposed approach against the state-of-the-art baseline (Section IV). In Section V we then show that not all routers require to be modified in order to eliminate MPB. Unlike other approaches aiming to avoid MPB, ours is implementable using standard NoC signalling protocols between neighbouring routers and requires no global wires. We show implementation results in Section VI, including a comparison with a priority-preemptive wormhole NoC. Limitations, outstanding issues and future work required to address them are discussed in Section VII, and conclusions are provided in Section VIII.

II. Background

A. Wormhole networks

Wormhole switching [20] is a flow control protocol that provides a good trade-off between performance and buffering overheads. Each packet in a wormhole network is divided into a number of fixed size *flits*, each of which is usually transmitted in parallel via a number of wires that encode a

single data item plus various flow control signals. The first flit of a packet (header flit) holds the packet size and the routing information. As the header advances along the specified route, the remaining flits follow in a pipelined way. If the header flit encounters a link already in use, it is blocked until the link becomes available. In this situation, because network nodes have finite buffering capabilities, the second flit will then be blocked by the first one, and so on, until all flits stall in a process known as *backpressure*. All flits of the packet will then remain buffered in the routers along the packet route until the header is released, so the pipelined transmission can continue. The smaller the buffers on each router, the larger the number of routers that will have to store a given packet in a blockage scenario. If there is not enough buffer space distributed over routers in the packet route, the backpressure will propagate back to the packet sender, preventing it from injecting further flits into the network.

Since a packet can be stored by several routers and occupy multiple links at a time, the potential congestion over the network is increased. This makes it harder to predict the time it takes for a given packet to cross the network, because many of the links along its route may be blocked by other packets. This is not the case in store-and-forward (SAF) switching, where each packet uses only one link at a time, or in virtual cut-through (VCT), where packets are only stored in the router where they experience blocking [7]. In on-chip networks, wormhole is preferred over SAF or VCT because the possibility of small buffers is attractive due to limited overheads in silicon area and energy dissipation [3]. In order to cope with the difficulties to predict packet latencies in wormhole NoCs, several arbitration mechanisms were proposed using resource sharing policies such as time-division multiplexing [8] and prioritised virtual channels (VCs) [4], [29]. The first approach tries to avoid latency interference between packets by reserving link bandwidth to each packet flow. The second approach allows packets to interfere with each other but aims to quantify the upper bounds of that interference over each packet's latency. In this paper, we focus on the second approach since it is work conserving (does not reserve resources) and more flexible (does not require exact knowledge of packet sizes or injection periods, only upper bounds on the former and lower bounds on the latter). In particular, we choose priority-preemptive arbitration at flit level. We described this in detail in the next subsection.

B. Router architecture

Priority-arbitrated network-on-chip routers were first used in the QNoC architecture proposed by Bolotin *et al.* [4], relying on virtual channels (VCs) to prevent head-of-line blocking between packets of different priorities: high priority packets can preempt the transfer of low priority ones in case of contention for the same output port. VCs are usually implemented as a FIFO buffer per priority level, therefore imposing area and energy overheads, which make QNoC-like architectures less attractive in domains that are not performance-sensitive. However, previous work has shown that 4-16 VCs per port [21]

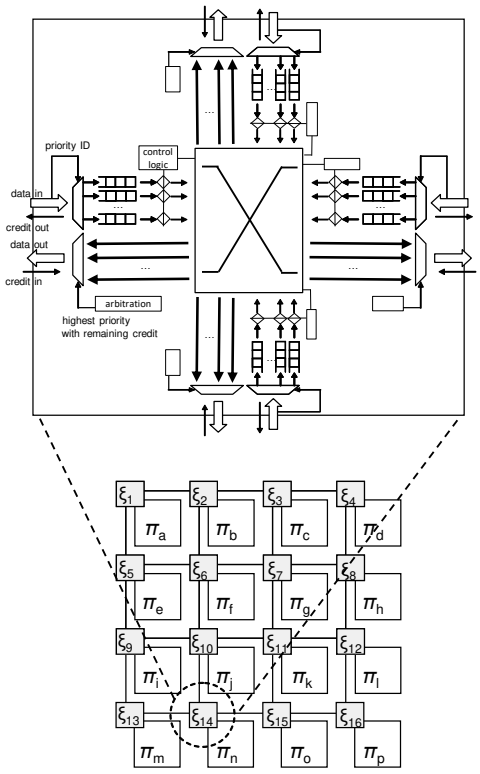

Fig. 1. 2D-mesh network-on-chip and detail of a router with priority-driven virtual channels

and 2-position FIFO buffers per VC [13] are ideal from the performance-predictability point of view while imposing acceptable overheads.

QNoC routers use credit-based flow control [3] to ensure data is only forwarded to output links when there is enough buffer space to hold it in the downstream router. The original QNoC architecture assumed buffering at the input ports of the router, which are connected to output ports via a crossbar, preventing packets routed to different outputs from interfering with each other's performance. Such an approach has been used in several works addressing performance-sensitive and real-time many-cores [29][12][15], has been extended to support mixed-criticality traffic [5][14], and has been modified to support output buffering and multiplexed input-output connections (which are respectively referred as *outq* and *inq-1* architectures in [34]).

In this paper, we use the original input-buffered QNoC architecture as our baseline, as shown in Figure 1. Virtual channels are shown as FIFO buffers multiplexed according to the packet's priority once they flow from each input port. The header of each packet provides the router with the network destination it aims to reach (distributed routing) or its desired output port (source routing). In either case, that information is used to control the crossbar and connect the packet's VC to the correct output port. A priority-preemptive arbitration mechanism on each output port then keeps forwarding data from the input VCs connected to it, always giving precedence to the one with the highest priority that has credit (i.e. buffer space in the downstream router).

Figure 1 also shows an overview of the complete NoC, including a processing tile attached to each router. They are depicted as simple white rectangles, but in practice each tile may include one or more cores, caches or scratchpads, local memory, hardware accelerators, etc. It also includes a network interface (NI), which manages the communication between the tile and the NoC. The NI connects to the NoC router through a bidirectional pair of input and output ports with credit-based flow control, same as the ones connecting routers to each other. So in the case of NoCs with 2D-mesh topologies, routers will have 3 (in corner routers), 4 (in edge routers) or 5 (in middle routers) pairs of input and output ports, one of them connecting the router to its tile (usually called local port, while inter-router ports are referred to by their direction: north, south, east or west). The router depicted in detail in Figure 1 is router ξ_{14}, which is in the lower edge of a 2D-mesh NoC, so the four pairs of ports shown in the figure are its local, west, east and north ports. In this paper, we aim to explore local ports and their connection to the local tiles to improve time predictability in such NoCs.

C. Schedulability model

We model a wormhole NoC such as the one depicted in Figure 1 as a set of tiles $\Pi = \{\pi_a, \pi_b, \ldots, \pi_z\}$, a set of routers $\Xi = \{\xi_1, \xi_2, \ldots, \xi_m\}$, and a set of unidirectional links $\Lambda = \{\lambda_{a1}, \lambda_{1a}, \lambda_{12}, \lambda_{21}, \ldots, \lambda_{zm}, \lambda_{mz}\}$. The input and output ports of a router are the endpoints of the incoming and outgoing links that connect it to neighbouring routers and its local tile.

To model the traffic load injected to the network, we define a set Γ of n real-time traffic-flows (or just *flows* for short) $\Gamma = \{\tau_1, \tau_2, \ldots \tau_n\}$. Each flow τ_i gives rise to a potentially unbounded sequence of *messages* in a similar way to that of a task giving rise to a series of jobs. An alternative term for message is *packet*. The flow has a set of properties and timing requirements which are characterised by a set of attributes: τ_i = (P_i, C_i, T_i, D_i, J_i^D, J_i^I, π_i^s, π_i^d). We assume that all the flows which require timely delivery are either periodic or sporadic. The lower bound interval on the time between releases of successive messages is called the period (T_i) for the flow. The maximum basic network latency (C_i) is the maximum duration of transmission latency when no flow contention exists [29]. Each real-time flow also has a relative deadline (D_i) which is the upper bound restriction on network latency. In this work we assume $D_i \leq T_i$. Any flow can suffer two forms of release jitter; J_i^D is direct jitter and denotes the maximum deviation of successive message releases from the flow's period. The other form of jitter, J_i^I, is the indirect interference the flow may suffer [29], [30]. Here τ_i suffers interference from some flow τ_j which itself suffers interference from flow τ_k. But this interference is not accounted for as τ_k does not directly interfere with τ_i.

In addition to these parameters, each flow has a priority P_i; the value 1 denotes the highest priority and larger integers denote lower priorities. It also has a source and destination tile (π_i^s and π_i^d). The usual X-Y routing [20] is assumed and hence the source and destination values fully define the route the flow will take. For example, with a 3x3 grid, a source (3,3) and destination (2,1), the flow will pass from (3,3) to (2,3) to (2,2) to (2,1). It follows that, with deterministic routing, only the header flit must carry the address of the destination node.

D. Schedulability analysis

Schedulability analysis for priority-preemptive wormhole networks has existed for more than two decades, even before networks-on-chip were a reality. Its aim is to check, for a set of sporadic flows of fixed-priority packets, whether the latency of all packets can be upper-bounded, and such upper-bounds are less than their respective deadline. Works by Mutka [19] and Hary and Ozguner [9] in the mid 1990s used classic fixed-priority schedulability analysis while considering the entire path of a given packet as a single shared resource, so that its worst-case latency bound can be found by analysing the direct interference caused by higher priority packets that share at least one link of their route. Kim *et al.* [17] also identified and accounted for the effects of indirect interference, which happens when two packet flows do not share any network links but one of them can still have an impact on the latency bounds of the other (by affecting the temporal behaviour of a third packet flow which shares links with both of them).

Most schedulability analyses developed for priority-preemptive NoCs (such as Shi and Burns [29] in 2008 and Kashif and Patel [16]) were based on those foundations until the discovery in 2016 of multi-point progressive blocking [33]. MPB arises when indirect interference happens downstream from the shared link(s) through which direct interference is caused (due to backpressure effects caused by finite buffering per router). Xiong *et al.* [33] identified this backpressure using simulations; they showed that downstream indirect interference can sometimes cause a single packet of some flow τ_j to directly interfere on packet τ_i by more than the amount of time that τ_j would take to traverse an unloaded network (i.e. τ_j's basic latency C_j). That scenario disproved one of the assumptions made by Shi and Burns, and Kashif and Patel, and showed that a flit of a packet of τ_j may interfere multiple times on a packet of τ_i over multiple shared links. Such scenario can arise when τ_j (1) suffers interference from a packet τ_k that does not interfere with τ_i and (2) shares links with τ_k downstream from the links it shares with τ_i. As MPB was underestimated in the earlier analysis it can leave to optimistic predictions of schedulability. More details on MPB are to be found in [34] and [13]).

Rate-limiting and buffer management approaches have been used to reduce backpressure (for example [10], [2], [11], [32], [25]), but they only focus on reducing the average packet latency and are not usable to bound worst-case latency (and therefore do not explicitly model MPB effects).

Since its discovery, MPB has been safely modelled (i.e. no known optimistic counter-examples) by three forms of analyses (in increasing order of tightness): Xiong *et al.* [34] (which corrected their unsafe attempt reported in [33]), Indrusiak *et al.* [13] and Nikolic *et al.* [24]. Despite the consistent increase

in tightness, all of them are still significantly more conservative than previous analyses that did not model MPB. For instance, Indrusiak *et al.* [13] show that for some utilisation ranges Shi and Burns [29] analysis can deem schedulable twice as many scenarios as the analysis they proposed, and four times as many as the analysis proposed by Xiong *et al.* [34]. That shows the magnitude of the potential improvements that can be achieved if MPB could be completely avoided and simpler analyses could be used.

The fact that it took 20 years to identify the MPB problem is witness to the fact that scenarios in which the simple analysis was demonstratively optimistic are very rare events. Counter-examples were not easily constructed [33], and general flow-set simulations did not illustrate the problem. Nevertheless, to guarantee real-time behaviour the applicable analysis must be sufficient (i.e. safe in all situations). In this paper we achieve this by retaining the simpler analysis but removing the potential for optimism by proposing a novel flow control mechanism, which in turn requires changes to typical NoC router architectures.

Shi and Burns Analysis: For completeness we briefly describe the analysis of Shi and Burns [29], which is sufficient for the proposed flow control mechanism and respective NoC router changes.

For flow τ_i its worst-case response time is given by:

$$R_i \;=\; C_i \;+\; \sum_{\tau_j \in \mathbf{Shp}(i)} \left\lceil \frac{R_i + J_j^D + J_j^I}{T_j} \right\rceil C_j, \qquad (1)$$

where $\mathbf{Shp}(i)$ is the set of higher priority flows that share any link with τ_i. This equation is solved using the standard techniques for solving recurrence relations (i.e. fixed point iteration). Once solved, a flow is deemed schedulable if $R_i \leq D_i$; the full set of flows is schedulable if all its flows are schedulable.

The value of J_j^D comes from whatever agent inputs the flow into the network. For example, if the source of τ_j is a periodic task executing on π_j^s with period 20ms and response time 15ms then τ_j could arrive at the router anytime within an interval of 15ms; hence $J_j^D = 15$ [1]; The value of the other release jitter, J_j^I, is given by [29]:

$$J_j^I \;=\; R_j - C_j. \qquad (2)$$

Indirect interference jitter occurs when a flow (τ_z) with higher priority than τ_j (and therefore higher than τ_i) shares a link with τ_j but not with τ_i. Flow τ_i is not directly impacted by τ_z, but it is effected indirectly via τ_j.

Each flow is analysed in turn, from the highest priority (to compute the Rs) to the lowest (using computed Rs from higher priority flows).

III. Proposed flow control protocol

The main contribution of this paper is a novel flow control protocol that avoids MPB effects, and a novel router architecture implementing that protocol. Our aim is to avoid backpressure, which is a common feature in wormhole switching, and the key cause of MPB [13]. Backpressure happens when a packet is blocked in a NoC router, i.e. its desired output link is used by another packet of higher priority, so its incoming flits are buffered in that router until the buffer is full. At that point, the next upstream router will not be able to forward flits to the congested router anymore, so it would start buffering the flits itself until its buffers are full, then the buffering will start in the next upstream router, and so on. In off-chip networks, backpressure has been avoided by using store-and-forward or virtual cut-through switching [20]. Those mechanisms require buffers that are large enough to hold a complete packet, which is impractical in NoCs due to the overheads in silicon area and power dissipation.

Recent works have also tried to address MPB, proposing novel protocols for link arbitration and flow control to avoid backpressure [22][31]. None of these papers address implementation issues or provide a prototype implementation. In both papers, the proposed protocols require signalling that goes beyond neighbouring routers, and possibly across the whole network. While acceptable in off-chip wormhole networks, such an approach is impractical in on-chip implementations, as they require global wires (which are one of the critical problems that NoCs are supposed to avoid in the design of on-chip systems, due to their excessive energy dissipation and difficulties in timing closure and routing [6]).

In this paper, we propose a novel flow control protocol that avoids MPB by preventing backpressure without resorting to global wires. Instead, we rely on the communication between a router and its local network interface. The proposed protocol avoids backpressure by temporarily ejecting flits from the network into the local tile whenever their desired output link is blocked. Once a flit has been ejected to the local tile, all subsequent flits from its packet will also be ejected. The ejection of a flit uses dedicated wires that are part of the link connecting the router's local output port and the local network interface, which we refer as an ejection sink, or simply *sink*. Once ejected via a sink, flits must be stored in the local tile until they can be re-injected to the network (i.e. when their desired output port becomes available) via dedicated local link wires that we refer as re-injection sources. That way, we can avoid backpressure in the same way as in SAF or VCT, but without requiring dedicated buffers in the router.

The implementation of the proposed protocol in a NoC router can actually simplify some of the components that are present in the priority-preemptive wormhole router described in subsection II-B. Firstly, the new router only requires a single two-position FIFO buffer per input port, i.e. to enable the pipelined forwarding of flits (one position to receive a flit, another to forward a flit). Separate buffers for distinct virtual channels are no longer necessary in input ports connected to neighbouring routers, as only one flit will be received at a time at each input port. By the end of the cycle, that flit will either be forwarded to the desired downstream router, or it will be ejected from the network. Likewise, credit-based flow control mechanisms are no longer needed. Notice that the notion of virtual channels is still maintained (and implemented

as additional wires on the link between routers, in order to indicate the virtual channel ID of each flit, exactly as in the baseline architecture).

On the other hand, the proposed ejection and re-injection of flits requires a number of changes to the NoC router and network interface. We hereby describe the proposed changes, and use their index number (e.g. change 2) to refer to each of them in the remainder of this paper.

- **change 1:** the control logic of the router crossbar must be changed so that if an input port does not receive a grant arbitration signal from its desired output port, it must be connected to the local output port instead, so that it can use its sink to forward the flit at the head of its FIFO queue to the local tile;
- **change 2:** the network interface should be aware of any flits temporarily stored in the tile's memory and their desired output ports, so that it can request arbitration to those output ports and forward flits through them whenever they are free;
- **change 3:** the arbitration logic of each output port must be changed so that it gives precedence to arbitration requests from the local port (issued by the NI) attempting to forward flits temporarily stored in the local tile, over new flits coming from a non-local port in the same VC. New incoming flits will instead follow the same path as their predecessors through the local tile, so that the flit ordering of the packet is not changed and backpressure does not occur. The change in arbitration must only be enforced for requests by the same VC (i.e. same priority), with no change to the usual preemptive arbitration for packets of different priority levels.

Let us describe a typical scenario to show how the proposed mechanism changes the behaviour of the NoC router. We assume a packet P arriving via VC 3 to the west input port, which must be routed to a non-local output port (the north port, in this example). Whenever that output port is busy forwarding a flit from an input VCs of higher priority (from a packet Q that was either there before the arrival of P, or arrived anytime during its transmission and thus preempted it), the flits of P will be redirected to the local tile via local output port, one by one, as long as the desired output port is busy (as specified by change 1 above) . If the desired output is free, flits of P are forwarded through it to the next downstream router (exactly as in the baseline architecture). Flits that are sent to the local tile will be stored there, and will trigger the NI to request arbitration on their behalf to their desired output port (as specified by change 2 above). The desired output port of packet P (the north port in this example) will then receive two arbitration requests from VC 3 (one from the remaining flits from P in the west input port, and another one from the NI over the local input port on behalf of the temporarily stored flits in the local tile) and possibly more from the higher priority packet(s) that prevented P from using the output port in the first place (Q in this example). The north output port will not grant arbitration to any of the requests from VC 3 until all flits of higher priority VCs are served. Every time arbitration is denied to VC 3 in the west input port, one more flit is sent out via local output port to the local memory. Once all flits of higher priority VCs are served, there are two possibilities: either the whole of P has been stored in the local memory and the local tile's NI is the only one requesting arbitration (which will then be obviously granted and the stored flits will be on their way in FIFO order) or there will be part of P still coming in via south input port (and thus competing with the local tile's NI for arbitration). Change 3 is necessary because of the latter possibility: if the highest priority arbitration requests are simultaneously coming from a local and a non-local input port, precedence should be given to the local port so it can forward the locally-stored flits that arrived earlier than those in the non-local port, maintaining the original order of flits of the packet.

Figure 2 shows two stages of the described scenario. On the left-hand side, it shows flits of P being sent to the local tile because the north port is busy forwarding flits of Q. On the right-hand side, it shows the situation after Q has finished its transmission and the arbitration of the north output is given to the local input driven by the NI. In that exact moment, VC3 is requesting arbitration from inputs west and local respectively for flits 2 and 6 of packet P. According to change 3, the local port will be granted arbitration to forward flit 2, while flit 6 will be denied arbitration and will be forwarded to the local memory.

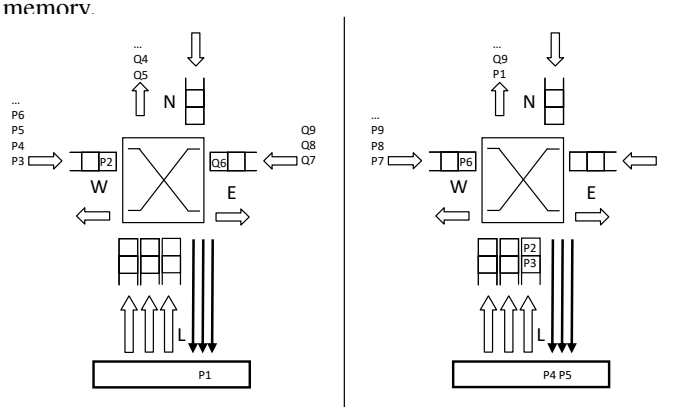

Fig. 2. Two stages of the scenario describing the proposed changes to NoC router, local link and network interface

One more change to the architecture is required before we can completely avoid backpressure. Even with the architectural upgrades proposed so far, it is still possible for a packet to be blocked and thus prevent the reception of its flits from upstream routers: when a flit needs to be ejected, but is unable to acquire arbitration to the local port. In that case, the flit will stay in the input buffer and will prevent other flits from moving forward, causing backpressure. This situation could happen when a higher priority packet has also been redirected to the local port (i.e. its desired output port was also busy, or the local tile is actually its final destination). We therefore propose the following change:

- **change 4:** the local links should be widened so that they can simultaneously input/output flits from/to all non-local

ports of the router, effectively eliminating the need for the arbitration unit. Additional control logic and wires are needed to notify the NI of valid flits to be stored, their respective VCs and their intended destination ports (as specified by change 2).

In Figure 2, change 4 is represented by the multiple arrows coming in and out of the local port of the router. Each black arrow coming out of the router represents a distinct sink, each of them used to prevent backpressure by ejecting flits from each of the router's non-local input ports (i.e. up to four sinks; in Section V we consider how this number can be reduced). Likewise, white arrows coming back into the router represent the re-injection sources (which can also be minimised, but we leave that as future work).

Change 4 eliminates contention when flits access the local tile. Assuming that the NI can always consume all incoming flits, the problem of backpressure in the NoC can be completely eliminated. That assumption is not unreasonable, as it is typically made by all congestion-free NoCs (such as several TDM architectures [28]). In the proposed approach, it is likely that the NI will have to use the local memory tile to store the ejected flits. Such integration of the NI with the memory controller of the local tile has already been investigated in [26], enabling direct memory access from incoming flits from the router as required in change 4. In our approach, however, it is possible that multiple flits can be ejected (i.e. one from each input port) and/or re-injected (i.e. one to each output port) at the same time, and that poses additional design challenges to the network interface and tile memory (which we will revisit in Section VII).

Besides contributing to the elimination of backpressure, and consequently of MPB, change 4 also provides additional bandwidth to the NoC, reducing the severity of local link bottlenecks (as we will show in Section IV). On the other hand, those benefits come with a cost. Despite the elimination of the local port arbitration unit, change 4 still imposes energy and area overheads with the additional logic and wires required for the widening of the local links. Previous research on wider NoC flits show that the router area has a greater-than-linear growth with the size of the flit, as the buffer area has a linear growth rate and the crossbar switch has a quadratic growth rate [18]. In our proposal, however, the overheads will be far less significant, since we are only changing the width of the local link (and not all of them, as in [18]) and we are not increasing the amount of buffering or the width of the links in the crossbar.

IV. Evaluation of the impact of the proposed approach on NoC schedulability

With the elimination of backpressure, there is no need to account for MPB effects when calculating worst-case packet latency. This means that in this case, the state-of-the-art analysis by Nikolic *et al.* [24] becomes too pessimistic, and analyses such as Shi and Burns [29] become safe. To quantify the impact of the proposed approach on the schedulability of NoCs, we have performed a large-scale comparison of different analyses applied to synthetic packet flows mapped to priority-preemptive NoCs with and without the proposed approach. Below, we describe the experimental setup in detail.

We perform separate experiments on NoC platforms of two different sizes: 5x5 and 10x10 (i.e. 25 and 100 routers). For each platform size, we compare the baseline QNoC architecture with an architecture implementing the proposed protocol described in Section III. The analysis of all architectures assumes implementations with 2D-mesh topology, deterministic XY routing, 2-position FIFO buffers per VC and operating frequency of 100 MHz.

To evaluate schedulability of flow sets over each platform, we generate sets of sporadic packet flows, randomly map each of the flows in the set onto the platform, and then apply different schedulability analyses to test whether each set is fully schedulable, i.e. if all its packet flows will deliver all their packets by their respective deadlines even in a worst-case scenario. All packet flows on each set are based on the following characteristics: periods uniformly distributed between 0.5 s and 0.5 ms, maximum packet lengths uniformly distributed between 128 and 4096 flits, rate-monotonic priority assignment, and deadlines equal to the respective periods.

To show how schedulability changes with the increase of the communication load handled by the NoC, we generate multiple flow sets with increasing number of flows in each of them. Given the random nature of the mapping and of the selection of periods and packet lengths, we generate 100 different flow sets for each load level (i.e. amount of flows in the set) and plot how many of them are fully schedulable in Figures 3 to 6.

Each of the lines on the plots in Figures 3 and 5 represents a different platform and a different analysis, as follows:

- **SotA:** flows mapped onto baseline NoC platform, flow set schedulability evaluated with the state-of-the-art MPB-aware analysis by Nikolic *et al.* [24].
- **SotAUp:** flows mapped onto baseline platform with widened local links (as described in change 4), flow set schedulability evaluated with the state-of-the-art MPB-aware analysis by Nikolic *et al.* [24]. The only difference w.r.t the **SotA** case is that, due to change 4, no interference will ever happen over the local port of the routers.
- **SB:** flows mapped onto baseline platform, flow set schedulability evaluated with unsafe MPB-unaware analysis by Shi and Burns [29].
- **SBUp:** flows mapped onto NoC platform with the proposed protocol and all 4 changes to router and NI, flow set schedulability evaluated with analysis by Shi and Burns [29].

The first conclusion we can take away from the experiment is that despite of all the improvements on tightness obtained by Nikolic *et al.* [24] over the preceding MPB-aware analyses [13][34], it still provides low levels of schedulability (SotA line in Figures 3 and 5) due to the difficulties of handling the corner cases imposed by MPB. If no changes to the NoC are possible, this is still the best we can do though.

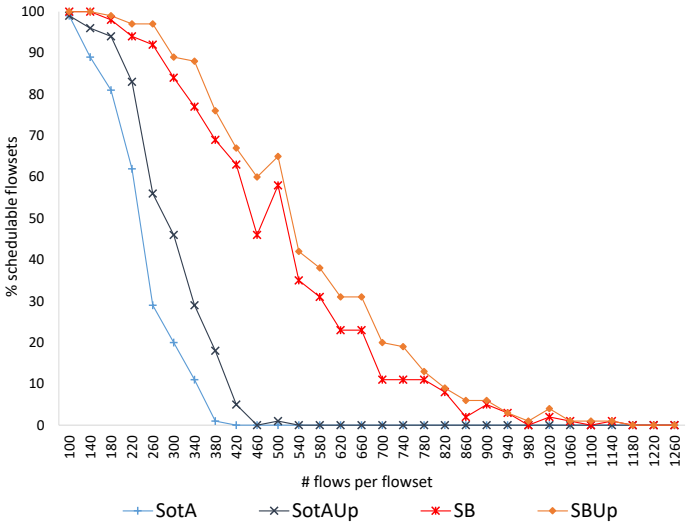

Fig. 3. Schedulability results for baseline and proposed platforms with 5x5 topologies. Each point represents the percentage of fully schedulable flow sets (out of a set with 100 flow sets, each of them with the number of flows indicated over the X-axis).

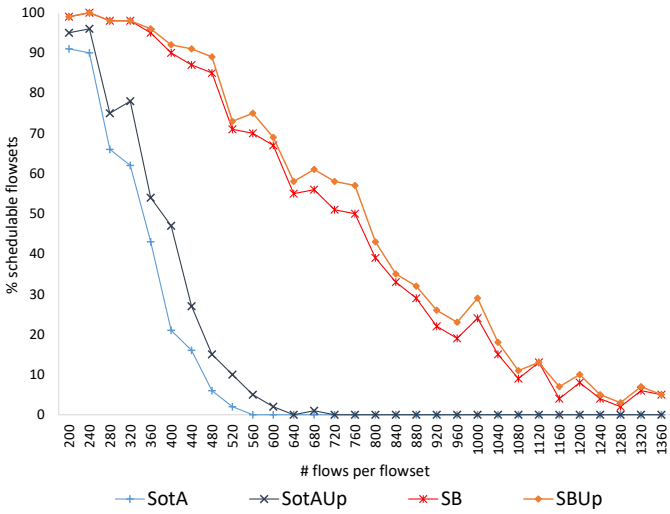

Fig. 5. Schedulability results for baseline and proposed platforms with 10x10 topologies. Each point represents the percentage of fully schedulable flow sets (out of a set with 100 flow sets, each of them with the number of flows indicated over the X-axis).

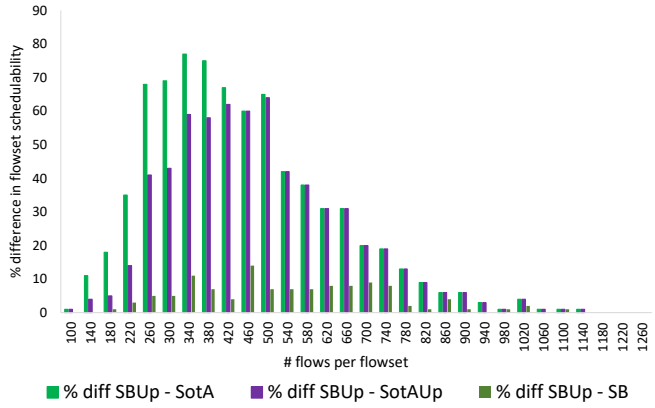

Fig. 4. Data for 5x5 showing difference in flow set schedulability

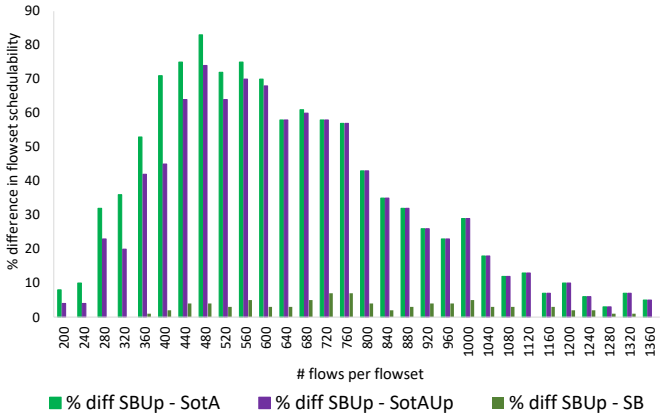

Fig. 6. Data for 10x10 showing difference in flow set schedulability

The second conclusion is that the widening of the local links proposed in change 4 can provide some improvement (SotAUp line in Figures 3 and 5) due to the increased bandwidth and reduced interference when packets leave and arrive to the local tile. If the arbitration rules are not changed (as proposed in changes 1 to 3), MPB is still an issue and so are the difficulties of handling its corner cases, as described above.

The third and main conclusion is that the proposed flow control protocol, which completely prevent MPB and thus allow us to analyse schedulability using Shi and Burns [29], gives us a significant increase in schedulability (SBUp line in Figures 3 and 5). Such increase reaches up to 77% against the baseline, and up to 74% against a baseline with widened local links from change 4 (which is a fairer comparison). Furthermore, due to the additional local link bandwidth enabled by change 4, the proposed approach provides even better schedulability than what could previously be obtained by the unsafe application of the Shi and Burns analysis on the baseline platform (SB line in Figures 3 and 5) (up to 14%).

Finally, we can see that the improvements due to change 4 are more prominent in the 5x5 architecture because the reduced number of cores boosts the importance of contention-free access to the local tile (as more flows will cross a local link, on average). Similarly, the improvements due to changes 1 to 3 are more prominent in the 10x10 platform, as the packet routes will be longer and therefore more prone to MPB [13].

V. Customised Widening of Local Links

The previous section has demonstrated the benefits that are obtained if MPB is eliminated. The cost of this performance gain is the changes that must be made to the router – as explained in Section III. The support for four ejection sinks per router proposed in change 4, aiming to widen the local port so that all non-local ports can simultaneously redirect traffic to

local memory, requires extra physical resources and increased energy consumption. However, not all routers require four sinks. Indeed, some will require none at all (and hence will require no widening of the local output link), as the flows going through them would not experience MPB effects. In this section, we derive the necessary conditions for an arbitrary router to require a sink for each of its input ports, and then evaluate the number of sinks required for a range of system configurations.

A. Requirements for a sink

For an input port to require a sink at a particular router, the port's incoming link must be used by two flows of different priority that diverge at the router (i.e. leave it via different output ports). Moreover, the higher priority flow must suffer interference from an even higher priority flow on a link elsewhere in the network that is not used by the lower priority flow. These necessary relationships can be modelled as follows:

Let τ, τ_1 and τ_2 be flows; let λ and λ_1 be links, and ξ be a router. Define functions $Pri(\tau)$ to deliver a flow's priority; $Use(\tau, \lambda)$ to be a predicate that is true if flow τ uses link λ; and $Des(\tau, \xi)$ to be the destination of flow τ from router ξ.

Link λ into router ξ will require a sink in ξ if and only if:

$$\exists \tau_1, \tau_2, \lambda, \lambda_1, \xi \bullet \; Use(\tau, \lambda) \; \wedge \; Use(\tau_1, \lambda) \; \wedge$$
$$Pri(\tau_1) > Pri(\tau) \; \wedge \; Des(\tau_1, \xi) \neq Des(\tau, \xi) \; \wedge$$
$$Use(\tau_1, \lambda_1) \; \wedge \; Use(\tau_2, \lambda_1) \; \wedge \; \neg Use(\tau, \lambda_1) \; \wedge$$
$$Pri(\tau_2) > Pri(\tau_1)$$

This implies that flows τ and τ_1 arrive at the router along the same link but leave by different links – and τ_1 has the higher priority. Moreover, there is a link (λ_1) that is used by τ_1, an additional arbitrary flow τ_2 (with higher priority than τ_1), but is not used by τ. Because τ_1 suffers interference (from τ_2) elsewhere in the network it can induce additional interference on τ at ξ; hence τ needs to be buffered on ξ, and so link λ needs a sink on ξ.

These conditions are unlikely to be widespread; if for example two flows share, say, 5 links then only in the last router will there be a possible MPB problem. And then only if the higher priority flow suffers interference from another even higher priority flow elsewhere in the network. In the following we examine how often are sinks actually required.

B. Evaluation

In this section we investigate the number of routers that typically require changes if MPB is to be avoided, and also the scale of such changes (i.e. how many sinks are needed, and therefore how much wider the local link needs to be). This evaluation is again undertaken by creating a large number of flow sets and randomly mapping them onto two different NoC architectures (once more, 5x5 and 10x10 mesh topologies). The method of generating the flow sets is the same as that used in Section IV.

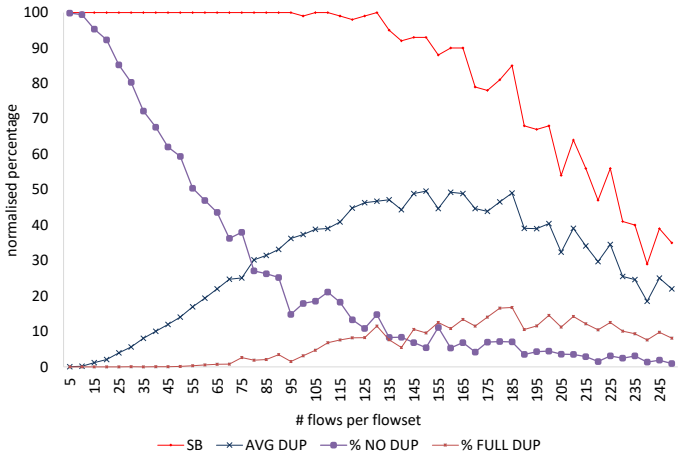

Fig. 7. Number of routers than require modification, against increasing load on the system for 5x5 NoC

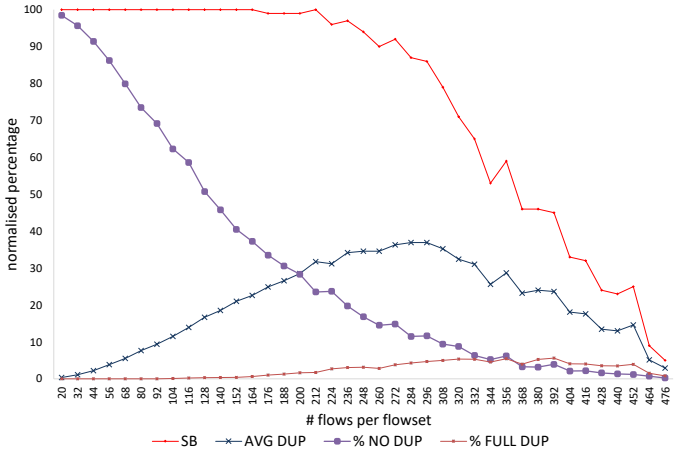

Fig. 8. Number of routers than require modification, against increasing load on the system for 10x10 NoC

For each random mapping we first establish if the flow set is schedulable by Shi and Burns [29] analysis. For those that are schedulable we then use the requirement defined above to check every link and every route to determine if it needs a sink. Figures 7 and 8 show the results of this evaluation. Each figure shows:

- **SB**: The schedulability curve (how many flow sets are deemed schedulable by the analysis reviewed in subsection II-D).
- **NO DUP**: The number of routers that require no widening of the local link (this is expressed as a percentage of the total number of routers in each network – 25 in Figure 7 and 100 in Figure 8).
- **FULL DUP**: The number (percentage) of routers that require the maximum widening of the local link (i.e. four sinks).
- **AVG DUP**: This is the average number of sinks required (per router) – here 100% on Y axis means 4, 50% 2 etc.

Note that towards the right-hand side of the graphs the number of schedulable flow sets is reduced and hence all the sink requirement counts reduce (as only schedulable flow sets are analysed).

The results of this evaluation are clear: there is never a need for all routers to have sinks for all their input ports. In all experiments, the average number of sinks per router never exceeds 2. If we consider the point on these graphs where 50% of the flow sets are schedulable then we see that, for the 5x5 NoC only 12% of the routers require sinks for all inputs and the average number of sinks is less than 2. For the 10x10 NoC, at the 50% schedulability bar, less than 5% of the routers require all four sinks, and the average number of sinks is around 1.

C. Implications for NoC design

The results from the targeting evaluations have a number of practical ramifications.

1) If the NoC is general purpose and not configurable then although all four sinks would be present on the fabric, a significant number could be disabled which would reduce the energy consumption of the NoC.
2) If the NoC is configurable (e.g. implemented using FPGA technology) then once an application is mapped to the NoC then the real requirement of each link on each router would be known and sinks provided only where needed.
3) If the application is flexible, in terms of where tasks (and hence flow sources and destinations) are allocated, and/or which routes are available (between sources and destinations) then as part of the mapping exercise – which may make use of search techniques such as Genetic Algorithms (GAs) as in [27] – routers that have, say, only two sinks could form the building blocks of the NoC platform.

In the latter case, as the targeting evaluation implies that on average less than two sinks per router are needed it is reasonable to ask if indeed two sinks are sufficient (in almost all cases) when there are sufficient degrees of freedom in terms of task allocation and flow routing. Experimental work to investigate this possibility is beyond the scope of this paper, but will form the basis of future work along other aspects covered in the Section VII.

VI. Implementation Results

To demonstrate that the proposed protocol is straightforward to implement and to evaluate the likely hardware overhead within a NoC router, we have designed and implemented the proposed router using a Xilinx FPGA through the Vivado design flow (including Vivado HLS). For the sake of fair comparison, we have also implemented the baseline wormhole router through the same design flow, and replicated in both routers the functionality that is common between them (e.g. header flit processing, routing). In both cases, we made the router designs customisable in terms of ports (to discriminate corner, edge and central routers in mesh NoCs, and to support other topologies) and virtual channels (to support applications

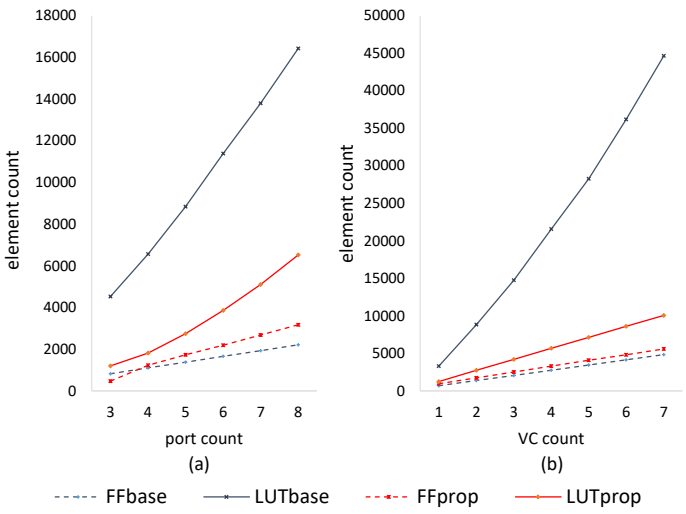

Fig. 9. FPGA usage in flip-flop and look-up tables for baseline and proposed NoC router

with distinct requirements when it comes to real-time guarantees). The flit width was set to 64 bits. In the case of the proposed router, we implemented the maximum widening of the local links, i.e. to allow for the simultaneous ejection of flits from all four input ports, as this is the maximum amount of overhead that the proposed approach would ever require (even though in practice this may be never needed, as we show in Section V). On the other hand, the baseline router was implemented in its leaner form without any local link widening.

In Figure 9, we show for both baseline and proposed router how FPGA usage (in terms of required number of flip-flops and look-up tables) scales with the number of (a) ports and (b) virtual channels. The plotted results are obtained from Vivado's out-of-context synthesis, which provides a fairer comparison by excluding potential optimisations done by mapping and place-and-route tools when targeting the design to a specific FPGA device. When scaling the number of ports, all designs are implemented with 2 VCs, and when scaling the number of VCs we fix all designs to be implemented with 5 ports. The results show that the proposed design is superior than the baseline, and scales far better due to the simpler buffering and crossbar structures requiring a lower usage of look-up tables. It requires a slightly higher number of flip-flops in larger routers, but this is because unlike the baseline it does not use any block RAMs to implement its buffers (while the baseline uses between 12 to 30 block RAMs, not plotted in Figure 9). We then performed full synthesis of a complete 2x2 NoC with routers implementing the proposed approach, mapped it to a Zynq-7000 device set to be clocked at 250MHz, and obtained the dynamic power dissipation of the NoC interconnect while scaling the number of VCs (Table I, error margin of 1mW). We also obtained the dynamic power dissipation of the complete FPGA (which also implemented network interfaces, packet sources and sinks, testing circuitry

TABLE I
DYNAMIC POWER DISSIPATION (MW) OF THE PROPOSED 2X2 NOC INTERCONNECT IMPLEMENTED ON A ZYNQ-7000 DEVICE

VCs	1	2	3	4	5	6	7
proposed NoC	28	47	59	67	82	98	111
total	1432	1449	1462	1471	1485	1501	1514

and two integrated ARM Cortex-A9 cores). The obtained figures for the NoC represented less than 10% of the total dynamic power dissipation of the device.

VII. LIMITATIONS AND FUTURE WORK

This paper focuses on a network protocol that aims to eliminate MPB, and provides a solution to this problem within the scope of a NoC router architecture. The proposed solution, however, makes two assumptions regarding the operation of the platform beyond the scope of the NoC router, namely:

1) the NI will always have the oldest ejected flits ready for re-injection when their desired output ports become free;
2) the NI can always consume all ejected flits.

The first assumption is not critical, and as long as re-injection happens in bounded time our solution would still hold (but the equations presented in subsection II-D would have to be modified to account for the re-injection delay).

The second assumption, on the other hand, is key to the elimination of backpressure (and of MPB) and therefore critical to the proposed solution. As mentioned in Section III when we discussed the introduction of change 4, the NI will often have to use the memory of the local tile as temporary storage of ejected flits for the second assumption to hold (unless the number of ejected flits is small enough to fit in NI buffers, which we assume to be the exception rather than the rule). In the case of routers with a single sink (which would be the most common scenario, as shown in Section V), the access to the tile memory could be solved with a NI design such as the one presented in [26]. For the cases where multiple sinks are needed, the problem becomes more complex as multiple ejected flits arriving to the NI at each cycle may require tile memory to support multiple simultaneous reads and writes, or require the tile to operate at a faster frequency than the network. In either case, we leave the detailed design and evaluation of the NI and memory management solutions as future work.

Besides the future work that is required outside the scope of the NoC router, there are router design alternatives that have been identified by this work but not yet exploited or evaluated. In change 4, we proposed the widening of both outgoing and incoming local links of the NoC router, but MPB can be completely avoided only with the widening of the outgoing local links (i.e. sinks). Backpressure only occurs if flits cannot advance from one router to the next, and not when the source is a tile. If only outgoing links are widened and re-injection happens over an unmodified link (i.e. no widening, no additional re-injection sources), the overheads in area and energy dissipation are potentially smaller. In that case, the improvements in schedulability would not be as good as reported in this paper because flits that are temporarily stored in the local tile will have to compete for arbitration upon re-injection to the NoC. In other words, even packet flows that did not interfere with each other in the baseline architecture would suffer or cause interference during re-injection. This imposes a compelling trade-off between schedulability gains and implementation overheads, and it would be interesting to know how significant the impact on schedulability would be and whether that impact could be mitigated with smart mapping and routing heuristics.

Also related to task mapping and flow routing, we aim to investigate how to optimise them so that we can minimise the number of sinks required on each router (following the findings from Section V). Finally, the proposed router design also opens new avenues of research from the point of view of the analysis, and additional work could be done to assess whether existing improvements to Shi and Burns analysis (such as [16] or [23]) are able to improve tightness when analysing the upgraded architecture in the same way they did to the baseline.

VIII. CONCLUSIONS

In this paper, we have presented a novel flow control protocol to avoid the problem of multi-point progressive blocking (MPB) in priority-preemptive networks-on-chip. By exploiting the memory available in the local tile and the widening of the local link, we could prevent backpressure and therefore avoid MPB completely. We showed that the proposed protocol results in significant benefits in schedulability of sporadic packet flows sent over the network, as simpler analyses can be used to evaluate the worst-case response time of such flows. Over a comprehensive series of experiments, we showed that the proposed approach can guarantee schedulability to up to 77% more cases than a typical priority-preemptive wormhole baseline.

To enable packets to access the local tile memory without the possibility of backpressure, we proposed a number of changes to a priority-preemptive wormhole NoC router architecture that is widely used in previous works. Unlike other approaches addressing MPB, ours does not rely on global wires and is fully compatible with typical on-chip implementation processes. To demonstrate the feasibility of our approach and to evaluate its hardware overheads, we designed and compared FPGA implementations of the proposed architecture and the baseline. We showed that the implementation overheads of the proposed approach are comparable or lower than those of the baseline (and much lower as the router scales up the number of ports or virtual channels). Furthermore, the implementation overhead of the proposed NoC can be minimised even further if routers can be configured according to their individual needs.

The contribution presented in this paper focuses only on the NoC router operation and analysis. It makes assumptions about the operation of the NI and the tile memory management system, and additional research is required to devise designs

that make sure those assumptions hold in a concrete implementation. Additional avenues of research opened by this work include the exploitation of the trade-off between hardware overheads and schedulability when widening local links for flit re-injection, as well as the optimisation of task mapping and packet routing to minimise local link widening for flit ejection.

REFERENCES

[1] N.C. Audsley, A. Burns, M.F. Richardson, and A.J. Wellings. Incorporating unbounded algorithms into predictable real-time systems. *Computer Systems Science and Engineering*, 8(2):80–89, 1993.

[2] D. U. Becker, N. Jiang, G. Michelogiannakis, and W. J. Dally. Adaptive backpressure: Efficient buffer management for on-chip networks. In *Proc. IEEE 30th International Conference on Computer Design (ICCD)*, pages 419–426, 2012.

[3] T. Bjerregaard and S. Mahadevan. A survey of research and practices of Network-on-chip. *ACM Comput Surv*, 38(1):1, 2006.

[4] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny. QNoC: QoS architecture and design process for network on chip. *J Syst Arch*, 50(2-3):105–128, 2004.

[5] A. Burns, J. Harbin, and L.S. Indrusiak. A Wormhole NoC Protocol for Mixed Criticality Systems. In *IEEE Real-Time Systems Symposium*, pages 184–195, 2014.

[6] W. J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *Proc. Design Automation Conference (DAC)*, pages 684–689, 2001.

[7] W.J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[8] K. Goossens, J. Dielissen, and A. Radulescu. AEthereal network on chip: concepts, architectures, and implementations. *IEEE Design & Test of Computers*, 22(5):414–421, 2005.

[9] S. L. Hary and F. Ozguner. Feasibility test for real-time communication using wormhole routing. *IEE Proceedings CDT*, 144(5):273–278, 1997.

[10] J. Hu and R. Marculescu. Application-specific buffer space allocation for networks-on-chip router design. In *Proc. IEEE/ACM International Conference on Computer Aided Design (ICCAD)*, pages 354–361, 2004.

[11] J. Hu, U. Y. Ogras, and R. Marculescu. System-level buffer allocation for application-specific networks-on-chip router design. *Proc. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 25(12):2919–2933, 2006.

[12] L. S. Indrusiak. End-to-end schedulability tests for multiprocessor embedded systems based on networks-on-chip with priority-preemptive arbitration. *J Syst Arch*, 60(7):553–561, 2014.

[13] L. S. Indrusiak, A. Burns, and B. Nikolic. Buffer-aware bounds to multi-point progressive blocking in priority-preemptive nocs. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 219–224, 2018.

[14] L. S. Indrusiak, J. Harbin, and A. Burns. Average and Worst-Case Latency Improvements in Mixed-Criticality Wormhole Networks-on-Chip. In *Proc. ECRTS*, 2015.

[15] H. Kashif, S. Gholamian, and H. Patel. SLA: A Stage-Level Latency Analysis for Real-Time Communication in a Pipelined Resource Model. *IEEE Trans Comp*, 64(4):1177–1190, 2015.

[16] H. Kashif and H. Patel. Buffer Space Allocation for Real-Time Priority-Aware Networks. In *RTAS Symposium*, pages 1–12, 2016.

[17] B. Kim, J. Kim, S. Hong, and S. Lee. A real-time communication method for wormhole switching networks. In *Int Conf on Parallel Processing*, pages 527–534, 1998.

[18] J. Lee, C. Nicopoulos, S. J. Park, M. Swaminathan, and J. Kim. Do we need wide flits in Networks-on-Chip? In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 2–7, 2013.

[19] M. W. Mutka. Using rate monotonic scheduling technology for real-time communications in a wormhole network. In *Workshop on Parallel and Distributed Real-Time Systems*, pages 194–199, 1994.

[20] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *Computer*, 26(2):62–76, 1993.

[21] B. Nikolic, 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 *Proc. of the 21st International Conference on Real-Time Networks and Systems*, RTNS '13, pages 13–22. ACM, 2013.

[22] B. Nikolic, R. Hofmann, and R. Ernst. Slot-Based Transmission Protocol for Real-Time NoCs - SBT-NoC. In *ECRTS Conf*, pages 26:1–26:22, 2019.

[23] B. Nikolic, L. S. Indrusiak, and S. M. Petters. A Tighter Real-Time Communication Analysis for Wormhole-Switched Priority-Preemptive NoCs. *arXiv:1605.07888 [cs]*, 2016.

[24] B. Nikolic, S. Tobuschat, L. S. Indrusiak, R. Ernst, and A. Burns. Real-time analysis of priority-preemptive NoCs with arbitrary buffer sizes and router delays. *Real-Time Syst*, 55(1):63–105, 2019.

[25] U. Y. Ogras and R. Marculescu. Prediction-based flow control for network-on-chip traffic. In *proc. 43rd ACM/IEEE Design Automation Conference*, pages 839–844, 2006.

[26] M. Ruaro, F. B. Lazzarotto, C. A. Marcon, and F. G. Moraes. DMNI: A specialized network interface for NoC-based MPSoCs. In *IEEE Int Symposium on Circuits and Systems (ISCAS)*, pages 1202–1205, 2016.

[27] M. N. S. M. Sayuti and L. S. Indrusiak. A function for hard real-time system search-based task mapping optimisation. In *IEEE Int Symposium on Real-Time Distributed Computing (ISORC)*, pages 66–73, 2015.

[28] M. Schoeberl, F. Brandner, J. Spars, and E. Kasapaki. A statically scheduled time-division-multiplexed network-on-chip for real-time systems. In *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip (NOCS)*, pages 152–160, 2012.

[29] Z. Shi and A. Burns. Real-time communication analysis for on-chip networks with wormhole switching. In *Proc. of the 2nd ACM/IEEE International Symposium on Networks-on-Chip(NoCS)*, pages 161–170, 2008.

[30] Z. Shi and A. Burns. Improvement of schedulability analysis with a priority share policy in on-chip networks. In *17th International Conference on Real-Time and Network Systems (RTNS)*, pages 75–84, 2009.

[31] N. Ueter, G. von der Brueggen, J. J. Chen, T. Mitra, and V. Venkataramani. Simultaneous progressing switching protocols for timing predictable real-time network-on-chips. *arXiv:1909.09457v1 [cs.DC]*, 2019.

[32] S. Umamaheswari, , D. Meganathan D, and J. Raja Paul Perinbam. Runtime buffer management to improve the performance in irregular network-on-chip architecture. *Sadhana*, 40(4):1117–1137, 2015.

[33] Q. Xiong, Z. Lu, F. Wu, and C. Xie. Real-Time Analysis for Wormhole NoC: Revisited and Revised. In *GLSVLSI Symposium*, pages 75–80, 2016.

[34] Q. Xiong, F. Wu, Z. Lu, and C. Xie. Extending Real-Time Analysis for Wormhole NoCs. *IEEE Trans Comput*, 66(9), 2017.