Providing Throughput Guarantees in Mixed-criticality Networks-on-Chip

Sebastian Tobuschat and Rolf Ernst
Institute of Computer and Network Engineering
Technische Universität Braunschweig, Germany
Email: {tobuschat, ernst}@ida.ing.tu-bs.de

***Abstract*—Future mixed-criticality systems must handle a growing variety of traffic requirements, ranging from safety-critical real-time traffic to bursty latency-sensitive best-effort traffic. Additionally, safety standards (e.g. ISO 26262) require sufficient independence among different criticality levels for mixed-criticality systems. Networks-on-Chip (NoCs), as a scalable and modular interconnect, are used as a promising solution for such systems. Hence, a NoC must provide performance isolation for safety-critical traffic and low latency for best-effort traffic at the same time. This paper presents a run-time configurable NoC design enabling throughput guarantees for selected traffic streams with reduced adverse impact on the performance of best-effort traffic. In contrast to existing approaches, we prioritize best-effort over guaranteed throughput traffic and only switch priorities when required, providing sufficient performance isolation among different criticality levels. We show that the overhead implementing our approach is affordable. And through an experimental evaluation, we show that the approach reduces the adverse effects through strict prioritization on best-effort applications.**

I. Introduction

Safety critical and dependable embedded systems play an important role in our daily life. For example, modern vehicles integrate over 100 electronic control units (ECUs). Due to the ever increasing demand for high performance together with low energy consumption and size, multicore systems, as known from general-purpose computing, are adopted by the safety-critical embedded market. The integration of multiple cores in a chip to a multiprocessor system on chip (MPSoC) offers the possibility to consolidate multiple functions or ECUs, which previously had been distributed and isolated by external buses. This consolidation of functions with different overall importance leads to mixed-criticality multicore systems [1]. In this context, a critical function is essential for the safety of the system. Therefore, this function is developed with high diligence and so the behaviour (i.e. timing) is well specified and tested. For non-critical functions the confidence in the characteristics is lower, i.e., the possibility that the function deviates from the specification is higher.

Networks-on-Chip (NoC), as a modular interconnect, are used as a promising solution for MPSoCs, due to their scalability and performance. In a NoC resources, such as the output ports of the routers, are shared among the different functions and safety-classes. Hence, applications of different safety levels will inevitably compete with each other in a NoC for resources. This resource sharing couples the execution behaviour across cores and, thus, impacts non-functional properties like timing, which are of particular interest in safety-critical environments. Safety standards explicitly mention this problem in context of mixing different criticalities (e.g. *sufficient independence* IEC 61508 [2]).

One approach to tackle the problem of mixed-criticality is to develop all functions to the highest relevant safety level. This leads to higher development costs and lower system utilization. Another approach is to provide *sufficient independence* through quality-of-service (QoS) mechanisms. The challenging part of this approach is to efficiently utilize system resources while providing a bounded and feasible interference. This typically leads to a trade-off between providing real-time guarantees for certain applications and performance for the others, as well as the introduced overhead by the quality-of-service mechanisms.

Contribution: In this paper we present a novel, run-time configurable NoC design enabling throughput guarantees with reduced impact on latency for best-effort traffic. In contrast to existing approaches, we prioritize best-effort (BE) over the guaranteed throughput (GT) traffic in NoC routers and only switch priorities when required. This enables us to exploit the throughput and latency slack of critical applications, while providing sufficient independence among different criticality levels w.r.t. timing properties. Additionally, we allow virtual channel sharing between concurrent GT streams to reduce the number of needed (virtual) channels and hence buffer space.

The rest of the paper is structured as follows. In Section II we provide an overview of related work. This is followed by the description of our design in Section III. We then evaluate the design in Section IV and conclude in Section V.

II. Related Work

There exist various packet-switched networks-on-chip providing quality-of-service (QoS) for mixed criticality systems that can be categorized by how they enforce service guarantees. One group uses time-division multiple-access (TDMA) to limit the interference between applications, e.g., [3]–[5]. These rely on a pre-allocation of time-slots in the network and provide strong isolation. However, the static timing schedules and slot assignments typically lead to some inflexibility and underutilized systems.

The second group uses more dynamic scheduling approaches, to increase the flexibility and utilization of the system. For

978-1-5386-4034-0/17/$31.00 ©2017 IEEE

this, they combine interface-based design and system analysis [6], [7]. Traffic sources in the network provide well defined interfaces and thus introduce a known and limited interference to the system. Based on this idea more dynamic and work-conserving QoS mechanisms can be constructed based on source rate limiting and dynamic scheduling. In [8], for example, the authors propose the QNoC architecture. It uses four traffic classes and a fixed priority scheme to arbitrate between packets of the different classes in the switches. The critical traffic has a higher priority than best-effort, providing isolation for the critical traffic. In the Mango NoC [9] switches consist of two parts, a best-effort (BE) and a guaranteed service (GS) switch, and implement virtual channels. The GS streams are prioritized over BE and a fair-sharing arbitration is used between multiple GS streams. The latency of a message is bounded and mainly depends on the number of VCs sharing a particular connection and the selected arbitration policy.

The authors of [10] introduce Globally-Synchronized Frames (GSF), for providing guaranteed QoS in terms of throughput and latency bounds. GSF coarsely divides the time into frames and introduces a scheme similar to *earliest deadline first* scheduling based on these frames. For this, each QoS packet from a source is assigned a frame number indicating the desired delivery time. Packets with an early delivery time are prioritized in the routers.

Besides the prioritization of GS traffic, some approaches try to improve the performance of BE traffic. For this, BE can have the same or even a higher priority than GS traffic. To limit the interference, i.e., to still guarantee a minimum throughout for GS, mechanisms to dynamically adapt the priorities are introduced. The authors of [11] present a protocol for priority-preemptive VC arbitration, guaranteeing that all (critical) packets will arrive by their deadlines. It uses runtime monitoring at the network interfaces (NI) to check whether critical traffic stays within a predefined behaviour (i.e. message sizes and inter-arrival time). If an injecting NI detects a deviation from this behaviour, routers on the desired path switch to a critical state, in which the priority of best-effort traffic is lowered below all critical traffic and hence BE can only use idling ports of a router. The authors of [12] introduce *backsuction*, in which BE is prioritized over GS traffic. This scheme uses rate limiters at the destinations nodes, which return a control signal upstream along the transmission path. If a router recognizes a too low throughput of critical traffic, it increases the priority of the critical stream at the upstream router. This check is based on a threshold value in the routers, to denote when a buffer underflow occurs. As this approach uses a rate limiter at the destination node, it relies on a (BE) packet before the transmission to setup the path. Additionally, to correctly route the control signal upstream, this scheme allows only a single ongoing transmission in a (virtual) channel.

The authors of [13] introduce a *fluid meter* in each router, which is used together with a (3 bit) header extension denoting the throughput requirement of a packet/stream. Based on these two values a router can dynamically de-/increase the priority of the packet. When GS requirements are satisfied, less VCs must be allocated to (multiple) GT streams (i.e. as they don't have to leave the router immediately). These VCs can be used by BE, to not sacrifice performance of BE. However, the approach increases the packet header and introduces an additional FSM in each router.

In summary, most of today's NoC architectures do not meet the requirements on isolation, low hardware requirements, and high system utilization at the same time. TDM based architectures introduce static overhead due to the static time schedule, reducing the performance in most cases. Most of the dynamic QoS approaches favour safety-critical over best-effort (BE) traffic (e.g. strict prioritization), thus reducing the BE performance or introduce complex additional logic in the routers. To tackle these problems, we propose a method that only introduces a low overhead in the routers. It exploits the slack of safety-critical applications to increase the BE performance compared to other approaches, while still providing sufficient isolation.

III. Forward Pressure

This section describes the new approach providing throughput guarantees while reducing the buffer requirements and adverse effects of state-of-the-art QoS mechanisms on non-critical applications. The goal is to exploit the throughput and latency slack of safety-critical guaranteed-throughput (GT) traffic to increase the performance for best effort (BE) traffic. Safety-critical applications do not benefit from receiving more resources than needed and thus the slack can be used to schedule other traffic [12], [14], [15]. For this, we give priority to BE traffic for optimal latency and at the same time monitor the progress of GT traffic, to change priorities if needed.

Although the proposed mechanism is not specific to a certain network architecture, we assume a mesh network as a baseline, where every router is connected to up to four neighbouring routers and one client, such as processing elements or memory. The routers use wormhole flow control, i.e., buffer management is performed on equally sized flits, and have a four-stage pipeline. Packets are composed of a head flit, multiple body flits, and a tail flit. Every input port comprises multiple virtual channels (VC) to isolate different criticalities, i.e., there is no VC sharing between different criticalities. We use a "winner-takes-all" arbiter for requests of the same class, which is similar to a round-robin arbiter, but maintains a grant until the end of a packet. This improves average latency, as packets are sent in one piece if possible [16].

The key idea of our approach is to prioritize BE traffic for optimal latency and at the same time monitor the progress of GT traffic, to change priorities if needed. For this, the proposed mechanism comprises up to three elements:

1) a *selective priority arbiter* that uses a *progress monitor* for GT (i.e. the current flit buffer levels) as a decision criterion in the input stages of each network switch,
2) an entity at the source that tags the last packet of a GT connection, and
3) a rate limiter at the source.

If a GT sender is guaranteed to behave as specified (e.g. through the design process), the rate limiters for GT are not needed.

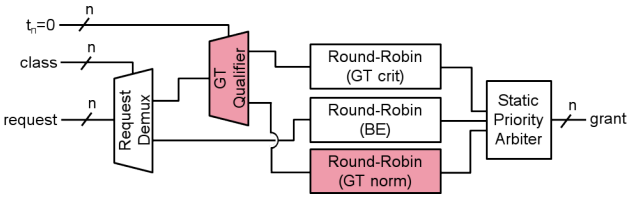

Fig. 1. Selective Priority Arbiter

A. Selective Priority Arbiter

The arbitration logic is located at each output port. It processes the requests from all input ports according with their class signal and the current state of GT. Figure 1 shows a simplified block diagram of the arbitration logic. In the figure, t_n denotes whether an input port with GT traffic needs a higher priority. Based on this signal, a request of a GT stream is forwarded to the corresponding GT arbiter (i.e. *critical* when above the threshold or *norm* when beneath). The *class* signal decides, whether the request results from a BE or GT stream. The *static priority arbiter* then selects the highest priority signal that has requests pending. If t_n is not asserted, the arbiter prioritizes BE, while GT can use empty slots (i.e. not used by BE).

The priority signal t_n is derived from the buffer fill level or the presence of a special tail flit, named *end of transmission (EoT)* flit. For this, the routers have a (configurable) threshold value. When the buffer fill level is above the threshold or the EoT flit is in the buffer, the priority signal is asserted. Routers typically already have a measure for the fill level for the buffers for flow control. We only extend it by an additional threshold, which is then used to construct a priority feedback signal. This threshold might be configurable or static.

In summary, the proposed arbiter selects BE requests, as long as all (requesting) GT channels are beneath their threshold, and a GT request otherwise. If there are no requests of a specific priority, requests from a lower priority are selected. This means that GT is allowed to send if the link is otherwise idle, but also enables BE traffic to use unused reserved GT throughput, avoiding waste of over-allocated throughput.

Contrary to [12], this scheme allows multiple GT streams to share the same VC, as the priority signal must not be forwarded upstream, but is derived locally in each router. However note, that this also allows for possible head of line blocking between GT streams. Hence, only streams where the sink is known to accept traffic should be allowed to share a VC, which can be guaranteed by design or an online control layer [17].

B. Sender Extensions

Besides the selective priority arbiter, the senders must be equipped with the possibility to tag the end of a GT transmission with the EoT flit. Additionally, if the behaviour of GT senders can not be guaranteed by design, rate limiters are needed. Booth mechanisms can be implemented in software or as hardware extensions in the network interface (NI). As backpressure might occur at the injecting interface and router, the source needs sufficiently sized buffers or a stateful rate-limiter. This is needed to catch up a possible backlog at the sender with a temporary higher rate than requested. A simple example for a stateful rate-limit, in this sense, is a token bucket shaper, where the bucket size covers the worst case backlog. The bucket size then allows a burst, where the sender obtains more throughput than initially requested, to catch up an initial too low accepted throughput.

C. Throughput Guarantees

In this section we show that the approach can guarantee a minimum accepted traffic rate for a sender. For this, we derive a lower bound on the minimum service and an upper bound on the backlog at the sender. In general, these values can be obtained utilizing any analysis framework, such as [18]–[20]. In the following, we will utilize the approach of [18], as it can handle backpressure for arbitrary sized buffers, to obtain the minimum accepted throughput for a sender and its backlog. As we only focus on the throughput property (and not latency) and due to limited space, we use a simplified version of the analysis. Note that, while this simplification is sufficient to provide lower bounds on the minimum service, the results become more pessimistic for certain streams. To obtain more tight results, and also the latencies of the transmissions, the per stream analysis from [18] can directly be used when extended for an additional delay through BE blocking.

To derive the worst-case accepted traffic of a sender with the approach from [18], we need to obtain the worst-case waiting time $\hat{B}_p^+$, output blocking B_i^{out}, and buffer backlog b^p.

Definition 1. *Let Θ^k denote the set of all possible mappings of* k *packets to available output ports. Then $\theta, \theta \in \Theta^k$ defines a specific mapping for* k *packets, such that θ denotes for each of the* k *packets the destined output port.*

Definition 2. *The worst-case waiting time $\hat{B}_p^+(q)$ at a router port* p *denotes the time until the port is ready to accept the* q*-th incoming flit. It can be bounded by [18]:*

$$\hat{B}_p^+(q) = \begin{cases} q \cdot C + \max_{\theta \in \Theta^k} \{A_\theta\}, & \text{if } b^p > Q_b \\ 0, & \text{otherwise} \end{cases}$$

$$\text{with} \quad A_\theta = \sum_{j \in \theta} \left\{ B_j^{out}(\hat{B}_p^+(q) - C_i, n) + \hat{B}_{P(j),k+1}^+(n) \right\}, \quad (1)$$

where b^p denotes the worst-case backlog of the port, Q_b the size of the buffer in flits, k *is a limit on the number of packets* q *flits form ($k = \lceil \frac{q}{n} \rceil$), B_j^{out} is the output blocking, and $\hat{B}_{P(j),k+1}^+$ is the waiting time at the next router. Note, that only ports used by GT streams must be accounted for Θ^k.*

This waiting time can only occur if the worst-case backlog of the port exceeds the buffer size ($b_p > Q_b$). If it occurs, we assume the port to be fully backlogged. Hence, to receive q incoming flits, the port must first transmit q flits. For these flits we must account for their transmission time ($q \cdot C$) and the worst-case interference they suffer. For this, the term $\max_{\theta \in \Theta^k} \{A_\theta\}$ obtains the worst-case blocking from each possible mapping

of flits to output ports. These might then experience output blocking and a waiting time at the next router.

Definition 3. *The output blocking B_i^{out} that a stream* i *experiences is bounded by:*

$$B_i^{out}(\Delta t, q) = \psi + \sum_{j \in Out_i} C \cdot \chi + \hat{B}^+_{P(j),k+1}(\chi)$$
$$with \quad \chi = \min\left\{ \left\lceil \frac{q}{n} \right\rceil, \left\lceil \frac{\eta_j^+(\Delta t)}{n} \right\rceil \right\} \cdot n,$$
$$and \quad \psi = \min\left\{ \Delta t \,\middle|\, \sum_{j \in Buf_{P(i)}} \left\{ \eta_i^-(\Delta t) \right\} \geq Q_t \right\} \qquad (2)$$

where n *denotes the maximum packet size in flits, Q_t the threshold, $\eta_j^+(\Delta t)$ ($\eta_j^-(\Delta t)$) the maximum (minimum) number of flits of stream* j *than can arrive in any time interval Δt, $Buf_{P(i)}$ the set of streams sharing the same buffer with* i *including* i*, and Out_i denotes the set of other input ports with GT streams that are mapped to the same output port as* i*. Hence, $j, j \in Out_i$ denotes the cumulative interference of input port* j*.*

In the worst-case, the stream waits until the threshold value is reached (ψ) and then competes with other GT streams for the router resources. Due to wormhole switching, once the scheduler grants access to an output port, no other input port can access this port until the port is released, i.e., the packet is fully transmitted. This is captured by the second term in the *min*-function, which considers that after a head flit from j arrives within the time interval Δt, the whole packet will be served before i. Additionally, due to the round-robin arbitration, each head flit belonging to stream i may only be blocked once by each other input port, where $\lceil \frac{q}{n} \rceil$ is an upper bound on the number of head flits. Each of these head flits can be blocked at most for n flits from each other input port. Moreover, each of the interfering flits then will block stream i for the flit transfer time C and the waiting time these flits experience at the next router.

Based on the worst-case waiting time, we can then define the minimum accepted throughput $\hat{\beta}_p^-$ at a router port as:

$$\hat{\beta}_p^-(\Delta t) = \min\left\{ \Delta t, \max\left\{ n \middle| \hat{B}_p^+(\max(0, n - Q_b)) < \Delta t \right\} \right\}$$
$$with \quad \Delta t \in \mathbb{N}^+. \qquad (3)$$

For simplification we assume that the time Δt is given in multiple of the flit transfer time. The *max*-function selects the highest number of events that can be accepted during a time interval Δt based on the waiting time. For this, only events that can arrive before Δt can be accepted. As the first Q_b flits can be accepted immediately, we only have to account for the waiting time of $n - Q_b$ flits. Additionally, the port can not accept more flits than time slots have passed, covered by the first term of the *min*-function, which only accepts one flit after one (flit) time unit has passed. The minimum accepted traffic for a sender then corresponds to the minimum accepted throughput at the first router port on the path of the sender. And if this service is equal or higher the requested throughput in the long term, the stream is schedulable.

With the known minimum service, we can also derive the backlog b^p at each port p and hence sender as:

$$b^p = \max\left\{ \sum_{i \in Buf_p} \left\{ \eta_i^+(\Delta t) \right\} - \hat{\beta}_p^-(\Delta t) \right\}, \qquad (4)$$

where Buf_p denotes the set of streams sharing port p. The equation compares the minimum service of the port ($\hat{\beta}_p^-$) with the requested service (η_i^+) of all streams. The difference then shows the maximum number of flits that can have arrived but not be transferred. This equation can be used to derive the size of the buffer at the sender or the bucket size of the shaper, to be able to recover from backlog. For this, the buffer or bucket size must be at least the maximum possible backlog at the sender.

IV. Evaluation

In this section, we evaluate our mechanism and compare it to the classic and widely used prioritization scheme (e.g. [8]), and the *backsuction* scheme [12]. We divide the evaluation in two parts. In the first part, we use synthetic workloads to evaluate the basic functioning and certain properties of our mechanism, such as isolation between BE and GT and the influence of different transmission sizes. In the second part, we use memory access and communication traces of general purpose applications, to investigate the performance of the mechanism on realistic workloads. All experiments were carried out with the *OMNeT++* simulation framework and the *HNOCS* library [21] using routers with a four-stage pipeline, four virtual channels (VCs), buffers to store 16 flits in each VC of each input port, and a packet size of four flits.

As a test scenario, we use a 8x8 mesh NoC with XY-routing, in which we denote the corner node on the north west as *node (0,0)* and the corner node on the south east as *node (7,7)* based on their XY coordinates. In the network we have two GT streams, one from node (0,1) to (6,4) requiring 30% of the link throughput and one from (0,2) to (5,4) requiring 20%. Hence, both GT streams overlap with a total requirement of 50% of the link throughput on the shared links.

A. Performance

In the first set of experiments we use synthetic workloads, generated based on average link loads. We use a BE stream sending from node (0,3) to (4,4), hence its whole path is overlapped by the GT streams. All other nodes inject BE traffic to random destinations. We then investigate different mechanisms: round-robin (RR), prioritization of GT (SP), backsuction (BS) and our approach (FP). As backsuction allows no channel sharing between GT, the stream from node (0,1) uses *VC0* and the one from node (0,2) *VC1*, leaving two VCs for BE traffic. For the other mechanisms, both GT streams share *VC0*. Here we additionally differentiate between the case where we allow BE to use two or three VCs, denoted respectively as FP2 and FP3 for FP. Note that in the former case, only three VCs are used, which corresponds to a smaller router design than in the case for BS that uses four VCs (cf. Table I).

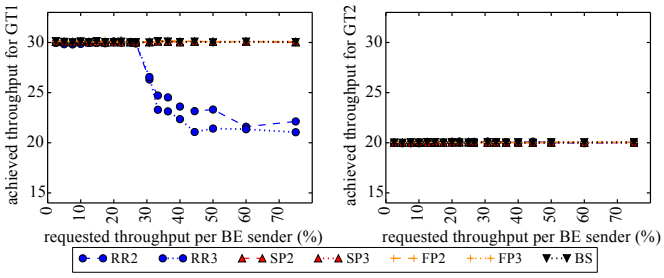

Fig. 2. Achieved GT throughput - periodic packets with 4 flits

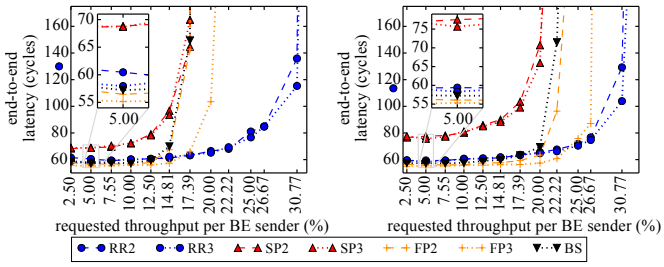

Fig. 3. BE Latency - periodic burst of 1 (left) and 4 (right) packets

Figure 2 shows the received throughput for all approaches over increasing BE load, where all nodes are periodically injecting single packets. As can be seen, all QoS mechanism provide the GT streams with the required throughput. Only in the round-robin (RR) case (i.e. no QoS), the streams drop below the required throughput when the BE load increases, which states the RR approach unsuitable for safety-critical designs.

For the same scenario, Figure 3 shows the latency for BE node (0,3) to (4,4) when the GT streams are sending single packets (left) and bursts of 4 packets (right). As expected, the prioritization leads to a higher latency for BE compared to the RR case. BS and our approach achieve similar latencies than RR for low loads, and hence improve the latency for BE up to 17%. Along with this, the saturation point, at which the latency for BE goes to infinity, can be shifted to higher workloads enabling a higher system utilization, compared to SP. Here, BS and FP2 achieve a similar performance (while FP2 needs only three VCs). And FP3 (with the same number of VCs as BS) achieves a better performance for BE than BS. Additionally, with an increasing burst size, the latency of BE increases for the simple prioritization. For BS and FP the latency increase for BE is less. Hence, for increased burst sizes, BS and FP can lead to a higher performance improvement for BE.

In the second set of experiments we use benchmark workloads to evaluate the performance of the proposed mechanism. For the experiments we obtained traces from the CHStone benchmark suite [22]. The traces were extracted using the Gem5 simulator and an ARMv7-a core with a 32 kB L1 cache and contained 100.000 accesses to the network, where each access can be a direct memory access, communication, or a cache access. The compilation was performed using standard

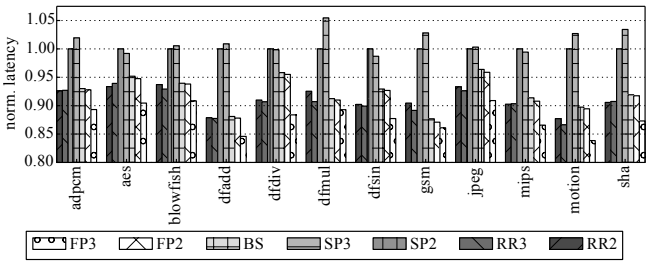

Fig. 4. Normalized BE Latency for various benchmarks from node 3 to 36

gcc compiler (ver. 4.7.3). For a simulation run, we assigned one benchmark to node (0,3) and then generated several random mappings of the benchmarks to the other nodes with random destinations for their traffic. We selected the destinations such that a traffic stream has to pass at least three routers (i.e. no traffic to direct neighbours).

Figure 4 shows the normalized latencies for this scenario. We generated for each listed benchmark 25 different sets of interfering workloads (i.e. random assignment of application to nodes) and a random destination for each sender. We then normalized the latency of the BE sender from node (0,3) to (4,4) to the case of simple prioritization and two VCs for BE (SP2). As can be seen, the results comply with the synthetic results, showing that the dynamic prioritization of our approach can improve the BE performance by up to 16%. Again, our approach leads to similar performance improvements as BS when using less virtual channels or better improvements when using the same number of VCs. Additionally, we can see a dependency on the BE traffic patterns. For example, in the case of the *motion* benchmark, we can see higher improvements than for the case of *adpcm*.

B. Synthesis Results

In this section we briefly present synthesis results for our approach. We implemented and synthesized a *2x2* NoC on a *Virtex-6 LX760 FPGA* using *Xilinx ISE 14.6* with default optimization settings and no special optimizations for the *VHDL* implementation. The device utilization data were collected from the *Module Level Utilization Summary Report* produced by ISE. Note that, as this NoC is not fully connected, each of the four routers has only three input ports fully instantiated. The results for the whole NoC are summarized in Table I. The table compares the used *registers* and *LUTs* for six different implementations. The *RR3* implementation corresponds to a basic round-robin router with 4 virtual channels (i.e. three VCs for BE and one for GT) and a buffer depth of 4 packets for each VC. This was extended in *SP3* to provide one prioritized VC for GT, e.g. *VC0*. In the *DP* design the priority of *VC0* can be changed via a configuration flag from the highest to the lowest priority during run-time. This was extended in *BS* to account for the backsuction signal. And finally, the *FP3* and *FP2* implementations denote our approach, where the priority of *VC0* is dynamically changed by the router based on the

TABLE I
SYNTHESIS RESULTS FOR 2X2 NOC ON VIRTEX-6 LX760 FPGA

Unit	RR3	SP3	DP	BS	FP3	FP2
#Registers	4368	4856	4868	4873	4875	3874
#LUTs	5720	6301	6322	6338	6346	5132

current fill level of the input buffer and the presence of a EoT flit, with respectively four and three VCs.

The synthesis shows, that our approach introduces less than 10% overhead for the used *2x2* NoC compared to a baseline (RR3) router and less than 0.5% compared to the SP3 design when using the same number of VCs. Note that, if throughput guarantees are required, the baseline approach can not be used. Here, we need an additional round-robin arbiter at an output port (one for BE and one for GT requests) when going from RR3 to SP3, leading to a higher overhead. When extending SP3 to dynamic priorities, only smaller changes are needed. In comparison to BS, our approach enables to share VCs between different GT streams and hence to lower the number of needed VCs (e.g. FP2) while achieving the same performance improvements (cf. Section IV-A) and thus also the overhead.

The achievable clock frequency was 210 MHz for all designs, showing that the extensions did not influence the critical timing path of a router. The frequency was restricted by the minimum achievable period, caused by a *data path delay* of *4,75ns*, consisting of *1,31ns* for logic and *3,44ns* route delay.

V. CONCLUSION

In this paper we presented a novel arbitration scheme for NoC routers in mixed-criticality systems. Unlike many other existing approaches, we prioritize best-effort over safety-critical guaranteed-throughput traffic whenever possible. To limit the interference, we online monitor the buffer fill level at each router and, based on this, increase its priority only when necessary. Doing this, we can exploit the latency and throughput slack of critical applications, leading to an improved performance for general purpose applications in mixed-criticality systems.

Our experimental evaluation showed that the approach can improve the latency of best-effort traffic by up to 17% compared to a standard prioritization scheme. At the same time, the approach provides throughput guarantees to safety-critical real-time functions and thus sufficient isolation as requested by safety standards.

REFERENCES

[1] S. Baruah, H. Li, and L. Stougie, "Towards the design of certifiable mixed-criticality systems," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2010 16th IEEE*, 2010, pp. 13–22.

[2] *IEC 61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety Related Systems*, Std., 1999.

[3] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip," in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, vol. 2, Feb. 2004, pp. 890–8952.

[4] K. Goossens and A. Hansson, "The aethereal network on chip after ten years: Goals, evolution, lessons, and future," in *Proceedings of the 47th Design Automation Conference*, ser. DAC '10. New York, NY, USA: ACM, 2010, pp. 306–311.

[5] A. Psarras, I. Seitanidis, C. Nicopoulos, and G. Dimitrakopoulos, "PhaseNoC: TDM scheduling at the virtual-channel level for efficient network traffic isolation," in *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, ser. DATE '15. San Jose, CA, USA: EDA Consortium, 2015, pp. 1090–1095. [Online]. Available: http://dl.acm.org/citation.cfm?id=2755753.2757066

[6] E. Wandeler and L. Thiele, "Real-time interfaces for interface-based design of real-time systems with fixed priority scheduling," in *Proceedings of the 5th ACM International Conference on Embedded Software*, ser. EMSOFT '05. New York, NY, USA: ACM, 2005, pp. 80–89.

[7] S. Baruah, A. Burns, and R. Davis, "Response-time analysis for mixed criticality systems," in *Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd*, Nov. 2011, pp. 34–43.

[8] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *J. Syst. Archit.*, vol. 50, no. 2-3, pp. 105–128, Feb. 2004.

[9] T. Bjerregaard and J. Sparsoe, "Scheduling discipline for latency and bandwidth guarantees in asynchronous network-on-chip," in *Asynchronous Circuits and Systems, 2005. ASYNC 2005. Proceedings. 11th IEEE International Symposium on*, Mar. 2005, pp. 34–43.

[10] J. W. Lee, M. C. Ng, and K. Asanovic, "Globally-synchronized frames for guaranteed quality-of-service in on-chip networks," in *Proceedings of the 35th Annual International Symposium on Computer Architecture*, ser. ISCA '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 89–100.

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

[12] J. Diemer and R. Ernst, "Back suction: Service guarantees for latency-sensitive on-chip networks," in *Networks-on-Chip (NOCS), 2010 Fourth ACM/IEEE International Symposium on*, 2010, pp. 155–162.

[13] W. C. Tsai, H. E. Lin, Y. C. Lan, S. J. Chen, and Y. H. Hu, "A novel flow fluidity meter for BiNoC bandwidth resource allocation," in *2015 28th IEEE International System-on-Chip Conference (SOCC)*, Sept 2015, pp. 281–286.

[14] J. A. Stankovic, K. Ramamritham, and M. Spuri, *Deadline Scheduling for Real-Time Systems: Edf and Related Algorithms*. Norwell, MA, USA: Kluwer Academic Publishers, 1998.

[15] S. Tobuschat, M. Neukirchner, L. Ecco, and R. Ernst, "Workload-aware shaping of shared resource accesses in mixed-criticality systems," in *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2014 International Conference on*, Oct. 2014, pp. 1–10.

[16] W. Dally and B. Towles, *Principles and Practices of Interconnection Networks*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2003.

[17] A. Kostrzewa, S. Saidi, L. Ecco, and R. Ernst, "Dynamic admission control for real-time networks-on-chips," in *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, Jan. 2016, pp. 719–724.

[18] S. Tobuschat and R. Ernst, "Real-time communication analysis for Networks-on-Chip with backpressure," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2017*, March 2017, pp. 590–595.

[19] H. Kashif and H. Patel, "Buffer space allocation for real-time priority-aware networks," in *2016 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, April 2016, pp. 1–12.

[20] J.-Y. Le Boudec and P. Thiran, *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet*. Berlin, Heidelberg: Springer-Verlag, 2001.

[21] Y. Ben-Itzhak, E. Zahavi, I. Cidon, and A. Kolodny, "Hnocs: Modular open-source simulator for heterogeneous NoCs," in *Embedded Computer Systems (SAMOS), 2012 International Conference on*, Jul. 2012, pp. 51–57.

[22] Y. Hara, H. Tomiyama, S. Honda, and H. Takada, "Proposal and quantitative analysis of the chstone benchmark program suite for practical c-based high-level synthesis," *Journal of Information Processing.*, vol. 17, pp. 242–254, 2009.