Adaptive Allocation of Default Router Paths in Network-on-Chips for Latency Reduction

Jan Moritz Joseph
Otto-von-Guericke-Universität Magdeburg
Institut für Informations- und Kommunikationstechnik,
39106 Magdeburg, Germany
Email: jan.joseph@ovgu.de

Chritopher Blochwitz
Universität zu Lübeck
Institut für Technische Informatik,
23562 Lübeck, Germany
Email: blochwitz@iti.uni-luebeck.de

Thilo Pionteck
Otto-von-Guericke-Universität Magdeburg
Institut für Informations- und Kommunikationstechnik,
39106 Magdeburg, Germany
thilo.pionteck@ovgu.de

***Abstract*—We present a novel prioritization technique to reduce latencies in Network-on-chips. For individual routers, we adaptively allocate default paths assuming that subsequent packages are part of a data stream and, thus, routing decisions are identical. Since proactive routing to an output port is performed, the conventional router pipeline is partly bypassed. The method is deterministic, non-speculative with local and autonomous decisions, retains the standard network load, and does not penalize non-prioritized links. Virtual point-to-point connections emerge, which span multiple hops and accelerate interleaved streams. We achieve an average package latency reduction of 4.8% to 12.2% in simulations for PARSEC benchmarks.**

***Keywords*—Network-on-Chip, router architecture, latency reduction, path prioritization, data streams**

I. Introduction

Nowadays, chip design is facing multiple challenges concerning limits to performance, power consumption, and costs. Hence, the complexity of on-chip interconnects is growing due to the increasing requirements for on-chip bandwidth, the restricted scaling of global interconnects, and the need for higher degrees of re-usability. Network-on-Chips (NoCs) are a promising approach to tackle these limitations and to meet today's performance demands. Thus, NoCs are widely researched in academia [1], [2] and industry [3].

All on-chip communication infrastructures, in particular NoCs, must fulfill key metrics. In general, the network's performance must meet the demands of specific applications. As a critical design parameter, it is linked to the network's latency in NoCs: Every router in the network, which is passed by a packet, adds an additional latency as the router has to evaluate packet information. To reduce this effect, global concepts have been proposed, e.g. express virtual channels [4] or long range links [5]. In addition to performance demands, power consumption and area costs may not exceed specific constraints. Thus, other publications address these topics through global optimizations such as bus widths and reduced link buffer depths [6]. All these methods optimize the design metrics from a system point of view by adopting global parameters. As another approach, the router architecture itself can be adopted: The multiplexer router design is more efficient than the matrix router design [7], architectures are aware of congestion [8], or designs couple system level and architectural aspects [9].

The exploitation of traffic characteristics offers a significant potential for router architecture optimizations. One option bases on real world on-chip network loads, which comprise data streams [10]. Advantage can be taken of this, since routers in NoCs, which are passed by data streams, come to the same decision for every subsequent packet. Thus, parts of the router pipeline can be bypassed.

Here, we propose a novel router technique with an adaptive allocation of default paths in routers. Our method is non-speculative and, thus, the standard network load is retained. Furthermore, packets on non-prioritized routes are not penalized by the default paths. Using the proposed technique, routers locally configure themselves. Hence, virtual point-to-point connections emerge temporarily within the NoC and the acceleration comprises aggregates of interleaved data streams. For comprehensive and fair benchmarks, PARSEC-based [11] trace files are used. The paper is structured as follows: In Sec. II, we will give an overview of the related work. In Sec. III, we will motivate and introduce the technique and we will present modifications in the router architecture. In Sec. IV, the performance will be evaluated in benchmarks and, finally in Sec. V, the results will be discussed.

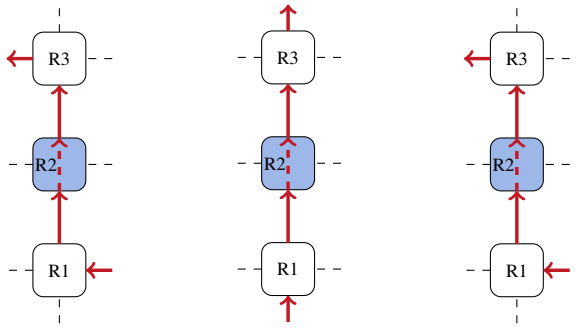
II. Related work

Reducing NoC latencies and, thus, optimizing the interconnection performance is a vital research topic. On the one hand, latency reductions are conducted for case studies, which are strongly linked to single applications: E.g., in [12], an automated, application specific NoC design flow is proposed and in [13], buffer sizes are optimized for application-specific

performance guarantees. This approach offers a large optimization potential yet lacks generality. On the other hand, more broad approaches cover the requirements of multiple use cases. As an example, in [14], the performance is increased by 25% via dynamic virtual channel (VC) regulation.

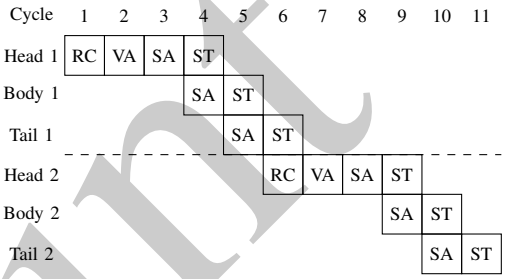
In general, packet transit times can be decreased by exploiting characteristics of streams of packets. This can be accomplished on a micro-electronic level via new router architectures or with novel routing techniques from a system design perspective. Pursuant to the former approach, the Æthereal NoC [15] consists of two separated subrouters, which are optimized for either best-effort or guaranteed throughput traffic. Here, data streams can be accelerated using strict service guarantees. In another approach, QNoC [16] offers a Quality-of-Service (QoS) with different priorities depending on traffic characteristics. However, the existence of a data stream must be known prior to the transmission and aggregates of interleaved data streams are not accelerated. The methods in [17] follow the latter approach. Here, pre-configured routes reduce network latencies. However, this method requires speculative forwarding of packets resulting in unnecessary network load and higher energy consumption for false routed packets. This increases the latency for other packets in the network. Extending this technique, in [18], virtual point-to-point connections are allocated in the NoC for fast packet stream traversal. However, this does not allow for acceleration of interleaved data streams. Another approach for throughput optimization is look-ahead bypassing [19]. Via extra flits, lookahead information is transmitted to adjacent routers, which attempt a pre-allocation of a bypass path. In [20], the SMART router is proposed, which aims to accelerate data streams in NoCs with a single-cycle router design using repeater cells. In contrast to this design, our proposed method focuses solely on local decisions. Thus, SMART's Switch Allocation Global (SA-G) cycle is not required, which adds an additional cycle of delay to each packet. In addition, the SMART router transmits additional (SSR) signals via dedicated wires. Furthermore, our design focuses on standard (synchronous) hardware platforms and does not rely on any special or modified cells or repeaters. Thus, it can be implemented on any commercial FPGA. Relying on local decisions only, the Aeolian technique [21] allows for pre-allocated multi-hop paths bz utilizing asynchronous decisions. In another approach, Deja Vu [22] prepares routers in advance for injected packages as well, which results in additional communication overhead. Moreover, in [23] a hybrid router design is proposed combining packet-switched with circuit-switched methods. This results in an average system performance improvement of 10%. In contrast, our method does not require flits to be either packet-switched or circuit-switched.

Due to simplicity, NoC benchmarking is regularly conducted using synthetic traffic patterns. This method does not fully reflect properties of real-world data streams. Therefore, full system simulation (e.g. co-simulation with Gem5 [24]) or trace-driven benchmarks with dependency tracking [25] can be used. In addition, abstract task graph modeling is a common

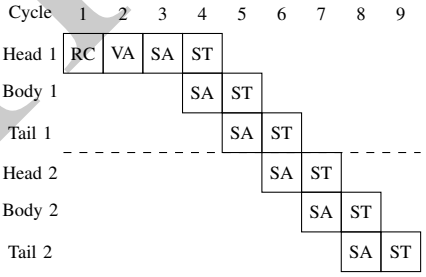


(a) First packet (b) Second packet (c) Third packet

Fig. 1: default path of data stream in pouter R2



(a) Normal router pipeline for two packets in R2



(b) Fast router pipeline for two packets in R2

Fig. 2: Exemplary router pipeline for router R2

technique [26]–[28], since it allows for fast evaluation of the influence of data streams on NoC designs. As analyzed in the MCSL benchmark suite [27] or the Nocbench traffic generator [26], data streams are typical for NoC network loads. In [10], data streams are identified, which do not change direction during a time interval. Thus, they are called *semi-static data streams*. Based hereupon, concepts for router architectures that allow for prioritized connections between in- and outputs within a router are published [29].

III. Methods

Routers reach identical decisions for every subsequent packet during data streams. For instance, in Fig. 1, the router R2 makes identical decisions for all three packets and, thus, potentially could skip parts of its routing pipeline reducing the latency of the two latter packets and increasing the network's performance. In the normal router pipeline, first, a route is calculated per packet (RC), then, a VC is allocated (VA), and finally, after winning switch arbitration (SA), the packet is

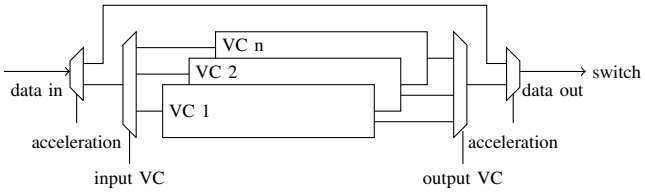


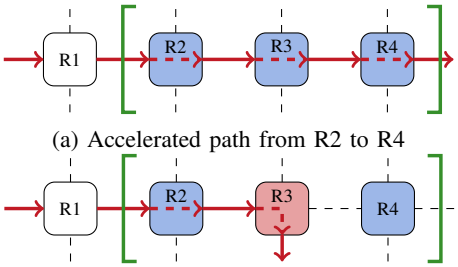
Fig. 3: Simple input unit with buffer bypass

sent on a flit-per-flit basis (ST) as shown in Fig. 2a. For data streams, routing, VC selection, and arbitration process can be abbreviated due to identical results. The optimized router pipeline allows for a faster processing of subsequent packets in the example as shown in Fig. 2b. Since routing and VC calculation are skipped, the second packet leaves the router two clock cycles earlier. Hence, along a serious of accelerated routers, the NoC behaves like a circuit switching network while for routers in standard mode classical packet switching is performed.

Here, we propose a deterministic router architecture with wormhole transmission that allows for prioritized paths between in- and output ports within a router. To prioritize aggregates of data streams as well, they are detected locally and the prioritization is decided autonomously. Generally, in a deterministic NoC, packets must not be routed in a false direction. In addition, flits of a packet must not be split up because the information is only stored in the head flit and, thus, would be lost for trailed flits. For the same reason, the flit order must be preserved. This work unifies the advantages of both circuit switching and packet based switching: Standard packed based traffic is not blocked by the circuit switched connections and, thus, disadvantage of circuit switching, i.e. poor link utilization and blocking of other traffic, do not apply.

In Fig. 3, the input data path of a router is shown that bypasses the VC buffers in the router data path, which can be used to accelerate packets and data streams. This idea is promising for simplicity and low overhead. However, in the next paragraph, critical network configurations are discussed, which prevent a simple solution. In general, there are two main issues: First, the instant of time when starting the acceleration both must occur often enough for a sufficient performance advantage and be safe (i.e. fulfill the functional requirements). If the input buffers are bypassed as shown in Fig. 3, a safe functionality can be secured by activating the bypass only if all buffers are empty. Otherwise, flits that are still stored in the buffers are lost or delivered out of order. However, this network configuration extremely rarely occurs, especially during data streams. Thus, the accelerated pipeline begins very seldom and is not worth the effort. A second key issue is that deterministic dimension order routing must be preserved during the acceleration. Hence, if a packet enters an accelerated path and must be routed in another direction (as shown in Fig. 4), the accelerated path must be reset. This can either be done by another participant in the network or a router resets itself. There are the following design options:

A: Adaptive routing algorithms correct the error. This is not an acceptable solution, since the performance advantage for data streams is lost for correction of the false routing. In addition, deterministic routing offers small overhead and prevents deadlocks.



(a) Accelerated path from R2 to R4

(b) Reset must be issued to R3

Fig. 4: Reset case for accelerated path

B: Alternatively, the acceleration is reset externally using an adjacent router or processing element. I.e. a global reset is issued whenever a processing element or a router sends a packet whose destination address differs from the destination address of the previous packet. Thus, every accelerated router in the NoC is reset. Although this ensures a fault-free function of the network, it is not a viable design choice since in general, numerous resets are accumulated and accelerations are reset too often. Thus, the area overhead of the global reset wires is not compensated by the increased performance of the network.

C: A separated reset network, which sends the reset signal along the path of the packet, is also not an applicable design due to area costs and design complexity.

D: Reset generation is possible in routers, which are adjacent to an accelerated path and if they are not in an accelerated state themselves. The reset is sent along with packets, which have a different destination address than the previous packet. Thus, routers at the beginning of an accelerated path manage the pipelines' states. Unfortunately, it is impossible to ensure that every reset condition is met without transmitting status information between routers. This contradicts our proposition that decisions are made locally only. In addition, if multiple consecutive routers are part of an accelerated path, and two data streams are transmitted, in which alternating packets travel to the end of the path or leave the path at an intermediate router, all routers at the beginning of the path are reset with each "shorter" packet. This network configuration is a periodic transition between the configurations from Fig. 4a and Fig. 4b. The router R2 never holds its acceleration although it is passed by a semi-static data stream. I.e. the pipeline will never be started at the beginning of the path although aggregates of data streams must be accelerated as well.

E: Each router locally resets itself. This design choice has the disadvantage that routers have to inspect packets before sending although the path is accelerated. However, it ensures that packets are not on false routes and routers come to a decision locally. On the positive side, it is possible that the reset can be issued in parallel to the accelerated pipeline, which nullifies the disadvantage of the packet inspection.

Therefore, the message format is modified. In the header flit, beside the RI (routing information) and SN (sequence number) fields, an RS (reset information) field is introduced. For dimension order routing, it contains the number of router steps that are left for the packet until the routing dimension changes. The field is modified when leaving routers by decreasing the corresponding value. Accelerated routers can check in parallel whether the accelerated path matches the field values with negligible overhead. Asynchronous and safe resets are issued ensuring that the packet is routed correctly.

These network configurations explain why a simpler and less complex router method than proposed here is not feasible. Thus, we identified the following start and stop conditions for the acceleration complying with design choice **E**.

Start condition: In each router, there is a counter, which evaluates how many consecutive packets were routed from one input port to every output port. If this counter transcends a given threshold k (here called "acceleration threshold"), the router checks the following start condition per flit in the router: The accelerated pipeline is started after transmitting a tail flit if zero or one VCs have flits stored at the corresponding input port (i.e. maximum one VC is active). The accelerated pipeline cannot be started if two or more VCs are active since those could allocate different routes.

In general, at high loads one would expect that more than one VC is occupied. Thus, if the network load is at saturation, no acceleration can be expected. However, the application based benchmarks in the results show that a network configuration, which allows for acceleration, is regularly encountered for realistic traffic loads. Thus, our results support that the method allows for an acceleration despite this start condition.

Stop condition 1 – newly injected data stream: If there is an accelerated pipeline from input port i to output port j and the input port l routes to output port j as well, the accelerated pipeline is terminated.

Stop condition 2 – other routing decision for the packet: If there is an accelerated pipeline from input port i to output port j and at input port i, a packet arrives that routes to output port $l \neq j$, the acceleration ends. This condition is checked using the design choice and message format from case **E**.

With these start and stop conditions, the regular state machine can be extended by the accelerated state as shown in Fig. 5. To guarantee that it is always possible to switch from the accelerated pipeline to a normal router pipeline without any loss of information and to allow for the start and stop conditions, the router architecture must be adopted as well. The architecture is based on a standard router design [30] and is shown in Fig. 6. The modified or new units are marked in blue. The function of every unit is shortly described in Tab. I and elaborated in the next paragraphs. The aim of the proposed architecture is to reduce the required modifications and re-use as many standard components as possible. The architecture allows to use a standard routing unit, VC-allocator, and switch. The following units are modified:

Input unit: The modified input unit of the router is shown in detail in Fig. 7. There are two standard components: The

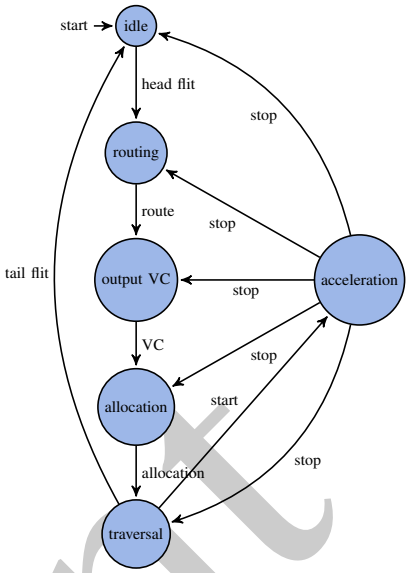


Fig. 5: State machine of accelerated router

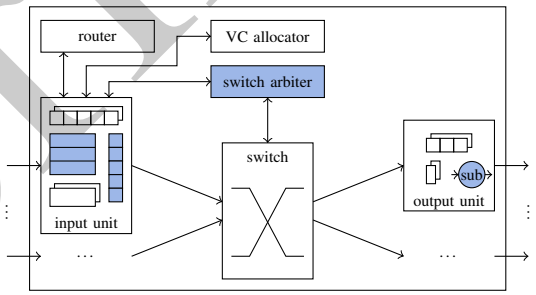


Fig. 6: modified router design

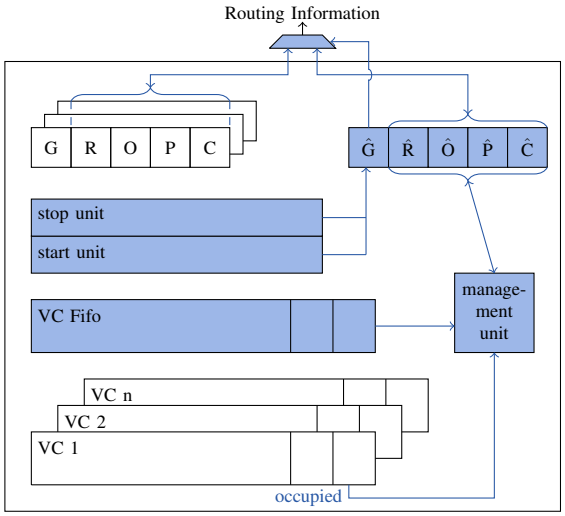


Fig. 7: Input unit architecture

TABLE I: Router units and status fields

Unit / Field	Purpose
input unit	handles incoming flits
VC status registers	every VC has the following statuses: G – state, R – route, O – output VC, P– pointers to current flit, C – credit count
input unit status registers	overrides VC status registers, if $\hat{G}$ is set. $\hat{R}$ – accelerated route, $\hat{O}$ – accelerated output VC, $\hat{P}$ – pointers to current flit, $\hat{C}$ – credit count
start unit	starts acceleration, if start condition is met; holds acceleration threshold k
stop unit	checks in parallel, if reset condition is met via RS field in head flits and stops acceleration
VC FiFo	stores input VCs in order of arrival during acceleration
management unit	set $\hat{P}$, $\hat{C}$ during acceleration
output unit	handles outgoing flits
substractor	modifies RS field in head flits in parallel to sending
status registers	handle down-stream VC and flow control
routing unit	calculates routes for packets
VC allocator	allocates VCs for packets
switch arbiter	arbitrates the switch between input units and VCs
switch	switches flits according to arbitration

VC buffers, which store incoming flits, and the VC status registers (one per VC). In the status registers (G, R, O, P, C), the field G is the state of the VC, which corresponds to the states on the left hand side of Fig. 5. The route field is R, which stores the route that the VC allocates. The output VC in the route is given by the field O. Finally, P and C are the pointer to the current active flit and the credit count. In addition, there are four new components: The start and stop units handle the accelerated state. The *input unit status registers* $\hat{G}, \hat{R}, \hat{O}, \hat{P}$, and $\hat{C}$ are active during the accelerated state and override all VC status registers. Thus, multiplexers are needed as schematically shown on the top of Fig. 7. These modifications are identical for all VCs and pairs of status registers. Furthermore, an additional Fifo buffer is required, which has the depth of a single VC buffer to store the order of the VCs in which flits are received during the acceleration for flit order preservation.

The *start unit* traverses every state in each VC to "accelerated" if the start condition is met. First, the input unit status register $\hat{G}$ is set, which overrides the VC status registers G, R, O, P, and C from the VCs with the information stored in the registers $\hat{R}, \hat{O}, \hat{P}$, and $\hat{C}$. Second, the input unit status registers are updated: The routing field $\hat{R}$ is filled with the last allocated route. The register for the output VC $\hat{O}$ and the fields pointing to the active flit and holding the credit count $\hat{P}$ and $\hat{C}$ is modified so that the flit order is preserved. This happens in two phases and is managed by the *management unit*: In the first phase, the single active VC (if available) is arbitrated for sending until its remaining flits are transmitted. Meanwhile, the router may receive other flits. The VC-order of these flits is stored in the VC-Fifo. The flit order itself is kept by the VC-buffers. Thus, in the second phase, the head flit of the VC, which is stored in the head element of the VC-Fifo, is arbitrated for sending until the accelerated pipeline is stopped. The VC-Fifo must be of the same length as one VC buffer since prior to starting the second phase, at maximum one VC is completely filled. The flow control is not modified. Thus, if the input unit's buffers of VC-Fifo are filled, the flow control denies further input.

The *stop unit* is responsible for the transition from the accelerated state back to the standard state in each VC. In order to meet the second stop condition, the RS field of every head flit is inspected. Since the field contains information when the packet changes the dimension of routing (using two subfields in 2D- or three in 3D-mesh topology), the overhead is small and the check is possible in parallel to the router pipeline. For the RS field, only zero or non-zero conditions (depending on the path of the acceleration) have to be checked. Thus, the stop unit checks whether the accelerated route matches the data path field by simple comparison. If this test fails or if the first stop condition is met in the switch allocator, the stop unit restores the state of the VCs. This is done by copying the information from $\hat{R}, \hat{O}, \hat{P}$, and $\hat{C}$ to the input unit status register for active VCs and set G to "traversal". For non-active VCs, G is set to "idle" and the other VC status registers are cleared. Finally, for the VC, from which the stop condition originated, G ist set to "routing" to start the regular router pipeline.

Output unit: The output unit is modified to transmit the reset information RS to the next router. As this router is designed for 2D- and 3D-mesh topologies, in the data path of each output unit, an additional subtraction unit is included, which modifies the reset field RS when sending a head flit.

Switch arbiter: The arbiter generates the first stop condition if an additional data stream allocates the same output port as an accelerated input-output port pair. Thus, an additional data path connecting the input unit and the arbiter is required. In addition, the arbitration logic must be modified to issue the reset. The arbitration process itself is as usual, since the accelerated state is managed by the input unit and transparent for the arbiter.

Summing up, the cycle-accurate behavior of the routers is modeled as follows: If the accelerated state is not activated, the router pipeline and, thus, the cycle accurate model is standard. If the pipeline is activated, the routing calculation and VC allocation are skipped for head flits resulting in a decreased latency of the router as shown in Fig. 2. If the stop unit in the input units meets a reset condition, this input of the router is stalled for one clock cycle to restore the VC status registers. Thus, resetting the router pipeline is associated with additional costs, which minimizes the latency reduction of the acceleration.

IV. Results

To estimate the performance of the proposed router architecture, a cycle-accurate NoC simulator was implemented in SystemC. As a first step, we evaluated if the acceleration threshold k in the start unit has an influence on the NoC's

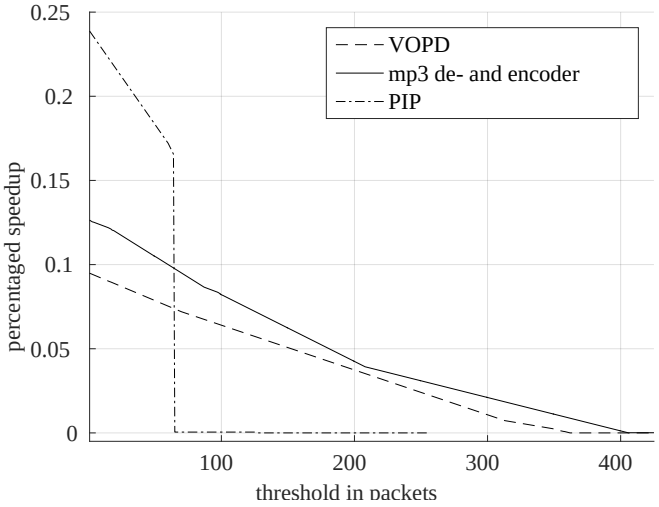


Fig. 8: simulated benchmarks for different thresholds k

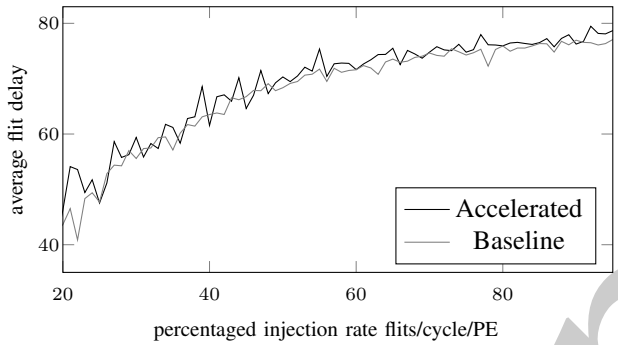


Fig. 9: Average flit delay for uniform traffic

performance. For this pre-evaluation, we used a simplified model with a 2D-mesh NoC and 16 processing elements. For traffic injection, we used abstract task graph modeling with application graphs for the video object plain decoder (VOPD), an mp3 de- and encoder and the picture-in-picture (PIP) application [7], [31]. These task graphs offer well-arranged (semi-static) data streams and allow for a fast evaluation of the router architecture's parameters. The tasks were manually mapped to cores. The results are shown in Fig. 8. The steep cut-off for PIP at 64 packets is a consequence of the longest path in the task graph, which is 64 packets long. The pre-evaluation shows that a larger threshold generally results in a larger average packet latency. Thus, we set the threshold to a single packet for the following detailed performance evaluation.

As a second step, we conducted a detailed benchmarking of the routers using synthetic traffic patterns. The NoC consisted of 64 routers (8×8) with four VCs with dedicated buffers and an input buffer depth of eight flits. There were ten flits per packet. First, synthetic traffic patterns were simulated with flit injection in between 20% and 95% of the clock cycles per processing element. The results are shown in Fig. 9 for uniform traffic pattern as the median of 16 simulation runs. In Fig. 10 the results for transpose traffic and in Fig. 11 for complement

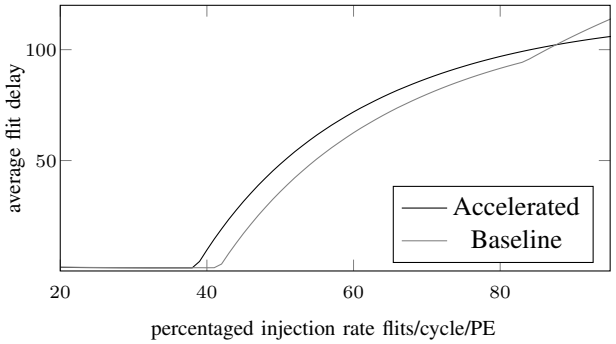


Fig. 10: Average flit delay for transpose traffic

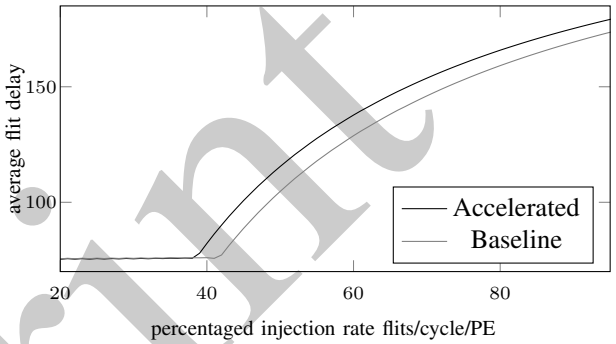


Fig. 11: Average flit delay for complement traffic

traffic pattern are depicted. The speedup in comparison with the baseline router for each benchmark is shown in Fig. 12. In general, the speedup results are ambiguous. In case of both transpose and complement traffic, the proposed router with allocation of defaults paths offers better performance of maximum 1.6% reduced delay for traffic injection rates below 35%. For medium injection rates the proposed router method does not offer a performance advantage due to back-pressure from filled buffers at the end of accelerated paths. Thus, the average flit delay increases earlier in comparison to a standard router as shown in Figs. 10 and 11. For higher injection rates, the routers regularly encounter reset conditions, which are associated with an additional delay. Thus, the performance of the proposed method is below the standard

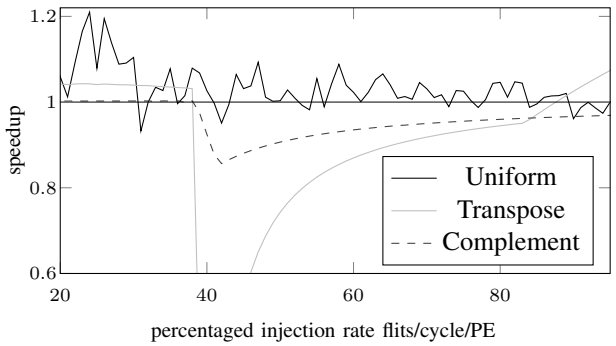


Fig. 12: Speedup of the accelerated router

TABLE II: benchmark results and speedup with 64 cores

Benchmark	Average Package Latency in ms		Speedup
	baseline	accelerated	
fluidanimate simsmall	0.83	0.73	12.2%
x264 simsmall	1.08	0.97	10.7%
dedub simmedium	0.88	0.80	9.2%
ferret simsmall	0.84	0.79	5.2%
canneal simlarge	0.84	0.80	5.0%
swaptions simlarge	0.80	0.76	5.0%
blacksholes simsmall	0.82	0.78	4.9%
bodytrack simlarge	0.93	0.89	4.3%

for complement traffic. The impact of filled buffers is reduced since fewer routers allocate default paths. For transpose traffic, the relation changes since the packet order allows for streamed processing. The speedup results for uniform traffic are better: A maximum speedup of 20% is achieved for low injection rates. As expected, it declines for higher injection rates due to additional delay from router resets.

The results for the synthetic traffic pattern implicate that the proposed method is advantageous for low network loads. These are regularly found in real world applications. Thus, as a third step, we benchmark of the NoC's routers with standard applications using the PARSEC benchmark suite. Therefore, 25 ms of CPU time in the PARSEC region of interest (ROI) are simulated. We implemented a SystemC-simulator with 64 cores running at 2 GHz connected by a 2D-mesh NoC. For the routers, we assumed a conservative clock frequency speed estimation of 100 MHz. Netrace trace files are used for traffic injection. The benchmark results are shown in Tab. II. The application based benchmarks show 4.3% average packet latency reduction for bodytrack to 12.2% for fluidanimate. Summing up, for real world applications, the proposed router method offers an increased performance. Thus, as future work, we will extend the method with a dynamical and local adoption of the acceleration threshold. Thus, we will reduce the influence of the additional clock cycle of delay due to resetting the accelerated routers state.

V. Conclusion

We propose a novel NoC router method to reduce latencies considering characteristics of data streams. The technique allows for prioritized connections between in- and output ports in a router. Since the data streams are detected locally and the prioritization is decided autonomously in each router, aggregates of data steams are prioritized as well. For synthetic traffic patterns in a NoC connecting 64 cores the results are ambiguous. For low network loads, we observed up to approx. 20% average performance increase for uniform traffic. High network loads with many resets reduce the network's performance. For real-world based traffic patterns, we evaluate the latency reductions with PARSEC benchmarks. The proposed router method offers between 4.8% and 12.2% latency reduction for these applications. As future work, we will extend the method by a dynamical and local adoption of the acceleration threshold to reduce the influence of costs that are associated with reset cases.

Acknowledgments

This work is funded by the German Research Foundation (DFG) project PI 447/5-1.

References

[1] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology," in *VLSI, 2002. Proceedings. IEEE Computer Society Annual Symposium on*, 2002, pp. 105–112.

[2] P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-chip interconnect architectures," *Computers, IEEE Transactions on*, vol. 54, no. 8, pp. 1025–1040, Aug 2005.

[3] V. Soteriou, N. Eisley, H. Wang, B. Li, and L.-S. Peh, "Polaris: A system-level roadmapping toolchain for on-chip interconnection networks," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 15, no. 8, pp. 855–868, Aug 2007.

[4] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Express virtual channels: towards the ideal interconnection fabric," in *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2. ACM, 2007, pp. 150–161.

[5] U. Ogras and R. Marculescu, ""it's a small world after all": Noc performance optimization via long-range link insertion," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. 14, no. 7, pp. 693–706, July 2006.

[6] Z. Yang, A. Kumar, and Y. Ha, "An area-efficient dynamically reconfigurable spatial division multiplexing network-on-chip with static throughput guarantee," in *Field-Programmable Technology (FPT), 2010 International Conference on*, Dec 2010, pp. 389–392.

[7] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for network-on-chip design," *Jour. of Sys. Arc.*, 2013.

[8] C. Wang, W.-H. Hu, and N. Bagherzadeh, "Congestion-aware network-on-chip router architecture," in *Computer Architecture and Digital Systems (CADS), 2010 15th CSI International Symposium on*, Sept 2010, pp. 137–144.

[9] X. Bin, F. Degui, J. Guanjun, W. Chao, Z. Nan, and C. Tianzhou, "A tightly coupled network-on-chip router architecture," in *Scalable Computing and Communications; Eighth International Conference on Embedded Computing, 2009. SCALCOM-EMBEDDEDCOM'09. International Conference on*, Sept 2009, pp. 279–284.

[10] T. Pionteck, C. Osterloh, and C. Albrecht, "Latency reduction of selected data streams in network-on-chips for adaptive manycore systems," in *NORCHIP, 2010*, Nov 2010, pp. 1–6.

[11] C. Bienia, "Benchmarking modern multiprocessors," Ph.D. dissertation, Princeton University, January 2011.

[12] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli, "Noc synthesis flow for customized domain specific multiprocessor systems-on-chip," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 16, no. 2, pp. 113–129, Feb 2005.

[13] A. Hansson, M. Wiggers, A. Moonen, K. Goossens, and M. Bekooij, "Enabling application-level performance guarantees in network-based systems on chip by applying dataflow analysis," *Computers Digital Techniques, IET*, vol. 3, no. 5, pp. 398–412, September 2009.

[14] C. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. Yousif, and C. Das, "Vichar: A dynamic virtual channel regulator for network-on-chip routers," in *Microarchitecture, 2006. MICRO-39. 39th Annual IEEE/ACM International Symposium on*, Dec 2006, pp. 333–346.

[15] K. Goossens, J. Dielissen, and A. Radulescu, "Aethereal network on chip: concepts, architectures, and implementations," *Design Test of Computers, IEEE*, vol. 22, no. 5, pp. 414–421, Sept 2005.

[16] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "Qnoc: Qos architecture and design process for network on chip," *Journal of systems architecture*, vol. 50, no. 2, pp. 105–128, 2004.

[17] G. Michelogiannakis, D. Pnevmatikatos, and M. Katevenis, "Approaching ideal noc latency with pre-configured routes," in *Networks-on-Chip, 2007. NOCS 2007. First International Symposium on*, May 2007, pp. 153–162.

[18] M. Modarressi, H. Sarbazi-Azad, and A. Tavakkol, "Virtual point-to-point links in packet-switched nocs," in *Symposium on VLSI, 2008. ISVLSI '08. IEEE Computer Society Annual*, April 2008, pp. 433–436.

[19] T. Krishna, J. Postman, C. Edmonds, L.-S. Peh, and P. Chiang, "Swift: A swing-reduced interconnect for a token-based network-on-chip in 90nm cmos," in *Computer Design (ICCD), 2010 IEEE International Conference on*, Oct 2010, pp. 439–446.

[20] C.-H. O. Chen, S. Park, T. Krishna, S. Subramanian, A. P. Chandrakasan, and L.-S. Peh, "Smart: A single-cycle reconfigurable noc for soc applications," in *Proceedings of the Conference on Design, Automation and Test in Europe*, ser. DATE '13. San Jose, CA, USA: EDA Consortium, 2013, pp. 338–343. [Online]. Available: http://dl.acm.org/citation.cfm?id=2485288.2485371

[21] W. Jiang, K. Bhardwaj, G. Lacourba, and S. M. Nowick, "A lightweight early arbitration method for low-latency asynchronous 2d-mesh noc's," in *Proceedings of the 52Nd Annual Design Automation Conference*, ser. DAC '15. New York, NY, USA: ACM, 2015, pp. 203:1–203:6. [Online]. Available: http://doi.acm.org/10.1145/2744769.2744777

[22] A. K. Abousamra, R. G. Melhem, and A. K. Jones, "D x0e0; vu switching for multiplane nocs," in *Networks on Chip (NoCS), 2012 Sixth IEEE/ACM International Symposium on*, May 2012, pp. 11–18.

[23] N. E. Jerger, M. Lipasti, and L. S. Peh, "Circuit-switched coherence," *IEEE Computer Architecture Letters*, vol. 6, no. 1, pp. 5–8, January 2007.

[24] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2024716.2024718

[25] J. Hestness and S. W. Keckler, "Netrace: Dependency-tracking traces for efficient network-on-chip experimentation," The University of Texas at Austin, Department of Computer Science, Tech. Rep. Technical Report TR-10-11, May 2011.

[26] E. Pekkarinen, L. Lehtonen, E. Salminen, and T. Hamalainen, "A set of traffic models for network-on-chip benchmarking," in *System on Chip (SoC), 2011 International Symposium on*, Oct 2011, pp. 78–81.

[27] W. Liu, J. Xu, X. Wu, Y. Ye, X. Wang, W. Zhang, M. Nikdast, and Z. Wang, "A noc traffic suite based on real applications," in *VLSI (ISVLSI), 2011 IEEE Computer Society Annual Symposium on*, July 2011, pp. 66–71.

[28] J. Joseph and T. Pionteck, "A cycle-accurate network-on-chip simulator with support for abstract task graph modeling," in *System-on-Chip (SoC), 2014 International Symposium on*, Oct 2014, pp. 1–6.

[29] T. Pionteck and C. Osterloh, "Prioritizing semi-static data streams in network-on-chips for runtime reconfigurable systems," in *High Performance Computing and Simulation (HPCS), 2013 International Conference on*, July 2013, pp. 229–232.

[30] J. W. Dally and B. P. Towles, *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004.

[31] van der Tol, Erik B. and E. G. Jaspers, "Mapping of mpeg-4 decoding on a flexible architecture platform," *Media Processors*, 2002.