This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCE.2021.3062001, IEEE Consumer Electronics Magazine

Revising NoC in Future Multi-Core based Consumer Electronics for Performance

Abhijit Das IIT Guwahati
Abhishek Kumar Oracle Inc.
John Jose IIT Guwahati
Maurizio Palesi University of Catania

Abstract—**With the increasing demand for efficient data processing, the latest consumer electronics are powered by multi-core processors. Many applications that run on these devices expect minimum data access latency, where Network-on-Chip (NoC) plays a crucial role. However, NoC is found to be responsible for 60–75% of miss penalty. The objective of this article is to present a revised NoC communication framework that can possibly reduce the miss penalty. The proposed architecture orchestrates buffers of NoC routers as a victim cache to store evicted cache blocks. Future references to such stored blocks are replied from NoC routers which significantly reduces miss penalty and improves system performance. Minimising miss penalty by maximising NoC resource utilisation in these devices is a win-win situation.**

I. Introduction and Background

In the world as it is today, data drives everything, including most of the devices in consumer electronics. The ever increasing data demand efficient processing to make a quick decision. As a result, all the devices that require processing are realised by some form of a multi-core processor. For example, even the simplest of hand-held devices in consumer electronics, like smartphones are powered with at least a quad-core or even an octa-core processor, and the count keeps increasing [1]. Such devices rely on Network-on-Chip (NoC) for an efficient, reliable and scalable communication infrastructure. Conceptual illustration of an NoC based multi-core processor used in such devices is shown in Figure 1.

Some of the most popular multi-core based consumer electronics like smartphones, tablets, personal computers, autonomous vehicles, etc., have up to 32 processing cores [2]. With the ongoing increase in data driven applications that run on these devices, future consumer electronics may have even higher number of processing cores. When applications run, they expect minimum data access latency, where on-chip caches play a major role. However, due to the increasing core counts, limited on-chip area and the associated cost, most of these devices have only two levels of on-chip caches, as shown in Figure 1. In case of an L1 cache miss, the requested block is brought from the next level of memory (L2 cache). Similarly, during an L2 cache miss, the requested block is brought from the off-chip memory. The time spent in bringing a requested block into the L1 cache is called miss penalty. As the entire communication between L1, L2 and off-chip memory is done through the underlying NoC, it plays a significant role in the miss penalty. Hence, NoC in traditional multi-core processors is found to be responsible for 60-75% of miss penalty and thus has a significant say on performance [3].

Due to limited capacity, servicing L1 cache miss beyond a point requires evicting a valid block. If an evicted L1 cache block is clean, it is discarded whereas, if the block is dirty (modified), it is sent for write-back. The evicted, dirty L1 cache block enters NoC (as a packet) through the *local router*; which connects a core to the underlying NoC (refer Figure 1). In-transit packets are stored in router buffers during their routing and arbitration decisions. Hence, the packet carrying the dirty block traverses multiple routers to finally reach its destination L2 cache bank for write-back. In applications that exhibit decent temporal locality of reference, if a recently evicted L1 cache block is re-referenced, it needs to be brought back from the L2 cache bank. In the worst case, when the re-referenced block is not present in the L2 cache bank (L2 miss), it is brought from off-chip memory. If the recently evicted block could have stayed in L1 cache a little longer until its re-reference,

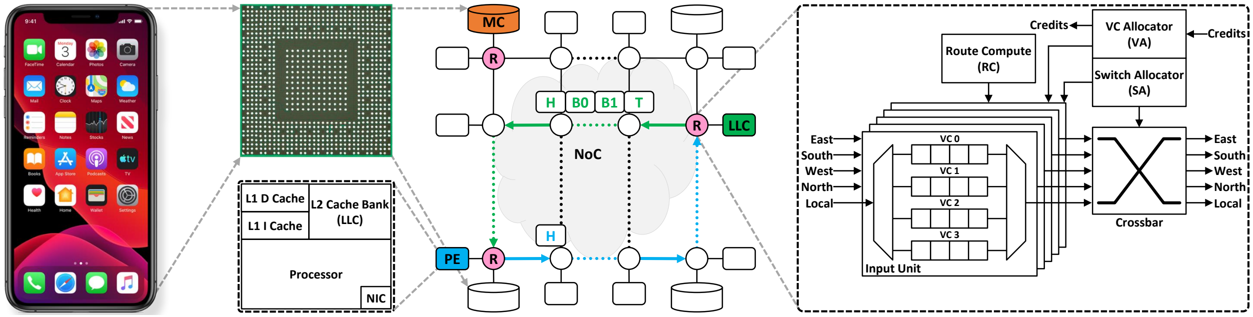


Figure 1: Conceptual view of an NoC based multi-core processor. A packet in NoC is divided into multiple smaller units called flits. A control packet consists of a single head flit (H), whereas a data packet consists of a head flit followed by multiple body flits and ended by a tail flit (H, B0, B1, T).

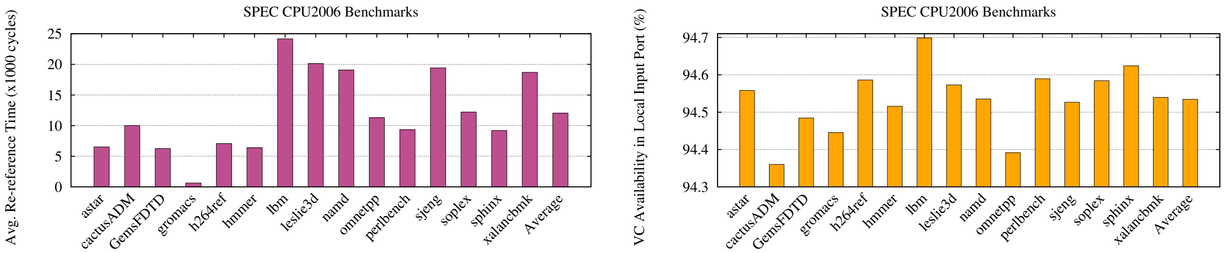
these kinds of miss penalty could be avoided. Such unfortunate miss penalties delay applications and affect system performance. Increasing the size of L1 cache may temporarily work but is not feasible due to its impact on L1 cache hit time, on-chip area and power constraints of consumer electronics.

Due to very high L1 cache hit rate of around 90-95%, L1 cache misses are sporadic. Hence, average packet injection rate is only around 5%, and the NoC resources remain severely underutilised [4]. With low packet injection rate, buffers in NoC routers are underutilised. Since buffers are both area and power-hungry, promising alternatives like bufferless and minimally buffered NoC routers are proposed [5]. There are also solutions like run-time power gating with look-ahead routing that optimises power consumption of routers [6]. However, these alternatives fail to provide the necessary performance bandwidth during on-chip network congestion. Buffered NoC routers provide lower latency and higher throughput per unit power under most conditions [7]. With consumer electronics, users always look for solutions with more functionality and hence maintaining and even improving performance is important. Consumer electronics also strive to maximise resource utilisation. In these devices, any optimisation that improves resource utilisation, as well as performance, will go a long way.

In this article, we attempt such an optimisation. Since NoC plays a significant role in cache miss penalty, we propose to include it in the memory hierarchy. We propose to consider underutilised NoC router buffers as a victim cache between L1 and L2 cache. When evicted, dirty L1 cache blocks enter the local router to travel towards L2 cache bank for write-back; we disable their routing and arbitration. Such blocks remain stored in the router buffers until they are not interrupted. We also bring and store some evicted, clean L1 cache blocks (which are normally discarded) to the local routers. Now, when an application re-references a recently evicted L1 cache block, we are able to reply quickly from its local router. This reply avoids the whole NoC travel resulting in a negligible miss penalty. Applications can resume their execution at the earliest, which can improve overall system performance. Many applications that run on consumer electronic devices like autonomous vehicles, e-healthcare equipment etc., are latency-critical. Hence the proposed optimisation will be more effective in improving performance for them.

II. Related Work

As the number of cores continues to increase in multi-core processors, considering NoC and memory hierarchy together is the way forward [3]. One of the first works that tried using NoC for storage dates back to late ‘80s [8]. The authors advocated that NoC can be included in the memory hierarchy by placing a small cache in the routers. There are other significant works in the recent past that are focused on NoC aware cache and coherence implementations [9][10]. A recent article specifically emphasised the need for NoC based consumer electronic devices to improve the performance as



(a) Re-reference time of evicted L1 cache blocks

(b) VC availability in local input port

Figure 2: Key observations that motivated the proposed architecture

we go into the future [11]. There is also an exciting work about power-aware efficient data transfer in NoC based consumer electronic devices [12]. Nevertheless, most of these works have extra storage.

A recent work employed underutilised NoC router buffers to store evicted last-level cache (LLC) blocks to reduce off-chip miss penalty [13]. However, blocks evicted from LLC is very less when compared to the L1 cache. In a promising attempt, another work proposed to store evicted, dirty L1 cache blocks in NoC routers [14]. The work attempted to reply future references to the stored blocks from NoC routers and reduce miss penalty. However, most of the evicted blocks from L1 cache are clean as the number of data reads are always way more than data writes. Hence, in this article, we present an analysis of how the miss penalty will be when all the evicted L1 blocks (both clean and dirty) are stored in the NoC routers. The proposed work is similar to victim caching [15] between L1 and L2 cache without any additional storage. The proposed work can also be complemented with any other optimisation in the memory hierarchy.

III. Motivation

The duration between evicting and then requesting the same L1 cache block in near future, by the same core is called re-reference time. To understand how frequently a cache block is re-referenced in standard applications, SPEC CPU2006 multi-programmed benchmarks are analysed. We simulate a 64-core NoC based multi-core processor by running 64 copies of the same benchmark. This simulation setup is employed to mimic a multi-core based consumer electronic system where multiple applications run in parallel. Figure 2a shows the average re-reference time of evicted L1 cache blocks for different benchmarks. To understand with an example, in *astar* benchmark, an evicted L1 cache block is re-referenced within an average time of 6532 cycles. Across all the simulated benchmarks, an evicted L1 cache block is re-referenced within an average interval of 12000 cycles. This interval indicates how frequently an application suffers from L1 cache miss penalty due to unfortunate evictions.

Recently evicted L1 cache blocks are re-referenced within a small time interval.

To meet the worst case performance bandwidth, NoC based systems prefer input buffered routers. Buffers are further divided into virtual channels (VCs) for deadlock-free routing and better utilisation. As shown in Figure 1, packets entering through different input ports (east, south, west, north and local) get temporarily stored in the available VCs and take part in routing and arbitration decisions. However, due to low packet injection rate, VCs are underutilised and thus remains free most of the time. VC availability in local input port of NoC routers is shown in Figure 2b for different SPEC CPU2006 benchmarks. Since these benchmarks have an average injection rate of only around 5%, at least one VC is always free (≈ 95%) except during peak on-chip network congestion.

NoC based systems use buffered routers for worst case bandwidth, but buffers remain underutilised.

Since consumer electronics desire performance as well as strive for maximising resource utilisation,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/MCE.2021.3062001, IEEE Consumer Electronics Magazine

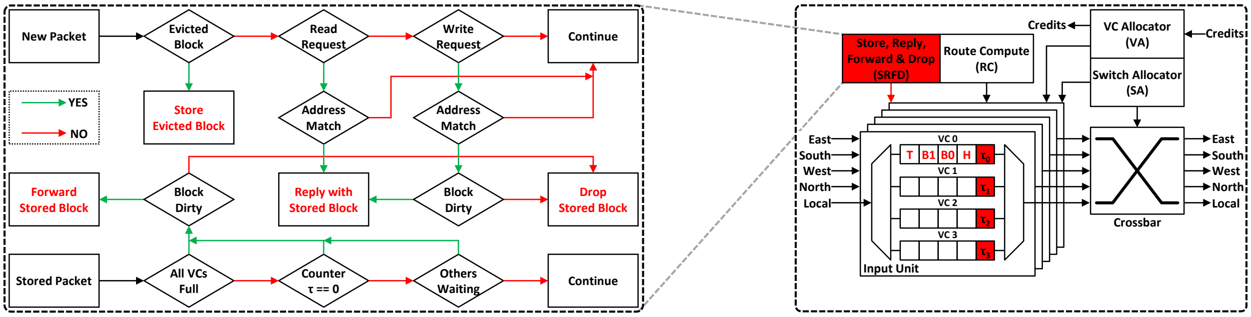


Figure 3: Conceptual view of the proposed router microarchitecture.

merging the above two observations achieves both.

> Keep evicted blocks in NoC router buffers (VCs) and when re-referenced, generate local replies from the routers to reduce L1 cache miss penalty.

Src	Dest	Addr	. . . . .	Store	Dirty	S	X

Figure 4: Updated message/packet header

IV. Proposed Architecture

We talk about the proposed architecture in this section. It uses MOESI directory protocol based on-chip caching, with L2 being the last level cache (LLC). Figure 3 presents a conceptual view of the modified NoC router microarchitecture. The working is explained in the following sub-sections:

A. Storing Evicted L1 Cache Blocks in Routers

The proposed architecture targets evicted, dirty L1 cache blocks on their way for the write-back. As shown in Figure 4, the modified header has a 1-bit flag *Store* to identify write-back messages. These messages enter NoC as packets through the local routers to travel towards their destination for write-back. While these in-transit packets are stored in the VCs for routing and arbitration, an additional **Store, Reply, Forward & Drop (SRFD)** unit (refer Figure 3) checks their *Store* flag. SRFD unit avoids critical path by working in parallel with the RC unit, as we use two-stage NoC routers (1: RC, 2: VA and SA). If SRFD unit finds the *Store* flag SET for a packet, it disables the VC and switch arbitration (VA and SA). The SRFD unit in this way stores evicted, dirty L1 cache blocks in the local router.

The proposed architecture also considers sending some evicted, clean L1 cache blocks for write-back, which usually are discarded (dropped). The LLC bank controller (LLC CTLR) never expects these blocks for write-back. They are kept stored in the local routers to increase the chances of local reply (to be explained in Section IV-B). Since the clean blocks which are in S state have multiple sharers, only the blocks in E state are sent for write-back to avoid incoherence (to be explained in Section IV-D). Clean write-back messages also use the *Store* flag, much like dirty write-back messages. To facilitate local reply, both of these messages use another 1-bit flag *Dirty*. The actions of the SRFD unit in the modified routers are shown in Figure 3.

B. Replying to Re-reference Requests from Routers

The proposed architecture identifies read/write requests in the local routers on their way to the destination. Read and write requests are identified using 1-bit flags *S* and *X*, respectively (refer Figure 4). When a new packet enters the local router with its *S* flag SET (read request), the requested address is compared with the addresses of the stored blocks (packets). If SRFD unit finds a match, it initiates a local reply to the read request by swapping the source and destination between the matched block (stored packet) and the read request (new packet). SRFD unit enables the VA and SA for the stored packet which were disabled during local store (refer Section IV-A). The stored packet simply gets ejected through the local output port as its destination L1 cache is connected to the very same router. Local reply reduces miss penalty significantly by avoiding travel to fetch the requested block. In essence, a read request is satisfied with a stored clean/dirty block from the local router.

SRFD unit works similarly when a new packet enters the local router with its *X* flag SET (write request). However, there is an additional check of the *Dirty* flag to identify whether the matched block is clean or dirty. Only when the *Dirty* flag is found SET, a write request can be satisfied with a local reply. If not SET (matched block is clean), then the write request travels towards LLC bank to fetch the requested block (to be explained in Section IV-D). The matched clean block is dropped before the write request proceeds towards LLC bank to avoid leaving an old version of the block behind. Note that a write request can be satisfied only with a matching block which is dirty and not clean.

C. Forwarding/Dropping of Stored Cache Blocks

When a new packet does not get a free buffer (VC) in the local router for injection, SRFD unit vacates a VC occupied by the stored blocks (evicted clean or dirty). The oldest one is vacated in case multiple VCs have stored blocks. After the VC to be vacated is identified, SRFD unit checks the *Dirty* flag of the block stored in that VC. If the *Dirty* flag is found SET, the stored block is dirty and hence is forwarded for write-back by enabling VA and SA. Otherwise, the stored block is clean and is silently dropped since LLC bank already has the block.

There can be other applications waiting in the LLC CTLR to get the status of evicted, L1 cache blocks to resume their execution. Hence, even in the absence of injection pressure, when the VCs are not yet full, some of the stored blocks need to be either forwarded or dropped. The proposed architecture employs two techniques to identify the VC to be vacated; **Time-Triggered VC Vacate (TT-VCV)** and **Message-Triggered VC Vacate (MT-VCV)**. TT-VCV dictates that an evicted block remains stored in the local router till the expiry of a specific time threshold. With each VC, a threshold counter (τ_i) is attached in the local input port (refer Figure 3). MT-VCV dictates that when the status of a stored block is requested by the LLC CTLR, the VC containing the stored block is vacated.

Table 1: Simulation configuration

Processor	64 OoO x86 cores
L1 Cache	16KB, 4-way, 64B blocks, private, split
L2 Cache (LLC)	128KB×64 cores, 8-way, 64B blocks, shared
Memory Bank	4; one located at each corner
Cache Coherence	MOESI distributed directory
NoC	8×8 2D mesh, 128-bit flit channel 3 Virtual Networks (VNs), VN0, VN1, VN2 4 Virtual Channels (VCs)/VN 1-flit depth control VC, 5-flit depth data VC
Routing	2-stage routers, X-Y dimension-order routing
Packets	1-flit for control packets, 5-flit for data packets
Time Threshold (τ)	256 cycles
Benchmarks	SPEC CPU2006 (multi-programmed)

D. Maintaining Cache Coherence

Evicted L1 blocks are stored only in the local routers. Dirty blocks are always private (no sharer), so when they are stored in local routers, there is no violation of coherence. However, when local replies are generated with these stored, dirty blocks, their state requires to be preserved. Hence, irrespective of whether its a read or write request, a local reply with a stored, dirty block is always sent in M state. Since a block in M state can not be discarded, eventually it will be forwarded for write-back after eviction from L1 cache in the near future.

Clean blocks can be shared (S state) or private (E state). Since storing a shared cache block in routers may violate coherence, the proposed architecture considers only private, clean blocks. So, coherence is not violated while these blocks are kept in the local routers. When re-referenced, a local reply with a stored, clean block is always sent in E state. However, this is not enough for a write request. An application initiates a write request to modify the content of a block, and hence it needs the block in M state. If local reply for a write request is generated in E state, the application goes ahead with its execution and modifies the block. However, these changes are not updated to the next level of memory, since a block in E state is never sent for write-back. This leads to data inconsistency which violates coherence. To avoid all of these, the proposed architecture uses stored clean blocks to satisfy only read requests from the local router.

V. Performance Evaluation

The following architectures are evaluated:

- **Baseline:** Bare version with no optimisation.

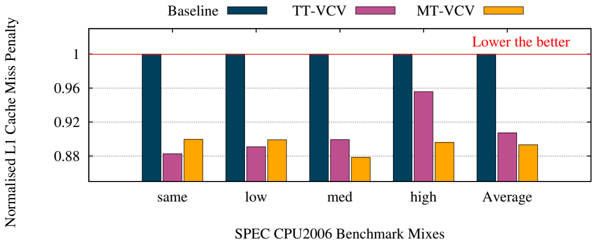


Figure 5: L1 cache miss penalty

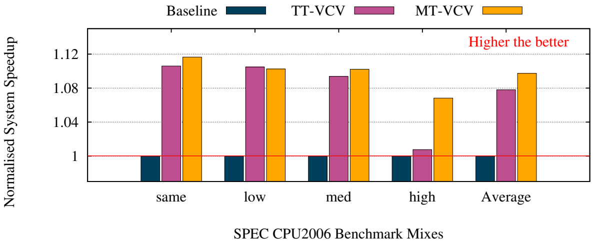


Figure 6: Overall system speedup

Table 2: Workload mixes

Mix	Benchmarks	Copies
Same	astar, cactusADM, GemsFDTD, gromacs, h264ref, hmmer, lbm, leslie3d, namd, omnetpp, perlbench, sjeng, soplex, sphinx, xalancbmk	1×64: 64
Low	gromacs, GemsFDTD, hmmer, astar, h264ref	4×16: 64
Med	sphinx, perlbench, cactusADM, omnetpp, soplex	
High	xalancbmk, namd, sjeng, leslie3d, lbm	

- **TT-VCV:** Stores evicted L1 blocks in local router and uses time-triggered VC vacate.
- **MT-VCV:** Stores evicted L1 blocks in local router and uses message-triggered VC vacate.

A. Simulation Framework and Workloads

Event-driven gem5 simulator [16] is used to model all the architectures, with the system configuration presented in Table 1. To evaluate performance, SPEC CPU2006 benchmarks are run on the baseline and proposed architectures. Different workload mixes are created based on the re-reference time of benchmarks (refer Figure 2a) as given in Table 2. *Same* is a homogeneous workload mix that runs the same benchmark on all the cores (1×64: 64). *low*, *med* and *high* workload mixes are created by grouping benchmarks with low, medium and high re-reference times, respectively. Each of these mixes run 4 different benchmarks with 16 copies from their groups, in a random combination (4×16: 64). We plot normalised results of proposed work compared with the baseline architecture.

B. Results and Discussion

L1 Cache Miss Penalty: It is the number of cycles spent in bringing a requested block into the L1 cache. Figure 5 shows whether the optimisations employed in the proposed architectures are effective in reducing L1 cache miss penalty. Local reply from routers reduces L1 cache miss penalty in the proposed architectures for all the workload mixes, when compared with the baseline. In general, *same* and *low* are performing slightly better than *med* and *high* mixes as they have benchmarks with low re-reference time (refer Figure 2a). A maximum reduction of up to 13% and an average reduction of 11% in L1 cache miss penalty is achieved.

Overall System Speedup: The overall system speedup is compared using the total instructions per cycle (IPC) of an architecture; baseline and the proposed. Significant reduction in L1 cache miss penalty has already signalled towards improved system speedup. Figure 6 shows how the proposed architectures have a greater system speedup when compared with the baseline. A maximum of up to 12% and an average of 10% overall system speedup is achieved for the presented workload mixes.

Limitations: Since the proposed architectures rely on availability of VCs, a sensitivity analysis on the number of VCs will help better assess the work. Additionally, as the count of evicted blocks is much higher than the capacity of available VCs, there can be instances of unwanted thrashing.

C. Storage, Area and Power Overhead

Far smaller than the 128-bit flit channel, a standard packet header is only around 64-bits. Hence, the additional flags *Store*, *Dirty*, *S* and *X* can be accommodated in the header without any storage overhead. McPAT [17] is run at 22nm processor technology by feeding the configuration and output files of gem5 [16] to get the area, leakage and dynamic power overheads. A small area overhead of 1.82% and a leakage power overhead of 2.73% is incurred in the proposed architectures, due to the inclusion of SRFD unit and threshold counters (τ_i). Nevertheless, a significant reduction of 4.39% in dynamic power is achieved with improved system performance. SRFD unit avoids the critical path of execution as it functions with the RC unit in parallel (refer Section IV-A). However, A real hardware synthesis will give more accurate overheads.

VI. Conclusion

This article explored one of the most common challenges latest consumer electronics face due to increasing processing cores. Based on some critical observations, the proposed architecture puts NoC into the memory hierarchy using router buffers as a victim cache. It significantly reduces miss penalty and increases system performance. Our future work includes a head-to-head comparison with a state-of-the-art victim cache based architecture. We also plan on real hardware synthesis for overhead analysis and a case study on an actual system.

References

[1] (2020) Smartphone Processors Ranking. [Online]. Available: https://nanoreview.net/en/soc-list/rating

[2] (2019) AMD unveils world's most powerful desktop CPUs. [Online]. Available: https://tinyurl.com/fastest1

[3] D. Sanchez, G. Michelogiannakis, and C. Kozyrakis, "An analysis of on-chip interconnection networks for large-scale chip multiprocessors," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 7, no. 1, pp. 1–28, 2010.

[4] P. Gratz and S. W. Keckler, "Realistic workload characterization and analysis for networks-on-chip design," in *4th workshop on chip multiprocessor memory systems and interconnects*, 2010, pp. 1–10.

[5] C. Fallin, G. Nazario, X. Yu, K. Chang, R. Ausavarungnirun, and O. Mutlu, "Minbd: Minimally-buffered deflection routing for energy-efficient interconnect," in *2012 IEEE/ACM Sixth International Symposium on Networks-on-Chip*. IEEE, 2012, pp. 1–10.

[6] H. Matsutani, M. Koibuchi, H. Amano, and D. Wang, "Run-time power gating of on-chip routers using look-ahead routing," in *2008 Asia and South Pacific Design Automation Conference*. IEEE, 2008, pp. 55–60.

[7] G. Michelogiannakis, D. Sanchez, W. J. Dally, and C. Kozyrakis, "Evaluating bufferless flow control for on-chip networks," in *2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*. IEEE, 2010, pp. 9–16.

[8] H. E. Mizrahi, J.-L. Baer, E. D. Lazowska, and J. Zahorjan, "Introducing memory into the switch elements of multiprocessor interconnection networks," in *Proceedings of the 16th annual international symposium on Computer architecture*, 1989, pp. 158–166.

[9] L. Huang, "Leveraging on-chip networks for efficient prediction on multicore coherence," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–4.

[10] W. Shu and N.-F. Tzeng, "Nuda: Non-uniform directory architecture for scalable chip multiprocessors," *IEEE Transactions on Computers*, vol. 67, no. 5, pp. 740–747, 2017.

[11] A. K. Swain, A. K. Rajput, and K. K. Mahapatra, "Network on chip for consumer electronics devices: An architectural and performance exploration of synchronous and asynchronous network-on-chip-based systems," *IEEE Consumer Electronics Magazine*, vol. 8, no. 3, pp. 50–54, 2019.

[12] M. Robaei and H. Zhao, "Broadcast-based hybrid wired-wireless noc for efficient data transfer in gpu of ce systems," *IEEE Consumer Electronics Magazine*, vol. 8, no. 6, pp. 62–67, 2019.

[13] A. Das, A. Kumar, and J. Jose, "Reducing off-chip miss penalty by exploiting underutilised on-chip router buffers," in *2020 IEEE 38th International Conference on Computer Design (ICCD)*. IEEE, 2020, pp. 230–238.

[14] A. Das, A. Kumar, J. Jose, and M. Palesi, "Exploiting on-chip routers to store dirty cache blocks in tiled chip multi-processors," in *IEEE Computer Society Annual Symposium on VLSI*, 2020, pp. 147–152.

[15] N. P. Jouppi, "Improving direct-mapped cache performance by the addition of a small fully-associative cache and prefetch buffers," *ACM SIGARCH CAN*, vol. 18, no. 2SI, pp. 364–373, 1990.

[16] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 simulator," *ACM SIGARCH CAN*, vol. 39, no. 2, pp. 1–7, 2011.

[17] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "Mcpat: an integrated power, area, and timing modeling framework for multicore and manycore architectures," in *IEEE/ACM International Symposium on Microarchitecture*, 2009, pp. 469–480.

About the Authors

Abhijit Das is a PhD Scholar at Indian Institute of Technology Guwahati, India. Contact him at abhijit.das@iitg.ac.in.

Abhishek Kumar is a Member of Technical Staff at Oracle Inc., Bengaluru, India. Contact him at ak60370@gmail.com.

John Jose is an Assistant Professor at Indian Institute of Technology Guwahati, India. Contact him at johnjose@iitg.ac.in.

Maurizio Palesi is an Associate Professor at University of Catania, Italy. Contact him at maurizio.palesi@dieei.unict.it.