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/TCAD.2019.2907909, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

Enhancing Network-on-Chip Performance by Reusing Trace Buffers

Neetu Jindal, Shubhani Gupta, Divya Praneetha Ravipati, Preeti Ranjan Panda, Smruti R. Sarangi

Department of Computer Science and Engineering, Indian Institute of Technology Delhi, New Delhi, India
E-mail: {neetu, shubhani, divyapraneetha, panda, srsarangi}@cse.iitd.ac.in

***Abstract*—Ensuring the functional correctness of Networks-on-Chip (NoC) can be particularly challenging, and communication-centric debug methodologies have been widely used by engineers to validate NoC functionality during post-silicon validation. Design-for-Debug structures such as trace buffers and monitors are usually inserted in such Systems-on-Chip to enhance signal visibility. However, this debug hardware becomes underutilized once the chip goes into production. While the size and organization of the router buffers directly impact network throughput, these buffers also dominate the on-chip router area. We propose a scheme AugVC to reuse trace buffers to augment router buffers, with the objective of improving the overall network performance. Experimental results for a 64-node mesh network show that our proposed approach can reduce latency by up to 38.25% for transpose traffic compared to a baseline design with reduced buffer sizes. We also propose an extension, Output Port directed Virtual Channel (ODVC), that uses a modified virtual channel assignment strategy, on the basis of the designated output port of a network packet. This strategy reduces the average packet latency and area of the router by 45% and 32.4%, respectively.**

***Index Terms*—Network-on-chip (NoC), Design-for-debug, post-silicon validation, trace buffer, virtual channels.**

I. Introduction

System-on-chip (SoC) complexity has increased significantly in terms of both the processor core count as well as the communication among these cores, which makes debugging these systems very challenging. A post-silicon validation phase is usually necessary for ensuring proper system validation. In addition to the traditional focus on component functionality, validation researchers have recently focused on a communication-centric debug methodology to ensure the correctness of on-chip networks. A large part of debug complexity lies in validating the interaction between the system components.

Debugging during post-silicon validation is aided by Design-for-Debug (DFD) hardware which provides visibility into the chip by recording its state. This includes monitors and trace buffers. Monitors are programmed to observe whether the system satisfies various specified conditions [1] and trace buffers are the memory elements that record the router state periodically. This state consists of the flits stored in the router buffers, intended for forwarding to other routers. Router buffers are First-In, First-Out structures that temporarily store the packets that cannot be immediately forwarded due to contention. A flit is a fixed-size unit, one or more of which constitute a packet [2]. There is a tradeoff involved in designing the DFD hardware; increasing this hardware provides higher visibility during debug, but the increased area goes largely unutilized during normal system functioning.

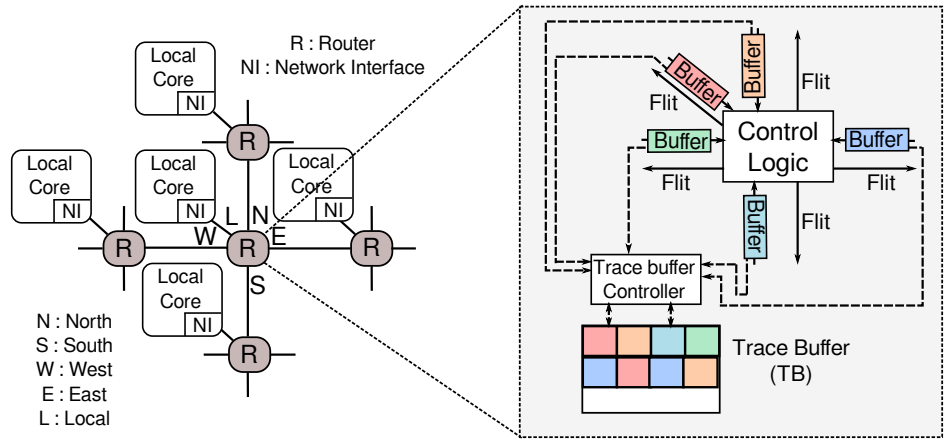
A router buffer can be organized either as a single queue as in a wormhole router, or further divided into multiple independent queues called virtual channels (VCs) to avoid head-of-line blocking as in a virtual channel router [3]. These buffers have a significant impact on the network throughput, especially when the network becomes congested. Increasing the buffer size reduces the packet drop probability [4]; however, it also increases the on-chip router area [5], [6] and power [7], [8]. Consequently, the buffer design plays an important role in architecting low cost, high performance, and energy efficient on-chip networks.

We propose to leverage the presence of a trace buffer within the router to effectively increase the capacity of the router buffers present at each port. The trace buffer is reused to augment the router buffers, with the objective of improving the overall network performance. Since this buffer already exists in the hardware, our proposed reuse incurs minimal area overhead. The scenario is illustrated in Figure 1. Figure 1a shows the trace buffer being used to store debug data during the post-silicon validation phase [9] and Figure 1b shows the normal operation where the design-for-debug hardware is power-gated. In Figure 1c the trace buffer is re-used to aid the normal system operation in-field. Here, the trace buffer is used as a backup storage; it stores the flits that cannot be accommodated in the private buffers present at the input ports.

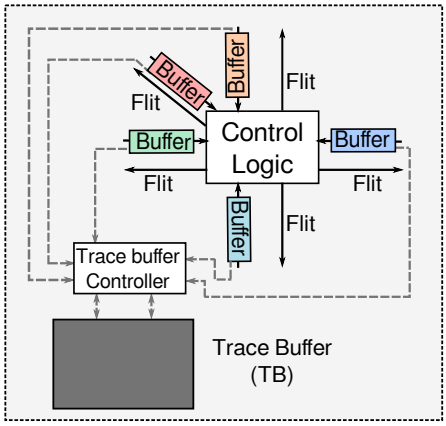
Our main contribution is to demonstrate that such a reuse improves network performance with reduced area, through using smaller VC buffers, with the trace buffer serving as backup storage for flits. We also propose an extension, called Output port Directed Virtual Channel (ODVC), that uses a modified VC assignment where the VCs are assigned on the basis of the designated output port of the packet. The cardinality of the arbiters used in the switch allocation and VC allocation stage of the pipeline affects the clock period of the router. Our proposed design reduces the cardinality of the different arbiters, and thus, reduces the clock period and area of the router by 30% and 32.4% respectively.

This paper is organized as follows. In Section II we summarize the background on the conventional router architecture. In Section III we review the related work on both validation hardware and management of on-chip router buffers. In Section IV we present our proposed router architecture with all the components in detail. We discuss the experimental results

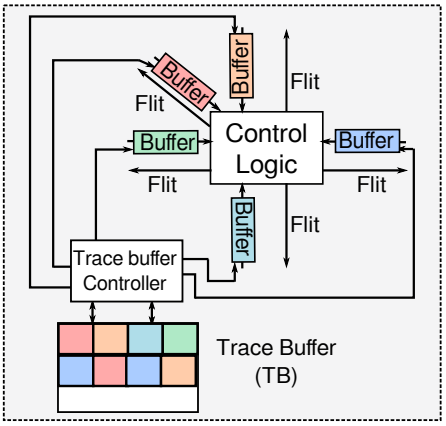
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2907909, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems



(a) During Debug



(b) During Operation



(c) During Operation (proposed)

Fig. 1: Trace Buffer reused to augment router buffers.

in Section V. Finally, we conclude the paper in Section VI.

II. Background: Conventional Router Architecture

We first review the conventional on-chip router architectures. Figure 2 depicts the input queued VC router architecture in a mesh topology, with five ports corresponding to the four directions (North, South, West, and East) and the local core. The router is implemented as a four-stage pipeline:

1) Look ahead routing and VC allocation
2) Switch arbitration
3) Switch traversal
4) Link traversal

Each port has an input buffer consisting of multiple independent queues called virtual channels (VCs) that operate in parallel. All flits of a packet occupy the same VC. This allows the traversal of flits from multiple packets in an interleaved manner over a single physical channel as the VCs are assigned on a per-packet basis. When a header flit arrives at a router, it is first buffered in the VC's buffer based on the VC identifier, following which the routing computation unit finds the output port based on the routing algorithm and the destination information present in the header flit. To eliminate the delay associated with the route computation from the router's critical path, *Look Ahead Routing* (LAR) is used [10], where the current router determines the output port that the next router will forward a flit to. For each header flit, the VC allocation unit attempts to allocate an output VC (this refers to the input VC in the downstream router). The VC allocation unit performs arbitration among all the flits requesting the same output VC. Non-availability of an output VC for a header flit causes VC allocation to be re-tried in the next cycle. LAR and VC allocation are performed in parallel. If a flit secures a free VC, it proceeds to the Switch Arbitration (SA) unit. The SA unit then determines which flits are to be forwarded to their corresponding output ports. SA is performed in two stages: the first stage selects one of v VCs at each input port, requiring a total of p arbiters of cardinality $v:1$. The second stage arbitrates between the winning requests of the first stage to decide who wins each output port. Therefore, a total of p arbiters of cardinality $(p-1):1$ are required. The flit then traverses the switch to reach the output port in switch traversal stage. It then traverses the link in the Link traversal stage to reach the downstream router. LAR and VC allocation are done only for the header flits, whereas switch allocation is done on a per-flit basis. Once the tail flit leaves the router, it deallocates the VC reserved for the packet. Each pipeline stage takes one cycle to execute. This organization uses a static partitioning

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/TCAD.2019.2907909, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

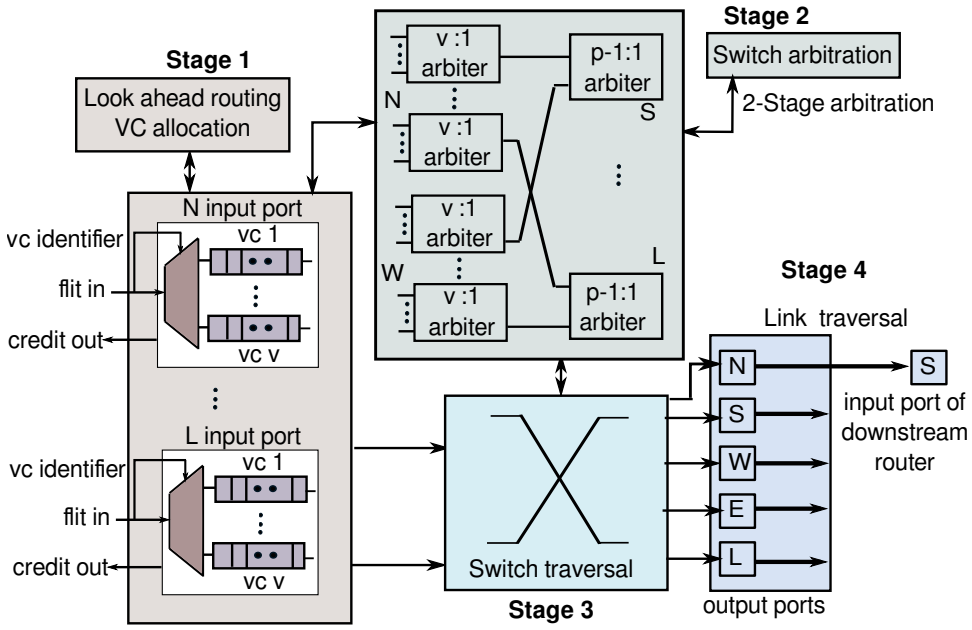


Fig. 2: Conventional Router Architecture.

of buffer resources where each input port has a fixed number of VCs, each with a fixed flit count.

Since NoC applications cannot tolerate dropping of packets, a credit based flow control mechanism is used. Flits are forwarded only if the next-hop router has buffers available at its corresponding input port. Each router knows the number of unassigned VCs in each of its downstream routers, as well as the number of available slots in the assigned VCs. While allocating a VC for a head flit, the router looks for an unassigned VC in the downstream router. When forwarding a body or tail flit, the router looks for a free slot in the corresponding private VC. Whenever a flit leaves a router, the upstream routers are requested to update their credit knowledge accordingly.

III. Related Work

Post-silicon validation has received considerable research attention in recent times. Techniques have been proposed for efficient validation of processors and logic [11], [12]. Goossens et al. [13] propose a communication centric debug architecture that uses structural and temporal abstraction techniques to visualize the SoC's state at a logical communication level. Monitors are used for performance analysis and validation of data flow errors of network operations [1] and observing traffic events [14], [15]. These events and transactions are then transferred over the network for further analysis. Gharehbaghi and Fujita [16] proposed an approach to monitor the bus in which the transactions are stored in a trace buffer. The trace buffer content is analyzed offline for anomalous patterns, which requires the validation engineer to identify anomalous transaction traces and perform backtracking from the observed failure state. A transaction based online debug approach is proposed by Dehbashi and Fey [17]. Ghofrani et al. [18] also propose a solution for online detection and diagnosis of permanent faults in NoCs. This approach uses an error syndrome collection mechanism to localize datapath faults and timeout based techniques to localize the fault in control logic.

Debugging techniques that target functional errors such as deadlock, livelock, and starvation errors by taking snapshots of packets traversing a router to reconstruct the packet path, have been studied by Abdel-Khalek and Bertacco [19]. Additional validation hardware consisting of small buffers used to store a copy of header flits and snapshot buffers are added at every port of the router. A transaction buffer of size 32×128 is used inside the debug probe by the NoC-based multicore debug platform reported by Tang et al. [9]. A run-time solution REPAIR [20] has also been proposed to recover from functional design errors using retransmission based techniques. DiAMOND [21] focuses on the validation of functional bugs in the control flow portion of the NoC designs. It uses in-flight packets to log debugging data by replacing the data contents of a packet with debugging data. Hardware checkers are added to the routers to monitor the execution and flag functional bugs.

Efficient management of on-chip router buffers is another related research area. Buffers are instrumental in the overall operation of on-chip networks. Kumar et al. [4] discuss the variation of packet drop probability with buffer size. Increasing the buffer size reduces the drop probability; however, the area occupied by an on-chip router is dominated by these buffers. Reducing the size of these buffers is generally not feasible because it directly affects the packet latency.

The VC organization impacts overall system performance. Rezazad et al. [22] evaluate the trade-off between the number and depth of the VCs for a fixed size buffer. A small number of VCs are sufficient for low intensity traffic; however, increasing the number of VCs is more effective for high intensity traffic rather than increasing the buffer depth. Huang [23] customizes the VCs according to the bandwidth utilization of each port in a static approach based on the detailed analysis of the application-specific traffic patterns. Bufferless routing meth-

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/TCAD.2019.2907909, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

ods [24]–[26] remove buffers from the router to save area and power; however, they result in larger latencies when the packet injection rate is high.

The size of VC queues could be controlled either statically or dynamically. Static queues are easier to implement, whereas dynamic queues significantly improve buffer utilization at the cost of increased complexity of the related control logic. Various dynamic VC management techniques [2], [27], [28] have been proposed in recent times. They unify the buffer resources into a common pool and allocate to different VCs at run-time. The ViChar design [27] dynamically varies the number and size of VCs depending on traffic conditions. It maintains similar packet latencies even with half the buffer size, but requires an arbiter count equal to the buffer size in both VC allocation and switch allocation stages, which incurs a significant area overhead. The RoShaQ design [29] uses shared queues between all the ports instead of a fixed number of queues on individual ports. Xu et al. [2] improve over ViChar design by using a VC allocation mechanism where VCs are assigned on the basis of a designated output port. It maintains a correspondence between the output port requested on the downstream router and the output VC assigned in the current router. However, it is a priority based scheme, and a fixed mapping between an output port and a VC is not feasible in this design. The number of VCs allocated to the output port depends on the traffic condition. In our work, we use a fixed mapping by associating each queue at the input port to a single output port, which enables us to reduce the cardinality of arbiters, resulting in a smaller area and clock period. We have considered the mesh topology in our design; however, reuse opportunities do exist for other topologies such as the ring used in NuMachine multiprocessor [30], where the trace buffer can be used at the inter-ring interface to facilitate the communication between the local and central rings. Similarly, the trace buffer can also be reused to reduce the size of extension buffers in routerless NoC architectures [31].

The virtual output queuing (VOQ) technique [32] also uses a fixed mapping of each VC to a single output port. However, it requires a sophisticated switch allocation algorithm such as ApSLIP [33] or maximum weight matching to support high network throughput, resulting in significant power, area, and clock cycle time overheads. This makes it hard to use in NoCs [34], but it is commonly used for off-chip networks [35] which do not need flow control, can drop packets upon congestion, and are not sensitive to the clock period. VOQ organization is more suitable for a topology with good load balancing characteristics such as clos NoC [36]. Our proposed technique uses a simple round-robin switch allocation, and still achieves the benefits of VOQ routers. The key difference that enables this with a much smaller area overhead is the reuse of the trace buffer for backup storage, which reduces the input buffer sizes. Queue lengths in case of non-uniform traffic are not a matter of concern in our design, as the trace buffer stores the flits if the corresponding port's queue is full. Zhang et al. [28] also proposed a buffer sharing mechanism where a shared buffer is present on every port, with a small prefetch buffer for every VC to reduce the delay in reading data from the shared buffer. This approach cannot be used in a trace buffer scenario as a single trace buffer unit is available for all the ports.

Recent works by Basak et al. [37] and Jindal et al. [38], [39] have also identified the advantages of reusing DFD hardware in-field, and proposed the re-purposing of these structures. Basak et al. [37] exploit the DFD structures to implement customized security wrappers that enable on-field update of security requirements, while being transaparent to debug use cases. Jindal et al. [38], [39] proposed a non-standard victim cache design which reuses the storage area of the trace buffer to enhance in-field performance.

IV. Trace Buffer Augmented Router Architecture

We propose to integrate the Trace Buffer, a Design-for-Debug structure already present in the router, into the router architecture, which enables the reclamation and reuse of its storage space for functional purposes. Since all the input ports may not receive packets at the same time, many VCs may go un-utilized even if a demand for VCs exists. We propose to reuse the trace buffer as augmented VC buffers (AugVC). This reuse of the DFD infrastructure reduces the NoC's packet latencies, while also permitting a reduction in the area occupied by the input buffers. In this section we first discuss our proposed router architecture design AugVC. Following this, we discuss an extension in which we use an output port directed VC allocation scheme to reduce the clock period and router area.

A. AugVC Router Architecture

In the AugVC design, we propose to reduce the width of the private VCs present at each port and use the trace buffer as a backup storage. To support this new functionality, a new controller is added. The controller first checks the availability of private VCs for an incoming flit and if no space is available in the private VCs, it accommodates those flits in the trace buffer. Similarly, for an outgoing flit, it checks whether a flit of the same packet is present in the trace buffer, in which case, it migrates the flit from the trace buffer to the private VC. Each port can receive a maximum of 1 flit per cycle. Now, a trace buffer of width w flits only serves as a backup storage for w ports because a trace buffer of width w flits allows a maximum of w flits to be written to the trace buffer per cycle. Let the number of ports be m and the width of the trace buffer be w flits, there are two possible scenarios:

1) $m \leq w$: In this case, the trace buffer can be used as a backup storage for all the m ports, which further allows to reduce the size of all the private VCs.
2) $m > w$: This scenario is elaborated further. Here, the trace buffer can only be used for w ports and the remaining $m - w$ ports need to use the larger VC sizes to maintain NoC performance.

Figure 3 illustrates the proposed Augmented Virtual Channel (AugVC) router architecture. The additional components required to support the new functionality are highlighted in blue. For illustration, we consider a router architecture with five input and output ports (North, South, West, East and Local), a flit size of 32 bits [29], and a trace buffer of width

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/TCAD.2019.2907909, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

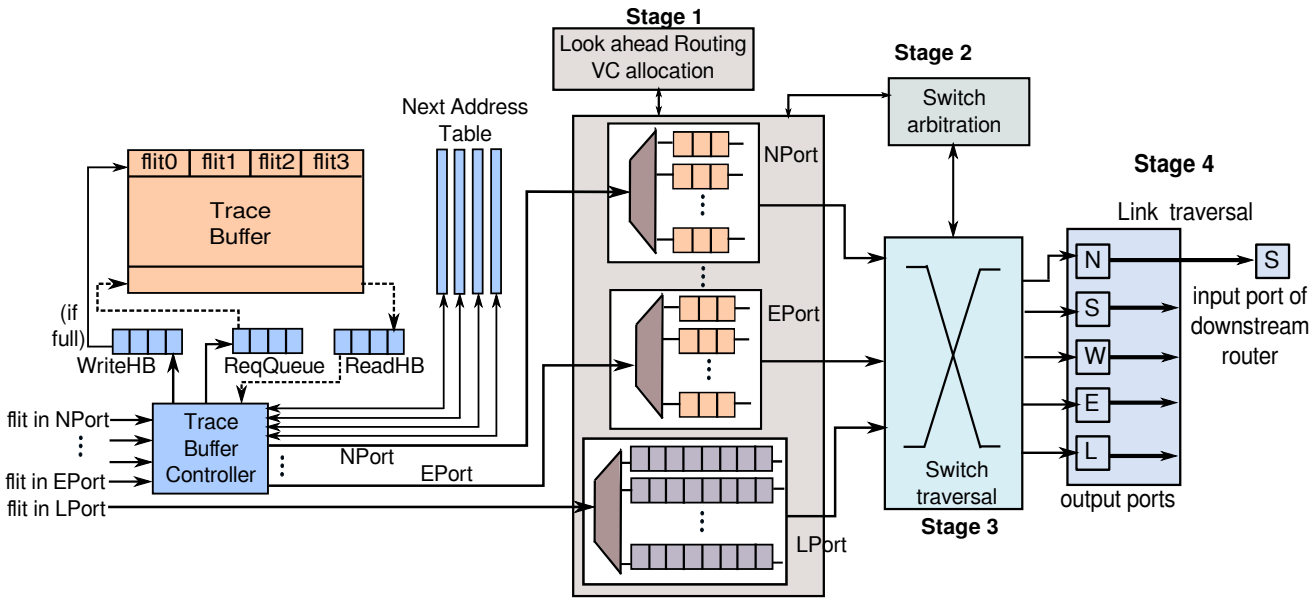


Fig. 3: AugVC Router Architecture

128 bits [9]. The trace buffer can accommodate only four ports, because a maximum of four flits can be written to the trace buffer per cycle. This allows us to migrate flits from a maximum of four ports. We therefore permit the four non-local ports (N, S, W, E) to utilize the trace buffer and consider the local port differently. These can be any four ports. For illustration and evaluation, we have used four non-local ports. The local port has v VCs, each is of size k flits. The other four ports (N, S, E, and W) also have v VCs each, each of size k' flits, where $k' < k$. A relatively small VC size is possible because of the trace buffer being available for backup storage.

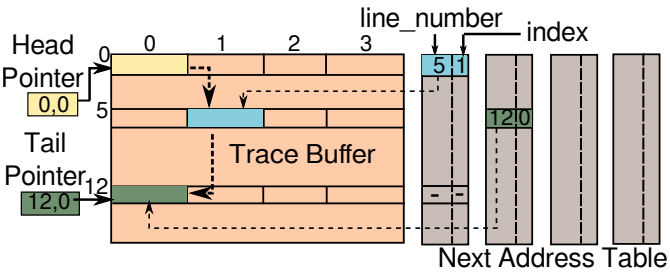


Fig. 4: Linked list of all the flits belonging to the same VC maintained using Next Address Table

Incoming flits from the local core can occupy only one of the k private VC slots available at the local port. Incoming flits that need to be forwarded can occupy the allocated private VC slots (k'), or may be accommodated in the trace buffer if the private VC is full.

We also employ a credit-based flow control system. Each router knows the number of unassigned VCs in each of its downstream routers, the number of available slots in the assigned VCs, as well as the number of available lines in the trace buffer. While allocating a VC for a head flit, the router looks for an unassigned VC in the downstream router, just as in the base router architecture. When forwarding a body or tail flit, the router looks for either a free slot in the corresponding private VC, or an available line in the trace buffer. Whenever a flit leaves a router, the upstream routers are requested to update their credit knowledge accordingly.

When a flit leaves the router, a flit of the same packet is migrated from the trace buffer (if available) to the newly freed slot in the private VC. Since the trace buffer is uniformly shared among all non-local ports, the storage is better utilized than in the case of completely private VCs as employed by the baseline router architecture.

Writing to the Trace Buffer: If there is no space to accommodate an incoming body or tail flit from a non-local port in the private VC, the Trace Buffer Controller stores the flit in a temporary area called the *Write Holding Buffer* (Write HB). Once the occupancy of the Write HB reaches four, the flits are written to the trace buffer (since writes to the trace buffer are done in blocks of 128 bits or 4 flits). This is done to utilize the trace buffer space efficiently as we may not have flits from all the ports writing into the trace buffer in the same cycle. Thus, a line in the trace buffer may possibly contain data from different VCs of different ports or the same port.

Reading from the Trace Buffer: When a flit exits the private VC, the trace buffer controller checks whether a flit corresponding to that VC is present in the trace buffer, and if so, writes it to the private VC. To search the trace buffer, the Trace Buffer Controller maintains a linked list of all flits residing in the trace buffer that belongs to the same VC in an auxiliary *Next Address Table* (Figure 4). The same two values are required to index into the trace buffer and the Next Address Table: line number and position of the entry within the line. For each VC, a head pointer (<line number, position> tuple) and a tail pointer are maintained, which contain the location of the first and last flits residing in the trace buffer. For a flit f residing at index i_f in the trace buffer, the entry at index i_f in the Next Address Table gives the index of the flit following f. The size of Next Address Table is derived from the number of lines and the width (flits per line) of the trace

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/TCAD.2019.2907909, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

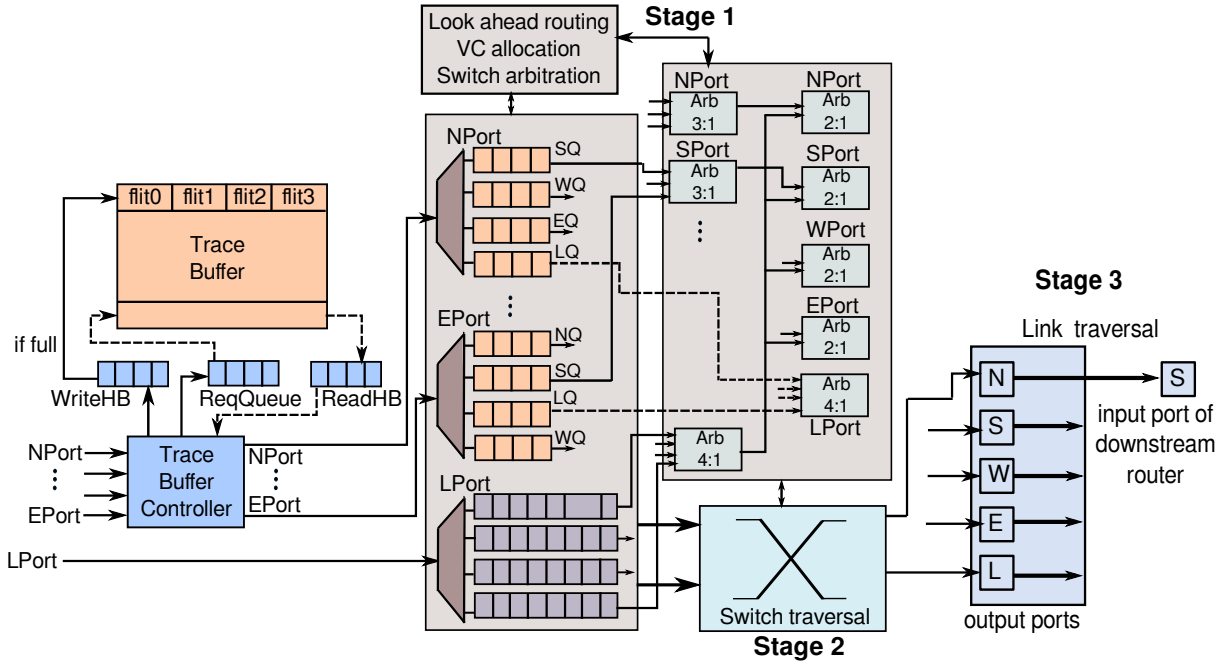


Fig. 5: Output Port Directed VC Assignment (ODVC)

buffer. For example, if the number of lines is n and the trace buffer is w flits wide, then the size of the next address table is $n \times ((\log_2 n + \log_2 w) \times w)$ bits. The parameters n and w are fixed at design time.

If the head pointer is not null, the trace buffer controller adds a request for the VC in the *Request Queue*. In one cycle, up to four requests may be added, one from each input port. In every cycle, the head of the Request Queue is processed and a line is read from the trace buffer and placed in a temporary area called the *Read Holding Buffer* (Read HB). Further, in every cycle, the Request Queue is scanned to find all those requests that the contents of the Read and the Write HB can satisfy. A flit in the Read HB can be used to satisfy a request only if it is pointed to by the head pointer corresponding to that VC. A flit in the Write HB can be used to satisfy a request only if no flit of that VC is present in the trace buffer. These conditions are necessary for ensuring that the flits are processed in their order of arrival.

A significant fraction of the requests is serviced directly from the Read and Write HBs, reducing the number of trace buffer reads. This is due to the presence of temporal locality; flits that arrive together at the router, tend to enter the trace buffer, and also leave the router together. No changes are made to VC allocation and switch arbitration stages of the pipeline in the AugVC design.

B. Output Port Directed VC Assignment (ODVC)

The VC allocation unit and the switch allocation (SA) unit typically form the bottleneck stages of the router pipeline [27], affecting the clock period of the router. The latencies of these two stages depend on the cardinality of the arbiters used in these stages. During VC allocation, the unit may have to arbitrate as many as $v \times p$ requests in a single cycle, where v is the number of VCs per port, and p is the number of ports. This may make the unit reasonably complex, requiring a relatively large clock period. SA is performed in two stages: the first stage selects one of v VCs at each input port, requiring a total of p arbiters of cardinality $v : 1$. The second stage arbitrates between the winning requests of the first stage to decide who wins each output port. Therefore, a total of p arbiters of cardinality $(p - 1) : 1$ are required. The two stages of arbitration require a large clock period, and also consume a significant fraction of the router area. Our proposed design reduces the cardinality of the different arbiters, and thus, reduces the overall clock period and the router area. To achieve this, an output directed VC allocation scheme is used.

Figure 5 describes the new router architecture. The VCs are populated on the basis of the intended packet direction, unlike in the baseline, where the VCs are reserved on a per-packet basis. For illustration, here also, we have considered 5-port design (North, South, West, East and Local) and have also considered the local and the non-local ports separately for the same reason as in Section IV-A. Flits from the local core are not accommodated in the trace buffer as the trace buffer is 128 bits (4 flits) wide, and a maximum of 4 flits can be written into it every cycle. This allows us to migrate flits from a maximum of four ports. We therefore choose the four non-local ports to utilize the trace buffer.

Each of the four non-local input ports has four VCs corresponding to the four possible output ports. Illustrating through an example, the North input port has four VCs: SQ, WQ, EQ, and LQ. A flit arriving at the North input port, and expected to leave through the South output port is accommodated in the SQ of the North port. This is possible because of the look ahead routing done in the previous router. If a VC in any of the input ports is found full, the flit is accommodated in the trace buffer, space permitting. The writing to and reading from the

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/TCAD.2019.2907909, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

trace buffer is performed as in Section IV-A. The ordering of flits within a packet is still maintained. Flits originating from the local core are allowed to occupy any of the VCs in the local port, regardless of the intended direction of the flits.

The new architecture described above uses a 3-stage pipeline. When a flit arrives at an input port, SA and LAR are first performed in parallel, followed by VC allocation. These three steps are performed in the same cycle, unlike the base architecture, where VC allocation is performed in parallel to LAR, followed by SA in the next cycle. Additionally, in the base architecture, LAR and VC allocation were performed only for header flits, while these steps are performed for all flits in the ODVC architecture (flit-level routing) [40]. Here, every flit needs to contain routing information. This is necessary, as in ODVC, flits of different packets occupy the same VC. The overhead of routing information with every flit is considered in ODVC experiments by increasing the number of flits in a packet.

SA is performed in two steps (Figure 5). In the first step, a system of four 3:1 arbiters decides the non-local candidate flits for the four non-local output ports. Additionally, a 4:1 arbiter decides the local candidate flit destined for one of the four non-local output ports. In the second step, a system of four 2:1 arbiters decides the winners for the four non-local output ports. Additionally, a 4:1 arbiter decides the winner for the local output port. Thus, ODVC manages to reduce the cardinality of the required arbiters significantly, resulting in reduced area and a shorter critical path. Note that a 2-stage arbitration process is required because of the different designs of local and non-local ports which, in turn, was the consequence of the trace buffer width constraint.

We can extend the design to any number of ports. Let the total number of ports be p and the trace buffer width be w flits. The trace buffer can be used as a backup storage for only w ports, which will be referred to as the TB-backed ports. The other $p - w$ ports that are not backed by the TB will be referred to as the unbacked ports. There are $p - 1$ VCs, each of size k flits, per TB-backed port and v VCs, each of size k' flits, per unbacked port, where $k < k'$. Moreover, the first stage of switch arbitration requires w arbiters of cardinality $w - 1 : 1$ and $p - w$ arbiters of cardinality $v : 1$. The second stage of switch arbitration requires w arbiters of cardinality $(p - w + 1) : 1$ and $p - w$ arbiters of cardinality $p - 1 : 1$.

LAR is performed in parallel with SA. VCs are assigned only for the SA winners. The VC assignment logic is very simple as the VCs are already mapped to the output ports. This mapping is possible because of the availability of the trace buffer that can be shared by all packets alike. The alternative design without a shared buffer would mean having four output port-mapped VCs of large widths. However, such a design fails to efficiently handle commonly occurring scenarios where heavy traffic arriving at an input port is headed towards a single direction, leading to under-utilization of the other VCs at the input port. Priority-based schemes [2] have been proposed, but these require complex SA mechanisms.

V. Experimental Results

A. Experimental Setup

In this section, we evaluate our proposed design in terms of overall network performance, area overhead, and power dissipation. We use a flit-level cycle-accurate NoC simulator *BookSim* [41] to conduct a detailed evaluation of different designs under discussion. We use XY dimension-ordered routing, although our proposal is independent of the specific routing algorithm and the topology employed in the system. The configuration parameters for the proposed and the baseline network are listed in Table I and Table II, respectively.

We evaluated our design on both synthetic traffic patterns and *Synfull* traffic models [42]; these two cover a wide range of network utilization scenarios and exercise both cache coherence traffic and memory traffic. The several synthetic traffic patterns include: (1) *uniform*, where each node sends an equal amount of traffic to each destination and (2) Permutation traffic pattern (*transpose, tornado, neighbor*), where each node selects a fixed destination based on the permutations. The packet size is varied from 5 flits (a header, a tail, and 3 body flits) to 100 flits, where each flit size is 32 bits [29]. BookSim is warmed up for 10,000 clock cycles; then the performance statistics are computed for the next 100,000 cycles at different injection rates starting at 0.04 flit/node/cycle and incremented by 0.04 flit/node/cycle until the throughput is reached. As part of these experiments, we integrated Booksim with Synfull [42] to model the inherent communication pattern of a cache coherent system for multi-threaded applications from SPLASH2 and PARSEC benchmark suite. These workloads consist of single-flit control packets and multi-flit data packets.

We also implemented our design by extending an open source NoC router RTL model [43] and synthesized it using Cadence Encounter RTL compiler with a 90nm standard cell library, to understand the area, timing, and power implications of our proposed design. The activity factors for dynamic power evaluation are obtained from Booksim simulations, and then fed to Cadence Encounter.

B. Synthetic workloads

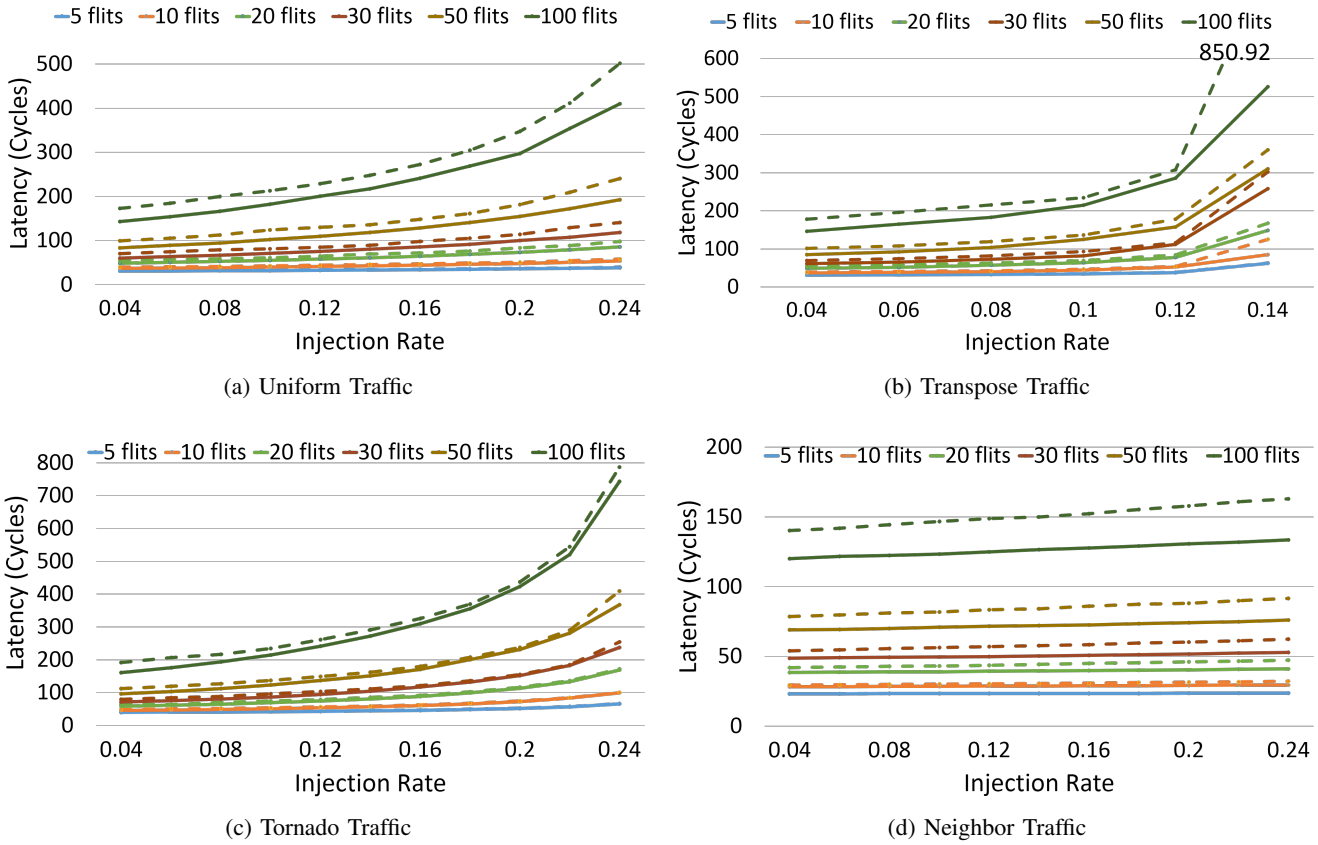
1) Impact of AugVC Router on latency: Figures 6a–6d show the variation in average packet latency with increasing injection rate for the four different traffic workloads – uniform, transpose, tornado, and neighbor, and compare it with Baseline5. The packet latency is computed as the time elapsed between the sending of the first flit by source and the receipt of the last flit by destination. The injection rate is defined as the number of flits per cycle per node. The figure shows that our scheme exhibits lower average packet latency at all injection rates and packet sizes. We observe that the latency reduction is more for larger packet sizes, which is expected, as these require deeper VCs. The latency reduction over the Baseline5 NoC for packet size 100 and injection rate 0.24 is 18.16% for uniform traffic, 6.1% for tornado traffic, 18.4% for neighbor traffic, and 38.25% for transpose traffic (injection rate = 0.14).

Figure 7 shows the average packet latency for different trace buffer sizes by varying the number of lines from 6 to 32 under uniform traffic (packet size 100). The two dashed lines

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/TCAD.2019.2907909, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

Parameter	AugVC	ODVC
Topology (Mesh)	4×4, 8×8, 16×16	4×4, 8×8, 16×16
Routing algorithm	X-Y dimension order	X-Y dimension order
#Virtual channel per port	8	4
#Flits per VC	3 (non-local ports), 8 (local port)	4 (non-local ports), 8 (local port)
Flit size (bits)	32	32
Trace buffer width	128 bits (4 flits)	128 bits (4 flits)
#Lines in trace buffer	32	32
Write holding buffer	4 flits	4flits
Read holding buffer	4 flits	7 flits
Request queue size	12 entries	12 entries

TABLE I: Proposed network configuration parameters



(a) Uniform Traffic

(b) Transpose Traffic

(c) Tornado Traffic

(d) Neighbor Traffic

Fig. 6: Average packet latency under four synthetic traffics for different packet sizes (5, 10, 20, 30, 50, 100 flits). Dashed lines and solid lines represent Baseline5 and AugVC, respectively.

Topology (Mesh)	4×4, 8×8, 16×16
#Virtual Channels (per port)	8
#Flits per VC	5 in Baseline5 8 in Baseline8
Flit size	32 bits
Packet size	5 flits to 100 flits
Routing algorithm	X-Y dimension order

TABLE II: Baseline configuration parameters

represent the baseline architecture Baseline5 and Baseline8. We observe that AugVC performs better than Baseline5 at almost all the trace buffer sizes and injection rates, even those as small as 8 lines. A trace buffer with 6 lines performs well for smaller injection rates. This is significant considering that the DFD structure would have gone unutilized without this reuse. Although AugVC (with a trace buffer of 32 lines) and Baseline8 achieve similar packet latencies, the area requirement of our scheme is 7% lower (Section V-D). Similar trends were observed for other synthetic traffics and the results for Baseline8 were omitted from Figure 6 for clarity.

2) Impact of ODVC Router on latency: ODVC reduces the cycle time by 30%, and therefore, allows the router to operate at a higher frequency. This design also reduces the number of pipeline stages by 1, thereby reducing the overall latency

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/TCAD.2019.2907909, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

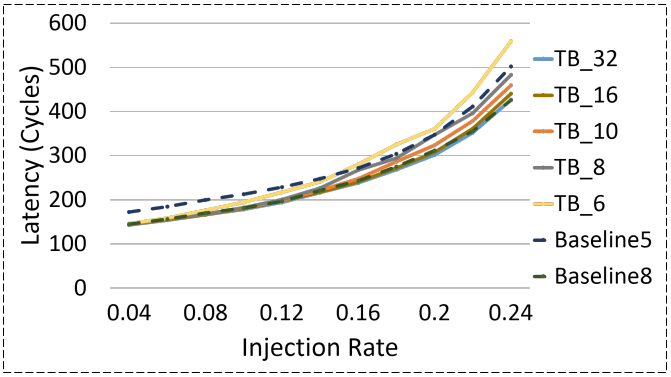


Fig. 7: Impact of different trace buffer sizes on average packet latency under uniform traffic (packet size = 100 flits)

as well. In the ODVC design, every flit needs to carry the destination address (flit-level routing), unlike Baseline5 and AugVC, where only the header flit carries the destination address. To make a fair comparison, we have increased the number of flits in a packet in the ODVC design. Since the destination address is 6 bits wide for a 8×8 2-D mesh network, we have added 1 flit for every 5 flits. Thereby, the packets with sizes {5, 10, 20, 30, 50 and 100} are changed to {6, 12, 24, 36, 60 and 120} in the ODVC design.

Figure 8a–8d show the variation of the normalized average packet latency of Baseline5, AugVC, and ODVC design, with increasing injection rate for the four synthetic traffic workloads. The latencies are normalized with respect to Baseline5. The figure shows that ODVC exhibits minimum average latency for uniform, neighbor, and transpose traffic for different packet sizes; it saturates (at a latency threshold of 1500 cycles) early for higher injection rates for tornado traffic because of the highly imbalanced nature of the traffic. ODVC maps one VC to each possible outgoing direction. The Tornado traffic pattern uses only one private VC of the port to forward most of the packets, resulting in packets accumulating in the trace buffer at higher injection rates. At lower injection rates, packets are present in a mixed format, leading to better overall progress. This is also the reason for better improvements with smaller packet sizes as compared to larger packet sizes. A larger packet covers most of the private VC space and other packets are accommodated in the trace buffer, which increases the latency of the packet present in the trace buffer. We also observe some erratic behavior at the extreme point corresponding to saturation (saturation points for different traffics are reported in Table III).

In Figure 9, the ODVC design is compared with the VOQ technique [32], which also uses a fixed 1-1 mapping of VCs to output ports. The figure shows the variation of the normalized average packet latency of VOQ for five different VC sizes (4, 5, 6, 7, and 8 flits per VC), with increasing injection rates under four synthetic traffic workloads. The VC size in ODVC is fixed at 4 flits. ODVC exhibits the lowest average packet latency for all the traffic patterns, primarily because of the backup storage shared between all the VCs. Also, the VOQ design saturates early for smaller VC sizes. In comparison, ODVC gives better latency reduction and throughput with smaller VCs.

Traffic	Baseline5	Baseline8	AugVC	ODVC
uniform	0.28	0.28	0.29	0.25
transpose	0.14	0.14	0.14	0.14
tornado	0.25	0.25	0.25	0.14
neighbor	0.77	0.77	0.81	0.51

TABLE III: Maximum injection rate of source processor with latency threshold and packet size of 1500 cycles and 100 flits, respectively.

We also compared ODVC with the proposal by Xu et al. [2], and observed similar latencies in terms of cycles. However, our approach reduces the clock period by 22% because of the reduction in the arbiter's cardinality.

Table III shows the maximum injection rate supported by different designs over different synthetic traffics while considering the latency threshold of 1500 cycles. We observe a slightly better saturation throughput for AugVC design compared to Baseline5 and Baseline8 for uniform and neighbor traffic, primarily because of a reduction in the average packet latency in AugVC. For transpose traffic, all the routers of the same row send flits to the same output direction and therefore, are limited by the output port of the last router on that row; all the designs exhibited the same saturation throughput of 0.14. We observe higher latencies with increasing injection rate for ODVC design, primarily because of packet accumulation in the trace buffer. AugVC design improves performance to a greater extent than the ODVC design. However, it has a higher associated area overhead, leading to an area-performance trade-off between the two techniques.

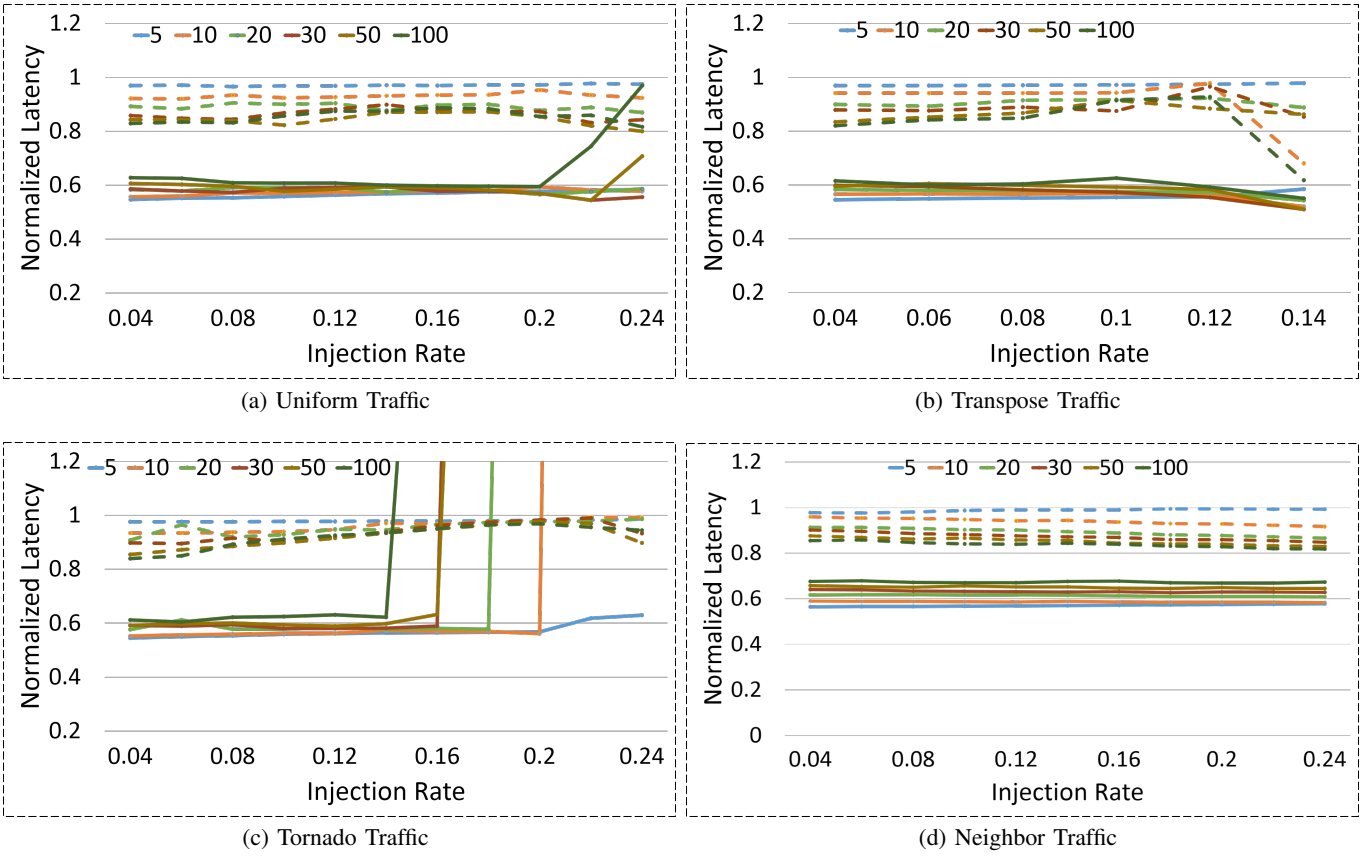
C. PARSEC and SPLASH2 Benchmarks

We study the performance impact under 16 PARSEC and SPLASH2 benchmarks for three NoC sizes 4×4, 8×8, and 16×16 (Figure 13). We do not observe much reduction in latency by increasing the VC width from 5 to 8 flits because of the smaller packet sizes. Therefore, AugVC also does not provide a significant latency reduction. AugVC was evaluated with different trace buffer sizes (6 to 32 lines), yielding similar results. However, ODVC reduces the latency significantly for all the benchmarks because of the aforementioned reduction in both the cycle time and the number of pipeline stages.

We also observed the impact of varying trace buffer access latencies in our experiments as it may not be always possible to obtain a single-cycle access to the trace buffer. The results in Figure 11 exhibit a significant reduction in packet latency even with higher trace buffer access latencies; this can be further improved by increasing the size of the Read HB (Section IV-A). Read HB sizes of 7 and 10 flits with latencies of 2 and 3 cycles, respectively, provide similar results to the single cycle latency designs.

Figure 12 shows the maximum, minimum, and variance values of packet latency for different benchmarks. The results are normalised to Baseline5. The minimum latency is the same in all of the 3 designs because the number of pipeline stages are the same. For maximum and variance in latencies,

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/TCAD.2019.2907909, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems



(a) Uniform Traffic

(b) Transpose Traffic

(c) Tornado Traffic

(d) Neighbor Traffic

Fig. 8: Impact of AugVC (Dashed lines) and ODVC (solid lines) design on average packet latency (normalized to Baseline5)

Design	Area	Static Power		
		Buffer (VCs)	TB, TBCtlr	Others
Baseline5	1	1	1	1
Baseline8	1.152	1.38	1	1.05
AugVC	1.082	0.72	1.90	0.99
ODVC	0.676	0.46	1.67	0.36

TABLE IV: Synthesis results normalized to Baseline5

we observed mixed results. AugVC manages to reduce the maximum latency and variance in latencies for *cholesky* and *swaptions* benchmarks; however, high values are observed for *raytrace*. This is primarily determined by how the flits are stored and accessed from the trace buffer.

D. Synthesis Results

The area and static power overheads for different designs are summarized in Table IV. The results are normalized to Baseline5. We observe an area overhead of 8.2% for AugVC compared to Baseline5, mainly because of the bookkeeping structure such as next address table, read HB, and write HB, required for the trace buffer reuse. There is an area saving of 7% for AugVC compared to Baseline8. The ODVC scheme reduced the critical path by 30%, area by 32.4% and total power by 17.8% compared to Baseline5, primarily because of the reduction in VC count and arbiter cardinality. The static power of the trace buffer controller is smaller for ODVC compared to AugVC, because of a reduction in VC count which further reduces the number of bookkeeping structures such as head pointers and tail pointers.

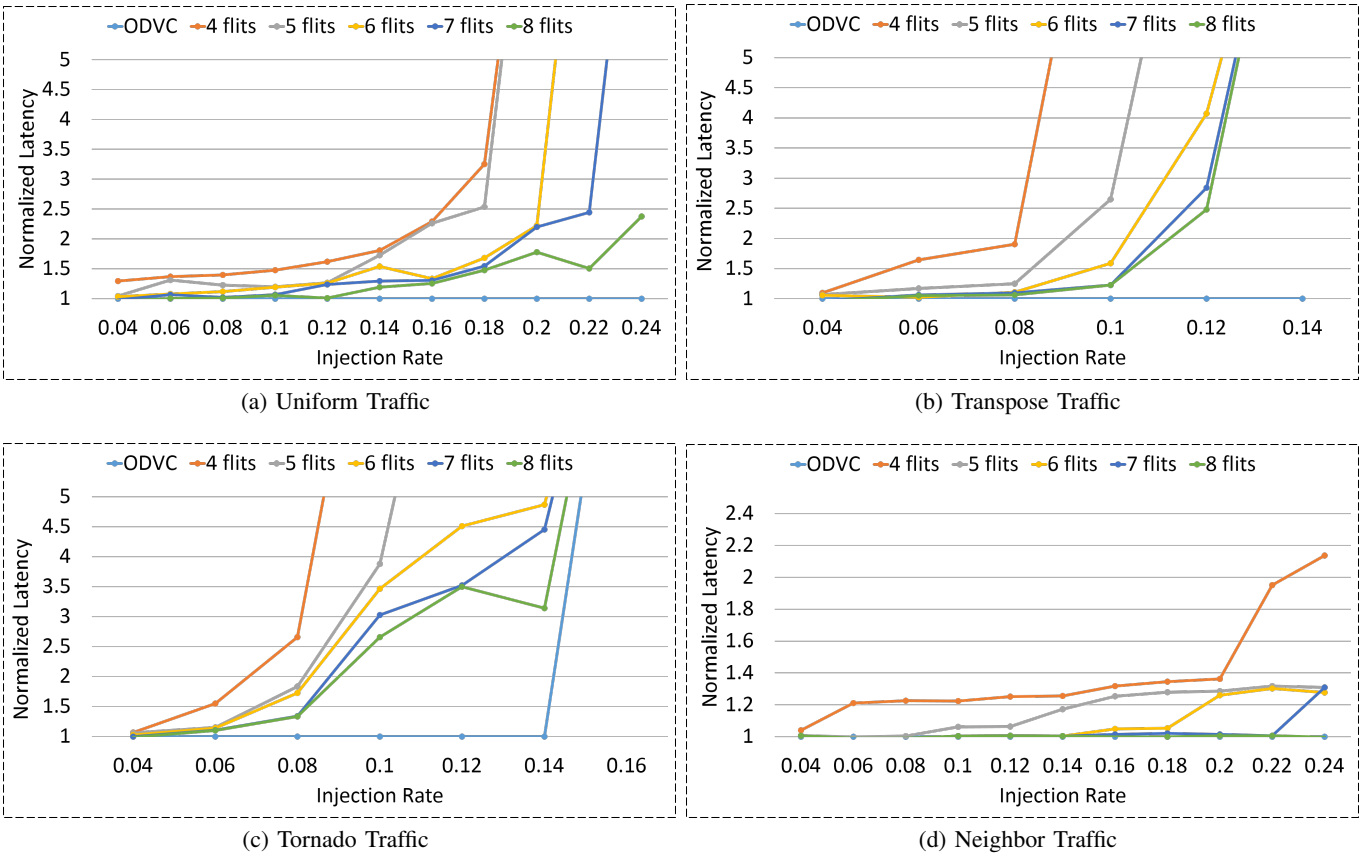
The dynamic power overheads for four different synthetic traffics are shown in Figure 10. The results are normalized to the total power of Baseline5. There is no dynamic power observed for the trace buffer and its controller for Baseline5 and Baseline8 as the validation structures are power gated in-field in conventional router architectures. The additional dynamic power in AugVC compared to Baseline5 is mostly due to the accesses to the trace buffer when there is no space available in private VCs. We observed less switching activity at the trace buffer with the use of write holding buffer as all the write accesses are not directed to the trace buffer.

The changes proposed in Section IV-A impact only the writing in the VCs. Even though this lengthens the path delay, this does not impact the clock period because it remains within the slack provided by the slower arbiters. Therefore, we do not observe any cycle time overhead for AugVC, as no changes are made to VC allocation and switch allocation stages.

VI. Conclusion and Future Work

As network-on-chip performance is directly related to router buffer configuration, the buffer architecture plays an important role in designing low cost, high performance, and energy efficient on-chip networks. In this paper, we proposed and evaluated an approach to reuse trace buffers to augment router

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/TCAD.2019.2907909, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems



(a) Uniform Traffic

(b) Transpose Traffic

(c) Tornado Traffic

(d) Neighbor Traffic

Fig. 9: Average packet latency (normalized to ODVC) variation of VOQ scheme under four synthetic traffics for different VC sizes (4, 5, 6, 7 and 8 flits per VC and packet size = 100 flits)

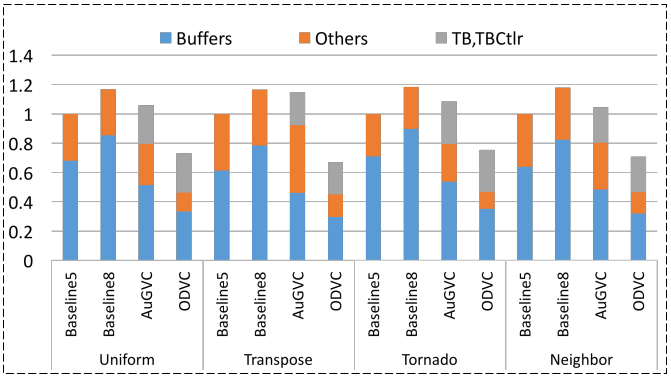


Fig. 10: Normalized dynamic power (normalized to total power of Baseline5) under four synthetic traffics (injection rates = 0.24 for uniform, tornado and neighbor; 0.14 for transpose, packet size = 100 flits)).

buffers, which improves the overall NoC performance. The trace buffer is reused as a backup storage for the flits that cannot be accommodated in the private VCs. Reusing design-for-debug structures enables us to allocate more space for on-chip debug hardware, resulting in better validation. The AugVC simulation results using a cycle accurate simulator indicate latency reductions by up to 38.25%. We also proposed a scheme ODVC that uses a modified VC assignment where the VCs are assigned on the basis of the designated output port of the packet, resulting in significant reduction of the clock period and area of the router by 30% and 32.4% respectively. Our proposed architecture may lead to effects on the cache coherence handling mechanism that require further investigation. This is an area of future research. We also plan to investigate ways to decrease the bookkeeping overhead for the trace buffer reuse.

VII. Acknowledgment

This research was partially supported by research grant 2014-TJ-2528 from Freescale Semiconductor and Semiconductor Research Corporation.

References

[1] B. Vermeulen and K. Goossens, "A network-on-chip monitoring infrastructure for communication-centric debug of embedded multi-processor socs," in *VLSI Design, Automation and Test, 2009. VLSI-DAT'09. International Symposium on*. IEEE, 2009, pp. 183–186.

[2] Y. Xu, B. Zhao, Y. Zhang, and J. Yang, "Simple virtual channel allocation for high throughput and high frequency on-chip routers," in *High Performance Computer Architecture (HPCA), 2010 IEEE 16th International Symposium on*. IEEE, 2010, pp. 1–11.

[3] W. J. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed systems (TPDS)*, vol. 3, no. 2, pp. 194–205, 1992.

[4] 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 (ISVLSI). Proceedings. IEEE Computer Society Annual Symposium on*. IEEE, 2002, pp. 117–124.

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/TCAD.2019.2907909, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

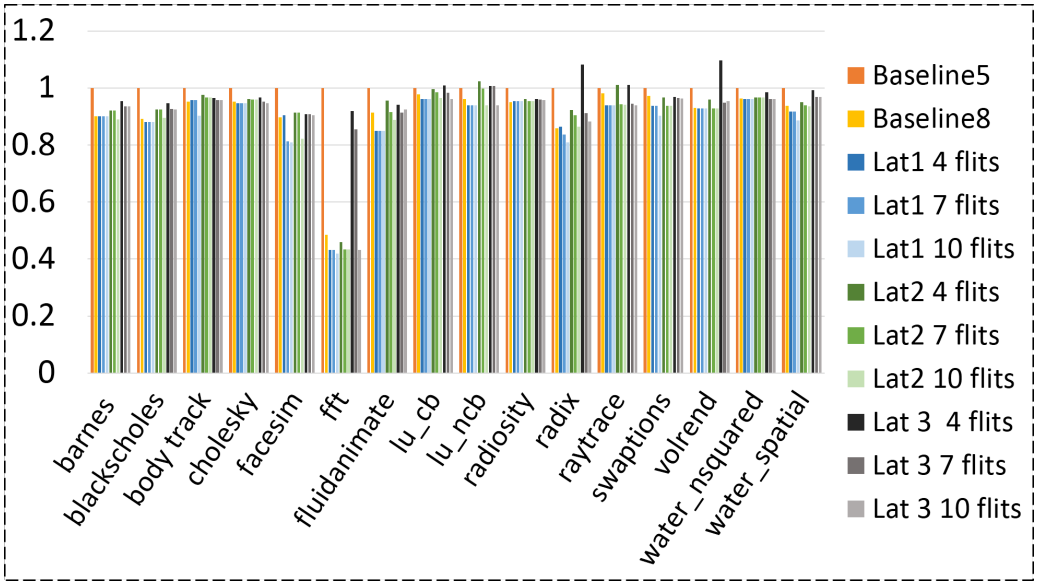


Fig. 11: Impact of varying trace buffer latencies and read holding buffer sizes.

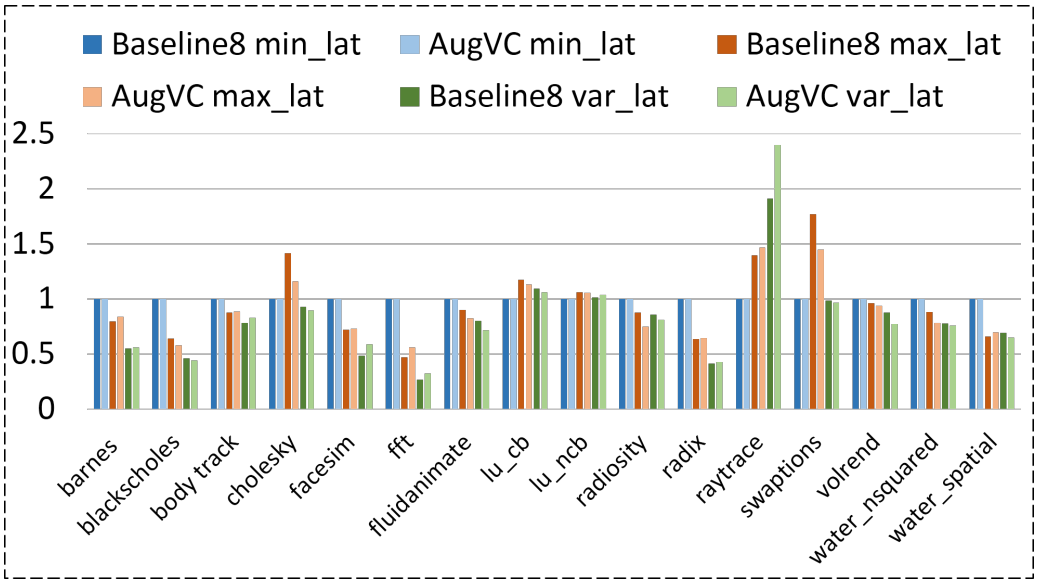


Fig. 12: Minimum, maximum and variance values of packet latencies for PARSEC abd SPLASH2 benchmarks .

[5] Y. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, "A 5-GHz mesh interconnect for teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, 2007.

[6] J. Hu and R. Marculescu, "Application-specific buffer space allocation for networks-on-chip router design," in *Proceedings of the 2004 IEEE/ACM International conference on Computer-aided design (ICCAD)*. IEEE Computer Society, 2004, pp. 354–361.

[7] H. Wang, L.-S. Peh, and S. Malik, "Power-driven design of router microarchitectures in on-chip networks," in *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2003, p. 105.

[8] T. T. Ye, L. Benini, and G. De Micheli, "Analysis of power consumption on switch fabrics in network routers," in *Design Automation Conference, 2002. Proceedings. 39th (DAC)*. IEEE, 2002, pp. 524–529.

[9] S. Tang and Q. Xu, "A multi-core debug platform for NoC-based systems," in *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE'07*. IEEE, 2007, pp. 1–6.

[10] M. Galles, "Spider: A high-speed network interconnect," *IEEE Micro*, vol. 17, no. 1, pp. 34–39, 1997.

[11] K. Rahmani, S. Ray, and P. Mishra, "Postsilicon trace signal selection using machine learning techniques," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 2, pp. 570–580, 2017.

[12] P. Taatizadeh and N. Nicolici, "Emulation infrastructure for the evaluation of hardware assertions for post-silicon validation," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 6, pp. 1866–1880, 2017.

[13] K. Goossens, B. Vermeulen, and A. B. Nejad, "A high-level debug environment for communication-centric debug," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*. European Design and Automation Association, 2009, pp. 202–207.

[14] C. Ciordas, K. Goossens, T. Basten, A. Radulescu, and A. Boon, "Transaction monitoring in networks on chip: The on-chip run-time perspective," in *Industrial Embedded Systems, 2006. IES'06. International Symposium on*. IEEE, 2006, pp. 1–10.

[15] C. Ciordas, T. Basten, A. Radulescu, K. Goossens, and J. Meerbergen, "An event-based network-on-chip monitoring service," in *High-Level Design Validation and Test Workshop (HLDVT), 2004. Ninth IEEE International*. IEEE, 2004, pp. 149–154.

[16] A. M. Gharehbaghi and M. Fujita, "Transaction-based debugging of system-on-chips with patterns," in *Computer Design, 2009. ICCD 2009. IEEE International Conference on*. IEEE, 2009, pp. 186–192.

[17] M. Dehbashi and G. Fey, "Transaction-based online debug for NoC-

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/TCAD.2019.2907909, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems

based multiprocessor SoCs," *Microprocessors and Microsystems*, vol. 39, no. 3, pp. 157–166, 2015.
[18] A. Ghofrani, R. Parikh, S. Shamshiri, A. DeOrio, K.-T. Cheng, and V. Bertacco, "Comprehensive online defect diagnosis in on-chip networks," in *VTS*. IEEE, 2012, pp. 44–49.
[19] R. Abdel-Khalek and V. Bertacco, "Functional post-silicon diagnosis and debug for networks-on-chip," in *ICCAD*. ACM, 2012, pp. 557–563.
[20] R. Abdel-Khalek and V. Bertacco, "Correct runtime operation for NoCs through adaptive-region protection," in *Design, Automation & Test in Europe (DATE), 2016*. IEEE, 2016, pp. 1189–1194.
[21] R. Abdel-Khalek and V. Bertacco, "Diamond: Distributed alteration of messages for on-chip network debug," in *NoCS*. IEEE, 2014, pp. 127–134.
[22] M. Rezazad and H. Sarbazi-Azad, "The effect of virtual channel organization on the performance of interconnection networks," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*. IEEE, 2005, pp. 8–pp.
[23] T.-C. Huang, U. Y. Ogras, and R. Marculescu, "Virtual channels planning for networks-on-chip," in *Quality Electronic Design, 2007. ISQED'07. 8th International Symposium on*. IEEE, 2007, pp. 879–884.
[24] H. Kim, C. Kim, M. Kim, K. Won, and J. Kim, "Extending bufferless on-chip networks to high-throughput workloads," in *Networks-on-Chip (NoCS), 2014 Eighth IEEE/ACM International Symposium on*. IEEE, 2014, pp. 9–16.
[25] X. Xiang, W. Shi, S. Ghose, L. Peng, O. Mutlu, and N.-F. Tzeng, "Carpool: a bufferless on-chip network supporting adaptive multicast and hotspot alleviation," in *Proceedings of the International Conference on Supercomputing*. ACM, 2017, p. 19.
[26] X.-Y. Xiang and N.-F. Tzeng, "Deflection containment for bufferless network-on-chips," in *Parallel and Distributed Processing Symposium, 2016 IEEE International (IPDPS)*. IEEE, 2016, pp. 113–122.
[27] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das, "ViChaR: A dynamic virtual channel regulator for network-on-chip routers," in *Microarchitecture, 2006. MICRO-39. 39th Annual IEEE/ACM International Symposium on*. IEEE, 2006, pp. 333–346.
[28] H. Zhang, K. Wang, Y. Dai, and L. Liu, "A multi-VC dynamically shared buffer with prefetch for network on chip," in *Networking, Architecture and Storage (NAS), 2012 IEEE 7th International Conference on*. IEEE, 2012, pp. 320–327.
[29] A. T. Tran and B. M. Baas, "Achieving high-performance on-chip networks with shared-buffer routers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 6, pp. 1391–1403, 2014.
[30] R. Grindley, T. Abdelrahman, S. Brown, S. Caranci, D. DeVries, B. Gamsa, A. Grbic, M. Gusat, R. Ho, O. Krieger *et al.*, "The numachine multiprocessor," in *Parallel Processing, 2000. Proceedings. 2000 International Conference on*. IEEE, 2000, pp. 487–496.
[31] F. Alazemi, A. AziziMazreah, B. Bose, and L. Chen, "Routerless network-on-chip," in *High Performance Computer Architecture (HPCA), 2018 IEEE International Symposium on*. IEEE, 2018, pp. 492–503.
[32] M. J. Karol, K. Y. Eng, and H. Obara, "Improving the performance of input-queued atm packet switches," in *INFOCOM'92. Eleventh Annual Joint Conference of the IEEE Computer and Communications Societies, IEEE*. IEEE, 1992, pp. 110–115.
[33] S. A. R. Jafri, H. B. Sohail, M. Thottethodi, and T. Vijaykumar, "apslip: A high-performance adaptive-effort pipelined switch allocator," 2013.
[34] C. Li, D. Dong, X. Liao, J. Wu, and F. Lei, "Rob-router: Low latency network-on-chip router microarchitecture using reorder buffer," in *High-Performance Interconnects (HOTI), 2016 IEEE 24th Annual Symposium on*. IEEE, 2016, pp. 68–75.
[35] S. Han, K. Kang, and J. Ryu, "Determination of delay bound over multi-hop real-time switches with virtual output queuing," in *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on*. IEEE, 2014, pp. 892–898.
[36] Y.-H. Kao, N. Alfaraj, M. Yang, and H. J. Chao, "Design of high-radix clos network-on-chip," in *Proceedings of the 2010 Fourth ACM/IEEE International Symposium on Networks-on-Chip*. IEEE Computer Society, 2010, pp. 181–188.
[37] A. Basak, S. Bhunia, and S. Ray, "Exploiting design-for-debug for flexible SoC security architecture," in *Proceedings of the 53rd Annual Design Automation Conference (DAC)*. ACM, 2016, p. 167.
[38] N. Jindal, P. R. Panda, and S. R. Sarangi, "Reusing trace buffers to enhance cache performance," in *Proceedings of the Conference on Design, Automation & Test in Europe (DATE)*. European Design and Automation Association, 2017, pp. 572–577.
[39] N. Jindal, P. R. Panda, and S. R. Sarangi, "Reusing trace buffers as victim caches," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2018.
[40] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *ACM SIGARCH Computer Architecture News*, vol. 37, no. 3. ACM, 2009, pp. 196–207.
[41] N. Jiang, J. Balfour, D. U. Becker, B. Towles, W. J. Dally, G. Michelogiannakis, and J. Kim, "A detailed and flexible cycle-accurate network-on-chip simulator," in *Performance Analysis of Systems and Software (ISPASS), International Symposium on*. IEEE, 2013, pp. 86–96.
[42] M. Badr and N. E. Jerger, "Synfull: Synthetic traffic models capturing a full range of cache coherent behaviour," in *in ISCA*. Citeseer, 2014.
[43] D. U. Becker, "Efficient microarchitecture for network-on-chip routers," Ph.D. dissertation, Stanford University, 2012.

Neetu Jindal is a research scholar in the Department of Computer Science and Engineering at Indian Institute of Technology, Delhi. She received her Bachelors' degree in Computer Science and Engineering from Kurukshetra University. Her research interests include post-silicon validation methodologies, architectural design-space exploration and machine learning applications to computer architecture optimizations.

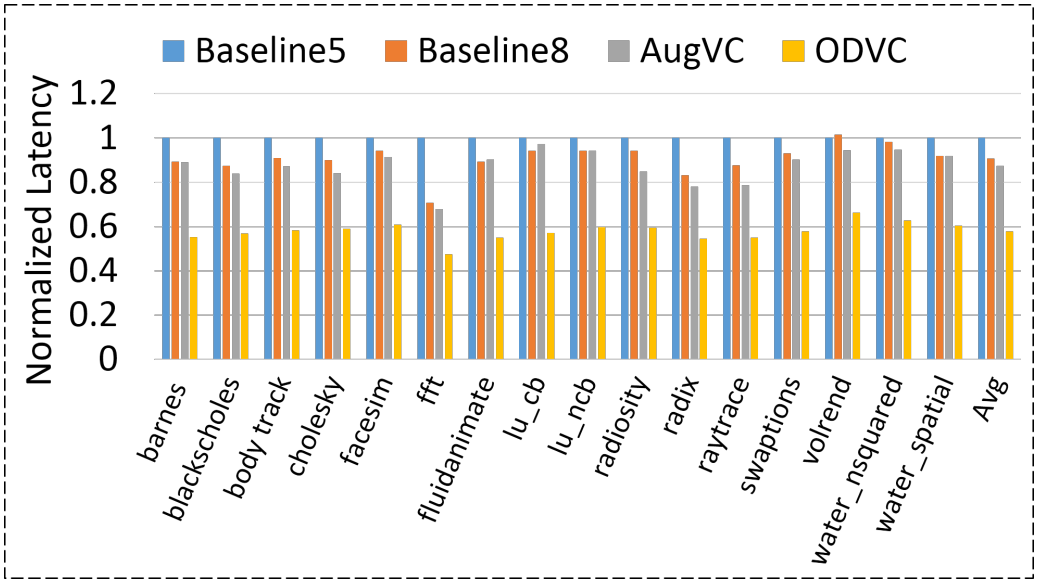
Shubhani Gupta received the B.Tech. degree in electronics and communication engineering from NIT Kurukshetra. She has more than 7 years of design experience in companies including Bharat Electronics Ltd. and Samsung Research Institute. She is currently a Research Scholar with the School of IT at IIT Delhi. Her research interests include high level synthesis and system-on-chip architectures.

Divya Praneetha Ravipati is pursuing her PhD at Indian Institute of Technology Delhi. Prior to joining as Research Scholar, she worked as a System Design Engineer in Kasura Technologies Private Limited for 2 years and as an Assistant Professor in Vignan University for an year. Her research interests include embedded systems and digital design.

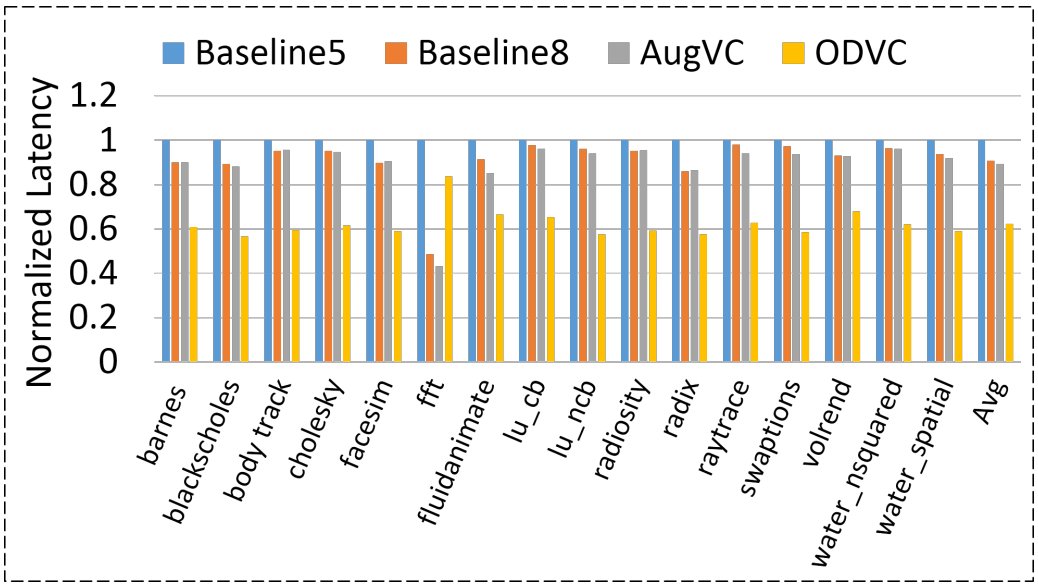
Preeti Ranjan Panda received his B. Tech. degree in Computer Science and Engineering from the IIT Madras and his M. S. and Ph.D. degrees from the University of California at Irvine. He is currently a Professor in the Department of Computer Science and Engineering at IIT Delhi. He has previously worked at Texas Instruments and Synopsys, and has been a visiting scholar at Stanford University. His research interests include Embedded Systems and Design Automation. He is the author of two books: *Memory issues in Embedded Systems-on-chip: Optimizations and Exploration* and *Power-efficient System Design*. He is a recipient of an IBM Faculty Award and a Department of Science and Technology Young Scientist Award. Prof. Panda has served on the editorial boards of IEEE TCAD, ACM TODAES, IEEE ESL, and IJPP, and as Technical Program co-Chair of CODES+ISSS and VLSI Design. He has also served on the technical program committees and chaired sessions at several conferences including DAC, ICCAD, DATE, CODES+ISSS, ISLPED, and EMSOFT.

Smruti R. Sarangi received his B.Tech degree in Computer Science from IIT Kharagpur and the MS and Ph.D degrees from the University of Illinois, Urbana-Champaign. Currently, he is an Usha Hasteer Chair Professor in the Computer Science and Engineering Department at IIT Delhi. He holds a joint appointment with the Department of Electrical Engineering and the School of Information Technology. His research interests include processor reliability, architectural support for operating systems, and processors for the internet of things. He has published extensively in peer reviewed conferences and journals, holds 5 US patents, and has filed 3 Indian patents. He is the author of the popular undergraduate textbook on computer architecture titled, "Computer Organisation and Architecture", published by McGrawHill. He is a member of the IEEE and ACM.

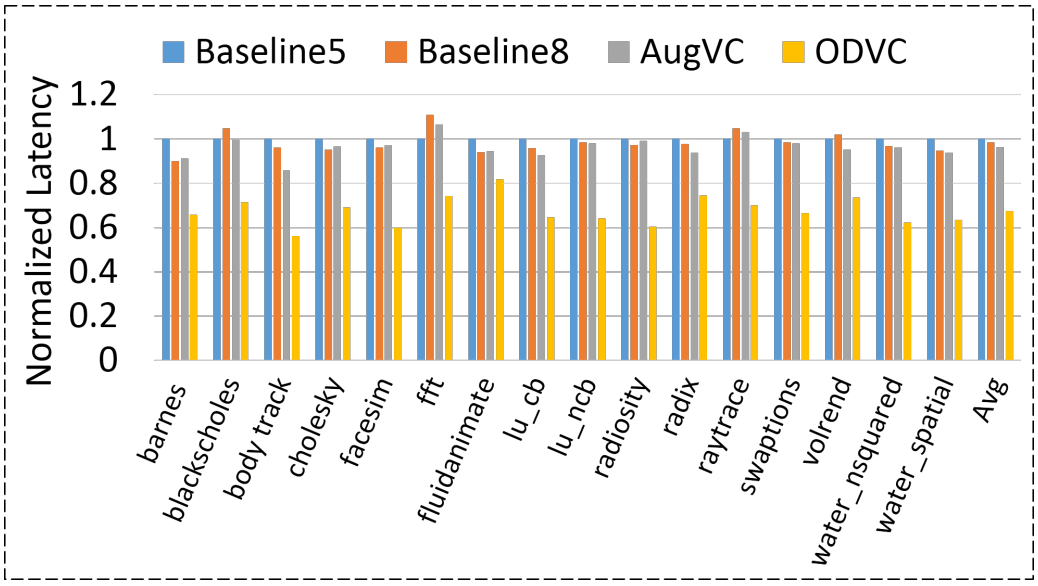
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2019.2907909, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems



(a) 4×4 network



(b) 8×8 network



(c) 16×16 network

Fig. 13. Normalized average packet latency (normalized to Baseline5) for PARSEC and SPLASH2 benchmarks for three network sizes.