Simple Virtual Channel Allocation for High Throughput and High Frequency On-Chip Routers *

Yi Xu†, Bo Zhao†, Youtao Zhang‡, Jun Yang†
† Dept. of Electrical and Computer Engineering
‡ Dept. of Computer Science
University of Pittsburgh, Pittsburgh, PA 15621
†{yix13, boz6, juy9}@pitt.edu, ‡{zhangyt}@cs.pitt.edu

Abstract

Technology scaling has led to the integration of many cores into a single chip. As a result, on-chip interconnection networks start to play a more and more important role in determining the performance and power of the entire chip. Packet-switched network-on-chip (NoC) has provided a scalable solution to the communications for tiled multi-core processors. However the virtual-channel (VC) buffers in the NoC consume significant dynamic and leakage power of the system. To improve the energy efficiency of the router design, it is advantageous to use small buffer sizes while still maintaining throughput of the network.

This paper proposes two new virtual channel allocation (VA) mechanisms, termed Fixed VC Assignment with Dynamic VC Allocation (FVADA) and Adjustable VC Assignment with Dynamic VC Allocation (AVADA). The idea is that VCs are assigned based on the designated output port of a packet to reduce the Head-of-Line (HoL) blocking. Also, the number of VCs allocated for each output port can be adjusted dynamically. Unlike previous buffer-pool based designs, we only use a small number of VCs to keep the arbitration latency low. Simulation results show that FVADA and AVADA can improve the network throughput by 41% on average, compared to a baseline design with the same buffer size. AVADA can still outperform the baseline even when our buffer size is halved. Moreover, we are able to achieve comparable or better throughput than a previous dynamic VC allocator while reducing its critical path delay by 60%. Our results prove that the proposed VA mechanisms are suitable for low-power, high-throughput, and high-frequency on-chip network designs.

1. Introduction

Technology scaling has enabled the integration of billions of transistors on a single chip. Chip multiprocessors (CMP) have emerged as an effective design for utilizing on-chip transistors to continue the growth of chip performance with integration density [9, 24]. With the proliferation of CMPs, on-chip interconnection networks start to play a more and more important role in determining the performance and power of the entire chip [14]. Among various network on-chip (NoC) designs, packet-switched network is recognized as a scalable and flexible solution for communication in 10's∼100's-core CMPs.

As with designing a general microprocessor, one of the major challenges in packet-switched NoC is how to limit its power consumption. Studies have shown that on-chip network can consume 30%∼40% of the chip power [10, 32], which is a major limitation to future NoC development. Hence, many techniques such as packet bypassing [16, 17], router concentration [1], crossbar and buffer optimization [32], bufferless routing [19] etc. have been proposed to save dynamic power or make the NoC more energy efficient. On the other hand, there has been an incessant effort in pursuing aggressive high-throughput and high-frequency router designs. Examples include the 5GHz router design in Teraflops chip [10, 31] and the 4 GHz 256Gbits/s per node router design [30]. Extremely high frequency NoCs may require long pipeline stages [10, 30] in the router, which may be of a concern under low traffic load where packet latency is more important. High-frequency routers are typically of high power too. Hence, to achieve high-frequency, low-latency and high-throughput under power constraint, one must adopt simple yet still highly efficient router design.

One of the most power hungry components in a router is its buffer [15, 32]. A buffer is used as a set of virtual channels (VCs) [3, 4] to store incoming or outgoing packets. How buffers are used impacts both network throughput and power. They are mainly determined by VC number, VC sizes and how they are allocated. The size of a VC queue can be either static or dynamic. A static queue always has a fixed size while a dynamic queue can grow and shrink. Although a dynamic queue size helps to improve the buffer utilization efficiency such that more flits can be stored in the buffer, it does however increase the complexity of the related control logic. For example, the DAMQ design [2, 28] used hardware to implement a linked list to

*This work is supported in part by NSF grants 0747242, 0641177, 0720595, and 0734339.

manage dynamic queue size, resulting large delay in every flit arrival/departure. Later designs avoided using linked list, but still incur high cost in control logic. For example, The ViChaR design [23] can support a VC size anywhere from one flit to a whole packet, generating a maximum VC count as the size of the whole buffer in terms of flits (VC count = buffer size / VC size). This requires f:1 arbiters, where f is the buffer size, in both VC allocation (VA) and switch allocation (SA) stage. Such cardinality of an arbiter may introduce latency bottleneck in the critical path of a router, which can limit the frequency of the network. Hence, a static VC queue design is preferred in a high-frequency router design unless high-complexity logic is employed for short cycle time.

From VC allocation point of view, a VC can be statically assigned to traffic in one direction, or dynamically assigned to any incoming/outgoing traffic on-demand. Static allocation, though not used as much nowadays, has its advantage in that traffic to/from all directions have equal opportunities to compete for the switch. Dynamic allocation, on the other hand, can better tolerate burst traffic from one direction, or uneven traffic distribution among all directions. As we can see, both allocation schemes have their own advantages. A design that integrates features in both schemes would be most beneficial to a high-throughput router.

In this paper, we propose simple VC allocation schemes to achieve high-frequency and high-throughput router design. We use a small maximum number of VCs with fixed VC size to keep the arbiter size small and fast. To integrate the advantages of both static allocation and dynamic allocation schemes, we designate each VC for one output port (direction) but allow such assignment to vary according to traffic fluctuation. Since our total VC count is small, we give body and tail flits higher priorities than a header flit during VA to recycle a VC quickly. Our changes required to a baseline router are extremely simple, compared to previous designs which improve network performance at the cost of major changes in the router. Simulation results show that our proposed scheme performs equally well compared to the baseline router design with double the buffer size. When buffer sizes are identical, our schemes achieves 41% better throughput than the baseline. When compared to ViChaR, the most aggressive dynamic VC allocation scheme that best utilizes buffer space, our results are similar or even better for some traffic patterns. However, we reduced its critical path delay by 60%, which allows us to clock the network at 1.5$\times$ higher frequency.

The remainder of this paper is organized as follows. Section 2 discusses related works on buffer designs. Section 3 discusses the details of proposed virtual channel allocation mechanisms as well as router architecture and adaptive routing design. Section 4 shows the experimental results. Finally, Section 5 concludes this paper.

2 Related Work

Due to the increasing stringent power constraint, there have been many work on reducing the buffer power dissipation in a router. One approach is to directly reduce the buffer size for lower power. The "iDEAL" [13] design utilizes the channel buffers to store flits during traffic congestion. Through this way, the buffer size inside the router can be decreased to save both dynamic and static power. This method has also been pushed to one extreme to eliminate entirely the buffers inside a router such as the Elastic Buffer [18] and bufferless routing [19] to achieve simple router design and improve energy efficiency. Flits are immediately passed to an output port upon arriving at an input port [19]. This is suitable for low traffic load where buffers are mostly under-utilized.

When buffer size is reduced, fewer flits can be stored in the router, and the network throughput is penalized. Therefore, many works have focused on improving the utilization of buffers to achieve high-throughput with small buffer sizes. ViChaR [23] is such a design which makes full use of every flit slot in a buffer. Whenever a flit comes in, it is stored in the buffer as long as there is a free slot. Flits within a packet may be distributed anywhere in the buffer, but their locations are bookkept in a control table. A VC is reserved when a header flit comes in, and released when its tail flit leaves the router. Hence, the number of VCs inside the router can range anywhere from f/p to f, where f if the buffer size in flits, and p is the number of flits per packet. As a result, the arbiters in the VA and SA stages need to be as wide as f:1. For example, if a buffer can store 4 complete packets, and each packet contains 5 flits, then the arbiters have a cardinality of 20:1. Our design in this paper reduces it to 4:1 without losing the throughput of the network. The buffer pool design [15] is similar to ViChaR except that each VC can store multiple packets, and its VC count is fixed. Though the buffer pool design uses smaller arbiters, its VC allocation scheme is dynamic while ours is between static and dynamic allocation, as explained next.

The DAMQ [28] and its improvements [22, 25] use fixed number of VCs, each dedicated to an output port with an extensible queue length. Hence, traffic heading to one direction is stored in one queue. The benefit of static VC allocation is that traffic from all directions have equal opportunities to compete for the crossbar, resulting good arbitration efficiency. Such philosophy can also be seen in the Row-Column(RoCo) Decoupled Router [12] design where VCs are grouped for traffic in x-direction and y-direction separately. The contention in SA arbitration is effectively reduced as shown in that work. The dynamic queue length in DAMQ is helpful to handle burst traffic from one direction. However, managing the dynamic queue is very costly in the original design. Even if the control table used in ViChaR was adopted, the cost would still be high as the table has to account for the longest queue length for each VC, resulting inefficient use of control table space. Therefore, our design fixes the VC queue size, and starts with a static VC allocation for good arbitration efficiency. When traffic becomes imbalanced, we start dynamic VC allocation to compensate for the inflexibility of the fixed queue size.

3 Proposed Virtual Channel Allocation

In this section, we will describe the state-of-the-art router architecture designs, followed by the introduction of our proposed router design with the simple buffer design.

3.1 Background: A Generic Router Architecture

A generic router in mesh topology has 5 input/output ports for four cardinal directions (North, South, West, East), and one local processing element (CPU or Cache Bank). When a header flit arrives at the router, it is first buffered in the determined VC in the buffer writing (BW) pipeline stage. Then, the routing computation(RC) unit determines the output port based on the destination information in the header flit and the routing algorithm. Next, the virtual channel allocation(VA) unit performs arbitration among all flits requesting for the same output VC. If the flit is able to obtain a free VC, it proceeds to the switch allocation(SA) stage where it arbitrates for the switch input and output ports. Once the flit is granted to use the output port, it proceeds to crossbar traversal(ST) stage where the flit traverses the crossbar, followed by the link traversal(LT) to the next router or PE [26]. Body and tail flits do not need to go through RC and VA stage, but SA is still necessary and it is done on 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. To shorten the pipeline depth, many prior works have proposed techniques such as look-ahead routing(LA) [7], speculative allocation [26], pipeline bypassing [15, 17, 21], aggressive speculation [21] to remove dependencies and parallelize pipeline stages. We use two-stage router design similar to [11] in this paper.

VC_ID	OP	RP	WP	Status	Pre-route	OVC
0	EAST	0	1	SA	EAST	0

Table 1. A sample entry of VC control table.

3.2 VC Buffer Organization and VC Control Table

The components we revise over the baseline two-stage router are in the VA stage. First, the buffers are implemented as SRAM (Static Random Access Memory) arrays. Each input port has a buffer which is organized as 4 VCs, each having 5 flits, the size of a packet. We will defer the allocation schemes of the VCs to later sections. The management of the VCs is carried through a control table as shown in Table 1. The table maintains the state of each VC. The "OP" is the output port produced by the RC of the last router to denote the required output port of the current node. The "RP" is the read pointer used as address to read flit from the SRAM and pass it on to the crossbar. The "WP" is the write pointer which indicates the write address for the next incoming flit. If the read pointer and write pointer are the same after a write operation, the VC buffer is full. If they are the same after a read operation, the buffer is empty. The "Status" field indicates the status or in which stage the packet in this VC is in — idle, RC, VA, SA, ST, and others. The "Pre-route" produced by RC at local node is the output port index for the next node downstream. The "OVC" is the VC ID for the flit to switch to in the downstream router. We use the credit-based flow control in this work, where a credit is the number of available buffer slots for each VC of the downstream node. It decrements whenever a flit is read out of the buffer and leaves the current router, and increments when it receives the one-bit control signal from downstream router to denote that a flit has left. If a flit obtained an output VC successfully, won the SA stage and the credit of the chosen VC was larger than zero, it can proceed to the ST stage and then traverse to the next node.

3.3 Arbiter

The arbitration stages in VA select a winner per output VC among all flits at the head of the VCs. This process VA together with SA are the bottleneck stages of the router pipeline [23]. since they account for most of the router control logic. Hence, their latencies determine the clock cycle time of the router. The VA and SA latencies are dependent on the cardinality of the arbiters, or the worst-case delay on the critical path. For our buffer size of 20 flits, an aggressive dynamic allocation scheme such as ViChaR requires 20:1 arbiters. This is a significant delay, area and power budget in the router, even when the actual number of requests are smaller than 20. Table 2 compares essential metrics for a 20:1 and 4:1 arbiter, the latter being used in our VA and SA. Both arbiters are implemented in HSPICE with 45nm PTM [33] device models at 1.1V and a temperature of 90°C. As we can see, the 20:1 arbiter is more than 4× slower, consumes more than 2× dynamic power and 7× static power, and takes more than 4× the area than a 4:1 arbiter. We also varied the number of inputs to each arbiter and observed that the dynamic power consumption of the arbiter is independent of the number of inputs. This is shown in Table 3. The reason behind it is that there is a priority vector inside the arbiter, and this vector changes according to some algorithm, e.g. Round Robin, on every arbitration. Hence, larger arbiters constantly consume more power than small arbiters. Based on the above observations, we conclude that simple arbiter design is more advantageous.

Arbiter Design	Delay (ps)	Dynamic Power (mW)	Static Power (μW)	Area (μm^2)
4:1 arbiter	140	1.07	1.59	120
20:1 arbiter	600	2.49	11.2	530

Table 2. Latency, area and power of an arbiter.

Arbiter Design	Request Type	Power (mW)
4:1 arbiter	0 request	1.049
	1 request	1.059
	4 requests	1.037
	random requests	1.079
20:1 arbiter	0 request	2.478
	1 request	2.459
	20 requests	2.455
	random requests	2.521

Table 3. Power of an arbiter with different types of inputs.

3.4 Arbitration Priority

Following our study, we use small arbiters in VA and SA for low latency and low power. We use 4 VC/port for our Fixed VC Assignment with Dynamic VC Allocation (FVADA) scheme, and 2∼5 VC/port for Adjustable VC Assignment with Dynamic VC Allocation (AVADA) scheme respectively. FVADA and AVADA will be introduced in next sections. The choice of VC count will also follow naturally there.

Since the VC resources in our design is not as ample as other dynamic VC allocation designs, we do not want a packet to hold many VCs across the intermediate router nodes. In current router design, once a portion of a packet occupies a VC queue, other packets are not allowed to use the rest free VC flit slots until the tail flit of the current packet comes, so as to avoid interleaving of flits from different packets. As a result, this scheme implies a poor buffer resource allocation since the vacant VC resources are wasted if the rest of the packet is blocked in some router upstream. To alleviate this problem, especially for small VC count design, the arbitration in our design gives higher priority to the body and tail flits over a new header flit. This way the body or tail flit can release their VC number sooner than otherwise. This scheme was used in [15] to improve SA efficiency. We use it here for fast recycling of VCs. If, however, the body or tail flit is not in the router yet, the arbiter still selects a header flit for VC allocation.

Note that such a priority rule will not incur starvation because at any time, there are only limited number of VCs in the middle of transferring a packet, and each packet has only limited number of flits. Hence, a header flit can always become a SA winner within finite number of arbitrations. Such a priority rule is not useful for other buffer management schemes with large number of VCs because VC resources are abundant. However, too many VCs could cause a packet to spread across many routers, which increases its latency in the network [5]. This will be discussed further in the performance evaluation section later.

In summary, therefore, there are two distinct characters in our design. The first one is that we use small number of VCs to keep the latency of every major component of the router low so that the router is able to accommodate high clock frequency. The second one is that the body/tail flits are given higher priority to mitigate inefficient buffer resources allocation. Next, we will describe our VC allocation mechanisms.

3.5 FVADA

Alleviating HoL. Since our VC number is small, using existing dynamic allocation would create many Head-of-Line (HoL) blocking, meaning that the flit at the head of a VC could not move forward due to output port being busy or no free buffer slot in the downstream router, then all other flits following this flit in that VC are blocked even if their required port and buffer resources are available. In order to reduce the HoL blocking and increase the network throughput, FVADA maps each VC to an output port. For example, the buffer for the East input port has 4 VCs, corresponding to West, South, North, and Local respectively. The packets destined to a particular output port will be queued into the corresponding VC. This is supported by the pre-routing outcome in the upstream router. Such a mapping is fixed and pre-defined. Figure 1 illustrates how this allocation can alleviate the HoL blocking.

Assume there are four incoming packets P1∼P4, with designated output ports EAST, EAST, WEST and WEST respectively in the current router. Consider a condition that the EAST output port is busy now, or, the downstream router does not have free buffer resources so that the flits heading to EAST have to stay in the current buffer. In Figure 1(a), it shows the scenario of the baseline router design: P1 and P2 at the head of VC0 and VC1 cannot move forward, so they block subsequent header flits of P3 and P4 that are routed to the WEST port. While in FVADA design, shown in Figure 1(b), the packets routed to the same directions are stacked in the same queue. The flits in VC1 routed to the WEST port are now separated from flits in VC0 since they go to different directions. They are not blocked from being transmitted to their next node.

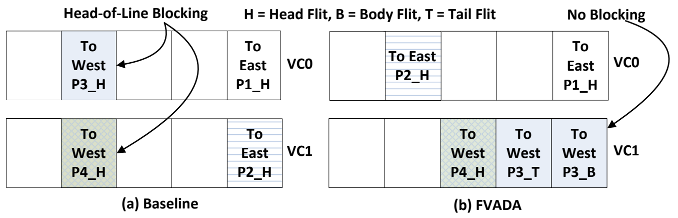


Figure 1. VC Allocation in FVADA. Four incoming packets P1∼P4: P1 to East, P2 to East, P3 to West, P4 to West. Flits heading to EAST cannot move forward now as EAST port is busy or the downstream router does not have free buffer resources.

Better Arbitration Efficiency. In addition to reducing the HoL blocking, FVADA can also improve the efficiency of the two-level SA arbitration. The first stage of SA selects a local winner from 4 VCs, the second stage selects from local winners those that can be forwarded to the next routers at the same time through the switch. With FVADA, more local winners may be selected in the second stage of SA, and so more flits can be passed on to their next routers, improving the overall network throughput. This is mainly attributed to the fixed mapping between the VCs and the output ports.

We use an example to illustrate this benefit. Figure 2 compares the arbitration results of a baseline router and FVADA for two cycles: t and $t + 1$. In a baseline router, there could be more number of VCs than in FVADA. Hence, packets heading to the same output port can be allocated to different VCs, such as the V0 and V1 in Figure 2(a). Both are routed to the EAST port. Assume that V0 and V1 are the local winner at t and $t + 1$ respectively. Let us consider two scenarios where they might fail the global arbitration in the baseline router but not in FVADA. In the first scenario,

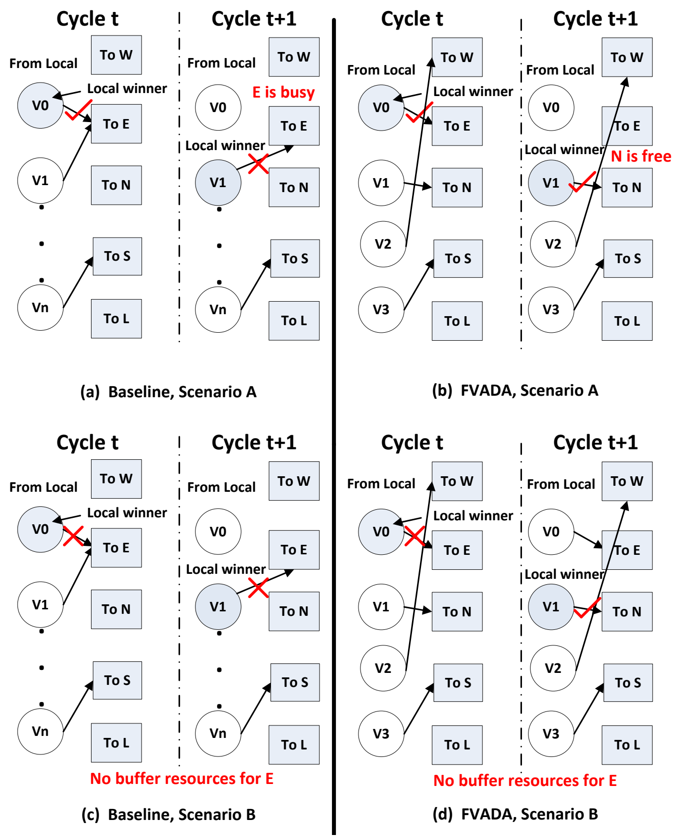


Figure 2. An example of FVADA Arbitration. Scenario A: Local input port has competitors for EAST output port. Scenarios B: The buffers of downstream router at EAST side are full.

suppose there is a Vx in *another* input port requesting also for the EAST port. In the baseline router, even if V0 wins the EAST port at t, V1 cannot win again at $t+1$ because the global arbitration will grant the request to Vx to ensure fairness. While in FVADA (Figure 2(b)), V1 may still win together with Vx because V1 stores flits heading to the North port. In the second scenario, suppose the EAST port downstream router is temporarily out of buffer resources. In the baseline router, as shown in Figure 2(c), both V0 and V1 will fail. Whereas in FVADA, V0 fails but V1 may still win at $t+1$ because the North port downstream router may have free buffer slots. As we can see, FVADA design is able to improve the global arbitration efficiency by improving the fairness among the *output ports* instead of the VCs. This can better utilize the available resources such as idle output ports and the vacant buffer slots in downstream routers.

Dynamic VC Allocation. When traffic is fairly evenly distributed across the network, FVADA performs well and delivers good throughput. However, if the distribution is not always even and packets traveling in certain directions dominate, the buffer utilization efficiency of FVADA will be significantly impaired because each direction has only one fourth of the total available buffer space. This pressure can quickly backfires to the sender where packets to other directions can be blocked simply because the local queue is filled.

To avoid performance degradation, FVADA employs dynamic VC allocation instead of static partitioning. During VC allocation, the "home" VC was first considered. A "home" VC is the one corresponding to the flit's output port. If this home VC is full or already reserved, the allocation logic will consider other free VCs with vacant buffer slots. If there is no such VC, the home VC will still be assigned to this packet. We term this process "VC Selection", a preparation step for "VC Assignment", illustrated in Figure 3. The selection logic can be implemented using very simple circuit, as one depicted in Algorithm 1 that produces the corresponding selection decision. This simple circuit can operate in parallel with the SA stage. The latency is only $90ps$ in 45nm node, compared to the $280ps$ two-level arbiter latency in the baseline router. Since FVADA can also allocate VCs dynamically, packets routed to different ports may still mingle together, though not as much as in the baseline dynamic allocated VCs. For example, we observed that the chances of a packet being mingled as such in the Bit-Complement Traffic at its saturation point is only 3%.

We remark that FVADA has two major advantages over the DAMQ design and its improvements. First, FVADA is capable of adjusting VC allocation based on traffic variation, which reduces more blocking than in DAMQ. The VC allocation of DAMQ is always fixed, which is inferior when the traffic distribution is uneven. For example, if the packet at the head of the VC for EAST port is blocked in some upstream router, then the entire VC is simply occupied and later packets cannot go into this VC even if the EAST port and the link are idle, wasting network resources. While FVADA takes a more flexible approach by starting with fixed allocation, and adjusting to varying VC occupancy. As a result, one blocking packet does not block all the packets to the same direction. Second, the overhead in managing the variable VC queue was overly expensive in DAMQ. Though later improvements could reduce the hardware implemented linked list, other overhead introduced turned out to be also prohibitive [22]. As we discussed earlier, even if we use a VC control table like the ones in [15, 23], its size must accommodate the longest queue length which is very inefficient in both area and power consumption. FVADA on the other hand has a very small control table. And the VA selection logic only requires several 2:1 multiplexers. Also it can be operated in parallel with the SA stage.

3.6 AVADA

In FVADA, the mapping between a VC and an output port index is fixed, meaning that VC_0 is always mapped to EAST, for example. An alternative design is to use an *adjustable* mapping between VCs and the output ports. The goal is to adapt the number of VCs to the varying traffic loads in all directions. For example, VC_0 can be mapped to EAST and later WEST output port. Also multiple VCs can be mapped to the same output port, depending on the traffic conditions. This can provide more flexibility in using

VCs. We term this mechanism Adjustable Number of VCs with Dynamic VC Allocation, or AVADA. AVADA does not require 4 VCs per input port so that the number of VCs does not depend on the number of output ports. This can vary, as shown later in simulation results where VC number per input port varies from 2 to 4.

The mapping between VCs and ports can be stored in a small table that is content addressable (CAM). Each entry corresponds to a VC number in the input port of the downstream router, and stores the mapped output port in that router, as indicated in the VC mapping table of Figure 3. During the VC selection stage, the outcome of the pre-routing will be used to lookup for an available VC from the mapping table. If found, then the VC (home VC) will be assigned to the flit if it has an available slot. Otherwise, an empty VC is identified and assigned to the flit. The mapping table is updated accordingly. If no empty VC can be found, then the selection logic will pick any VC that has an available slot. Here an available slot means a vacant one that follows a tail flit. The mapping table is extremely small — only 12 bits are necessary for a 4-VC buffer organization since 3 bits are necessary to index each output port and empty status.

3.7 FVADA and AVADA Implementation

In this section, we introduce the implementation of our proposed VC allocation mechanisms: FVADA and AVADA. We adopted the non-speculatively VA design [15] in our router. In a generic router designs, the routing unit returns an output VC for each packet. Then, the VA stage arbitrates among conflicting requests holding the same VC of the same input port in the next router. Our baseline VA stage is similar to ViChaR where VC assignment is performed *after* the arbitration stage, when the global winners are clear. This design results in simpler VA logic since it only needs to arbitrate for each output port, not each VC as in the generic case, and then assigns each global winner a VC pulled from a free-VC pool using some priority rules such as Random or FIFO.

Algorithm 1 **Control Logic for VC Selection in FVADA**

```
VD: home VC
VF: VC with free buffer slots
VO: Output VCID
**if** (VD does not have free buffer slots and there is VF) **then**
   VO = VF;
**else**
   VO = VD;
**end if**
```

In this paper, we adopt the concept of assigning VCs to the SA winners since there are at most 4 of them. The pipeline stages are shown in Figure 4, along with the baseline pipeline for comparison. The architecture details of the VC allocation is shown in Figure 3(b). Our router has only two stages. The first stage includes parallelized Lookahead routing (LA), Buffer Writing (BW), Switch Allocation and VC Allocation (SA+VA). The second stage includes Buffer Read (BR) and Switch Transmission (ST). In the first stage, we split VA into “VC Selection” and “VC Assignment”, and parallelize “VC Selection” with LA and SA. The SA performs typical two-stage arbitration to produce local and global winners for the output ports. The LA also performs typical pre-routing for the next router using either DOR or adaptive routing. The “VC Selection” logic, being in parallel with SA and LA does not have the knowledge of the global winners and their output ports in the next router. However, what it can do is to assume that each output port will have a winner, and prepare the VC candidates for that winner with all possible output directions it will go in the next router. For example, the “VC Selection” will use either the FVADA or AVADA algorithm to compute 4 VCs for the flit that will traverse through the EAST output to the next router, assuming the flit will be pre-routed to the EAST, NORTH, SOUTH and LOCAL port in the next router. Hence, once the outcomes of the SA and LA are available, the “SA Assignment” can simply choose from the VC candidates for the true winner (indicated by SA) and its true output direction (indicated by LA). Continuing the example, if there is a global winner for the EAST output port, and this flit will go to the SOUTH output port in the next router, then the VC prepared for that path is used in the assignment stage. As we can see, the only dependencies are from LA and SA to “VC Assignment”. There are no dependencies among LA, SA and “VC Selection”, and hence they can be parallelized.

Algorithm 2 **Control Logic for VC Selection in AVADA**

```
VD: home VC
VE: VC which is currently empty
VF: VC with free buffer slots
VFIFO: VCID at the head of available VC queue
VO: Output VCID
**if** (there is VD and VD has empty buffer slots or there is no VF) **then**
   VO = VD;
**else**
   **if** there is VE and VD has no free buffer slots or there is no VD **then**
      VO = VE;
   **end if**
**else**
   **if** there is VF and there is no VE and (no VD or VD has no free buffer slots) **then**
      VO = VF;
   **end if**
**else**
   VO = VFIFO;
**end if**
```

The algorithms of the “VC Selection” logic for FVADA and AVADA are shown in Algorithm 1 and 2 respectively. The “Home VC”, or VD is determined by the fixed map-

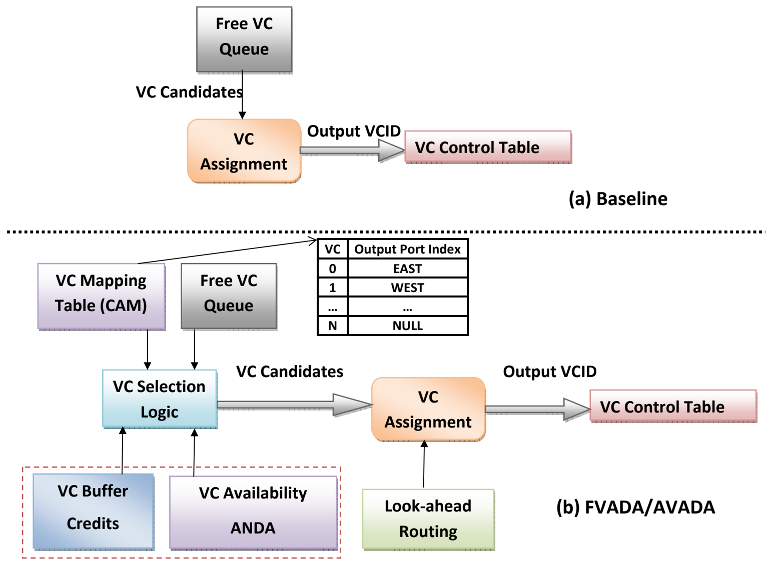


Figure 3. Proposed VC allocation architecture.

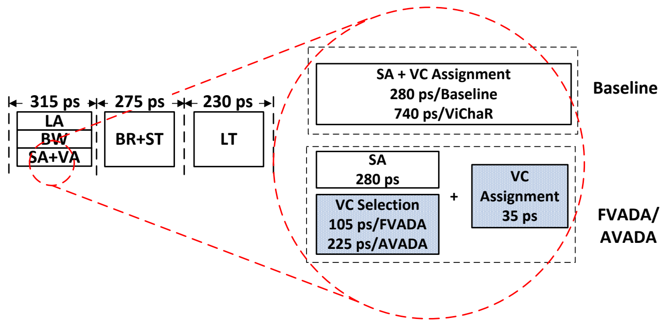


Figure 4. FVADA/AVADA router pipeline design.

ping in FVADA, or the CAM mapping in AVADA. "VF" is derived from the buffer credit. The "VE" used in AVADA is also provided by the CAM mapping table. The "VFIFO" is the head of free-VC queue, similar to that in the baseline and ViChaR. Finally, "VO" is decided based on both the buffer fullness and VC status. The former is indicated by the credit-based flow control, and the latter is provided by the AVADA's CAM table.

We now discuss the timing of different pipeline stages in Figure 4. The shaded boxes highlight the differences between the baseline and FVADA/AVADA. If Determined-Ordered Routing(DOR) is used, then the LA circuit requires only 70ps, measured from our circuit implementation using 45nm technology. The SA stage in baseline and our design requires 280ps, versus the 740ps required by ViChaR. This is because ViChaR uses large arbiters in this stage. The control logic of VC selection is very simple, and the sizes of VC mapping table and VC queues are small. The selection logic requires 105ps and 225ps for FVADA and AVADA respectively. The logic for AVADA includes a 12-bit CAM, which is slower than FVADA's logic but still tolerable. From those latencies, we can see that parallelizing LA, SA and VC Selection stages does not increase the stage time since SA is the bottleneck. The VC Assignment stage is different from the baseline and ViChaR, but it is only a 4:1 MUX (select one winner from four candidates). The total latency of (SA+VA) stage is 315ps, also listed in Table 4, which is much shorter than the delay in ViChaR.

3.8 Adaptive Routing

We also designed our buffer management to support adaptive routing, which is used to balance the link loads and improve the network performance, or to tolerate network failures.

To avoid deadlocks, we adapt a previous deadlock-free design [6] to our FVADA/AVADA. In that scheme, the VCs are divided into two classes: one employs minimum adaptive routing(AR) and the other employs DOR routing. Once packets are stored in the VC of DOR class, they can only use the VCs in the same class. For packets in the AR class, they can also pick VCs in the DOR class if the routing decision generates the same direction as DOR routing. The VCs of DOR class are used as escape VCs to break the deadlock, so at least one VC is reserved for this type of VC class. For FVADA, one more VC is necessary to apply deadlock avoidance since the other four VCs are mapped to four output ports. AVADA has more flexibility in VC mapping so deadlock-free design does not require additional VCs.

Previous dynamically allocated buffer designs [13, 23] use an existing VC as one of the escape VCs to break deadlock when a pre-specified time threshold is exceeded. Since FVADA/AVADA uses a small number of VCs, using a dedicated VC for escape is inefficient as it is not used frequently. Our objective here is to ensure deadlock-free without sacrificing the throughput. Therefore, The DOR VCs set in our design is not dedicated for escape. It can also be allocated to the packets when all the VCs of AR class are not available or the router is out of buffer resources. Once the packet is steered into the DOR class VCs, it uses DOR routing and stays in this class. This helps greatly in improving the VC and buffer utilization for small number of VCs design. The efficiency will be shown in section 4.

We remark that DAMQ would not work well with adaptive routing algorithm unless it doubles its VC count. This is because every VC is dedicated to one direction. If only one VC is added as escape VC, it would be filled with packets going to all directions, creating many HoL blocking which can severely hurt the network performance.

4 Performance Evaluation

In this section, we present simulation-based performance evaluation for our proposed FVADA/AVADA. We compare the results with the state-of-the-art generic router and dynamic VC Regulator (ViChaR) designs developed in previous researches [23].

Router Pipeline	Critical path delay (ps)
BW Stages	145
SA(baseline)	280
SA(FVADA/AVADA)	315
SA (ViChaR)	740
BR+ST	275

Table 4. Critical path delay for FVADA/AVADA and ViChaR designs.

Component (one input port)	Area (μm^2)	Dynamic Power (mW)	Static Power (μW)
Buffer(20 flits)	1500	37.39	1030
SA+VA logic (Baseline)	173	1.23	6.99
SA+VA logic(FVADA)	213	1.27	7.12
SA+VA logic(AVADA)	635	2.03	33.66
SA+VA logic (ViChaR)	666	2.79	24.77
Crossbar	36335	101.06	326.1

Table 5. Power and area for FVADA/AVADA and ViChaR designs.

4.1 Simulation Infrastructure

To model and compare different network designs, we use a cycle-accurate 2D NoC simulator Noxim [34] developed in SystemC. The simulator models all major components of the NoC. An 8×8 mesh topology with 2-stage pipelined routers are modeled in this paper. Each router has 5 ports, and each port has 4 VCs in the baseline design. Each VC is 5-flit deep. The size of each flit and the link bandwidth is 128 bits. The baseline, ViChaR and FVADA/AVADA are compared with equal size of buffers ($5 \times 4 = 20$ buffer slots for each input port). Each packet consists of one head flit, three body flits and one tail flit. The baseline router has fixed number of VCs with fixed VC size, but uses dynamic VC allocation. The ViChaR design used dynamically allocation buffer regulator to achieve variable number of VCs and adjustable VC depth.

We tested (1) Uniform Random traffic type where each node uniformly injects packets into the network with random destinations and (2) Permutation traffic type, where each node has dedicated destination node. We evaluated Bit-Complement, Transpose, Tornado, Butterfly, Bitreversal and Shuffle traffic. We used both DOR and deadlock-free minimum adaptive routing to measure the average network latency and saturation throughput of baseline, ViChaR and FVADA/AVADA designs.

4.2 Latency, Power and Area Estimation for the Router

The main components of the router were implemented to analyze the power, area and latency overhead. An SRAM based buffer in 45nm technology node with one read/write port is modeled using CACTI [29]. There are 20 entries in the buffer for each input port, with 128 bits in each entry. A 4-input arbiter, a 20-input arbiter and a crossbar are built and simulated in HSPICE with the 45nm PTM [33] device model. For the arbiters, we utilize a typical arbiter design described in [5], which updates the priority in a round-robin manner. The 5-port, 128 bit wide crossbar is built as a matrix crossbar. The control logics of VC allocation mechanisms are also simulated in HSPICE using 45nm Technology. All circuits are simulated with 1.1V V_{dd} and a temperature of 90°C. We use the πRC model for modeling the wires, and we assume a 25% activity factor in the simulation. The critical path delay of each pipeline stage of our design and ViChaR design is shown in Table 4.

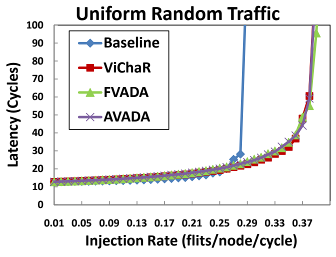
From Table 4, we can see the critical path delay of FVADA/AVADA is only 42.6% of ViChaR design. It is mainly because large number of VCs used by ViChaR requires complicated control logics in SA stage. The single-cycle of SA stage design for ViChaR can sustain a network frequency of no more than 1.35GHz. To obtain higher clock frequencies, the SA stage must be divided to multiple pipeline stages to accommodate smaller cycle time, such as the SA design in [20] which consists of 2 pipeline stages. We implemented a 3-stage ViChaR router that can sustain a higher frequency, e.g. 2.5GHz network. Since the critical delay of FVADA and AVADA is less than 400ps, the SA stage can complete in one cycle at 2.5GHz. We model both low frequency(1GHz) router and high frequency (2.5GHz) routers, and compare the performance among three designs.

Table 5 shows the power and area estimation of major router components. As FVADA/AVADA adds the VC selection logic and use different VC assignment from baseline, they consume higher power and area than the baseline design. However, the increase is still trivial. Power increases by 0.58%/1.1% of the total router power and area increases by 0.35%/4.1% for FVADA and AVADA respectively. The VC selection of AVADA also costs a little more than FVADA. Both schemes provide dynamic power and area savings in control logic compared to ViChaR, whose overhead mainly comes from the large arbiters.

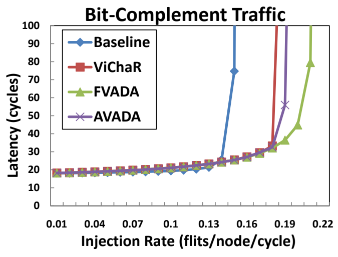
4.3 Simulation Results and Discussion

For DOR, baseline and FVADA/AVADA have 4VC/port, each VC can hold 5 flits, while ViChaR has equal-sized buffer slots (20 slots per port). Figure 5 plots the average flit latencies for a 8×8 mesh topology using DOR (a-g) and Adaptive Routing, AR, (h-i) with both types of traffic. Figure 5(a) and 5(b) show the average latency for uniform random (UR) and bit-complement traffic (BC). Both perform well with DOR but not AR [8] since the traffic by itself is fairly balanced. Local congestion information could be a disturbance and create global imbalance. For other traffic types in (c)$\sim$(g), imbalanced loads cannot be solved by DOR, and their saturation occurs earlier than in UR.

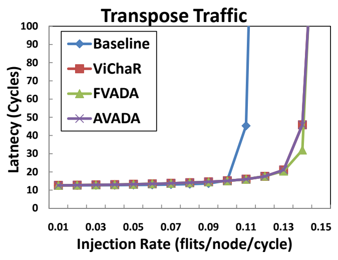
As we can see, FVADA/AVADA consistently outperform the baseline in throughput. The improvement is 41% on average and 66.7% maximally, even though both of them employs statically VC number and size. It proves that FVADA/AVADA can work well either in uniform or non-uniform traffic types. FVADA/AVADA have comparable performance in almost all traffic patterns. FVADA achieves better throughput than ViChaR in BC by 10.5% and in Shuffle Traffic Pattern by 4.7%. The reasons are two-fold: (1) FVADA improves the SA efficiency as we discussed in Sec-



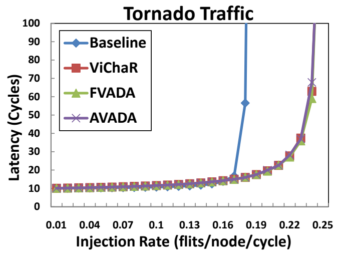
(a) Uniform Random



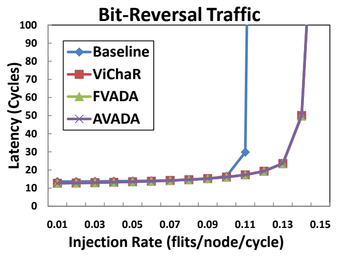
(b) Bit-Complement



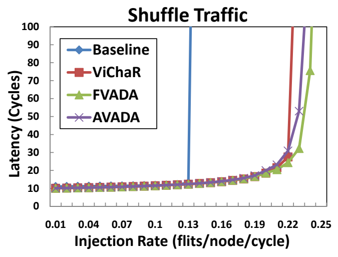
(c) Transpose



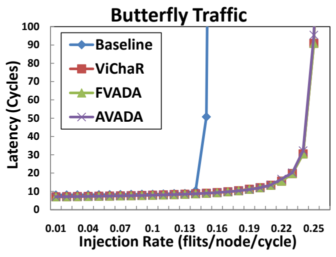
(d) Tornado



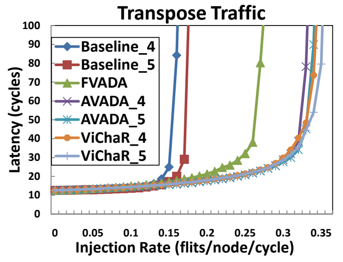
(e) Bit-Reversal



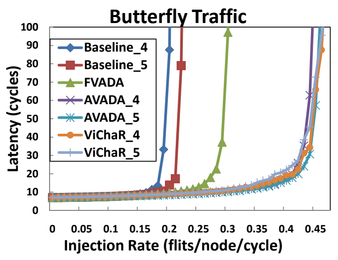
(f) Shuffle



(g) Butterfly



(h) Adaptive routing for Transpose



(i) Adaptive routing for Butterfly

Figure 5. Average network latency for a 8×8 network at 1GHz under DOR and Adaptive Routing.

tion 3.5. Ideally, we hope to produce winners in the first stage of SA that are heading to as many output ports as possible to reduce the contention in the second stage. For example, if the 4 first-stage winners go to 4 different directions, they are then all winners in the second stage because they can share the switch in one cycle. We collected this data for BC in Figure 6. It shows the number of output ports requested by the first stage local winners, relative to the ViChaR results. We can see that FVADA wins over ViChaR in 4 output ports by 20%. This is a strong proof of the SA efficiency in FVADA design. Again, this is mainly because VA assigns the VC based on their designated output port. (2)The average path length in BC is longer than other traffic types. ViChaR has a large number of VCs, which introduces higher number of interleaving packets, especially when the path is long. The packets occupy VCs resources throughout their routing path, increasing the average packet latency. AVADA, on the other hand, cannot match the performance of FVADA in BC and Shuffle traffic due to the dynamical VC mapping. Multiple VCs can be mapped to the same output ports, generating contention in SA. However AVADA has its advantages in adaptive routing as shown in Figure 5(h) and (i).

Figure 5(h) and (i) show the network latency under AR for Transpose and Butterfly traffic. "AVADA_5"means that it has 5 VC/port etc. As we mentioned before, FVADA uses fixed VC and output port mapping. To have a fair comparison, we also increase the number of VCs in the baseline and ViChaR. FVADA/AVADA still outperforms the baseline in terms of throughput by 52.5%/84.2%. AVADA has better performance than FVADA in adaptive routing because it has more flexible mapping. Multiple VCs can be assigned to DOR class to alleviate the HoL blockings without increasing the total number of VCs. AVADA with 4VC/port has 2.2% less throughput than VichaR with 4VC/port due to less VC resources. The scenario was discussed in section 3.8. For example, the buffer credits of the downstream router may show that the packet should pick a direction using AR over DOR. However, there might not be available VCs of class AR at current node. Then, VA cannot allocate a VC for the packet and it has to wait for several cycles. If the port select logic picks the DOR direction based on the VC resources information, then it has chosen a busier path for the packet. And once the packet enters the DOR class, it cannot come out and return to AR class VCs, which weakens the advantage of adaptive routing. For ViChaR design, VCs number is not a constraint factor so it does not limit the adaptive routing performance, resulting a slightly better throughput than AVADA.

Figure 8 shows the average latency for higher frequency router design. The no-load latency of FVADA/AVADA is reduced by 33% on average compared to ViChaR design. It

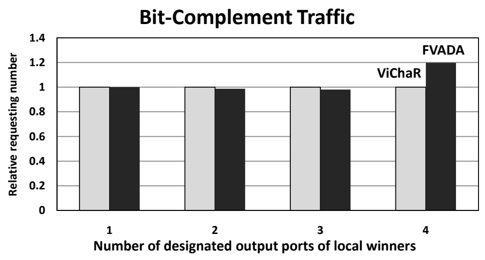


Figure 6. Number of designated output ports for local winners in FVADA and ViChaR at saturation injection rate.

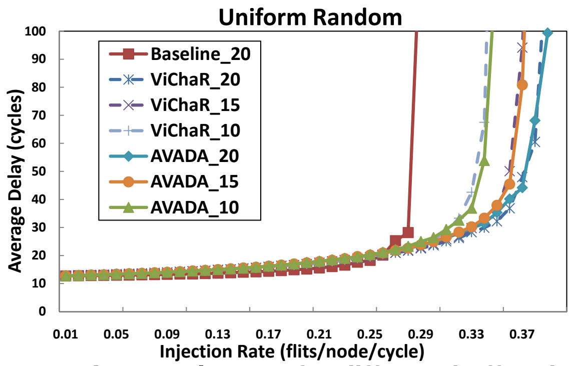


Figure 7. Average latency for different buffer sizes.

is mainly because the critical path delay of ViChaR is too long to fit in one cycle. FVADA/AVADA still have comparable saturation throughput with ViChaR design, and even better throughput in BC and the Shuffle traffic. Figure 8(h) and (i) show the network latency using adaptive routing. The saturation point of AVADA is able to match the ViChaR design for butterfly traffic. Because ViChaR has a deeper pipeline under high frequency, the delay of the credit loop also increases. The latency of the backward credit path impairs the utilization of buffer resources in a router because it increases buffer idle time between uses. Increasing the buffer turnaround time hurts the network throughput [27]. Therefore, the throughput of ViChaR is now the same as the AVADA design.

Finally, Figure 7 plots the latency curves for AVADA with various buffer sizes compared to ViChaR using equal-sized buffers, and the baseline with fixed buffer size under UR. This figure proves that our AVADA design can outperform the baseline even with half of the buffer size. We are always comparable to ViChaR under different sizes. Reducing buffer size by 50% could produce sigificant savings in both area and power, which alleviates the limitation of tight power budget in network design.

5 Conclusion

We propose two new VA mechanisms, termed Fixed VC Assignment with Dynamic VC Allocation (FVADA) and Adjustable VC Assignment with Dynamic VC Allocation (AVADA) to improve the buffer utilization. VCs are assigned based on the designated output port of a packet to reduce the HoL blocking. Also, the number of VCs allocated for each output port can be adjusted dynamically according to the traffic condition. A small number of VCs is used to keep the arbitration latency low. Simulation results show that for either uniform or non-uniform traffic, FVADA can improve the network throughput by 41% on average, compared to a baseline design with the same buffer size using DOR. Compare to FVADA, AVADA is more suitable for adaptive routing due to its dynamic VC mapping. AVADA outperforms the baseline even when our buffer size is halved. Therefore by using small buffer sizes, we could save significant power and area overhead and the throughput of the networks does not degrade. Moreover, we are able to achieve comparable or better throughput than a previous work ViChaR while reducing its critical path delay by 57.4%. Our results prove that the proposed simple VA mechanisms are suitable for low-power, high-throughput, and high-frequency on-chip network designs.

References

[1] J. Balfour and W. J. Dally, “Design tradeoffs for tiled CMP on-chip networks, *ACM International Conference on Supercomputing,* pp. 187-198, 2006.

[2] Y. Choi and T. M. Pinkston, “Evaluation of queue designs for true fully adaptive routers, ” *Journal of Parallel and Distributed Computing,* vol. 64, no.5, pp. 606-616, 2004.

[3] W. J. Dally, C. L. Seitz, “Deadlock-free message routing in multiprocessor interconnection networks, ” *IEEE Transactions on Computers,* vol. C-36, no.5, May 1987.

[4] W. J. Dally, “Virtual-channel flow control, ” *International Symposium on Computer Architecture,* pp. 60-68, 1990.

[5] W. J. Dally and B. Towles, “Principles and practices of interconnection networks,” Morgan Kaufmann, 2004.

[6] J. Duato, “Deadlock-free adaptive routing algorithms for multicomputers: evaluation of a new algorithm,” *Proceedings of the Third IEEE Symposium on Parallel and Distributed Processing,* pp. 840-847, 1991.

[7] M. Galles, “Scalable pipelined interconnect for distributed endpoint routing: the SGI SPIDER chip, ” *Hot Interconnect,* pp.141-146, 1996.

[8] P. Gratz, B. Grot and S. W. Keckler, “Regional congestion awareness for load balance in network-on-chip,” *International Symposium on High-Performance Computer Architecure,* pp. 203-214, 2008

[9] L. Hammond , B. A. Nayfeh , K. Olukotun, “A single-chip multiprocessor, ” *Computer,* vol.30 no.9, pp.79-85, Sep. 1997.

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

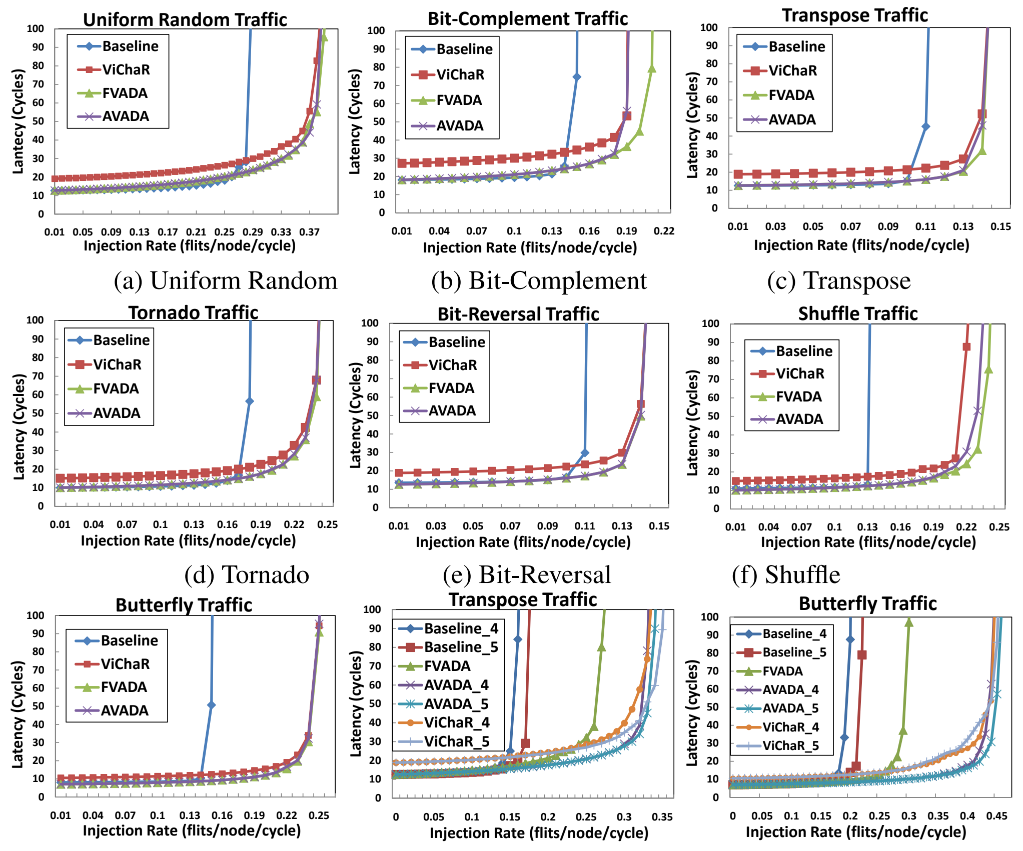
[11] J. Kim, D. Park, T. Theocharides, N. Vijaykrishnan and C. R. Das, “ A low latency router supporting adaptivity for on-chip interconnects, *DAC,* pp.559-564, 2005.

[12] J. Kim, C. A. Nicopoulos, D. Park, N. Vilaykrishnan, M. S. Yousif and C. R. Das, “A gracefully degrading and energy-efficient modular router architecture for on-chip networks,” *International Symposium on Computer Architecture,* pp. 4-15, 2006.

[13] A. K. Kodi, A. Sarathy, Vivek Venkatesan and A. Louri, “iDeal: Inter-router dual-function energy and area-efficient links for network-on-chip (NoC) architectures,” *International Symposium on Computer Architecture,* pp. 241-250, 2008.

[14] R. Kumar, V. Zyuban and D. M. Tullsen, “Interconnections in multicore architectures: understanding mechanisms, wverheads and scaling,” *International Symposium on Computer Architecture,* pp. 408-419, 2005.

[15] A.Kumar, L.-S. Peh, P. Kundu and N.K. Jha, “A 4.6 Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS,” *International Conference on Computer Design,* pp.63-70, 2007.



(a) Uniform Random (b) Bit-Complement (c) Transpose

(d) Tornado (e) Bit-Reversal (f) Shuffle

(g) Butterfly (h) Adaptive routing for Transpose (i) Adaptive routing for Butterfly

Figure 8. Average network latency for a 8×8 network at 2.5 GHz under DOR and Adaptive Routing.

[16] A. Kumar, L.-S Peh, P. Kundu and N. K. Jha, " Express virtual channels: torwards the ideal interconnection fabric," *International Symposium on Computer Architecture,* pp. 150-161, 2007.

[17] A. Kumar, L.-S Peh, P. Kundu and N. K. Jha, "Token Flow Control, " *International Symposium on Microarchitecture,* pp. 342-353, 2008.

[18] G. Michelogiannakis, J. Balfour and W. J. Dally, "Elastic-buffer flow control for on-chip networks," *International Symposium on High-Performance Computer Architecture,* pp. 151-162, 2009.

[19] T. Moscribroda and O. Mutlu, " A case for bufferless routing in on-chip networks," *International Symposium on Computer Architecture,* pp. 196-207, 2009.

[20] S. S. Mukherjee, P. Bannon, S. Lang, A. Spink, and D. Webb, "The Alpha 21364 network architecture, " *IEEE Micro,* vol. 22, no. 1, pp. 26-35, Jan./Feb. 2002.

[21] R. Mullins, A. West, and S. Moore, "Low-latency virtual channel routers for on-chip networks," *International Symposium on Computer Architecture,* pp. 184-197, 2004.

[22] N. Ni, M. Pirvu, and L. Bhuyan, "Circular buffered switch design with wormhole routing and virtual channels, " *International Conference on Computer Design,* pp. 466-473, 1998.

[23] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishman, M. S. Yousif, and C. R. Das, " ViChaR: A dynamic virtual channel regulator for network-on-chip routers," *International Symposium on Microarchitecture,* pp. 333-344, 2006.

[24] K. Olukotun, B. Nayfeh, L. Hammond, K. Wilson, and K.-Y. Chang, "The case for a single-chip multiprocessor," *International Symposium on Architectural Support for Programming Lanuages and Operating Systems,* pp. 2-11, 1996.

[25] J. Park, B. W. O'Krafka, S. Vassiliadis, and J. Delgado-Frias, "Design and evaluation of a DAMQ multiprocessor network with self-compacting buffers, " *Conference on Supercomputing,* pp. 713-722, 1994.

[26] L. S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," *International Symposium on High-Performance Computer Archtiecure,* pp. 255-266, 2001.

[27] L. S. Peh and W. J. Dally, "A delay model for router microarchitectures, " *IEEE Micro,* vol. 21, pp. 26-34, 2001.

[28] Y. Tamir and G. L. Frazier, "High-performance multiqueue buffers for VLSI communication switches, " *International Symposium on Computer Architecture,* pp. 343-354, 1988.

[29] S. Thoziyoor, J. H. Ahn, M. Monchiero, J. B. Brockman, N. P. Jouppi, "A comprehensive memory modeling tool and its application to the design and analysis of future memory hierarchies," *International Symposium on Computer Architecture,* pp. 51-62, 2008.

[30] S. Vangel et al., "An 80-tile 1.28TFLOPS network-on-chip in 65nm CMOS, " *The International Solid-State conference,* pp. 98-99, 589, Feb. 2007.

[31] S. Vangel, A. Singh, J. Haward, S. Dighe, N. Borkar and A. Alvandpour, "A 5.1GHz 0.34mm^2 router for network-on-chip applications, " *IEEE Symposium on VLSI Circuits,* pp.42-43, 2007.

[32] H.-S Wang, L.-S Peh, and S. Malik, "Power-driven design of router microarchitectures in on-chip networks, " *International Symposium on Microarchitecture,* pp. 105- 116, 2003.

[33] "PTM interconnect model," http://www.eas.asu.edu/∼ptm/interconnect.html

[34] "Noxim, an open network-on-chip simulator," http://noxim.sourceforge.net