RoShaQ: High-Performance On-Chip Router with Shared Queues

Anh T. Tran and Bevan M. Baas
University of California, Davis, USA
{*anhtr, bbaas*}*@ucdavis.edu*

***Abstract*—On-chip router typically has buffers dedicated to its input or output ports for temporarily storing packets in case contention occurs on output physical channels. Buffers, unfortunately, consume significant portions of router area and power. While running a traffic trace, however, not all input ports of routers have incoming packets needed to be transferred at the same time. As a result, a large number of buffer queues in the network are empty while other queues are mostly busy. This observation motivates us to design RoShaQ, a router architecture that maximizes buffer utilization by allowing to share multiple buffer queues among input ports. Sharing queues, in fact, makes using buffers more efficient hence is able to achieve higher throughput when the network load becomes heavy. On the other side, at light traffic load, our router achieves low latency by allowing packets to effectively bypass these shared queues. Experimental results show that RoShaQ is 21% less latency and 14% higher saturation throughput than a typical virtual-channel (VC) router with 4% higher power and 16% larger area. Due to its higher performance, RoShaQ consumes 7% less energy per a transferred packet than a VC router given the same buffer space capacity.**

I. Introduction

Systems on chip toward multicore design for taking advantage of technology scaling and also speeding up system performance through increased parallelism in the fact that power wall restricts increasing more clock frequency [1]–[3]. Networks on chip were shown to be feasible and easy to scale for supporting a large number of processing elements rather than point-to-point interconnect wires or shared buses [4]. Fig. 1 depicts a multicore system in which processors communicate together through a 2-D mesh network of routers. Each router has five ports which connect to four neighboring routers and its local processor. A network interface (NI) locates between a processor and its router for transforming processor messages into packets to be transferred on the network and vice versa.

In a typical router, each input port has an input buffer for temporarily storing packets in cases that output channels are busy. This buffer can be a single queue as in a wormhole (WH) router or multiple queues in parallel as in virtual-channel (VC) router [5]. These buffers, in fact, consume significant portions of area and power that are can be 60% and 30% of the whole router, respectively [6]. Bufferless routers remove buffers from the router so save much area [7], [8]; however, their performance becomes poor that in many cases cannot meet application requirements in which packet injection rates are high. Furthermore, due to having no buffers, packets have to be dropped and retransmitted or deflected once network contention occurs that can consume even higher energy than a router with buffers [9].

Other approach is sharing buffer queues that allows to utilize idle buffers [10] or emulate an output buffer router in order for obtaining higher throughput [11]. Our work differs from those router designs by allowing input packets at input ports to bypass shared queues so that it achieves lower zero-load latency. In addition, the proposed router architecture has simple control circuitry making it dissipate less packet energy than VC routers while achieving higher throughput by letting queues sharing workloads together when the network load becomes heavy.

The main contributions of this work are:

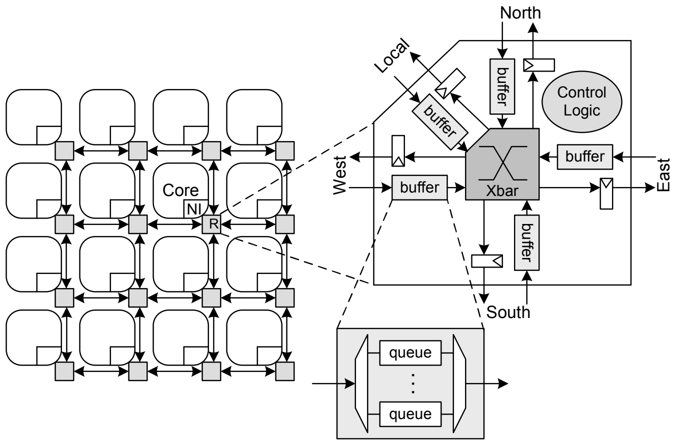

Fig. 1. A chip multi-processors (CMP) interconnected by a network of virtual-channel routers. NI: Network Interface; R: Router.

- exploring and analyzing shared-queue router architectures that maximize buffer utilization for boosting network throughput.
- proposing a router architecture that allows input packets to bypass shared queues for reducing zero-load packet latency.
- evaluating and comparing the proposed router with VC routers in terms of latency, throughput, power, area and packet energy.

This paper is organized as follows: Section II provides the background on router designs and motivation of this work. Section III presents our router architecture with all its components in details. The experimental results are shown in Section IV with analysis and comparison against VC routers. Section V reviews related work and, finally, Section VI concludes the paper.

II. Background and Motivation

We first review conventional on-chip router architectures with brief evaluation of their performance; and then we analyze their buffer utilization that is motivation of our new router design with shared queues.

A. Typical Router Architectures

Fig. 2(a) shows a typical WH router with three pipeline stages. The figure only shows details of one input port for simple view. At first, a packet at the head of an input queue will decide the output port for its next router (based on destination information contained in its head flit) instead of for the current router (known as lookahead routing computation (LRC) [12]). At the same time, it arbitrates for its output port at the current router because there may be multiple packets from different input queues having the same output port. If it wins the output switch allocation (SA), it will traverse across the crossbar in next cycle. One cycle after that, it then traverses on the output link towards next router. Both LRC and SA are done by the head flit of each packet; body and tail flits will follow the same route that has already been reserved by the head flit, except the tail flit should release the reserved resources once it leaves the queue.

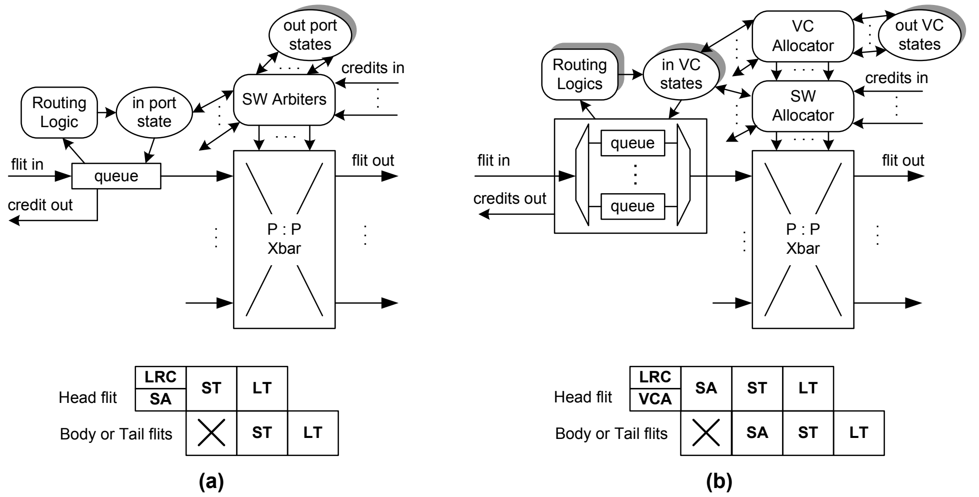

Fig. 2. Typical router architectures and their pipelines: (a) 3-stage wormhole (WH) router; (b) 4-stage virtual-channel (VC) routers. LRC: Lookahead Route Computation; VCA: Virtual Channel Allocation; SA: Switch Allocation; ST: Switch Traversal; LT: Output Link Traversal; (X): a pipeline bubble or stall.

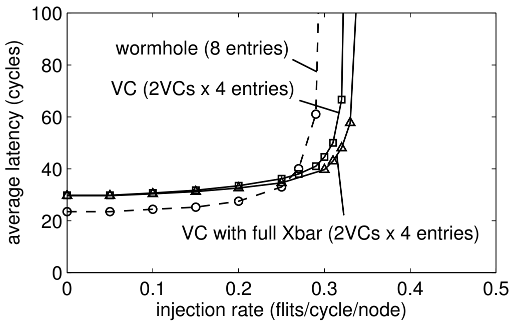

Fig. 3. Average packet latency simulated on a 8x8 2D-mesh network over uniform random traffic pattern.

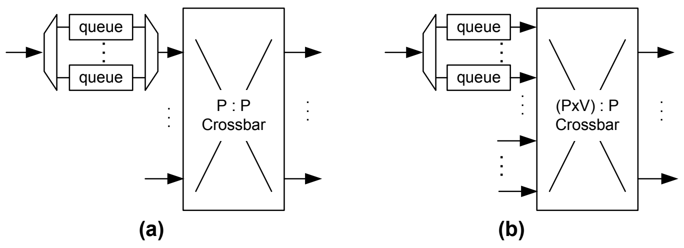

Fig. 4. Crossbar designs for a virtual-channel router: (a) P:P crossbar with V buffer queues of an input port are multiplexed; (b) PV:P crossbar that connects directly to all input buffer queues.

In a WH router, if a packet at the head of a queue is blocked (because it is not granted by the SA or the corresponding input queue of the down stream router is full), all packets behind it also stall. This head of line blocking problem can be solved by a virtual-channel (VC) router [5] as shown in Fig. 2(b). In this VC router design, an input buffer has multiple queues in parallel, each queue is called a virtual-channel, that allows packets from different queues can bypass each other to advance to the crossbar stage instead of being blocked by a packet at the head queue (however, all queues can also be blocked if all of them do not win SA or if all corresponding output VC queues are full). Because now an input port has multiple VC queues, each packet has to choose a VC of its next router's input port before arbitrating for output switch. Granting an output VC for a packet is given by a virtual-channel allocator (VCA); and this VC allocation is performed in parallel with the LRC; hence the router now has 4 stages as shown in Fig. 2(b). As a result, although a VC router achieves higher saturation throughput than a WH router while having the same number of buffer entries per input port, it also has higher zero-load latency due to deeper pipeline.

Fig. 3 shows the latency-throughput curves of a 8x8 2-D mesh network over uniform random traffic pattern with packet length of 4 flits. As shown in the figure, a VC router with 2 queues per input port (each queue has 4 entries) has 11% throughput gain compared to a WH router with 8 entries per queue; but its zero-load latency is 29 cycles that is also 26% higher than that of a WH router (23 cycles). Speculative virtual-channel routers proposed by Peh *et al.* and Mullins *et al.* can reduce zero-load latency of a VC router to the same as a WH router [13], [14]. However, at heavy network loads, speculation usually fails that makes its network same throughput as a typical VC router while consuming more power.

For achieving higher throughput, all VC queues of an input port can be connected directly to a crossbar with full input degree instead of being multiplexed as shown in Fig. 4. With this full-degree crossbar, after allocated an output VC, packet from a queue can directly arbitrate for its output port then would advance to next router if it wins; while with multiplexed-input crossbar, queues of the same input port have to beat together first before arbitrating for an output port. Clearly, the probability of winning both arbitration stages is less than winning only one arbitrator; hence, a VC router with full-crossbar (full-Xbar) achieves higher throughput than a typical VC router given the same number of VCs and buffer entries as shown by the curves in Fig. 3.

In section III, we will present our new router architecture that has low-load latency as a wormhole router while achieving higher throughput than a full-Xbar VC router without the need of pipeline speculation.

B. Buffer Utilization

Although buffers are expensive in terms of both power and area, they are not utilized well because at the same time not all input ports have packets to forward. Few input ports receive several packets while others may be empty. Fig. 5 shows two extreme states (full and empty) of buffer queues in a VC routers with 4 VCs per input port and 4 buffer entries per VC. This figure shows the percentages of

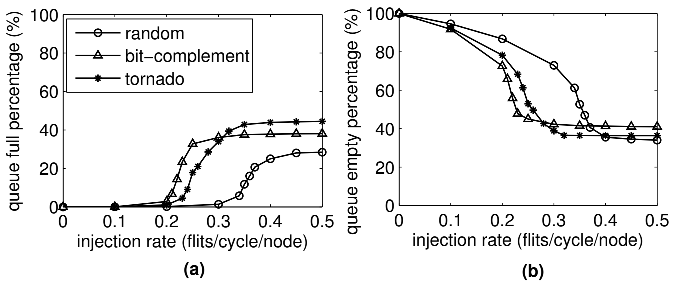

Fig. 5. Activities of router buffers vs. injection rate over synthetic traffics: (a), (b) are percentages of cycles that queues are full or empty in the whole simulation time.

number of cycles a queue is full and empty in 50,000 simulation cycles after 10,000 warmup cycles (that allows the network to be stable) over three synthetic traffic patterns on a 8x8 network.

When the load is low, queues are almost empty and never get full. When the network load increases, queues get full in more cycles and also less cycles to get empty. At the saturation status, around 30% and 40% of time queues get full over random traffic and bit-complement or tornado traffic, respectively. However, there are also around 40% of time queues get empty. We wish at this situation, empty queues can share their storage capacity with full queues that may allow more packets to be advanced rather than to be stalled, hence should improve more throughput. This motivates us to design a router with shared buffer queues as detailed in following section.

III. RoShaQ: Router Architecture with Shared Queues

A. The Initial Idea

For maximizing queue utilization, input ports of a router can share all queues as depicted in Fig. 6(a). With this architecture, incoming packets from an input port can be written to any shared queues. However, this architecture has critical drawbacks explained as follows. Because there is no buffer at input ports, when a packet from a upstream router needs to be forwarded, it has to send a request to downstream router and waits to receive the grant before sending data. As a result, the shared queue arbitrator for each router is highly complicated because it has to handle multiple requests from many shared queues of all neighboring routers. Furthermore, the round-trip inter-router request/grant delay can take several cycles plus the intra-router pipeline making zero-load network latency very high [15].

To alleviate this latency, each input port is dedicated one buffer queue and share all remaining queues as depicted in Fig. 6(b). With this design, since each output port connecting to an input queue of downstream router, shared queues only arbitrate for output port that is similar to a wormhole router. Input queues of each router also beat together to get grants to the shared queues. All request/grant signals are intra router, hence reduces latency and also allocation complexity. With this architecture, however, packets from input queues have to be buffered into the shared queues again before sent to output ports. This is actually unnecessary in the case when network load is low that unlikely causes much contention at output channels.

From this observation, we move on one more step by allowing input queues to bypass the shared queues as shown in Fig. 6(c). With this design, a packet from an input queue simultaneously arbitrates for both shared queues and an output port; if it wins the output port, it would be forwarded to the downstream router at next cycle. Otherwise, that means having congestion at the corresponding output port, it can be buffered to the shared queues. Intuitively, at low load, the network would has low latency because packets seem to frequently bypass shared queues. While at heavy load, shared queues are used to temporarily store packets hence reducing their stall times at input ports that would improve the network throughput. In next subsection, we will show in details circuit components that realize this router architecture.

B. RoShaQ Architecture

RoShaQ, a router architecture with shared queues based on Fig. 6(c), is shown in Fig. 7. When an input port receives a packet, it calculates its output port for the next router (lookahead routing), at the same time it arbitrates for both its decided output port and shared queues. If it receives a grant from the output port allocators, it will advance to its output port in next cycle. Otherwise, if it receives a grant to a shared queue, it will be written to that shared queue at next cycle. In case that it receives both grants, it will prioritize to advance to the output port.

Shared-queues allocator (SQA) receives requests from all input queues and grants the permission to their packets for accessing non-full shared queues. Output port allocator (OPA) receives requests from both input queues and shared queues. Both SQA and OPA grant these requests in round-robin manner to guarantee fairness and also to avoid starvation. Input queue, output port, shared-queue receiving/writing and shared-queue transmitting/reading states maintain the status (idle, wait or busy) of all queues and output ports, and incorporate with SQA and OPA to control the overall operation of the router. Another notice is that only input queues of RoShaQ have routing computation logic because packets in the shared queues were written from input queues so they already have their output port information. RoShaQ has the same I/O interface as a typical router that means they have the same number of I/O channels with flit-level flow control and credit-based backpressure management [16].

C. RoShaQ Datapath Pipeline

At first, a packet at the head of an input queue simultaneously performs three operations: LRC, OPA and SQA. At low network load, there is a high chance the packet to win the OPA due to low congestion at its desired output port; hence it is granted to traverse through the output crossbar and output link towards next router. Therefore, it incurs three stages including link traversal as depicted in Fig. 8(a). that is similar to a WH router pipeline.

When network becomes heavy, incoming packet may fail to get granted from OPA, but it can get a grant from SQA and is allowed to traverse the shared-queue crossbar and write to the granted shared queue in next cycle. After that, it arbitrates for the output port again and would traverse across the output crossbar and output channel toward the next router at next cycles if it is granted by the OPA at this time. Thus, in this situation, it incurs five inter-router stages as shown in Fig. 8(b). This larger number of traversing stages, in fact, allows the router to utilize shared-queues for reducing stall times of packets at input queues, hence improves throughput at heavy network load.

In both cases, body and tail flits of a packet traverse through the router on the same way as its head flit, except they do not need to arbitrate for resources (output ports and shared queues) that were already reserved by the head flit. The tail flit should also release these reserved resources once it leaves the queue.

D. Design of Allocators

This subsection describes the design of allocators for VC and RoShaQ routers. Let P and V be number of router ports and number

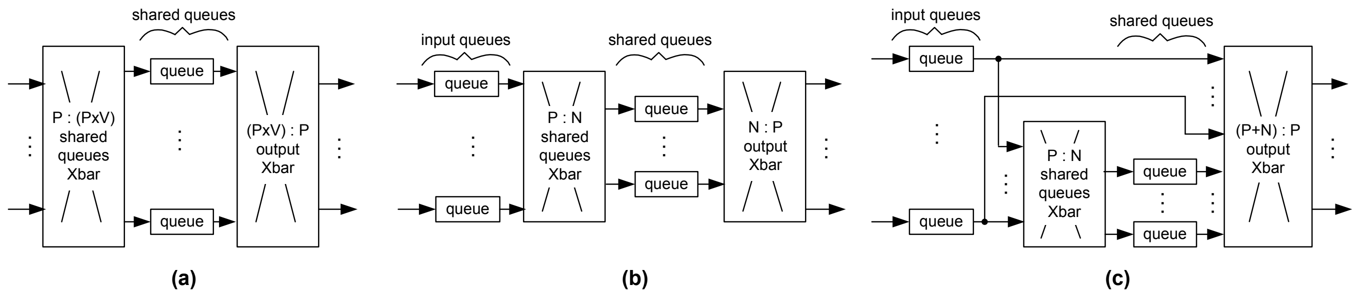

Fig. 6. Development of our ideas for sharing buffer queues in a router: (a) Shares all queues; (b) Each input port has one queue and shares the remaining queues; (c) Allows input packets to bypass shared queues.

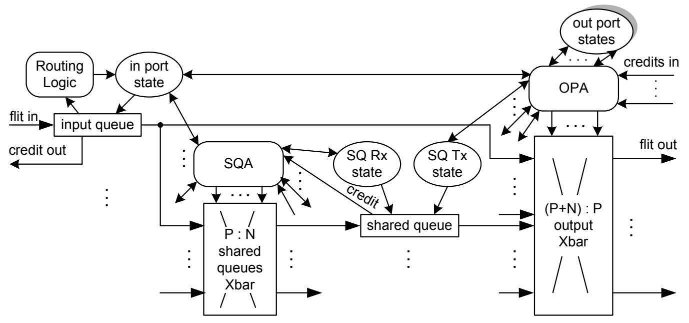

Fig. 7. RoShaQ router microarchitecture. SQA: shared queue allocator; OPA: output port allocator; SQ Rx state: shared queue receiving/writing state; SQ Tx state: shared queue transmitting/reading state.

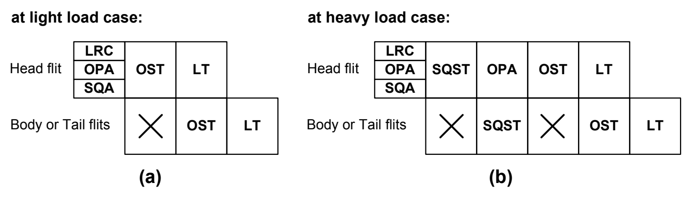

Fig. 8. RoShaQ pipeline characteristics: (a) 3 stages at light load; (b) 5 stages at heavy load. SQA: Shared Queue Allocation; OPA: Output Port Allocation; OST: Output Switch/Crossbar Traversal; LT: Output Link Traversal; SQST: Shared-Queue Switch/Crossbar Traversal.

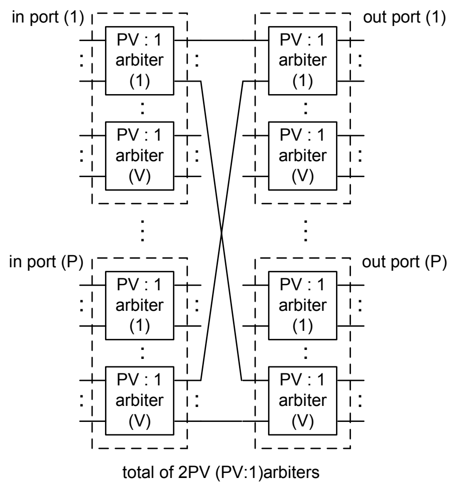

Fig. 9. Output virtual-channel allocator (VCA) in a virtual-channel router

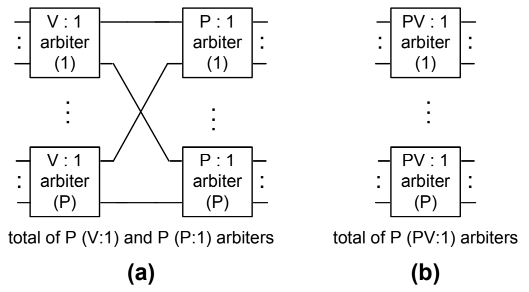

Fig. 10. Output switch allocator (SA) in: a) VC router with crossbar inputs multiplexed; b) VC router with full crossbar.

of VC queues per port in a VC router, respectively. Its VCA circuit is shown in Fig. 9 that has two stages of arbiters [13]. Each arbiter in the first stage chooses which output VC for a specific input VC; while an arbiter in the second stage chooses an input VC among several input VCs that were granted to the same output VC at the first stage. In total, this VCA consists of $2PV$ (PV:1) arbiters.

Fig. 10 shows the SA circuit designs for a typical VC router and for a VC router with full crossbar. Because input queues in the typical VC router are multiplexed before connected to the crossbar, its SA has two stages as shown in Fig. 10(a). First stage decides which input VC wins the input crossbar port; while second stage choose one among these winning input VCs for output ports. This SA consists of P (V:1) and P (P:1) arbiters. For a full-Xbar VC router, all input VCs directly arbitrate for output ports; so its SA consists of P (PV:1), each for one output port as depicted in Fig. 10(b).

The OPA and SQA of RoShaQ router are shown in Fig. 11. The OPA includes P (P+N:1) arbiters; each chooses one queue among input queues and shared queues that have the same output ports, where N is number of shared queues. In order for the total number of buffer queues to be identical to that of a VC router (PV queues in total), N is equal to $P(V-1)$ because each input port has one queue. So, the OPA is exactly the same as the SA of a full-Xbar VC router. The SQA includes two stages to allocate P input queues to N shared queues; so its circuit is the same as the SA of a VC router. This SQA is much low cost than a VCA; as a result, OPA and SQA of RoShaQ consume less area and power than VCA and SA of both typical and full-Xbar VC routers as will be shown in next section.

IV. Experimental Results

A. Experimental Setup

For evaluating performance of VC, full-Xbar VC and RoShaQ routers, we developed three cycle-accurate simulators, each for one router model. Experiments are performed over four synthetic traffic patterns as described in Table I. For uniform random traffic, each source processor chooses its destination randomly with uniform distribution, packet-by-packet. For other patterns, destination of each

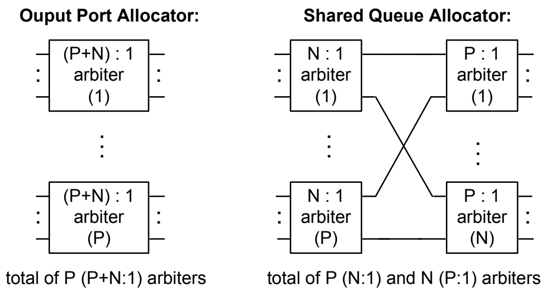

Fig. 11. Output port allocator (OPA) and shared queue allocator (SQA) structures in a RoShaQ router

TABLE I
SYNTHETIC TRAFFIC PATTERNS

traffic patten	description
uniform random	destination is randomly chosen, uniform dist.
transpose	[x,y] to [y,x]
bit-complement	[x,y] to [$\bar{x}$, $\bar{y}$]
tornado	[x,y] to [(x+3) % 8, (y+3) % 8]

source node is decided based on location of the source [16] as detailed in the table.

Performance of each router is evaluated by running simulation in 50,000 cycles with 10,000 warmup cycles on a 8x8 2-D mesh network where each network node consists of a processor and a router. Processor injects and consumes packets into and out of the network with each packet length is four 32-bit flits. We can employ any routing algorithm proposed in the literature [17] for routers; however, for comparing only the performance purely achieved by different architectural designs, we use the same XY dimension-ordered routing algorithm for all routers in this work. Latency of a packet is measured from the time its head flit is generated by the source to the time its tail flit is consumed by the destination. Average network latency is mean of all packet latencies in the network.

Table II describes six router configurations used in our experiments. VC2 and VC4 have 2 and 4 VC queues per input port, respectively. For fair evaluation, each queue of VC2 has 8 flit-entries while each queue of VC4 has 4 flit-entries. VC2-fullXbar and VC4-fullXbar have the same buffer configurations as VC2 and VC4 except their crossbars are full-degree (10:5 crossbar for VC2-fullXbar and 20:5 crossbar for VC4-fullXbar). For comparing with VC2 and VC2-fullXbar where each has total of 10 queues, RoShaQ5 that has 5 shared queues is used. Similarly, for comparing with VC4 and VC4-fullXbar where each has total of 20 queues, we use RoShaQ15 that has 15 shared queues. All routers have the same 80 flit buffer entries in total.

B. Latency and Throughput

The average packet latency of network corresponding to six router configurations over uniform random traffic is given in Fig. 12. As

TABLE II
ROUTER CONFIGURATION USED IN EXPERIMENTS. EACH ROUTER HAS 80 BUFFER ENTRIES IN TOTAL

router name	description
VC2	2 queues/input port, 8 entries/queue
VC2-fullXbar	the same as VC2, but using fullXbar
RoShaQ5	1 queue/input port, 5 shared queues, 8 entries/queue
VC4	4 queues/input port, 4 entries/queue
VC4-fullXbar	the same as VC4, but using fullXbar
RoShaQ15	1 queue/input port, 15 shared queues, 4 entries/queue

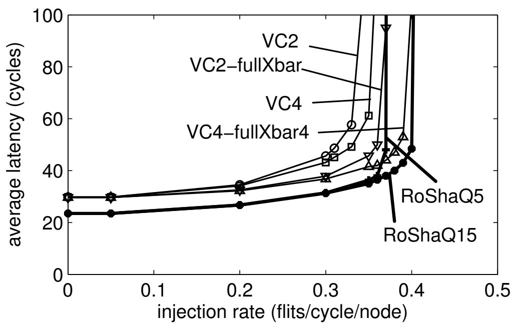

Fig. 12. Latency-throughput curves over uniform random traffic

shown, even having the same number of buffer entries, VC4 has higher saturation throughput than VC2 that is identical with results reported by Peh *et al.* [13]. Increasing number of crossbar input ports improves much more throughput. As shown, VC2-fullXbar achieves saturation throughput even higher than VC4. VC4-fullXbar has 11% saturation throughput higher VC4 (0.40 flits/cycle vs. 0.36 flits/cycle).

RoShaQ5 has the same saturation throughput as VC2-fullXbar (0.37 flits/cycle), while RoShaQ15 achieves 0.41 flits/cycle that is 2% higher VC4-fullXbar and 14% higher than VC4. More importantly, both RoShaQ5 and RoShaQ15 have zero-load latency of 23 cycles similar to a WH router that is 21% lower than all VC routers with and without full-degree crossbar. From these results, for simplicity, from now on we only provide the comparison results among RoShaQ15, VC4 and VC4-fullXbar in the rest of this paper. Comparison among RoShaQ5, VC2 and VC2-fullXbar gives a similar conclusion.

Fig. 13 shows packet latency of routers over three other traffics: transpose, bit-complement and tornado. As shown, similar to random traffic, RoShaQ outperforms VC4 over both bit-complement and tornado traffics. For transpose traffic, routers on the same row send packets to the same output direction; therefore, at saturation, throughput is limited by the output channel of the last router on that row. So all routers have the same saturation throughput of 0.14 flits/cycle. For bit-complement, RoShaQ15 is 2% and 8% higher saturation throughput; and for tornado-traffic, RoShaQ15 is 4% and 17% higher saturation throughput than VC4 and VC4-fullXbar, respectively. Again, over all traffics, RoShaQ15 achieves far lower latency than both VC routers before saturated because packets have a high chance to bypass shared queues.

C. Power, Area and Energy

Three router models (VC4, VC4-fullXbar and RoShaQ15) in Verilog RTL are synthesized targeting ST Microelectronics 65 nm low-power standard cells using Synopsis Design Compiler. Buffer queues are built from flip-flop registers; while each crossbar is a set of multiple multiplexers. Environmental parameters for the compiler are set at 1.2 V, 25^{o}C. We let the synthesis tool does all optimizations and automatically picks up standard cells in the library in order for all routers to meet 1 GHz clock frequency in the worst case.

Fig. 14 shows the synthesis power and area of three routers. 'Other' circuits in the figure include state of queues, routing computation and credit calculating circuitry. For taking into account the pipelined architecture of routers, the reported power and area of all components are also included their output pipeline registers. As seen in the figure, in the typical router VC4, buffers are expensive that occupy 53% area and consume 68% power of the whole router; while its crossbar only occupies 8%. VC4-fullXbar increases number of crossbar input ports that makes its router 7% larger area and 19% larger power than VC4

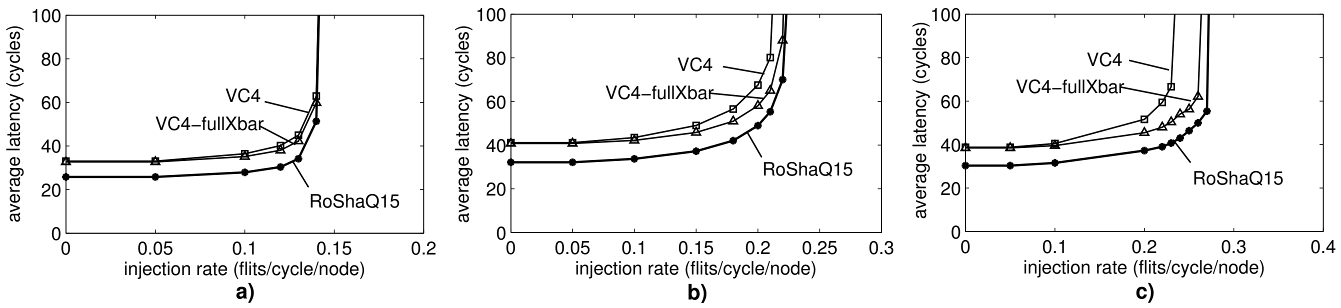

Fig. 13. Latency-throughput curves over traffic patterns: a) transpose; b) bit-complement; c) tornado.

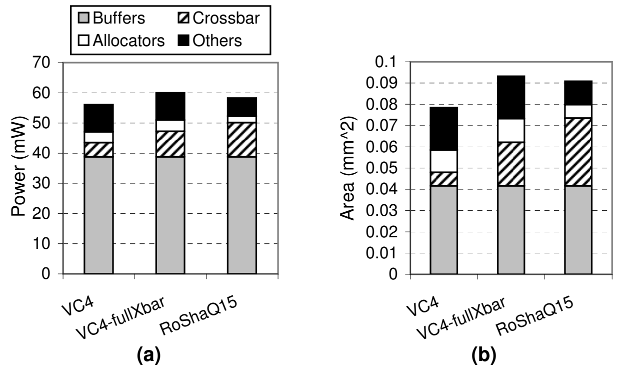

Fig. 14. Synthesis results: (a) power; (b) area.

TABLE III
ROUTER POWER AND AREA COMPARISON

	VC4	VC4-fullXbar	vs. VC4	RoShaQ15	vs. VC4	vs. VC4-fullXbar
power (mW)	56	60	+ 7%	58	+ 4%	-3%
area (mm^2)	0.078	0.093	+19%	0.091	+16%	-3%

as listed in Table III.

Because RoShaQ15 has two crossbars, its crossbars are 56% larger and consumes 35% higher power than the VC4-fullXbar's crossbar. However, interestingly enough, due to the simplicity of its allocators' circuits and less number of routing computation blocks (5 for 5 input queues compared to 20 for 20 virtual-channels in VC routers), the total router area and power of RoShaQ15 is even 3% less than these of VC4-fullXbar, respectively. Compared to VC4, RoShaQ is 4% and 16% larger power and area.

Another important metric to compare among router designs is the energy that routers in the network dissipate for transferring data packets [18]. As seen from latency-throughput curves, given a targeted packet latency of D cycles, the maximum injection rate is limited to R flits/cycle. For example, over uniform random traffic, from Fig. 12, to achieve an average packet latency of 60 cycles, R is 0.35, 0.39 and 0.40 for VC4, VC4-fullXbar and RoShaQ15, respectively. With injection rate of less than R, it needs more cycles to transfer a whole packet that means consuming more energy per packet. Therefore, the injection rate that minimizes packet energy while satisfying a given average packet latency should be equal to R.

Let L be packet length in the number of flits, then L/R is number of cycles to send a packet. Let E_c be energy per cycle of a router

TABLE IV
MAXIMUM INJECTION RATE R OF SOURCE PROCESSOR (FLITS/CYCLE) WHILE TARGETING AN AVERAGE PACKET LATENCY OF 60 CYCLES OVER SYNTHETIC TRAFFICS

	VC4	VC4-fullXbar	RoShaQ
random	0.35	0.39	0.40
transpose	0.14	0.14	0.14
complement	0.18	0.20	0.21
tornado	0.22	0.26	0.27

TABLE V
MINIMUM PACKET ENERGY E_p (PJ/PACKET) OF ROUTERS WHILE TARGETING AN AVERAGE PACKET LATENCY OF 60 CYCLES OVER SYNTHETIC TRAFFICS

	VC4	VC4-fullXbar	RoShaQ
random	640	615	580
transpose	1600	1714	1657
complement	1244	1200	1105
tornado	1018	923	859
average	1126	1113	1050

that is equal to router power P_r multiplying with cycle time T_c; [1] then energy per a transferred packet is:

$$E_p = E_c \frac{L}{R} = \frac{P_r \times T_c \times L}{R} \quad (1)$$

We choose 60-cycle packet latency for comparison that is two times of zero-load latency of VC routers and also before all routers get saturated. From figures 12 and 13, while targeting the same average packet latency of 60 cycles, the maximum injection rate R of each router corresponding to different traffics is given in Table IV. The router power P_r of routers at 1 GHz (T_c = 1 ns) was shown in row 2 of Table III. Hence, with packet length L of 4 flits, from Eqn. (1), the average packet energy each router dissipates corresponding to different traffics is given in Table V. As shown, due to higher injection rate given the same packet latency, RoShaQ15 consumes less energy per packet than both VC4 and VC4-fullXbar over three traffic patterns except the transpose one. This is because, for the transpose traffic, all routers have the same throughput of 0.14 flits/cycle at latency of 60 cycles while RoShaQ15 consumes more power. Averaging from all four traffic patterns, consequently, RoShaQ15 consumes 7% and 5% less energy per packet than VC4 and VC4-fullXbar, respectively.

[1] We assume a router consuming the same energy for each cycle that it transfers a flit. This is not always a case because activity of components may differ in cycle by cycle. However, for relative comparison among designs, this assumption can be used here for simplicity. For accurate evaluation, it requires a full simulation which takes into account activities of all router components that allows to accumulate their energy in whole simulation time before obtaining the average packet energy. This simulation-based evaluation is left for our future work.

V. Related Work

Peh *et al.* proposed a speculative technique for VC routers allowing a packet to simultaneously arbitrate for both VCA and SA giving a higher priority for non-speculative packets to win SA; therefore reducing zero-load latency in which the probability of failed speculation is small [13]. This low latency, however, comes with the high complexity of SA circuitry and also wastes more power each time the speculation fails. A packet must stall if even it wins SA but fails VCA, and then has to redo both arbitration at next cycle. Reversely, RoShaQ is non-speculative architecture. An incoming packet in RoShaQ only stalls if it fails both OPA and SQA; therefore it has high chance to advance either to be written to a shared queue (if it wins SQA) or be sent to output port (if it wins OPA) instead of stalling at an input port, and also reducing re-arbitration times.

Increasing crossbar input ports, that allows to directly connect to all virtual-channels of an input port instead of muxing them, improves much network throughput for VC routers. Using a large-radix crossbar is feasible and low-cost than adding more buffers as the results reported by DeMicheli *et al.* [19]. Recently, Passas *et al.* designed a 128×128 crossbar allowing to connect 128 tiles while occupying only 6% of their area [20]. This fact encourages us to build RoShaQ that has two crossbars while sharing cost-expensive buffer queues. The additional costs of crossbars are compensated by the simplicity of allocators and reducing the number of routing computation circuits that make our router better VC routers in many-fold: throughput, latency and packet energy.

IBM Colony router has a shared central buffer which is built from a time-multiplexed multi-bank SRAM array with wide word-width in order that it can be simultaneously written/read multiple flits (defined as a chunk) by input/output ports [21]. As a result, the central buffer is high cost and not identical with input queue design. RoShaQ has all buffer queues (both input and share queues) to be the same structure that allows to reuse the existing generic simple queues reducing practical design and test costs.

Latif *et al.* implemented a router with input ports sharing all queues [10] that is similar to the architecture illustrated in Fig. 6(a). Its implementation on FPGA shows more power and area-efficient than typical input VC routers. However, no router performance was reported and compared to VC routers. A similar approach is proposed by Tran *et al.* [15]; however, due to the high complexity of its allocators and also inter-router round-trip request/grant signaling, its performance is actually poorer than a typical router.

Ramanujam *et al.* recently proposed a router architecture with shared-queues named DSB which emulates an output-buffered router [11]. This router is similar to one illustrated in Fig. 6(b) that has higher zero-load latency than a VC router. This is because a packet has to travel through both two crossbars and be buffered in both input and shared queues at each router. Besides that, the timestamp-based flow control of DSB router design is highly complicated and hence consumes much larger area and power than a typical VC router (that are 35% and 58%, respectively). RoShaQ allows input packets to bypass shared-queues hence achieves lower zero-load latency compared to VC routers. RoShaQ also achieves much higher saturation throughput than VC routers, with only small area and power overheads while consuming less average energy per packet.

VI. Conclusion

We have presented RoShaQ, a new router architecture which allows to share multiple buffer queues for improving network throughput. Input packets also can bypass shared queues to achieve low latency in the case that the network load is low. Compared to a typical VC router, it is 21% lower zero-load latency and 14% higher saturation throughput with only 4% higher power and 16% larger area. It is also 2% higher throughput than a full-crossbar VC router with 3% less than power and area. While targeting the same average packet latency of 60 cycles, RoShaQ has 7% and 5% less energy dissipated per packet than typical VC and full-crossbar VC routers, respectively, while having the same buffer space. Its low latency, high throughput and low energy are achieved without the need of pipeline speculation.

Acknowledgments

This work was supported by a VEF Fellowship, SRC GRC Grant 1971 and CSR Grant 1659, ST Microelectronics, UC Micro, NSF Grants 0903549, 1018972 and CAREER Award 0546907, and Intel.

References

[1] J. L. Hennessy and D. A. Patterson, *Computer Architecture: A Quantitative Approach*, Morgan Kaufmann, San Francisco, USA, 2007.

[2] S. Borkar, "Thousand core chips: a technology perspective," in *Design Automation Conference (DAC)*, June 2007, pp. 746–749.

[3] C. H. V. Berkel, "Multi-core for mobile phones.," in *Design, Automation and Test in Europe (DATE)*, 2009, pp. 1260–1265.

[4] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Design Automation Conference (DAC)*, 2001, pp. 684–689.

[5] W. J. Dally, "Virtual-channel flow control," *IEEE TPDS*, vol. 3, pp. 194–205, Mar. 1992.

[6] Y. Hoskote et al., "A 5-GHz mesh interconnect for a teraflops processor," *IEEE Micro*, vol. 27, no. 5, pp. 51–61, Sept. 2007.

[7] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *Int. Symp. on Computer Architecture (ISCA)*, 2009, pp. 196–207.

[8] M. Hayenga et al., "SCARAB: A single cycle adaptive routing and bufferless network," in *IEEE/ACM Int. Symp. on Microarchitecture (MICRO)*, 2009, pp. 244–254.

[9] G. Michelogiannakis et al., "Evaluating bufferless flow control for on-chip networks," in *ACM/IEEE Int. Symp. on Networks-on-Chip (NOCS)*, 2010, pp. 9–16.

[10] K. Latif et al., "Power and area efficient design of network-on-chip router through utilization of idle buffers," in *IEEE Int Conf. and Workshops on Engineering of Computer Based Systems (ECBS)*, 2010, pp. 131–138.

[11] R. S. Ramanujam et al., "Design of a high-throughput distributed shared-buffer NoC router," in *ACM/IEEE Intl. Symp. on Networks-on-Chip (NOCS)*, 2010, pp. 69–78.

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

[13] L. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Intl. Symp. on High-Performance Computer Architecture (HPCA)*, Jan. 2001, pp. 255–266.

[14] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *Intl. Symposium on Computer Architecture (ISCA)*, Mar. 2004, p. 188.

[15] A. T. Tran and B. M. Baas, "DLABS: A dual-lane buffer-sharing router architecture for networks on chip," in *IEEE Workshop on Signal Processing Systems (SiPS)*, Oct. 2010, pp. 327–332.

[16] W. J. Dally and B. Towles, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, San Francisco, USA, 2004.

[17] N. Enright-Jerger and L. Peh, *On-Chip Networks*, Morgan-Claypool, 2009.

[18] A. Banerjee et al., "An energy and performance exploration of network-on-chip architectures," *IEEE TVLSI*, vol. 17, no. 3, pp. 319–329, 2009.

[19] G. De Micheli et al., "Networks on chips: from research to products," in *ACM/IEEE Design Automation Conference (DAC)*, 2010, pp. 300–305.

[20] G. Passas et al., "A 128x128 x 24gb/s crossbar interconnecting 128 tiles in a single hop and occupying 6% of their area," in *ACM/IEEE Int. Symp. on Networks-on-Chip (NOCS)*, 2010, pp. 87–95.

[21] C. B. Stunkel et al., "A new switch chip for IBM RS/6000 SP systems," in *ACM/IEEE Conference on Supercomputing (CS)*, 1999.