Achieving High-Performance On-Chip Networks With Shared-Buffer Routers

Anh T. Tran, *Member, IEEE*, and Bevan M. Baas, *Senior Member, IEEE*

***Abstract*— On-chip routers typically have buffers dedicated to their 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 budgets. While running a traffic trace, however, not all input ports of routers have incoming packets needed to be transferred simultaneously. Therefore, a large number of buffer queues in the network are empty and other queues are mostly busy. This observation motivates us to design router architecture with shared queues (RoShaQ), router architecture that maximizes buffer utilization by allowing the sharing 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 on a 65-nm CMOS standard-cell process show that over synthetic traffics RoShaQ has 17% less latency and 18% higher saturation throughput than a typical virtualchannel (VC) router. Because of its higher performance, RoShaQ consumes 9% less energy per transferred packet than VC router given the same buffer space capacity. Over real multitask applications and E3S embedded benchmarks using near-optimal NMAP mapping algorithm, RoShaQ has 32% lower latency than VC router and targeting the same application throughput with 30% lower energy per packet.**

***Index Terms*— Application mapping, networks on-chip, router architecture, shared-buffer, synthetic traffics.**

I. Introduction

SYSTEMS on chip toward multicore design for taking advantage of technology scaling and also for speeding up system performance through increased parallelism in the fact that power wall limits the increase of the clock frequency [1]–[3]. Networks on chip are 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]. A multicore system in which processors communicate together through a 2-D mesh network of routers is shown in Fig. 1. Each router has five ports that 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.

Manuscript received October 20, 2012; revised May 24, 2013; accepted June 9, 2013. Date of publication July 3, 2013; date of current version May 20, 2014. This work was supported in part by a VEF Fellowship, SRC GRC under Grant 1971 and CSR under Grant 1659, ST Microelectronics, UC Micro, NSF under Grant 0903549, Grant 1018972 and CAREER Award 0546907, and Intel.

The authors are with the Electrical and Computer Engineering (ECE) Department, University of California at Davis, Davis, CA 11111 USA (e-mail: anhtr@ucdavis.edu; bbaas@ucdavis.edu).

Digital Object Identifier 10.1109/TVLSI.2013.2268548

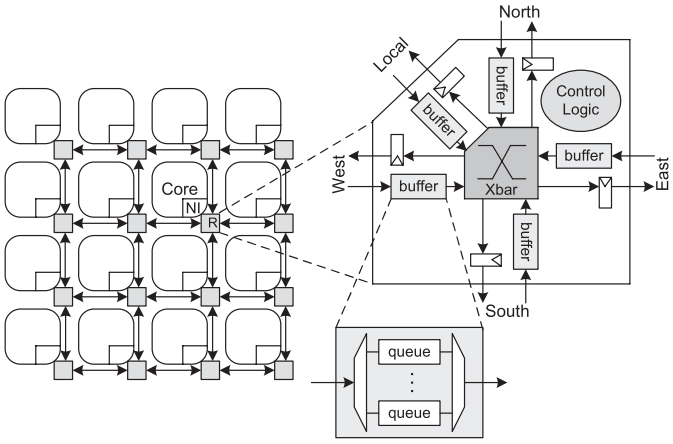

Fig. 1. Chip multiprocessors interconnected by a network of VC routers. NI: network interface. R: router.

In a typical router, each input port has an input buffer for temporarily storing the packets in case that output channel is busy. This buffer can be a single queue as in a wormhole (WH) router or multiple queues in parallel as in virtual channel (VC) routers [5]. These buffers, in fact, consume significant portions of area and power that can be more than 60% of the whole router [6]. Bufferless routers remove buffers from the router hence save much area [7], [8]; however, their performance becomes poor in case packet injection rates are high. Because of having no buffers, previous router designs proposed to drop and retransmit packets or to deflect them once network contention occurs that can consume even higher energy per packet than a router with buffers [9].

Another approach is by sharing buffer queues that allows utilizing idle buffers [10] or emulating an output buffer router to obtain higher throughput [11]. Our work differs from those router designs by allowing input packets at input ports to bypass shared queues hence, 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 and achieving higher throughput by letting queues share workloads when the network load becomes heavy.

Other techniques, such as dynamic voltage and frequency scaling [12]–[14] and globally asynchronous locally synchronous [15]–[17] can be used for reducing router power and energy while having only small effect on network performance. These techniques, however, are orthogonal to our work here which only focuses on architectural designs of on-chip routers. These techniques can be applied to this paper for further reducing the network power consumption.

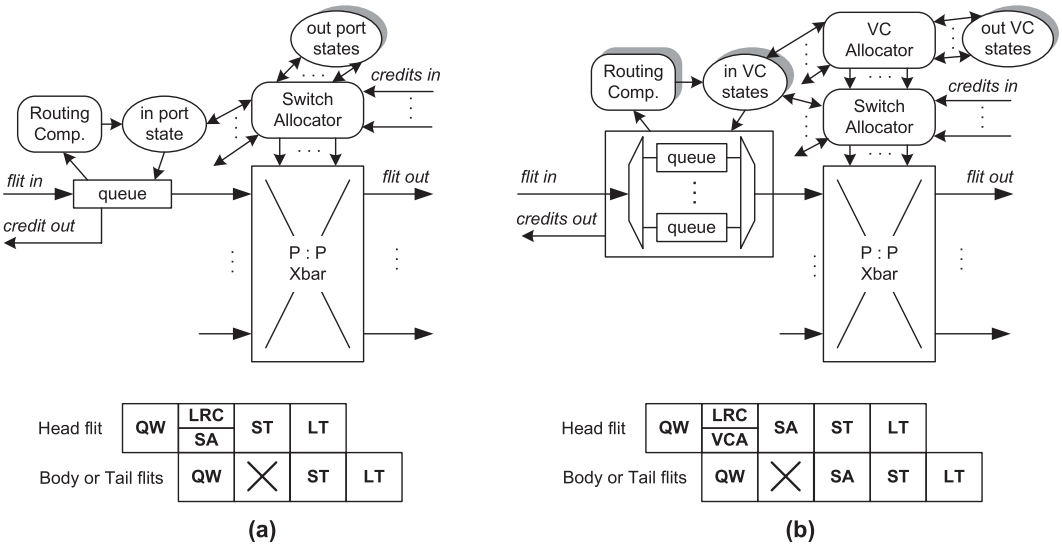

Fig. 2. Typical router architectures and their pipelines. (a) Four-stage WH router. (b) Five-stage VC routers. QW: queue write. LRC: lookahead route computation. VCA: virtual channel allocation. SA: switch allocation. ST: switch traversal. LT: output link traversal. (X): pipeline bubble or stall. P: number of router ports.

The main contributions of this paper are:

1) exploring and analyzing shared-queue router architectures that maximize buffer utilization for boosting network throughput;
2) proposing a router architecture which allows input packets to bypass shared queues for reducing zero-load packet latency;
3) evaluating and comparing the proposed router with VC routers in terms of latency, throughput, power, area, and packet energy over both synthetic and embedded application traffic patterns.

This paper extends the previous version published in the International Conference on Computer Design (ICCD 2011) [18] by providing more analysis on the router properties, such as deadlock and livelock freedom as well as the supported routing algorithms and network topologies. We also add more experimental results and comparison on both synthetic traffic patterns and real application traces with the performance, power, and energy data are based on cycle-accurate simulations. This paper is organized as follows. Section II provides the background on router designs and motivation of this paper. 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 this paper.

II. Background and Motivation

We first review conventional on-chip router architectures with brief evaluation of their performance, and then derive the motivation of our new router design using shared queues.

A. Typical Router Architectures

A typical WH router with four pipeline stages is shown in Fig. 2(a). This figure shows details of only one input port for simple view. The traveling process of a flit through a WH router is described as follows.

1) At first, when a flit arrives at an input port, it is written to the corresponding buffer queue. This step is called buffer write or queue write (QW).
2) Assuming without other packets in the front of the queue, the packet starts deciding the output port for its next router (based on the destination information contained in its head flit) instead of for the current router (known as lookahead routing computation (LRC) [19]). Simultaneously, 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. This step is called switch allocation (SA).[1]
3) If it wins the output SA, it will traverse across the crossbar. This step is called crossbar traversal or switch traversal (ST).
4) After that, it then traverses on the output link toward next router. This step is called link traversal (LT).

Both LRC and SA are done by the head flit of each packet; body and tail flits will follow the same route that is already reserved by the head flit, except the tail flit should release the reserved resources once it leaves the queue.

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 downstream router is full), all packets behind it also stall. This head of line blocking problem can be solved by a 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 VC, that allows packets from different queues to bypass each other to advance to the crossbar stage instead of being blocked by a packet at the head of the queue (however,

[1]With LRC, a router sends both output port information along with the packet to its downstream router. Therefore, the downstream router knows the output port of the packet at the time it receives the packet; hence, it can do both LRC (for next router) and SA at the same cycle.

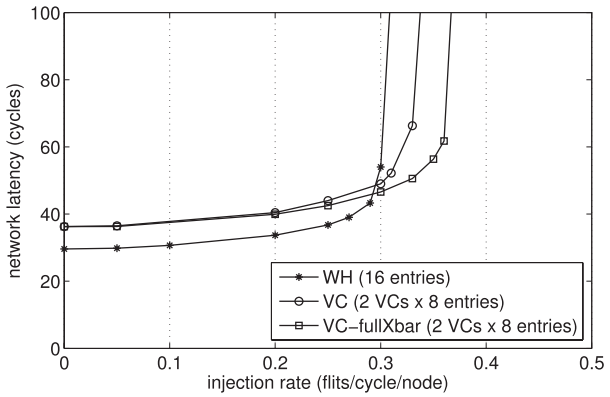

Fig. 3. Average packet latency simulated on a 8 × 8 2-D-mesh network over uniform random traffic pattern.

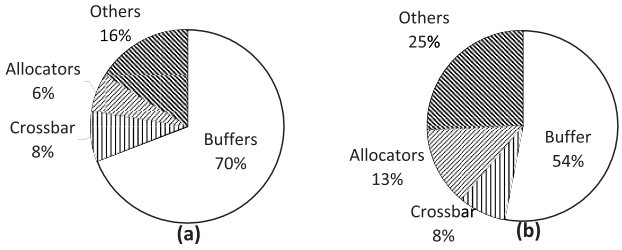

Fig. 4. Power and area costs of circuit components in a VC router with two VCs × eight flits per input port. (a) Power breakdown. (b) Area breakdown.

all queues at one input port can be still 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 five stages as shown in Fig. 2(b). Therefore, 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.

The latency-throughput curves of a 8 × 8 2-D mesh network over uniform random traffic pattern with packet length of four flits are shown in Fig. 3. As shown in this figure, a VC router with two queues per input port (each queue has eight entries) has 11% throughput gain compared with a WH router with 16 entries per queue; but its zero-load latency is 36 cycles which is also 20% higher than that of a WH router (30 cycles).

B. Router Cost and Opportunities for Improving Performance

We synthesized a 32-bit datapath 2 × 8 VC router using Synopsys Design Compiler with ultra optimization settings targeting a 65-nm CMOS standard cell process. The environmental parameters for the dc compiler are set at 1.2 V and 25 °C with clock frequency of 1 GHz. The router is hierarchically synthesized hence we could obtain the area and power data for all its circuit components. The router area and power breakdowns with the assumed switching activity factor

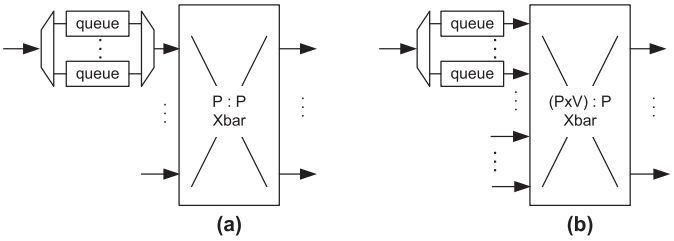

Fig. 5. Crossbar designs for a VC router. (a) P: P crossbar with V buffer queues of an input port are multiplexed. (b) PV: P crossbar that connects directly to all input buffer queues. P: the number of router ports. V: the number of queues per input port.

of 25% are shown in Fig. 4. As shown, buffer queues occupy 54% area and consume 70% power of the whole router; while crossbar consumes only 8%. If we use a higher radix crossbar, we could achieve higher throughput with a cost overhead still small compared with the cost of buffers as will be shown in Section IV.

Two crossbar designs for VC routers are shown in Fig. 5. The first one is for the traditional VC router where the input queues of each input port are multiplexed before being connected to the crossbar. The second one allows all input queues to connect directly a full input degree crossbar. With this full-degree crossbar, after allocated an output VC, a 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 compete 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 also shown in Fig. 3.

We also observe that, although the buffers are costly they are not well utilized. When an output channel of an upstream router does not sending packets, the input port of its downstream router is also idle while other input ports may be busy. This situation frequently happens for nonrandom deterministic traffic patterns after we mapped multitask applications onto a many-core platform. Under these traffic patterns, at run time, not all input ports of the routers have packets for processing. At many routers in the platform, a few their input ports receive packets all the time while others are often empty. Clearly, we wish at this situation, idle queues would share their storage capacity with busy queues of other input ports. This workload sharing would allow more packets to advance rather than being stalled at upstream routers hence, should improve the network throughput. This motivates us to design a router that can maximize the utilization of these high-cost buffer queues by sharing them in the run-time rather than by firmly dedicating them to each input port.

The proposed router should achieve the best performance in both cases: when the traffic load becomes heavy, the router allows utilizing shared buffers reducing packet stall times at input ports hence it can achieve higher throughput than a full-Xbar VC router; while at low-load packets can bypass shared queues hence has low latency similar to a WH router.

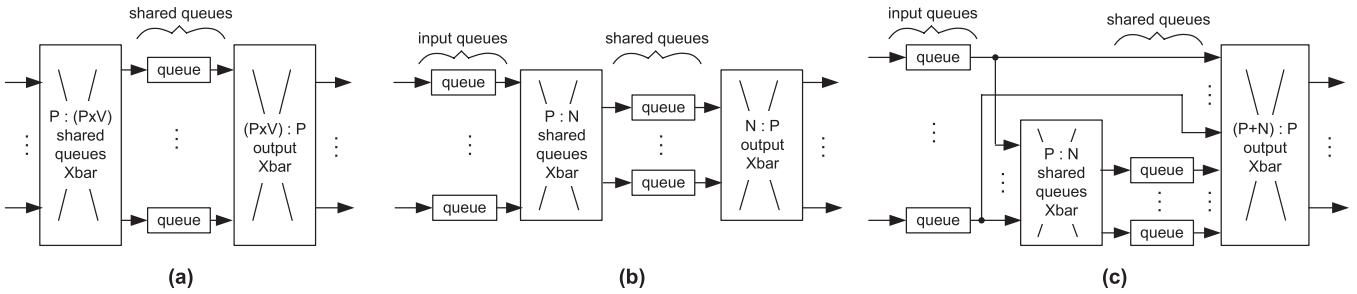

Fig. 6. Development of our ideas for sharing buffer queues in a router. (a) Shares all queues. (b) Each input port has one queue and shares the remaining queues. (c) Allows input packets to bypass shared queues. P: the number of router ports. V: the number of VC queues per input port in a VC router.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

III. RoShaQ: Router Architecture With Shared Queues

A. Initial Idea

For maximizing queue utilization, input ports of a router can share all queues as shown in Fig. 6(a). With this architecture, incoming packets from an input port can be written to any shared queue. 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 wait to receive the grant before sending data. Therefore, the shared queue arbitrator for each router is highly complicated because it must handle many external requests from multiple shared queues of all neighboring routers. In addition, the round-trip interrouter request/grant delay can take several cycles plus the intrarouter pipeline making zero-load network latency very high [20].

To alleviate this latency, each input port is dedicated one buffer queue and shares all remaining queues as shown in Fig. 6(b). With this design, as each output port connecting to an input queue of downstream router, shared queues arbitrate for an output port that is similar to a WH router. Input queues of each router also compete together to get grants to the shared queues. All request/grant signals are intrarouter signals, hence reduces latency and also allocation complexity. With this architecture, however, packets from input queues must be buffered into the shared queues again before being sent to output ports. This is actually unnecessary in case the network load becomes 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 the 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 have 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 the next section, we will show in detail circuit components that realize this router architecture.

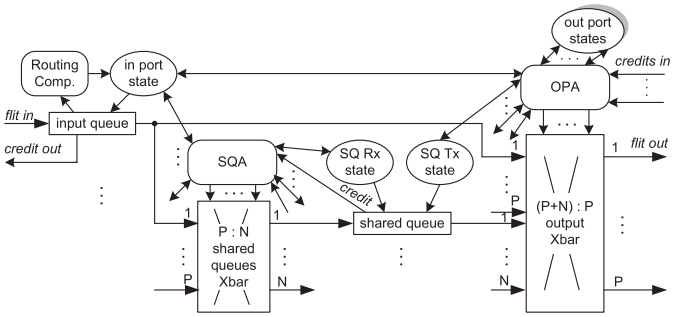

Fig. 7. RoShaQ router microarchitecture. SQA: shared-queue allocator. OPA: output port allocator. SQ Rx state: shared queue receiving/writing state. SQ Tx state: shared queue transmitting/reading state. P: the number of router ports. N: the number of shared queues.

B. RoShaQ Architecture

RoShaQ, a router architecture with shared queues based on the idea of 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 (OPAs), it will advance to its output port in the next cycle. Otherwise, if it receives a grant to a shared queue, it will be written to that shared queue at the 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 nonfull shared queues. Packets from input queues are allowed to write to a share queue only if: 1) the shared queue is empty or 2) the shared queue is containing packets having the same output port as the requesting packet. This shared queue writing policy guarantees deadlock-free for the network as will be explained in Section III-E.

The 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 and livelock. Input queue, output port, and shared-queue states maintain the status (idle, wait, or busy) of all queues and output ports, and incorporate with SQA and OPA

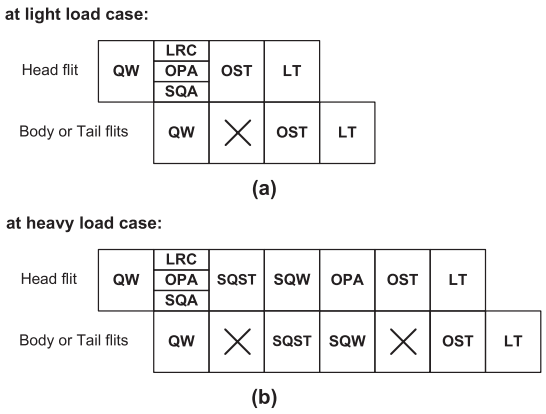

Fig. 8. RoShaQ pipeline characteristics. (a) Four stages at light load. (b) Seven stages at heavy load. QW: queue write. LRC: lookahead routing computation. OPA: output port allocation. SQA: shared queue allocation. OST: output switch/crossbar traversal. LT: output link traversal. SQST: shared-queue switch/crossbar traversal. SQW: shared-queue write. (X): pipeline bubble or stall.

to control the overall operation of the router. Only input queues of RoShaQ have routing computation logic because packets in the shared queues were written from input queues hence 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 [21].

C. RoShaQ Datapath Pipeline

After a packet is written into an input queue in the first cycle, in the second cycle it 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 toward next router. Therefore, it incurs four stages including link traversal as shown in Fig. 8(a) that is similar to a WH router pipeline.

When network load becomes heavy, the packet at an input queue 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 cycles. 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 seven interrouter stages as shown in Fig. 8(b). This larger number of traversing stages, in fact, allows the router to use 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 in the same way as its head flit, except they do not need to arbitrate for the resources (output ports and shared queues) that are already reserved by the head flit. The tail flit should also release these reserved resources once it leaves the queue.

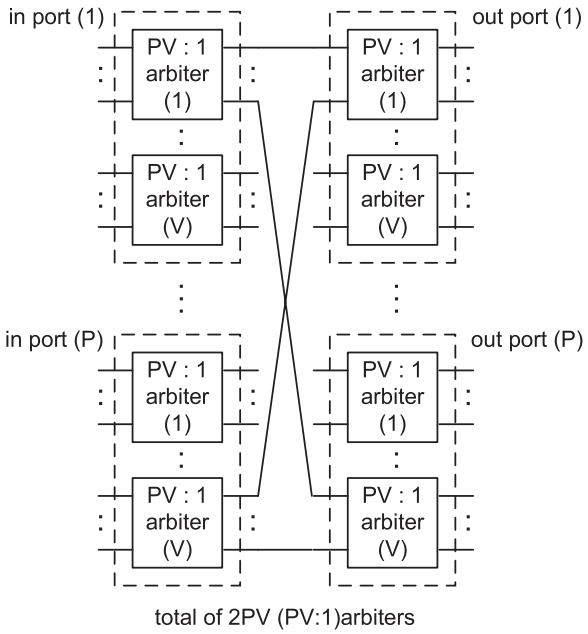

Fig. 9. Output VCA in a VC router. P: the number of router ports. V: the number of virtual channels per input port.

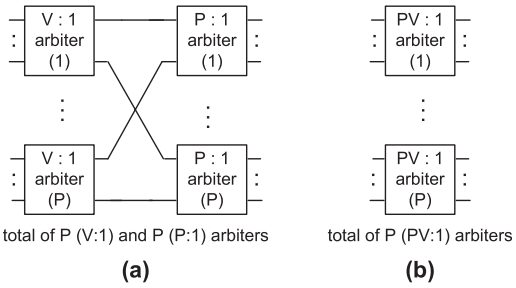

Fig. 10. Output switch allocator (SA) in (a) VC router with crossbar inputs multiplexed and (b) VC router with full crossbar. P: the number of router ports. V: the number of virtual channels per input port.

D. Design of Allocators

This section describes the design of allocators for VC and RoShaQ routers. Let P and V be the number of router ports and number of VC queues per port in a VC router, respectively. Its VCA circuit is shown in Fig. 9 that has two stages of arbiters [22]. 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 are granted to the same output VC at the first stage. In total, this VCA consists of $2PV$ (PV:1) arbiters.

The SA circuit designs for a typical VC router and for a VC router with full crossbar are shown in Fig. 10. Because input queues in the typical VC router are multiplexed before being connected to the crossbar, its SA has two stages as shown in Fig. 10(a). The first stage decides which input VC wins the input crossbar port; while the second stage chooses 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; hence its SA consists of P (PV:1) arbiters, each for one output port as shown in Fig. 10(b).

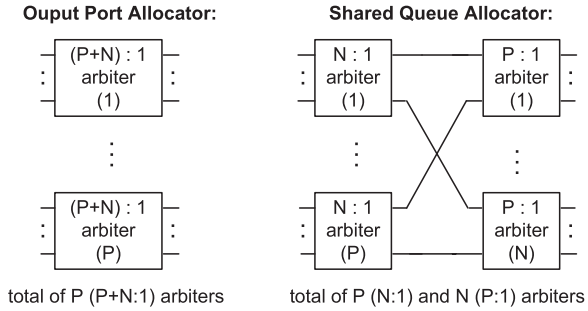

Fig. 11. OPA and SQA structures in a RoShaQ router. P: the number of router ports. N: the number of shared queues.

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. 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. Therefore, 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; hence its circuit is the same as the SA of a VC router. This SQA is much low cost than a VCA; therefore, OPA and SQA of RoShaQ consume less area and lower power than VCA and SA of both typical and full-Xbar VC routers as will be shown in next section.

E. RoShaQ's Properties

1) *A network of RoShaQ routers is deadlock-free*. At light load, packets normally bypass shared queues, so RoShaQ acts as a WH router hence the network is deadlock-free [23]. At heavy load, if a packet cannot win the output port, it is allowed to write only to a shared queue which is empty or contains packets having the same output port. Clearly, in this case RoShaQ acts as an output-buffered router which is also shown to be deadlock-free [24].
2) *A network of RoShaQ routers is livelock-free*. Because both OPA and SQA use round-robin arbiters, each packet always has a chance to advance to the next router closer to its destination; hence the network is also free from livelock.
3) *RoShaQ supports any adaptive routing algorithm*. The output port for each packet is only computed at its input queue, not at shared queues. Therefore, any adaptive routing algorithm which works for WH routers also works for RoShaQ.
4) *RoShaQ can be used for any network topology*. If we hide all design details inside RoShaQ, we would see RoShaQ only has one buffer queue at each input port similar to a WH router. Therefore, we can change the number of RoShaQ's I/O ports to make it compatible with any network topology known in the literature along with an appropriate routing algorithm.

TABLE I
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

IV. EXPERIMENTAL RESULTS

A. Experimental Setup

Six router configurations used in our experiments are listed in Table I. VC2 and VC4 have two and four VC queues per input port, respectively. For fair evaluation, each queue of VC2 has eight 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 in 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 ten queues, RoShaQ5 that has five 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.

For evaluating performance of VC, full-Xbar VC and RoShaQ routers, we developed three cycle-accurate simulators, each for one router model [25]. Experiments are performed over eight common synthetic traffic patterns, seven real-world multitask applications and three E3S embedded benchmarks that have large number of tasks.

B. Latency and Throughput

1) Synthetic Traffic Patterns: We conducted the experiments over eight common synthetic traffic patterns which cover a wide range of interconnect patterns on 2-D mesh networks [21]. For uniform random traffic, each source processor chooses its destination randomly with uniform distribution in packet-by-packet basic. For other patterns, destination of each source node is decided based on the location of the source as follows:[2]

1) *bit-complement*: from $[x, y]$ to $[\bar{x}, \bar{y}]$;
2) *transpose*: from $[x, y]$ to $[y, x]$;
3) *bit-shuffle*: from $[x_2x_1x_0, y_2y_1y_0]$ to $[x_1x_0y_2, y_1y_0x_2]$;
4) *tornado*: from $[x, y]$ to $[(x+3)\%8, (y+3)\%8]$;
5) *bit-rotate*: from $[x_2x_1x_0, y_2y_1y_0]$ to $[x_0x_2x_1, y_0y_2y_1]$;
6) *neighbor*: destination is randomly chosen among four nearest neighbors of the source on a probability of 80%, and is randomly among other processors on a probability of 20%;
7) *regional*: destination is randomly chosen among processors with distances to the source of at most three on

[2]Here x and y are values of horizontal and vertical coordination of a node in a 8 × 8 mesh; $x_2x_1x_0$ and $y_2y_1y_0$ are binary representatives of x and y, respectively.

TABLE II
ZERO-LOAD LATENCY AND SATURATION THROUGHPUT OF ROUTERS OVER EIGHT DIFFERENT SYNTHETIC TRAFFIC PATTERNS

	Zero-Load Latency (cycles)			Sat. Throughput (flits/cycle/node)		
Traffic Patterns	VC4	VC4-fullXbar	RoShaQ15	VC4	VC4-fullXbar	RoShaQ15
random	36.01	36.01	29.83	0.35	0.39	0.40
bit-complement	49.06	49.06	40.27	0.18	0.20	0.21
transpose	39.71	39.71	32.73	0.17	0.17	0.17
bit-shuffle	30.01	30.01	24.97	0.21	0.22	0.23
tornado	46.85	46.85	38.53	0.22	0.26	0.27
bit-rotate	30.04	30.04	25.01	0.20	0.23	0.24
neighbor	14.30	14.30	12.38	0.75	0.80	0.83
regional	22.68	22.68	19.08	0.60	0.69	0.76
Average	33.58	33.58	27.84	0.33	0.37	0.39

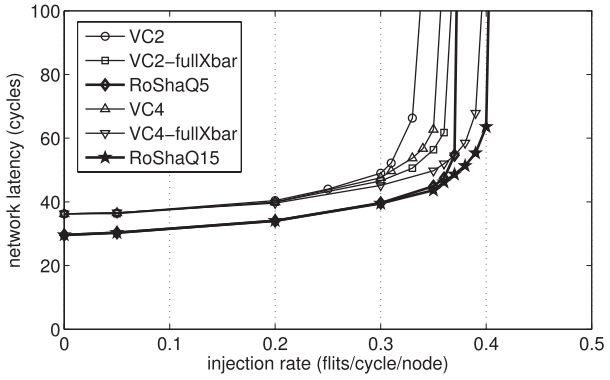

Fig. 12. Latency-throughput curves over uniform random traffic.

a probability of 70%, and is randomly among other processors on a probability of 30%.

Performance of each router is evaluated by running simulations of 100 000 cycles with 20 000 warmup cycles on a 8×8 2-D mesh network where each network node consists of a processor and a router. Processors inject and consume packets into and out off the network, and each packet length is four 32-bit flits. As mentioned in Section III-E, we can employ any routing algorithm proposed in the literature [26] for routers; however, for comparing the performance purely achieved by different architectural designs, we use the same XY dimension-ordered routing algorithm for all routers in this paper. 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 the mean of latency of all packets in the network.

The average packet latency of networks corresponding to six router configurations over uniform random traffic is shown in Fig. 12. As 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.* [22].[3] Increasing the number of crossbar input ports improves throughput significantly. As shown, VC2-fullXbar achieves saturation throughput even higher than VC4. VC4-fullXbar has 15% saturation throughput higher VC4 (0.39 flits/cycle versus 0.35 flits/cycle).

[3]Saturation throughput is defined as the injection rate at which network latency reaches about three times of the zero-load latency [27]. In this paper, the saturation throughput is assumed when network latency is 100 cycles.

RoShaQ5 achieves a saturation throughput of 0.37 flits/cycle which is 3% higher than VC2-fullXbar. RoShaQ15 achieves 0.40 flits/cycle throughput that is 3% higher VC4-fullXbar and 14% higher than VC4. More importantly, both RoShaQ5 and RoShaQ15 have zero-load latency of 30 cycles similar to a WH router that is 17% lower than all VC routers with and without a full-degree crossbar (36 cycles). 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.

We run simulations for all eight synthetic traffic patterns; zero-load latency and throughput of routers are listed in Table II. As shown, RoShaQ outperforms both VC routers in seven traffic patterns, except in transpose pattern. 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. Therefore all routers have the same saturation throughput of 0.17 flits/cycle.

Especially, for neighbor and regional patterns, because destination of each packet is quite close to its source, the network incurs less congestion. Therefore, packets in RoShaQ routers often bypass shared queues to achieve both lower latency and higher throughput than both VC routers. On average over all eight traffic patterns, RoShaQ has 18% and 5% higher throughput than VC and VC-fullXbar routers, respectively, with 17% lower zero-load latency.

2) Real Application Communication Traffic: Many DSP and embedded applications can be represented by a communication task graph where each task can be mapped onto one or multiple processing units (processors, accelerators, or memory modules) [28]. The task graph of a video object plan decoder (VOPD) application [29] which also shows the intertask communication bandwidth requirements is shown in Fig. 13(a).

For generating the experimental traffic of this application, we adopted the method proposed by Hu *et al.* [30] and Lan *et al.* [31]. In this method, we transform the required bandwidth on each intertask connection into the injection rate of the corresponding sending task. A task which requires large sending bandwidth also has large injection rate, and vice versa. Let bw_i and bw_j be the required bandwidth of tasks T_i and T_j to other tasks in the communication graph,

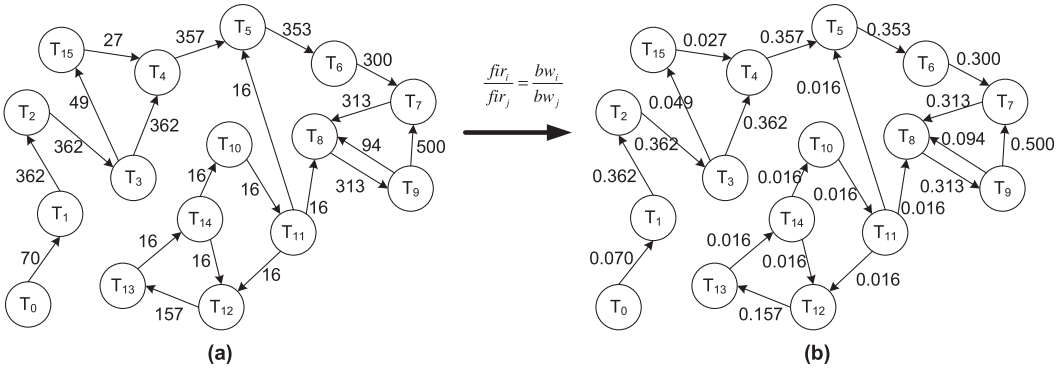

Fig. 13. Communication graph of the VOPD application and the corresponding injection rate of each processor used in our simulation. (a) Required intertask bandwidths in Mb/s. (b) Corresponding injection rates of processors in flits per cycle.

TABLE III
SEVEN EMBEDDED APPLICATIONS AND THREE E3S BENCHMARKS USED IN OUR EXPERIMENTS

Applications	No. Tasks	Net. Size
Video object plan decoder (vopd) [29]	16	4×4
Multimedia system (mms) [32]	25	5×5
Multi-window display (mwd) [33]	12	4×3
WiFi baseband receiver (wifirx) [34]	25	5×5
H.264 CAVLC encoder (cavlc) [35]	16	4×4
MPEG4 (mpeg4 [36]	12	4×3
Video conference encoder (vce) [37]	25	5×5
E3S auto-indust (autoindust) [38]	24	6×4
E3S consumer (consumer) [38]	12	4×3
E3S telecom (telecom) [38]	30	6×5

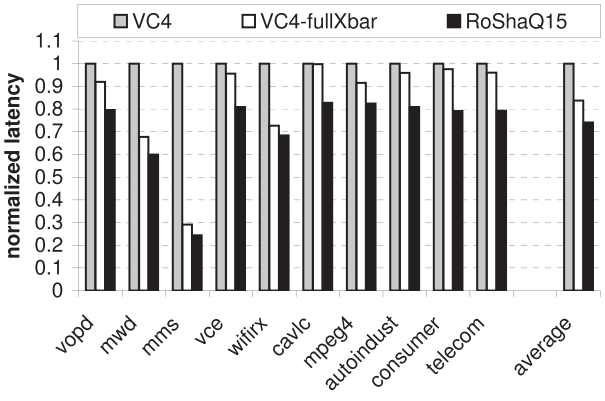

Fig. 14. Normalized latency of real applications with task-to-core random mapping.

then injection rates of tasks T_i and T_j on the corresponding links follow the equation $fir_i/fir_j = bw_i/bw_j$. Therefore, if given an injection rate of any task, we can easily calculate injection rates for all other tasks on all links in the graph.

An example for setting the injection rates of tasks which are corresponding to the communication graph of VOPD application are shown in Fig. 13(a) and (b). In this example, if we choose injection rate for task T_0 to task T_1 is 0.07 flits/cycle, then injection rate for task T_1 to task T_2 is $0.07 \times 362 / 70 = 0.362$ flits/cycle. Similarly, we can derive injection rates for all tasks in the application graph as shown in the figure.

In this paper, we use communication graphs of seven real-world applications and three E3S embedded benchmarks [38] which have large numbers of tasks for our experiments. Application names and their number of tasks are shown in Table III. Depending on the number of application tasks, we decide the network size correspondingly. For example, an application with 16 tasks is mapped on a 4×4 network, an application with 24 tasks is mapped on a 6×4 network, and so on which are also shown in Table III. After network size for each application is decided, each task is mapped to one processor in the network using both random and near-optimal NMAP-based mapping techniques. In this section, we present the results with random mapping; results with NMAP mapping are analyzed in Section IV-D.

For evaluation of these embedded applications, we fix the injection rates of all tasks, then application running latency is measured after a total of one million packets are successfully transferred. For each application, we assume that the most busy task spends 50% times for execution and 50% times for communication which means it aggressively executes one cycle and then sends one output to the downstream task in next cycle, repeatedly. With this assumption, the most busy task in each application has an injection rate of 0.5; the injection rates of other tasks are computed according to the their required bandwidth given in the graph using the method described above.

For clear comparison, we normalize the latency of each application running on different routers to the latency when running on the typical VC router which are shown in Fig. 14. As shown, RoShaQ has lower latency than both VC routers in all ten applications. On average, RoShaQ has 26% and 12% less latency than VC and VC-fullXbar routers, respectively.

C. Power, Area, and Energy

Three router models (VC4, VC4-fullXbar, and RoShaQ15) in Verilog RTL are synthesized targeting a 65 nm low-power CMOS standard-cell process 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,

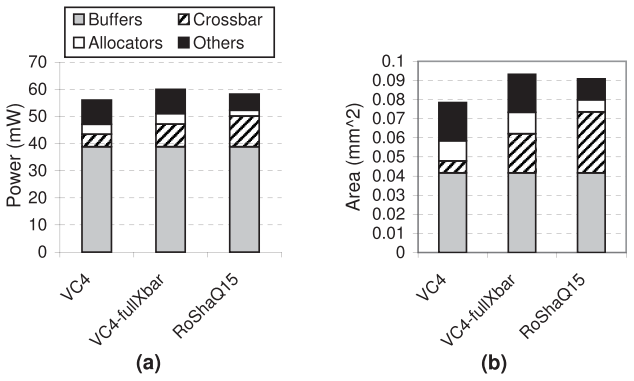

Fig. 15. Synthesis results. (a) Power. (b) Area.

TABLE IV
ROUTER POWER AT 1.2 V, 1 GHZ AND AREA COMPARISON

	VC4	VC4-fullXbar	vs. VC4	RoShaQ15	vs. VC4	vs. VC4-fullXbar
Power (mW)	56.08	60.05	+ 7%	58.26	+ 4%	-3%
Area (mm^2)	0.078	0.093	+19%	0.091	+16%	-3%

25 °C. We let the synthesis tool do all optimization steps and automatically pick standard cells in the library in order for all routers to meet 1 GHz clock frequency in the worst case.

For accurate evaluation and comparison, postlayout data are highly desired. Unfortunately, our limited EDA software infrastructure prevents us to have post-layout data for comparison. However, for fair comparison using synthesis data, we compare router designs in a relative basic by normalizing all data to the VC router data. The comparison fairness is achieved by conducting the experiments on large number of different synthetic traffic patterns and real multitask application traces based on cycle-accurate simulations.

The 100%-active synthesis power and area of three routers are shown in Fig. 15. Other circuits in this figure include state of queues, routing computation, and credit calculating circuitry. For considering the pipelined architecture of routers, the reported power and area of all components are also included in their output pipeline registers. As seen in this figure, in the typical router VC4, buffers are expensive that occupy 54% area and consume 70% 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 as listed in Table IV.

Because RoShaQ15 has two crossbars, its crossbars are 56% larger and consume 35% higher power than the VC4-fullXbar's crossbar. However, due to the simplicity of its allocators' circuits and fewer routing computation blocks (5 for 5 input queues compared with 20 for 20 VCs in VC routers), [4] the total router area and power of RoShaQ15 is 3% less than VC4-fullXbar router. Compared with VC4, RoShaQ is 4% and 16% larger power and area.

[4]Because LRC routing is done at the same cycle with SA or VCA, each input queue must be equipped with a routing circuit block. VC routers have multiple buffer queues per input port; therefore, they have larger routing circuitry area and power than RoShaQ.

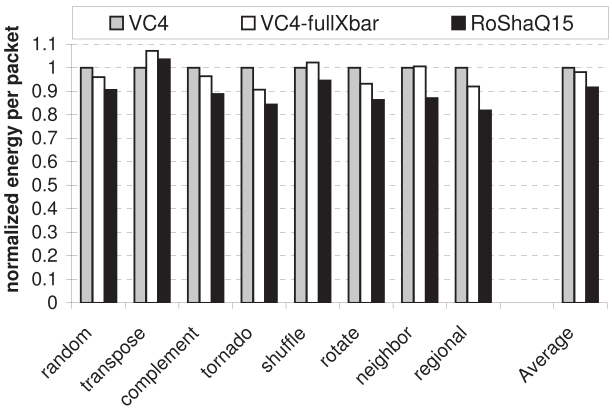

Fig. 16. Normalized energy per packet over synthetic traffic patterns.

Another metric to compare among router designs is the energy that routers in the network dissipate for transferring data packets over a traffic pattern [39]. The power consumption of a circuit is formulated by $P = \alpha C V^2 f$ where α is the circuit switching activity factor, C is the circuit capacitance (which is proportional to the circuit area), V is the supply voltage and f is the clock frequency. Therefore, at the same supply voltage and clock frequency a circuit with low activity would consume low power even it has large capacitance (which causes high 100%-active power). Let P_i be 100%-active synthesis power of component i (queues, crossbar, routing computation, arbiters, states, and credit calculation) of a router; let n_{ri} be the number of cycles in which the component i of router r is active in the whole simulation time, then the total energy router r consumes is

$$E_r = \sum_{\text{all } i} n_{ri} P_i T_{\text{clk}} \tag{1}$$

where T_{clk} is the clock period.

Therefore, the average packet energy spent on each router in the network is given by the following:

$$E_p = \frac{1}{N_p N_r} \sum_{\text{all } r} E_r = \frac{T_{\text{clk}}}{N_p N_r} \sum_{\text{all } r} \sum_{\text{all } i} n_{ri} P_i \tag{2}$$

where N_r is the number of routers in the network and N_p is the total number of consumed packets in the whole simulation time.

For each synthetic traffic pattern, we choose the injection rates at which networks have the same packet latency of 100 cycles that is where the networks start reaching saturation. For each simulation, we run 100 000 cycles, then router activity information and the number of received packets are collected after 20 000 warmup cycles. Applied these statistic information into (2) with T_{clk} of 1 ns (1 GHz clock rate) and N_r of 64 (8 ×8 network), the normalized energy per packet of each router over eight synthetic traffic patterns is shown in Fig. 16.

As shown in the figure, although VC-fullXbar has higher throughput than VC router, it consumes more energy over three traffic patterns transpose, bit-shuffle and neighbor. This is

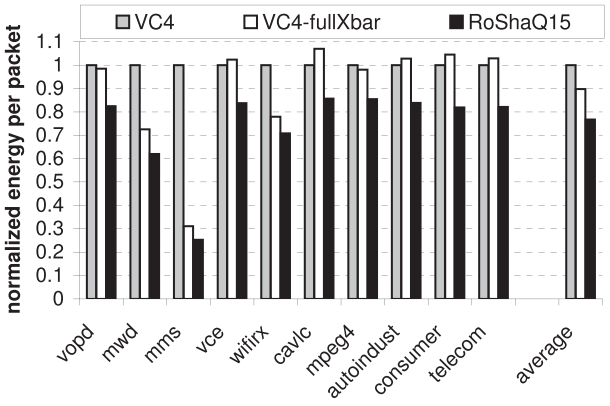

Fig. 17. Normalized energy per packet over real application traffic patterns using task-to-core random mapping.

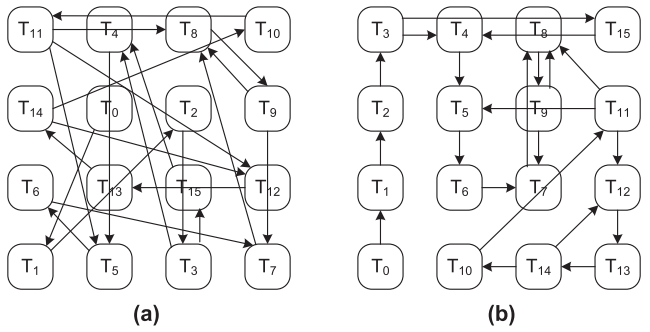

Fig. 18. Mapping results of VOPD application to a 4 × 4 platform using (a) random mapping and (b) NMAP algorithm.

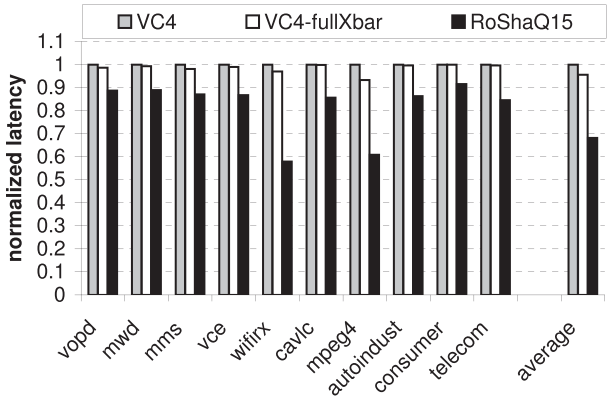

Fig. 19. Normalized latency of real application traffic patterns with task-to-core mapping using NMAP algorithm.

because it has higher active power than VC routers as listed in Table IV. However, averaged over all eight traffic patterns, VC-fullXbar router has 2% lower energy per packet than VC router. RoShaQ has lower energy than VC router over seven traffic patterns, except transpose because they have the same throughput over this pattern but RoShaQ has higher active power. RoShaQ consumes lower energy than VC-fullXbar routers over all traffic patterns. Averaging from all eight traffic patterns, RoShaQ15 consumes 9% and 7% lower energy per packet than VC4 and VC4-fullXbar, respectively.

For each real application, we set the injection rate for the task with largest required bandwidth to 0.5 flits/cycle, and then derive the injection rates for other tasks using the method presented in Section IV-B2. In this experiment, we map tasks of the application to the core array randomly (near-optimal mapping experimental results are presented in next section). Statistic activity information of each router and the total simulation cycles are collected after one million packets are received. Again, these information are applied into (2) for evaluating packet energy of each router.

The normalized energy per packet each router consumes over ten real applications is shown in Fig. 17. As shown, VC-fullXbar has higher energy than VC router over five applications *vce*, cavlc, autoindust, consumer and telecom; while RoShaQ consumes lower energy than both VC and VC-fullXbar routers over all applications. Averaged over all ten applications, RoShaQ has 23% and 14% lower energy per packet than VC and VC-fullXbar routers, respectively.

D. On the Effect of Application Mapping

Given a many-core platform, a good application mapping algorithm can significantly improve the data communication time and network energy consumption [30]. For example, the mapping results of the same 16-task VOPD application (shown in Fig. 13) to a 4 × 4 network using random and NMAP [40] algorithms are shown in Fig. 18. As shown, NMAP algorithm gives a higher-desired mapping result than the random one by allowing mapping tasks with high-bandwidth links near together, hence reduce not only the time packets traveling but also packet contention on the network. These reductions clearly help to achieve better overall application performance and energy consumption. We can find in the literature several other mapping algorithms which could lead to the same effect, such as CMAP [41], A3MAP [42], and UNISM [43].

In this section, for more practical than random mapping, we present the comparison results of router architectures over real multitask application traffic traces using NMAP algorithm.

The normalized latency using NMAP algorithm of each application running on different routers to the latency when running on the typical VC router is shown in Fig. 19. The NMAP algorithm reduces the communication distance and congestion of tasks, hence improves the latency for all applications. As shown, for all applications, the latency while running on VC-fullXbar and VC routers are almost the same. This is because VC and VC-fullXbar routers have the same internal router latency; hence under low-congested network traffic, their contributions to the overall application latency are not much different. On average over all ten applications, VC-fullXbar has only 4% less latency than VC router.

On the other hand, for low-congested traffic packets often bypass the shared-queues of RoShaQ hence achieve less internal router latency. Therefore, under NMAP mapping, RoShaQ achieves much better application latency reduction than VC routers. Averaged over ten applications, RoShaQ has 32% and 28% less application latency compared with VC and VC-fullXbar routers, respectively.

With NMAP mapping, while the VC-fullXbar latency is only slightly less, its higher active power makes it consume higher energy than VC router as shown in Fig 20.

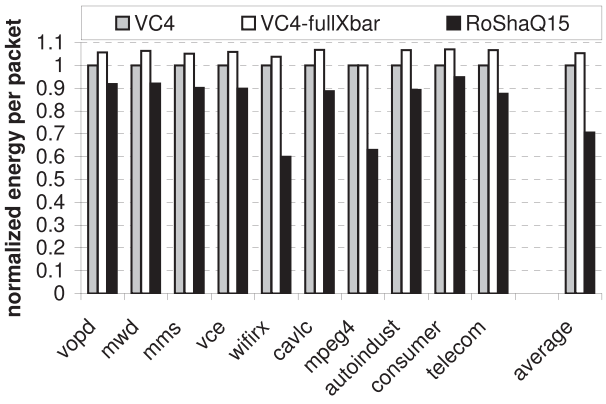

Fig. 20. Normalized energy per packet over real application traffic patterns with task-to-core mapping using NMAP algorithm.

VC-fullXbar now has higher energy than VC router over all applications which results in 5% greater packet energy on average over ten applications. On the other side, because RoShaQ achieves much better latency reduction with NMAP mapping, it also consumes less communication energy. On average, RoShaQ consumes 30% and 33% less energy per packet than VC and VC-fullXbar routers, respectively.

V. Related Work

Peh *et al.* [22] and Mullins *et al.* [44] proposed speculative techniques for VC routers allowing a packet to simultaneously arbitrate for both VCA and SA giving a higher priority for nonspeculative packets to win SA; therefore reducing zero-load latency in which the probability of failed speculation is small. 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 nonspeculative 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 rearbitration times.

Increasing crossbar input ports, that allows directly connecting to all VCs 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.* [45]. Recently, Passas *et al.* [46] designed a 128 $\times$ 128 crossbar allowing connecting 128 tiles while occupying only 6% of their total area. 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 multibank 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 [47]. Therefore, 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 reusing the existing generic simple queues reducing practical design and test costs.

Latif *et al.* [10] implemented a router with input ports sharing all queues that is similar to the architecture shown in Fig. 6(a). Its implementation on FPGA shows more power and area-efficient than typical input VC routers. A similar approach is proposed by Tran *et al.* [20]; due to the high complexity of its allocators and also interrouter round-trip request/grant signaling, however, its performance is actually poorer than a typical router.

Ramanujam *et al.* [11] recently proposed a router architecture with shared-queues named DSB which emulates an output-buffered router. This router is similar to one shown in Fig. 6(b) that has higher zero-load latency than a VC router. This is because a packet must travel through both two crossbars and be buffered in both input and shared queues at each router even without network congestion. In addition to that, the timestamp-based flow control of double sideband (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 with VC routers. RoShaQ also achieves much higher saturation throughput than VC routers, with only small area and power overheads while consuming lower average energy per packet.

Nicopoulos *et al.* [48] proposed ViChar, a router architecture which allows packets to share flit slots inside buffer queue so that can achieve higher throughput. Our paper manages buffers at coarser grain that is at queue-level rather than at flit-level, hence allows reusing existing generic queue design which makes buffer and router design much simpler and straightforward. ViChar's idea, however, is orthogonal with our research and can be applied to RoShaQ forming a router with fined-grain shared buffers which could improve more network performance.

The majority of state-of-the-art on-chip router designs utilize input queuing buffers; we, however, can find in the literature a few output queuing router architectures [24], [49], [50]. If looking into the whole network picture, buffers at an output router port should act the same as input buffers of its downstream router. Therefore, without some special techniques for speeding up the internal router circuits, output buffering routers offer no advantage in term of network performance but complicate the flow control circuit designs. The output buffering was shown to achieve higher throughput than input buffering if the router is clocked at P times higher the link's clock frequency or if the output buffers have P write ports each where P is the number of router ports [11], [51]. These speedup techniques are not trivial and are expected to dissipate much more power. RoShaQ can dynamically adapt to use the bypass paths or the shared queues depending on network load hence, achieves both low latency as input

buffering routers and high throughput as output buffering ones without the needs of internal router speedup.

VI. Conclusion

We presented RoShaQ, a novel router architecture that allowed sharing multiple buffer queues for improving network throughput. Input packets also can bypass the shared queues to achieve low latency in the case that the network load was low. Compared with a typical VC router, while having the same buffer space, over synthetic traffic patterns it had 17% lower zero-load latency and 18% higher saturation throughput on average with only 4% higher power and 16% larger area. It had also 5% higher throughput than a full-crossbar VC router with 3% lower power and 3% less area. While targeting the same average packet latency of 100 cycles where all the routers start saturating, RoShaQ had 9% and 7% lower energy dissipated per packet than typical VC and full-crossbar VC routers, respectively.

We also presented a method for evaluating and comparing performance and energy-efficiency of routers over real multitask embedded applications. Over these applications with random mapping, RoShaQ had 26% and 12% lower latency than typical VC and full-crossbar VC routers, respectively, while targeting the same intertask communication bandwidth requirements. In term of energy, RoShaQ consumes 23% and 14% lower energy per packet than typical VC and full-crossbar VC routers, respectively. Using NMAP mapping algorithm, RoShaQ achieved higher improvement in application performance and energy consumption than other routers. On average, RoShaQ had 32% and 28% less application latency and consumed 30% and 33% less energy per packet than VC and full-crossbar VC routers, respectively.

References

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

[2] S. Borkar, "Thousand core chips: A technology perspective," in *Proc. 44th DAC*, Jun. 2007, pp. 746–749.

[3] C. H. V. Berkel, "Multi-core for mobile phones," in *Proc. DATE*, 2009, pp. 1260–1265.

[4] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. DAC*, 2001, pp. 684–689.

[5] W. J. Dally, "Virtual-channel flow control," *IEEE Trans. Parallel Distrib. Syst.*, vol. 3, no. 2, pp. 194–205, Mar. 1992.

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

[7] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," in *Proc. ISCA*, Jun. 2009, pp. 196–207.

[8] M. Hayenga, N. E. Jerger, and M. Lipasti, "SCARAB: A single cycle adaptive routing and bufferless network," in *Proc. 42nd Ann. IEEE/ACM Int. Symp. Microarchitect.*, Dec. 2009, pp. 244–254.

[9] G. Michelogiannakis, D. Sanchez, W. J. Dally, and C. Kozyrakis, "Evaluating bufferless flow control for on-chip networks," in *Proc. 4th NOCS*, 2010, pp. 9–16.

[10] K. Latif, T. Seceleanu, and H. Tenhunen, "Power and area efficient design of network-on-chip router through utilization of idle buffers," in *Proc. 17th IEEE Int. Conf. Workshops ECBS*, Mar. 2010, pp. 131–138.

[11] R. S. Ramanujam, V. Soteriou, B. Lin, and L.-S. Peh, "Extending the effective throughput of NoCs with distributed shared-buffer routers," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 4, pp. 548–561, Apr. 2011.

[12] T. Nakamura, H. Matsutani, M. Koibuchi, K. Usami, and H. Amano, "Fine-grained power control using a multi-voltage variable pipeline router," in *Proc. Int. Symp. Embedded Multicore Socs*, Sep. 2012, pp. 59–66.

[13] H. Matsutani, Y. Hirata, M. Koibuchi, K. Usami, H. Nakamura, and H. Amano, "A multi-Vdd dynamic variable-pipeline on-chip router for CMPs," in *Proc. 17th ASP DAC*, Feb. 2012, pp. 407–412.

[14] L. Shang, L.-S. Peh, and N. K. Jha, "Dynamic voltage scaling with links for power optimization of interconnection networks," in *Proc. Int. Symp. HPCA*, Feb. 2003, pp. 91–102.

[15] A. Bianco, P. Giaccone, G. Masera, and M. Ricca, "Power control for crossbar-based input-queued switches," *IEEE Trans. Comput.*, vol. 62, no. 1, pp. 74–82, Jan. 2013.

[16] A. T. Tran, D. N. Truong, and B. M. Baas, "A GALS many-core heterogeneous DSP platform with source-synchronous on-chip interconnection network," in *Proc. ACM/IEEE Int. NOCS*, May 2009, pp. 214–223.

[17] E. Beigne, "An asynchronous power aware and adaptive NoC based circuit," *IEEE J. Solid-State Circuits*, vol. 44, no. 4, pp. 1167–1177, Apr. 2009.

[18] A. T. Tran and B. M. Baas, "RoShaQ: High-performance on-chip router with shared queues," in *Proc. ICCD*, Oct. 2011, pp. 232–238.

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

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

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

[22] L. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Proc. Int. Symp. HPCA*, Jan. 2001, pp. 255–266.

[23] W. J. Dally and C. L. Seitz, "Deadlock-free message routing in multiprocessor interconnection networks," *IEEE Trans. Comput.*, vol. 36, no. 5, pp. 547–553, May 1987.

[24] M. G. Hluchyj and M. J. Karol, "Queueing in high-performance packet switching," *IEEE J. Sel. Areas Commun.*, vol. 6, no. 9, pp. 1587–1597, Dec. 1988.

[25] A. T. Tran and B. M. Baas, "NoCTweak: A highly parameterizable simulator for early exploration of performance and energy efficiency of networks on-chip," Dept. Electr. Comput. Eng., Univ. California, Davis, CA, USA, Tech. Rep. ECE-VCL-2012-2, Jul. 2012.

[26] N. Enright-Jerger and L. Peh, *On-Chip Networks*. San Rafael, CA, USA: Morgan & Claypool, 2009.

[27] A. Kumar, L.-S. Peh, P. Kundu, and N. K. Jha, "Towards ideal on-chip communication using express virtual channels," *IEEE Micro*, vol. 28, no. 1, pp. 80–90, Jan. 2008.

[28] A. T. Tran, D. N. Truong, and B. M. Baas, "A reconfigurable source-synchronous on-chip network for GALS many-core platforms," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 29, no. 6, pp. 897–910, Jun. 2010.

[29] E. B. Van der Tol and E. G. Jaspers, "Mapping of MPEG-4 decoding on a flexible architecture platform," *Proc. SPIE*, vol. 4674, pp. 1–13, Dec. 2001.

[30] J. Hu and R. Marculescu, "Energy-and performance-aware mapping for regular NoC architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 24, no. 4, pp. 551–562, Apr. 2005.

[31] Y.-C. Lan, H.-A. Lin, S.-H. Lo, Y. H. Hu, and S.-J. Chen, "A bidirectional NoC (BiNoC) architecture with dynamic self-reconfigurable channel," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 3, pp. 427–440, Mar. 2011.

[32] M. Palesi, R. Holsmark, S. Kumar, and V. Catania, "Application specific routing algorithms for low power network on chip design," in *Low Power Networks on Chip*. New York, NY, USA: Springer-Verlag, 2011, pp. 113–150.

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

[34] A. T. Tran, D. N. Truong, and B. M. Baas,"A complete real-time 802.11a baseband receiver implemented on an array of programmable processors," in *Proc. ACSSC*, Oct. 2008, pp. 165–170.

[35] Z. Xiao and B. M. Baas, "A high-performance parallel CAVLC encoder on a fine-grained many-core system," in *Proc. ICCD*, Oct. 2008, pp. 248–254.

[36] D. Gebhardt, J. You, and K. S. Stevens, "Design of an energy-efficient asynchronous NoC and its optimization tools for heterogeneous SoCs," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 30, no. 9, pp. 1387–1399, Sep. 2011.

[37] K. Latif, "PVS-NoC: Partial virtual channel sharing NoC architecture," in *Proc. 19th Euromicro Int. Conf. PDP*, Feb. 2011, pp. 470–477.
[38] E3S. (2013). *Embedded System Synthesis Benchmarks Suite*, New York, NY, USA [Online]. Available: http://ziyang.eecs.umich.edu/~dickrp/e3s/
[39] A. Banerjee, P. T. Wolkotte, R. D. Mullins, S. W. Moore, and G. J. M. Smit, "An energy and performance exploration of network-on-chip architectures," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 3, pp. 319–329, Mar. 2009.
[40] S. Murali and G. De Micheli, "Bandwidth-constrained mapping of cores onto NoC architectures," in *Proc. Conf. DATE*, Feb. 2004, pp. 896–901.
[41] R. Tornero, J. M. Orduña, M. Palesi, and J. Duato, "A communication-aware topological mapping technique for NoCs," in *Proc. Int. Euro Par Conf. Parallel Process.*, Aug. 2008, pp. 910–919.
[42] W. Jang and D. Z. Pan, "A3MAP: Architecture-aware analytic mapping for networks-on-chip," *ACM Trans. Design Autom. Electron. Syst.*, vol. 17, no. 3, pp. 1–22, Jul. 2012.
[43] O. He, S. Dong, W. Jang, J. Bian, and D. Z. Pan, "UNISM: Unified scheduling and mapping for general networks on chip," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 8, pp. 1496–1509, Aug. 2012.
[44] R. Mullins, A. West, and S. Moore, "Low-latency virtual-channel routers for on-chip networks," in *Proc. 31st ISCA*, Mar. 2004, p. 188.
[45] G. De Micheli, C. Seiculescu, S. Murali, L. Benini, F. Angiolini, and A. Pullini, "Networks on chips: From research to products," in *Proc. 47th ACM/IEEE DAC*, Jun. 2010, pp. 300–305.
[46] G. Passas, M. Katevenis, and D. Pnevmatikatos, "A 128 × 128 × 24 Gb/s crossbar interconnecting 128 tiles in a single hop and occupying 6% of their area," in *Proc. NOCS*, 2010, pp. 87–95.
[47] C. B. Stunkel, J. Herring, B. Abali, and R. Sivaram, "A new switch chip for IBM RS/6000 SP systems," in *Proc. ACM/IEEE CS*, Nov. 1999, pp. 1–16.
[48] C. A. Nicopoulos, D. Park, J. Kim, N. Vijaykrishnan, M. S. Yousif, and C. R. Das, "ViChaR: A dynamic virtual channel regulator for network-on-chip routers," in *Proc. IEEE/ACM MICRO*, Dec. 2006, pp. 333–346.
[49] H. Elmiligi, M. W. El-Kharashi, and F. Gebali, "Modeling and implementation of an output-queuing router for networks-on-chips," in *Proc. ICESS*, May 2007, pp. 241–248.
[50] S. Iyer, R. Zhang, and N. Mckeown, "Routers with a single stage of buffering," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun.*, Oct. 2002, pp. 251–264.
[51] A. Prakash, "Randomized parallel schedulers for switch-memory-switch routers: Analysis and numerical studies," in *Proc. IEEE INFOCOM*, vol. 3. Mar. 2004, pp. 2026–2037.

Anh T. Tran (S'07–M'12) received the B.S. degree (Hons.) in electronics engineering from the Posts and Telecommunications Institute of Technology, Hanoi, Vietnam, in 2003, and the M.S. and Ph.D. degrees in electrical engineering from the University of California, Davis, CA, USA, in 2009 and 2012, respectively.

He is currently a Senior ASIC Research Engineer with Xpliant, San Jose, CA, USA, where he is participating in the development of software defined network ICs for the next generation of networking routers targeting enterprises and data centers. His current research interests include VLSI designs, multicore architectures, on-chip interconnects, and reconfigurable systems on chip.

Dr. Tran was a recipient of the Best Paper Award at ICCD in 2011 for his work on high performance on-chip router designs. He was a VEF Fellow from 2006 to 2012.

Bevan M. Baas (M'99–SM'12) received the B.S. degree in electronic engineering from California Polytechnic State University, San Luis Obispo, CA, USA, in 1987, and the M.S. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 1990 and 1999, respectively.

He was with Hewlett-Packard, Cupertino, CA, USA, from 1987 to 1989, where he participated in the development of the processor for a high-end minicomputer. In 1999, he joined Atheros Communications, Santa Clara, CA, USA, and served as a Core Member of the team which developed the first IEEE 802.11a (54 Mb/s, 5 GHz) Wi-Fi wireless LAN solution. In 2003, he joined the Department of Electrical and Computer Engineering, University of California at Davis, Davis, CA, USA, where he is currently an Associate Professor. He leads projects in architecture, hardware, software tools, and applications for VLSI computation with an emphasis on DSP workloads. In 2006, he was a Visiting Professor with Intel's Circuit Research Laboratory. His recent projects include the 36-processor asynchronous array of simple processors chip, applications, and tools, a second generation 167-processor chip, low density parity check decoders, FFT processors, viterbi decoders, and H.264 video codecs.

Dr. Baas was the U.S. National Science Foundation Fellow from 1990 to 1993 and a NASA Graduate Student Researcher Fellow from 1993 to 1996. He was a recipient of the U.S. National Science Foundation CAREER Award in 2006, the Most Promising Engineer/Scientist Award by AISES in 2006, the Best Paper Award at ICCD in 2011, and Best Paper nominations at Asilomar in 2011 and BioCAS in 2010. He is an Associate Editor for the IEEE JOURNAL OF SOLID-STATE CIRCUITS and the Guest Editor of the *IEEE Micro* in 2012. He was the Program Committee Co-Chair of HotChips in 2011 and Program Committee Member of Hotchips from 2009 to 2010, ICCD from 2004 to 2005 and 2007 to 2009, ASYNC in 2010, and of the ISSCC SRP Forum in 2012.