DLABS: a Dual-Lane Buffer-Sharing Router Architecture for Networks on Chip

Anh T. Tran and Bevan M. Baas

Department of Electrical and Computer Engineering
University of California - Davis, USA
{*anhtr, bbaas*}*@ucdavis.edu*

***Abstract*—A significant portion of the conventional router's area is dedicated to its buffers at the input/output ports. For regular workloads, however, a large number of buffers are always idle while other buffers are always busy. This observation motivates us to design a new router architecture which allows buffers to be shared by multiple input ports. This architecture keeps buffers busy while working together to forward data, reducing the busy cycle times and pressure on each buffer, resulting in an improvement of the overall network performance. Sharing resources like buffers, however, has the potential of causing deadlock in the network. In this work, we propose a dual-lane architecture that is deadlock-free for our buffer-sharing routers, named DLABS (Dual-Lane Buffer-Sharing) routers. We design three DLABS routers and compare against a conventional wormhole router. Experimental results show the smallest DLABS router occupies an area of only 0.62% of a conventional router, but achieves 108% on the throughput per area (TPA) over regular traffic benchmarks. The largest DLABS router occupies 112% of the circuit area of the conventional router, but achieves 164% on the TPA.**

I. Introduction

As device size is still scaled following Moore's Law that is able to offer billions of transistors on a single chip nowadays, the number of processors/processing elements (PEs) in a general-purpose platform or a system on chip (SoC) increases to mostly take advantage of this huge transistor budget. This increase, in fact, allows to improve their performance while keeping them to stay well below acceptable power consumption levels [1]–[3]. Large number of PEs on chip requires an effective interconnection fabric for transferring data among them. On-chip network was shown to be the most promising interconnect technique compared to a large crossbar or a shared bus [4], [5]. The network, however, can easily become the system bottleneck; and therefore its performance improvement has been a popular research topic. As a result, many innovative router architectures have been proposed in recent years [6]–[9].

In a network on chip (NoC), routers are basic components for transferring data between processing elements (PEs). Fig. 1 depicts an array of 4x4 PEs connected in a 2-D mesh network of routers. Each router has five ports with one reserved for connecting between the router and its PE through a local adapter (A) (sometimes called a network interface (NI)) that converts data from its PE into the packet format that can be understood by routers, and vice versa. The other four ports are connected to nearest-neighbor routers constructing a 2-D mesh network.

Fig. 1 also shows a typical wormhole router datapath where each its input port has a queue buffer to temporarily store flits of incoming packets. The header flit of each packet contains the destination information that allows the router to decide its next output port through an appropriate routing algorithm. The buffer size may be less than size of a packet so that flits of one packet can spread into many consecutive routers like a worm, so the name it is. A typical router normally has a notable portion of its area and power spent on its buffers. The results from a test chip showed that buffers's

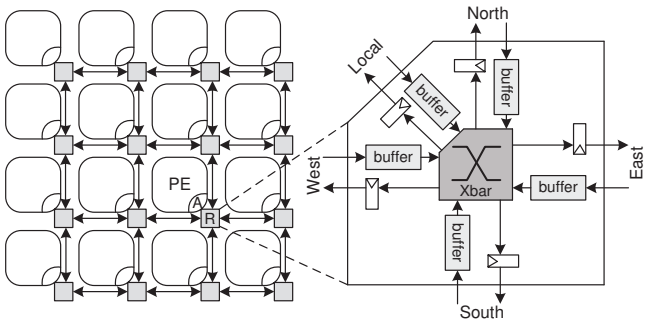

Fig. 1. A SoC of many PEs interconnected by five-port routers in a typical 2-D mesh topology

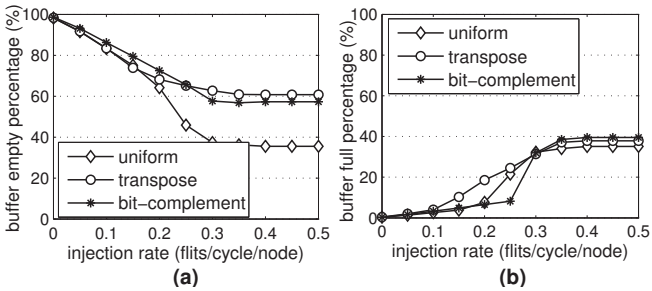

Fig. 2. The average buffer activity of a wormhole router-based NoC over three traffic patterns: uniform random, transpose and bit-complement. (a), (b) are percentages of the number of cycles buffers are empty or full, respectively, in the whole simulation time.

area and power are greater than 60% and 30% of the entire router, respectively [10].

Although much cost is spent on buffers, they are not always utilized efficiently. Fig. 2 shows the buffer activity of all wormhole routers in an array of 8x8 nodes simulated for 30,000 cycles over three traffic patterns: uniform random, transpose and bit-complement. Buffer activity is measured by the number of cycles where buffers are either empty or full during the whole simulation time. Two boundary states of a buffer is "empty," when it has no flits to be forwarded, and "full," when it cannot accept any more flits. When the injected traffic is low the routers are less busy, and therefore, they are almost empty. When the traffic increases, buffers begin to become more busy with increased cycles being full and less cycle times being idle.

The wormhole router utilizes buffers well over irregular patterns such as the uniform random, where a node sends its data to any destination randomly. Therefore, almost all buffers will have a packet to process at some time. Table I shows only 32 or 10% of total buffers that are always empty while running the uniform random benchmark. These 32 always-idle buffers lie on boundary of the network that never receive any packet. These buffers along with buffers that are sometimes empty due to upstream congestion

TABLE I
THE NUMBER OF BUFFERS WHICH ARE ALWAYS EMPTY IN THE WHOLE SIMULATION TIME OF 30,000 CYCLES FOR AN ARRAY OF 8X8 ROUTERS (TOTAL: 320 BUFFERS)

Traffic	uniform	transpose	bit-complement
always empty buffers	32	152	144
ratio	10.0%	47.5%	45.0%

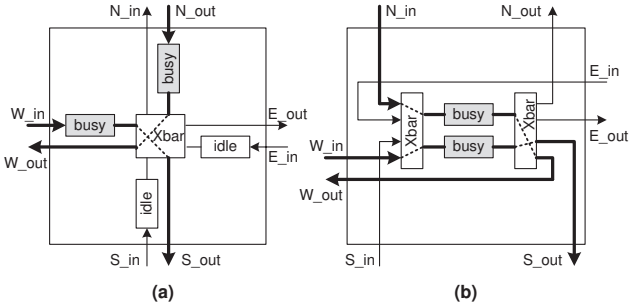

Fig. 3. Illustration of our motivation for designing a router with shared buffers. (a) For some kinds of traffic patterns buffers of many routers in the network are idle for a large fraction of time. (b) Using only two buffers per router while achieving a similar performance.

averages to approximately 35% of total simulation time the network have idling buffers at the saturation status as shown in Fig. 2(a).

Now let us consider two other traffic patterns: transpose and bit-complement. When the network is in the saturation status, the network has a large number of buffers that are busy with around 40% of time being full while there are also a large number of buffers that are idle with around 60% of the total simulation time being empty. Some buffers never even receive any packet. Table I, shows the statistics of always empty buffers at 47.5% and 45.0% for transpose and bit-complement, respectively (in a total of 320 buffers for a network with 8x8 routers). This large percentage of empty buffers essentially shows that more than half of the buffers have to carry all of the work of the whole network, which makes the performance of the network poor for such traffic patterns.

In order to efficiently use the buffer space, especially for regular traffic patterns, we propose a new router architecture that allows buffers the ability to be shared by multiple input ports instead of dedicating a single buffer per input port. Fig. 3 illustrates our idea. When a router has only two packet flows coming from the West and North input ports, and going to the South and West output ports, respectively, the wormhole router as shown in Fig. 3(a) has two idle buffers at the East and South input ports. We can eliminate these wasted buffers by using an architecture shown in Fig. 3(b) which shows a router using only two shared buffers while achieving the same performance. This sharing of buffer resources, however, can easily incur deadlocks and must be dealt with. Given these aforementioned issues, the main contributions of this work are:

- Designing a buffer-sharing router architecture that allows to more efficiently use buffers.
- Resolving the deadlock problem in a network using buffer-sharing routers.
- Exploiting the design space for improving the performance of buffer-sharing router given the same number of buffers as a conventional wormhole router.

The paper is organized as follows: Section II describes the potential of incurring deadlock in a network of buffer-sharing routers; and then proposes solutions for avoiding deadlocks that are the bases for our

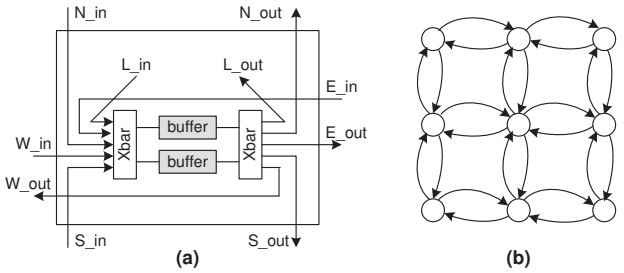

Fig. 4. (a) A router with all input ports sharing two buffers. (b) The resource dependency graph representative of a 3x3 array of routers.

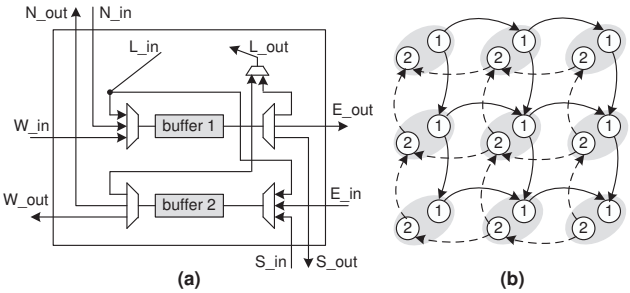

Fig. 5. Deadlock is avoided by separating buffers on two lanes: (a) dual-lane router architecture; (b) the corresponding resource dependence graph of an array of 3x3 routers.

deadlock-free dual-lane router architectures. Section III presents two techniques for enhancing performance of our buffer-sharing routers given the same number of buffers as a typical wormhole router. The experimental results of these routers are analyzed and compared against a typical wormhole router are shown in Section IV. Finally, Section V concludes this paper.

II. DLABS ROUTER ARCHITECTURE

A. Buffer Sharing and Deadlock Problem

Initially, we adopt the idea of the buffer-sharing router illustrated in Fig. 3(b) and redrawn in Fig. 4(a) with the addition of input and output links, from and to the local PE. Note that this router can cause deadlocks in a network. For example, when packets in both buffers request to go to the *E_out* port at the same time that packets from *E_in* request to go to *W_out*. Thus, the buffers of two routers (connected along the *E_out* of one router) are busy, and as a result no packet will get granted for forwarding.

The deadlock problem can be easily found through the resource dependency graph (RDG) representation of the network. A well-known theory developed by Dally *et al.* and also by Duato claims that a network is deadlock-free *if and only if* there is no cycle in the RDG of that network [11], [12]. For checking deadlock in our network of buffer-sharing routers, the shared buffers inside each router are depicted by a circle with links connecting the routers drawn as directional arrows in the RDG. Fig. 4(b) shows the RDG of a network of 3x3 routers given in Fig. 4(a). Loops in this RDG clearly show the potential of deadlock in the network. Solutions for resolving this deadlock problem are detailed in the following subsections that will be the bases for our dual-lane router architectures.

B. DLABS – Dual-Lane Buffer-Sharing Routers

The deadlock potential can be avoided by breaking all loops in the RDG. A loop appears in Fig. 4(b) because the shared buffers of near neighbor routers are connected on both unidirectional links over one I/O port. We eliminate this by making an input data link go to a buffer

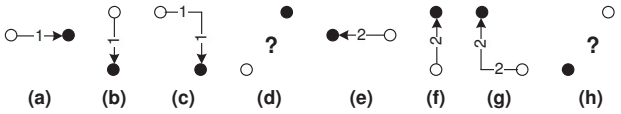

Fig. 6. Eight location patterns of source and destination pairs

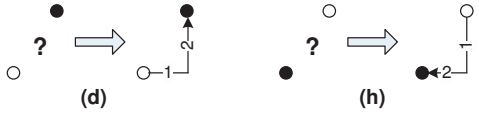

Fig. 7. The patterns (d) and (h) of source and destination pairs in Fig. 6 now can be routed on Lane 1 first from the source then turning on Lane 2 to reach its destination.

that differs from the buffer connecting to the output link of the same port. Fig. 5(a) illustrates this idea. In this router, Buffer 1 is shared by two input links, *W_in* and *N_in*, and sends output packets to *E_out* and *S_out*, respectively. Similarly, Buffer 2 is shared by two input links, *E_in* and *S_in*, and sends output packets to *W_out* and *N_out*, respectively. Each buffer is shared by more than one input link, and so an arbiter is needed on each buffer to determine which upstream router will win to send a packet if they have packets wanting to be forwarded simultaneously. Because the two buffers of this router are on separate lanes, we call this a dual-lane router.

Fig. 5(b) shows the RDG of an array of 3x3 dual-lane routers. Buffers 1 and 2 are depicted as circles 1 and 2, respectively. The links interconnecting Buffer 1 and Buffer 2 create two separate planes. On plane 1, each node's input channels are from the West and North, and its output channels go to the East and South. Conversely, on the plane 2, each node's input channels are from the East and South, and its output channels go to the West and North. Thus, there is no loop existing on each plane and the corresponding network is deadlock-free.

Unfortunately this dual-link router network cannot transfer packets between arbitrary pairs of source and destination nodes. Fig. 6 depicts all eight possible location patterns of source and destination pairs. For each pair, the white circle represents a source node while the black one represents a destination node. As shown in the figure, packets of pairs (a), (b) and (c) can be routed on Lane 1 of all intermediate routers; while pairs (e), (f) and (g) would be routed on Lane 2. However, if a pair of source and destination nodes falls under case (d) or (h), there is no way packets can be transferred.

In order to make these cases work, we must somehow allow a packet on Lane 1 to transfer to Lane 2 or vice versa. To avoid loops appearing in the RDG, just one direction is allowed, e.g. if we allow Lane 1 packets to transfer to Lane 2 then Lane 2 cannot send its packets to Lane 1. Since the two lanes of the router are similar, we chose to implement a router architecture that allows Lane 1 to transfer packets to Lane 2 only. With this implementation, packets of source and destination pairs in cases (d) and (h) of Fig. 6 now can be routed on Lane 1 at the source (PE), and later transferred to Lane 2 at an intermediate node for it to be advanced to the destination as depicted in Fig. 7.

One method of implementation for moving packets into Lane 2 at the output of Buffer 1 is shown in Fig. 8(a). This "turn link" is emphasized by a bold blue line. Fig. 8(b) presents the RDG of an array of these 3x3 dual-lane routers. This router requires that the source (PE) choose between Lane 1 and Lane 2 depending on its destination, and therefore a buffer is dedicated to the local port for this purpose. In this graph, each node has a unidirectional link connecting Buffer 1 to Buffer 2, and note that the RDG has no loop, i.e. the network is deadlock-free.

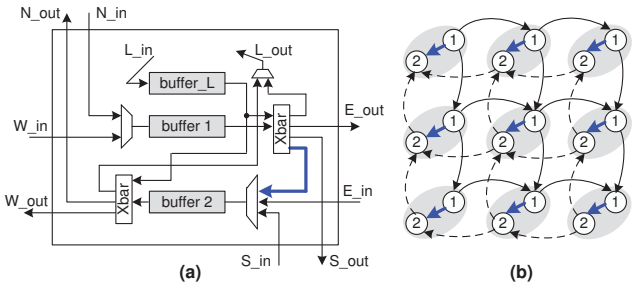

Fig. 8. Dual-lane router architecture with an output link of buffer 1 turned to buffer 2: (a) router datapath; (b) the corresponding resource dependence graph of a 3x3 array of routers.

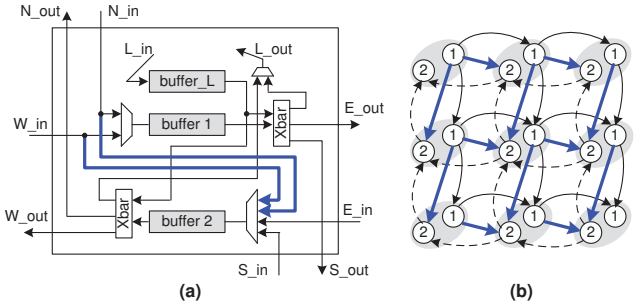

Fig. 9. Dual-lane router architecture with an input links bypassed from Lane 1 to Lane 2 (named DLABS_1+1): (a) DLABS_1+1 router datapath; (b) the corresponding resource dependence graph of an array of 3x3 routers.

Although this router technique works without deadlocks, it has poor performance. This is clearly seen when a packet wants to turn to Lane 2, it must be passed through Buffer 1 first before going to Buffer 2. Furthermore, this packet blocks other packets from other input links until it is forwarded to Buffer 2. To speed up network performance, we propose a router architecture as shown in Fig. 9(a). Instead of going to Buffer 1 before forwarded to Lane 2 the packets go directly to Buffer 2, and so does not disturb packets at other input links wanting to go to Buffer 1. This requires the upstream router be given the ability to forward packets to either Buffer 1 or Buffer 2 of a downstream router depending on their destination information. We name this router architecture DLABS_1+1 because it is a dual-lane router with one shared buffer on each lane. Fig. 9(b) shows the RDG formed from a 3x3 network of DLABS_1+1 routers confirming its deadlock-free nature.

III. Performance Enhancement

We present two solutions for enhancing the performance of our buffer-sharing router while restricting the number of physical buffers to five as in a typical wormhole router.

A. Sharing Multiple Buffers Per Lane

It is well-known that using deeper buffers for a wormhole router does not improve much the overall throughput due to the presence of head-of-line blocking [13]. A more effective solution is by using the virtual-channel technique that allows a physical input link to access multiple buffers, each serving as a virtual-channel [6], [7]. We adopt this idea through the use of multiple buffers for each lane in our dual-lane routers.

Sharing multiple buffers reduces the probability of head-of-line blocking at each lane and hence improve performance [14]. A DLABS router that allows multiple input ports to simultaneously share multiple buffers shows a high level of flexibility. For instance, if there is only one input port having packets to forward, it can access

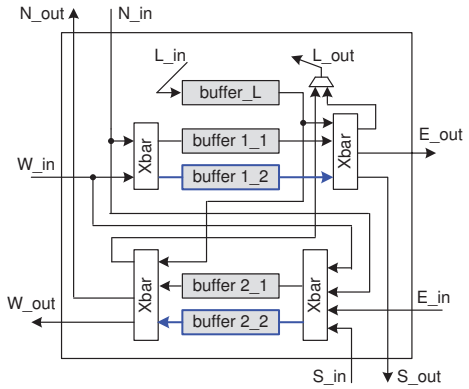

Fig. 10. Dual-lane router architecture with two buffers shared on each lane (named DLABS_2+2)

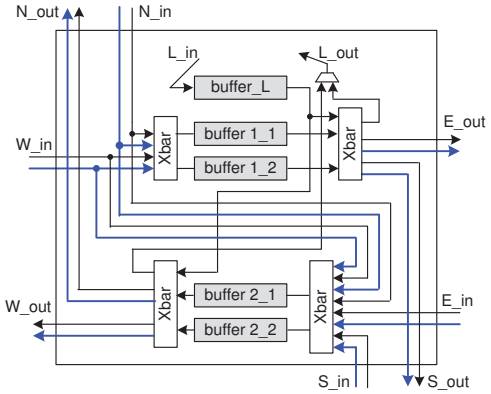

Fig. 11. Dual-lane router architecture with two shared buffers and two interconnect links on each lane (name DLABS_2+2_duallink)

all these shared buffers, so in this fashion it acts as a typical virtual-channel router. Moreover, if there are multiple input port accesses to the shared buffers, each input port can be granted a buffer among these buffers allowing them to forward packets in parallel. Fig. 10 shows such a DLABS router, DLABS_2+2, which has two buffers shared on each lane such that there are five physical buffers in total. This implementation requires four crossbars, although these crossbars are simpler and smaller than a 5x5 crossbar in a typical wormhole router. Furthermore, because input packets on each lane can now access two buffers simultaneously, the buffer arbiters now are replaced by separable buffer allocators that allow multiple input packets to receive grants simultaneously to different buffers [4], [15]. Clearly, adding of these both crossbars and allocators cost more hardware area, but they are negligible in comparison with the hardware cost used for buffers as we will show in Section IV.

B. Multiple Inter-Router Interconnect Links

While the DLABS_2+2 router allows for more efficient use of shared buffers, the output links of each lane will become bottlenecks when their corresponding shared buffers request to go out of the same output port. To alleviate the problem we propose to add one additional unidirectional link at each output port of the router as shown in Fig. 11, which is named DLABS_2+2_duallink. Each buffer on a lane can now route its packets on separate output links, so they can forward packets in parallel improving both latency and throughput of the network. This implementation is assumed that the wiring resource on chip is cheap by the fact that we can use multiple metal layers for routing wires as needed. Furthermore, the 2-D mesh network architecture with routers interconnected with only their nearest neighbors allows to easily route wires in layout.

TABLE II
FOUR ROUTER ARCHITECTURES USED IN OUR EXPERIMENTS

Architecture	Typical wormhole	DLABS_1+1	DLABS_2+2	DLABS_2+2 _duallink
Total buffers	5	3	5	5
Buffer depth (flits)	8	8	8	8
I/O links per port	2	2	2	4

IV. Experimental Results

A. Experimental Setup

We implemented all four router architectures: typical wormhole, DLABS_1+1, DLABS_2+2 and DLABS_2+2_duallink in cycle-accurate Verilog RTL models. All routers have three-pipeline stages with dimensional-order routing and on/off flow-control mechanism [15]. All Verilog implementation codes are synthesizable. The design characteristics of these four routers are summarized in Table II. DLABS_1+1 has total of three buffers; each of other routers has total of five buffers; each buffer is 8-flit depth. DLABS_2+2_duallink has two unidirectional links per an output port that total in four unidirectional links per an I/O port connecting with each its nearest neighboring router. Each of other router routers has two unidirectional links per an I/O port.

Three traffic patterns are used in the experiments: uniform random, transpose and bit-complement [4]. The simulation environment consists of an array of 8x8 nodes with each node consisting of a router and a PE. The PE injects and consumes packets into and out of the network with each packet has a fixed length of ten flits. For each packet injection rate, we run the simulation for 30,000 cycles. Activities of the system such as when packets enter or leave the network, as well as the status of routers' buffers (full or empty) are recorded into Matlab-readable files that are used for analyzing the overall network performance.

B. Performance Analysis

The average network latencies over various injection data rates of all four routers are shown in Fig. 12. For irregular random traffic, the typical wormhole achieves lower latency than DLABS_1+1 and DLABS_2+2 routers because it effectively utilizes its buffers. Over the whole execution time, each buffer has some packets to forward rather than be idle. The DLABS_1+1 performance is poorest because each shared buffer on a lane frequently has to process packets of multiple input ports simultaneously.

Since DLABS_2+2 has two buffers shared on each lane, pressure for each buffer is reduced, which improves the performance of the whole network. However, the pressure now moves to the output ports on each lane because they now have to receive multiple requests from their respective buffers. As a result its performance is still less than the wormhole router. The DLABS_2+2_duallink alleviates both these pressures but its overall performance is not much higher than the wormhole router. This is primarily due to the irregular and random nature of the traffic.

The performance of DLABS routers is shown to be more promising when running regular benchmarks such as transpose and bit-complement. In both benchmarks, the DLABS_2+2 achieves almost the same performance as the wormhole router while the DLABS_2+2_duallink has much lower latency and higher saturation throughput. The DLABS_1+1 also has low-load latency comparable to the other routers.

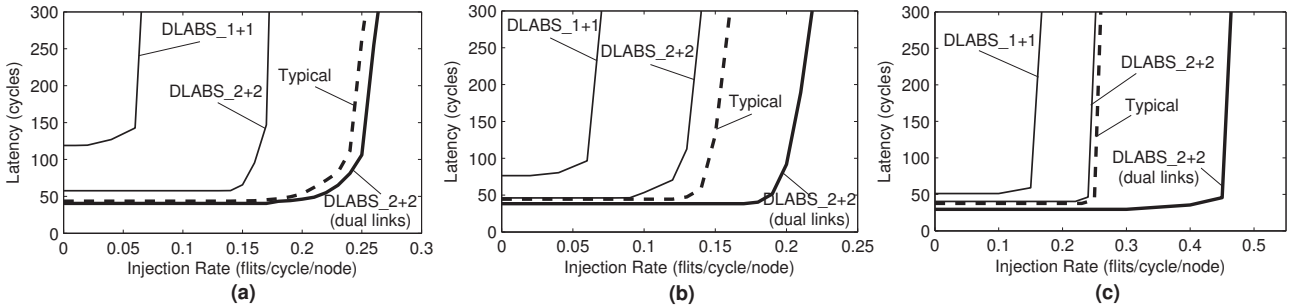

Fig. 12. Average latency over various packet injection rates in three traffic patterns: a) uniform random; b) transpose c) bit-complement.

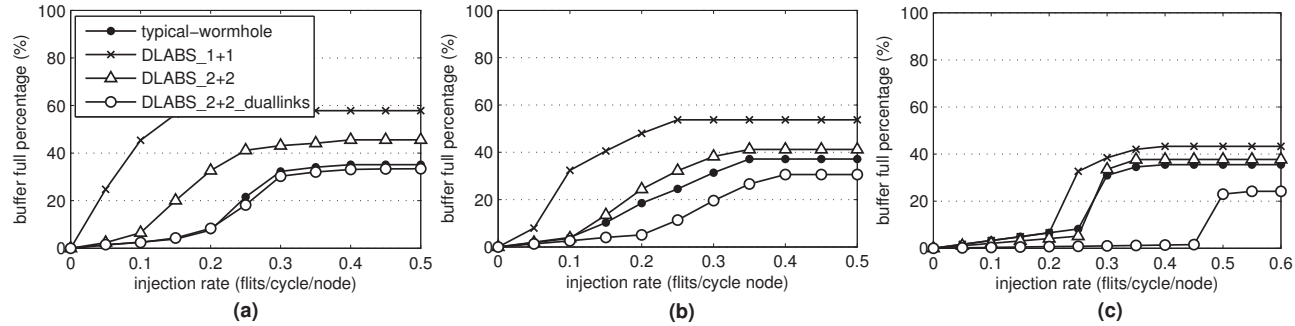

Fig. 13. Percentages of cycles that buffers of routers are full in the whole simulation time over different traffic benchmarks: a) uniform random; b) transpose; c) bit-complement

TABLE III
PERCENTAGE OF NUMBER OF BUFFERS THAT ARE ALWAYS IDLE IN THE WHOLE SIMULATION TIME OVER DIFFERENT TRAFFIC PATTERNS

Architect.	Typical wormhole	DLABS_1+1	DLABS_2+2	DLABS_2+2 _duallink
random	10.0%	1.0%	0.9%	0.9%
transpose	47.5%	16.2%	16.9%	16.9%
bit-comp.	45.0%	8.3%	9.8%	9.8%

Table III shows the ratio of number of always idle buffers over the total number buffers available in the system for all simulation time (320 buffers for routers having five buffers/router and 192 buffers for DLABS_1+1 which has three buffers/router). Less idling buffers means the system is effectively utilizing the buffer space. For the random traffic, the wormhole router network has only 10% of its buffers always idle. This boosts its overall network performance at the saturation status. However, for regular traffic patterns, the wormhole router has high percentage of idle buffers that are appropriately 50% in the whole simulation time. DLABS routers does better in reducing the number of idling buffers with around 1.0% always idle buffers for uniform random, and 8.3 to 16.9% for bit-compliment and transpose traffics.

Fig. 13 plots the percentage of cycles that buffers are full over the total simulation cycles at varying injection rate. Due to bottleneck on interconnect links, buffers in the DLABS_2+2 routers frequently stall; therefore obtaining a lower overall performance than wormhole routers. As shown, due to the good sharing among buffers, the DLABS_2+2_duallink router has a small number of buffers busy at the saturation status, and thus achieves higher performance than the wormhole routers for regular traffic patterns. Overall, these data highlight the ability of our DLABS architectures for reducing both overall busy and idle times of buffers resulting in a performance achievement comparable with the typical wormhole router.

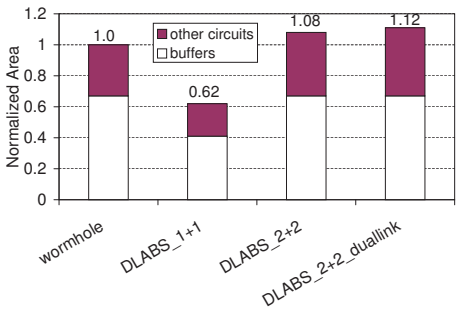

Fig. 14. The normalized area of router architectures after synthesized.

C. Area Comparison

We synthesized all four routers targeting ST Microelectronics 65 nm standard cells. The results are shown in Fig. 14. The five buffers occupy 66% of the area for the wormhole router. The DLABS_1+1 router has only three buffers with simple control circuits resulting in a 62% area when compared to the whole wormhole router. The DLABS_2+2 and DLABS_2+2_duallink have five buffers plus some control overheads making their total areas 108% and 112%, respectively, of the wormhole router.

D. Efficiency Analysis of Architectures

As shown above, the DLABS_1+1 has low area while achieving a modest performance; and DLABS_2+2_duallink obtains good performance while sacrificing some area cost. In order for a fair comparison of the efficiency of architectures, we analyze two other metrics: first, the product of the normalized area of each architecture with its average latency (ALP); second, the obtained throughput of each architecture per an area unit (TPA).

The ALPs of all four routers over three traffic benchmarks are shown in Fig. 15. As shown, they have curves not differing much from the average latency curves shown in Fig. 12. An important note is that

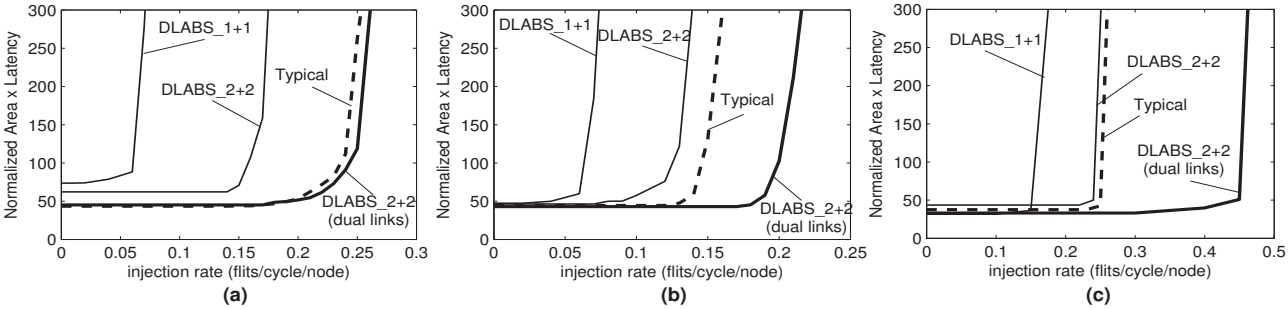

Fig. 15. Normalized Area Latency Product (ALP) of routers over three traffic patterns: a) uniform random; b) transpose; c) bit-complement

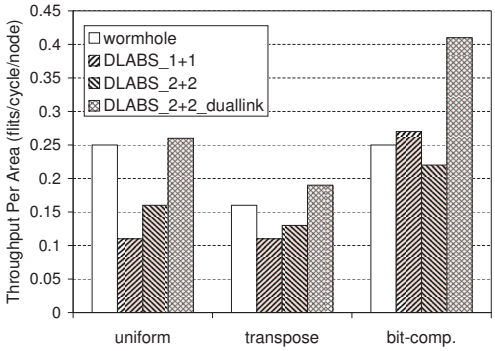

Fig. 16. Throughput per area (TPA) of architectures targeting an average network latency of 200 cycles over three traffic patterns: a) uniform random; b) transpose; c) bit-complement

the ALP of DLABS_1+1 reduces over all benchmarks which even lower than the typical one at low injection rates for regular patterns. The DLABS_2+2_duallink router still outperforms the typical one except it has a little bit higher ALP at low data rate over the random traffic.

We also consider the achieved throughput of routers per an area unit (TPA). Throughput of all routers are considered while obtaining an average latency of 200 cycles that is when they begin to reach the saturation status. The TPAs of networks using four routers are shown in Fig. 16. Higher TPA an architecture obtains, higher area and performance efficiency it is. As expected, over the random traffic, the typical wormhole router achieves better TPA than both DLABS_1+1 and DLABS_2+2 routers; and is only 4% smaller than the DLABS_2+2_duallink. For regular traffics, DLABS routers have TPA improved significantly. Especially, for the bit-complement traffic, TPAs of DLABS_1+1 and DLABS_2+2_duallink are 8% and 64% higher that of a typical one, respectively.

V. Conclusion and Discussion

In this paper, we have presented new buffer-sharing router architectures that allow efficient use of the buffer space for enhancing the network performance–especially for regular traffics. In order to avoid the potential appearance of deadlock in the network, we propose a dual-lane architecture where each lane is responsible for two input ports and two output ports that share one or more buffers.

In this work, we implemented three dual-lane buffer-sharing routers: DLABS_1+1 which has a single shared buffer on each lane; DLABS_2+2 which has two shared buffers on each lane in order that it has the same number of buffers as a wormhole router; and DLABS_2+2_duallink that has two links per router I/O port. The simulation results show our routers achieve good performances compared to typical wormhole routers over regular traffics. For the random traffic, on the other hand, due to its irregularity in packet distribution, the wormhole routers have good utilization of buffer space, and thus can achieve almost the same performance as a DLABS_2+2_duallink.

The analysis results offer two ways for efficient use of DLABS routers. For systems that do not need a very high performance network, but require low-area, then DLABS_1+1 is an appropriate candidate. It has a 62% area of a typical wormhole router and offers an acceptable performance for regular traffics. On the otherhand, if we need a router for high performance with small hardware overhead, DLABS_2+2_duallink is most suitable. Its performance is better than that of conventional wormhole routers over many different kinds of traffics with only 12% larger area.

Acknowledgments

This work was supported by a VEF Fellowship, SRC GRC Grant 1598 and CSR Grant 1659, ST Microelectronics, UC Micro, NSF Grant 0430090 and CAREER Award 0546907, IntellaSys 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 *ACM IEEE 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, *Principles and Practices of Interconnection Networks*, Morgan Kaufmann, San Francisco, USA, 2004.

[5] T. Bjerregaard and S. Mahadevan, "A survey of research and practices of network-on-chip," *ACM Computing Surveys*, vol. 38, pp. 1–51, Jan. 2006.

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

[7] 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.

[8] A. Kumar, L. Peh, et al., "Towards ideal on-chip communication using express virtual channels," *IEEE Micro*, vol. 2, pp. 80–90, Feb. 2008.

[9] A. Banerjee, P. T. Wolkotte, et al., "An energy and performance exploration of network-on-chip architectures," *IEEE TVLSI*, vol. 17, pp. 319–329, Mar. 2009.

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

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

[12] J. Duato, "A new theory of deadlock-free adaptive routing in wormhole networks," *IEEE TPDS*, vol. 4, no. 12, pp. 1320–1331, 1993.

[13] L. Peh and W. J. Dally, "A delay model for router microarchitectures," *IEEE MICRO*, vol. 21, pp. 26–34, Jan. 2001.

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

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