Buffered Crossbar Fabrics Based on Networks on Chip

Lotfi Mhamdi[1]   Kees Goossens[1,2]   Iria Varela Senin[1]

[1] Computer Engineering, Delft University of Technology, Delft, The Netherlands
[2] Eindhoven University of Technology, Eindhoven, The Netherlands

lotfi@ce.et.tudelft.nl, kees.goossens@es.ele.tue.nl

Abstract— Buffered crossbar (CICQ) switches have shown a high potential in scaling Internet routers capacity. However, they require expensive on-chip buffers whose cost grows quadratically with the port count. Additionally, similar to traditional crossbars, point-to-point switching mandates the use of long wires to connect inputs to outputs, resulting in non-negligible delays. In this paper, we propose a CICQ switching architecture where the buffered crossbar fabric is designed using a Network on Chip (NoC). Instead of a dedicated buffer for every pair of input-output ports, we use on-chip routers, one for each crosspoint. Our design offers several advantages when compared to traditional CICQs: 1) speedup, because the fabric can operate faster due to the small size of the NoC routers, their distributed arbitration and the short wires connecting them. This is in contrast to single-hop crossbars that use long wires and centralized arbitration. 2) Load balancing, because flows from different input-output port pairs share the same router buffers, contrary to the internal buffers of traditional CICQs that are dedicated to a single input-output pair. 3) Path diversity, allowing traffic from an input port to follow different paths to its destination output port. This results in further load balancing, especially for non-uniform traffic, and provides better fault tolerance in the presence of interconnect failures. We analyzed the performance of our architecture by simulation and presented its performance under wide traffic conditions and switch sizes. We prototyped, in CMOS technology, a 32×32 NoC-based crossbar switch. The implementation results suggest that we can clock the switch at a frequency of 413 MHZ, reaching an aggregate throughput in excess of 10^{10} ATM cells per second.

Index Terms— Scheduling, Buffered crossbar fabric.

I. INTRODUCTION

The crossbar-based fabric switch is the dominant architecture for today's high-performance packet switches [1] [2]. The fabric of a crossbar-based packet switch can be either unbuffered [1] or internally buffered [2]. The similarity between these two variants lies in the quadratic growth of their cost with the number of switch ports. Additionally, both architectures require a sophisticated input queueing structure, known as the virtual output queueing (VOQ) to achieve acceptable performance [3]. Unbuffered crossbar fabrics are cheaper than their buffered counterparts since they contain no internal buffers. However, they are hard to scale due to the high-computational complexity and centralized nature of their scheduling [3].

A buffered (CICQ) crossbar switch fabric (see Fig. 1(a)) overcomes the scheduling complexity by means of parallel and distributed schedulers, one per switch port [4]. However, it requires dedicated internal buffers for each pair of input-output ports of the switch. The cost of these internal buffers grows

as the square of the switch radix making CICQs unattractive for large port switch. Additionally, whether a crossbar switch is buffered or not, the fabric requires long point-to-point wires to interconnect the switch inputs to its outputs. This results in long delays and consumes high power to drive these wires.

In this paper, we propose a novel design for the CICQ switch architecture. Instead of using a dedicated internal buffer per input-output pair of ports, we design the whole buffered crossbar fabric as a Network on Chip (NoC) as depicted in Fig. 1(b). Our design offers several advantages when compared to traditional crossbar based fabric switches. First, using on-chip routers instead of dedicated internal buffers allows a better load balancing of the traffic passing through the switch. This is achieved by allowing the on-chip routers to switch traffic from any input to any output, resulting in sharing and better use of internal memory. This is in contrast to traditional CICQs that use dedicated internal buffers. The adoption of small routers and shared resources provides path diversity, by allowing multi-paths between every input-output pair of ports. This results in further load balancing especially in the presence of non-uniform traffic patterns and gives better fault tolerance in presence of interconnect failures. Traditional crossbar have no path diversity, they use expensive redundant planes instead [5]. Designing the fabric as a NoC allows scalability in port count and speed per port. This is achieved using short wires enabling reliable high-speed signaling as opposed to long wires. Using uniform short wires affords significant advantages in cost and performance. Additionally, a NoC based fabric requires simpler switch design by allowing simple input memory structure such as first in first out (FIFO) input queueing, as opposed to traditional design that requires sophisticated queueing structures such as the VOQ architecture.

The remainder of this paper is structured as follows: Section II discusses related work. Section III introduces our proposed multidirectional NoC-based crossbar switching architecture, named MDN. We describe its dynamics and explain its properties with emphasize on its routing mechanism. In Section IV, we present the hardware implementation results of the proposed architecture. Section V presents a detailed simulation study of the proposed switching architecture and compares it to the traditional CICQ switch. Finally we conclude the paper in Section VI.
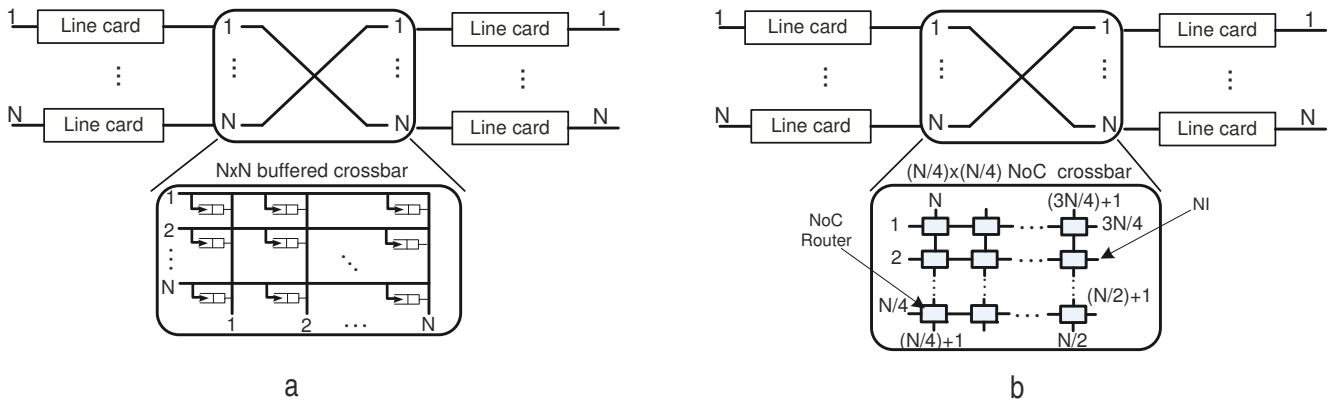
Fig. 1. Crossbar architecture: a) Conventional CICQ, b) Proposed MDN crossbar architecture.

II. RELATED WORK

The switch fabric is a key component for a wide spectrum of routers. Various switch fabric architectures have been proposed for high-performance routers, such as bus-based [6], shared memory [7] and crossbar [3]. The crossbar is considered the most suitable fabric architecture [3] due to its non-blocking properties, intrinsic multicast capabilities and its parallel point-to-point communication support. However, the crossbar switch fabric is considered non-scalable for two main reasons. First, its cost grows as the square of its input/output port count, making it unattractive for high-radix switches. Second, the centralized nature of the crossbar fabric scheduling makes it complex and hard to scale to high data rates and/or port counts.

Alternative solutions have been proposed. To overcome the crossbar quadratic cost growth, multistage switch fabrics were proposed [8]. Other solutions include the 3-D torus architecture used by the Avici routers [9]. The solution to solve the crossbar scheduling complexity was to use buffered CICQ crossbars. However, the scheduling simplification comes at the expense of a complicated and expensive crossbar, where internal memory banks are required, one per crosspoint of the crossbar. Proposals to reduce the internal memory requirement have recently appeared [10] [11] [12]. However, like crossbars, CICQs still suffer the inherent delay of the long crossbar wires.

We have recently proposed a fabric architecture based on NoCs, that we named UDN [13] and showed its good performance as compared to existing architectures. Input and output ports are laid out on two opposite sides of the UDN fabric chip. However, this layout mandated the use of long wires to wrap up the inputs/outputs around the UDN mesh. Our currently proposed architecture has the I/O pins spread over the four sides of the fabric chip (see Fig. 1(b)). This obviates the need for long wires by allowing the use of short uniform wires over the whole switch fabric, while using less crosspoints ($\frac{N^2}{16}$ vs. N^2). When compared to multistage fabrics or 3-D torus, our proposed architecture can be implemented on-chip and can provide a quick and straightforward upgrade to existing crossbar fabrics in a cost effective manner. Our work differs from previous art by proposing a *multi-hop* buffered NoC-based fabric architecture and studying its performance under various traffic models and switch settings.

III. MULTIDIRECTIONAL NOC-BASED CROSSBAR ARCHITECTURE

This section introduces the multidirectional NoC-based (MDN) crossbar switching architecture, describes its components and explains its dynamics.

A. Switch Model

We consider the MDN crossbar architecture depicted in Figure 1(b). The switch operates on fixed size packets (cells), where variable size packets are segmented upon entering the switch and reassembled at the switch output ports. There are N input ports and N output ports. Each input port contains a FIFO queue to hold incoming traffic during times of congestion. This in contrast to traditional crossbar based switches that use sophisticated input VOQs. The inputs and the outputs are connected to the four sides (the perimeter) of the crossbar fabric chip through I/O pads as depicted in Fig. 1(b).

As illustrated in the Figure, the crossbar fabric core consists of $(N/4) \times (N/4)$ mesh of on-chip routers, with traffic flowing in all four directions of the mesh. Hence, the name Multidirectional Network on chip based (MDN) crossbar switch. The MDN is a two-dimensional mesh of packet-switched routers, with network interfaces (NI) on the four sides of the mesh. Cells are transferred to the MDN fabric chip when there is space in the MDN's NI buffers. Cells are packetised by the ingress NIs; the routers then route these packets to the egress NIs, where the cells are depacketised and forwarded to the output line cards. The MDN on-chip routers employ store and forward packet switching with FIFO input queuing. Each router's input port maintains a small amount of FIFO buffering (≤ 4 packets per FIFO). Credit-based flow control is used between these on-chip routers to ensure a lossless communication. Each router uses a round-robin scheduling arbitration when selecting packet to forward along the mesh.

To make the MDN architecture scalable, we allow the crossbar fabric to have multiple (P) planes that are vertically connected only at the edges (border routers). This is illustrated in Fig. 2, where each plane is connected to its adjacent plane(s) via its edge routers. The crossbar I/Os connecting it to the input/output line cards are placed
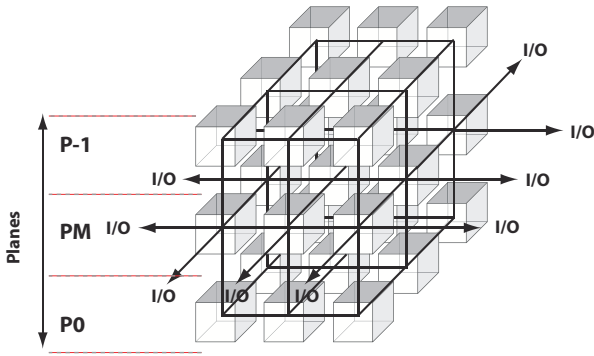
Fig. 2. The MDN crossbar architecture with P planes.

Fig. 3. The MDN router architectures for $P=1$.

Fig. 4. 4x4 MDN with VC paths.

only on the central plane ($PM = \lfloor P/2 \rfloor$). Depending on the plane they belong to and their placement in the mesh, the MDN on-chip routers can have different degrees. Routers of planes 0 and $P-1$ have degree 4×4, except the corner routers with degree 3×3. Routers of plane P ($P \in \{0, ..., PM-1\} \cup \{PM+1, ...P-1\}$) have degree 4×4, except the border routers (excluding the corners) with degree 5×5. The border routers of the central plane ($\lfloor P/2 \rfloor$) have a degree of 6×6, and the rest have degree 4×4 each.

B. Routing in MDN

Input traffic is routed through the MDN based crossbar fabric before reaching its destination output port. The route of a packet through the MDN is determined upon its entry to the NoC, which is encoded in the packet header. Packets follow a deterministic minimal routing while inside the MDN. In each MDN plane (P) we use two routing algorithms: XY routing and balanced XY routing. XY routing is used for traffic coming from an input i and heading to output j, where input i and output j reside in two adjacent (perpendicular in this case) sides of the MDN mesh. Balanced XY routing is used when the destination output port is in the opposite side of the source input port from which the traffic is originated. In XY routing, the position of the mesh routers is described by coordinates, the x-coordinate for the horizontal and the y-coordinate for the vertical position. Packets are routed to the correct column (X) first and then to the correct row (Y).

Balanced XY is an enhanced version of the standard XY routing, wherein an extra turn in one of the earlier columns is introduced in order to load balance the traffic in the mesh. A packet for output x turns South/North (East/West) when $x = (N/4-i+j) \bmod (N/4)+k(N/4)$, ($k \in [0,3]$) and East/West (South/North) when $x = j$, where i, j indicate the current router position in the mesh, N the number of inputs/outputs of the switch and k refers to one of the four sides of the mesh. In balanced XY, all packets of an input-output pair follow the same path. This is also the case with standard XY routing, where packets from the same input-output pair follow the same path, obviating the need for reordering.

Since the MDN uses more than one plane (P), the path of each packet and the plane(s) through which it is routed is computed upon its entry to the MDN. All packets enter and exit the MDN through the central plane (PM), where the I/Os
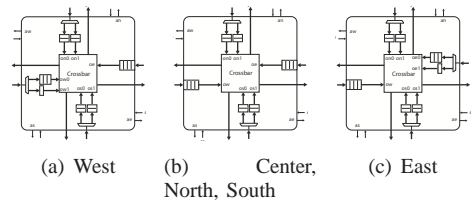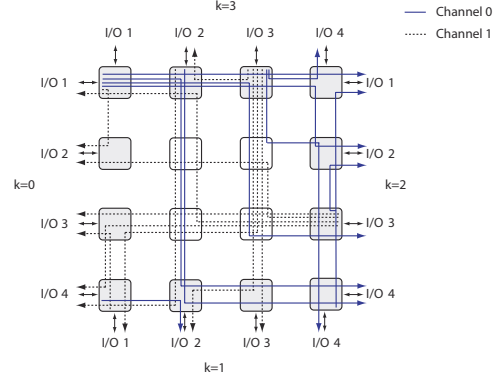
are connected. Depending on the desired output port, incoming packets get distributed over the lower or upper MDN planes as follows: First, the packet is routed up or down with respect to the central plane. Our routing strategy routes packets with odd destination output port number up, and down otherwise. Once a packet, going to output x, arrives at network interface (i, j), it is routed up or down. We need to determine at which plane (P_z) it will be routed. This is determined as follows: $(N/4 - i((k+1) \bmod 2) - (j(k \bmod 2)) + P_z) \bmod \lfloor P/2 \rfloor = x \bmod \lfloor P/2 \rfloor$. The modulo operation aims at distributing the traffic over the P planes to maintain a load balance and avoid congesting paths while idling others.

Recall that the inputs/outputs of the switch are spread over the four sides of the MDN crossbar fabric. Packets can then flow in all directions and therefore deadlock can occur. We avoid this by using two virtual channels (VC) at North and South inputs of the router, as shown in Fig. 3. Routers at the East and West edges of the mesh additionally have the VCs at their East and West inputs, respectively. Packets use VC 0 if their destination is East of their starting position, and VC 1 otherwise. This way, the MDN crossbar mesh is virtually divided in two deadlock free networks, wherein the traffic travels in one direction. Fig. 4 is an example of the VC distribution and balanced XY routing in a 4x4 mesh.

IV. IMPLEMENTATION RESULTS

We synthesized a NoC-based MDN in an ASIC 65 nm CMOS technology to assess its performance. Similar to [13], we used the Æthereal NoC [14], with input-queued routers, two VCs, and a non-blocking crossbar. Static-priority arbitration is used between the two VCs, and round robin per VC. This router and the MDN router will not differ significantly in terms of area and speed. The area of a router is dominated by

the number of registers, which is the same for both routers. The arbitration of the Æthereal router is more complex than that of the MDN router and hence will be slower.

A 3×3 MDN topology is generated with 1 plane, RTL VHDL, and SystemC models of the NoC from a high-level specification [15]. This instance contains different router degrees, and allows us to compute the area of any size MDN crossbar. Synthesis for a 65 nm CMOS technology, without any optimisations, achieved 413 MHz with a cell area of 4.8 mm^2. Routers of degree 3 and 4 occupy 0.29 and 0.38 mm^2 respectively, and NIs 0.32 mm^2. Router of degree 5 and 6 routers are estimated to occupy 0.49 and 0.64 mm^2 respectively. Denote the area of a router of degree d by $\mathcal{A}_d$. The area of a crossbar with N input-output ports and P planes is then:
$[2(\mathcal{A}_4(\frac{N^2}{16} - 4) + \mathcal{A}_3 * 4)] + [\mathcal{A}_4(\frac{N}{4} - 2)^2 + \mathcal{A}_6 * (N - 4)] + [(P - 3) * \mathcal{A}_4((\frac{N}{4} - 2)^2 + 4) + \mathcal{A}_5 * (N - 8)] + N * \mathcal{A}_{NI}$ mm^2. For $N = 32$ and $P = 3$, we need $156\mathcal{A}_4 + 28\mathcal{A}_6 + 8\mathcal{A}_3 + 32\mathcal{A}_{NI} = 90$ mm^2. The registers that are used for FIFO buffers dominate the area. By using dedicated hardware ripple-through FIFOs, described in [16], the area drops significantly. In a 90 nm CMOS process a 48-word 37-bit FIFO occupies a third of a register-based FIFO. Using the same scaling factor for all FIFOs in 65 nm, an N port crossbar would occupy much less than previously shown.

The router data path is 36 bits, hence the cycle time (to move one packet or ATM cell from one router to the next) is $\frac{53*8}{36} + 1 = 13$ cycles (assuming store and forward). The maximum sustainable throughput of an $N \times N$ crossbar (diagonal traffic), is therefore $\frac{N}{13*2.4 \text{ ns}} = 32N10^7$, or 10^{10} cells/second for $N = 32$. The minimum cell latency is $(\frac{3N}{4})13$ cycles, or $(9.75N)2.4 \simeq 0.75$ milliseconds for $N = 32$.

V. SIMULATION RESULTS

This section presents the experimental results of the proposed NoC-based switch and compares it to a traditional CICQ buffered crossbar switch with N^2 internal buffers each of size 1 cell and running round robin scheduling. We tested the performance of the MDN architecture for different switch sizes, various speedup values, different planes (P) and different router buffer sizes. Simulation run for one million timeslots and we gather the data when a tenth of the simulation time has elapsed. The performance evaluation is carried under various traffic conditions, including: i) Bernoulli uniform and bursty unform with different burst sizes; ii) The diagonal traffic as defined in [17] and; iii) The unbalanced traffic model as defined in [18].

A. Uniform Traffic

Fig. 5 depicts the average cell delay of the proposed MDN architecture and compares it to that of a CICQ switch. The MDN uses just one plane here and employs a speedup of one and two (referred to as SP1 and SP2 in the Figure). When the traffic (bernoulli and bursty) is uniformly distributed over the outputs, CICQ outperforms MDN. The higher delay of MDN as compared to CICQ, is attributed to its much smaller interconnection stage as compared to the CICQ. The switch size used here is 32×32, meaning that the CICQ uses up
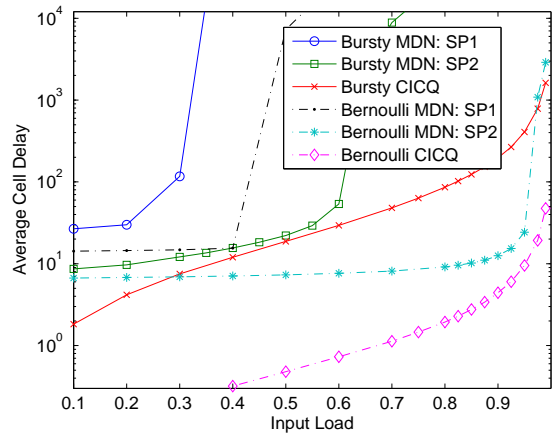

Fig. 5. Cell delay comparison between the MDN and CICQ switch of size 32x32 under Uniform traffic.
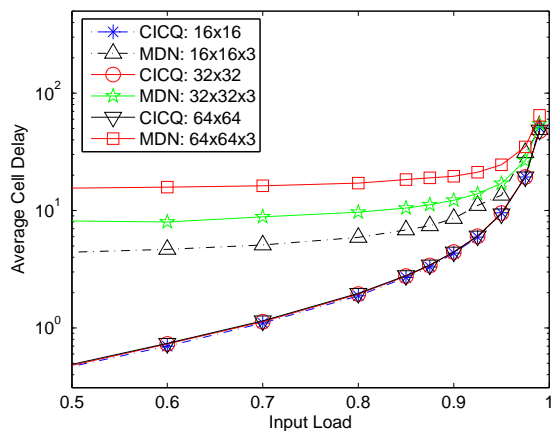

Fig. 6. CICQ vs. MDN with 3 planes under Bernoulli uniform traffic and different switch sizes.

to 32^2 (or 1024) internal crosspoint buffers. However, MDN uses just $(32/4)^2$ (or 64) internal crosspoints (on-chip routers). This results in MDN being highly congested as compared to CICQ, therefore resulting in higher cell delays.

A way to alleviate the MDN congestion is to use more planes instead of just only one. This is illustrated in Fig. 6, where we compare the delay of a CICQ switch to that of an MDN using three planes and speedup two with varying switch sizes. We can see that the CICQ has a lower delay than that of the MDN under light loads. This is attributed to the multi-hop nature of the MDN, where packets experience initial constant delay to pass through the MDN mesh. The MDN delay approaches that of the fully buffered CICQ under high loads ($> 90\%$), despite their significantly different fabric sizes (N^2 vs. $\frac{3}{16}N^2$). Observe that, unlike CICQ, the delay of MDN is not sensitive to varying traffic loads and/or switch sizes. This is important particularly for quality of service purposes. The same trend is observed under bursty uniform traffic (burst size of 16) as depicted in Fig. 7. Increasing the number of planes improves the MDN delay performance. The MDN is scalable with the switch radix, where the delay of $64 \times 64 \times 3$ (a 64×64 MDN with 3 planes) is comparable to that of $32 \times 32 \times 3$.
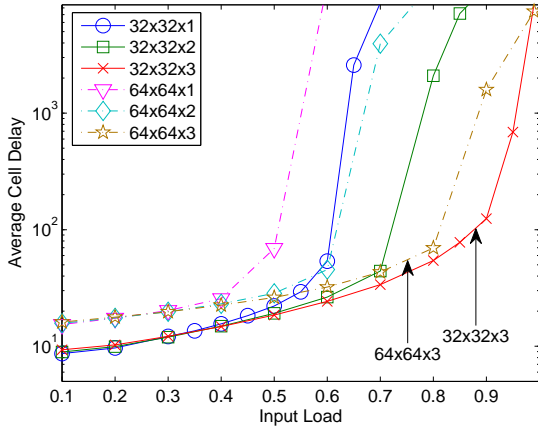
Fig. 7. MDN under Bursty uniform traffic with different planes and switch sizes.
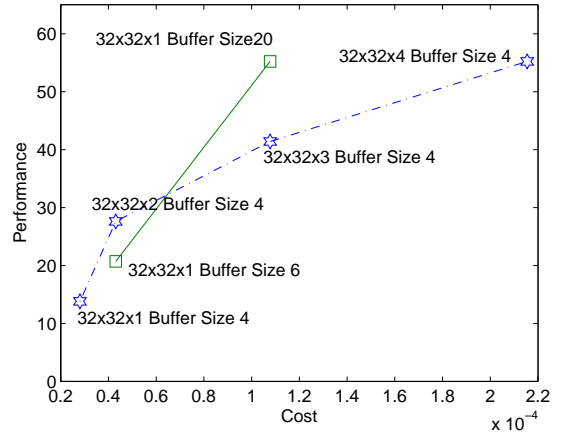

Fig. 8. MDN cost-performance for Bursty uniform traffic and different planes and buffer sizes for SP2.

We want to study the cost-performance curves of the MDN architecture. However, in order to do this, we need to tune various parameters such as, switch size, speedup (SP), depth (number of planes) of the interconnect, buffer depth, etc. We, therefore, define the cost to be: $cost = SP \times area$, where SP is the speedup of the NoC and $area$ is the chip area as presented in Section IV. We define the performance to be: $Performance = 1/Delay$, where $Delay$ refers to the average cell delay. Fig. 8 depicts different cost-performance settings of a 32×32 MDN under bursty traffic. We vary the buffer size of each MDN router as well as the number of planes and observe the cost-performance outcome. We can see from the Figure that it is better to increase the number of planes (the dotted graph on Fig. 8) rather than the on-chip router buffer size (the solid line graph) to get a good cost-performance ratio.

B. Non uniform Traffic

This section studies the MDN performance under non-uniform unbalanced and double diagonal traffic patterns. We study the switch throughput stability under unbalanced traffic as well as the average cell delay under both unbalanced and diagonal traffic conditions. Fig. 9 compares the throughput stability of the MDN (with one plane) to that of a traditional CICQ switch. We can see that the MDN with speedup one is unable to deliver more than 78% throughput and has lower performance than CICQ. This is attributed to its highly congested single plane. When we increase the speedup value, MDN outperforms CICQ by delivering 100% across all ranges of the unbalanced coefficient, ω.

We studied the average cell delay of the MDN, using one plane and a speedup of two, and compared it to that of a CICQ switch. Fig. 10 shows the delay of the two switches under double diagonal traffic as well as unbalanced traffic (for $\omega = 0.5$). We can see that MDN outperforms CICQ under both traffic conditions when the input load is high ($\geq 85\%$). The higher delay of MDN under light loads is attributed to its multi-hop nature, incurring a delay of packets to go through the MDN on-chip network. However, as the load increases,
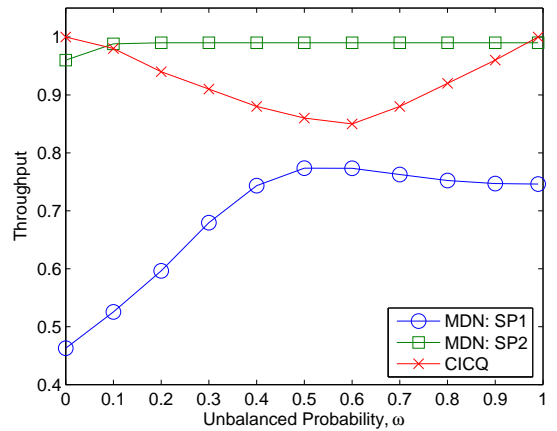

Fig. 9. Performance of a 32x32 MDN and CICQ switch under Unbalanced traffic.

the MDN delay increases much slower than that of the CICQ and this is desirable as it keeps the switch delay jitter low.

Next, we study the effect of the number of planes in MDN on the switch throughput. Fig. 11(a) depicts the stability of a 32×32 MDN with varying speedup values and number of planes respectively. When the speedup is just one, the increase in the number of planes helps in achieving higher throughput, but full throughput is not achieved. Using a speedup of two, the switch throughput reaches 100% with two or three planes. The same trend is observed for higher radix switch of 64×64 ports, as shown in Fig. 11(b). When the number of planes is set to 3 (the dotted line $64 \times 64 \times 3$) or higher, the MDN delivers 100% throughput.

Using just 3 planes seems to be enough for the MDN to achieve optimal performance under wide traffic conditions (both uniform and non-uniform). The crossbar of an $N \times N$ MDN switch with 3 planes would require $(N/4)^2 * 3 \simeq N^2/5$ crosspoints (on-chip routers). This is equivalent to a cost reduction by a factor of 5 when compared to the fully buffered crossbar of a CICQ switch that uses N^2 internal buffers.

VI. Conclusion

This article proposes a novel design of buffered cross-bar CICQ switches. Instead of a dedicated crosspoint-based buffered crossbar switch, we propose a NOC-based crossbar architecture. Our proposal has several advantages compared to previous designs. First, a multi-hop buffered interconnect divides the long wires between crossbar inputs and outputs into a number of shorter wires between the individual hops. This can alleviate the long wiring delay faced in traditional crossbar design. Second, the adoption of a NoC-based crossbar chip allows internal buffer optimization and sharing. Third, a multi-hop interconnect provides path diversity and therefore load balancing and reliability in presence of interconnect failure. We proposed a NoC-based architectures, namely the MDN switching architecture, studied its performance under various metrics and showed that it can achieve high-performance. We prototyped a 32×32 NoC-Based crossbar switch and found that we can clock the switch at 413 MHZ with a clock cycle time of 2.4 ns, achieving an aggregate switching bandwidth in the Tbps ranges.

Fig. 10. Cell delay comparison between the MDN and a CICQ switch of size 32 x 32 under non-uniform traffic.

(a) 32x32

(b) 64x64

Fig. 11. MDN under Unbalanced traffic with different switch sizes, number of planes and speedup values.

References

[1] N. McKeown and al., "The Tiny Tera: A Packet Switch Core," *IEEE Micro*, pp. 26–33, January/February 1997.

[2] F. Abel and al., "A Four-Terabit Packet Switch Supporting Long Round-Trip Times," *IEEE Micro*, vol. 23, no. 1, 2003.

[3] N. McKeown, "Scheduling Algorithms for Input-Queued Cell Switches," Ph.D. dissertation, University of California at Berkeley, May 1995.

[4] L. Mhamdi and M. Hamdi, "MCBF: A High-Performance Scheduling Algorithm for Buffered Crossbar Switches," *IEEE Communications Letters*, vol. 07, no. 09, pp. 451–453, September 2003.

[5] R. R. Cessa, "Design and Analysis of Reliable High-Performance Packet Switches," Ph.D. dissertation, Plytechnic University, April 2001.

[6] T. Aramaki and al., "Parallel 'ATOM' Switch Architecture For High Speed ATM Networks," *IEEE ICC*, 1992.

[7] M. Devault, J. Y. Cochennec, and M. Servel, "The Prelude ATD Experiment: Assessments and Future Prospects," *IEEE JSAC*, vol. 06, no. 09, pp. 1528–1537, December 1998.

[8] S. Iyer and N. McKeown, "Analysis of the Parallel Packet Switch Architecture," *IEEE/ACM Transactions on Networking*, vol. 11, no. 2, pp. 314–324, 2003.

[9] W. Dally, "Scalable switching fabrics for internet routers," http://www.avici.com/technology/whitepapers/TSRfabric-WhitePaper.pdf, 2002.

[10] R. Rojas-Cessa and Z. Dong, "Combined Input-Crosspoint Buffered Packet Switch with Flexible Access to Crosspoint Buffers," *IEEE ICCDSC*, April 2006.

[11] N. Chrysos and M. Katevenis, "Scheduling in Switches with Small Internal Buffers," *IEEE Globecom*, pp. 614–619, Nov. 2005.

[12] L. Mhamdi, "PBC: A Partially Buffered Crossbar Packet Switch," *IEEE Transactions on Computers*, vol. 58, no. 11, pp. 1568–1581, Nov. 2009.

[13] K. Goossens, L. Mhamdi, and I. Varela Senin, "Internet-router buffered crossbars based on networks on chip," *Proc. Euromicro Symposium on Digital System Design (DSD)*, Aug. 2009.

[14] K. Goossens and al., "The Æthereal network on chip: Concepts, architectures, and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, 2005.

[15] ——, "A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification," *DATE*, pp. 1182–1187, Mar. 2005.

[16] P. Wielage and al., "Design and DFT of a High-Speed Area-Efficient Embedded Asynchronous FIFO," *DATE*, Apr. 2007.

[17] P. Giaccone and al., "An implementable Paraller Scheduler for Input-Queued Switches," *IEEE Micro*, vol. 22, no. 01, pp. 19–25, 2002.

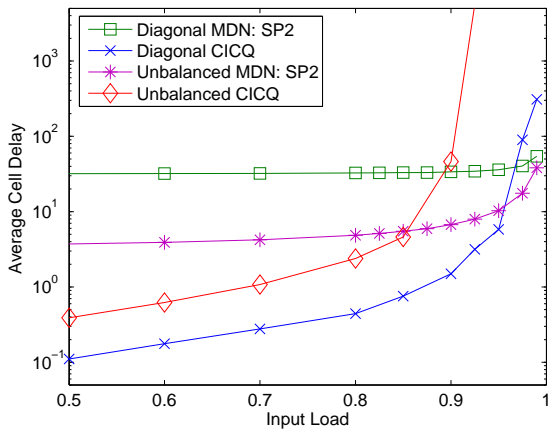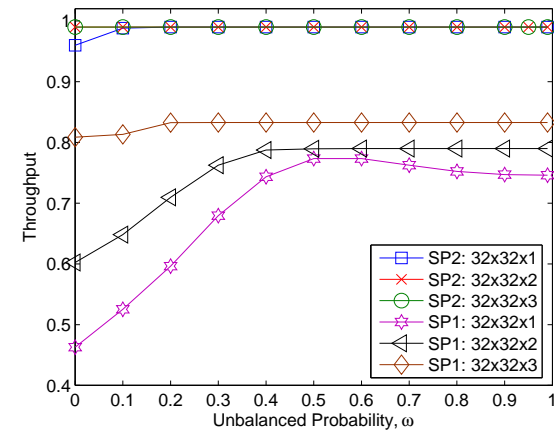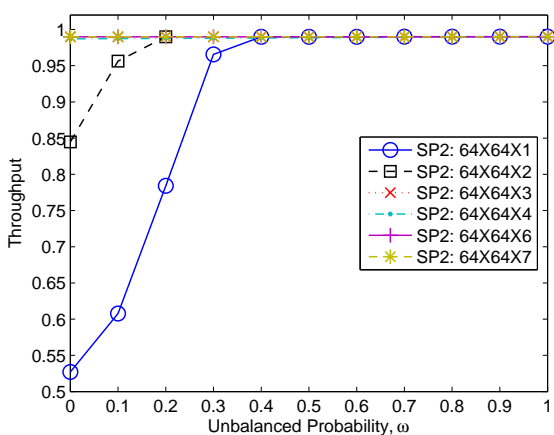[18] R. Rojas-Cessa, Z. J. E. Oki, and H. J. Chao, "CIXB-1: Combined Input One-Cell-Crosspoint Buffered Switch," *IEEE HPSR*, pp. 324–329, 2001.