

High-Radix Crossbar Switches Enabled by Proximity Communication

Hans Eberle*, Pedro J. Garcia†, José Flich‡, José Duato‡,
Robert Drost*, Nils Gura*, David Hopkins*, Wladek Olesinski*

*Sun Microsystems, 16 Network Circle, Menlo Park, CA 94025, USA

Email: {hans.eberle, robert.drost, nils.gura, david.hopkins, wladek.olesinski}@sun.com

†Universidad de Castilla-La Mancha, Escuela Superior de Ingeniería Informática, C.P. 02071, Albacete, Spain

Email: pgarcia@dsi.uclm.es

‡Universidad Politécnica de Valencia, Camino de Vera, s/n, C.P. 46022 Valencia, Spain

Email: {jflich, jduato}@disca.upv.es

Abstract—We describe a novel way to implement high-radix crossbar switches. Our work is enabled by a new chip interconnect technology called Proximity Communication (PxC) that offers unparalleled chip IO density. First, we show how a crossbar architecture is topologically mapped onto a PxC-enabled multi-chip module (MCM). Then, we describe a first prototype implementation of a small-scale switch based on a PxC MCM. Finally, we present a performance analysis of two large-scale switch configurations with 288 ports and 1,728 ports, respectively, contrasting a 1-stage PxC-enabled switch and a multi-stage switch using conventional technology. Our simulation results show that (a) arbitration delays in a large 1-stage switch can be considerable, (b) multi-stage switches are extremely susceptible to saturation under non-uniform traffic, a problem that becomes worse for higher radices (1-stage switches, in contrast, are not affected by this problem).

I. INTRODUCTION

Today, most high-performance compute systems such as supercomputers and server farms are based on commodity compute nodes tied together by high-performance interconnects. With the commoditization of the compute node, the interconnection network is becoming an increasingly critical system component and, often, the key differentiator that determines the scale and performance of the overall system.

Traditionally, scientific compute clusters have been the main drivers for the development of high-performance interconnects. A state-of-the-art example is the Sun Datacenter Switch 3456 by Sun Microsystems [20]. This switch is housed in a single rack and interconnects a total of 3,456 4x InfiniBand nodes through 1,152 copper cables. The internal architecture corresponds to a 5-stage Clos network.

Lately, high-performance interconnects also play an increasingly crucial role in commercial data centers. Not only are ever more powerful compute nodes based on multi-core processors driving up communication requirements, several trends such as network consolidation and hardware virtualization also demand an increasingly powerful communication infrastructure.

This work was supported by the Spanish MEC under Grant TIN2006-15516-C04-01, by CONSOLIDER-INGENIO 2010 under Grant CSD2006-00046, and by Junta de Comunidades de Castilla-La Mancha under grant PBC08-0078-9856.

To illustrate the scale of commercial installations, we briefly describe a typical blade server system used in the telecommunications industry based on the Advanced Telecommunications Computing Architecture (ATCA) standard [2]. Such systems are managed in entities corresponding to a data center aisle or a dozen racks, with each rack made up of three chassis and each chassis holding a dozen blades; this configuration corresponds to a total of 432 blades. The latest generation of ATCA blades has a 10 Gbit/s interface to the chassis backplane. Within a chassis, blades are connected in a star topology. Thus, total backplane bandwidth is 120 Gbit/s. Though too costly to realize today, the ideal interconnect for this system has a bisection bandwidth of 4.32 Tbit/s and provides homogeneous access to all 432 blades in the sense that the interconnect is non-blocking and that latency is equal for data transfers between any pair of nodes. The homogeneity of this interconnect greatly simplifies system management tasks such as load-balancing or allocating resources to application pipelines that span multiple blades.

Taking all this into account, the idea of high-performance, high-radix switches looks nowadays rather appealing to the interconnect research community. In fact, there have been some recent publications that address this topic [18], [11], [14]. However, these proposals are mainly about novel switch architectures and organizations rather than new chip interconnect technologies. Indeed, our contribution in this paper complements these proposed architectures.

Implementations of high-radix switches are typically IO limited. To overcome this limitation, parallel sliced switches such as the Tiny Tera switch [12] and the Prizma switch [1] have been proposed. These designs require a separation of the data path and its control logic in that a centralized scheduler coordinates transfer of data through the sliced switch fabric.

In this paper, we describe an alternative design for high-radix switches with a single-stage architecture that makes use of a novel multi-chip module packaging technology. Applications range from high-performance cluster interconnects to data center backbone grids. In the former case, its main attraction is its single-stage architecture that reduces latency and, in the latter case, the main benefits are cost reduction thanks to a lower component count and less susceptibility to

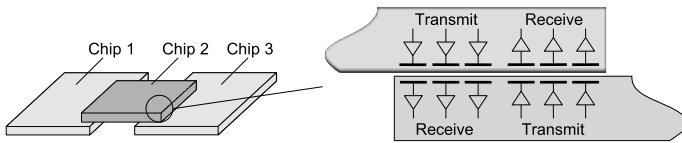


Fig. 1. Proximity Communication

hot spot traffic.

Our paper is structured as follows. Section II introduces a new chip interconnect technology called Proximity Communication that enabled this work. Section III shows how this technology can be applied to large switch fabrics, namely, the implementation of a large flat switch architecture. Section IV compares the performance of conventional multi-stage switches and single-stage switches based on Proximity Communication. Finally, in Section V some conclusions are provided.

II. PROXIMITY COMMUNICATION

Proximity Communication (PxC) is a novel interconnect technology for multi-chip modules. Details of this technology have been described in [5], [6], [10]. Here, we only provide a summary of the technology so that the reader has sufficient background to follow our explanations of a switch application. Figure 1 shows an example of a PxC assembly consisting of three chips with each pair of chips interconnected by a PxC interface. PxC relies on capacitive coupling to transmit signals from one chip to another. Microscopic metal pads are created in the top metal layers of the semiconductor chips during chip fabrication. Two chips, with receiver and transmitter pads, are then placed facing each other such that the pads are only a few micrometers apart. Each transmitter-receiver pad pair forms a plate capacitor, and voltage changes on the transmitter pad cause voltage changes on the receiver pad.

PxC offers many advantages over conventional chip interconnect technologies. By relying on structures directly integrated on the chip and omitting relatively large contacts with a printed circuit board, chip IO bandwidth density is improved by two orders of magnitude in comparison with conventional chip packaging technologies such as ball grid arrays. In absolute numbers, PxC offers up to 10 Tbit/s of chip IO bandwidth per mm^2 of chip overlap. Another noteworthy benefit is the possibility to rework a multi-chip module should there be a need to remove and replace a faulty chip. This is possible thanks to the lack of permanent contacts.

The dramatic increase in chip IO bandwidth made possible by PxC allows the system designer to completely rethink the architecture of large-scale systems. In this paper, we focus on large switches and try to answer the question of how their architecture and implementation changes with this disruptive technology.

III. PxC-ENABLED FLAT SWITCH ARCHITECTURE

Today's largest commercially available single-chip switches are the Mellanox Infiniscale III 24-port 4x InfiniBand switch

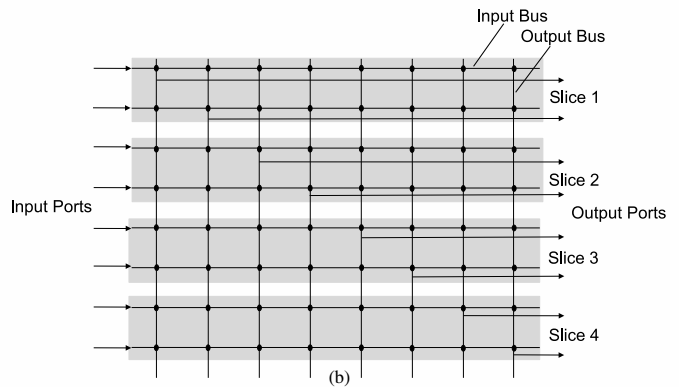
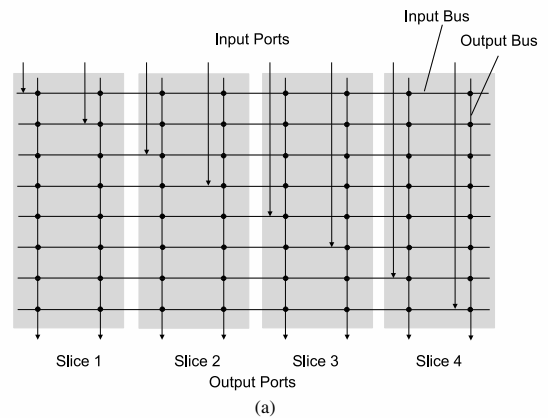


Fig. 2. Vertically (a) and horizontally (b) sliced crossbar switch

[13] and the Fulcrum FocalPoint FM2224 24-port 10 Gbit/s Ethernet switch [8]. If a larger switch is required, designers have to resort to hierarchical topologies that make it possible to build a switch that is larger than the individual switch element. Examples of popular hierarchical networks are Clos networks [4] and Beneš networks [3]. As these hierarchical networks require several stages of switches, they are also referred to as multi-stage interconnection networks (MINs). There is significant cost and complexity incurred in building a MIN. Furthermore, its performance characteristics are typically inferior to those of a single-stage network. For example, as we will show in Section IV, a MIN can saturate much more easily than a single-stage network.

In the following, we will show how PxC simplifies the way we build large-scale switch fabrics. Thanks to the large amount of chip IO bandwidth offered by PxC, it is now possible to implement a flat switch architecture rather than a complex hierarchical design. In the following, we will show how multi-chip switch fabrics can be implemented with a simple crossbar architecture that was previously only applicable to single-chip switch implementations. This simple architecture is made possible by the observation that PxC offers enough bandwidth that a switch can be partitioned in such a way that the full bisection bandwidth can be exposed at the chip boundaries. This approach works for even the largest switch fabrics with multiple Tbit/s bisection bandwidth.

When partitioning a design, the obvious two choices are to either partition the design vertically or horizontally. Slicing a crossbar switch vertically as shown in Figure 2a corresponds to segmenting the input busses¹ while horizontal slicing as shown in Figure 2b means segmenting the output busses. Though there are more ways to partition a switch, these two options have the important property that their slices are identical. That is, the slices are identical in their IO structure as well as in their internal logic. With a single chip design, we minimize design cost as well as manufacturing cost.

Partitioning a switch as shown in Figure 2 exposes the full switch bisection bandwidth at the interfaces of a slice. This bandwidth is required at each boundary of a slice and, thus, the more partitions or slices there are, the more interfaces there are that require the full switch bisection bandwidth. We are not aware of any chip packaging technology other than PxC that provides for enough IO bandwidth to support this kind of partitioning for a high-radix switch. With PxC, this organization becomes feasible making the many advantages of a flat switch architecture available at a much larger scale than has been possible so far.

The implementation of a large flat switch also poses challenges with respect to switch architectures. This is, however, not the topic of this paper and has been addressed in separate papers. In [15] we are describing a scalable data path architecture that reduces the amount of memory needed for implementing a buffered crossbar and in [16], [17] we introduce a pipelined version of the wrapped wave front arbiter [19] for scheduling large switches.

A. Multi-chip Module Topologies

In this section, we describe how a crossbar switch can be implemented in a PxC-based multi-chip module (MCM). We examine two MCM topologies, namely a one-dimensional vector MCM shown in Figure 3a and a two-dimensional matrix MCM shown in Figure 3b.

An MCM uses two types of chips: Island chips implement logic and Bridge chips contain point-to-point links connecting two neighboring Island chips. There are PxC interfaces at both ends of these links. Referring to Figures 3a and 3b, the Island chips face up and the Bridge chips face down. By restricting application logic to the Island chips, the packaging solution is simplified as heat has to be removed from one side of the MCM only allowing for a conventional thermal solution. For example, the package developed for the prototype switch described in Section III-D uses a metallic heat sink for removing heat from the Island chips, whereas heat removal from the Bridge chips relies on thermal convection only.

The two topologies vastly differ in complexity and performance. Clearly, as the dimensionality of the package goes up, the mechanical aspects become more complex as mechanical compliance and physical effects such as thermal expansion, contraction, and warping have to be considered for every dimension.

¹Note that, in strictly technical terms, a horizontal wire does not constitute a bus as there is only one source in the form of the input port.

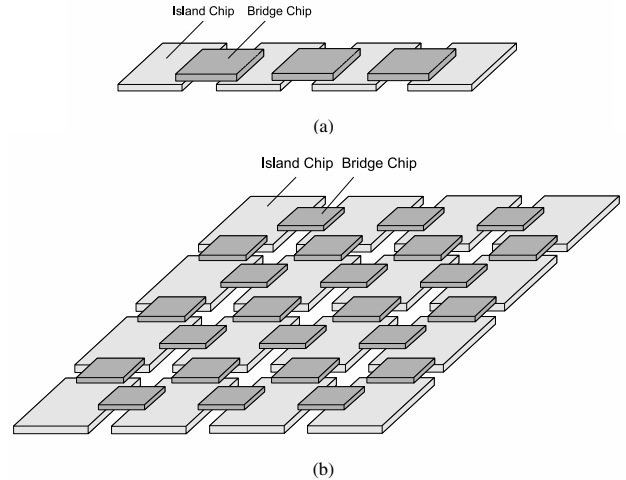


Fig. 3. Vector MCM (a) and matrix MCM (b)

The vector MCM is attractive not only because of relaxed mechanical requirements, but also because of simpler off-chip IO options. Since only two edges of the chip are used for PxC interfaces, the other two edges are available for connecting the chips with a substrate or printed circuit board. This is exemplified by the prototype switch described in Section III-D that uses conventional wirebonds to connect the MCM with the printed circuit board (PCB). In a matrix MCM, wirebonds could only connect to the outermost rows and columns of chips, thus, severely limiting access to internal chips. This asymmetry in the IO capabilities of the individual chips also has the unwanted consequence that outer and inner chips of an MCM are different. To provide a more uniform IO solution, a matrix MCM requires a more complex interconnect technology, for example, by making use of the third dimension through direct die-attachment of fiber-optic cables, or flip-chip attachment to connectors and copper cables. Such an interconnect has the potential to supply significantly higher IO bandwidth than is possible with wirebonds.

While higher dimensionality increases the complexity of the packaging solution, the higher cost comes with larger scale and higher performance. Assuming that up to n chips can be connected in one dimension, the vector MCM can accommodate n chips and the matrix MCM can contain n^2 chips. There are limits to the scale given by its mechanical properties but also by the amount of power that can be brought into the array and the amount of heat that can be removed from the array. The matrix MCM further offers higher PxC connectivity. While two chip edges are reserved for PxC in a vector MCM, all four chip edges are used for PxC in a matrix MCM. Thus, the matrix MCM doubles the number of PxC channels available per Island chip.

B. Vector Switch

We now want to examine how a crossbar architecture can be mapped onto a PxC MCM. It is easy to map a partitioned crossbar onto a vector MCM. Referring to Figures 2a and 2b the crossbar slices are mapped onto Island chips and the bus

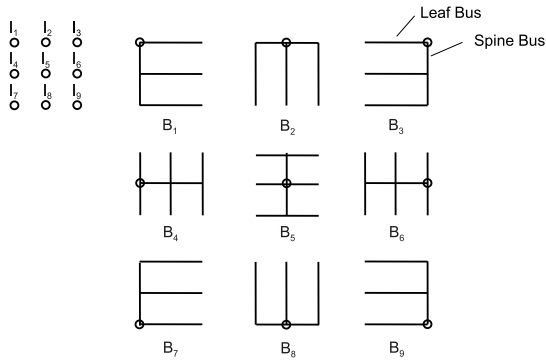


Fig. 4. Wiring layout for matrix MCM

segments are stitched together by links on the Bridge chips.

C. Matrix Switch

The sliced designs shown in Figures 2a and 2b seem to suggest an implementation through a linear array of chips only, but as we will show, it is equally straightforward to map these designs onto chips arranged in a matrix topology.

Mapping a sliced crossbar onto a matrix rather than a vector only changes the way we connect the bus segments. The logic that goes into the Island chips, that is, the logic needed for the switch ports and crosspoints is the same for a vector and a matrix. When looking at the matrix, the question is how we chain together the bus segments, now arranged in two dimensions.

There are many ways to connect the Island chips in a matrix so that the bus segments form a bus. Before we describe an example of such an interconnect strategy, we will outline some desirable properties it should have. The number of links the Bridge chips provide for connecting bus segments on the Island chips should be balanced, that is, there should be an equal number of horizontal and vertical links. Further, the bus segments should be linked in such a way as to provide the fewest hops between two non-neighboring Island chips. The strategy described below fulfills these requirements.

Figure 4 shows a possible strategy. The shown example uses a 3×3 matrix with a total of nine Island chips $I_i, i = 1..n, n = 9$. For this discussion, we assume that there is only one input and one output per Island chip. There is a bus B_i associated with each input. The figure shows the layout for each bus separately; the superimposition of all layouts represents the connectivity the Island and Bridge chips have to provide. Let us first have a look at bus B_1 connected to the input of Island chip I_1 . The rule that was applied to its construction is as follows. First, each row of chips is connected by a horizontal bus, that we refer to as a leaf bus. Next, the resulting three leaf busses are connected by a vertical bus, that we call a spine bus, in the column of the chips containing I_1 . This rule could be further applied to the construction of the remaining eight busses. However, this scheme would result in an asymmetric distribution of bridge links: each row of Island chips would contain $n = 9$ busses and each column would contain $\sqrt{n} = 3$

busses only. To avoid this imbalance, we alternate between two bus patterns that differ in how they favor horizontal and vertical busses. Thus, now looking at bus B_2 , we rotate the previously applied pattern by 90° and first connect all three columns of Island chips with vertical leaf busses and then put a horizontal spine bus through the row that contains Island chip I_2 . These two patterns can be alternately applied to the remaining seven busses.

The busses can be implemented through programmable wires. That is, much like in a field-programmable gate array, there are programmable switches in the Island chips that determine how the bus segments are connected to the links on the Bridge chips, and how inputs and outputs connect to the busses. Programming of the switches can, for example, be accomplished through a configuration stream at power up.

While the vector switch required n links between each pair of Island chips, it is less obvious how many links are required for the matrix switch. We can derive the number of links needed from the wiring rules just described. Let us first look at the number of links contributed by leaf busses. Roughly half of the busses use horizontal leaf busses and the other half uses vertical leaf busses—in Figure 4, the five busses $B_{1,3,5,7,9}$ use horizontal leaf busses and the four busses $B_{2,4,6,8}$ use vertical leaf busses. Thus, leaf busses require up to $\lceil n/2 \rceil$ links between neighboring Islands. Next, we want to determine the number of links needed for wiring the spine busses. Half of the spine busses are laid out horizontally and the other half vertically. Spine busses are evenly spread out over rows and columns, respectively. Thus, spine busses add up to $\lceil \frac{\lceil n/2 \rceil}{\sqrt{n}} \rceil$ links between neighboring Islands. In total, an Island requires a maximum of $\lceil n/2 \rceil + \lceil \frac{\lceil n/2 \rceil}{\sqrt{n}} \rceil$ links per edge. For a 3×3 matrix, this corresponds to 7 links.

Making the connections between the links on the Bridges and the bus segments on the Islands programmable offers some interesting opportunities. For example, by adding redundant links, erroneous links or even erroneous Island chips can be bypassed. Also, different configurations using smaller matrix assemblies that require different routing patterns could be accommodated.

D. Prototype Switch

We have built a small-scale switch prototype with the main goal to demonstrate that it is feasible to build a system enabled by PxC technology. There have been previous prototype implementations of PxC such as the one described in [10] that demonstrated data transmission over a small number of PxC channels between two manually aligned test chips. The prototype described here extends this work in several ways: (i) it uses larger chips that are more representative of real systems applications and that are more challenging from a mechanical and thermal perspective, (ii) it uses a larger chip assembly that consists of four Island chips and two Bridge chips², (iii) it uses a first-generation PxC package to align the chips, (iv)

²To simplify the mechanical design, we put two bridges into one long Bridge chip.

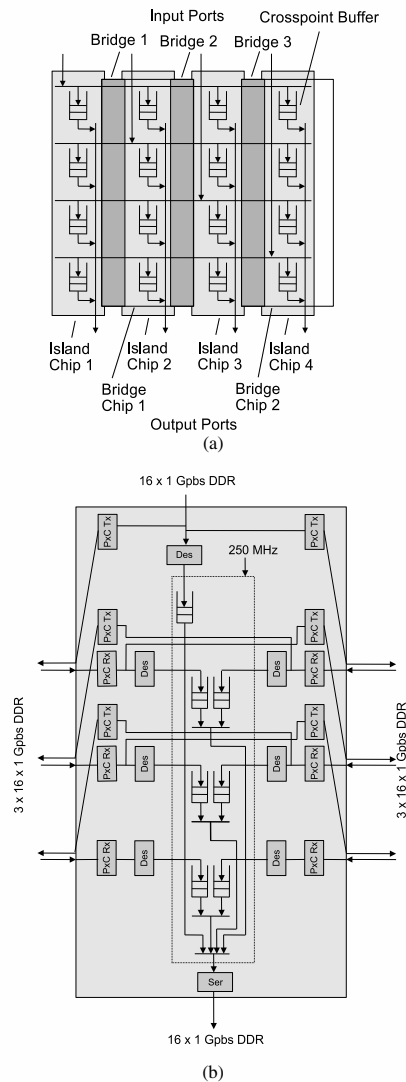


Fig. 5. Fully buffered crossbar (a) and Island chip (b)

it demonstrates an actual system application in the form of a switch. We briefly describe our prototype implementation here to convince the computer architect that PxC is feasible and thus enabling research into novel systems applications.

Since PxC technology is still in its infancy, the scale and performance of the switch implemented by the prototype is rather modest. In fact, realizing that the development of the PxC technology itself poses most challenges and risks we tried to keep the architecture of the switch itself as simple as possible. Now that we have demonstrated the feasibility of a PxC-based system, we are planning to build a larger switch that exploits the full potential of PxC.

To reduce development risks, technology has to be evolved in manageable steps. Looking at the alternatives outlined above, we felt that the vector MCM is the right first step as it simplifies the design and implementation in many ways. Applied to switch fabrics, the vector MCM already offers a significant increase in scale. Therefore, taking any additional risks that would result from developing the more ambitious

matrix MCM is not justified at this time.

Our prototype switch implements a layer-2 Ethernet switch with four 10 Gbit/s ports. The internal architecture resembles a fully-buffered crossbar. The 4 x 4 crossbar is vertically sliced such that each slice or Island chip implements four crosspoints, one input port and one output port. Figure 5a shows the logical organization of the switch partitioned into four slices or Island chips connected by three bridges (located on two Bridge chips). A more detailed block diagram of the Island chip is given in Figure 5b. The Island chip connects to the PCB through two LVDS links: a 16-bit wide data input path for the input port and a 16-bit wide data output path for the output port. There are three pairs of 16-bit wide PxC interfaces on the left side and the right side of the chip to connect to the neighboring Island chips—each input bus is implemented by two unidirectional busses. LVDS links as well as PxC links are clocked at 500 MHz DDR providing a data rate of 1 Gbit/s. Since the internal data paths are clocked at 250 MHz, which is a quarter of the external data rates, there are deserializers and serializers around the chip core. Since data transmission over the PxC links has to be DC-balanced, packets are 8B/10B-encoded when they enter an Island chip. The resulting 25% encoding overhead is compensated by the speedup provided by the PxC links running at 16 Gbit/s.

The core of the Island chip is mainly made up of buffer memories. Each crosspoint connects to two 8 kB buffers, one for each direction of the input bus. Though only one of the two buffers is used (as determined by the relative position of the input port connected to it), we chose this design as we were not constrained by available chip resources and as it simplified the design by avoiding a multiplexer at the input of the crosspoint buffer selecting between an input port on the left and an input port on the right, each providing its own clock domain.

An Island chip contains several clock domains. There is a separate clock domain associated with each input port and each output port with clock boundaries going through the crosspoint buffers. A clock domain given by an input port spans all four Island chips in that data and clock are forwarded from an input port to all four Island chips. Clock forwarding is attractive as it avoids any re-synchronization with a local clock. This clocking scheme does not scale well, though, and can only be applied to a small-scale MCM such as the prototype switch described here as skew between data and clock accumulates along the transmission path to the point where the clock edges do not form a reliable reference for sampling the data³. In addition to the four input clock domains, each Island chip also contains an output clock domain that covers the output port logic.

The switch forwards Ethernet packets up to a length of 1,500 bytes. A header is prepended to the packet that specifies the output port the packet is destined for. The input bus is a data pipeline that works like a broadcast bus in that every crosspoint buffer connected to it reads the header to decide

³There are, of course, other clocking techniques that scale better. For example, data could be resynchronized in every Island chip.

whether the packet has to be enqueued. Each buffer provides an xon/xoff flow control signal that tells the corresponding input port whether there is room for a packet or not. Each output port has a simple round-robin arbiter that decides which crosspoint buffer has to be dequeued.

To make it possible for the same chip design and IO layout to be used at any position within a vector, the input bus segments cannot be connected by straight horizontal wires. If a wiring layout was chosen as shown in Figure 2b, the input ports would have to connect to different horizontal busses requiring four different layouts for the Island chips. Figure 5b shows a layout for connecting the bus segments that only requires one type of chip layout. A staggered wire layout is chosen that makes it possible to always connect an input port to the top bus segment. This segment then steps down to the next lower bus segment in its neighboring Island chips.

E. Advantages of a Flat Switch

PxC offers the opportunity to realize a large-scale switch as a 1-stage network, which we also refer to as a flat switch, without having to resort to a MIN. A flat switch offers many advantages including low uniform forwarding delay, lack of tree saturation, scalability, reduced component count, reduced power consumption, and higher reliability.

Since the forwarding delay in a multistage network is typically proportional to the number of stages, latency for a flat switch can be significantly lower than for a MIN. In MINs, delays are not only longer, they can also vary. For example, in the Clos network shown in Figure 6, transfers take less time if nodes are connected to the same leaf switch. Furthermore, many MINs are susceptible to hot spots that cause further increases in forwarding delays. Even worse, hot spots lead to tree saturation that can severely impact overall network performance. These effects not only increase forwarding delays, they also make it difficult to provide any guarantees about delays and variations in delays. In contrast, it is much easier to architect a flat switch in such a way that a congested output does not affect traffic flow to other outputs.

Often, it is a requirement for a design and architecture to cover a wide and contiguous range of switch sizes (where size refers to the number of ports). A sliced crossbar scales well as it allows for building switches with any number of slices, of course, up to a given maximum number of slices. This is not true for MINs that provide little flexibility in sizing a design. The size of a MIN is determined by the radix of the switch element, the number of stages, and the network topology. Typically, the radix of the switch element and the network topology are given, allowing the designer to only change the number of stages. Thus, only one configuration size per number of stages is optimal—of course, other configuration sizes are possible if a designer is willing to not use some of the ports of the switch elements. It should also be noted that the performance and latency characteristics of these configurations can vastly differ as the topologies look quite different.

Finally, a flat switch requires fewer switch elements than a MIN of the same size. For example, the 3-stage 288-

port network shown later in Figure 6a requires 36 switch elements whereas a comparable PxC switch requires only 12 switch elements (each implementing 24 ports). As the radix is increased, the ratio of switch elements needed will favor the PxC switch over the MIN more and more. Reducing component count not only reduces cost, other benefits are reduced power consumption and higher reliability.

Though we have given a qualitative analysis of the advantages a flat switch offers, we want to extend our discussion in the next section with a thorough simulation and quantitative examination of the performance of a flat switch in comparison with a MIN. We will look at two switch configurations that represent large-scale switch deployments and will analyze their performance under different types of load.

IV. PERFORMANCE EVALUATION

In this section we evaluate the potential of 1-stage high-radix switch architectures enabled by PxC. We compare their performance to the one of MINs with the same number of end nodes. This evaluation is based on simulation results obtained for different traffic patterns and different network configurations. In the following subsections, we first describe the simulation model, then the different network configurations and traffic patterns used in the experiments, and, finally, we provide the simulation results and analyze them in detail.

A. Simulation Model

We have used a detailed event-driven simulator that allows us to model the network at the register transfer level. The simulator models both flat high-radix (PxC) switches and MINs consisting of multiple switches, end nodes, and links.

For comparison purposes, we evaluate the behavior of different 1-stage PxC switch and MIN configurations. For both configurations, we consider two network sizes: 288 and 1,728 end nodes, respectively. The MIN configurations are built out of 24-port switches. This switch configuration represents the largest single-chip commercial switch available today (e.g. the Mellanox Infiniscale III switch chip has 24 ports). Specifically, the following MIN configurations are analyzed:

- A (12, 12, 24) Clos network (Figure 6a): 288 end nodes connected by a 3-stage Clos network with 24 switches in the combined first/third stage and 12 switches in the middle stage. 12 ports are connecting each switch in the first/third stage to the middle stage. To sum up, this network consists of a total of 36 switches, each with 24 bidirectional ports. Notice that the figure shows a folded network.
- A 24-ary 5-fly MIN (Figure 6b): 1,728 end nodes connected by a 5-stage network (notice that the figure shows a folded network). The central part of the network is made up of 6 logical 288-port switches. Each of these switches represents a 24-ary 3-fly Clos network (previous configuration). In total, this network consists of 360 switches. Notice that the overall network is not strictly a Clos network since it uses twice the number of links between the first two stages.

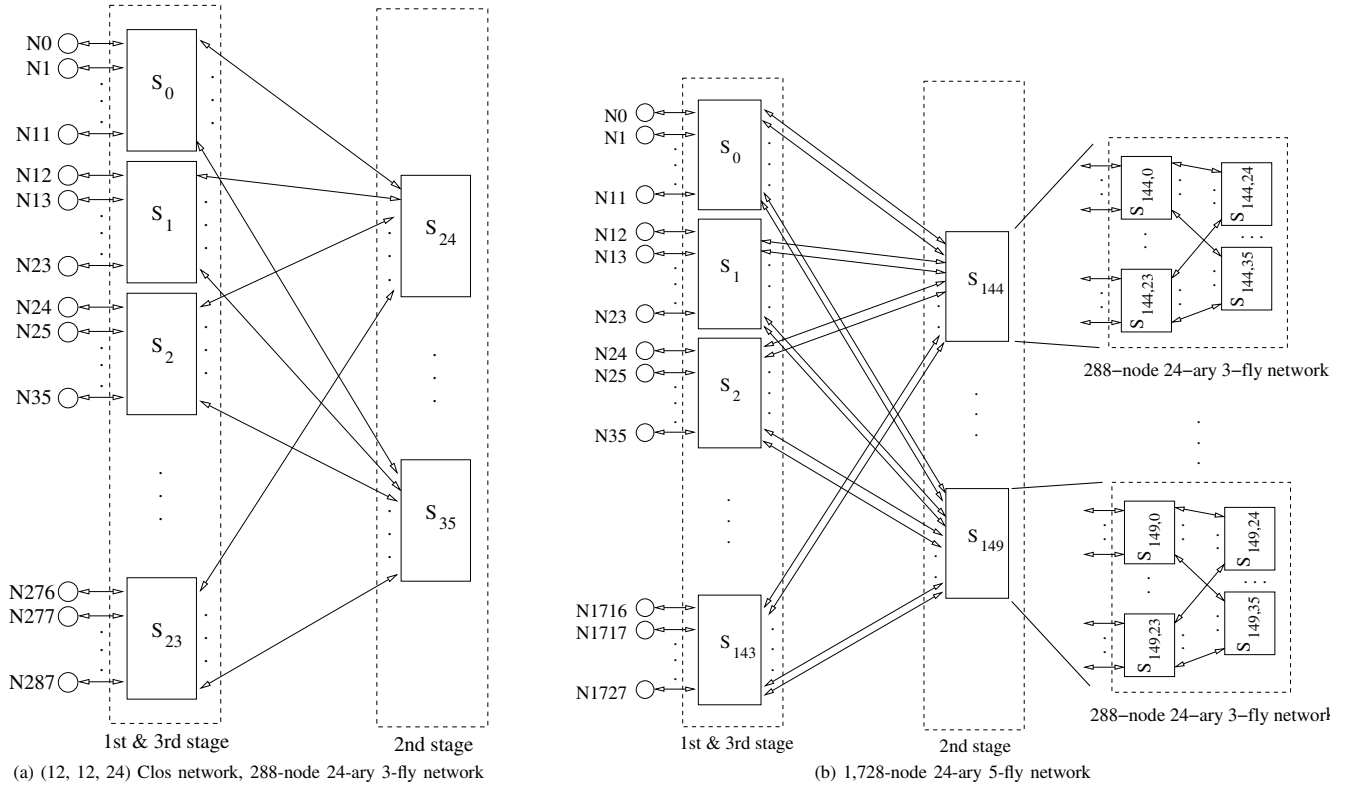


Fig. 6. Modeled multi-stage networks

The MIN configurations were compared with the following two PxC switch models:

- A 288 x 288 crossbar sliced into 12 24-port chips. The chips are packaged in a vector MCM similar to the one shown in Figure 3a.
- A 1,728 x 1,728 crossbar sliced into 16 chips packaged in a 4 x 4 matrix MCM as shown in Figure 3b. This configuration requires a much higher port density of 108 ports per chip. As mentioned in Section III-C, we anticipate that future IO technologies for PxC MCM will provide the necessary bandwidth.

In all configurations, deterministic routing is used. However, for MINs, special care has been taken to balance the traffic. In order to do so, each switch at the first stage computes the output port for each packet using the modulo function of the destination ID of the packet ($destination_ID \bmod \#output_ports$). For example, for the 288-node network, each output port of each switch at the first stage (12 ports leading to the second stage) routes packets for a disjoint group of 23 destinations (the remaining 12 destinations are attached to the switch). Once packets arrive at the second stage, they can only follow a specific route since there is only one path from every switch at the second stage to any destination. For the 1,728-node MIN a similar routing scheme is assumed. This scheme is in fact an adaptation of the one proposed in [9], which guarantees that packets addressed to different destinations do not share links from the middle stage onwards, thereby balancing traffic very efficiently.

To model the links in all the configurations⁴, we assume serial full-duplex pipelined injection links. We model links with 10 Gbit/s bandwidth, assuming a 4x InfiniBand link. Credit-based flow control is implemented at the queue level. Flow control packets are modeled and they share the link bandwidth with data packets.

Memories at switches are modeled in all the cases for both input and output ports. The PxC switch is modeled assuming two memory configurations at every input or output port. In the first configuration we use a Virtual Output Queuing mechanism with as many queues as output ports. Each queue is four packets deep - we assume fixed-size packets with a length of 128 bytes. This leads to $288 \times 4 \times 128$ bytes (144 kB) for the 288-node PxC switch and $1,728 \times 4 \times 128$ bytes (864 kB) for the 1,728-node PxC switch. The second PxC memory configuration is only applied to the 1,728-node PxC switch, and in this case smaller port memories are used for reducing the overall switch memory requirements. Specifically, only 64 queues each four packets deep are used at each port (32 kB). As the number of destinations is higher than the number of queues we use a modulo mapping policy to compute the queue for an incoming packet as described in [7]. This configuration will be referred to as PxC-MM.

In the case of MIN configurations, we use the same memory size of 144 kB used in the 288-node PxC switch for input and output ports. This accounts for implementing, at every switch

⁴Though PxC switch configurations do not require internal cabling, cabled links are still needed for connecting to end nodes.

port, 24 queues with 48 credits for 128-byte packets, thus allowing a VOQ mechanism at the switch level. Therefore, HOL blocking at the switch level is eliminated.

At every switch (in both MIN and PxC switch configurations), packets are forwarded from any input queue to any output queue through a multiplexed crossbar. Only one connection is allowed from every input port to every output port. Therefore, neither input speedup nor output speedup is implemented. Thus, we have considered a crossbar bandwidth equal to a link bandwidth of 10 Gbit/s.

The access to the crossbar is controlled by an arbiter that receives requests from packets at the head of any input queue. A requesting packet is forwarded only when the corresponding crossbar input and crossbar output are free. At output ports, another arbiter selects the output queue for injecting packets to the output link. This selection is made following a round-robin scheme.

End nodes are connected to switches using Input Adapters (IAs). Every IA is modeled with a fixed number of N message admittance queues following a VOQ scheme. When a message is generated, it is stored completely in the admittance queue assigned to its destination, and is packetized before being transferred to an injection queue. The transfer from admittance queues to injection queues is controlled by an arbiter that follows a round-robin scheme. The injection of packets to the network is also controlled by an arbiter that selects the next packet to be transmitted, using a round-robin scheme among all the injection queues.

In the case of MINs, we have based the main simulation timing parameters on the specifications of the Infiniscale III 24-port Mellanox switch [13]. In particular, port-to-port intra-switch latency is set to 200 ns and total link delay is 19 ns (assuming total link length⁵ is 9.5 ft for a single board switch design) for the 24-ary 3-fly network (288-node system) and 27 ns (assuming link length is 13.5ft for a midplane chassis design) for the 24-ary 5-fly network (1,728-node system).

On the other hand, regarding single-stage PxC switch timing parameters, we assume 14 ns as the total wire delay (total wire length is 7 ft) for the 288-node switch and 20.5 ns (total wire length is 10.25 ft) for the 1,728-node switch. The latency contributed by the switch fabric consists of a data path delay and an arbitration delay. The path delay is 140 ns for the 288-node switch and 95 ns for the 1,728-node switch. The path delays include forwarding delays through Island and Bridge chips. We based our simulation model on the switch architecture described in [15], [17] and, in particular, adapted our simulator to model variable arbitration delays, that is, arbitration delays that vary as a function of load.

B. Traffic Load

Table I shows the different traffic patterns used for the evaluation. In the first case (case #1) a uniform distribution of packet destinations is used, that is, all end nodes generate traffic addressed to random destinations (except to themselves).

In cases #2a, #3a, #4a, and #5a, different hot spot traffic patterns are used. In each case, a different percentage of end nodes generate traffic only to a unique destination, while the remaining end nodes generate traffic with random destination addresses. Alternatively, a different hot spot traffic pattern is used in cases #3b, #4b, and #5b. In these cases all end nodes generate both uniform traffic and hot spot traffic. For each case, different percentages of hot spot traffic are generated at the source nodes and addressed to a unique destination node, while the destinations of the rest of the traffic are randomly selected. It is expected that the latter cases will induce a much more severe congestion situation in the network.

We evaluated the full range of traffic, from low load to network saturation. We ran different simulations for each load, taking values for calculating the considered metrics only after a certain time for allowing the network to reach traffic stability. For each simulation, the generated traffic was considered (and plotted in the figures) as a percentage of network capacity. On the other hand, we calculated average packet header latency as the main performance metric. Packet header latency is the time (expressed in ns) the packet header is in the network (from injection to delivery).

C. Evaluation Results

Figure 7 shows the evaluation results for both the MIN network and the PxC switch for 288 and 1,728 ports, respectively, using traffic case #1 (random traffic pattern). As a first observation, average packet header latency from low load to medium load is significantly lower for the PxC switch than for the MIN in both configurations. Indeed, zero load latency is significantly lower. For the PxC switch, end-to-end latency is 195 ns for the 288-node system and 254 ns for the 1,728-node system, with arbitration delays contribution the largest part. In the multi-stage case, however, latency for low load is around 600 ns for the 288-node system and 1 μ s for the 1,728-node system. In the MIN, the number of hops dominates the latency equation. For the 288-node MIN, 23 out of 24 packets travel three hops within the network (200 ns each) and the remaining packet takes a one-hop path. This leads to 600 ns latency, on average. However, once the system reaches half the total network capacity the average packet latency of the PxC switch becomes slightly higher than the one achieved in the MIN. This is due again to the complexity of the arbiter, that dominates the latency equation for the PxC switch. This effect is more noticeable in the 1,728-node system, with higher arbitration delays.

On the other hand, the achieved network throughput is around 90% for both configurations. This is an obvious result since both configurations (PxC switch and MIN) are well balanced and traffic is uniform. However, as can be seen for the larger network configuration, the PxC switch architecture does not achieve the same throughput as the MIN. Again, the complex scheduler is introducing some degree of blocking. An interesting point in the evaluation is the performance of the 1,728-node PxC switch with a reduced set of queues (PxC-MM). With only 64 queues at each port (only 32 kB per port),

⁵Total link length includes all wiring between an input and an output port.

TABLE I
TRAFFIC PATTERNS EVALUATED FOR THE 288-NODE AND 1,728-NODE CONFIGURATIONS

Case	network	% sources	destination
#1	288- 1,728-node	100%	randomly among N-1 end nodes
#2a	1,728-node	99%	randomly among N-1 end nodes
		1%	one destination
#3a	288- 1,728-node	95%	randomly among N-1 end nodes
		5%	one destination
#4a	288- 1,728-node	90%	randomly among N-1 end nodes
		10%	one destination
#5a	288-node	80%	randomly among N-1 end nodes
		20%	one destination
#3b	288-node	100%	95% of packets randomly among N-1 end nodes 5% of packets to one destination
#4b	288-node	100%	90% of packets randomly among N-1 end nodes 10% of packets to one destination
#5b	288-node	100%	80% of packets randomly among N-1 end nodes 20% of packets to one destination

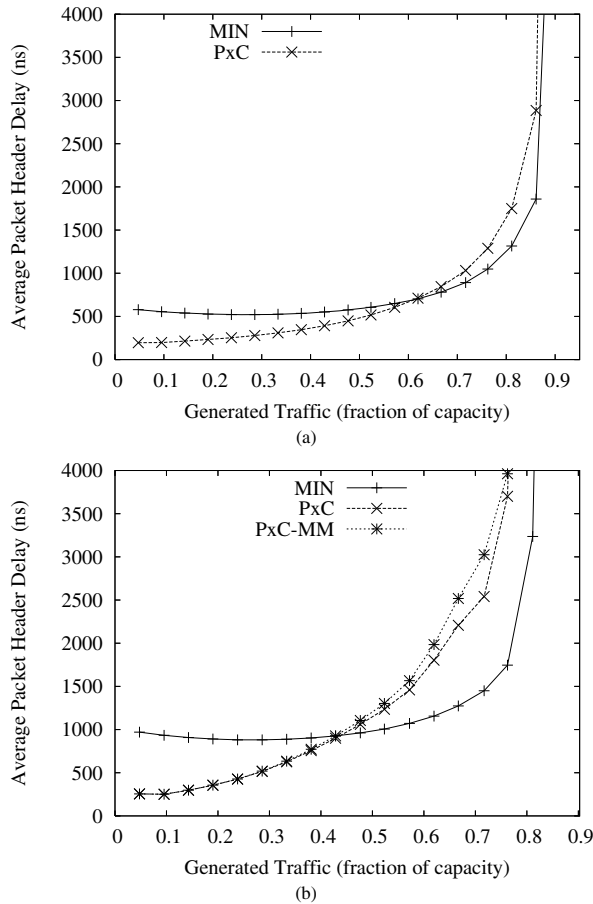


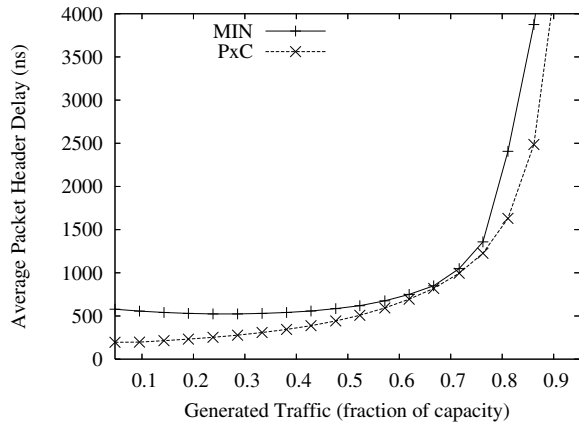
Fig. 7. Performance results for uniform traffic case #1 (288-node (a) and 1,728-node (b) systems)

the switch is able to get performance numbers very close to the PxC switch with a VOQ scheme (1,728 queues per port). Even for low load, the average packet header latency does not suffer from a reduced set of queues.

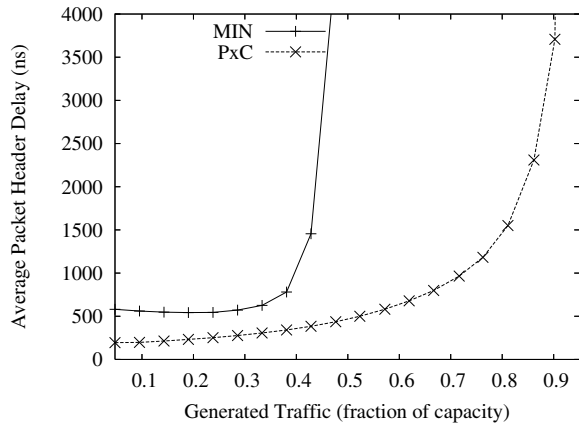
Figure 8 shows more interesting results for non-uniform traffic. In particular, for the 288-node system, different percentages of hot spot traffic generated at a separate set of source nodes (traffic cases #3a, #4a, and #5a). The first interesting observation is that low load latencies are still much better for the PxC switch. The packet latency experienced in the MIN network is simply halved in comparison with the PxC switch. Throughout all the load range, latency is smaller with the PxC switch. The MIN is experiencing a high degree of network contention. For the 5% hot spot case, the MIN is able to achieve near maximum network throughput keeping packet latency reasonably low. However, as the uniform distribution of packet destinations dilutes in the network (higher hot spot percentages), the MIN tends to rapidly saturate. In particular, for the 10% and 20% hot spot cases, the network throughput is only 40% of the network capacity. On the other hand, the PxC switch achieves roughly maximum network throughput regardless of the hot spot intensity as it uses VOQ at every input port of the switch.

Figure 9 shows the performance achieved for the 1,728-node systems for the same hot spot traffic (with a different percentage of source nodes generating hot spot traffic). As can be seen, differences between MIN networks and the PxC switch are even greater. Indeed, for the 5% and 10% hot spot case, the MIN network simply collapses, even with extremely low load (zoom figures are provided in Figure 11). This happens because, as the network increases in size, the number of stages increases as well (5 stages in the 1,728-node MIN). As a result, HOL blocking introduced by congested packets is higher (regarding the time packets are blocked) and may appear in more points inside the network. Additionally, the number of sources contributing to congestion increases as well. Notice that the PxC switch also performs well when using a reduced set of queues per port (PxC-MM). Indeed, even with a modest number of queues the PxC switch still achieves good performance levels in terms of latency and throughput.

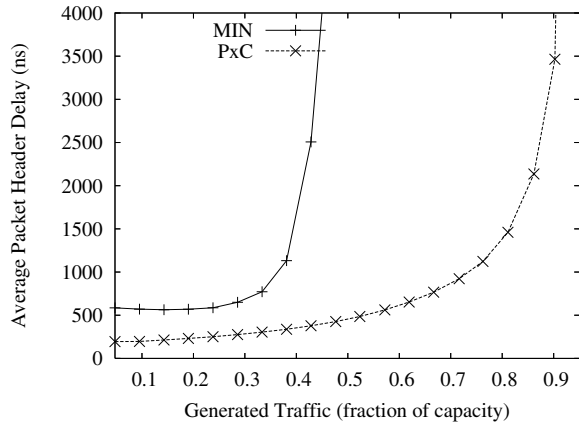
Making congestion even worse, we observe even greater



(a) 5% hot spot traffic



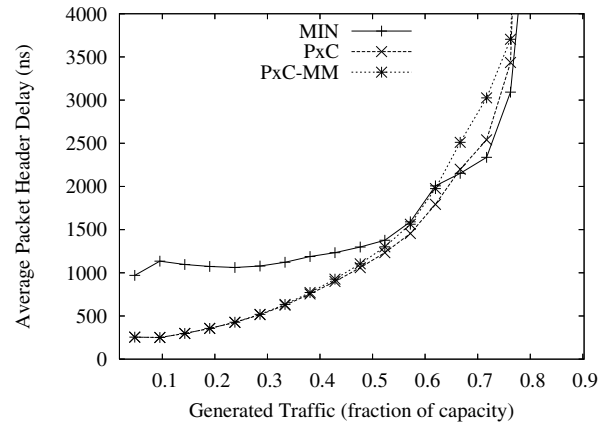
(b) 10% hot spot traffic



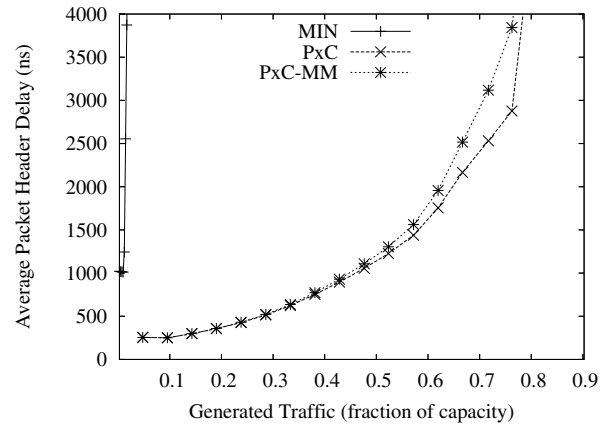
(c) 20% hot spot traffic

Fig. 8. Performance results for traffic cases #3a, #4a, and #5a (288-node system)

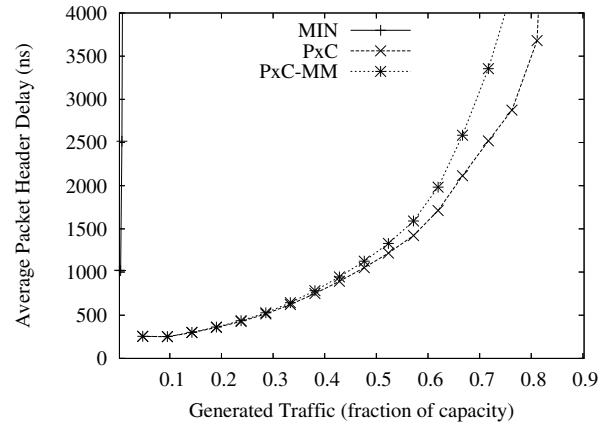
differences. Figure 10 shows results for the 288-node system (a zoom figure is provided in Figure 11). In this traffic case, all the end nodes generate a small percentage of traffic addressed to a unique destination. As can be seen, the MIN network rapidly saturates, even with a small amount of hot spot traffic (5%). As the output becomes oversubscribed, the “branches” of the resulting “congestion tree” reach all the sources. Thus, all packets experience congestion as soon as they are injected into the network. Hence, the effect of a congestion tree is much



(a) 1% hot spot traffic



(b) 5% hot spot traffic



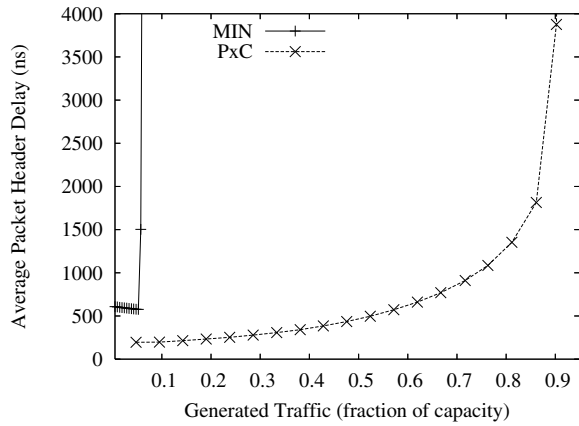
(c) 10% hot spot traffic

Fig. 9. Performance results for traffic cases #1a, #2a, and #3a (1,728-node system)

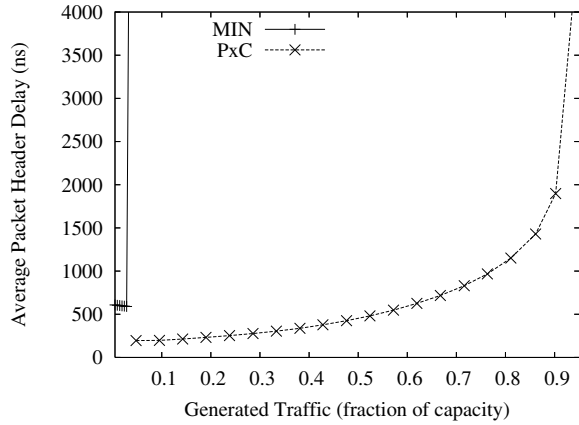
more severe. With this traffic pattern, network throughput is one order of magnitude higher for the PxC switch.

V. CONCLUSIONS

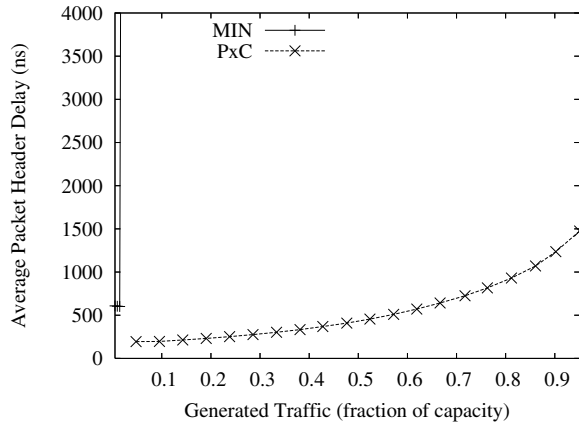
PxC technology changes the way we build large-scale systems based on MCMs. Offering many times more chip-to-chip bandwidth than off-module bandwidth, PxC forces the designer to rethink how systems are partitioned. In this paper, we looked at large switch fabrics and how PxC changes their



(a) 5% hot spot traffic



(b) 10% hot spot traffic

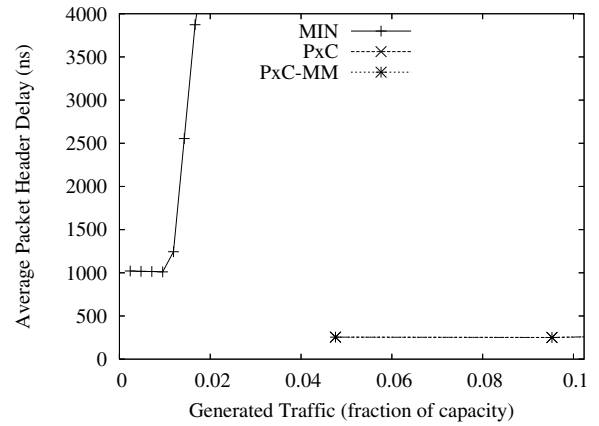


(c) 20% hot spot traffic

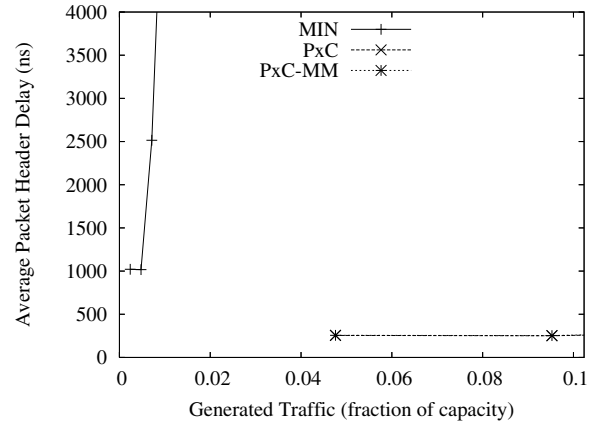
Fig. 10. Performance results for traffic cases #3b, #4b, and #5b (288-node system)

architecture and implementation. We realize that high-radix switches can now be implemented as a port-sliced crossbar and no longer require MINs.

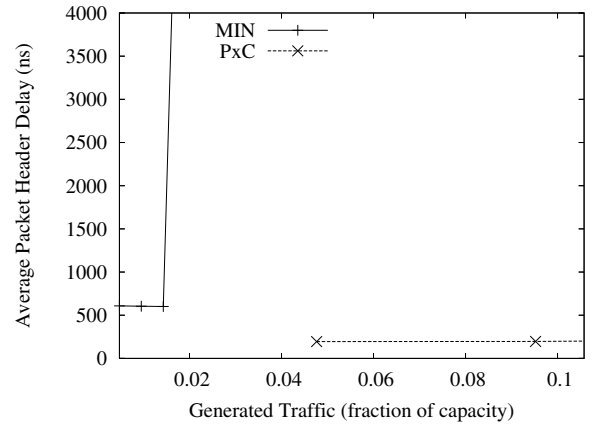
Evaluation results comparing different network sizes demonstrate the benefits of building a high-radix switch based on PxC technology. MINs tend to saturate quite rapidly as the uniformity of the traffic is reduced. As the network size increases, the number of stages increases as well, thus, the effect of congestion is becoming much more severe. On the



(a) 5% hot spot traffic, 1,728-node system



(b) 10% hot spot traffic, 1,728-node system



(c) 20% hot spot traffic, 288-node system

Fig. 11. Zoom figures for cases #2a and #3a (1,728-node system), and case #5b (288-node system)

other hand, congestion in a flat PxC switch doesn't spread as all the packets are stored in VOQs at the network level (this is even true in a reduced set of modulo-mapping queues) and there are no intermediate shared queues.

Additionally, the base latency of packets is much higher in MIN networks as the latency equation is dominated by the hop count. In contrast, for a flat PxC switch all the packets have a one-hop latency.

On the down side, the architecture for the flat PxC switch

still needs improvement. Under high load, arbitration delays become significant dominating overall switch latency. The reason is that efficient arbitration for a high-radix switch takes time. This issue certainly needs to be addressed by future work.

REFERENCES

- [1] F. Abel, C. Minkenberg, R. Luijten, M. Gusat, I. Iliadis, "A four-terabit packet switch supporting long round-trip times," *IEEE Micro*, vol. 23, no. 1, pp. 10-24, Jan./Feb. 2003.
- [2] ATCA Web Site, www.advancedtca.org.
- [3] V. Beneš, *Mathematical Theory of Connecting Networks and Telephone Traffic*, New York, Academic Press, 1965.
- [4] C. Clos, "A Study of Non-blocking Switching Networks," *Bell System Technical Journal*, vol. 32, no. 5, pp. 406-424, March 1953.
- [5] R. Drost, R. Hopkins, I. Sutherland, "Proximity Communication," *IEEE Custom Integrated Circuits Conference*, pp. 469-472, Sep. 2003.
- [6] R. Drost, R. Hopkins, I. Sutherland, "Electronic Alignment for Proximity Communication," *IEEE Solid-State Circuits Conference*, pp. 144-518, Feb. 2004.
- [7] J. Duato, J. Flich, T. Nachiondo, "Cost-Effective Technique to Reduce HOL-blocking in Single-Stage and Multistage Switch Fabrics," *Euromicro Conference on Parallel, Distributed and Network-based Processing*, pp. 48-53, Feb. 2004.
- [8] Fulcrum Web Site, <http://www.fulcrummicro.com/products/focalpoint/fm2224.htm>.
- [9] C. Gomez, F. Gilabert, M.E. Gomez, P. Lopez and J. Duato, "Deterministic versus Adaptive Routing in Fat-Trees," *Workshop on Communication Architecture for Clusters*, pp. 1-8, March 2007.
- [10] D. Hopkins, A. Chow, R. Bosnyak, B. Coates, J. Ebergen, S. Fairbanks, J. Gainsley, R. Ho, J. Lexau, F. Liu, T. Ono, J. Schauer, I. Sutherland, R. Drost, "Circuit Techniques to Enable 430Gb/s/mm² Proximity Communication," *IEEE International Solid-State Circuits Conference*, pp. 368-369, Feb. 2007.
- [11] J. Kim, W. Dally, B. Towles, A. Gupta, "Microarchitecture of a High-Radix Router," *IEEE/ACM 32nd International Symposium on Computer Architecture*, pp. 420-43, June 2005.
- [12] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, M. Horowitz, "Tiny Tera: a Packet Switch Core," *IEEE Micro*, vol. 17, no. 1, pp. 26-33, Jan./Feb. 1997.
- [13] Mellanox Web Site, http://www.mellanox.com/products/switch/_silicon.php.
- [14] G. Mora, J. Flich, J. Duato, E. Baydal, P. Lpez, O. Lysne, "Towards an Efficient Switch Architecture for High-Radix Switches," *ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, Dec. 2006.
- [15] W. Olesinski, H. Eberle, N. Gura, "OBIG: the Architecture of an Output Buffered Switch with Input Groups for Large Switches," *IEEE GLOBECOM 2007*, Nov. 2007.
- [16] W. Olesinski, H. Eberle, N. Gura, "PWWFA: The Parallel Wrapped Wave Front Arbiter for Large Switches," *IEEE Workshop on High Performance Switching and Routing*, May/June 2007.
- [17] W. Olesinski, N. Gura, H. Eberle, A. Mejia, "Low-Latency Scheduling in Large Switches," *ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, Dec. 2007.
- [18] S. Scott, D. Abts, J. Kim, W. Dally, "The BlackWidow High-Radix Clos Network," *IEEE/ACM 33rd International Symposium on Computer Architecture*, pp. 16-28, June 2006.
- [19] Y. Tamir, H. C. Chi, "Symmetric Crossbar Arbiters for VLSI Communication Switches," *IEEE Transactions on Parallel and Distributed Systems*, vol. 4, issue 1, pp. 13-27, Jan. 1993.
- [20] Sun Microsystems, Inc., "The Path to Massive Scale High Performance Computing," www.sun.com/products/networking/datacenter/ds3456.