Internet-Router Buffered Crossbars Based on Networks on Chip

Kees Goossens[1,2] Lotfi Mhamdi[2] Iria Varela Senín[2]
[1] NXP Semiconductors, Eindhoven, The Netherlands
[2] Computer Engineering, Delft University of Technology, The Netherlands
kees.goossens@nxp.com *lotfi@ce.et.tudelft.nl*

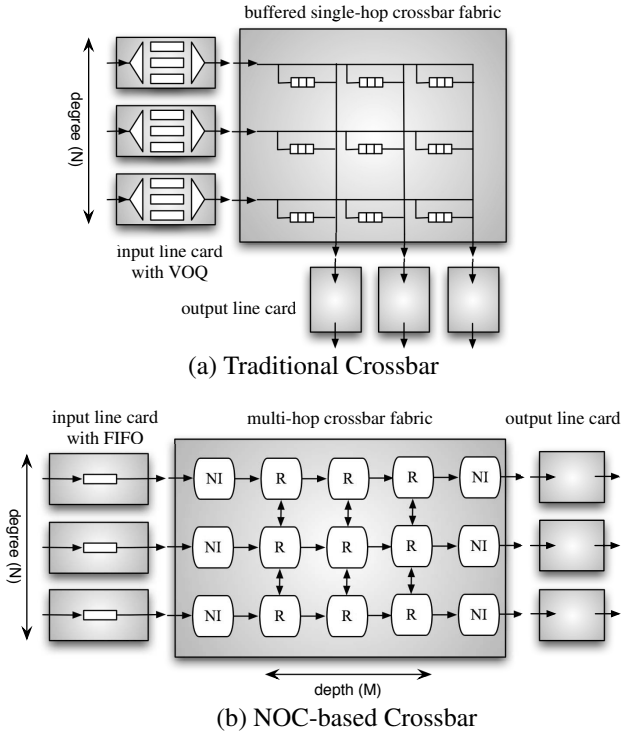
***Abstract*—The scalability and performance of the Internet depends critically on the performance of its packet switches. Current packet switches are based on single-hop crossbar fabrics, with line cards that use virtual output-queueing to reduce head-of-line blocking. In this paper we propose to use a multi-hop network on a chip (NOC) as the crossbar fabric, with FIFO-queued line cards. The use of a multi-hop crossbar fabric has several advantages. 1) *Speed-up*, i.e. the crossbar fabric can operate faster because NOC inter-router wires are shorter than those in a single-hop crossbar, and because arbitration is distributed instead of centralised. 2) *Load balancing* because paths from different input-output port pairs share the same router buffers, unlike the internal buffers of buffered crossbar fabric that are dedicated to a single input-output pair. 3) *Path diversity* allows 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 patterns. 4) *Simpler line-card design*: the use of FIFOs on the line cards simplifies both the line cards and the (inter-chip) flow control between the crossbar fabric and line cards, reducing the number of (expensive) chip pins required for flow control. 5) *Scalability*, in the sense that the crossbar speed is independent of the number of ports, which is not the case for single-hop crossbar fabrics. We analyzed the performance of our architecture both analytically and by simulation, and show that it performs well for a wide range of traffic conditions and switch sizes. Additionally we prototyped a 32×32 NOC-based crossbar fabric in a 65nm CMOS technology. The unoptimised implementation operates at 413 MHz, achieving an aggregate throughput in excess of 10^{10} ATM cells per second.**

I. Introduction

Current high-performance Internet packet switches are based on crossbar fabrics [1], [2], and come in two flavours: unbuffered and internally buffered. The general architecture of a traditional packet switch is shown in Figure 1(a). Packets or ATM cells arrive in the line cards, where they are buffered until they are accepted by the crossbar fabric. Then they are switched to the appropriate output line card, where they are buffered before being forward to the next packet switch. Flow control between line cards and packet switch ensures that no buffers overflow.

Both unbuffered and buffered crossbar fabrics grow quadratically in the number of ports. To avoid head-of-line (HOL) blocking on the line cards, which severely reduces performance, both require virtual output queueing (VOQ) [3]. VOQ is complex to implement because multiple virtual FIFO queues must be administrated on the line cards, and requiring more flow control signals between the line card and the crossbar fabric.

The unbuffered crossbar fabric is cheaper than its buffered counterpart since it contains no internal buffers. However, it requires a centralized and complex scheduler to configure the crossbar matrix for packet transfer from the input ports to the output ports of the switch [3]. Long, and hence slow, wires are required to connect the crossbar input ports to output ports, and input/output ports to the scheduler.



(a) Traditional Crossbar

(b) NOC-based Crossbar

Figure 1. Packet switches.

A buffered crossbar fabric (known as combined input-crosspoint queued, or CICQ crossbar) overcomes the scheduling complexity of traditional crossbars by means of distributed and parallel schedulers, one per port of the crossbar [4]. However, it requires dedicated buffers for each pair of input-output ports of the crossbar fabric. Wires are still long and slow to connect each input port to an output port via the intermediate buffer, and to their corresponding scheduler.

In this paper, we propose to implement the crossbar fabric

as a *multi-hop network on a chip* (NOC), as shown in Figure 1(b). This offers several advantages when compared to traditional single-hop crossbar-based fabrics. First, in a multi-hop interconnect the wires between NOC routers are short, especially when only neighbouring routers communicate, such as in a mesh topology used here. Additionally, because arbitration is distributed over the routers, the long wires from crossbar input/output ports to the central scheduler (for unbuffered crossbar) or scheduler per input-output port pair (for buffered crossbar) are eliminated. The arbiters also have fewer inputs (3-5: the degree of the router) than before ($2N$, corresponding to all input-output port pairs for an unbuffered crossbar, or N corresponding to all input ports for an output port for a buffered crossbar). As a result, the crossbar fabric can operate at a higher speed. This so-called *speed-up* strongly improves performance, as we will show in Section VI.

Second, our design is *scalable* in the sense that its speed is independent of the crossbar size, unlike single-hop crossbar fabrics that employ long wires. Instead it is limited by the speed of the (small) routers and their short local wires. Section VI comments on the scalability w.r.t. throughput and latency.

Third, buffered crossbar fabrics dedicate a buffer to each input-output port pair. In contrast, in a NOC each buffer in a router can be used by packets of different input-output port pairs. This sharing results in *load balancing of buffer space* between different flows, and possibly in smaller buffers.

Fourth, a multi-hop NOC has *path diversity*, i.e. a packet can use different paths from an input port to an output port. This results in further load balancing especially in the presence of non-uniform traffic patterns.

Finally, single-hop buffered crossbar fabrics suffer from head-of-line (HOL) blocking, where packets that are waiting in the line card because their output port is not available block other packets queued behind them in the same line card, even though they are destined for another free output port. Virtual output queueing (VOQ) addresses this problem by replacing, on each line card, a single FIFO by a (virtual) FIFO per output port, as shown in Figure 1(a). Multi-hop NOC-based crossbar fabrics do not suffer from HOL blocking because the NOC is pipelined (multi-stage). As a result, packets from a single input port heading for different output ports are accepted by the NOC, even when some of the output ports may be blocked. The path diversity and buffer load balancing ensure that packets destined to different output ports interfere much less with each other. A NOC-based crossbar fabric can therefore use *FIFO buffering on the line cards*, as shown in Figure 1(b) instead of VOQ buffers. FIFO buffers are simpler to implement than VOQ buffers. Additionally, the FIFO flow control to avoid buffer over/underflow between the crossbar fabric chip and the line cards communicates the available buffer space. For VOQ buffers, the VOQ identifier must also be communicated. This either requires additional chip pins, or a higher pin frequency.

A NOC-based crossbar fabric also has some potential disadvantages: 1) distributed buffering may be more expensive than centralized buffering. Large SRAMs occupy a smaller area per bit than small SRAMs and register-based FIFOs. 2) Distributed arbitration is suboptimal compared to global arbitration. 3) Multi-hop interconnects have a higher latency than a single-hop crossbar.

This paper is structured as follows. In Section II we discuss related work with respect to both Internet router design and NOC design. In Section III we introduce our NOC-based crossbar fabric, including its novel routing algorithm. We perform an analytical throughput analysis of the architecture in Section IV. Section V introduces a hardware implementation, followed by a detailed simulation-based throughput and latency analysis for different traffic types and speed-ups in Section VI. We show cost:performance results, and compare the performance to that of a traditional CICQ switch. We conclude in Section VII.

II. Related Work

The packet switch fabric is the key component of modern Internet routers. The most common fabric architectures in use today are bus-based [5], shared memory [6] and crossbar [3]. The crossbar fabric has the best performance [3]. It is non-blocking, inherently supports multicast, and provides parallel point-to-point communication making it attractive for real-time applications. The main challenge in the design of a crossbar is its scalability in terms of data rate and number of ports. The cost of a crossbar fabric grows as the square of the number of its ports. Additionally, for a crossbar with a medium to large number of ports, it is difficult to achieve a high data rate due to the dominant delay incurred by the long point-to-point wires connecting inputs to outputs.

Several solutions have been proposed to scale the performance of crossbar fabrics. To cope with high data rates, one of the most used solutions is bit-slicing [7], where the crossbar consists of multiple lanes each of which switches part of the data in parallel. Inputs are fed to a serial-to-parallel converter, sent through a multiple bit-slice crossbar and serialized at the output. However, for high data rates, using parallel slices the crossbar remains the bottleneck. Additionally, bit-slicing adds significantly increases die size of the whole crossbar, making it infeasible [7].

Traditional buffered crossbar fabrics (e.g. CICQ) employ a dedicated buffer for each input-output port pair and parallel scheduling, to avoid the complex centralised scheduling of unbuffered crossbar fabrics. The number of buffers is quadratic in the number of ports, and they are connected by long wires, as mentioned above. Recent proposals attempt to reduce the number of buffers. A CICQ switching architecture with flexible access to crosspoint buffers has been recently proposed [8]. This approach tries to achieve better internal buffer usage by sharing them. However, the implementation cost of this approach is high. Alternative solutions have

minimised the internal buffers by using partial buffering rather than per input-output port pair [9], [10].

Our *multi-hop buffered* NOC-based architecture both reduces the buffering requirements through a multi-hop/stage architecture with path diversity, and to increase the crossbar operating frequency by reducing the length of wires and distributed arbitration.

Our work differs from [11], [12] that advocate multi-hop buffered switch architectures by studying the performance of such architectures under a variety of traffic models and architectural parameters, and comparing it in detail with CICQ crossbars.

III. The Unidirectional NOC Crossbar Fabric

The Unidirectional NOC (UDN) crossbar fabric is shown in Figure 1(b). The input line cards with FIFO queues are connected to the input of the crossbar and the output ports of the switch are connected to the crossbar chip outputs. As is common, our switch operates on ATM cells instead of (possibly much larger) Internet Protocol (IP) packets. ATM cells have a fixed size of 53 bytes of which 48 bytes are payload. IP packets can be split in ATM cells before being switched, and reassembled afterwards. In the following *cells* will refer to the ATM cells arriving at the line cards, *packets* will refer to the packets in the NOC, and *routers* are part of the NOC in the crossbar fabric. Cells arrive at the input line cards and are transferred to the crossbar fabric chip when there is space in the crossbar's network-interface (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 NOC is a two-dimensional mesh of packet-switched routers, with network interfaces (NI) on two opposing sides of the mesh. The mesh has N inputs and N outputs, and is scalable in the number of stages.

Packets contain a single ATM cell (including its header), and flow in one direction through the mesh. Packets are received entirely by a router before being forwarded to the next router, known as store and forward. Packets advance at a maximum rate of one packet per cycle. The actual operating frequency is discussed in Section VI. The router architecture, shown in Figure 2, uses FIFO input buffering with 4 packets per FIFO. Credit-based link-level flow control is used to ensure packets are sent only when the receiving router has space. (Buffers must be at least two packets deep for this reason.) Every router output has a round-robin arbiter. Because packets do not flow from right to left, the router is asymmetric, and no deadlock can occur. Routers at the edge of the mesh omit either the North or South ports.

The route of a packet from the ingress NI to the egress NI is determined by the ingress NI, and is part of the packet header. Each router uses the leading *2 log degree* bits to switch the packet to the correct output, and shifts the path in the packet by the same number of bits before forwarding it. Packets follow deterministic minimal paths through the NOC, using one of two routing algorithms: XY and balanced XY. In standard XY routing, all packets travel East to the right column (X) and then to the correct row (Y). This results in a very unbalanced NOC usage because all vertical traffic occurs in the column of egress routers, as shown in Figure 3(a).

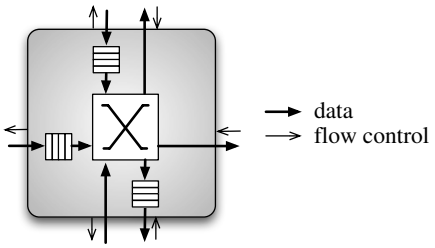
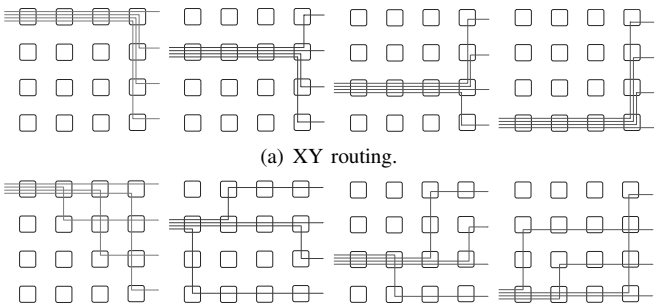


Figure 2. The UDN router.



(a) XY routing.

(b) Balanced XY routing.

Figure 3. Example of XY and Balanced XY Routing in a 4x4 UDN.

Balanced XY, illustrated in Figure 3(b), remedies this by introducing an extra turn in one of the earlier columns. A packet for output x turns South/North when $x \ mod \ M = (N-i+j) \ mod \ M$, and East when $x = j$, where i, j indicate the current router position in the mesh, N the number of inputs/outputs, and M the number of stages of the mesh. Algorithm 1 describes the actions of router (i,j) on a packet bound for output x). A flow is an uninterrupted sequence of packets from a single input to a single output; these packets should not be reordered. There is no ordering requirement on packets to different outputs, but all packets of an input-output pair follow the same path.

IV. UDN Throughput Analysis

In this section, we analytically study the throughput of an $N \times N$ UDN crossbar fabric that uses the balanced XY routing algorithm. Throughput is defined as the number of packets exiting the crossbar per output per time step, at 100% input load. We assume that the input traffic is

Algorithm 1 Balanced XY.

```
Switch (packet comes from input)
  case North:
    if i == x then go East else go South
  case South:
    if i == x then go East else go North
  case West:
    if x % M == (N-i+j+t) % M then
       if x > j then go South else go North
    else go East
```

random, the output-port arbitration is random, router buffers are infinitely large, and the number of ports of the switch is even. Note that our architecture uses store-and-forward flow control to remove inter-router dependencies, and hence make it amenable to mathematical analysis. We assume that the switch speed-up is one, i.e. the frequencies of the line cards and the crossbar are the same.

In the following we first identify five types of routers, and compute the number of each. Then we analyse the throughput of the UDN crossbar both with and without the effect of HOL blocking. We then show that analysis and simulation of the UDN crossbar correlate well, and that the effect of HOL blocking is small. This justifies replacing VOQ buffering on the input line cards by FIFOs.

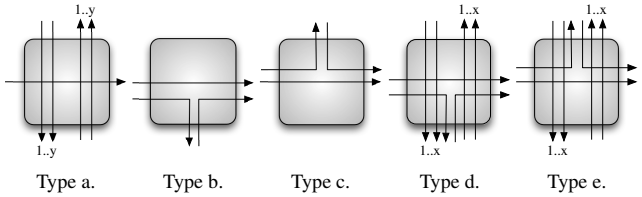


Figure 4. Router types.

A UDN crossbar fabric contains five types of routers based on the type of flows that pass through them. The flows are classified based on their directions: West→East, West→South, West→North, South→East, South→North, and North→South. The router types are illustrated in Figure 4. As an example, by overlaying all routes of Figure 3(b) on one mesh, all router types emerge. We now compute the number of routers of each type in an $N \times N$ UDN crossbar.

Type a routers do not carry packets that go West-South, South-East, West-North, or North-East. The number of flows that can go North-South (equal to those that go South-North) depends on the position of the router in the mesh: $y = 0..(N-2)/2$, where y is the number of flows for North-South and South-North. For each y there are 2 routers, and hence $N_a = N$ type-a routers.

Type b routers receive the packets that (only) travel West-East, West-South, and South-East.

Type c routers are symmetric to case b. There are $N_{b,c} = 2N-3$ of these routers.

Type d routers transport packets of type a and type b. The number of flows that go North-South or vice versa depends on the router position in the matrix: $x = 1..(N-2)/2$, where x is the number of possible flows for North-South and for South-North. For each number of flows there are $2(2N-4x-3)$ routers.

Type e routers transport packets of types a and c. They are symmetric to type d, and the number of routers of this type is: $N_{d,e} = ((1+(N-2))(N-2)+(1+(N-3))(N-3)-2(N-2))/4$.

The total number of routers adds up to $N_a+N_b+N_c+N_d+N_e = N+2(2N-3)+((1+(N-2))(N-2)+(1+(N-3))(N-3)-2(N-2))/2 = N^2$. Having identified the types of routers, we first study the throughput of the UDN-based crossbar taking into account HOL blocking on each router. We compute the probability that x packets exit the router, with $0 \leq x \leq$ the number of flows through the router. To contain the analytical complexity we model a memory-less router, i.e. packets that did not exit the router in previous time steps due to HOL blocking are ignored. We compute the throughput in the ideal case, assuming that all packets can exit every router without HOL blocking. Finally, we compare both cases to obtain the (normalised) throughput of the crossbar, and compare it with simulations.

In the following, R_x^y indicates the number of packets per cycle of a router of type x, and A_x^y the average number of packets per cycle for all the routers of type x. The superscript y is r for real throughput, taking into account HOL blocking, and i for ideal, without HOL blocking. The average throughput per type taking into account HOL blocking is:

$R_a^r = 3P(3\ \textit{diff dest}) + 2P(2\ \textit{diff dest}) + 1P(1\ \textit{diff dest})$
$= 3\left(\frac{y}{N}\right)^2 + 1\left(1-\frac{y}{N}\right)^2 + 4\left(1-\frac{y}{N}\right)\frac{y}{N} = 1+\frac{2y}{N}$
$A_a^r = 2/N\sum_{y=0}^{(N-2)/2} 1 + 2y/N = \frac{3N-2}{2N}$

$R_{b,c}^r = 2P(2\ \textit{diff dest}) + 1P(1\ \textit{diff dest})$
$= 2\left(\frac{1}{N^2}\right) + 1\left(\frac{1}{N}\left(1-\frac{1}{N}\right)\right) + 1\left(1-\frac{1}{N}\right) = \frac{1}{N^2}+1$
$A_{b,c}^r = \frac{1}{N^2}+1$

$R_{d,e}^r = 3P(3\ \textit{diff dest}) + 2P(2\ \textit{diff dest}) + 1P(1\ \textit{diff dest})$
$= 3[(1-\frac{1}{N})\frac{x^2}{N^2}] + 2[\frac{x}{N}(1-\frac{x}{N}-\frac{1}{N})(1-\frac{1}{N})$
$+(\frac{x}{N})(1-\frac{1}{N})(\frac{1}{N})] + 2[(1-\frac{x}{N})(1-\frac{1}{N})(\frac{x}{N})$
$+(\frac{1}{N})(\frac{1}{N}) + (\frac{1}{N})(\frac{x}{N})] + 1[(1-\frac{1}{N})(1-\frac{x}{N})(1-\frac{x}{N}-\frac{1}{N})]$
$+1[(1-\frac{1}{N})(1-\frac{x}{N})\frac{1}{N} + \frac{1}{N}\frac{x}{N}(1-\frac{x}{N}-\frac{1}{N})+$
$\frac{1}{N}(1-\frac{x}{N})(1-\frac{x}{N}-\frac{1}{N})] = \frac{(2N-1)x+N^2+1}{N^2}$
$A_{d,e}^r = \frac{\sum_{x=1}^{(N-2)/2}(2N-4x-3)((2N-1)x+N^2+1)/N^2}{(1+(N-2))(N-2)/2+(1+(N-3))(N-3)/2-(N-2)}$
$= \frac{16N^3-48N^2+17N-36}{6N^2(N-3)}$

The average throughput of the crossbar is then:
$A^r = (N_aA_a^r + N_bA_b^r + N_cA_c^r + N_dA_d^r + N_eA_e^r)/N^2 = (\frac{3N-2}{2} + (4N-6)(\frac{1}{N^2}+1) + 2(\frac{16N^3-48N^2+17N-36}{6N^2(N-3)})$
$(\frac{1+(N-2)}{2}(N-2) + \frac{1+(N-3)}{2}(N-3) - (N-2)))/N^2$
$= \frac{16N^3-14N^2+29N-22}{12N^3}$

Similarly, the ideal throughput per type, i.e. in the absence of HOL blocking is:

$R^i_a = 2y/N + 1$
$A^i_a = 2/N \sum_{y=0}^{(N-2)/2} \frac{2y}{N} + 1 = (3N-2)/(2N)$
$R^i_{b,c} = 1 + 2/N$
$A^i_{b,c} = 1 + 2/N$
$R^i_{d,e} = 2x/N + 2/N + 1$
$A^i_{d,e} = \frac{\sum_{x=1}^{(N-2)/2}(2(2N-4x-3)(\frac{2x}{N}+\frac{2}{N}+1))}{\frac{1+(N-2)}{2}(N-2)+\frac{1+(N-3)}{2}(N-3)-(N-2)}$
$= \frac{(8N^3-27N^2-14N+72)/(6N)}{\frac{1+(N-2)}{2}(N-2)+\frac{1+(N-3)}{2}(N-3)-(N-2)} = \frac{8N^2-11N-36}{6N(N-3)}$
The average ideal throughput in the switch is $A^i = (\frac{3N-2}{2} + (1+\frac{2}{N})(4N-6)+(\frac{8N^3-27N^2-14N+72}{6N}))/N^2 = \frac{4N^2+3N-4}{3N^2}$.

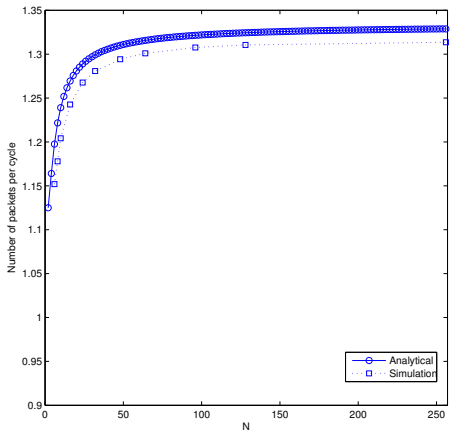


Figure 5. Number of Packets per Cycle.

Figure 5 shows the analytical average number of packets A^r of all router types for different crossbar sizes compared with that obtained by simulation. For simulation Bernoulli uniform traffic was used with an input load of 1, and a router buffer size of 500, which is effectively infinite, like in the analysis. (Section VI contains more details on the simulation set-up.) We conclude that our analysis corresponds well with the simulation results.

From the ideal throughput A^i and real throughput A^r (without and with HOL blocking, respectively), we calculate the ratio of all packets that leave the crossbar. Figure 6 shows *throughput* $= A^i/A^r$ for varying crossbar size N. As the crossbar size N grows, the real performance converges to the ideal because increasing N reduces $P(i \rightarrow j)$. In other words, the probability that a packet goes from input i to output j in a router decreases, and the mesh is more lightly loaded. Hence HOL blocking caused by South-East and West-South packets is reduced in each router. Only type-a routers have to deal with more packets (South-North and North-South increase), However, this does not increase HOL blocking due to the perpendicular directions of the flows.

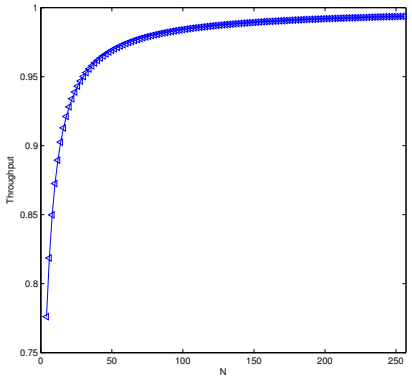


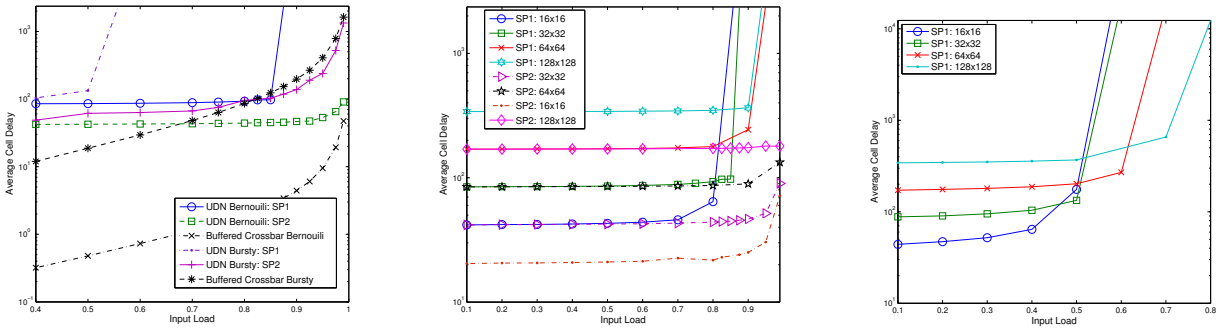
Figure 6. Throughput of UDN Crossbar.

V. Implementation

To assess the cost and performance of our proposed architectures we synthesised a NOC in an ASIC 65 nm CMOS technology. We use the Æthereal NOC [13], with input-queued worm-hole routers, link-level flow control, flit size three, 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 UDN 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 NOC offers guaranteed-throughput and best-effort services, but here we only use the latter. The arbitration of the Æthereal router is more complex than that of the UDN router and hence will be slower. The NIs are more general than required here because they implement shared-memory transaction semantics, resulting in (costly) response buffers that are unused here.

We generated a 3×3 UDN topology, RTL VHDL, and SystemC models of the NOC from a high-level specification [14]. This instance contains all different router degrees, and allows us to compute the area of any size UDN 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. The area of a crossbar with N ports and M stages is then $0.29 * 2M + 0.38(N-2)M + 0.32 * 2N$ mm^2, e.g. 403 mm^2 for $N = M = 32$. The registers that are used for (4-packet or 60-word) FIFO buffers dominate the area. By using dedicated hardware ripple-through FIFOs, described in [15], the area drops significantly. In a 90 nm CMOS process a dedicated FIFO is three times smaller than a register-based FIFO. Using the same scaling factor for all FIFOs in 65 nm, an N-port crossbar would occupy approximately $0.15 * 2M + 0.12(N-2)M + 0.14 * 2N$ mm^2, e.g. 134 mm^2 for $N = M = 32$.

The data path of the router is 36 bits, hence the cycle



(a) Bernoulli Uniform and Bursty traffic, UDN vs. CICQ, $N = M = 32$.

(b) Bernoulli Uniform Traffic for Varying N and *SP*.

(c) Bursty Uniform Traffic for Varying N.

Figure 7. Delay comparison between UDN and CICQ crossbars.

time (to move one packet or ATM cell from one router to the next) is $53 * 8/36 + 1 = 13$ cycles (assuming store and forward, and adding one for the NOC packet header). The maximum sustainable throughput of an $N \times N$ crossbar (diagonal traffic with no contention), is therefore $N/(13{*}2.4$ ns$) = 32N10^7$ cells/second, or 10^{10} cells/second for $N = 32$. The minimum cell latency is $13M$ cycles, or $13 * 32 * 2.4 = 1$ millisecond for $M = 32$. Many optimisations are possible, such as doubling the data width of the NOC which immediately doubles the throughput, at limited additional area cost because the number of registers remains the same. The NI area can be reduced significantly by removing the response buffers and programming infrastructure [16], which are superfluous in the current context.

VI. Simulation

This section presents the experimental results of the proposed NOC-based crossbar and compares it to a traditional CICQ buffered crossbar with round-robin scheduling and N^2 internal buffers of one cell each. We used the Stanford University simulator [17]. The router architecture of Section III was implemented with parameterised N, M, B, routing algorithm, and arbitration (round robin, random, etc.). We define the cost of a UDN crossbar as $SP \times$ *silicon area*, and performance as $1/$*average cell delay*. We simulated with different traffic conditions. 1) *Bernoulli uniform and bursty uniform traffic* [3] with different burst sizes b. Uniform traffic is simple and smooth, and handled well by most crossbar architectures. In Bernoulli traffic, in each time slot a cell is generated with probability p; the load is equally distributed over all outputs. Bursty traffic models Internet traffic, and contains alternating series of (in)active periods, where the average number of cells generated in an active period follows a geometric distribution with burst size b. 2) *Diagonal traffic* [18], where input i sends $2/3$ and $1/3$ of its load to outputs i and $i{+}1$, respectively. 3) In *Unbalanced traffic* [19] the parameter w determines the rate of unbalance. $w = 0$ corresponds to uniform traffic where every input sends cells to every output, and $w = 1$ to completely unbalanced traffic where every input sends cells to exactly one output. This tests the robustness of the crossbar architecture to traffic unbalances. All simulations last one million time slots, and data collection starts after 1/10th of the simulation.

We vary the following crossbar parameters: size N, stages M, speed-up SP (ratio of crossbar : line-card frequency), and buffer size B. We evaluate UDN-crossbar cost:performance, and compare it with that of a CICQ crossbar, under the traffic conditions defined above. The goal of our simulations is to determine the relative importance of the parameters M, *SP*, and B, as well as the scalability in performance with varying N, and the robustness of the UDN crossbar to traffic variations.

A. Uniform Traffic

Figure 7(a) shows the average cell delay of and a fully buffered CICQ crossbar, and a UDN crossbar with speed-up 1 and 2. When the traffic is Bernoulli and uniformly distributed over the outputs, CICQ outperforms UDN. The Bernoulli uniform traffic is a simple smooth traffic and therefore most switching architectures perform well under this traffic pattern. UDN has a higher delay than CICQ, due to the multi-hop nature of UDN as opposed to the direct point-to-point interconnect of CICQ. But considering that the hardware implementation of a UDN crossbar can operate faster than a CICQ crossbar, the difference may be smaller. For bursty traffic, UDN with speed-up 2 outperforms the CICQ architecture at higher loads (most relevant).

Figure 7(b) illustrates the average cell delay of UDN with varying crossbar sizes N and speed-up *SP*. The UDN architecture has a scalable delay performance for increasing switch sizes, in the sense that for larger N the crossbar saturates at a higher load. However, the delay at larger N is larger due to the increase in number of hops. The same hold for bursty uniform traffic with speed-up one, as shown in Figure 7(c). These results confirm the analysis of Section IV. Note that, unlike CICQ, the UDN delay is not

sensitive to varying traffic loads and/or switch sizes. Hence flows through the crossbar have a constant delay, which is important for Quality of Service, e.g. for (soft) real-time flows such as video and audio. This is due to UDN's path diversity and balanced XY routing that balance the load, unlike unique fixed path for every input-output pair in CICQ crossbars.

Figure 8 contains various cost:performance trade offs. Recall that cost is *SP* $\times$ *silicon area*, performance $1/$*average cell delay*. Figure 8(a) shows that the cost of a UDN crossbar sharply increases with increasing size N, but without a corresponding increase in performance, for speed-ups 1 and 2. In fact, at speed-up 2, the performance reduces with increasing N. The reason is that the multiple stages $M = N$ become counter-productive for larger N. i.e. all packets have already arrived at their output row, and incur additional delays travelling West-East. Figure 8(b) therefore studies the effects of the reducing the number of stages M, and the buffer size B, for $N = 32$ and $SP = 2$. For $B = 4$, performance increases sharply for $1 \leq N \leq 7$. For $N > 7$ the cost increases, but the performance reduces, because the extra stages are not required to switch packets to their correct output and just add delay. The best cost:performance point is therefore $N = 32, M = 7$ for $B = 4$, which is a cost reduction by a factor 5 compared to the baseline implementation with $N = M$. The cost:performance inversion is not seen with $B = 2$ (performance only increases with larger N), from which we can conclude that the buffer size is the limiting factor rather than the number of stages. Figure 8(c) shows that the factor $N/M \approx 5$ is maintained with $N = 64$ and $M = 12$. Figure 8(d) shows that speed-up can be traded for number of stages. A speed-up from 2 to 3 reduces M from 7 to 4.

B. Non-Uniform Traffic

This section studies the UDN performance under non-uniform unbalanced and double diagonal traffic patterns, for $N = M$, $B = 4$. Figure 9(a) shows that the UDN crossbar with *SP* $= 2$ consistently outperforms the CICQ crossbar in terms of throughput, especially under heavily unbalanced traffic loads ($\omega = 0.5$). Throughput is defined as the number of packets exiting the crossbar per output per time step, at 100% input load. Note that we modelled UDN *with FIFO-queued line-cards* and CICQ *with VOQ-queued line-cards*. Therefore UDN crossbars (at speed-up 2) can use FIFO buffers instead of more complex VOQ buffering, and still outperform CICQ with VOQ buffers.

In terms of time-slots delay, Figure 9(b) shows for unbalanced traffic that the UDN performs better than CICQ under higher loads, even with speed-up one. At light loads the UDN performs worse, due to its multi-hop delay. However, as mentioned before, the UDN can operate at higher speeds, which will narrow or eliminate this gap. Similarly, under double diagonal traffic shown in Figure 9(c), the UDN performs better at higher loads, but only for speed-up 2.

To assess the effect of the number of stages M for various N on the UDN throughput, we studied the throughput and stability performance of UDN under unbalanced traffic conditions. Figure 9(a) shows that speed-up 2 achieves 100% throughput, and in Figure 10 we determine by how much M can be lowered, to reduce cost, while retaining the same performance. In fact, 100% throughput is achieved at $M = 3$ for $N = 16$ (see Figure 10(a)), at $M = 6$ for $N = 32$ (see Figure 10(b)), and at $M = 12$ for $N = 64$ (see Figure 10(c)). This confirms the $N/M \approx 5$ cost savings also found for uniform traffic. For $N/M < 5$, the number of hops is not enough to distribute the load in the mesh. This degradation is more noticeable for smaller meshes because more routers are removed, relatively to their size. (50%, 33%, 25% etc. for $N = 2, 3, 4$, respectively.)

Finally, we study the effect of the router buffer size B on performance, as shown in Figure 11. Even for speed-up two, the buffer size B becomes important when M is reduced. With $B = 2$, the UDN crossbar does not achieve 100% throughput when $M = 15$, whereas at $B = 4$ we can reduce M to 4. In both Figures 10 and 11 performance deteriorates mainly for small ω, when the traffic is more balanced. Balanced traffic is harder to handle and requires more stages because flows from a single input fan out to more outputs, and hence have more crossings (points of contention) with other flows. In fully unbalanced (diagonal) traffic ($\omega = 1$) all inputs i send packets only West-East to corresponding output i, without any contention, and even $M = 1$ will suffice.

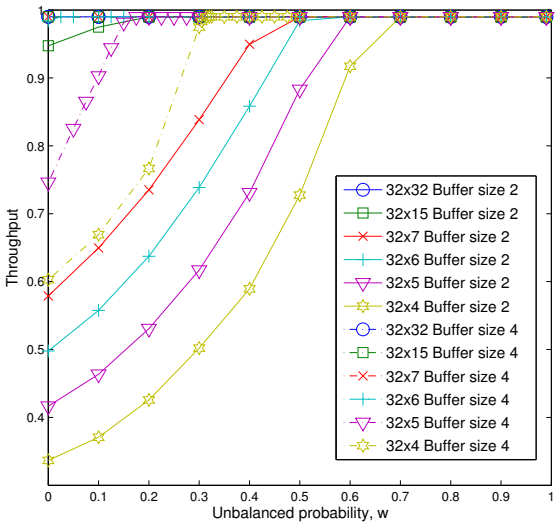
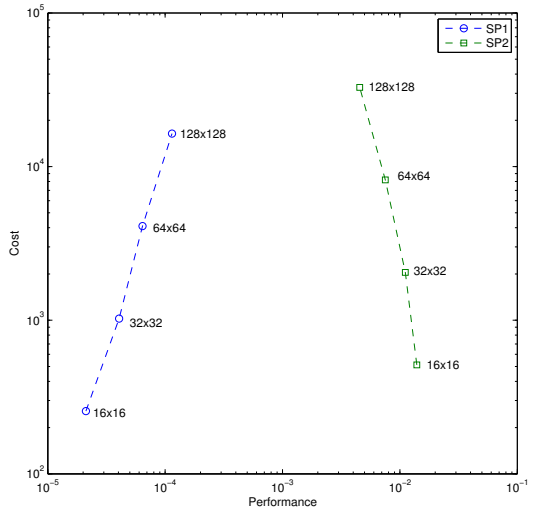
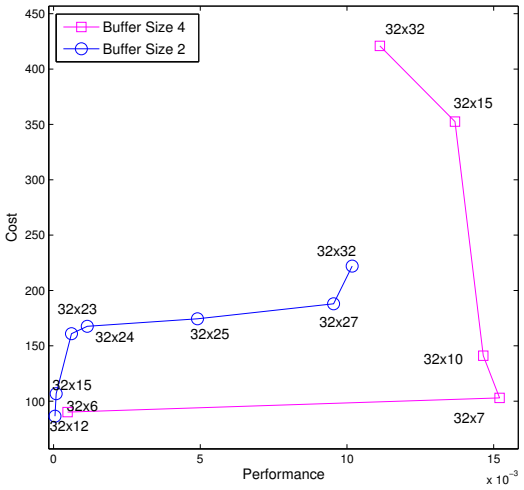


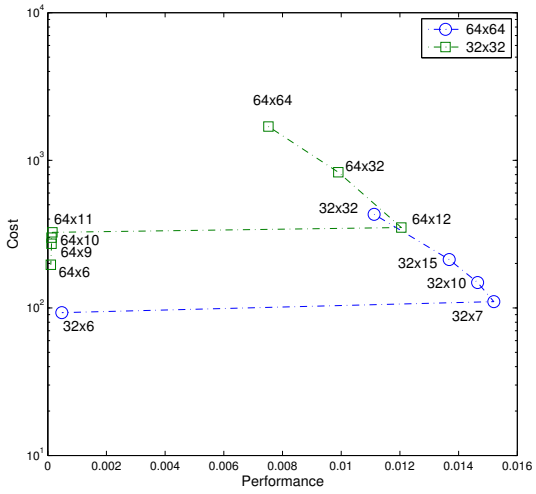
Figure 11. Unbalanced Traffic for $SP = 2$, $N = 32$ and varying B, M.



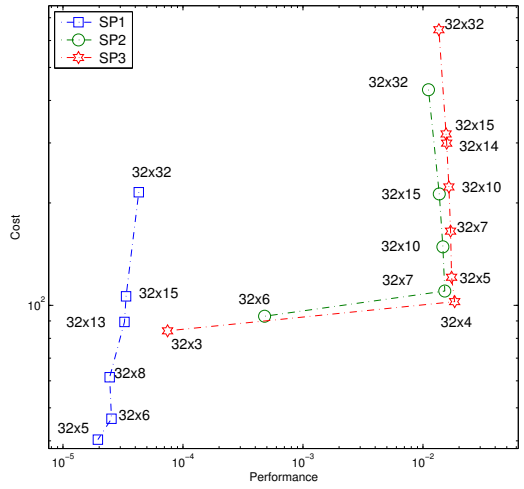
(a) Cost-Performance with $B = 4$ and varying SP and $N = M$.



(b) Cost-performance with SP= 2, $M = 32$, varying M, B.

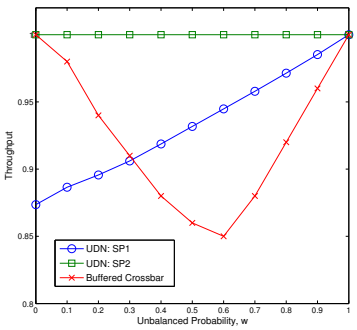


(c) Cost-Performance with $B = 4$ and varying N.

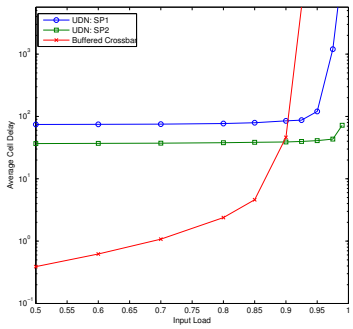


(d) Cost-performance with $B = 4$ and varying SP and M.

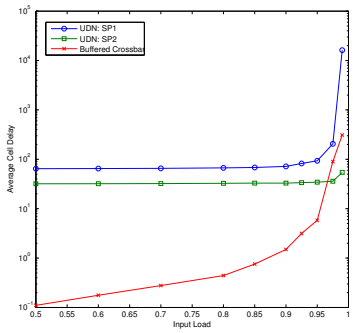
Figure 8. Cost-Performance of UDN under Bernoulli Uniform Traffic.



(a) Throughput Stability.

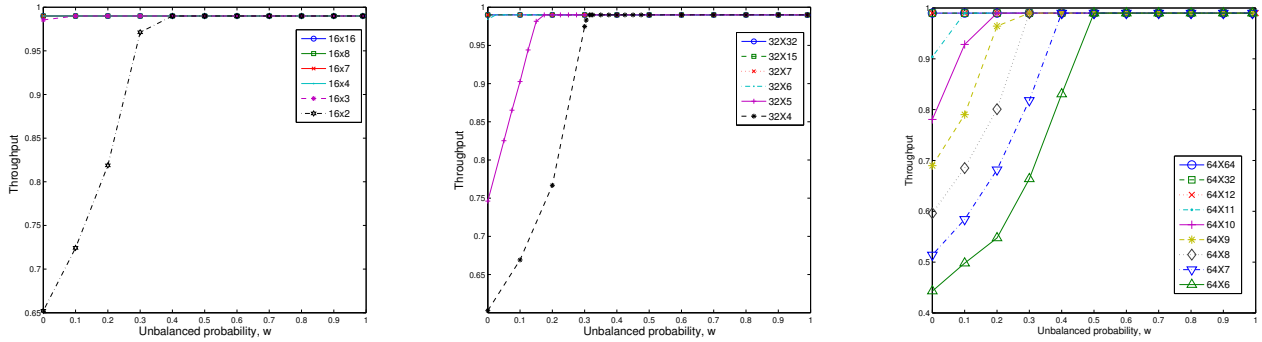


(b) Unbalanced traffic, $\omega = 0.5$.



(c) Double diagonal traffic

Figure 9. Throughput and Delay Comparison between UDN and CICQ Crossbars for $N = 32$.



(a) 16x16 (b) 32x32 (c) 64x64

Figure 10. Unbalanced Traffic with $SP = 2$, $B = 4$, and varying M, N.

C. Parameter Analysis

We can draw a number of conclusions from our simulation experiments. First, the UDN parameters in order of increasing positive impact on the performance are: speed-up *SP*, mesh stages M, and router buffer size B. Performance increases most when increasing speed-up, and speed-up two achieves 100% throughput for uniform and unbalanced traffic.

Except for bursty traffic, for $SP = 1$, reducing the number of stages M reduces performance, but for $SP = 2$ the number of stages M can be reduced to $M \approx N/5$ without loss of performance. However, when using a smaller buffer size $B = 2$ the number of stages M cannot be reduced below N without lowering performance. Hence, in terms of a cost:performance trade-off, it is best to use speed-up two, and $B = 4$ with $M = N/5$ instead of $B = 2$ with $M > N/2$ (Figure 11), even though the total crossbar buffering is roughly equal in both cases ($N * M * B = N*N/5*4 \approx N*N$ vs. $N*M*B \approx N*N*2/2 = N*N$). The UDN crossbar performs worse with bursty traffic than for Bernoulli and unbalanced traffic. Speed-up is still the most important parameter.

VII. Conclusions

We propose an Internet-router crossbar fabric, based on a multi-hop network on chip (NOC) with FIFO-queued line cards. Our design has several advantages over traditional single-hop buffered crossbar fabrics (CICQ) with VOQ line cards. 1) *Speed-up*, i.e. faster crossbar operation, which significantly improves performance. 2) *Load balancing* because router buffers are shared between all flows, and due to path diversity. 3) *Simpler line-card design* due to FIFO instead of VOQ queuing. 4) *Scalability*, in the sense that a) the crossbar speed is independent of the number of ports, and b) larger crossbars have better performance.

We implemented a prototype 32×32 crossbar in 65 nm CMOS to determine its silicon performance (413 MHz, or more than 10^{10} ATM cells per second) and cost. We also investigated the performance and cost of the NOC-based crossbar both analytically and using simulation. We determined the relative importance of several crossbar parameters with Bernoulli uniform, bursty uniform, and unbalanced traffic. In order of importance to performance these are speed-up, number of mesh stages, and router buffer size. The NOC-based crossbar performs well for all switch sizes with Bernoulli and unbalanced traffics, and a somewhat less well with bursty traffic. Speed-up two achieves 100% throughput, when the number of stages $M \geq N/5$ for the mesh size N. Alternatively, with $N = M$ the router buffer size can be reduced from four to two. But it is preferable to reduce M rather than B according to our cost:performance analysis.

The NOC-based crossbar, using FIFO queuing on the input line cards, with speed-up two has a higher (100%) throughput than a CICQ crossbar with VOQ queuing, for Bernoulli and unbalanced traffic. However, for Bernoulli traffic and light loads, it has higher average cell delays than CICQ due to its multi-hop nature. The NOC-based crossbar has a scalable performance in the sense that larger crossbars saturate at a higher load. A prototype 32×32 NOC-based crossbar in 65nm operates at 413 MHz, with an aggregate throughput of 10^{10} ATM cells per second.

References

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

[2] F. Abel, C. Minkenberg, P. Luijten, M. Gusat, and I. Iliadis, "A Four-Terabit Packet Switch Supporting Long Round-Trip Times," *IEEE Micro*, vol. 23, no. 1, pp. 10–24, January/February 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] T. Aramaki, H. Suzuki, S. Hayano, and T. Takeuchi, "Parallel 'ATOM' Switch Architecture For High Speed ATM Networks," *IEEE International Conference on Communications (ICC)*, pp. 250–254, 1992.

[6] M. Devault, J. Y. Cochennec, and M. Servel, "The Prelude ATD Experiment: Assessments and Future Prospects," *IEEE Journal on Selected Areas in Communications*, vol. 06, no. 09, pp. 1528–1537, December 1998.

[7] K. K. Chang, S. Chuang, N. McKeown, and M. Horowitz, "A 50 Gb/S 32x32 CMOS Crossbar Chip Using Asymmetric Seriallinks," *Symposium on VLSI Circuits*, pp. 19 – 22, 1999.

[8] R. Rojas-Cessa and Z. Dong, "Combined Input-Crosspoint Buffered Packet Switch with Flexible Access to Crosspoint Buffers," *IEEE International Caribbean Conference on Devices, Circuits and Systems, Playa del Carmen*, April 2006.

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

[10] L. Mhamdi, "A Partially Buffered Crossbar Packet Switching Architecture and Its Scheduling," in *Proceeding of IEEE International Symposium on Computers and Communications (ISCC)*, July 2008.

[11] W. J. Dally, P. P. Carvey, and L. R. Dennison, "The Avici terabit switch/router," in *Proc. Symposium on High Performance Interconnects*, Aug. 1998, pp. 41–50.

[12] D. Whelihan and H. Schmit, "Memory optimization in single chip network switch fabrics," in *Proc. Design Automation Conference (DAC)*, Jun. 2002.

[13] K. Goossens, J. Dielissen, and A. Rădulescu, "The Æthereal network on chip: Concepts, architectures, and implementations," *IEEE Design and Test of Computers*, vol. 22, no. 5, pp. 414–421, Sept-Oct 2005.

[14] K. Goossens, J. Dielissen, O. P. Gangwal, S. González Pestana, A. Rădulescu, and E. Rijpkema, "A design flow for application-specific networks on chip with guaranteed performance to accelerate SOC design and verification," in *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)*. Washington, DC, USA: IEEE Computer Society, Mar. 2005, pp. 1182–1187.

[15] P. Wielage, E. J. Marinissen, M. Altheimer, and C. Wouters, "Design and DFT of a high-speed area-efficient embedded asynchronous FIFO," in *Proc. Design, Automation and Test in Europe Conference and Exhibition (DATE)*, 2007.

[16] A. Rădulescu, J. Dielissen, S. González Pestana, O. P. Gangwal, E. Rijpkema, P. Wielage, and K. Goossens, "An efficient on-chip network interface offering guaranteed services, shared-memory abstraction, and flexible network programming," *IEEE Transactions on CAD of Integrated Circuits and Systems*, vol. 24, no. 1, pp. 4–17, Jan. 2005.

[17] "SIM," High-Performance Networking Group, Stanford University http://klamath.stanford.edu/tools/SIM/.

[18] P. Giaccone, D. Shah, and B. Prabhakar, "An implementable Paraller Scheduler for Input-Queued Switches," *IEEE Micro*, vol. 19, no. 01, pp. 1090–1096, January/February 1999.

[19] R. Rojas-Cessa, E. Oki, Z. Jing, and H. J. Chao, "CIXB-1: Combined Input One-Cell-Crosspoint Buffered Switch," *IEEE Workshop on High Performance Switching and Routing (HPSR)*, pp. 324–329, 2001.