Hardware Design and Implementation of a Network-on-Chip Based Load Balancing Switch Fabric

Turhan Karadeniz*, Lotfi Mhamdi†, Kees Goossens‡ and J.J. Garcia-Luna-Aceves*
*Computer Engineering Department, University of California Santa Cruz
Email: tkaradeniz, jj@soe.ucsc.edu
†School of Electronic and Electrical Engineering, University of Leeds, UK
Email: l.mhamdi@leeds.ac.uk
‡Electrical Engineering Faculty, Technical University of Eindhoven, Netherlands
Email: k.g.w.goossens@tue.nl

***Abstract*—Network routers rely on an important hardware component, namely the switch fabric, responsible for forwarding incoming packets to their respective output ports according to a scheduling algorithm. A switch fabric mainly consists of buffering memories for temporary queuing and the scheduling unit(s) for forwarding.**

In this paper, we revisit our previously proposed Network-on-Chip (NOC) based switch fabric architecture and: 1) propose an FPGA based hardware implementation of the NOC switch; 2) carry out performance tests, both via RTL simulations and actual execution on FPGA, under uniform traffic flows; and 3) present results in terms of throughput, average latency, and average bitrate. Our results show that our architecture i) performs as good as other buffering schemes/scheduling algorithms that theoretically achieve 100% throughput, ii) is at least as scalable as other architectures in terms of hardware cost, iii) is perfectly implementable and iv) introduces NOC concepts, which have originally been borrowed from computer networks, back into computer networks.

***Keywords*-Switch fabric; computer networks; router; Network on Chip; NOC; FPGA; scheduling**

I. Introduction

In today's world, billions of users all over the world are connected through networks of different sizes, purposes and scopes. It might be a local area network (LAN), the Internet backbone, infrastructure nodes of a wireless network, or hosts in an ad-hoc network that act like routers for forwarding packets, which connect a number of nodes or networks.

The communication in between the nodes/networks is realized by a broad and diverse body of electronic and optical technology. The routers carry out the task of connecting two or more nodes/networks, and perform two important functions: routing path determination, computing the route of a packet that has been injected in the router, and packet forwarding, transmitting the packet to the destination address. The switch fabricis a key building block of a router, which implements the latter function. When a packet is injected into the router, it is stored in a buffering memory; its corresponding output port is computed; and finally the packet is forwarded through the output port to its next hop, according to a scheduling algorithm. A switch fabric consists of buffering memories for temporary storage, scheduling unit(s) for forwarding, and other computational components that facilitate these tasks.

The routers need to support high bandwidths, in order not to become the bottleneck in the communication themselves. The switch fabric, being the key component in a router, constitutes an important part of the router design effort, and therefore it remains to be an open research problem. The switch fabric design consists of architecture design and scheduling algorithm design. The architecture design is about interconnect topology and the buffering memory organization. The scheduling algorithm carries out the task of deciding which packet is to be forwarded, in case a number of packets compete for the same output port, resulting in contention; one of these packets will be forwarded, whereas others will need to be stored in the buffering memory, until later rounds.

The rest of the paper is organized as follows: in Section 2, we present the background information and the related literature. This section includes the information on various architectures that historically have been the milestones in the network switch design. Also, the Network-on-Chip (NOC) related concepts are described. In Section 3, we present our switch fabric architecture and its corresponding hardware implementation; the organization of components such as the buffering memory, Network Interfaces (NI) and Mini-Routers (MR); the routing algorithm (among MRs); and finally the scheduling algorithm. In Section 4, we present the RTL synthesis, RTL simulation and the actual execution results on FPGA. Section 5 concludes the paper.

II. Related Work

A. The Switch Fabric

The switch fabric is one of the most important building blocks of a network router. Moreover, it requires the implementation of a scheduling unit, which regulates and grants

978-1-4673-2921-7/12/$31.00

permission for the pairing of input-output ports and buffers in between.

The main design challenges for implementing switch fabrics include bandwidth, latency, scheduling algorithms, interfacing, and routing algorithms as in NOC based solutions. Several switch fabric architectures have been proposed, including the crossbar [1], shared-bus [2] and shared-memory [3] switches, which deal with these design challenges in various ways. The crossbar switch is the dominant architecture in today's high-performance switches, due to a number of reasons: crossbar switch is more scalable than the shared-bus and shared-memory; this is due to the limitations in bus transfer bandwidth and memory access bandwidth, respectively. Crossbar switch provides point-to-point connections and non-blocking properties, as well as supporting multiple simultaneous transactions, increasing the bandwidth and speed of the router [4].

The FIFO scheduling, perhaps the simplest scheduling scheme for input buffering suffers head-of-line (HOL) blocking, where a packet at the head of the queue cannot be delivered, and therefore blocks the others behind it, resulting in throughput decrease (58.6%), increased delays, and congestion. A number of algorithms/architectures were proposed in order to remediate this shortcoming, including PIM, RRM, iSLIP [5][6][7], based on virtual output queues (VOQ), claiming a theoretical 100% throughput.

Another proposal, the load-balancing switch [8], claims greater scalability. VOQ architectures do not scale optimally as the number of ports is increased, and therefore become impractical. The load-balancing switch architecture does not have a scheduler, at the cost of duplicating the packets within the switch fabric.

B. Network-on-Chip

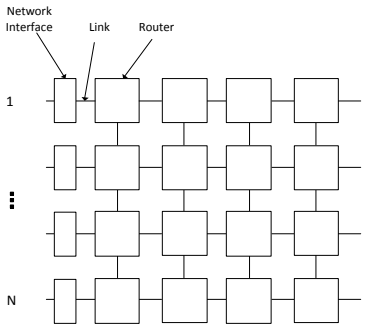

Figure 1. NOC Switch Fabric

Network-on-chip, a relatively new concept that emerged as a system-on-chip (SOC) communication methodology, borrows many ideas from the computer networks, the domain in which the research on routers and packet switching has matured. However, they need to be adapted, since there is no direct translation of these methodologies.

Figure 1 represents a NOC router, in the form of a regular N-by-N mini-router (MR) mesh. Computational cores in a SOC are connected to each other via this communication fabric, composed of network interfaces (NI), and MRs.

The NIs act as an abstraction layer between the computational cores and MRs. The data to be communicated in between these cores are packetized in NIs and transmitted to the next-hop router, in equally sized flits.

An MR might have multiple packets in the buffers competing for the same output port, resulting in contention. A scheduling algorithm computes which packet has to be forwarded prior to the other packets. The arbiter, the hardware embodiment of the scheduling algorithm, makes a link in between the chosen buffer and output port, such that the packet is forwarded. In this paper, the scheduling and arbitration terms are used interchangeably. Round Robin offers fair scheduling, assigning each resource equal usage in circular order, which results in a starvation-free system.

The communication through the NOC is pipelined automatically due to the nature of the point-to-point communication in between MRs. In this way, the critical path is restricted to the control signals in the switch NOC fabric, improving the scalability and throughput.

Some other important concepts in NOC are topology, routing, flow control, buffer management, quality of service and network interfaces and they have been studied in acclaimed proposals such as Aetherial [9], Nostrum [10], Xpipes [11], Intel 80 Core NOC [12], and Mango [13]. They all make different design decisions, to achieve their design goals.

Mesh-based NOC architectures and NOC routing algorithm have been discussed in various other publications, including [14], [15], [16], [17].

C. Network-on-Chip Based Switch Fabric

Recently, functional-level designs of two novel Network-on-Chip (NOC) based switch fabric architectures were proposed: Unidirectional NOC (UDN) and Multidirectional NOC (MDN) [18][19], as a replacement of the buffered crossbar switch fabric architecture (Figure 2), targeting greater scalability and flexibility, as well as greater performance per hardware cost, compared to buffered crossbar switch fabric.

The crossbar-based switch fabric architectures offer very high performance and are widely used in high-performance routers. However, their cost grows quadratically with the input/output port count, since they require internal crosspoints (and buffers) for every input/output port pair. The UDN proposal decouples the number of ports from the cost growth, and therefore is able to achieve subquadratic growth. The MDN, in return, is quadratic, however it achieves greater performance/cost ratio, for smaller number of ports. Both of the proposals introduce load-balancing without duplicating the packets, which in turn improves the throughput and latency.

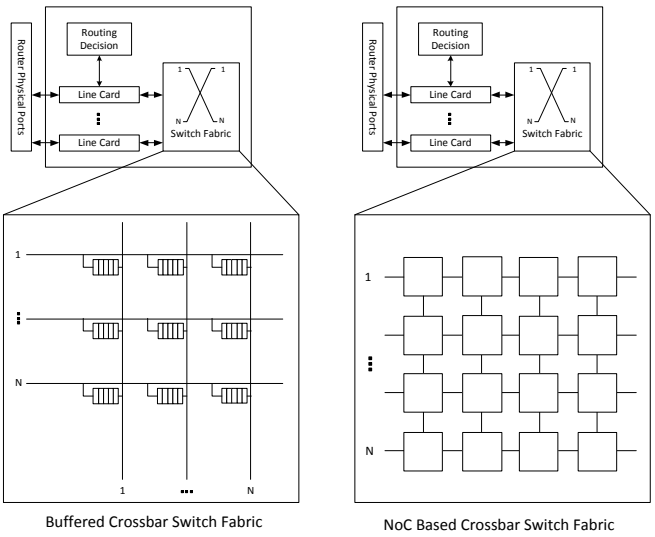

Figure 2. Buffered Crossbar and NOC Based Crossbar Switch Architectures

NOC, a paradigm of on-chip communications, with its basic concepts borrowed from computer networks, is proposed to be applied back to its original domain, to remedy some shortcomings in the switch fabric design. In regard to this matter, the basic building blocks of NOC, including the buffers, flow control, arbitration and routing decision units need to be used in the correct combination of schemes/specifications, to be able to provide a competitive solution.

III. Hardware Design

A. UDN and MDN Architectures & Algorithms

The block diagrams for UDN and MDN switch architectures are presented in Figure 3 and Figure 4. The main difference among the two switch fabrics is how their input/output pins are placed in the layout. In UDN, the input pins are placed on the west side of the layout, whereas the output pins are on the east side. On the other hand, in the MDN switch, the pins are placed all around the peripheral, where input and output pins are next to each other. UDN MRs have either 2 or 3 I/O ports, whereas MDN MRs have 4 I/O ports.

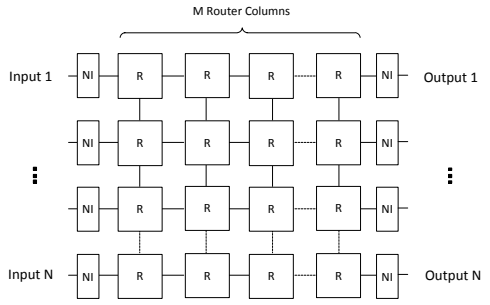

Figure 3. UDN Architecture

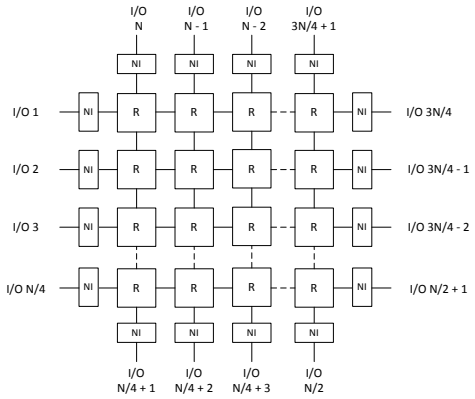

Figure 4. MDN Architecture

The UDN and MDN switch architectures have the same NOC specifications. The switching mode is store and forward. We choose to apply buffered flow control, implemented by FIFO input buffers. Buffered flow control requires a (buffer management/backpressure) mechanism, which we choose to be the valid/accept scheme, instead of the credit-based scheme in the original proposal [18], in order to decrease inter-MR communication overhead. The buffer size is four packets. The scheduling is based on iSLIP [7]. XY Modulo Algorithm allows deterministic and adaptive routing within the MR mesh, where routing path decision is made in an incremental fashion, at each MR. We choose to implement our switch fabric for fixed-length packets, for simplicity.

The size of the UDN switch, as shown in Figure 3, is defined by the 2-tuple (N, M), where N denotes the number of input-output ports, and M denotes the number of MR columns. Some restrictions apply to (N, M) values: $N \in \mathbb{N}$, and $N \geq 2$; $M \leq N$ and $M = 2^m$, where $m \in \mathbb{N}_0$. The restrictions on M are caused by the routing algorithm involving Modulo M operations (See Section III-F and [18]). Because Modulo M operation requires division in case $M \neq 2^m$, but it is a simple bit-selection operation in case $M = 2^m$, we can avoid the extra cycles caused by the division operation by applying this restriction. With some minor modification in the routing algorithm, $M = N - 1 \Rightarrow N = 2^n$, where $n \in \mathbb{N}_1$ is also a possible (N, M) combination. The number of routers in the UDN switch is equal to $N \times M$.

The size of the MDN switch, as shown in Figure 4, is a function of N, which denotes the number of input/output ports. Because the input/output ports are placed around the peripheral of the switch, some restrictions apply to N: $N = 4 \times n$, where $n \in \mathbb{N}_1$. The input/output ports are placed counter-clockwise, starting from the West side of the layout. The ports in between 1 and (N/4) are placed on the West; (N/4 + 1) and (N/2) on the South; (N/2 + 1) and (3N/4) on the East; and (3N/4 + 1) and N on the North side of the layout. The number of routers is equal to $(N/4)^2$.

In the UDN and MDN switch design, the same network interface, flow control unit, and buffering memory modules are used. The UDN and MDN MRs are different, due to the different number of ports.

The UDN is a deadlock-free architecture due to its uni-directional nature, whereas the possibility of deadlocks in MDN are avoided with the use of virtual channels [20].

B. Input Buffers (FIFOs)

The input buffer is implemented as a circular queue. The read/write register positions are marked by the Read Pointer (RP) and Write Pointer (WP). During a write operation, the data is written into this register, and the WP is incremented. In the same way, after a successful read operation, the RP is incremented to point to the following data. Status register (SR) is incremented after each write operation and decremented after each read operation. If SR = 0, then the buffer is empty (Empty Status Signal is set); therefore there cannot be made any read operations. If SR = Buffer Size, then the buffer is full (Full Status Signal is set); therefore no more write operations are allowed until SR is decremented. The Empty Status Signal informs the packet forwarding unit (PFU) in the current module (NI or Routers) about the availability of a valid packet for a read operation. The Full Status Signal generates Accept Signal, which informs the previous module's (NI or MR) PFU about the availability of buffer space.

C. Network Interface (NI)

The Network Interface (NI) is the module that acts as an abstraction layer in between the network protocol and internal UDN/MDN switch protocols. There are two types of NI: Input NI (INI) (Figure 5) and Output NI (ONI). When a packet is injected into the switch, INI encapsulates (packetizes) them into U(M)DN packets, and transmits them to the next router, in equally sized flits. ONI, in return, receives the U(M)DN flits, strips (depacketizes) the original packet, and ejects it from the switch.

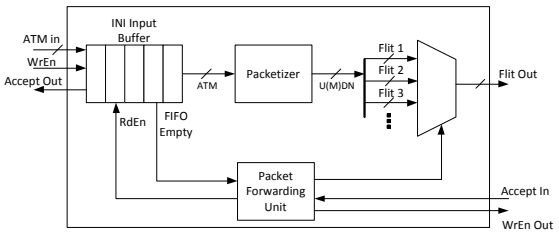

Figure 5. Input Network Interface Block Diagram

D. UDN 3 I/O MR

The block diagram for a 3 I/O Port Router is given in Figure 6. There are 3 input ports, West, North and South; and there are 3 output ports, East, North and South. The router input buffer modules are placed at each input port, whereas the PFUs are placed at each output port. At each round, the arbiter virtually connects input ports to output ports, according to the scheduling algorithm.

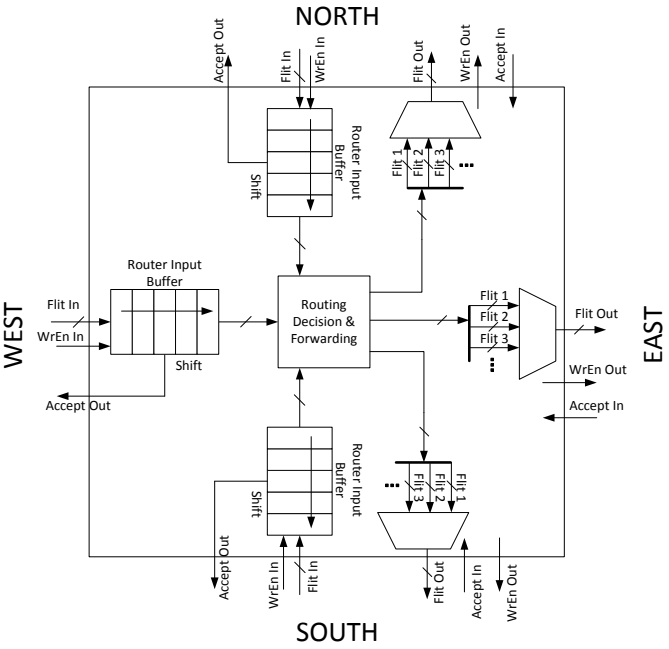

Figure 6. 3 I/O Port UDN Router, Top-Level Block Diagram

E. MDN 4 I/O MR

The MDN architecture is based on UDN; therefore, most of the UDN modules are also being used in the MDN switch, including NIs, PFUs and Input Buffer. The arbiter is also very similar in terms of basic principles, however it is more complex due to handling the connectivity between four input buffers and four PFUs. Moreover, MDN requires virtual channels to avoid deadlocks: WEST, NORTH and SOUTH input ports of the WEST-most MR column; EAST, NORTH and SOUTH input ports of the EAST-most MR column; and NORTH and SOUTH ports of the other MDN MRs have a pair of buffers, demuxed at their input and muxed at their output (Figure 7). In the functional-design proposal [18], the virtual channels are not muxed; however this results in the arbiter handling seven pairings, rather than four, and thus not feasible in terms of an actual hardware implementation.

F. Modulo XY Routing Algorithm

Both UDN and MDN routing algorithms are based on modulo operation. The algorithms make a balanced distribution of the traffic over the columns or rows, thus earning its name XY Modulo; this means that the modulo operation is applied if the communication is on the X axis (from MR's West Input Port to MR's East Output Port as in UDN and MDN, or from MR's East Input Port to MR's West Output Port or vice versa as in MDN) or on the Y axis (from MR's

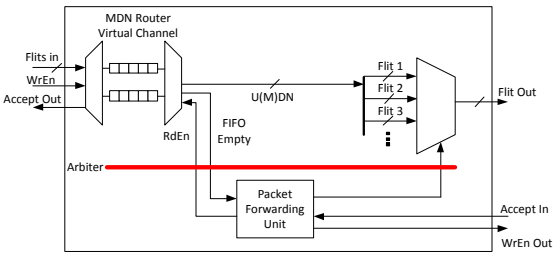

Figure 7. Virtual Channel for an Input Ports of MDN Router

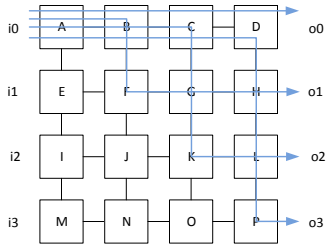

Figure 8. XY Modulo Routing

North Input Port to MR's South Output Port, or vice versa, as in MDN). In Figure 8, we exemplify the XY Modulo routing on the X axis, where packets are injected into the switch from i0 input port, with destinations to the o0 - o3 output ports. The packets are routed on different router columns per input/output port pairs, distributing the load on the switch.

G. Scheduling Algorithm

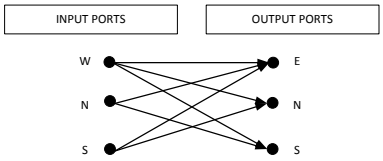

Figure 9. Bipartite Graph Matching Problem in UDN

The scheduling unit resolves the contention among the input buffers of a router, competing for the same output port, as well as controlling the arbiter to virtually connect the chosen input buffer to the output port.

The scheduling of the UDN switch is a bipartite graph matching problem. This can be formulated as G = (I, O, E) where I denotes input ports (W, N, S), O denotes output ports (E, N, S) and E denotes the edges. The graph is a directed graph. The unmatched graph is presented in Figure 9. Similarly, MDN scheduling is a H = (I', O', F); I' = (W, E, N, S), O' = (W, E, N, S) and F denotes the edges. The scheduling is based on the iSLIP algorithm [7] running a single round.

Table I
SYNTHESIS RESULTS FOR INDIVIDUAL UDN/MDN MODULES

Module Name	Size (# of Slice LUTs)		Max Frequency
	as LUTs	as Regs	(MHz)
NI Buffer	1278	853	734.7
Input NI	359	857	597.229
Output NI	450	857	510.843
XY MODULO	2	-	-
Router Buffer	531	1016	508.414
Virtual Channel	532	1016	508.414
Router (2 I/O Port)	1679	2045	269.485
Router (3 I/O Port)	3073	3073	257.107
Router (4 I/O Port)	6197	4112	255.115

IV. SIMULATIONS, EXECUTION & RESULTS

A. RTL Synthesis

The RTL synthesis is carried out on Xilinx ISE 11.1, using Xilinx Synthesis Technology (XST) tool, with the settings 'Optimization Goal: Speed', 'Optimization Effort: Normal' and 'Keep Hierarchy: No'. The specific FPGA device is Virtex 5 - XC5VTX240T, FF1759, -2.

The XST tool reports the area results in terms of 'Number of Slice LUTs' and 'Number of Slice Registers', unlike the results for older platforms like Virtex 4, which reported a single value for slice usage; therefore, all the performance & cost analysis will be reported in 2-tuple. We do not present the results in metrics such as logic cell count, gate count or ASIC area size, since the design is made for the reconfigurable platforms, and the conversions from FPGA metrics are not meaningful, even in terms of providing estimates.

The modules that constitute the UDN/MDN switches have different tasks and therefore different weights on the combinational and sequential circuits. This can be observed by comparing the weights of the number of LUTs and Registers for any module. The synthesis results of the individual modules are given in Table I.

Synthesis results show that the frequency of the UDN switch fabric is independent of M. In Table II we present frequency results for various (N, 1) UDN and (N) MDN switches. (N, N-1) UDN has higher operational frequencies than the corresponding (N) MDN switch, at the expense of greater cost per port number. UDN arbiter is responsible for three I/O ports, whereas MDN's is responsible for four ports, resulting in longer critical path. The (8, 7), (16, 15), (32, 31) UDN switches require more than 100% of the resources on a Virtex-5 (XC5VTX240T, FF1759, -2), and therefore cannot be placed on the FPGA. On the other hand, only (32) MDN switch cannot be placed on the FPGA.

The UDN vs. MDN switches comparison is tabulated in Table III. The cost of the UDN switch is a function of the product of N and M, with M varying from any small number to N-1. The UDN cost growth is only quadratic in the worst case of M = N-1 (yielding the greatest throughput); however keeping M small, it's possible to keep the growth sub-

Table II
COMPARISON OF SYNTHESIS RESULTS, I

Switch Sizes		# of Slice LUTs		# of Slice REGs	
UDN	**MDN**	**UDN**	**MDN**	**UDN**	**MDN**
(4, 3)	**4**	31759	8933	37590	10969
(8, 7)	**8**	179978	30387	171508	30153
(16, 15)	**16**	762421	108996	734114	93203
(32, 31)	**32**	3137269	410697	3039532	318016

Switch Sizes		Frequencies	
UDN	**MDN**	**UDN**	**MDN**
(4, 3)	**4**	290.252	250.591
(8, 7)	**8**	283.168	240.667
(16, 15)	**16**	252.139	212.227
(32, 31)	**32**	220.146	173.214

quadratic, unlike the crossbar switch fabric, while achieving acceptable throughputs. The MDN yields quadratic growth of $(N/4)^2$. Even though the 4 I/O Port MDN MR consumes twice the amount of resources as the 3 I/O Port UDN MR, and 3.5 times as the 2 I/O Port UDN MR, the comparison of the overall UDN and MDN switch fabrics show that MDN is more cost efficient, for small N values. On the other hand, the operational frequencies of the MDN switches of various sizes are below the operational frequencies of the UDN switches, which would affect the performance.

Table III
COMPARISON OF SYNTHESIS RESULTS, II

	Arbiter Size	**MR Size (LUTs/REGs)**	**MRs**	**Cost Increase**
UDN	3	3073/3073	N×M	Subquadratic
MDN	4	6197/4112	$(N / 4)^2$	Quadratic

B. Simulation

The simulations were carried out on RTL descriptions of hardware components, with functional models of traffic generators/sinks. For the UDN, M=N-1 yields the greatest throughput for all N values; therefore, in this section M is always chosen to be N-1. The throughput for (N, N-1) UDN switches, under uniform traffic, is 99.54% and 97.34%, where N=4 and N=32 respectively. The throughput for (N) MDN switches, under uniform traffic, is 98.23% and 96.12%, where N=4 and N=32 respectively. The average latency for (N, N-1) UDN switches, under uniform traffic, is 10.8 and 94.7 cycles/'Number of flits a packet is divided to', where N=4 and N=32 respectively. The average latency for (N) MDN UDN switches, under uniform traffic, is 14.2 and 126.7 cycles/'Number of flits a packet is divided to', where N=4 and N=32 respectively.

For the bitrate computations, we use both the frequency results of the RTL synthesis, as well as the RTL simulation results. The UDN offers 6.56 and 42.12 Gbytes/sec aggregate bandwidth for (4, 3) and (32, 31) UDN switches; which imply 1.64 and 1.32 Gbytes/sec bandwidth per port, respectively. On the other hand, the MDN offers 5.22 and 21.36 Gbytes/sec aggregate bandwidth for (4) and (32) switch sizes; which imply 1.31 and 0.67 Gbytes/sec bandwidth per port, respectively. High performance Ethernet cables offer 1.25 Gbytes/sec bandwidth; therefore, the UDN switch proves to be a competitive architecture to comply with the market products, whereas MDN's performance does not match this bandwidth, as the number of input/output ports is increased.

C. Validation on FPGA

The UDN switch (N, M) = (4, 3) has been validated on Virtex 5-XC5VTX240TFF1759-2. Each input port of the switch fabric is connected to a linear feedback shift register (LFSR) based pseudo-random packet generators.

The packet generators and switch fabric are initially idle. Once the PowerPC CPU is initiated and our validation software starts to run, a trigger register is set to 1, which starts the packet generation. The headers of the incoming packets are written on a dual port RAM, which can be accessed by the CPU. When the packets are transmitted through the destination output port, their headers are also written to the dual port RAM. In this way, the validation software can compare the inbound and outbound packets, and verify if the system works correctly. With this approach, we were able to validate that all the packets have been switched correctly and that the simulation results hold correct.

Using this system, we also carried out some performance analysis. Once the system has been run a certain period of time, the trigger register was reset, stopping the packet generation. Measuring the time period (in terms of cycles) in between the set and reset of the trigger register, it has been observed that the performance on the FPGA matches the simulation results. The error percentage is ˜9%, for latency and throughput measurements, which are mainly due to: 1) pseudo-random generation based on LFSR is not uniform, therefore number of generated 1 and 0s are not equal; and 2) measurement of time period is not accurate.

V. CONCLUSION

In this paper, we proposed the hardware design and implementation of the two NOC based switch fabric architectures (UDN and MDN) for FPGA. We further improved the routing and scheduling algorithms, for the feasibility of their hardware design. The synthesis and RTL simulations are carried out over a range of switch sizes. The simulation results are also validated on FPGA, with packets generated by LFSR based pseudo-random traffic generators. The results show that UDN outperforms MDN in terms of throughput, whereas MDN offers greater performance-cost ratio. UDN's high performance makes it suitable for performance critical cases, whereas MDN is a better solution for cases that require cost efficiency. Both architectures offer scalability,

flexibility and high performance, confirming the ideas in the original proposal [18].

The results show that our architecture 1) performs as good as other buffering schemes/scheduling algorithms that theoretically achieve 100% throughput, 2) is at least as scalable as other architectures in terms of hardware cost, 3) is perfectly implementable, 4) implements load-balancing by nature and 5) introduces NOC concepts, which have originally been borrowed from computer networks, back into computer networks.

The UDN and MDN NOC switches would benefit from fault-tolerance, which can be implemented by exploiting FPGAs' property of dynamic reconfigurability. In case of a malfunctioning MR, other UDN and MDN MRs could be reconfigured to route the packets through an alternative path. This is left as future work, to improve the systems' performance, reliability and service capability further.

REFERENCES

[1] N. McKeown, M. Izzard, A. Mekkittikul, W. Ellersick, and M. Horowitz, *The Tiny Tera: A Packet Switch Core*. 1996.

[2] D. Torres, J. Gonzalez, M. Guzman, and L. Nunez, "A new bus assignment in a designed shared bus switch fabric," in *Circuits and Systems, 1999. ISCAS '99. Proceedings of the 1999 IEEE International Symposium on*, vol. 1, pp. 423–426 vol.1, 1999.

[3] H. Kuwahara, N. Endo, M. Ogino, T. Kozaki, Y. Sakurai, and S. Gohara, "A shared buffer memory switch for an ATM exchange," in *Communications, 1989. ICC '89, BOSTONICC/89. Conference record. 'World Prosperity Through Communications', IEEE International Conference on*, pp. 118–122 vol.1, 1989.

[4] L. Mhamdi, "PBC: A partially buffered crossbar packet switch," *IEEE Transactions on Computers*, vol. 50, pp. 1568–1581, November 2009.

[5] T. E. Anderson, S. S. Owicki, J. B. Saxe, and C. P. Thacker, "High-speed switch scheduling for local-area networks," *ACM Transactions on Computer Systems*, vol. 11, pp. 319–352, Nov. 1993.

[6] N. McKeown and T. E. Anderson, "A quantitative comparison of iterative scheduling algorithms for input-queued switches," *COMPUTER NETWORKS AND ISDN SYSTEMS*, vol. 30, pp. 2309—2326, 1998.

[7] N. McKeown, "The iSLIP scheduling algorithm for input-queued switches," *IEEE/ACM Transactions on Networking*, vol. 7, pp. 188–201, Apr. 1999.

[8] I. Keslassy, C.-S. Chang, N. McKeown, and D.-S. Lee, "Optimal load-balancing," in *Proceedings IEEE INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, pp. 1712– 1722 vol. 3, IEEE, Mar. 2005.

[9] K. Goossens, J. Dielissen, and A. Radulescu, "AEthereal network on chip: concepts, architectures, and implementations," *Design & Test of Computers, IEEE*, vol. 22, no. 5, pp. 414–421, 2005.

[10] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch, "Guaranteed bandwidth using looped containers in temporally disjoint networks within the nostrum network on chip," in *Design, Automation and Test in Europe Conference and Exhibition, 2004. Proceedings*, vol. 2, pp. 890–895 Vol.2, 2004.

[11] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, and L. Benini, "Xpipes: a latency insensitive parameterized network-on-chip architecture for multiprocessor SoCs," in *Computer Design, 2003. Proceedings. 21st International Conference on*, pp. 536–539, 2003.

[12] T. Mattson, R. Van der Wijngaart, and M. Frumkin, "Programming the intel 80-core network-on-a-chip terascale processor," in *High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for*, pp. 1–11, 2008.

[13] T. Bjerregaard and J. Sparso, "A router architecture for connection-oriented service guarantees in the MANGO clockless network-on-chip," in *Design, Automation and Test in Europe, 2005. Proceedings*, pp. 1226–1231 Vol. 2, 2005.

[14] D. Wiklund, A. Ehliar, and D. Liu, "Design of an internet core router using the SoCBUS network on chip," in *Signals, Circuits and Systems, 2005. ISSCS 2005. International Symposium on*, vol. 2, pp. 513 – 516 Vol. 2, July 2005.

[15] A. Ehliar and D. Liu, "An FPGA based open source network-on-chip architecture," in *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*, pp. 800 –803, Aug. 2007.

[16] G. Luo-Feng, D. Gao-ming, Z. Duo-Li, G. Ming-Lun, H. Ning, and S. Yu-Kun, "Design and performance evaluation of a 2D-mesh network on chip prototype using FPGA," in *Circuits and Systems, 2008. APCCAS 2008. IEEE Asia Pacific Conference on*, pp. 1264 –1267, Dec. 2008.

[17] S. Asghari, H. Pedram, and M. Khademi, "A flexible design of network on chip router based on handshaking communication mechanism," in *Computer Conference, 2009. CSICC 2009. 14th International CSI*, pp. 225 –230, Oct. 2009.

[18] K. Goossens, L. Mhamdi, and I. Senin, "Internet-router buffered crossbars based on networks on chip," in *Digital System Design, Architectures, Methods and Tools, 2009. DSD '09. 12th Euromicro Conference on*, pp. 365–374, 2009.

[19] L. Mhamdi, K. Goossens, and I. V. Senin, "Buffered crossbar fabrics based on networks on chip," in *Communication Networks and Services Research Conference (CNSR)*, pp. 74–79, 11-14 May 2010. 10.1109/CNSR.2010.18.

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