TABLE I
QUANTITY PER CLUSTER AND NUMBER OF INPUTS FOR EACH TYPE OF ROUTING SIGNAL IN THE VPR ARCHITECTURE [2]

Signal	Quantity/Cluster	Number of inputs
LUT input	$kN = 24$	$F_{cint}I + F_{cfb}N = 20$
Cluster input	$I = 14$	$F_cW = 24$
Routing track	$2W/L_{wire} = 24$	$2F_s + L_{wire}NF_{out}/2 = 8$

number of MUX inputs required (see Table I). A MUX with n inputs contributes at most $\log(n)$ to the entropy, so we sum the log of the number of inputs over all the signals. We obtain an entropy of 240 bits per cluster or 40.0 bits per basic logic cell. This looks reasonable compared to the lower bound. Now suppose we alter the previous parameters to $F_{\text{cint}} = 0.25$, $F_{cfb} = 0.25$, and $F_c = 0.1$. Then the entropy per cell becomes 26.6, which we can confidently say is insufficient (based on our lower bound of 27) even for PLDs with only 65K cells. This is true for any detailed routing architecture consistent with these parameters and for any routing algorithm.

REFERENCES

[1] V. Betz, J. Rose, and A. Marquardt, *Architecture and CAD for Deep-Submicron FPGAs*. Norwell, MA: Kluwer, 1999.

[2] G. Lemieux and D. Lewis, *Design of Interconnection Networks for Programmable Logic*. Norwell, MA: Kluwer, 2004.

[3] W. E. Donath, "Placement and average interconnection lengths of computer logic," *IEEE Trans. Circuits Syst.*, vol. CAS-26, no. 4, pp. 272–277, Apr. 1979.

[4] U. Malik and O. Diessel, "The entropy of FPGA configuration," in *Proc. FPL*, 2006, pp. 261–266.

[5] A. DeHon, "Rent's rule based switching requirements," in *Proc. SLIP*, 2001, pp. 197–204.

[6] R. Rubin and A. Dehon, "Design of FPGA interconnect for multilevel metallization," in *Proc. Int. Symp. FGPAs*, 2003, pp. 154–163.

[7] A. DeHon, "Entropy, counting, and programmable interconnect," in *Proc. Int. Symp. FGPAs*, 1996, pp. 73–79.

[8] Y. Matsumoto and A. Masaki, "FPGAs with multidimensional switch topology," *IEICE Trans. Inf. Syst.*, vol. E88-D, pp. 775–778, Apr. 2005.

[9] H. Schmit, "Extra dimensional island style FPGAs," in *Proc. Field Program. Logic Appl. (FPL)*, 2003, pp. 406–415.

[10] C. E. Shannon, "A mathematical theory of communication," *Bell Syst. Tech. J.*, vol. 27, pp. 379–423, Jul./Oct. 1948.

[11] B. S. Landman and R. L. Russo, "On a pin versus block relationship for partitions of logic graphs," *IEEE Trans. Comput.*, vol. C-20, no. 12, pp. 1469–1479, Dec. 1971.

[12] W. E. Donath, "Wire length distribution for placements of computer logic," *IBM J. Res. Develop.*, vol. 25, pp. 152–155, May 1981.

[13] D. Stroobandt, *A Priori Wire Length Estimates for Digital Design*. Norwell, MA: Kluwer, 2001.

[14] D. Stroobandt, "Multi-terminal nets do change conventional wire length distribution models," in *Proc. SLIP*, 2001, pp. 41–48.

[15] W. Feng and J. Greene, "Post-placement interconnect entropy: How many configuration bits does a PLD need?," in *Proc. Syst.-Level Interconnect Prediction (SLIP)*, 2006, pp. 41–48.

[16] T. Cover and J. Thomas, *Elements of Information Theory*. New York: Wiley, 1991.

Routability of Network Topologies in FPGAs

Manuel Saldaña, Lesley Shannon, Jia Shuo Yue, Sikang Bian, John Craig, and Paul Chow

***Abstract*—A fundamental difference between application-specific integrated circuits (ASICs) and field-programmable gate arrays (FPGAs) is that the wires in ASICs are designed to match the requirements of a particular design. Conversely, in an FPGA, the area is fixed and the routing resources exist whether or not they are used. In this paper, we investigate how well several common network topologies map onto a modern FPGA routing fabric. Different multiprocessor network topologies with between 8 and 64 nodes are mapped to a single large FPGA. Except for the fully-connected networks, it is observed that the difference in logic resources used and routing overhead among these topologies is insignificant for the systems tested. Fully-connected networks up to about 22 nodes are also feasible on the same FPGA although the logic and routing utilization clearly grows much faster. The conclusion is that a modern FPGA fabric is very rich in resources and capable of supporting highly interconnected topologies. For systems with a modest number of nodes implemented on current large FPGAs, it is not necessary to use the connectivity-limited topologies typically used for networks-on-chip. Rather, direct point-to-point connections between all communicating nodes can be considered.**

***Index Terms*—Computer networks, field programmable gate arrays (FPGAs), multiprocessor interconnection, reconfigurable architectures, topology.**

I. INTRODUCTION

With the growing complexity of system-on-chip (SoC) circuits, more sophisticated communication schemes are required to connect the increasing number and variety of intellectual property (IP) blocks. Approaches like AMBA [1], CoreConnect [2], WISHBONE [3], and SiliconBackplane [4] follow a shared bus scheme that works well for master-slave communication patterns, where there are only a few masters. When there are several masters (e.g., processors) in the system, synchronization, data interchange, and input/output (I/O) may saturate the bus and contention will slow down data transfers.

A network-on-chip (NoC) [5], [6] provides a possible solution for this problem by creating a scalable interconnection scheme. The concept uses a set of buses connected to routers or switches that interchange packets, much in the same way as traditional computer networks or multiprocessor machines do. Consequently, NoC approaches have design parameters and properties similar to traditional networks. One of these parameters is the topology, which defines the interconnection pattern between the routers and switches.

Multiple topologies have been studied for NoCs on application-specific integrated circuits (ASICs) [7], [8]. Topologies such as the hypercube, ring, star, torus, and trees have simple packet routing algorithms. However, a popular choice is the mesh [6], [9] because it provides structure, better control over electrical characteristics, and has an easy packet routing algorithm. These advantages are clear for ASICs, but not necessarily for field-programmable gate arrays (FPGAs) [10]. The electrical characteristics of the FPGA are solved by the chip vendor, not by the user. The placement of components by the computer-aided

Manuscript received July 1, 2006; revised February 3, 2007. This work was supported in part by Xilinx, by CONACYT, Mexico, by the O'Brien Foundation, and by the Natural Sciences and Engineering Research Council of Canada.

The authors are with the Department of Electrical and Computer Engineering, University of Toronto, Toronto, ON M5S 3G4, Canada (e-mail: msaldana@eecg.toronto.edu).

Digital Object Identifier 10.1109/TVLSI.2007.900746

design (CAD) tools for an FPGA will not result in symmetric, well-organized structured layouts that make the wiring easier. Furthermore, manually restricting the placement of components or routing of nets on an FPGA may lead to inefficient resource utilization.

The FPGAs programmable fabric can be used to implement any topology without worrying about the physical implementation. Instead, the concern becomes whether or not a particular topology is too complex to be routable given the fixed set of logic and routing resources available on the chip.

In this paper, we investigate the routability of several network topologies on FPGAs, measuring the impact of soft multiprocessor system topologies and their characteristics when implemented on a modern commercial FPGA. We do this by measuring the logic utilization, logic distribution (area), maximum clock frequency, number of nets, and the place and route time for six well-known network topologies and a number of randomly generated topologies.

The rest of this paper is organized as follows. Section II provides some background about research on NoCs. Section III describes the methodology and testbench infrastructure implemented. Section IV presents the resource utilization results obtained from the experiments. Finally, Section V provides some conclusions.

II. Related Work

Much research has been done on NoCs [6], [7], [9], [10], but the focus is mostly on the dynamics of the network (e.g., efficient network interface devices, packet routing algorithms, deadlock avoidance, flow control), ASIC technology, simulation models, or they focus on a single network topology. The synthesis of application-specific network topologies provides multiprocessor SoC designers with a tool to map a particular application to a particular topology and generate an *ad hoc* NoC, which can be simulated [11]. In this paper, we focus on the next level in the design flow after mapping an application to a particular topology: we map several topologies to a particular chip and investigate the limits of current commercial FPGA technology.

III. Experimental Environment

The experimental methodology and testbench infrastructure we use focuses on static message passing networks. The ring, star, mesh, hypercube, 2-D-torus, fully-connected, and some random topologies are selected as a representative sample of static networks, ranging from the simplest ring topology to the routing-intensive fully-connected system. The actual processor and network interface used are not the critical elements in this study. What is required is to create circuits that force particular communication patterns between the computing nodes to see how the resources required to implement these circuits on the FPGAs varies as the patterns (i.e., topologies) are changed.

We are interested in three network topology characteristics: node degree, link complexity, and regularity [12]. The node degree is the number of links from a node to its nearest neighbors. The link complexity is the number of links the topology requires. A network is deemed to be regular when all the nodes have the same node degree. Fig. 1 shows examples of systems with different numbers of nodes and topologies that are implemented to carry out our experiments. The topologies are generated with a varying number of nodes ranging from 8 to 64. Every topology can be seen as a graph that is made of edges (links) and vertices (computing nodes). In our implementations, the links are 70 bits wide with 32 bits used for transmission and 32 bits used for reception. The links also include six control lines used by the network interface. The computing nodes in Fig. 1 consist of a computing element and a network interface module. Fig. 2 shows the structure of a computing node.

We use a Xilinx MicroBlaze soft processor [13] as the computing element so the data memory and program memory are accessed by

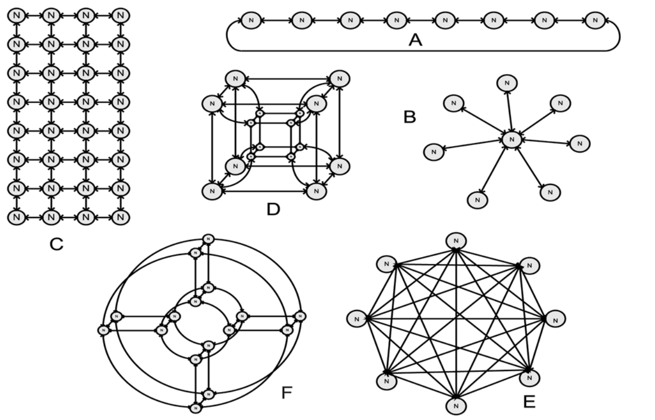

Fig. 1. (A) 8-node ring. (B) 8-node star. (C) 32-node mesh. (D) 16-node hypercube. (E) 8-node fully connected topology. (F) 16-node 2-D-torus.

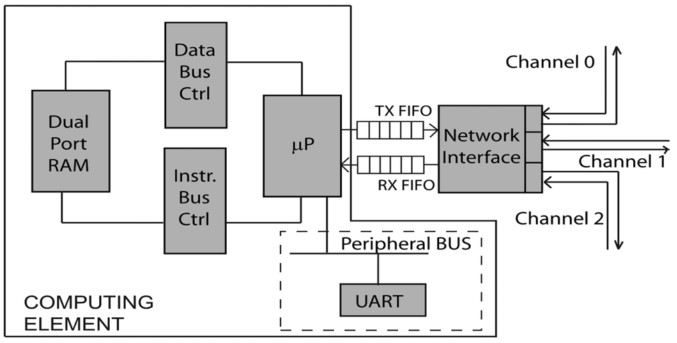

Fig. 2. Computing node architecture. The UART is only on one node in the system.

independent memory buses. The hard multiplier option for the MicroBlaze is disabled to minimize the impact of hard core blocks that may influence or limit the placement and routing. The internal Block RAM memory (BRAMs) in Xilinx FPGAs are hard core blocks that also affect the placement and routing, but they are essential for the MicroBlaze core to synthesize so they cannot be removed. The communication between the computing element and the network interface is achieved using two 32-bit wide fast simplex links (FSLs), which is a Xilinx first-input first-output (FIFO) core that provides a unidirectional communication channel. One FSL is for transmission and one for reception.

The network interface module is an extremely simple block, but it is sufficient for our purposes as the focus of this paper is on the routability of various topologies, not on the switching element architecture or the network interface itself. It interfaces with the network via several links (channels) according to the degree of the node. The two FIFOs that connect the network interface to the computing element are used as message buffers for the processor. We use network interfaces with varying numbers of channels based on the node degree of a particular topology to make sure that only the necessary logic is included.

Manually describing large multiprocessor systems is time consuming and error prone. Therefore, an automated process for generating the systems used in our experiments is developed and the flow is shown in Fig. 3. The flow starts by using the topology generator, which is a Perl script that takes a high-level description of the system, such as the type of topology and the number of nodes. The topologies are described as graphs in a graph description file that is input to the system generator [14]. The system generator translates the graph into IP cores and buses and generates the files used by Xilinx's

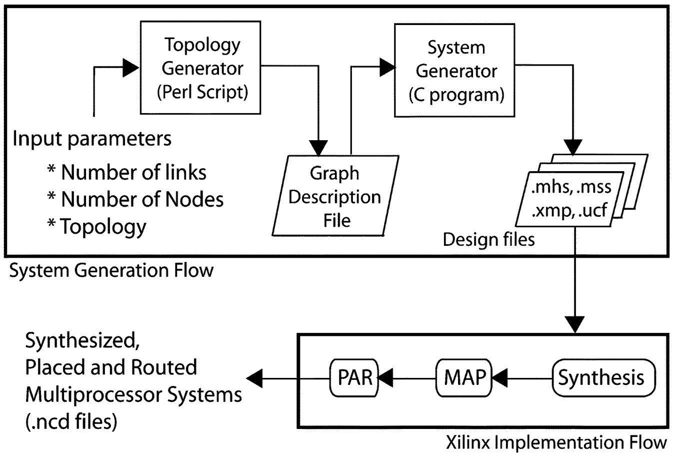
Topology Generator (Perl Script)
System Generator (C program)
Input parameters
* Number of links
* Number of Nodes
* Topology
Graph Description File
.mhs, .mss .xmp, .ucf
Design files
System Generation Flow
Synthesized, Placed and Routed Multiprocessor Systems (.ncd files)
PAR
MAP
Synthesis
Xilinx Implementation Flow

Fig. 3. Design flow for generating our multiprocessor systems.

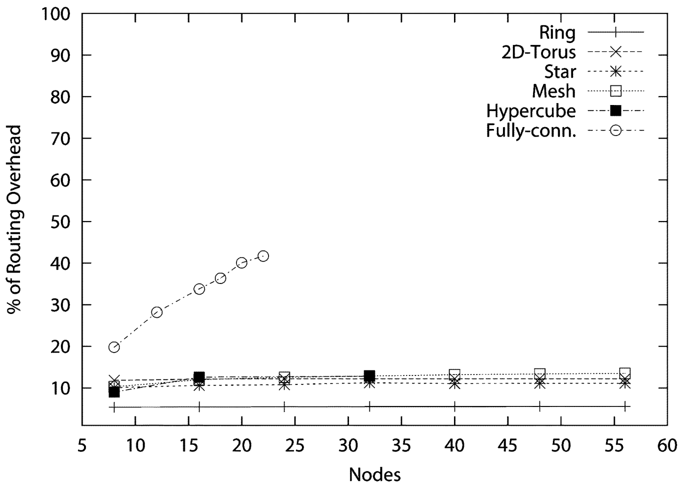

Fig. 5. Topology impact on global routing.

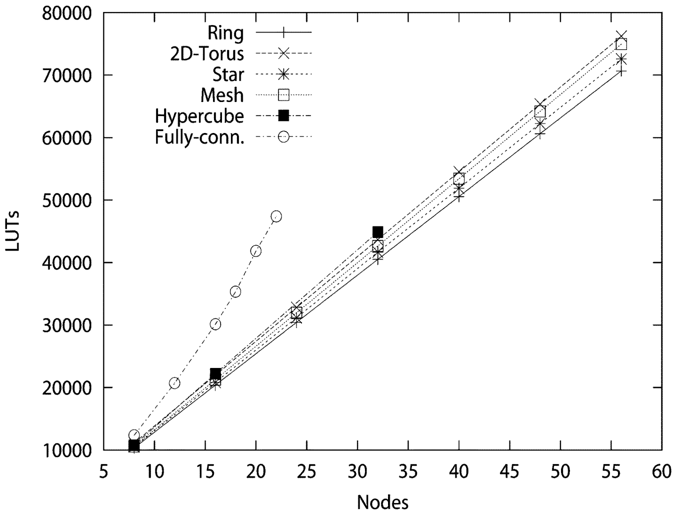

Fig. 4. Logic utilization of systems. The fully connected system did not route for more than 22 nodes. The hypercube did not place and route with 64 nodes; therefore, 32 nodes is the largest hypercube implemented.

EDK/ISE tools suite version 7.1i [13] to synthesize, place, and route the multiprocessor designs. Once the systems are generated, the routed nets are counted with the help of Xilinx's XDL tool, which provides a direct access to the native circuit description files (NCD) and provides a report of the physical resource utilization for any design.

The Xilinx XC4VLX160 FPGA was used because it is the largest LX version Virtex4 chip sold at the highest speed grade. The LX version of the Virtex4 family has the highest logic density and fewer DSP and BRAM cores than the SX version. It also does not have the PowerPC processors or multi-gigabit transceivers (MGTs). Thus, the LX version of the Virtex4 family provides the most homogeneous architecture, limiting the additional factor of hard cores embedded in the FPGA.

IV. Resource Usage Results

The logic resource usage is measured in terms of the total number of lookup tables (LUTs) required for the entire design. The number of LUTs is obtained from the EDK log file. Fig. 4 shows a plot of the number of LUTs needed to implement the complete systems, including the MicroBlaze, FSLs, memory interface controllers, network interfaces, UART, and OPB bus. As expected, the system with the fully-connected network has the highest logic utilization, and as the system size increases, the difference with respect to the other topologies becomes more pronounced because of the $O(n^2)$ growth in size.

The routing resource utilization is measured in terms of the number of nets, extracted from the XDL report. Unfortunately, the distance of each routed net cannot be determined directly by the existing tools. Scripts can be developed to compute the Manhattan distance based on the XDL file report and investigate the locality of the connections, but that remains as future work. The routing resource utilization for these systems exhibits a similar trend to the logic resource utilization shown in Fig. 4. The fully-connected system requires the most nets, as expected, while the difference among the remaining topologies is not significant. The 22-node fully-connected topology can be mapped and placed using only 38% of the total logic resources in the XC4VLX160 FPGA. Beyond 22 nodes, the fully-connected topology fails to route because there are insufficient routing resources for this particular topology. This means that any other topology with 22 nodes or less, will successfully place and route since the fully-connected topology is the worst case.

An interesting observation from the experimental data is that the logic and routing resource utilization exhibits a linear trend for all the topologies used in this experiment, except for the fully-connected, which has a square trend. The difference in the slope reflects the complexity of the different network interfaces required by each topology. This means that for a small number of nodes, all the topologies, but the fully-connected, have similar implementation requirements in the FPGA. This is consistent with the fact that for a small n, $O(n \log n) \approx O(Kn)$, where K is a constant. A larger FPGA would be required to implement more than 64 nodes where the topology complexities would start to be more noticeable.

Fig. 5 shows the routing overhead of the network interface modules and topology as a percentage of the entire design. The routing overhead is calculated as the sum of the number of nets that are related to the topology links plus all the nets from the network interface modules divided by the total number of nets in the entire design. As expected, the ring topology has the lowest routing overhead with about 6% of the total number of nets. Moreover, for those topologies that have a linear increase in routing resources, the routing overhead increases insignificantly with the system size because for these system sizes, the nodes are all fixed-degree, i.e., they each have k links, where $k = 1$ to 5. The fully-connected topology overhead increases rapidly as the number of

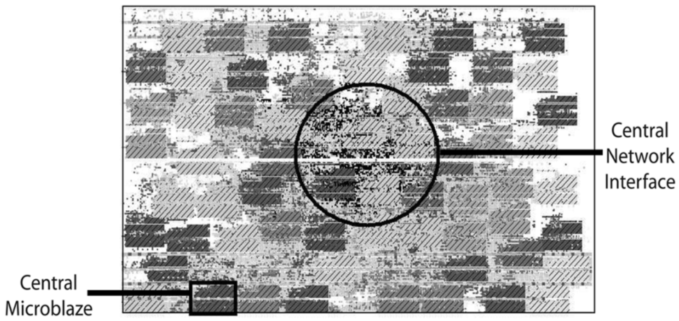

Fig. 6. 62-node star topology placement with empty spaces caused by the RPMs. Note the location of the network interface and processor for the central node.

nodes increases. With 22 nodes, the fully-connected interconnection network accounts for 42% of the total routing in the system, compared to only 13% routing overhead for the 32-node hypercube, which is the next most complex topology.

While the fully-connected topology stresses the routing resources in the entire chip, the star topology stresses the routing resources around a single node (central node). We are able to build star systems up to 63 nodes; the 64-node star system did not route. Fig. 6 shows the placement of the 62-node star topology. It is not surprising that the central network interface is in the center of the chip, as the tools try to minimize the distance to the rest of the nodes. Note that the central node's processor is placed away from the center of the chip indicating that the placement of the network interface is more important to timing than the placement of the central processor.

In our multiprocessor systems, the slowest block is the MicroBlaze, which is synthesized to a maximum frequency of 150 MHz. Therefore, the timing constraint for the entire multiprocessor system is also set to 150 MHz. By using the star topology, we investigate the effects of congestion around the central node on the operating frequency of the star topology. We found that the congestion only caused degradation in the design's performance by reducing the maximum frequency of 150 MHz for 28 nodes or less, down to 109 MHz with 63 nodes. In contrast, the ring topology, which does not experience that congestion, meets the 150 MHz requirement up to 64 nodes.

The largest systems that successfully placed and routed for each topology are: 22-node fully-connected, 32-node hypercube, 56-node mesh, 56-node 2-D-torus, 63-node star; and 64-node ring. For the 64-node ring, which is the largest system, only 44% of the LUTs and 65% of the available slices in the chip are used. One factor that limits the placement of larger systems is shown in Fig. 6. A visual inspection reveals that the design has a porous placement suggesting that the logic could be more tightly packed. This is caused by the relationally placed macro (RPM) constraints on the MicroBlaze core, which forces the core's logic into a fixed relative placement on the FPGA to achieve higher operating frequencies. Removing the constraint would allow better packing of the design's logic, but, depending on the placement, could degrade the core's operating frequency.

The place and route time data varies considerably because of the random nature of the place and route algorithm. However, the ring, star, mesh, and hypercube topologies have average times to place and route that are approximately the same for a given number of nodes. For example, on our IBM workstations with 3.0-GHz Pentium Xeon processors and 3 GB of memory, the 8-, 16-, and 32-node systems have average run times of 15 min, 35 min, and 1 h 58 min, respectively. The torus topology requires considerably more time to place and route with 37 min, 72 min, and 3 h 15 min, for 8, 16, and 32 nodes, respectively. Finally, the fully-connected topology required 16 min for the 8-node system, 4 h for the 16-node system, and 1 day and 5 h for the 22-node system.

V. Conclusion

When implementing an NoC using an ASIC, only the required routing is included. Thus, it is important to make the correct tradeoffs between the communication requirements for the application and the routing requirements of the interconnect topology. FPGAs have a rich routing fabric and the wires exist independent of their usage by the design. This work has shown that the ring, star, mesh, 2-D-torus, and hypercube topologies can all be routed on a modern FPGA and are able to scale well for systems with a large number of nodes. Given a multiprocessor system similar to those described in this paper, it would make no significant difference in logic and routing resources which topology to use. Hence, there is no need to limit the connectivity to economize the use of resources at the expense of performance. Even a fully-connected topology can work for small numbers of nodes (22 nodes), which suggests that for small networks, it is less important to worry about routing considerations when picking the network topology for an NoC on an FPGA.

This paper has considered only one particular FPGA family. Further study of how these trends change with different FPGA fabrics, for example with different FPGA architectures or families of FPGAs, or with embedded hard blocks in the FPGA fabric, is an area for future work.

Acknowledgment

The authors would like to thank Xilinx and CMC Microsystems/SOCRN for providing the tools, hardware, and computing equipment. They would also like to thank the reviewers for their comments.

References

[1] ARM Corp., Cambridge, U.K., "AMBA specification," (1999). [Online]. Available: http://www.arm.com
[2] IBM Corporation, NY, "The Coreconnect Bus Architecture," (1999). [Online]. Available: http://www.chips.ibm.com
[3] OpenCores.org, "The WISHBONE system architecture," (2002). [Online]. Available: http://opencores.org/projects.cgi/web/wishbone
[4] Sonics Inc., Mountain View, CA, "Sonics Inc. homepage," [Online]. Available: www.sonicsinc.com/sonics/products/siliconbackplaneIII
[5] G. de Micheli and L. Benini, "Networks on chip: A new paradigm for systems on chip design," in *Proc. Conf. Des., Autom. Test Eur. (DATE)*, 2002, pp. 418–418.
[6] S. Kumar, A. Jantsch, J. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani, "A network on chip architecture and design methodology," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, 2002, pp. 105–112.
[7] A. Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez, and C. A. Zeferino, "Spin: A scalable, packet switched, on-chip micro-network," in *Proc. Conf. Des., Autom. Test Eur. (DATE)*, 2003, pp. 20070–20070.
[8] P. P. Pande, C. Grecu, M. Jones, A. Ivanov, and R. Saleh, "Performance evaluation and design trade-offs for network-on-Chip interconnect architectures," *IEEE Trans. Comput.*, vol. 54, no. 8, pp. 1025–1040, Aug. 2005.
[9] W. J. Dally and B. Towles, "Route packets, not wires: On-chip interconnection networks," in *Proc. 38th Conf. Des. Autom. (DAC)*, 2001, pp. 684–689.
[10] G. Brebner and D. Levi, "Networking on chip with platform FPGAs," in *Proc. IEEE Int. Conf. Field-Program. Technol. (FPT)*, 2003, pp. 13–20.
[11] D. Bertozzi and A. Jalabert, "NoC synthesis flow for customized domain specific multiprocessor systems-on-chip," *IEEE Trans. Parallel Distrib. Syst.*, vol. 16, no. 2, pp. 113–129, Feb. 2005.
[12] J. Duato and L. Y. Ni, *Interconnection Networks, an Engineering Approach*. Los Alamitos, CA: Computer Society Press, 1998.
[13] Xilinx, Inc., San Jose, CA, "Xilinx Inc. homepage," (2006). [Online]. Available: http://www.xilinx.com
[14] L. Shannon and P. Chow, "Maximizing system performance: Using reconfigurability to monitor system communications," in *Proc. Int. Conf. Field-Program. Technol. (FPT)*, 2004, pp. 231–238.