Exploring FPGA Network on Chip Implementations Across Various Application and Network Loads

Graham Schelle and Dirk Grunwald
Deptartment of Computer Science
University of Colorado at Boulder
Boulder, CO

Abstract—

The network on chip will become a future general purpose interconnect for FPGAs much like today's standard OPB or PLB bus architectures. However, performance characteristics and reconfigurable logic resource utilization of different network on chip architectures vary greatly relative to bus architectures. Current mainstream FPGA parts only support very small network on chip topologies, due to the high resource utilization of virtual channel based implementations. This observation is reflected in related research where only modest 2x2 or 2x3 networks are demonstrated on FPGAs.

Naively it would be assumed that these complex network on chip architectures would perform better than simplified implementations. We show this assumption to be incorrect under light network loading conditions across 3 separate application domains. Using statistical based network loading, a synthetic benchmarking application, a cryptographic accelerator, and a 802.11 transmitter are each demonstrated across network on chip architectures. From these experiments, it can be seen that network on chips with complex routing and switching functionality are still useful under high network loading conditions. Additionally, it is also shown for our network on chip implementations, a simple solution that uses 4-5x less logic resources can provide better network performance under certain conditions.

I. Introduction

For FPGA designers, there is always a tendency to look at standard processor communication paradigms and adopt them in standard on chip FPGA communication protocols. This is apparent from Xilinx FPGAs adopting IBM's PLB bus architecture to be compatible with their embedded PowerPCs. But FPGA designers typically keep the arbitration and resource utilization low for these bus architectures, more so than processor designers. This simplified implementation is due to equivalent transistor densities on FPGAs lagging ASICs by an order of magnitude [1]. The bus architectures utilize so little logic resources, that they can follow ASIC implementations fairly close without taking up any sizable amount of onchip resources.

But now moving towards network on chip (NoC) architectures, the actual "wire" now consists of distributed switches, buffering, and arbitration. FPGA designers therefore cannot treat the network on chip "wire" as they similarly treat the bus shared wire described above when it comes to reconfigurable logic utilization. FPGA transistor densities cannot support large network on chip dimensions like it could with large counts of processing elements connected to a bus.

As we will discuss further in section II, there are many implementations of virtual channel NoCs for FPGAs. However, these implementations are also very small, only fitting 2x2 or 2x3 dimension NoCs on current FPGAs. While FPGA transistor densities will steadily increase, network on chips should stay as small as possible while meeting application specific performance constraints.

In this paper, we show that complex network on chip implementations are not always necessary to get the best network performance. In our case, a *complex* network on chip is a virtual channel implementation with a 5x5 crossbar at each switching node. A *simple* network on chip utilizes no virtual channels, has single word buffers per channel and a simplified switch. Also, "best" performance is measured on 3 domain specific applications with application specific metrics of performance. Specifically we look at three applications:

- A synthetic benchmarking application. In this application, all nodes on the networks send packets to uniformly distributed destinations across the NoC.
- A cryptographic accelerator. Using a processor and an accelerator communicating over the NoC, the processor requests text to be encrypted for use in a DES brute force attack.
- An 802.11 transmitter. This dataflow application streams and processes data packets through the network to a DAC (digital to analog converter).

From these three applications, we load the network on chip with increasing traffic to show that in each domain, the performance graphs vary greatly depending on the NoC implementation and the network loading by applications running in parallel.

This paper is organized as follows: Section II discusses how today's bus architectures are utilized and how network on chips scale in size. Two representative network on chip implementations are presented in section III that will be used in this research's experiments. Measured runtime results are described in IV, followed by conclusions.

II. Onchip Interconnects: Size and Utilization

In the introduction, we made claims about the reconfigurable logic utilization of network on chips placed on FPGAs. In this section, we quantify those claims looking at various network on chip projects targeting FPGAs.

978-1-4244-1961-6/08/$25.00 ©2008 IEEE.

Authorized licensed use limited to: University of Florida. Downloaded on March 3, 2009 at 17:02 from IEEE Xplore. Restrictions apply.

TABLE I

NUMBER OF FPGA FLIP FLOPS (FFS) AND LUTS USED BY A NOC FOR A VIRTEX-5 ARCHITECTURE. SPECIFICALLY, THE XILINX FPGA XC5VLX50T IS BEING EXAMINED.

Virtual Channel Implementation (NoCem)

Topology	Datawidth	FFs/LUTs	FFs/LUTs used (%)
2x2	16b	2,322 / 3,632	8% / 12%
3x3	16b	6,621 / 9,237	22% / 32%
4x4	16b	12,625 / 19,691	43% / 68%
2x2	32b	3,864 / 5,534	13% / 19%
3x3	32b	10,778 / 15,593	37% / 54%
4x4	32b	20,551 / 30,102	71% / 104%

Simple NoC Implementation

Topology	Datawidth	FFs/LUTs	FFs/LUTs used (%)
2x2	16b	476 / 714	2% / 2%
3x3	16b	1,246 / 1,961	4% / 7%
4x4	16b	2,369 / 3,742	8% / 13%
2x2	32b	732 / 1,034	3% / 4%
3x3	32b	1,918 / 2,777	7% / 10%
4x4	32b	3,649 / 5,278	13% / 18%

Additionally, we examine some representative applications that run on Xilinx's OPB or PLB buses. This examination is done to show just how often onchip interconnects are used in current applications. This initial survey will be necessary to draw broader conclusions about how heavily network on chips will be utilized by FPGA designs.

A. Network on Chip Reconfigurable Logic Utilization

Using a very popular FPGA in the research community (a Virtex-II Pro xc2vp30), the LiPaR [2] research project was able to emulate a 3x3 NoC architecture with 8b datapaths using 27% of the FPGA resources. This NoC is *not* a virtual channel implementation, but does make for a light-weight router architecture which was the intention of the project. The open source NoC emulation project, NoCem [3] is the NoC architecture we based our work from as it is freely available. Table I shows the utilization of this emulator on top of a Xilinx Virtex-5 FPGA. Additionally within the table is the sizing of a extremely lightweight network on chip that we developed for this research. This implementation will be further discussed in section III. This comparison is presented here to show the extreme size difference between network on chip implementations. For now all that is needed to know for the comparison is that the "Simple NoC Implementation" does not use virtual channels or high throughput switches within the network, trading higher bandwidth for a size reduction.

The Virtex-5 architecture is the latest FPGA produced by Xilinx, yet still can only hold modest sized virtual channel network on chips. Specifically within that emulation environment, a 3x3 NoC architecture with 16b datapaths used 22% of the FPGA resources (32b datapath used 54%), making the NoC itself a large consumer of logic resources. In several other NoC frameworks [4], [5], a 2x2 or 2x3 sized NoC were examined. These projects do show that a single FPGA can only hold a very small NoC architecture across several NoC implementations.

TABLE II

BUS UTILIZATION OF SEVERAL OPENLY AVAILABLE XILINX EDK PROJECTS.

Virtual Channel Implementation (NoCem)

Design	Board	Bus	Bus Utilization (%)
Audio Filter	XUP (Virtex2Pro)	OPB	1.93%
uClinux (bootup)	XUP (Virtex2Pro)	OPB	3.42%
uClinux (standby)	XUP (Virtex2Pro)	OPB	0.20%
Web Server	XUP (Virtex2Pro)	PLB	15.30%
DMA Transfer	Xilinx ML505	OPB	8.32%
FFT Accelerator	Xilinx ML505	OPB	2.57%

B. Current Onchip Interconnect Utilization

As a precursor to building these various network on chip implementations, we first examined how today's applications use FPGA on chip interconnects. Specifically, we looked at how OPB and PLB bus architectures were utilized across a variety of applications. Table II lists out the applications we examined and the utilization of the onchip interconnect.

To do these experiments, we took off the shelf example projects provided by Xilinx or other openly available research project source files. Inserting chipscope cores onto the bus, the bus could be monitored for transaction requests and replies going across the interconnect. The utilization of the bus was measured over 1 million clock cycles of execution. The monitoring started at an application specific point in time (e.g. for the DMA Transfer application, the monitoring started once the DMA controller was initialized). These numbers may not capture bursty behavior of the applications, but give good insight into how little onchip interconnects are used.

The majority of the applications presented utilize the bus very sporadically. As these are processor designs, there is typically use of a cache which limits the number of necessary instruction and data transactions. Not all the applications use a bursting mechanism either on the bus, creating less traffic on the bus, as each transaction requires an arbitration step. Interestingly, whenever a UART is involved that accepts interactive input, there is a constant stream of polling operations on the UART (most noticeable in the uClinux applications).

And while these applications measured only reflect processor designs, memory intensive applications (e.g. the DMA transfer and web server) resemble data flow applications from the view of the interconnect. The memory reads and writes are being bursted on the interconnect, not being hindered by the standard processor to logic interfaces. The PLB has much better burst support and the web server design uses the bus at a high utilization rate. The web server design utilization counter was triggered on a Ethernet frame reception, leading to a high utilization time period.

Interestingly from this precursory work, the bus is used very little in most cases (under 10% for all but one experiment), with most of the heavy traffic coming from polling operations when a processor is being used. While most FPGA general purpose interconnects (i.e. buses) are used for processor

Authorized licensed use limited to: University of Florida. Downloaded on March 3, 2009 at 17:02 from IEEE Xplore. Restrictions apply.

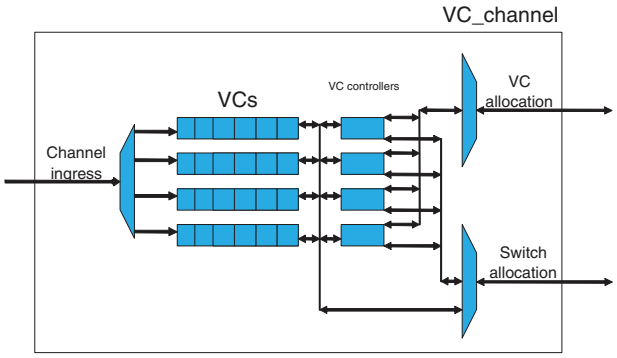

Fig. 1. Block diagram of a NoCem virtual channel implementation.

control-flow applications, network on chips will be used for both processor and reconfigurable logic-only designs. We explore both a cryptographic accelerator (processor + HW) and an 802.11 transmitter (HW only) in order to capture both forms of FPGA designs.

III. Network on Chip Configurations and Experiment Methodology

This section briefly describes the network on chip implementations we examine and the experimental methodology for running applications over these NoCs. The general terminology of network of chips is well-known, and Dally et al. [6] provides the basics of a network on chip vocabulary. The basic blocks, routing, and construction of a network on chip is described here in terms of how the implementations are realized on FPGAs.

Typical NoC designs are constructed to be a virtual channel, flit based networks whether an ASIC or FPGA architecture is targeted. Various other configuration parameters can exist for buffer lengths, datawidth, and topologies. Deterministic routing is the basic path creation for packets through the network, leading to provable deadlock-free routing, a highly desirable characteristic. Also, the flit size (i.e. the unit of flow control) typically equals the phit size (i.e. the physical transmission unit across any link) within these networks.

A. Network on Chip Components

Any network on chip is constructed of network links and switches in an arbitrary topology. We briefly look at the traditional implementations of these components, focusing on how their design can be challenging to place efficiently on FPGAs. The components considered in this work utilize various switching, buffering, arbitration, and physical link characteristics.

Physical Channel. Most network on chip implementations seen today use virtual channels within each physical link as shown in figure 1. Using virtual channels gives each physical link in the network multiple lanes so that packets can bypass one another at the network switches. This plurality of lanes in each link can lead to higher throughput of the network and facilitate deadlock free routing. While this higher bandwidth is ideal in any networking scenario, the resource utilization of a virtual channel implementation can be unattractive. As can be seen in the figure, state must be kept for each virtual channel and then arbitration must occur for the virtual channels communicating to the switch. NoC implementations differ on how many virtual channels can reach the ingress and egress switches at any given clock cycle, but this figure shows one representative implementation.

Node arbitration. Fair arbitration within the switch itself requires much more work if virtual channels are involved. There must be arbitration for both outgoing virtual channel allocation and for switch allocation. Both arbitration schemes must be able to communicate with ALL incoming virtual channels and with ALL outgoing virtual channels in order to make arbitration decisions. One of the first and best virtual channel allocation scheme involved flit-reservations in a sliding window implementation of virtual channel buffer allocations [7]. This complex arbitration scheme has not yet been implemented on FPGAs due to the fact that such large decision making storage and control would presumably not fit on current FPGA architectures.

Node Switch. The switch in a virtual channel implementation can be constructed in a variety of ways in order to gain the maximum throughput within the switch. The switch used in our virtual channel implementation is an all-to-all mux that allows multiple paths of communication simultaneously. Five transfers could occur at once theoretically, but this would require each incoming channel to have something to send AND with destinations to unique outgoing channels. Within FPGAs, this becomes a large wire routing challenge.

B. The Simple NoC Implementation

With those 3 components that create a basic network on chip, each component can be simplified in order to create the simplest network on chip implementation. The actual simplifications are described here.

Shrinking the Physical Channel. We simply throw away the notion that FPGA based network on chips should use virtual channels. By doing so, a simple one-word FIFO can be employed to act as the physical channel link between any two switches. All the allocation logic and state machines associated with pushing flits through the virtual channel lanes are no longer necessary. As long as deterministic routing is used to move packets through the network, the network is still deadlock-free. Keeping the network deadlock free is accomplished by using XY routing, where packets traverse in the X direction first, then once in the correct column, the packet traverses in the Y direction.

Shrinking the Node Arbitration With no virtual channels, there is no need for virtual channel allocation, a large control structure within each node. Without the virtual channel allocation step, there is also less sideband state and signaling that must occur within packets.

Shrinking the Node Switch. We simplify the switch by only allowing 1 switching decision to be made at any given time. This is the smallest switch we could conceivably create

Authorized licensed use limited to: University of Florida. Downloaded on March 3, 2009 at 17:02 from IEEE Xplore. Restrictions apply.

TABLE III
NETWORK ON CHIP IMPLEMENTATION COMPARISON.

	Virtual Channel	Simple
Dimensions	8x8 mesh	8x8 mesh
Buffer space per channel	8	1
# Virtual channels	2	N/A
# Transfers per switch per cycle	5	1
Max packet size	8 words	8 words
Datawidth	32b	32b

at a high level. This simplification will of course cause the total throughput of the network on chip to be lower, but this would only occur only when several applications are competing for NoC resources.

C. Network on Chip Implementation Summary

Table III summarizes the differences between our virtual channel and simple network on chip implementations. The datawidth is kept the same for both implementations at 32b. We keep the maximum packet size at 8 words per packet. This limitation keeps some fairness in the network in the virtual channel implementation (i.e. not allowing any given packet to control a large amount of buffer space within the network).

We chose to keep the datawidths and buffer slots constant throughout the experimental applications. We hold the number of virtual channels per physical channel constant at *two* lanes per channel. We found that higher count virtual channels increased performance at diminishing rates as compared to the reconfigurable logic utilization. Furthermore, the benefit of deadlock free routing can be accomplished utilizing two lanes, where additional lanes would only increase channel buffer space. While arguably altering datawidths and buffer slots would change the results, we found that comparing a single virtual channel implementation to a single simple network on chip implementation leads to interesting results.

D. Experiment Methodology

The next three sections describe the actual applications (a synthetic benchmarking application, a cryptographic accelerator, and a 802.11 transmitter). Briefly, we describe the environment in which each experiment is ran, and how results are collected.

Each application is written to run on both the simple and virtual channel NoC implementations. Functionality is identical for the applications regardless of which NoC they run on top of. One instance of the application is run on 4 locations on an 8x8 NoC; each successive placement being more affected by network traffic (i.e. placements in the middle of the chip see heavy network traffic, while placements on the corners of the mesh will not). We do not show these placements to save space for analysis and results, but the placements were made in an application specific way in order to get meaningful averages across all possible placements.

During these runs, the network is loaded by NoC traffic generators to create a desired loading on the network. This loading is made to represent other applications running in parallel to the application under test. The traffic generator communicates only to other tiles not involved in the application under test. For example, our 802.11 transmitter requires 4 tiles on the NoC. The traffic generators will ignore these tiles and only send packets to the other traffic generators. The traffic generators consume packets as soon as they are present at the NoC access point interfaces.

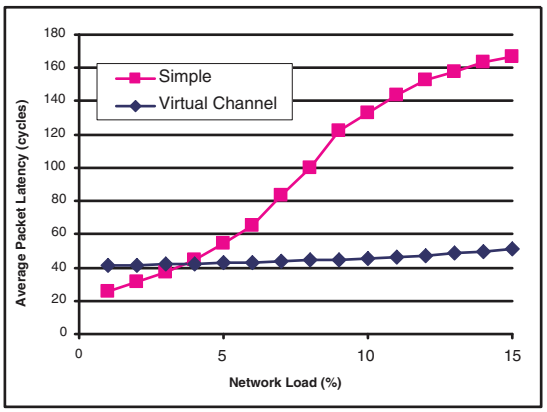

Fig. 2. The benchmarking application's average packet latency across 2 NoC implementations. Note that the Y-axis is measured in packet latency, where lower latency equates to better performance for this application.

The traffic generators create traffic using a exponential distribution of packet arrival times to the network on chip interface. This distribution models a Poisson process leading to the desired average loading of the network on chip from 1-15%. One the network on chip is loaded heavier than 15%, the network on chip was observed to be completely saturated and results reached a steady state.

The various placements of the single application on the NoC return application specific metrics of performance. These metrics will be described in the results section (section IV). These various placements are each averaged together in order to observe how the application behaves across various placement strategies.

E. Development Tools, Methodology, and Experimental Environment

All source code and development was done within the Xilinx toolflow. Specifically Xilinx ISE 9.1 and EDK 9.1 tools were used to create the FPGA designs. Modelsim 6.2b was used to debug, verify, and simulate the 8x8 network on chips. Each application was verified in hardware on either a Xilinx University Program Virtex2Pro board or a Xilinx ML505 board.

IV. RESULTS

A. The Synthetic Benchmarking Application

This application continuously sends 8 word packets among participating tiles on the network on chip. The destination addresses are randomly generated and uniformly distributed

Authorized licensed use limited to: University of Florida. Downloaded on March 3, 2009 at 17:02 from IEEE Xplore. Restrictions apply.

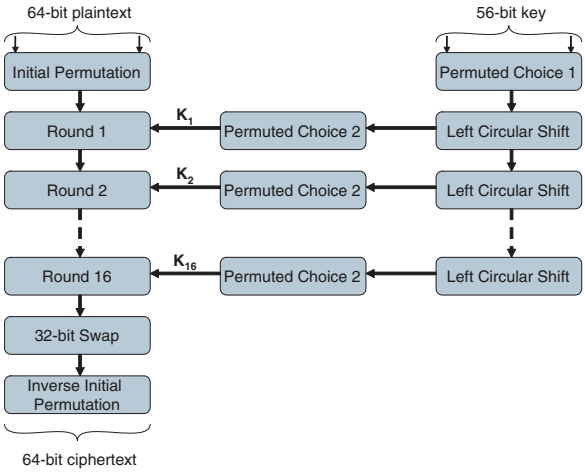

Fig. 3. The DES algorithm.

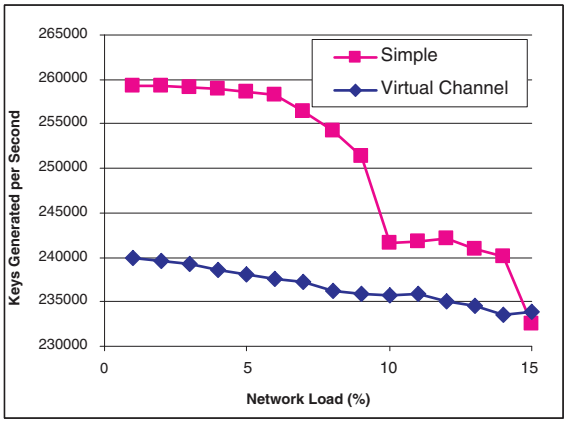

Fig. 4. The cryptographic accelerator's performance across the 2 NoC implementations.

and the application itself consumes the entire 64 nodes on both NoC implementations. The metric of performance of this application is *average packet latency*, where of course a lower latency equates to better system performance. Figure 2 on the previous page shows the results of running this 64-node benchmarking application on both network on chip implementations.

As can be seen, at low levels of network loading (i.e. the rate at which the tiles send packets), the simple implementation of the network on chip has a lower packet latency than the virtual channel implementation. At 4% loading, the two implementations cross and the simple implementation quickly struggles to keep up with the virtual channel implementation. The virtual channel implementation moves packets consistently through the network as each traverse in the path reserves a flow for following packet words. This behavior is similar to most other virtual channel benchmarking applications published in various research [8]. The simple implementation at the highest network loading levels is still able to deliver packets, but as there is little in-network buffering, packets spend a good deal of time waiting to gain access to the interconnect.

This number of 4% should also be compared back to Table II where many of the applications did not use the onchip interconnect frequently. Again, it is difficult to compare bus to network on chip utilization numbers, but this does show that some applications may not benefit from a virtual channel implementation. The next two sections show how real applications utilize network on chip architectures.

B. The Cryptographic Accelerator Application

Cracking cryptographic algorithms are well understood when using simple brute force methods. Typically, this is done giving the algorithm plain text P_1 and retrieving the cypher text C_1. Then, the cryptographic algorithm is cycled with P_1 and every possible key $K_1 ... K_n$ until the same cypher text is generated. From that information, the original key is found the attack has been successful. Clearly though, like any usable cryptographic algorithm, it is provably hard to do this task in a timely manner.

The DES standard specifically uses 16 rounds of permuting a key and plain text resulting in a cipher text. Figure 3 shows the algorithm at a high level. In this example application domain, a simple SW/HW DES brute force cracker is created that uses a single processor and a HW accelerator to encrypt plain text with ALL possible keys. Architecturally, this is accomplished with a Microblaze processor that communicates to a dedicated DES encryption engine that is located on the NoC. The Microblaze simply manages the keys and surrounding software, while the DES encryption algorithm can quickly (10 clock cycle average) encrypt plain text. The performance metric with this application is how many keys can be generated per second.

This task uses the network on chip sporadically, feeding the accelerator keys as quickly as possible, but spending most of its time either checking (in SW) or generating keys (in HW) as part of the algorithm. Fig 4 shows that only after a heavy loading of the network on chip, do the two implementations produce the same amount of keys per second. Interestingly, it can be observed that the virtual channel implementation has a much more graceful degradation in performance, while the simple implementation again has a rapid slope in its performance curve.

C. The 802.11 Transmitter Application

We took an existing 802.11 transmitter partitioned over a network on chip [9] and made slight modifications so that it could work on top of both network on chip implementations. Four processing tiles are required for this application. The first processing tile, a source node, generates samples for transmission. The second tile modulates the signal using a BPSK modulation scheme. The third tile then takes the modulated samples and using a DFT (discrete Fourier transform), sends the signals to the fourth processing tile, a sink node on the NoC that is attached to a digital to analog converter.

This pipeline of signal processing will stress the network on

Authorized licensed use limited to: University of Florida. Downloaded on March 3, 2009 at 17:02 from IEEE Xplore. Restrictions apply.

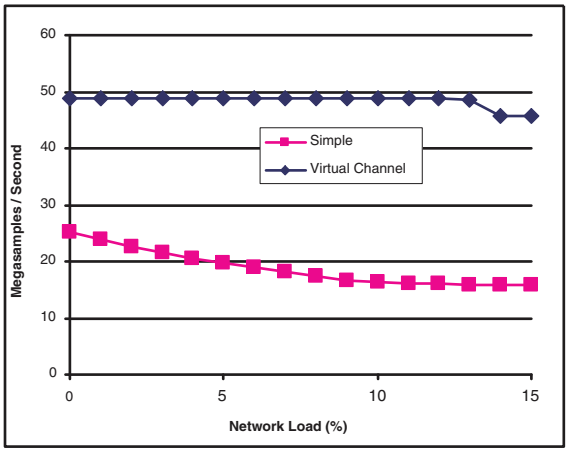

Fig. 5. The 802.11 transmitter application's behavior across the 2 NoC implementations.

chip, as it truly is a dataflow application that will attempt to push as many packets through the network as possible. In the face of contention on the network on chip with competing applications, this 802.11 transmitter will create bottlenecks on the network, testing arbitration schemes and switching bandwidth.

Figure 5 shows the results of the 802.11 transmitter on the two NoC implementations. This application loads the network at approximately a 50% rate, limited by the DFT block implementation and handshaking costs at the network interfaces. This high level of loading is typically not achievable by FPGA designs utilizing a processor-bus interface, but is clearly possible with network on chip architectures.

This application is also interesting in that the network on chip has little influence on the virtual channel implementation, while also showing a substantial degradation in performance on the simple implementation. While the synthetic benchmarking application and the cryptographic accelerator did not benefit much from a virtual channel implementation, clearly here the opposite is true. The network is so saturated with this application's packets, that the actual computation delay (within the modulation and the DFT blocks) is great enough to mask away any delay caused by samples reserving paths through the network.

Overall, a virtual channel implementation guarantees a high throughput (2-2.5x more than simple implementation) for a 802.11 transmitter across all network loads.

D. Results summary

Looking at the three applications ran, there is not one best network on chip implementation for all applications. For streaming data through the network, virtual channels are a must, allowing larger packets to travel through the network in a flowing manner. In control flow systems (i.e. processors –> memory, processor –> computation accelerators), applications are fairly resilient to low network loads. Specifically, this can be seen in the cryptographic accelerator, where most of the work is done in computation, not moving data on chip. In data hungry applications, where computation is cheap and communication expensive, more logic resources should be spent on the NoC, leading to results similar to the 802.11 transmitter experiment.

One metric of success that we did not examine was logic resource utilization of these 8x8 NoCs. The virtual channel implementation is on average 400-500% larger than the simple implementation as shown back in Table I. This factor has influenced current research projects and can not be ignored in industry.

V. Conclusions

In this paper, we presented two representative general purpose NoC implementations (virtual channel and simple physical channel) to demonstrate how real applications would perform under a range of network loads. We showed that a complex NoC architecture does not always lead to better application performance across a benchmarking and a cryptographic accelerator application. We also showed that data-flow applications benefit greatly from virtual channel implementations demonstrated by the 802.11 transmitter.

FPGAs and reconfigurable computing allow users to configure and combine pre-existing IP to create complex designs. From this work, it is clear that the chosen configuration of a network on chip greatly determines whether a design will meet area and performance constraints. Hopefully as general purpose NoC interconnects become available as standard IP, the underlying NoC architecture will be flexible enough to support the wide variety of application and network loads that will exist on these reconfigurable platforms.

References

[1] I. Kuon and J. Rose, "Measuring the gap between FPGAs and ASICs," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 26, no. 2, pp. 203–215, Feb. 2007.

[2] B. Sethuraman, P. Bhattacharya, J. Khan, and R. Vemuri, "LiPaR: A lightweight parallel router for FPGA-based networks-on-chip," in *GLSVSLI '05: Proceedings of the 15th ACM Great Lakes symposium on VLSI*. New York, NY, USA: ACM Press, 2005, pp. 452–457.

[3] G. Schelle and D. Grunwald, "Onchip interconnect exploration for multicore processors utilizing FPGAs," in *2nd Workshop on Architecture Research using FPGA Platforms*, 2006.

[4] N. Genko, D. Atienza, G. D. Micheli, J. M. Mendias, R. Hermida, and F. Catthoor, "A complete network-on-chip emulation framework," *Design, Automation and Test in Europe*, vol. 01, pp. 246–251, 2005.

[5] J. B. Pérez-Ramas, D. Atienza, M. Peón, I. Magan, J. M. Mendias, and R. Hermida, "Versatile FPGA-based functional validation framework for networks-on-chip interconnections designs." in *PARCO: International Conference on Parallel Computing*, 2005, pp. 769–776.

[6] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the Design Automation Conference*, Las Vegas, NV, June 2001, pp. 684–689.

[7] L.-S. Peh and W. J. Dally, "Flit-reservation flow control," *HPCA*, vol. 00, p. 73, 2000.

[8] D. Wiklund, S. Sathe, and D. Liu, "Benchmarking of on-chip interconnection networks," in *Proc of the International Conference on Microelectronics (ICM)*, 2004.

[9] G. Schelle, J. Fifield, and D. Grunwald, "A software defined radio application utilizing modern FPGAs and NoC interconnects," *Field Programmable Logic and Applications, 2007. FPL 2007. International Conference on*, pp. 177–182, Aug. 2007.

Authorized licensed use limited to: University of Florida. Downloaded on March 3, 2009 at 17:02 from IEEE Xplore. Restrictions apply.