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.

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 / 19,601	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

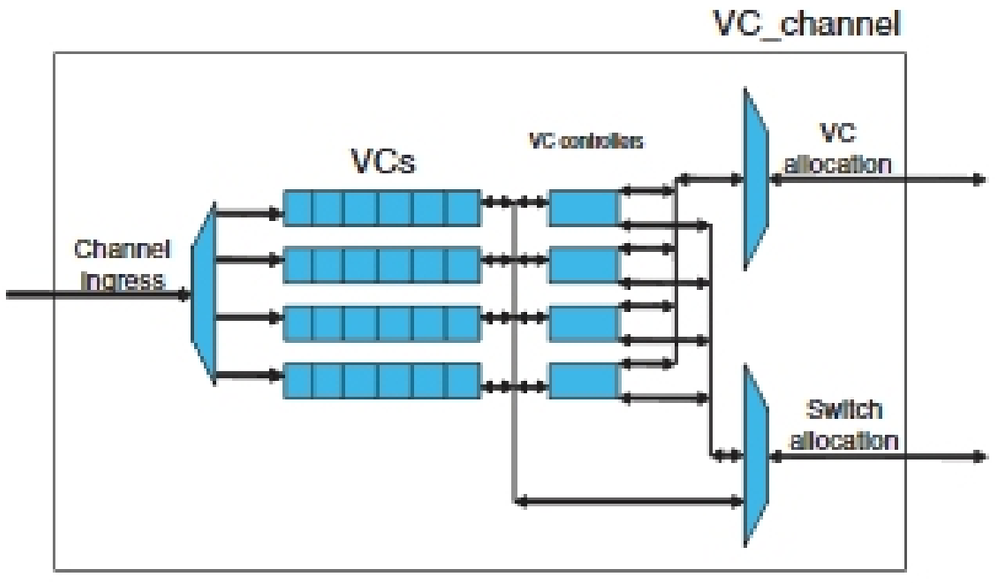

Fig. 1. Block diagram of a NoCem virtual channel implementation.

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