A Network on Chip based gigabit Ethernet router implemented on an FPGA

Andreas Ehliar and Dake Liu
Dept. of Electrical Engineering
Linköping University
S-581 83 Linköping, Sweden
{ehliar,dake}@isy.liu.se

Abstract— **In this paper we will share the experiences we have gained from implementing an FPGA-based gigabit Ethernet router based on the SoCBUS network on chip architecture. The main reason for this project has been to test the SoCBUS architecture in a real design in order to investigate the performance and possible shortcomings. Another reason is to evaluate how SoCBUS performs in an FPGA. The results highlights several areas where the FPGA implementation of SoCBUS can be improved.**

I. Introduction

Networks on Chip (NoC) has been a hot research topic for a long time and there are many publications in the area. Most of the publications deal with subjects like simulation environment, verification, testing, power consumption, and implementations of core components like router nodes. However, surprisingly few publications deal with implementations of entire systems using a general network on chip architecture. Therefore we decided to implement a NoC-based system to widen our knowledge in this area. The system is based on an earlier case study on an Internet core router with 16 ports with a full duplex bandwidth of 10 gigabit each. The limitations of the FPGA board made it necessary to scale the design down to only 2 full duplex gigabit Ethernet ports. While this severely limits the usability of the router it will serve as a test bed for NoC in a real implementation.

Some of the problems that various network on chip projects are trying to solve are not applicable for an FPGA. For example, the problem of physically creating high speed global interconnection in the FPGA has already been solved by the FPGA vendor.

Other problems addressed by the NoC concept are very valid in an FPGA as well as in an ASIC. An example of this is the fact that a high speed crossbar with many ports does not scale very well, especially if low latency is desired.

On the other hand, some problems are more difficult to solve in an FPGA. For example, it is very difficult to create a high speed crossbar with many ports in an FPGA if low latency is desired.

The rest of this paper is organized as follows, section II contains background on the SoCBUS NoC architecture, section III describes the FPGA-based implementation, section IV contains the benchmarking results of the design, section V contains a discussion of the results, and finally section VI summarizes the paper.

II. The SoCBUS network on chip architecture

SoCBUS is a Network on Chip architecture designed with simplicity in mind. The transport protocol has been kept simple leading to small and efficient NoC nodes. The basis of SoCBUS is a circuit switched network. It may use either source routing or distributed routing.

A connection is setup in the following way; the source sends a setup request to the destination which will return an ACK if it is ready to receive data. If a link required to reach the destination is locked by another connection a NACK is returned to the source and the source will have to try again at a later time.

As soon as a connection has been established, the sender has an exclusive lock on that channel and may transmit data at will, without any flow control.

SoCBUS was originally envisioned for ASICs where it can operate at a clock frequency of 1.2 GHz in a 180 nm process [1]. The number of data lines in a SoCBUS link is not defined by the SoCBUS protocol but can be changed to match the requirements of the application.

III. The FPGA-based router prototype

The FPGA-based gigabit Ethernet router is based on an earlier case study which studied a 16 port 10 gigabit IP router [2]. The case study studied simulations that were run in the SoCBUS simulator to benchmark three different NoC configurations. The final architecture is shown in figure 1. This design was benchmarked in the SoCBUS simulator and the architecture could handle up to 14 Gbit/s per port when using Internet mix traffic while running SoCBUS at 1.2 GHz with a link width of 64 lines. With minimum size packets, only 2.6 Gbit/s per port could be handled. The main bottleneck in this case was the transmission of the requests to the forwarding table as such small packets are not handled gracefully by the NoC.

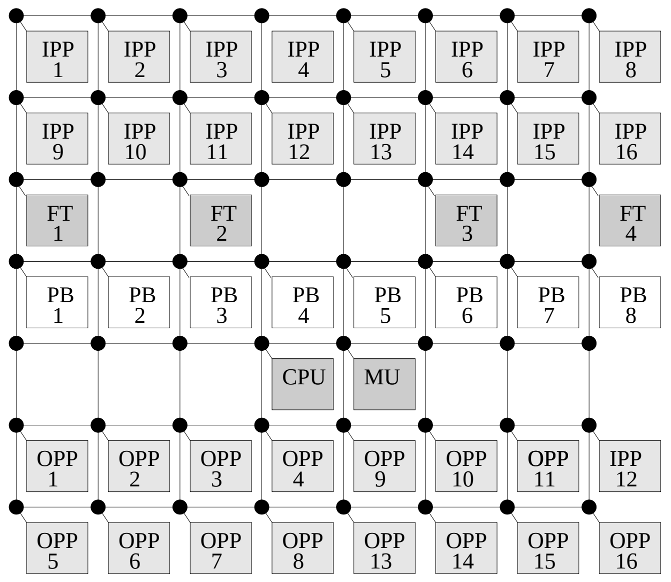

Fig. 1. The architecture of the earlier case study. IPP means Input Packet Processor, OPP is an Output Packet Processor, FT is a Forwarding Table node, PB is a packet buffer node, MU is a multicast unit and CPU is the control processor.

TABLE I

THE UTILIZATION OF THE FPGA. LUT REFERS TO THE 4 INPUT LOOK-UP TABLES THE FPGA CONSISTS OF AND BRAM REFERS TO THE 18 KBIT RAM BLOCKS LOCATED IN THE FPGA FABRIC.

Unit	LUT	Flip Flops	BRAM
Input module	1139	856	2
Output module	444	271	3
Packet buffer	3395	3470	18
Forwarding table	3234	1745	12
SoCBUS	11565	3944	0
Entire design	19047	10137	41
Total size of FPGA	46080	23040	120

Unfortunately, it would not be feasible for us to implement this architecture on any FPGA. Due to the limits of the Virtex II 4000 FPGA we were using we could only run SoCBUS at 80 MHz. We selected a SoCBUS link width of 32 lines instead of the 64 lines originally envisioned. SoCBUS was used in the distributed routing mode in this project.

The application was likewise scaled down to only two full duplex gigabit Ethernet ports due to limitations of the FPGA but mostly because that is all we had available on our prototype boards. (It may seem odd at first to have a router with only two gigabit Ethernet ports but logically they were used as two separate in ports and two separate out ports.)

In addition, another goal of this project was to serve as a test bed for other research projects. To this end, a programmable packet classifier and a forwarding table developed at our division has also been incorporated into the FPGA implementation [3] [4]. A DDR memory-based packet buffer was also developed specifically for this prototype.

The final architecture of the FPGA implementation is

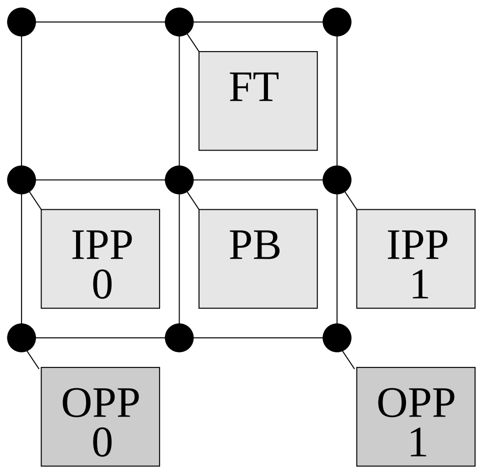

Fig. 2. The architecture of the FPGA prototype.

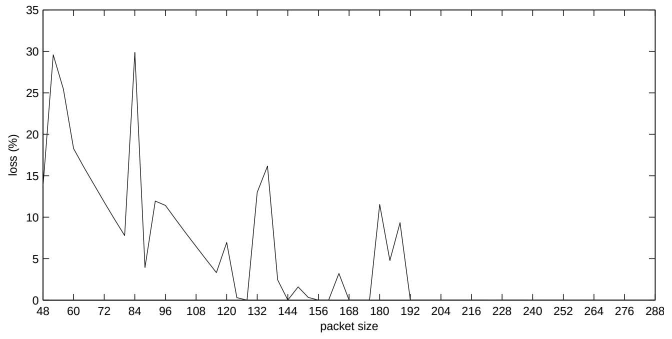

Fig. 3. The performance of the router for packet sizes 48, 52,. . .,288. The packet size refers to the payload size and does not include the Ethernet source address, destination address, and CRC. There is no packet loss for packet sizes larger than 288 bytes.

shown in figure 2 and table I summarizes the FPGA utilization. Note that the numbers for individual modules is taken from the synthesis report for that particular module. Therefore the total size of the design is somewhat smaller as global optimization can reduce the individual modules somewhat.

IV. BENCHMARKING OF THE FPGA IMPLEMENTATION

The system was tested and benchmarked in two ways. At first, iperf was used on two 3GHz Pentium 4-based computers to benchmark the throughput of the router. Unfortunately these computers were not fast enough once the packet size was decreased. Instead, a simple design was used on to generate traffic and verify that the traffic arrived correctly. This design was used on FPGA boards identical to the one the router was running on.

To simplify the testing equipment the router was benchmarked by sending data from inport 0 to outport 1 and from inport 1 to outport 0. A minimum gap between packets were used. The result of the benchmark is summarized in figure 3. As can be seen in the figure, the performance varies wildly for small packet sizes. The reason for the large performance differences has not been fully determined, but the initial spike at 52 byte can be explained by the fact that the input module allocates buffer space in chunks of 16 bytes. When going from 48 to 52 byte packets, the input module needs to allocate 4 buffers instead of 3.

V. Discussion

As expected, the NoC architecture made it easy to integrate the different components and if necessary change the network design without any big problems. However, we also experienced a number of problems. Some are specific to the FPGA solution and some are applicable to both an ASIC and an FPGA-based solution. Due to the experiences gained in this project we have some proposals for future on chip interconnect research.

A. Problems encountered

It is important to realize that the SoCBUS concept was never intended for an FPGA implementation and the authors therefore expected some problems in this particular implementations.

One of the most unexpected results of this project is that the hardware cost of the NoC router nodes in the FPGA was higher than expected. About 25% of the LUTs in the FPGA was used for SoCBUS. On average, this corresponds to almost 15 LUTs per line. The current SoCBUS router is not optimized for an FPGA architecture however and we believe that it could be improved.

A bigger problem is the latency of connection setup. In the case of a small packet it might take as much time sending the packet as setting up the connection if the connection was accepted immediately. This is a large problem if minimum sized packets arrive with minimum inter packet gap size and the main reason for the packet loss seen with smaller packets.

Another potential problem is that we have not as yet looked into how to floor plan the NoC as can be seen in figure 5. The network on chip nodes are placed relatively arbitrarily by the placement tool instead of the ordered architecture seen in figure 2.

Another problem is that the chosen architecture leaves no guarantees about performance. While it is possible to schedule SoCBUS connection setup and teardown to guarantee a certain level of performance this is hard to do in an IP router where the performance is heavily dependent upon the pattern of the incoming traffic. This is in contrast with for example a base station where the task load can be known beforehand making it possible to schedule SoCBUS transaction to handle the worst case load [5].

Finally, the chosen architecture does not scale gracefully as can be seen in figure 3. The performance oscillates wildly between different packet sizes once the NoC is starting to get congested. Further study of the network is required to understand this behavior.

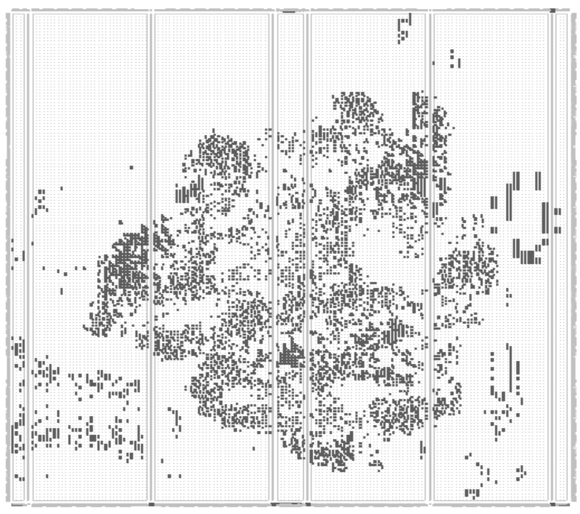

Fig. 5. The floorplan for the SoCBUS part of the FPGA implementation. SoCBUS is scattered arbitrarily and is placed together with other logic.

B.

Future possibilities There are two features that has not been used in the current design. The first is speculative sending of data, i.e., sending data before an acknowledgement that the connection setup is successful has been received. If a negative acknowledgement on the connection setup is received the data will have to be resent.

Another feature is an extension to SoCBUS specifically intended for short packets that can be transmitted in only one clock cycle. This would be ideal for requests to and from the forwarding table.

To summarize this project, we do not believe that the current implementation of SoCBUS is well suited to FPGAs, instead another NoC architecture should be developed to match the opportunities and constraints of current FPGAs. For example, they should take into account the wiring structure of FPGAs. It could also be argued that since an FPGA cannot contain a large number of IP cores, a NoC is not necessary to connect them together. This argument however does not take into account that it is hard to create a low latency cross bar with many high speed ports in an FPGA. On the other hand, the limited number of cores in an FPGA means that the NoC does not need to be able to scale very well as the number of network nodes increases.

It could therefore make a lot of sense to use a NoC in a rather conservative configuration, acting as a bridge between local crossbars of tightly connected cores. The NoC would in this case consist of very few nodes because the majority of the communication would happen in the local crossbars. This is illustrated in figure 4. The local crossbars could use either traditional protocols for on chip interconnect such as AMBA, CoreConnect, or slightly modified versions of these. The advantage is that most communication would not need to be aware of the

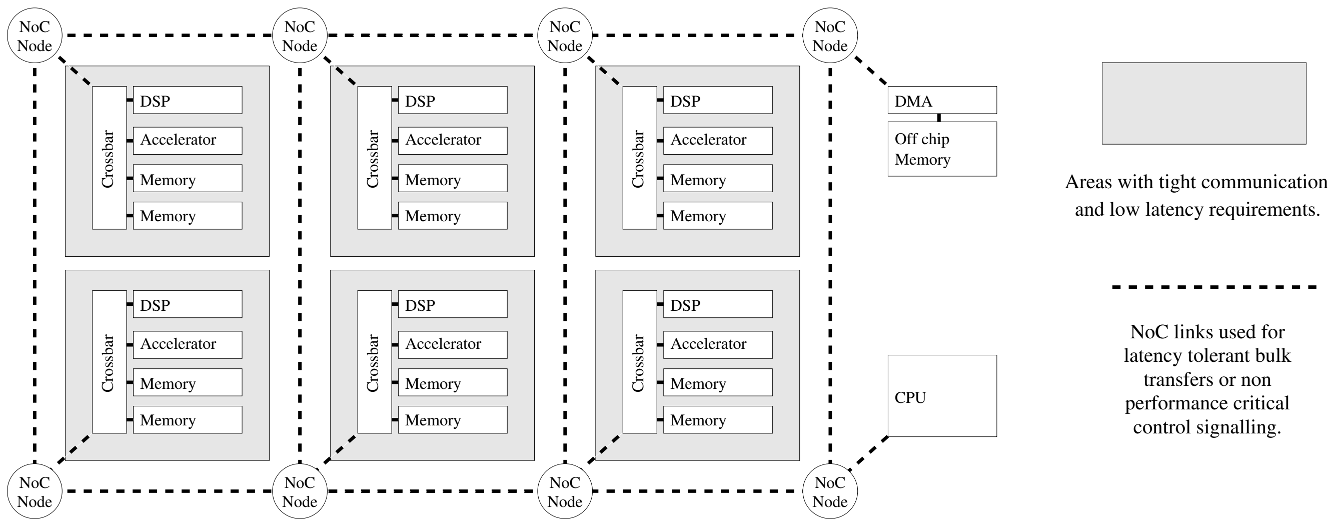
NoC Node
NoC Node
NoC Node
NoC Node
Crossbar
DSP
Accelerator
Memory
Memory
DMA
Off chip Memory
CPU
Areas with tight communication and low latency requirements.
NoC links used for latency tolerant bulk transfers or non performance critical control signalling.

Fig. 4. A conservative NoC approach where the NoC is used to connect crossbars using more traditional bus protocols.

NoC transport protocol at all and does not need to take the higher latency inherent in the NoC into account.

VI. Conclusions

In this paper we have investigated an IP router implemented using the SoCBUS network on chip architecture. Our experience during this project shows that a NoC designed for an ASIC does not map very well to an FPGA. The largest problem encountered is the latency involved in connection setup and tear down which caused drastically reduced performance when small packets were used.

On the other hand, there exists an opportunity to design a NoC suited to FPGA applications. This NoC should be optimized for the FPGA structure. However, since the number of cores on an FPGA will be quite limited compared to an ASIC, the NoC does not have to scale as well as an ASIC-based NoC.

Acknowledgements

Many thanks to Tobias Borslehag and Daniel Ferm for their valuable work on this project.

References

[1] S. Sathe, D. Wiklund and D. Liu, "Design of a switching node (router) for on-chip networks", *ASIC, 2003. Proceedings. 5th International Conference on*, vol. 1, 2003.

[2] J. Svensson, "Design of a core router using the SoCBUS on-chip network", Master's thesis, Linköping University, 2004.

[3] T. Henriksson, *Intra-Packet Data-Flow Protocol Processor*, PhD thesis, Linköping University, May 2003.

[4] A. Ehliar and D. Liu, "Flexible Route Lookup Using Range Search", *Communications and Computer Networks, IASTED International Conference On*, 2005.

[5] D. Wiklund and D. Liu, "Design, mapping, and simulations of a 3G WCDMA/FDD basestation using network on chip", *Proc of the International workshop on SoC for real-time applications*, 2005.