

EXTENDED-BUTTERFLY FAT TREE INTERCONNECTION (EFTI) ARCHITECTURE FOR NETWORK ON CHIP

Hemayet Hossain, Md. Mostofa Akbar and Md. Monirul Islam

Department of Computer Science & Engineering
Bangladesh University of Engineering & Technology
Dhaka-1000, Bangladesh
{hemayet, mostofa, mmislam}@cse.buet.ac.bd

ABSTRACT

System on Chip (SoC) design requires efficient communication between heterogeneous resources to meet the high-speed transmission needs. Therefore one of the key factors for the success of ultra-deep submicron technologies will be the capability of integrating different resources like processor core, memory, an FPGA, a custom hardware block or any other semiconductor intellectual property (SIP) block into a single piece of silicon. Non-scalable global wire delays, global synchronization failure, loss of signal integrity issues are the main problems. To address these problems, various interconnect architectures are proposed. Butterfly Fat Tree (BFT) is one of those. To improve the performance of BFT we introduce Extended-butterfly Fat Tree Interconnection (EFTI). Routing algorithm is provided for EFTI and comparative analysis is performed through the simulation result.

Keywords: SoC, SIP, Switch, BFT, EFTI.

1. INTRODUCTION

Ever since the innovation of the integrated circuit in the 1960's, the number of transistors available on one chip has increased with approximately 60 percent a year and with these continual developments of technology, communication between various ICs has also increased day by day. According to ITRS 2002 [1][2], the realization of complex Systems on a Chip (SoCs) consisting of billions of transistors fabricated in technologies characterized by 65 nm feature size and less will soon be feasible. By the end of the decade, SoCs, using 50-nm technology operating below one volt, will grow to 4 billion transistors running at 10 GHz, according to the ITRS. Such SoCs imply the seamless integration of numerous IPs like DSP and FPGA performing different functions and operating at different clock frequencies. The integration of several heterogeneous components into a single system gives rise to new challenges. With the change of dramatic improvement in this area it is essential

to have an adaptable communication facility that can cope up with the versatile programming of the cores. Such systems will have to process data in real time, perform data transfer at the rate of hundreds of Tbps [3], support multiple functions and protocols for communications with standard wired and wireless interface, provide security and secrecy and cope-up with time-to-market (TTM) pressures and so.

Existing SoCs' communications are based on dedicated wires and shared bus (single/hierarchical) having various constraints. Dedicated wires do not provide flexibility for communication needed for hardware platforms and cannot cope-up the increased number of cores [4]. Due to exclusive access of shared bus its utilization is as low as 10%. This is inflexible to parameters required supporting heterogeneous components of SoC and not scalable. Global synchronization is not possible with this technique [5]. It has very high signaling delay per unit length. Reduction in SNR and signal integrity and increase in cross talk occur with the increase in parasitic capacitance and resistor as bus wire length increases.

To achieve the above-mentioned requirements and to overcome the above-mentioned problems the use of a network-centric approach i.e. *Network on Chip* (NoC), which is Globally Asynchronous and Locally Synchronous (GALS) [5], to integrate IPs in complex SoCs is coming to the way. Here the most important thing is here to develop the communication IP (CIP) for structured and systematic integration of heterogeneous functional IP blocks to enable widespread use of the SoC design methodology.

Our proposed architecture is based on the butterfly fat tree architecture where the IP blocks are at the leaves of the tree and the switches at its vertices [2]. Here a group of IPs are connected to a neighboring switch. Global signals, spanning a significant portion of a die in more traditional bus-based architectures, have to span the distance separating switches in this interconnection. These switches (CIPs) are key components of this infrastructure enabling seamless integration of the functional IP blocks. In this scenario, global wires only consist of top level

interconnects between switches. The specification of the interconnection can be known at early stages of the design process and so a better prediction of the electrical parameters of the interconnection and overall system performance can be achieved.

In this paper we are proposing an extended-butterfly fat tree interconnection (EFTI). We provided the routing algorithm for the CIP to support the communication. We have developed a simulator to simulate both butterfly fat tree and extended-butterfly fat tree interconnection to calculate the throughput, average packet delay and some other parameters. We contrast these parameters of EFTI-based SoCs with those associated with BFTI-based SoCs. The remainder of this paper is organized as follows. In Sec. 2, an overview of the related work is given. Sec. 3 details the proposed EFTI interconnect architecture. In Sec. 4 detail routing algorithm for EFTI is provided. Sec. 5 describes the simulation environment. Sec. 6 analyses experimental results. Sec. 7 concludes by highlighting the salient features of our work and further study.

2. RELATED WORKS

Existing SoC designs mainly use shared bus to interconnect functional IP blocks. There are mainly three types of commercially used SoC interconnect specification [2], ARM AMBA [6] bus, Wishbone and IBM CoreConnect [7]. All of them suffer the drawback of non-scalability. A few on-chip micro network proposals for SoC integration can be found in the literature. Sonic's Silicon Backplane [8] is one example. This is a bus-based architecture in which the IP blocks are connected to the bus through specialized interfaces called *agents*. Each core communicates with an agent using the Open Core Protocol (OCP) [9]. Agents communicate with each other by using TDMA bus access schemes. These agents effectively decouple the IP cores from the communication network. The basic interconnect architecture is still bus-based and will hence suffer from performance degradation trends common for buses.

MIPS Technologies has introduced an on-chip switch integrating IP blocks in a SoC [10]. The switch called *SoC-it* is intended to provide a high-performance link between a MIPS processor and multiple third party IP cores. It is a central switch connecting different peripherals, but only in a point-to-point mode. Kumar [11] and Dally [12] have proposed mesh-based interconnect architectures. These architectures consist of an $m \times n$ mesh of switches interconnecting computational resources (IPs) placed along with the switches. Each switch is thereby connected to four neighboring switches and one IP block. In this case, the number of switches is equal to the number of IPs.

In [12], Dally and Towles proposed the use of a torus interconnect architecture. A variation of the torus

architecture, which eliminates the use of long wraparound wires, called a folded torus, is described in [13].

Saastamoinen [14] describes the design of a reusable switch to be used in future SoCs. The interconnect architecture is however not specifically discussed.

In [15] Forsell describes Eclipse (embedded chip level integrated parallel super computer) system that uses a sparse 2D mesh network, in which the number of switches is at least the square of the number of processing resources divided by four. Eclipse uses cache-less shared-memories, so it has no cache coherence problems, and communication will not jam the network even in the case of heavy random access traffic.

Guerrier and Greiner [3] proposed the use of a fat tree based interconnect (SPIN) and addressed system level design issues.

In [16] Karim et al. proposed the Octagon network in the context of network processor design. It is a direct network. Similar to that in the fat tree topology, the point-to-point delay is also determined by the relative source/terminus locations, and communication between any two nodes (within an octagon subnetwork) requires at most two hops.

In [17] and [18] P. P Pande et al. proposed the butterfly fat tree interconnect architecture, modified form of fat tree, for a networked SoC as well as provided the associated design of required switches and addressing mechanisms.

All of the above mentioned works proposed some kind of interconnect architecture to solve the global wire delay problem.

3. EXTENDED BUTTERFLY FAT TREE INTERCONNECT ARCHITECTURE

The extended butterfly fat tree interconnection (EFTI) (Fig. 1) is a derivative of the butterfly fat tree (Fig. 3) architecture which is the derivative of fat tree architecture [19]. The architecture uses switches of constant size. In this network, the IPs are placed at the leaves and switches placed at the internal nodes.

Each node (IP) is identified by a pair of coordinates (l, p) , where l denotes a node's level and p denotes its position within that level. In general, at the lowest level, there are N IPs with addresses ranging from 0 to $(N-1)$. The pair $(0, p)$ denotes the locations of IPs at that lowest level. Each IP has one resource network interface (RNI) through which the IP is connected with the switches. The RNI has one port having two unidirectional physical links.

Each switch, denoted by $S(l, p)$ has four child ports, two sibling ports and two parent ports. The IPs are connected to $N/4$ switches at the first level. The number of levels depends on the total number of IPs, i.e. for N number of IPs, the number of levels will be $\log_4 N$. In the l -th level of the EFTI tree, there are $N/2^{l+1}$ switches.

So for each functional IP at coordinate $(0,i)$ the parent is at coordinate $(1,p)$ where $p = \lfloor i/4 \rfloor$.

For each switch at coordinate (l,i) the parents are at coordinate $(l+1,p_1)$ and $(l+1,p_2)$ where $p_1 = \lfloor i/2^{l+1} \rfloor * 2^l + i \bmod 2^{l-1}$ and $p_2 = p_1 + 2^{l-1}$. Child relation is just reverse.

Siblings are at coordinate (l,s_1) and (l,s_2) where $s_1 = \lfloor i/2^{l+1} \rfloor * 2^{l+1} + (i+3*2^{l-1}) \bmod 2^{l+1}$ and $s_2 = \lfloor i/2^{l+1} \rfloor * 2^{l+1} + (i+2^{l-1}) \bmod 2^{l+1}$. The floor plan of this configuration is given in Fig. 2. By using sibling interconnection (in group) bottleneck of upper layer switches is removed.

4. SWITCHING AND ROUTING

Wormhole switching is used to reduce the buffer size of the switches. Here packets are divided into fixed size flow control units (*flits*). The first flit, i.e., *header flit* (fig. 4a) of a packet contains routing information. Header flit decoding enables the switches to establish the path and subsequent data flits (fig. 4b) simply follow this path in a pipelined fashion. As the transmission of distinct messages cannot be interleaved or multiplexed over a physical channel the message must cross the channel in its entirety before the channel can be used by another message. So the problem of deadlock arises causing the decreased overall system throughput. To solve this throughput degradation problem *Virtual Channel* concept [20] is proposed. A physical channel is divided into several logical or virtual channels multiplexed over a single physical channel. Basically, buffers are divided into *parallel lanes* and flits of different packets may occupy different parallel lanes corresponding to a single channel buffer. As a result, if flits from a particular packet get blocked into a lane then flits from other packets can use other lane buffers and, ultimately, the physical channel.

The routing determines of output physical channel taken by a header flit (data flit simply follows) in an intermediate switch to reach the destination IP. The following routing is used in EFTI to determine the next link for a header flit having destination *dest* at switch $S(l,i)$.

$$\begin{aligned} ownRangeStart &= \lfloor i/2^{l-1} \rfloor * 2^l \\ leftSibIndex &= \lfloor i/2^{l+1} \rfloor * 2^{l+1} + (i+3*2^{l-1}) \bmod 2^{l+1} \\ rightSibIndex &= \lfloor i/2^{l+1} \rfloor * 2^{l+1} + (i+2^{l-1}) \bmod 2^{l+1} \\ leftRangeStart &= \lfloor leftSibIndex/2^{l-1} \rfloor * 2^l \\ rightRangeStart &= \lfloor rightSibIndex/2^{l-1} \rfloor * 2^l \\ \text{if}(dest \geq ownRangeStart \text{ and } dest < (ownRangeStart + 2^l)) \\ &\quad \text{route to corresponding child link} \\ \text{else if}(dest \geq leftRangeStart \text{ and } dest < \\ &\quad (leftRangeStart + 2^l)) \end{aligned}$$

$$\begin{aligned} &\quad \text{route to left sibling} \\ \text{else if}(dest \geq rightRangeStart \text{ and } dest < \\ &\quad (rightRangeStart + 2^l)) \\ &\quad \text{route to right sibling} \\ \text{else} \\ &\quad \text{route to any one of the parent link randomly} \end{aligned}$$

5. SIMULATION ENVIRONMENT

We developed a simulator in java supporting wormhole switching and deadlock-free deterministic routing for both Butterfly Fat Tree and Extended-butterfly Fat Tree. The network is formed automatically depending on the number of IPs. Normal distribution (mean = 50 cycles/packet/node) was used to generate uniformly distributed traffic by functional IPs. Functional IP injected variable-size packets (mean = 200 bytes) to random destinations except for themselves at a uniform distribution rate. Each simulation was run 10 times for 100000 cycles and then average was taken. We investigated the average latency of packets, the network throughput, link utilization and buffer utilization by changing the number of virtual channels and buffer size. Latency of a packet is calculated from the instant the packet's flits are created to that the last flit of the packet is accepted at the destination, including source queuing time. Throughput is defined as the number of flits received per cycle per IP.

6. SIMULATION RESULT

Some of the simulation result is given in Fig.5 and Fig. 6. Others are not given here due space limitation. In Fig. 5 throughput of EFTI and BFT is plotted with the increase of buffer size. It shows that EFTI produces much better throughput as was expected. In Fig. 6 average latency of packet of EFTI and BFT are plotted with buffer size. It also shows dramatic improvement of latency in case of EFTI. Fig. 5 also shows that throughput increases with increase of buffer size. Fig. 6 also shows latency increase with the increase of buffer size as more queuing delay incurred.

7. CONCLUSION

In this paper we showed that by using EFTI as interconnection architecture packet latency and throughput can be improved dramatically. This is due to the interconnection between the switches at the same level and closer group. Here shortest paths are created and so for some communications the packets need not to travel up to least common ancestor. We also described a suitable deterministic routing algorithm. Future work will compute required total silicon area for switches an interconnection.

8. REFERENCES

[1] ITRS 2002 Documents, <http://public.itrs.net/Files/2002Update/Home.pdf>.

[2] C. Grecu, et al, "A Scalable Communication-Centric SoC Interconnect Architecture," *Proc. of ISQED 2004*, San Jose, California, USA, pp. 343-348, March, 2004.

[3] P. Guerrier, A. Greiner, "A generic architecture for on chip packet-switched interconnections," *Proc. of DATE 2000*, Paris, France, pp. 250-256, 2000.

[4] W. J. Dally, B. Towles, "Route packets, not wires: on-chip interconnection networks," *Proc. of DAC2001*, Las Vegas, Nevada, USA, pp. 684-689, June, 2001.

[5] A. Hemani, et al, "Lowering power consumption in clock by using globally asynchronous locally synchronous design style," *Proc. of DAC 1999*, pp. 873-878, June 1999.

[6] AMBA Bus specification, www.arm.com.

[7] CoreConnect Specification, www-3.ibm.com/chips/products/coreconnect/

[8] D. Wingard, "MicroNetwork-Based Integration for SoCs," *Proc. of DAC 2001*, Las Vegas, Nevada, USA, pp. 673-677, June, 2001.

[9] Open Core Protocol, www.ocpip.org.

[10] MIPS SoC-it, www.mips.com.

[11] S. Kumar, et al, "A Network on Chip Architecture and Design Methodology," *Proc. of ISVLSI*, pp. 117-124, 2002.

[12] W. J. Dally, B. Towles, "Route Packets, Not Wires: On-Chip Interconnection Networks," *Proc. of DAC 2001*, Las Vegas, Nevada, USA, pp. 683-689, June, 2001.

[13] W. J. Dally, C. L. Seitz, "The Torus Routing Chip," *Technical Report 5208:TR: 86, Computer Science Department, California Institute of Technology*, pp. 1-19, 1986.

[14] I. Saastamoinen, et al, "Interconnect IP Node for Future System on Chip Designs," *Proc. of the First IEEE International Workshop on Electronic Design, Test and Applications*, pp. 116-120, 2002.

[15] M. Forsell, "A scalable high-performance computing solution for networks on chips," *IEEE Micro, Volume 22 No 5*, pp. 46-55, 2002.

[16] F. Karim, A. Nguyen, S. Dey, "On-chip Communication Architecture for OC-768 Network Processors," *Proc. of DAC 2001*, Las Vegas, Nevada, USA, pp. 678-683, June, 2001.

[17] P. P. Pande, et al, "Design of a Switch for Network on Chip Applications," *Proc. of ISCAS*, Bangkok, pp. 217-220, May, 2003.

[18] P. P. Pande, et al, "High-Throughput Switch-Based Interconnect for Future SoCs," *Proc. of 3rd IEEE International Workshop on System-on-Chip for Real-Time Applications*, Calgary, Canada, pp.304-310, June-July, 2003.

[19] R. I. Greenberg, Lee Guan, "An Improved Analytical Model for Wormhole Routed Networks with Application to Butterfly Fat-Trees," *Proc. of International Conference on Parallel Processing*, pp. 44-48, 1997.

[20] W. J. Dally. "Virtual-Channel Flow Control," *IEEE Transactions on Parallel and Distributed Systems*, pp. 194-205, March 1992.

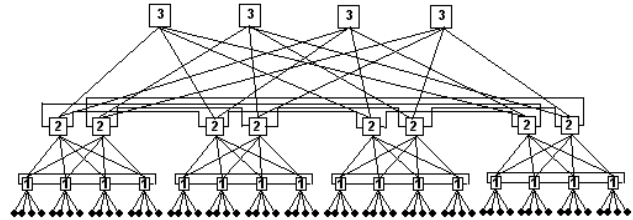


Fig. 1 Extended-butterfly Fat Tree Interconnection with 64 Ips

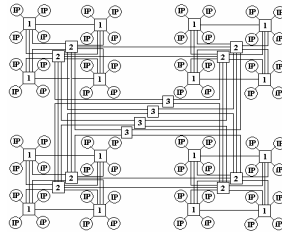


Fig. 2 EFTI floor plan

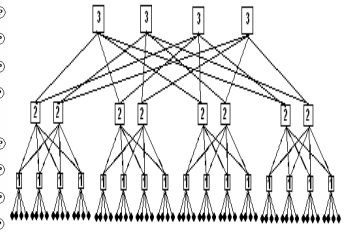


Fig. 3 BFT interconnection (64 IPs)

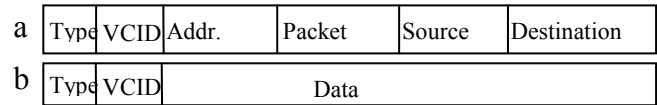


Fig. 4 flit format a) Header flit b) Data flit

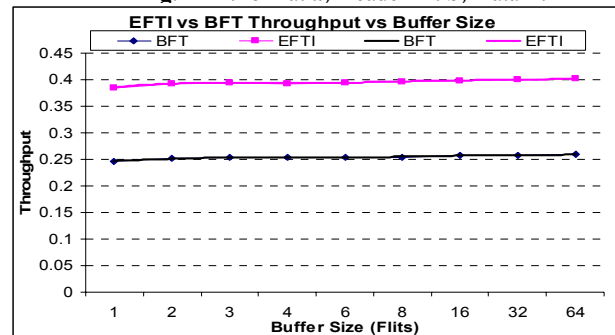


Fig. 5 Throughput EFTI vs BFT

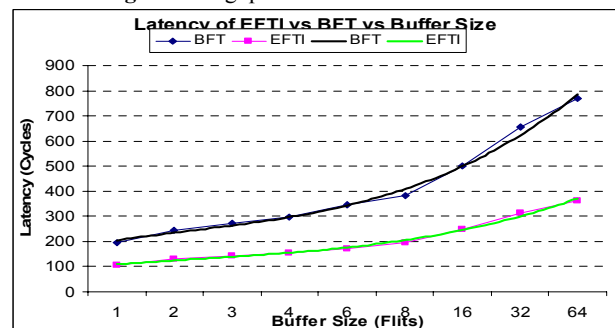


Fig. 6 Latency EFTI vs BFT