Design of a Switch for Network on Chip Applications

Partha Pratim Pande, Cristian Grecu, André Ivanov, Res Saleh
SOC Research Lab
Department of Electrical and Computer Engineering
University of British Columbia
2356 Main Mall Vancouver, BC, V6T 1Z4 Canada
Email: {parthap, grecuc, ivanov, res}@ece.ubc.ca

ABSTRACT

System on Chip (SOC) design in the forthcoming billion transistor era will involve the integration of numerous heterogeneous semiconductor intellectual property (IP) blocks. Some of the main problems in the ultra deep sub micron technologies characterized by gate lengths in the range of 50-100 nm arise from non-scalable global wire delays, failure to achieve global synchronization, errors due to signal integrity issues, and difficulties associated with non-scalable bus-based functional interconnect. These problems are addressed in this paper by introducing a new design methodology. A switch-based network-centric architecture to interconnect IP blocks is proposed. We introduce a butterfly fat tree architecture as an overall interconnect template. In this new interconnect architecture, switches are used to transfer data between IP blocks. To reduce overall latency and hardware overhead, wormhole routing is adopted. The proposed switch architecture supports this routing method. Initial implementation of the switch reveals that the total switch area is expected to amount to less than 2% of a large SOC.

Keywords- *System on chip, network on chip, switch, butterfly fat tree, wormhole routing, interconnect architecture.*

1. Introduction and Motivation

According to ITRS 2001 [1], 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 reality. Such SoCs imply the integration of numerous IPs performing different functions and operating at different clock frequencies. The integration of several heterogeneous components into a single system gives rise to new challenges.

One of the major problems associated with future SOC designs arises from non-scalable global wire delays. Global wires carry signals across a chip, but these wires typically do not scale in length with technology scaling [2]. Though gate delays scale down with technology, global wire delays typically increase or remain constant by inserting repeaters. Repeaters also have their inherent problems. They imply the use of an even number of inverters, require many via cuts, and, above all, repeaters consume silicon area and power. It is estimated that in 50 nm technology, with a clock frequency of 10 GHz, the global wire delay will be around 6-10 clock cycles [3]. Another important problem associated with global wires is that such wires are typically implemented in top level metal layers, with the routing performed automatically in latter stages of the design cycle. Such wires end up having parametric characteristics (parasitic capacitance and inductance) difficult to predict a priori and to maintain consistent.

In the forthcoming technologies, global synchronization of all IPs will lose its significance, due to difficulty/impossibility in sending signals from one end of the chip to another within a single clock cycle. Instead of aiming for global control, one attractive option is to allow self-synchronous IPs to communicate with one another through a network-centric architecture [1].

Still another significant problem in forthcoming SOC design arises from supply voltage reductions. Supply voltages at or below 1 volt will soon be common. This will result in noise susceptible logic. Moreover, long wires will act as lossy transmission lines. Hence, a main point of concern is that the transmission medium carrying digital signals will be inherently unreliable. Special error control mechanisms will be required for reliable system operation.

Existing on-chip interconnect architectures will give rise to other problems. The most frequently used on-chip interconnect architecture is the shared medium arbitrated bus, where all communication devices share the same transmission medium. In such architectures, only a single master can drive the bus at any time. Therefore, an arbitration mechanism is generally required to accommodate multiple masters. Such arbitration mechanisms, together with slow slave devices, degrade the bus performance. Finally, buses are not readily scalable, the overall throughput of the bus will not increase with the increase in number of IP blocks. For SOCs consisting of tens or hundreds of IP blocks, bus-based interconnect architectures will lead to serious bottleneck problems, as all attached devices share the bandwidth of the bus.

To overcome the above-mentioned problems, we propose the use of a network-centric approach to integrate IPs in complex SOCs. This new model allows decoupling of the processing nodes, i.e., the IPs, from the communication fabric. The need for global synchronization can thereby disappear. In our model, a group of IPs is connected to a neighboring switch and global signals, spanning significant portion of a die in current architectures now only have to circulate between switches. In this scenario, the top level interconnects will only consist of wires between switches. This information can be known at early stages of the design process, enabling a better prediction of the electrical parameters of the interconnect.

In our network-centric approach, the communication between IPs can take place in the form of packets. To overcome noise-induced errors due to unreliable physical channels, packet-based error control mechanisms can be adopted. We suggest that our on-chip network resemble the interconnect architecture of high-performance parallel computing systems, in which the IP blocks correspond to processing nodes communicating among each other using the on-chip network.

In this paper we propose a generic interconnect infrastructure where the IPs are connected according to the butterfly fat tree architecture [4]. A butterfly fat tree is a derivative of the fat tree architecture [5]. The fat tree architecture has found extensive use in different parallel machines and shown to be hardware efficient [5]. One advantage of the butterfly fat tree architecture springs from the fact that the number of switches in the network converges to a constant irrespective of the number of levels in the tree network.

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 interconnect architecture. In Sec. 4, a preferred routing algorithm is described. The packet structure is detailed in Sec. 5. The switch architecture is discussed in Sec. 6. Finally, Sec. 7 draws some conclusions and points to the future direction of this work.

2. Related Work

A few on-chip micro network proposals for SOC integration can be found in the literature. Sonic's Silicon BackPlane [6] is one example. This is a bus-based architecture in which the IPs are connected to the bus through specialized interfaces called *agents*. Each core communicates with an agent using the Open Core Protocol (OCP). Agents communicate with each other using TDMA (Time Division Multiple Access) bus access schemes. These agents effectively decouple the IP cores from the communication network. The basic data transfer mechanism between IP blocks is still based on arbitration and hence it is not scalable.

Kumar [7] and Dally [8] have proposed mesh-based interconnect architectures. These architectures consist of an *m x 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. Here the number of switches is equal to the number of IPs.

Saastamoinen [9] describes the design of a reusable switch to be used in future SOCs. The interconnect architecture is however not specifically discussed.

3. Interconnect Architecture

We propose a novel interconnect template to integrate numerous IPs of an SOC following a butterfly fat-tree architecture as shown in Fig. 1. In our network, the IPs are placed at the leaves and switches placed at the vertices. Fig. 2 illustrates a butterfly fat tree with *64* IPs. Each node is labeled 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, N)* denotes the locations of IPs at that lowest level. Each switch, denoted by *S (l, p)* has four child 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* IPs, the number of levels will be $log_4 N$. In the j^{th} level of the tree, there are $N/2^{j+1}$ switches. The number of switches in the butterfly fat tree architecture converges to a constant independent of the number of levels. If we consider a 4-ary tree as shown in Fig. 2 with four down links corresponding to child ports and two up links, corresponding to parent ports, then the total number of switches in level 1 is *N/4*.

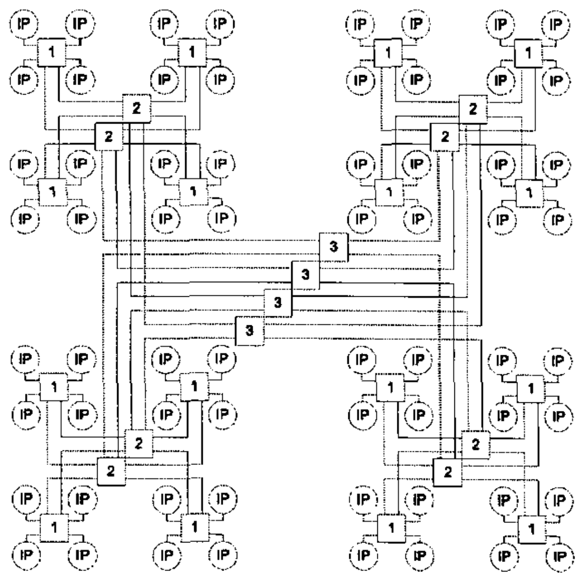

Fig. 1 Interconnection of 64 IP blocks using a butterfly fat-tree architecture.

At the next level the number of switches required reduces by a factor of 2, and so on. In this way the total number of switches, *S*, is given by:

$$S = \frac{N}{4} + \frac{1}{2}\frac{N}{4} + \frac{1}{4}\frac{N}{4} + \cdots \rightarrow \frac{N}{2}$$

which illustrates that in the limit, *S* goes to *N/2* as *N* grows arbitrarily large. In the case of 64 IPs the number of switches is 28 as shown in Fig. 1.

This switch-based architecture features different advantages. First, the wires between IP blocks and between switches are logically structured such that their lengths can be made largely predictable and largely consistent across the entire network. Secondly, the traffic emerging from or arriving at several IPs is combined and carried over a single wire, resulting in reduced wire congestion.

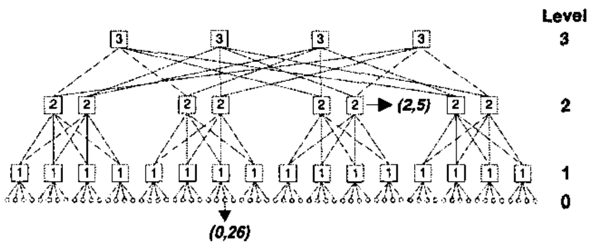

Fig. 2 Butterfly fat-tree graph with N= 64 IP blocks.

4. Routing Algorithm

We propose the adoption of a packet-based routing methodology for communicating between IPs. To reduce the communication latency and the switch memory requirements, we propose using *wormhole routing* [4]. In this method, packets, or worms, are comprised of fixed length flow control units *(flits)* The first flit, i.e., *header flit*, of a packet contains routing information. Header flit decoding enables the switches to establish the path and subsequent flits simply follow this path in a pipelined fashion. As a result, no packet reordering is required at destination. If a certain flit faces a busy channel, subsequent flits also have to wait at their current locations.

In the proposed butterfly fat-tree architecture there exists more than one shortest path between any pair of IPs [4]. A message can follow any one of the two up links from a switch. If the destination is not in the subtree rooted at the switch, the flit has to go up to the switch which is the least common ancestor of the source and destination, and from there, comes down in a unique path to reach its destination.

5. Packet Structure

The proposed packets consist of a header flit and one or more data flits. The header and data flit structures are is shown in Fig.3.

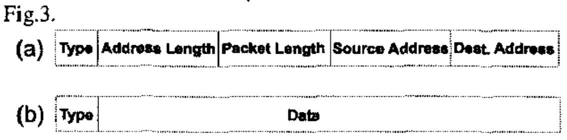

Fig. 3 a) Header flit; b) Data flit.

The first field denotes the flit type, namely header, data. The second field contains the length of address, which is dependent on the number of SOC IP blocks. The third field contains packet length information, i.e., number of flits in the corresponding packet. The next two fields give source and destination addresses. The flit length is constant but the total number of flits in a packet will vary according to the contents of the packet length field.

As an example, for an SOC comprised of up to 1024 IP blocks, ten bits are required for encoding the binary addresses of the IPs and four bits suffice for denoting the address field length. If k denotes the packet length then number of flits in the packet will be 2^k.

6. Switch Implementation

6.1 Internal Switch Architecture

The proposed interconnect is BI-DIrectional Multilevel INterconnect (BIDI-MIN) [10]. The switch should be designed accordingly to support this form of interconnect. The switch has six ports, four children ports denoted by ***child$_0$***, ***child$_1$***, ***child$_2$*** and ***child$_3$***, respectively, and two parent ports denoted by ***parent$_0$*** and ***parent$_1$***, respectively, as shown in Fig.4. In the first level of switches, four IPs are connected to the children ports and parents are connected to two switches at second level. The switches in level two are connected to four switches of level one in children ports and two switches of level three in parents ports. The switch is bi-directional in that each port is associated with a pair of opposite unidirectional channels, one for input and the other for output. This switch supports three types of connections: *forward*, *backward* and *turnaround*. The forward and backward connections support communications from children to parents and vice versa. The turnaround switching supports communication between children ports. Instead of providing two oppositely directed unidirectional channels in each port, one bi-directional channel could be used. However, this would amount to longer propagation delays.

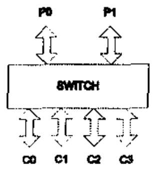

Fig. 4 A bi-directional switch.

Each port of the switch includes two FIFOs (one for each direction), a header decoder, and a controller. The switch also includes an arbiter and a crossbar. The FIFOs serve for buffering flits. The header decoder determines which output port is to be taken by a message subsequently to decoding the header flit. The arbiter receives requests from all the header decoders and grants output port access to a particular input. The controller is responsible for granting external physical channel. Through the crossbar, the path between input and output is established. Once a path from an input to an output channel (after decoding the header flit) has been established, the path remains established for all the subsequent flits of the corresponding packet.

The complexity of the switch depends on the routing algorithm and the FIFO design. FIFO design is critical as the latter may consume significant area and power. The size of the input FIFOs determines the buffering capability at the switches and, thereby, the overall latency. The controller architecture depends on the overall routing algorithm implemented.

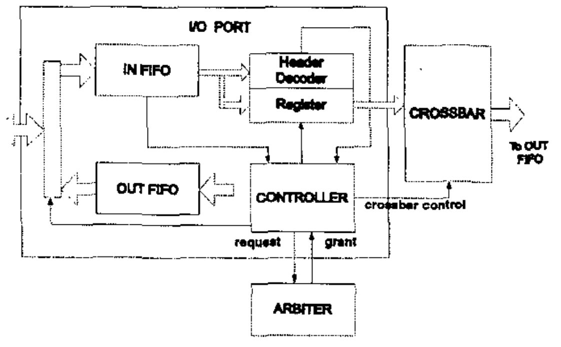

Fig. 5 Internal architecture of a switch port.

6.2 FIFO Buffer

The FIFO buffers are the most critical components of the switch. Their operating speed should be high enough to not become a bottleneck in a high-speed network. More specifically, the switches at level one need to be interfaced with the SOC's constituent IP blocks. Hence, the latter switches should be able to receive and transmit data at the rated speed of the corresponding IPs. Furthermore, the FIFOs should be able to operate with different read and write clocks as the SOC's constituent IPs are expected to generally operate at different frequencies. Conventional memory and counter-based FIFOs cannot achieve the required high speed and separate read and write clock requirements. The FIFO design proposed in [11] was adopted and modified to meet our requirements. Instead of using separate counters to implement read and write pointers, two tokens are circulated among the FIFO cells to implement read and write operations. A FIFO cell can be read from or written into only if it holds the corresponding token. Once a token is used while in a given cell, it is subsequently passed to the next cell. We have developed a VHDL model for this new FIFO and implemented it using a fully static, standard cell-based, CMOS 0.18μm technology. The number of gates required to implement a 6x7 FIFO buffer is 1500 equivalent two input NANDs. The size of the FIFO is determined by the length of each flit. In our example, this number is 42 bits, assuming 16 bits are used to denote packet length. A similar conventional counter-based FIFO requires fewer gates, around 900. Each switch requires 12 FIFOs, so the FIFOs consume most of the switch area and, from initial estimates, amount to an area of approximately 20,000 equivalent two input NAND gates. Hence, in a billion transistors SOC, the silicon area consumed by the switches is expected to be negligible, i.e., amounting to approximately 1-2%. The speed of operation predicted by static timing analysis tools for our design of this new type of FIFO is in excess of 1 GHz.

6.3 The Port Controller

The routing algorithm governs the controller design. Each port of the switch includes a controller. The controller must keep track of the input FIFO. Whenever the input FIFO stores one whole flit, it sends a *full* signal to the controller and the latter subsequently stores the flit into a register. If it's a header flit, then the header decoder determines the destination. The destination can be any of the child or parent ports. Once the destination has been determined, the controller checks whether the designated port is available to accept a new flit. Each switch includes a status register to indicate the availability of output FIFOs. If an output FIFO is empty, then the flit stored in the register is dumped into the output FIFO; otherwise, the flits wait in the register. The operation of the controller is illustrated through the FSM state diagram shown in Fig. 6.

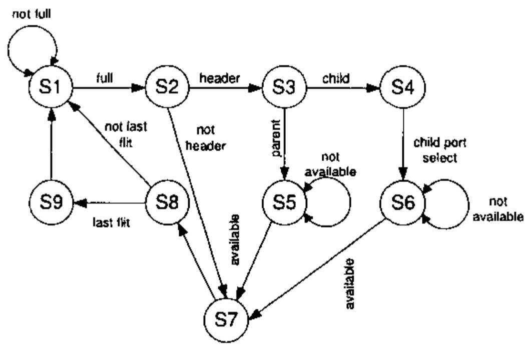

Fig. 6 State diagram of the port controller

A brief description of the states is as follows: **S1**: Input FIFO status check; **S2**: Storage of the flit into the register; **S3**: Determination of the destination address; **S4**: Child port selection; **S5**: Parent port availability check; **S6**: Child port availability check; **S7**: Sending of data to selected output port; **S8**: End of packet check; **S9**: Output status register update.

7. Conclusions and Future Work

We conjecture the future need and practicality of a paradigm shift in SOC design methodology from a conventional one to a network-centric approach. We illustrated in this paper how a network-centric approach can overcome the problems due to non-scalable global wire delay, difficulty in global synchronization, noise induction and bus-based interconnects in ultra deep sub micron technology.

This paper also presents a novel switch architecture, to support our proposed methodology. We propose a butterfly fat tree architecture as an overall interconnect template to integrate numerous heterogeneous IP blocks. The switch serves to integrate IP blocks in this template. In this on-chip micro-network environment, wormhole routing is the most appropriate. Our proposed switch architecture supports this type of routing. This leads to low hardware overhead and latency.

To achieve a practically realizable on-chip micro network, the switches need to be interfaced with different types of IP blocks in level one. To accommodate this requirement, we are in the process of designing a family of interfaces corresponding to different types of IP blocks, like general-purpose processors, DSPs, memories, UARTs, etc. These interfaces will decouple the processing blocks from the communication network.

The achievable throughput of our micronetwork is under study and efforts toward silicon implementation are underway.

8. Acknowledgements

The authors wish to thank Micronet, PMC-Sierra, Gennum and NSERC for their financial support and the CMC for providing access to CAD tools. The authors also thank Profs' M.Greenstreet, A. Hu and H.Alnuweiri for their suggestions.

9. References

[1] *ITRS 2001 Document http://public.itrs.net/ITRS/Files/2001ITRS/Home.htm*
[2] Horowitz, M.A. et al., "The Future of Wires" Proceedings of the IEEE, Volume: 89 Issue: 4, April 2001 pp.: 490 -504
[3] Benini, L., De Micheli, G., "Networks on chips: a new SOC paradigm" Computer, Volume: 35 Issue: 1, Jan. 2002, pp.: 70 –78
[4] Greenberg, R.I.; Lee Guan "An improved analytical model for wormhole routed networks with application to butterfly fat-trees ", Proceedings of the 1997 International Conference on Parallel Processing, pp.: 44 -48
[5] Leiserson, C.E. "Fat Trees: Universal networks for hardware efficient supercomputing". IEEE Transactions on Computers, C-34 (10): pp. 892-901. Oct. 1985.
[6] Wingard, D., " MicroNetwork-Based Integration for SOCs" DAC 2001, June 18-22, 2001, Las Vegas, Nevada, USA.
[7] Kumar, S. et al, " A Network on Chip Architecture and Design Methodology." Proceedings of ISVLSI 2002, pp. 117-124
[8] Dally, W.J. and Towles, B., "Route Packets, Not Wires: On-Chip Interconnection Networks", DAC 2001, June 18-22, 2001, Las Vegas, Nevada, USA.
[9] Saastamoinen, I. et al, "Interconnect IP node for Future System-on-Chip Designs", Proceedings of the First IEEE International Workshop on Electronic Design, Test and Applications, 2002, pp. 116 –120.
[10] Ni, L, Gui, Y., Moore, S., "Performance Evaluation of Switch-Based Wormhole Networks", IEEE Transactions on Parallel and Distributed Systems, Volume: 8 Issue: 5, May 1997, pp. 462-474
[11] Chelcea, T., Nowick, S.M., "A low-latency FIFO for mixed-clock systems", Proceedings of IEEE Computer Society Workshop on VLSI, 2000, pp. 119 –126