ELSEVIER

Microelectronics Journal 36 (2005) 833–845

Microelectronics Journal

www.elsevier.com/locate/mejo

Timing analysis of network on chip architectures for MP-SoC platforms

Cristian Grecu, Partha Pratim Pande*, André Ivanov, Res Saleh

Department of Electrical and Computer Engineering, SoC Research Lab, University of British Columbia, 2356 Main Mall, Vancouver, BC, Canada V6T1Z4

Received 1 August 2004; received in revised form 18 February 2005; accepted 20 March 2005

Abstract

Recently, the use of multiprocessor system-on-chip (MP-SoC) platforms has emerged as an important integrated circuit design trend for high-performance computing applications. As the number of reusable intellectual property (IP) blocks on such platforms continues to increase, many have argued that monolithic bus-based interconnect architectures will not be able to support the clock cycle requirements of these leading-edge SoCs. While hierarchical system integration using multiple smaller buses connected through repeaters or bridges offer possible solutions, such approaches tend to be ad hoc in nature, and therefore, lack generality and scalability. Instead, many different forms of network on chip (NoC) architectures have been proposed in the past few years to specifically address this problem. We believe that the NoC approach will ultimately be the preferred communication fabric for next generation designs. To support this conjecture, this paper demonstrates, through detailed circuit design and timing analysis that different proposed NoC architectures to date are guaranteed to achieve the minimum possible clock cycle times in a given CMOS technology, usually specified in normalized units as 10–15 FO4 delays. This is contrasted with the bus-based approach, which may require several design iterations to deliver the same performance when the number of IP blocks connected to the bus exceeds certain limits.

Keywords: MP-SoC; NoC; Pipelining; Bus; Scalability

1. Introduction and motivation

According to recent publications, e.g. [1–3], the emergence of system-on-chip (SoC) platforms consisting of large, heterogeneous sets of embedded processors is imminent. As described in [2], a common design paradigm emerging for SoC integration involves arrays of intellectual property (IP) blocks, such as embedded processors consisting of around 100 K gates, integrated according to a specific interconnect template. Collectively, these platforms are referred to as MP-SoC platforms [1], meaning that multiple processors are integrated on the same SoC design. While this is conceptually elegant, it implies a host of new issues, in addition to the already existing issues of SoC design. For example, embedded software and real-time operating system design, hardware/software co-design, system verification, configurability and programmability of such platforms must all be addressed in the context of MP-SoC.

Another key element of such platforms is the nature of the communication fabric. A common problem shared by all interconnect topologies implemented in ultra deep submicron (UDSM) technology is the communication latency that arises from global wire delays. Even with repeater insertion, the intra-chip propagation delay of long wires can exceed the limit of one clock cycle [4,5] if the die size is large enough. To overcome this problem, many large designs today use first-in first-out (FIFO) buffers to synchronously propagate data over long distances across the chip. The FIFO insertion process is ad hoc in nature, and therefore, not easily generalizable and scaleable. Wires need to be divided into segments that have a propagation time compatible with the clock cycle budget, and signals need to propagate along these segments in a pipelined fashion.

To date, the most frequently used on-chip interconnect architecture is the shared-medium arbitrated bus. Advantages of shared-bus architectures include topological simplicity, low silicon area requirements, and extensibility. Some of the associated disadvantages as the number of IP blocks increases include relatively large load per data-bus

* Corresponding author.
E-mail addresses: grecuc@ece.ubc.ca (C. Grecu), parthap@ece.ubc.ca (P.P. Pande), ivanov@ece.ubc.ca (A. Ivanov), res@ece.ubc.ca (R. Saleh).

0026-2692/$ - see front matter © 2005 Elsevier Ltd. All rights reserved.
doi:10.1016/j.mejo.2005.03.006

line, long delay for data transfer, high energy consumption, increasing complexity of the decoding/arbitration and low bandwidth [6]. Every additional IP block connected to the bus adds to the parasitic capacitance, in turn, causing increased propagation delay. For a relatively long bus line, the intrinsic parasitic resistance and capacitance can reach excessive values [4,5]. As the bus length increases and/or the number of IP blocks increases, the bus bit transfer delay may exceed the specified clock cycle.

One solution for such cases is to introduce a hierarchical architecture and split the bus into multiple shorter segments. This approach also reduces the bottleneck caused by relatively slower IP blocks, such as typical I/O devices. Each shorter bus section can be viewed as effectively constituting a pipeline stage with a propagation delay that must fit within the clock cycle budget. However, the delay of each stage in the case of a hierarchical bus-based system depends on the parasitic capacitance due to the IP blocks connected to the individual bus segments. Therefore, the above methodology only yields a solution, where the timing specifications and the interconnect design are tightly coupled, and the solution will depend on the specific IP blocks and their functional and parametric specifications. The ad hoc nature of the latter solution presents several disadvantages, particularly with regard to scalability and design automation.

Based on the premise that a structured solution is an effective way to manage complexity, the use of a network on chip (NoC) has been proposed by many researchers [1,2,15,19]. The NoC-based approach to the above-mentioned interconnect problem provides a highly-structured, multi-core SoC design methodology that can achieve aggressive clock/data rates without incurring undue design effort. The key is to use an on-chip interconnect topology resembling the interconnect architecture of high-performance parallel computing systems [3]. The common characteristic of these kinds of architectures is that the functional IP blocks communicate with each other through intelligent switches. These switches should be designed to be reusable and their primary role should be to ease the integration process of the functional IPs. As such, the switches can be considered as infrastructure IPs (I^2Ps) [7] providing a robust data transfer medium for the functional IP modules. Global signals, spanning a significant portion of a die in more traditional design styles, now only have to span the distance separating infrastructure switches. This switch-based architecture offers the advantage that the switches, along with the inter-switch wire segments, can now form the basis for a highly pipelined communication medium.

One of the challenges arising out of NoC design is to constrain the delay of the pipeline units to a limit of one clock cycle. The definition of a clock cycle is, of course, based on the required performance of a digital circuit, but this value changes from design to design, and from technology to technology. One way to normalize the clock cycle is to represent it in terms of a technology-independent

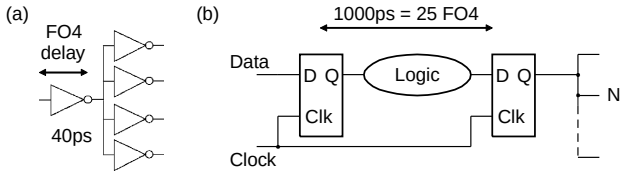

Fig. 1. Illustration of the FO4 metric.

value that still provides information about the speed of the design. The fanout-of-four (FO4) delay metric serves exactly this purpose. It is defined as the delay of one inverter driving four identical ones as shown in Fig. 1(a). For specific values of the supply and threshold voltages of the transistors, the FO4 delay is fixed for a given technology. For example, for a 90 nm technology with $V_{DD}=1$ V and $V_T=0.3$ V, the FO4 delay is approximately 40 ps, as is shown in Fig. 1(a). If the cycle time through the combinational logic between two flip–flops (including flip–flop overhead) is given as 1000 ps in a 90 nm technology, it would be represented by 25 FO4 delay units, as illustrated in Fig. 1(b). A design with a higher clock frequency would, of course, have fewer than 25 FO4 delays between clock edges.

Clock frequencies of high-performance microprocessors have improved by nearly 40% annually over the last decade. This increase in clock frequency has come from technology scaling and deeper pipelining of designs. The clock period scaling trend is summarized in Fig. 2 following data published by ITRS [8]. Around 1985, designs had roughly 100 FO4 delays per cycle. Since that time, the number has been steadily declining in value, primarily due to pipelining the computation, and has recently exhibited a tendency towards saturation. In accordance with Fig. 2 and ITRS [8], a, generally, accepted rule of thumb is that the clock cycle of high performance SoCs will saturate at a value between 10 and 15 FO4 delay units. This is a reasonable target saturation level since flip–flops incur overhead in the range of 4–5 FO4 units, leaving only about 10 FO4 units to carry out all necessary logic functions. It is difficult to imagine clock cycles significantly below this level.

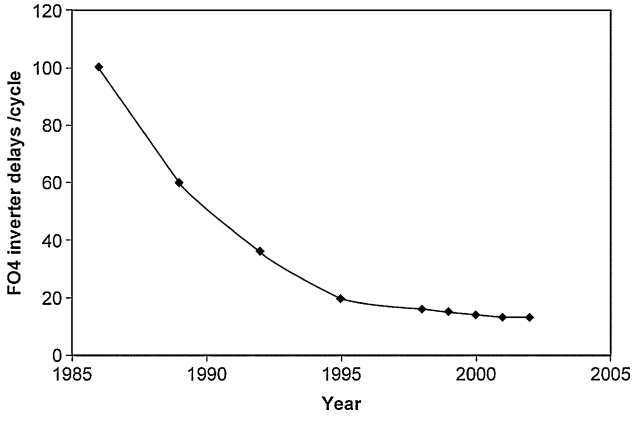

Fig. 2. Clock period trend [8].

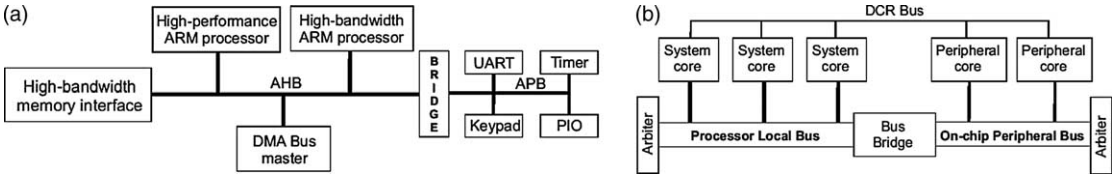

Fig. 3. Bus architectures for (a) AMBA and (b) CoreConnect.

Nevertheless, we will show that even this clock cycle limit is achievable by proper design of the switch elements and inter-switch wire segments of the NoC architecture. This makes communication pipelining with a pre-specified clock rate achievable regardless of the system size. On the other hand, achieving this in a bus-based approach is difficult. Hence, with a network on chip (NoC) architecture, the SoC inter-block communication fabric can be designed and optimized independently from the specific constituent blocks (i.e. processing elements).

2. Related work

SoC designs today predominantly use a shared-medium, bus-based functional interconnects to integrate IP blocks. The reason for this is that buses are well-understood, are suitable for small SoC designs and have standardized interfaces. Several industrial buses in use today have similar characteristics. They consist of a high-performance system backbone bus, able to sustain the bandwidth of the CPU and other direct memory access devices, plus a bridge to a lower speed bus on which lower bandwidth peripherals are located. Three popular commercial bus configurations include ARM AMBA [9] bus, Wishbone [10] and IBM CoreConnect [11]. In the case of the Advanced Microcontroller Bus Architecture (AMBA) [9], the high-speed buses are the Advanced High-performance Bus (AHB) and Advanced System Bus (ASB) and the lower speed bus is the Advanced Peripheral Bus (APB). A typical AMBA architecture is shown in Fig. 3(a).

Similarly, the IBM CoreConnect architecture provides three different types of buses, namely Processor Local Bus (PLB) for high-speed devices, On chip Peripheral Bus (OPB) for peripherals, and Device Control Register (DCR) Bus for status and configuration registers. Fig. 3(b) shows the typical CoreConnect architecture. According to several experts, including those found in [3,6], these bus-based systems may suffer from drawbacks of limited scalability as per the preceding discussion and arguments.

A few on-chip micro network proposals for SoC integration can be found in the literature. Sonic's Silicon Backplane [12] is one example. In this architecture, the IP blocks are connected to the communication medium through specialized interfaces called *agents*. Each core communicates with an agent using the Open Core Protocol (OCP) [13]. MIPS Technologies has introduced an on-chip switch integrating IP blocks in a SoC [14]. The switch called *SoC*—it is intended to provide a high-performance link between a MIPS processor and multiple third party IP cores.

The standard bus architecture works well when the number of IP blocks to be integrated is relatively small. To support higher degrees of integration, the single monolithic bus-based architectures evolved to hierarchical structures, where multiple smaller buses are integrated. This is illustrated in Fig. 4. Conceptually, the approach is simple but the design of such structures also tends to be ad hoc in nature. In the short-term, this will continue to be a viable solution. However, as the number of IP blocks grows, it too will reach its limitations.

True NoC architectures began to appear in the literature in 2001. Instead of building a multi-processor (MP) SoC around an ad hoc interconnection of multiple buses, different parallel computing architectures were mapped to the SoC domain. Consequently, most of the NoC architectures proposed by researchers in this domain evolved from parallel processing structures, e.g. [15,16,18,19]. Kumar [15] has proposed a mesh-based interconnect architecture named CLICHÉ. Dally et al. [16] suggested a Torus-based architecture. Both of 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. Saastamoinen [17] describes the design of a reusable switch to be used in future SoCs. The interconnect architecture is however not specifically discussed. Guerrier and Greiner [18] proposed the use of a tree-based interconnect (SPIN) and addressed system level design issues. In [19] and [20], the authors describe an interconnect architecture based on Butterfly Fat-Tree (BFT) topology for a networked SoC, as well as the associated design of the required switches and addressing mechanisms. This paper quantitatively discusses the effect of the system size on

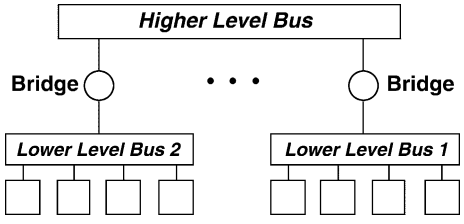

Fig. 4. Hierarchical buses.

the achievable clock cycle duration in a multi-core SoC environment.

3. Achievable clock cycle in a bus segment

We now return to the bus-based SoC and develop a simplified model to analyze how system size affects the achievable clock cycle. In this situation, multiple IP blocks share the same transmission media. As the number of connected IP blocks increases, the capacitance attached to the bus wires increases correspondingly. For ease of analysis (but without loss of generality), we assume this extra capacitance to be evenly distributed along the wire and model it as a parasitic capacitance.

This negatively impacts propagation delay and, ultimately, the achievable clock cycle. Viewed another way, this limits the number of IP blocks that can be connected to the bus and, thereby, the system scalability.

As many existing on-chip buses are multiplexer-based [9–11], they are basically unidirectional, and can therefore, easily be buffered. Attaching IP blocks to a bus adds an equivalent parasitic capacitance of C_{IP} per unit length of wire to an existing capacitance of C_w per unit length due to the wire itself. As a result, the driving capability of the bus is negatively affected, and buffer insertion is required to accommodate a number of IPs beyond a certain threshold while still satisfying a propagation delay of one clock cycle. If a bus wire is divided into N_{bus} segments, then each wire segment will have a capacitance of (C_w+C_{IP}) per unit length. The configuration of the wire with inserted buffers (repeaters) is shown in Fig. 5.

We now derive the delay equation for this situation. First, without any parasitic wire and IP capacitance and resistance, the total delay D_{inv} would simply be

$$D_{inv} = N_{bus}t_{inv} = N_{bus}(C_G + C_J)R_{eqn} \tag{3.1}$$

where C_G+C_J is the total gate and junction capacitance of the inverter, respectively, and R_{eqn} is the equivalent driving resistance of the inverter. Second, if we consider the wire and parasitic capacitance by themselves, we would represent it as a lumped π-model and derive the delay $D_{parasitics}$ as

$$D_{parasitics} = 0.4R_w(C_w + C_{IP})\frac{L_{bus}^2}{N_{bus}} \tag{3.2}$$

where R_w is the wire resistance per unit length and L_{bus} is the total length of the bus. Finally, when the inverter and parasitics are combined in each stage, we obtain a delay using Elmore [22] that is the sum of the above equations, plus and additional term representing the interactions between buffer and wire.

Therefore, the delay in the buffered bus wire $D_{buffered}$ can be computed according to the following equation

$$D_{buffered} = N_{bus}t_{inv} + \left(C_G R_w m + \frac{(C_w + C_{IP})R_{eqn}}{m}\right)L_{bus} + 0.4R_w(C_w + C_{IP})\frac{L_{bus}^2}{N_{bus}} \tag{3.3}$$

where t_{inv} is the delay of an inverter sized for equal rise and fall propagation delays, m is the size of the inverters, C_G is the gate capacitance of the minimum size inverter, R_{eqn} is the resistance of n-type diffusion region in Ω/γ, R_w and C_w are the resistance and capacitance per unit length of the wire, respectively, and L_{bus} is the bus length. The values of m and N_{bus} that minimize $D_{buffered}$ have been used in subsequent calculations. Consequently, the achievable clock cycle will be $1/D_{buffered}$. This approach of delay modelling optimizes the delay along a bus segment assuming uniformly distributed load. However, if the IP blocks load the bus wires non-uniformly then the buffers can have different sizes and spacings, depending on the specific parasitics distribution.

Using this equation, it is possible to study the variation of the clock cycle as a function of the value of C_{IP} and establish limits on the performance of the bus-based approach. The particular case of 130 nm technology node for a fixed bus length is shown in Fig. 6. From the figure, it is evident that as C_{IP} increases beyond a certain value, the clock cycle exceeds the limit of, say, 15 FO4 delay units. In this respect, the threshold value of C_{IP} can be considered as a metric for

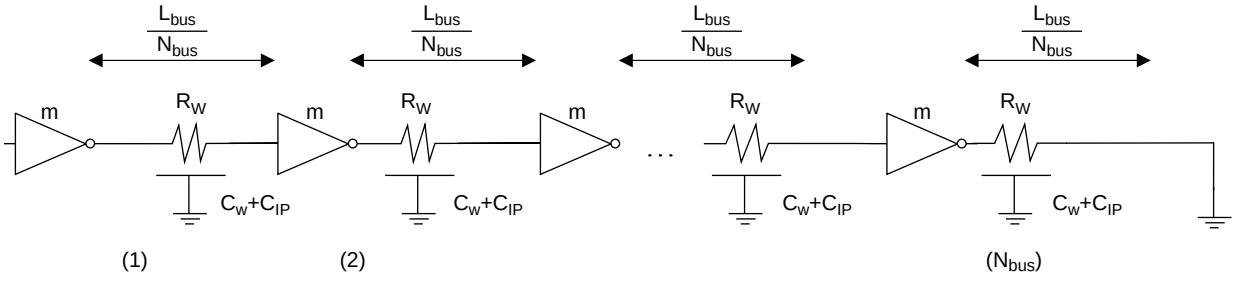

Fig. 5. A buffered wire segment.

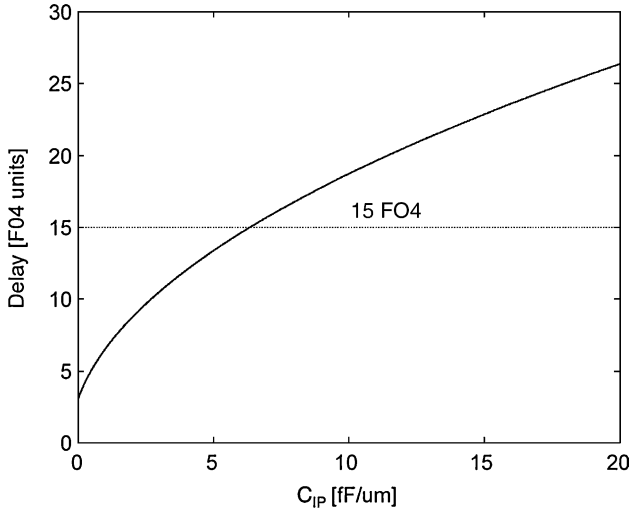

Fig. 6. Variation of delay with parasitic capacitance C_{IP} for a fixed bus length.

the scalability of a bus-based system as it relates to how many IP blocks can be attached to a bus before the delay exceeds one clock cycle. However, due to heterogeneous nature of constituent IP cores in a SoC (DSPs, MPEG decoders, memories etc.), it is not possible to quantify, a priori, the number of IPs that can be connected to a bus segment. Rather, by knowing C_{IP} and the types of IPs that need to be integrated for a particular application, we will be able to determine whether the clock cycle is achievable.

One way to deal with this scalability problem is to split the bus into smaller segments. A bus of shorter length can support more parasitic capacitance, C_{IP}, arising from attached IP blocks (assuming that the bus can physically accommodate them). Fig. 7 shows the delay along a bus segment as a function of L_{bus} for a 130 nm technology node, but here we assume a fixed value of C_{IP}. Again, beyond a certain limit, a 15 FO4 cycle time cannot be achieved.

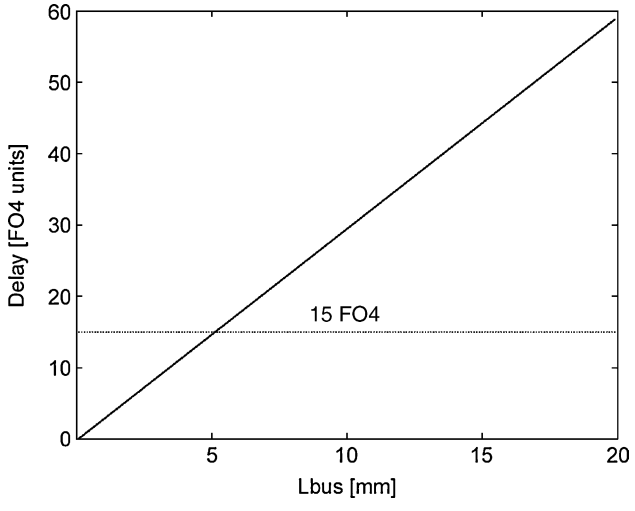

Fig. 7. Variation of delay with bus length for a fixed C_{IP}.

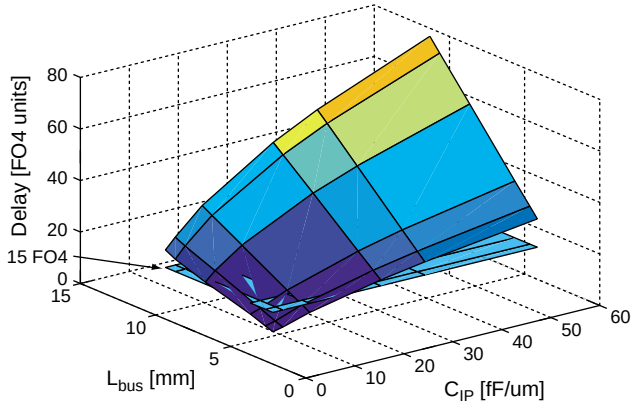

Fig. 8. Variation of delay with C_{IP} and bus wire length.

In reality, the two graphs of Figs. 6 and 7 should be combined to provide a three-dimensional view of the delay characteristics. In Fig. 8, we show, for illustrative purposes, the delay characteristics as a function of both L_{bus} and C_{IP} for the same technology node (130 nm). From the figure, the trend is that more parasitic capacitance C_{IP} can be supported by reducing the length of the bus. On the other hand, for any bus length, C_{IP} will be constrained by the target 15F04 limit. As an example, when the bus length is 6 mm, then the allowable parasitic capacitance is 20 fF/μm; when the bus length is increased to 10 mm then the parasitic capacitance reduces to 8 fF/μm to keep the delay along the wire segment within the specified limit of 15 FO4.

Based on this analysis, a single conventional shared bus-based system will not easily be able to achieve the clock cycle predicted in ITRS in high performance MP-SoCs. The relatively long bus needs to be split into smaller components. The lengths of each section of the shorter buses can be designed such that they are able to individually support a clock cycle of 15 FO4. The multiple, relatively shorter, buses can be integrated using repeaters or bridges.

For larger SoCs, the use of multiple hierarchical bus-based systems is recommended and, ultimately, it can be viewed as a form of network of chip. However, unless they are constrained to have certain topological characteristics a priori, such bus-based networks will vary widely (depending on the specific SoC), and will therefore, possess ad hoc characteristics. For more robust and scalable solutions to this problem, we propose that a design methodology for a structured network be imposed.

4. NoC-based structured interconnect architectures

Over the past few years, a number of NoC architectures have been proposed by different research groups. They can be classified into two basic groups: derivatives of generic *fat-trees* and the *k-ary n-cubes*. The SPIN and BFT belong to the first group while CLICHÉ, Torus and the Folded Torus fall into the second category. In the following

subsections, we provide a brief review of these architectures. In the illustrative examples, the functional IP blocks are denoted by white squares, while the infrastructure IPs (switches) are denoted by dark squares.

4.1. Spin

Guerrier and Greiner [18] have proposed a generic interconnect template called Scalable, Programmable, Integrated Network (SPIN) for on-chip packet switched interconnections, where they have used a generic fat-tree architecture to interconnect IP blocks. In this fat-tree, every node has four sons and the parent is replicated four times at any level of the tree. Fig. 9 shows the basic SPIN architecture with 16 functional IP blocks and Fig. 10 shows the corresponding conceptual block diagram. The size of the network grows as $(N \log N)/8$, where N denotes the number of functional IP blocks integrated [18]. The functional IP blocks reside at the leaves and the I^2Ps (switches) reside at the vertices.

4.2. BFT

We recently proposed another type of network architecture for SoCs-based on the Butterfly Fat-Tree (BFT) [19], as shown in Fig. 11. In our network, the IPs are placed at the leaves and switches placed at the vertices. A pair of coordinates labels each node, (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 jth level of the tree, there are $N/2^{j+1}$ switches. For a very large system size, the number of switches in the butterfly fat-tree architecture converges to a constant independent of the number of levels [19]. A conceptual block diagram of the BFT architecture with 16 functional IP blocks is shown in Fig. 12.

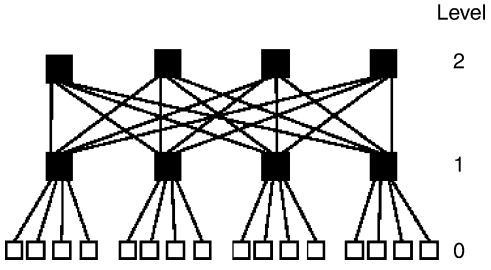

Fig. 9. SPIN architecture.

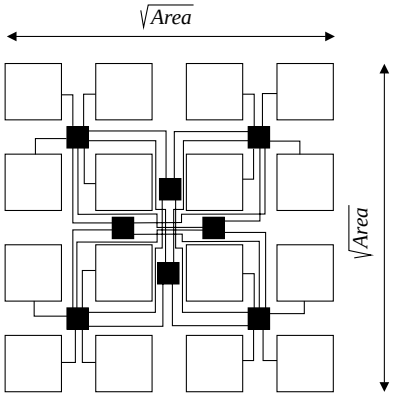

Fig. 10. Block diagram of SPIN.

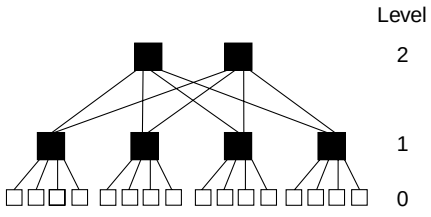

Fig. 11. BFT architecture.

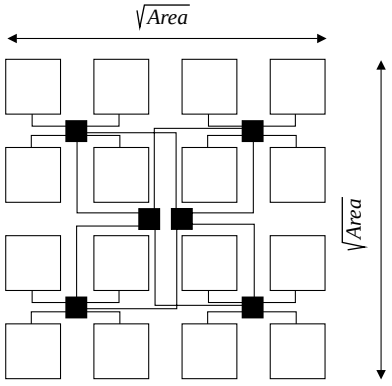

Fig. 12. BFT block diagram.

4.3. CLICHÉ

Kumar et al. [15] have proposed a mesh-based interconnect architecture called Chip-Level Integration of Communicating Heterogeneous Elements (CLICHÉ). This architecture consists of an $m \times n$ mesh of switches interconnecting computational resources (IPs) placed along the X- and Y-axes. Each switch, except those at the edges, is thereby connected to four neighboring switches and one functional IP block. In this case, the number of switches is equal to the number of functional IPs. The resources and the switches are connected through communication channels. A channel consists of two unidirectional links between the switches or between a switch and a functional IP (Fig. 13).

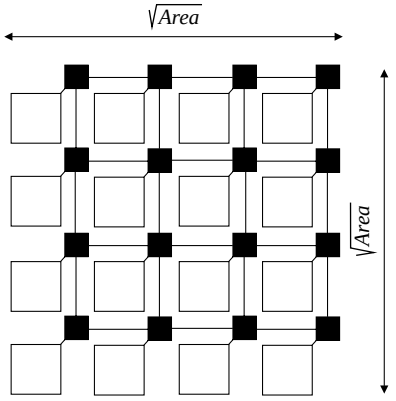

Fig. 13. CLICHÉ architecture.

4.4. Torus

Dally et al. [16] have proposed a 2D Torus as an NoC architecture. The Torus architecture is basically the same as the mesh; the only difference being that the switches at the edges are connected to the switches at the opposite edge through a wrap-around channel [23]. Each switch has five ports, one connected to the local functional IP and the others connected to the closest neighboring switches. The long end-around connections can yield excessive delays (Fig. 14). However, this can be avoided by folding the Torus as shown in Fig. 15 [26].

5. Switch architecture

Before carrying out the timing analysis of NoC architectures, it is important to describe the details of the switches that are used in the communication path between IP blocks. We do not discuss the functional IP blocks themselves, since they are dependent on the specific application. However, for the purposes of this paper, they may be considered as a set of embedded processors.

Switches are an integral part of all the NoC topologies and we assume that all of them use essentially the same switching circuitry to provide for a uniform comparison.

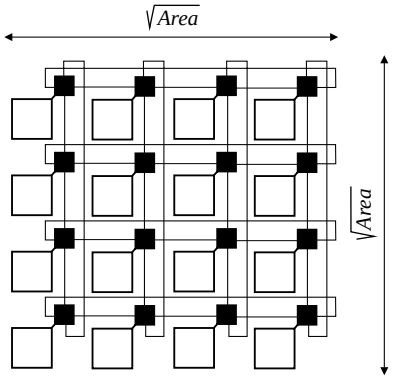

Fig. 14. Torus.

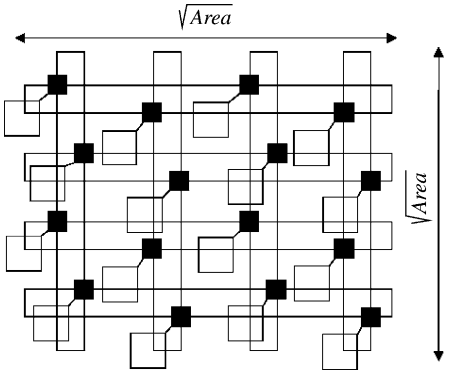

Fig. 15. Folded Torus.

The overall design of the switches depends on the adopted data transfer (switching) methodology. *Wormhole switching*, where the packets are divided into fixed length flow control units (*flits*) is assumed here. 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, each incoming data flit of a message packet is simply forwarded along the same output channel, as the preceding data flit and no packet reordering is required at destinations. If a certain flit faces a busy channel, subsequent flits also have to wait at their current locations.

One drawback of this simple wormhole switching method is that the transmission of distinct messages cannot be interleaved or multiplexed over a physical channel. Messages must cross the channel in their entirety before the channel can be used by another message. This will decrease channel utilization if a flit from a given packet is blocked in a buffer. By introducing virtual channels [23] in the input and output ports we can increase channel utility considerably. If a flit belonging to a particular packet is blocked in one of the virtual channels, then flits of alternate packets can use the other virtual channel buffers, and hence, ultimately, the physical channel. The canonical architecture of a switch having virtual channels is shown in Fig. 16.

The different components of the switch port are shown in Fig. 17. It mainly consists of input/output FIFO buffers, input/output arbiters, multiplexer (MUX) and demultiplexer (DEMUX) units, and a routing block. Each physical input port has more than one virtual channel, uniquely identified by its virtual channel identifier. Flits may simultaneously arrive at more than one virtual channel. As a result, an arbitration mechanism is necessary to allow only one virtual

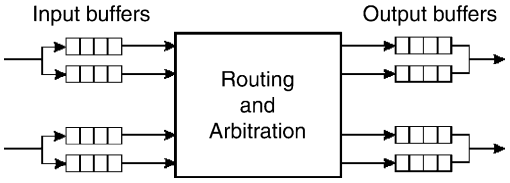

Fig. 16. Virtual-channel switch.

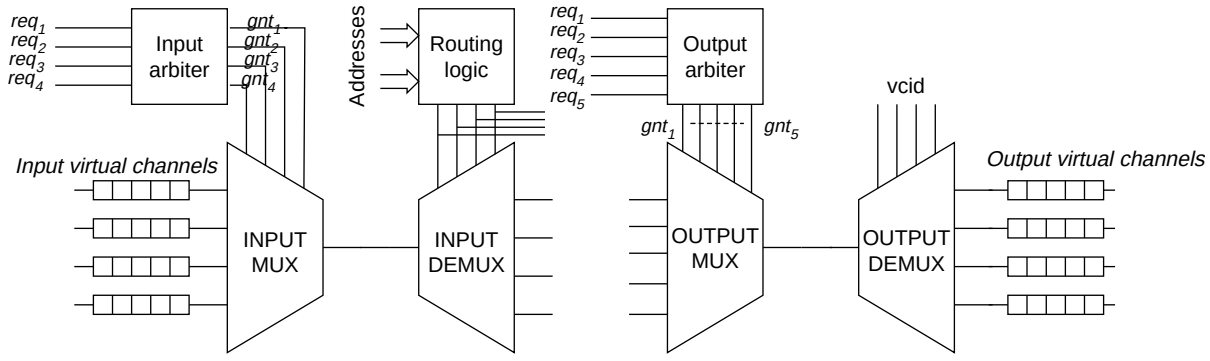

Fig. 17. Block diagram of a switch port.

channel to access a single physical port. Similarly, flits from more than one input port may simultaneously try to access a particular output port. Hence, another arbiter is required at each output port. The routing logic block determines the output port to be taken by an incoming flit, i.e. the next node to be followed in the path.

The operation of the switch consists of one or more processes depending on the nature of the flit. In the case of a *header* flit, the sequence of the processes is: (1) *input arbitration*; (2) *routing*; and (3) *output arbitration*. In the case of *body* flits, *switch traversal* replaces the routing process as the routing decision based on the header information is maintained for the subsequent body flits. This sequence of processes is conceptually shown in Fig. 18 for the header, data, and tail flits. Clearly, this processing sequence can be executed in a pipelined fashion, which is desirable in a NoC context.

6. Communication pipelining

A common characteristic of all the NoC architectures is that the whole communication medium can be divided into accurately predictable stages. The structured inter-switch wire segments together with the switches establish a highly pipelined communication infrastructure as shown in Fig. 19.

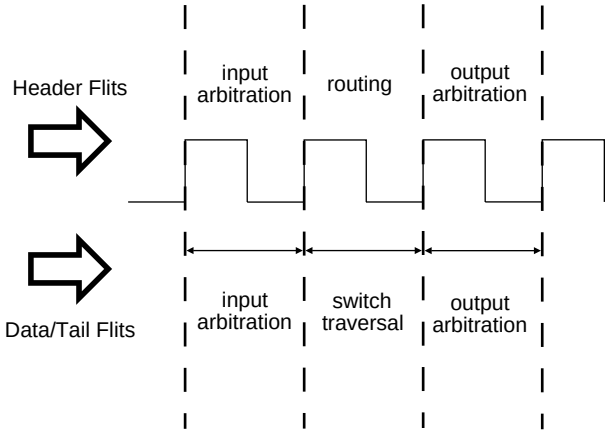

Fig. 18. Intra-switch pipeline stages.

In terms of interconnect wire delay, the NoC architectures offer the advantage that the wires between IP blocks and between switches are logically structured. Therefore, their lengths and delays are largely predictable and consistent across the entire network. This is a compelling feature of any structured approach to design.

The switches (I^2Ps) required in each of the network architecture solutions mentioned previously consist of multiple stages. We can differentiate between two types of delays, namely inter- and intra-switch. Through detailed circuit level design and analysis, we can constrain the delay of each pipelined stage to be within the ITRS suggested limit of 15 FO4 delay units. By doing this, and ensuring that wire delays also comply with this target, we can guarantee that the communication network will operate at any reasonable clock frequency.

6.1. Wire delay between switches

Using the earlier example layouts shown for the various NoC architectures, we will now determine the longest inter-switch wire that arises for each of them under the specific placement constraints. After determining the longest inter-switch wire segments that arises in each of the topologies, we determine the delay in sending data across each such wire segment. If the delay along the longest inter-switch wire segment can be constrained to be within the assumed clock cycle limit of 15 FO4, then the delay along all the other shorter inter-switch wire segments will follow suit. The inter-switch wire lengths depend on the specific topology adopted for the SoC infrastructure and the system size. The number of IPs interconnected in a single SoC will vary from one technology node to another, and also depend on the specific architecture. Letting A_{IP} denote the area of

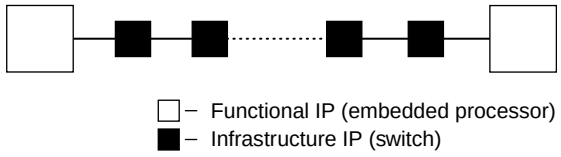

Fig. 19. Pipelined data transfer.

Table 1
Maximum number of 100 K IP blocks in different technology nodes

Technology node	130 nm	90 nm	65 nm	45 nm	32 nm
Max. no. of 100 K gates IP blocks	500	1000	2500	7500	10,000

Table 2
Distribution of functional and infrastructure IP blocks

Technology node (nm)	No. of functional IPs			No. of I^2Ps		
	BFT	SPIN	Mesh/Torus/ Octagon	BFT	SPIN	Mesh/Torus/ Folded Torus
130	428	400	375	214	300	375
90	856	800	750	428	600	750
65	2142	2000	1875	1071	1500	1875
45	6426	6000	5625	3213	4500	5625
32	8568	8000	7500	4284	6000	7500

the functional IP blocks and A_{I^2P} denote the area of the switches, then

$$\text{Area}_{\text{SoC}} = N_1 A_{\text{IP}} + N_2 A_{I^2P} \tag{6.1}$$

where N_1 and N_2 are the number of functional and infrastructure IPs, respectively, and Area_{SoC} is the total area of the SoC under consideration. For a BFT, $N_1 = 2N_2$; for SPIN architecture $N_1 = (4/3)N_2$; and for all the others under consideration here, $N_1 = N_2$. These numbers help to determine the distribution of functional and infrastructure IP blocks in a SoC. Assuming IP blocks consisting of 100 K gates, Table 1 shows the maximum number of 100 K gates IP blocks that can be integrated in a single SoC in different ITRS technology nodes [4,21]. There are plenty of evidences in support of IP blocks amounting to such sizes [21]. The values of the number of IP blocks shown in Table 1 are obtained by dividing the maximum number of gates[1] that can be integrated in a 20 mm × 20 mm die in different technology nodes by 100 K gates [8].

From our implementation, we found that the switches consist of 27–30 K gates depending on the specific architecture considered. Consequently, the distribution of functional and infrastructure IP blocks for the different NoCs is indicated in Table 2.

The wire lengths between switches in the BFT and the SPIN architectures depend on the levels of the switches. On the other hand, the number of switch levels can be expressed as a function of system size (N) as levels $= \log_4 N$ for both, where N is the number of functional IP blocks in a SoC. The inter-switch wire length is given by the following expression [21]

$$w_{a+1,a} = \frac{\sqrt{\text{Area}}}{2^{\text{levels}-a}} \tag{6.2}$$

where $w_{a+1,a}$ is the length of the wire spanning the distance between level a and level $a+1$ switches, where a can take integer values between 0 and (levels − 1). Table 3 shows the inter-switch wire lengths for the BFT and SPIN architectures in all the technology nodes assuming a die size of $\sqrt{\text{Area}} = 20$ mm. In the table, × indicates that the particular inter-switch wire is not present in the given technology node.

From Table 3, the longest inter-switch wire length in the BFT and SPIN is 10 mm.

In CLICHÉ, the inter-switch wire lengths can be determined from the following expression:

$$w = \frac{\sqrt{\text{Area}}}{\sqrt{N} - 1} \tag{6.3}$$

As the system size N, which is the number of functional IP blocks, varies among the technologies in CLICHÉ, the inter-switch wire lengths differ for the different technology nodes. However, in a specific technology node all of them are of same length. In the Torus architecture, all the inter-switch wire lengths are the same as those for the CLICHÉ except for the wraparound wires, which will have a length of $\sqrt{\text{Area}} = 20$ mm. In the Folded Torus, all the inter-switch wire lengths are double those for the CLICHÉ architecture [28]. Table 4 shows the inter-switch wire lengths in the case of CLICHÉ, Torus and Folded Torus architectures for different ITRS technology nodes.

As before, we can compute the intrinsic RC delay of a wire according to the equation below [22]

$$D_{\text{unbuffered}} = 0.4 R_w C_w L^2 \tag{6.4}$$

where L is the wire length. In different technology nodes, the corresponding FO4 delay can be estimated as $425 \times L_{\text{min}}$, where L_{min} is the minimum gate length in each technology node [8]. For long wires, the intrinsic delay will easily exceed the 15 FO4 limit. In those cases, the delay can, at best, be made to increase linearly with wire length by

[1] Here, we considered a two-input minimum-sized NAND structure as a reference gate.

Table 3
Inter-switch wire lengths in mm (Tree-based architectures)

Technology node (nm)	No. of levels	$W_{11,10}$	$W_{10,9}$	$W_{9,8}$	$W_{8,7}$	$W_{7,6}$	$W_{6,5}$	$W_{5,4}$	$W_{4,3}$	$W_{3,2}$	$W_{2,1}$
130	6	×	×	×	×	×	10.000	5.000	2.500	1.250	0.625
90	7	×	×	×	×	10.000	5.000	2.500	1.250	0.625	0.312
65	9	×	×	10.000	5.000	2.500	1.250	0.625	0.312	0.156	0.078
45	10	×	10.000	5.000	2.500	1.250	0.625	0.312	0.156	0.078	0.039
32	11	10.000	5.000	2.500	1.250	0.625	0.312	0.156	0.078	0.039	0.019

inserting buffers. If the wire is divided into n segments and n inverters are inserted, then the total delay of the buffered wire will be according to the following expression [22] (which is similar to the one given earlier for bus-based systems):

$$D_{\text{buffered}} = nt_{\text{inv}} + \left(C_{\text{G}}R_{\text{w}}m + \frac{C_{\text{w}}R_{\text{eqn}}}{m}\right)L + 0.4R_{\text{w}}C_{\text{w}}\frac{L^2}{n} \tag{6.5}$$

where m is the relative size of the inverters with respect to the minimum sized inverter, t_{inv} is the delay of an inverter driving another identical inverter, C_{G} is the gate capacitance of the minimum size inverter, R_{eqn} is the resistance of n-type diffusion region in Ω/γ, R_{w} and C_{w} are the resistance and capacitance per unit length of the wire, respectively, and L is the wire length.

Fig. 20 reports buffered global wire delay, D_{buffered}, versus wire length in successive technology nodes. The wire capacitance and resistance values are scaled for each technology according to ITRS [8]. From the plots, in the case of BFT and SPIN, the longest inter-switch wires can be driven by a clock of period of 15 FO4 after proper buffer insertion. Our analysis also shows that most of the inter-switch wires in case of BFT and SPIN need not be buffered [20]; shadings in Table 3 denotes the wire segments that need to be buffered. The only apparent exception occurs at the 32 nm technology node for the wire segment between levels 11 and 10 of the communication network. These wire segments are projected to be 10 mm long, whereas the maximum buffered wire length that can be driven by a clock of 15 FO4 in this particular technology node is 8 mm. There are techniques available to resolve this issue such as in [28], but undoubtedly new techniques will be developed by the time the industry reaches the 32 nm technology node.

Table 4
Inter-switch wire lengths for CLICHÉ, Torus and Folded Torus

Technology nodes (nm)	Inter-switch wire length		
	CLICHÉ (mm)	Torus (mm)[a]	Folded Torus (mm)
130	1.1	1.1	2.2
90	0.73	0.73	1.46
65	0.46	0.46	0.92
45	0.27	0.27	0.54
32	0.23	0.23	0.46

[a] With the exception of wrap-around wires.

Considering the same die size, inter-switch wires for the CLICHÉ architecture need not be buffered. The wrap-around wires in the Torus need buffer insertion, and, from Fig. 18, in some cases their delays will exceed the limit of one clock cycle even with buffer insertion. Once again, this can be overcome through the same procedure of [28]. In the case of the Folded Torus architecture the inter-switch wires need not be buffered. The delay will always be less than 15 FO4.

The above analysis illustrates the important feature of the NoC architectures, whereby all the inter-switch wire lengths and corresponding delays can be determined a priori. Moreover the parasitic capacitance due to the functional IP blocks does not load these wire segments directly. Consequently, the delay along these segments can be determined independently of the processing nodes. The regular structure of NoC topologies simplifies the layout and even with first-order placement solutions as depicted in Figs. 10, 12–15, the timing constraints of high performance SoC designs can be relatively easily met.

6.2. Circuit delay through the switches

The switches principally have three stages: input arbiter, routing (switch traversal) and output arbiter. The particular

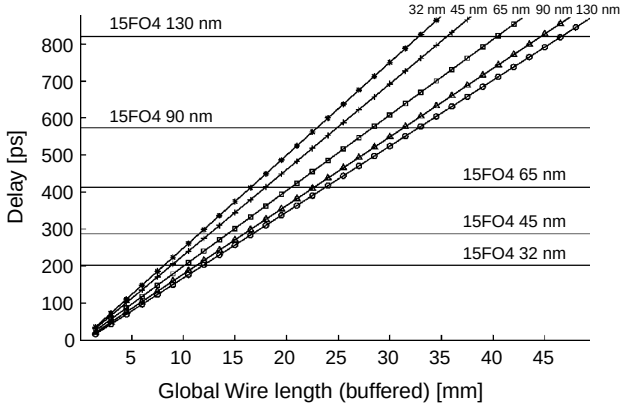

Fig. 20. Buffered global wire delay in different technology nodes.

$$\begin{bmatrix} X & p_{12} & p_{13} & p_{14} \\ p_{21} & X & p_{23} & p_{24} \\ p_{31} & p_{32} & X & p_{34} \\ p_{41} & p_{42} & p_{43} & X \end{bmatrix}$$

Fig. 21. Priority matrix.

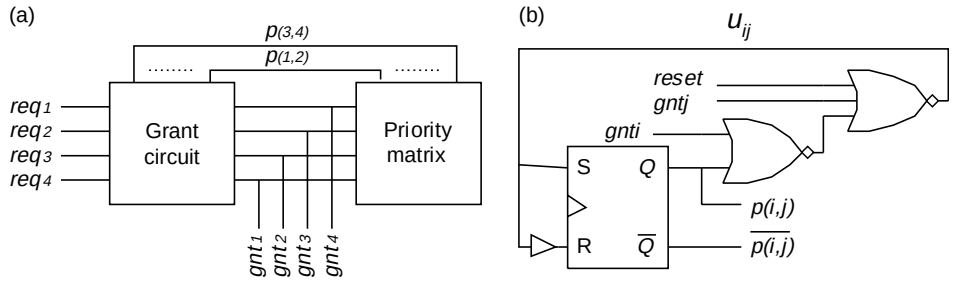

Fig. 22. (a) Block diagram of an arbiter; (b) one element of the priority matrix.

design of these blocks varies from one architecture to another. Through circuit-level design and analysis we show that the delay in each of these stages for all the topologies under consideration can be constrained within the limit of 10–15 FO4.

6.2.1. Arbiter

The arbiter circuit essentially consists of a priority matrix [24], which stores the priorities p_{ij} of the requesters and grants generation circuits used to grant (gnt_i) resources to requesters. The priority matrix stores priorities between n requesters in a binary $n \times n$ matrix. The structure of the matrix in case of four requesters is shown in Fig. 21. The priority of a requester with respect to itself does not have any physical significance and hence the elements along the main diagonal in the priority matrix are void and denoted by X.

Each matrix element $[i, j]$ records the binary priority between each pair of inputs. For example, suppose requester i has a higher priority than requester j, then the matrix element $[i, j]$ will be set to 1, while the corresponding matrix element $[j, i]$ will be 0. A requester will be granted the resource if no other higher priority requester is bidding for the same resource. Once a requester succeeds in being granted a resource, its priority is updated and set to be the lowest among all requesters. The block diagram of the matrix arbiter and the circuit diagram to implement one element of the priority matrix are shown in Fig. 22.

The delay of the arbiter circuit depends on the number of requesters. For the input arbiter, the number of virtual channels governs the number of requesters. The primary role of the virtual channels is to increase channel utilization so that overall throughput of the system increases [25,27]. In [27], it is shown that for a multi processing system- under different data patterns, throughput shows a saturating trend if the number of virtual channels is increased beyond four. Each extra virtual channel beyond this limit does not significantly improve system throughput. Consequently in our design, we set the number of virtual channels to four. Hence for all the NoC architectures, we need a 4:1 arbiter at the input.

The output arbiter depends on the total number of ports in a switch for a particular architecture. If a switch has k ports then a $(k-1)$:1 arbiter is needed at each output port. Consequently for BFT and SPIN we need 5:1 and 7:1 arbiter at the output respectively and for CLICHÉ, Torus and Folded Torus a 4:1 arbiter is needed at each output port.

6.2.2. Routing block

The hardware implementation of the routing block depends on the specific routing algorithm adopted. In the case of BFT and SPIN, the LCA (least common ancestor) based [23] routing algorithm is followed. In this case, the first step in the implementation of the routing logic involves the bit-wise comparison (XOR) of the source and destination addresses taking the most significant, i.e. $M = (\log_2 N - 2^l)$ bits, where N is the number of functional IP blocks in the system and l denotes the level number of the switch. Subsequently, the result of the comparison is checked, i.e. whether any '1' results from the bitwise XOR operation. The basic structure of the hardware block implementing the LCA algorithm is shown in Fig. 23.

The routing algorithm adopted for the CLICHÉ, Torus and the Folded Torus is the *e-cube* dimension order routing [23]. Here, the addresses of all the nodes are divided into two groups representing the X- and Y-coordinates. At each node, the X and Y parts of the destination addresses are

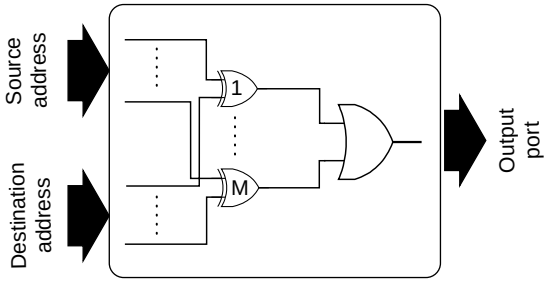

Fig. 23. Block diagram of the LCA routing circuit.

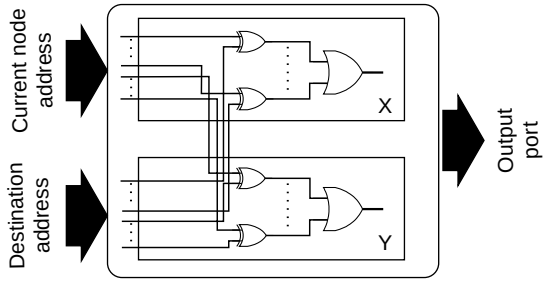

Fig. 24. Block diagram of the *e-cube* routing circuit.

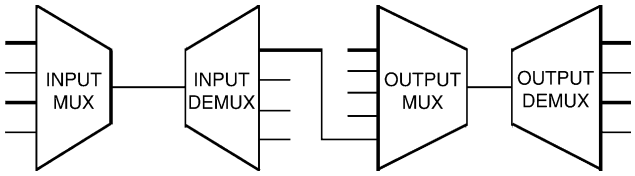

Fig. 25. Switch traversal circuit.

compared with the corresponding parts of the current nodes to determine the next node in the path. The hardware implementation of the *e-cube* algorithm is as shown in Fig. 24.

6.2.3. Switch traversal

Routing decisions are made once the header flit reaches the switch and subsequent body flits just follow the same path. Consequently, from Fig. 17, the switch traversal process involves a chain of multiplexers and demultiplexers as shown in Fig. 25.

6.3. Experimental results

We developed VHDL models for the complete switches in all the topologies mentioned above and synthesized them using Synopsys' synthesis tool in a CMOS 0.13 μ standard-cell based technology. We used Synopsys™ Prime Time to determine the delay along the critical path in all the building blocks of the switches. The results are shown in Table 5. To have a technology-independent measure of the delays we also converted the absolute values obtained from Prime Time timing analysis tool to FO4 delay units.

Due to the fact that there are four virtual channels at the inputs, irrespective of the NoC architecture, the delay of the input arbitration block has the same value for all of them. Furthermore, in the case of CLICHÉ, Torus and Folded Torus, the number of ports in a switch is identical, and thus, the delays of the output arbitration block are equal. These results indicate that the delay associated with each stage of operation of the switch is well within the ITRS-suggested limit of 15 FO4, and can therefore, be driven by a clock with a period of 15 FO4.

The structured inter-switch wires and the processes underlying the switch operations yield four types of pipelined stages. Together, these four types characterize the functionality of the communication fabric. This is illustrated in Fig. 26. Through careful design and analysis, we have shown how to constrain the delays of each of these stages such that they are bounded by the clock period limits suggested by ITRS [8] for high performance multi-core SoC platforms.

7. Conclusions

Multi-core SoC platforms are emerging as the trend for future SoC designs. A single monolithic bus-based architecture cannot generally meet the clock cycle requirements for these systems. One solution is to simply split up the single bus into a network of interconnected smaller buses. Ad hoc solutions tend to constitute locally optimized solutions that are not easily automated, portable, or scalable. The NoC design paradigm overcomes this clock cycle limitation problem based on a highly-structured architecture.

In this paper, we present compelling arguments for the adoption of structured NoC-based topologies as interconnect fabric as they greatly simplify the design process and provide predictable timing characteristics. A complete SoC interconnection network can be constructed by dividing the communication medium into multiple pipeline stages. From detailed circuit level design and analysis, we

Table 5
Delay through the switches

NoC architecture	Delay of the intra-switch pipelined stages							
	Input arbitration ($t_{i.a.}$)		Routing (t_r)		Switch traversal ($t_{s.t.}$)		Output arbitration ($t_{o.a.}$)	
	(ps)	FO4 units	(ps)	FO4 units	(ps)	FO4 units	(ps)	FO4 units
SPIN	500	9	360	6.5	310	5.6	608	11
CLICHÉ	500	9	331	6	221	4	500	9
Torus	500	9	331	6	221	4	500	9
Folded Torus	500	9	331	6	221	4	500	9
BFT	500	9	276	5	275	5	555	10

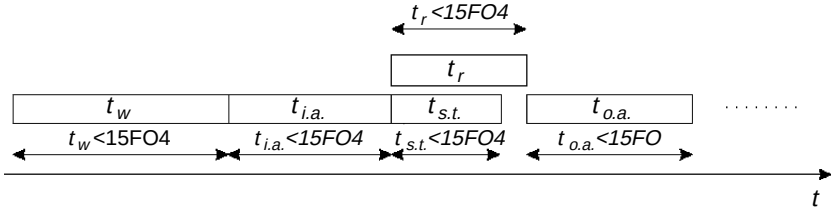

Fig. 26. Delays associated with the pipelined stages on data path.

demonstrated that for all the NoC topologies considered here, these stages can be clocked with a minimum feasible time period of 15 FO4 delay units irrespective of the system size.

It can be argued that this clock cycle constraint can also be achieved in a hierarchical bus-based system. However, multiple design iterations may be required due to the fact that the IP blocks directly affect the achievable clock cycle by capacitively loading the communication fabric. On the other hand the clock cycle requirement can be met independently of the IP blocks in NoC architectures. This would allow a decoupling of the design and optimization of the communication fabrics and the functional IP blocks.

In effect, we have demonstrated how network on chip (NoC) types of interconnect architectures overcome the inherent non-scalability associated with traditional bus-based systems. Any perceived constraints of the structured architectures are well outweighed by the performance benefits and other portability, scalability, and design automation benefits.

Acknowledgements

The authors wish to thank Micronet, Gennum, PMC-Sierra, and NSERC for their financial support and the CMC for providing access to CAD tools. We thank the expert reviewers for their insightfull comments.

References

[1] L. Benini, G. De Micheli, Networks on Chips: a New SoC paradigm computer 35 (1) (2002) 70–78.
[2] P. Magarshack, P.G. Paulin, System-on-Chip beyond the nanometer wall, Proceedings of DAC, Anaheim, USA, June 2–6, 2003. pp. 419–424.
[3] M. Horowitz, B. Dally, How scaling will change processor architecture, Proceedings of ISSCC, San Francisco, USA, February 15–19, 2004. pp. 132–133.
[4] M. Horowitz, et al., The future of wires, Proceedings of the IEEE 89 (4) (2001) 490–504.
[5] K.C. Saraswat, et al., Technology and reliability constrained future copper interconnects—part II: performance implications, IEEE Transactions on Electron Devices 49 (4) (2002) 598–604.
[6] Cheng-Ta Hsieh, Massoud Pedram, Architectural energy optimization by bus splitting, IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems 21 (4) (2002) 408–414.
[7] Y. Zorian, Guest editor's introduction: what is infrastructure IP?, IEEE Design & Test of Computers 19 (3) (2002) 3–5.
[8] ITRS 2003 Documents http://public.itrs.net/Files/2003ITRS/Home2003.htm.
[9] AMBA Bus specification, http://www.arm.com.
[10] Wishbone Service Center, http://www.silicore.net/wishbone.htm.
[11] CoreConnect Specification, http://www-3.ibm.com/chips/products/coreconnect/.
[12] D. Wingard, MicroNetwork-based integration for SoCs, Proceedings of DAC, Las Vegas, Nevada, USA, June 18–22, 2001. pp. 673–677.
[13] Open Core Protocol, www.ocpip.org.
[14] MIPS SoC-it, www.mips.com.
[15] S. Kumar, et al., A network on chip architecture and design methodology, Proceedings of ISVLSI, 2002. pp. 117–124.
[16] W.J. Dally, B. Towles, Route packets, not wires: on-chip interconnection networks, Proceedings of DAC, Las Vegas, Nevada, USA, June 18–22, 2001. pp. 683–689.
[17] I. Saastamoinen, 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.
[18] P. Guerrier, A. Greiner, A generic architecture for on-chip packet-switched interconnections, Proceedings of DATE, Paris, France, March 27–30, 2000. pp. 250–256.
[19] P.P. Pande, C. Grecu, A. Ivanov, R. Saleh, Design of a switch for network on chip applications, Proceedings of ISCAS, Bangkok, May, vol. 5 2003. pp. 217–220.
[20] C. Grecu, P.P. Pande, A. Ivanov, R. Saleh, Structured interconnect architecture: a solution for the non-scalability of bus-based SoCs, Great Lakes Symposium on VLSI, Boston, USA, April 26–28, 2004. pp. 192–195.
[21] D. Sylvester, K. Keutzer, Impact of small process geometries on microarchitectures in systems on a chip, Proceedings of the IEEE 89 (4) (2001) 467–489.
[22] D.A. Hodges, H.G. Jackson, R. Saleh, Analysis and Design of Digital Integrated Circuits third ed., McGraw-Hill, New York, 2003.
[23] J. Duato, S. Yalamanchili, L. Ni, Interconnection Networks—An Engineering Approach, Morgan Kaufmann, 2002.
[24] Hang-Sheng Wang, L.S. Peh, S. Malik, A power model for routers: modeling alpha 21364 and infiniband routers, Proceedings of 10th Symposium on High Performance Interconnects, Stanford, California, USA, 2002. pp. 21–27.
[25] P.P. Pande, C. Grecu, A. Ivanov, R. Saleh, High-throughput switch-based interconnect for future SoCs, Proceedings of third IEEE International Workshop on System-on-Chip for Real-Time Applications, June 30–July 2, Calgary, Canada, 2003. pp. 304–310.
[26] W.J. Dally, C.L. Seitz (1986) The Torus Routing Chip, Technical Report 5208: TR: 86; Computer Science Department, California Institute of Technology, 1986, pp. 1–19.
[27] W.J. Dally, Virtual channel flow control, IEEE Transactions on Parallel and Distributed Systems 3 (2) (1992) 194–205.
[28] K.S. Stevens, Energy and performance models for clocked and asynchronous communication, Proceedings of the Ninth International Symposium on Asynchronous Circuits and Systems, Vancouver, Canada, May, 2003. pp. 56–66.