Abstract—Network-on-Chip paradigm is emerging as the solution for the problems of interconnecting dozens of cores into a single system on chip. However, there are many problems associated with the design of such systems. These problems arise from non-scalable global wire delays, failure to achieve global synchronization, and difficulties associated with non-scalable bus-based functional interconnects. This paper first discusses design issues for on-chip interconnect network for future Gigascale MCSoC architectures. The architecture and characterization of various components of a basic network on chip, named BANC, are presented. Buffers are used in network on chip architectures to store packets. Compared to the control logic the buffers are functionally simple, but when implemented in NoCs architectures, they consume most of the silicon area. However, the smaller the buffers are the bigger is the possibility that some traffic is lost. This paper also studies the properties of these buffers for different traffic sizes and communication loads.

Index Terms—MCSoCs, Scalable, SmartRoute, Trade-offs, on-chip network.

I. INTRODUCTION

Deep sub-micron processing technologies have enabled the implementation of new application-specific embedded architectures that integrate multiple software programmable processors and dedicated hardware components together onto a single chip. Recently, these application-specific architectures are emerging as a key design solution for today’s non-electronics design problems, which are being driven by emerging applications in the areas of: (1) wireless communication, (2) broadband/distributed networking, (3) distributed computing, and (4) multimedia computing. Despite these new opportunities, designers of these systems are currently confronted with the enormously difficult task of designing these complex heterogeneous multicore architectures.

The International Technology Roadmap for Semiconductors (ITRS) predicted that ICs will have billions of transistors, with feature sizes around 50 nm and clock frequencies around 10 GHz in 2012 [16]. This rapid evolution challenges designers to develop more powerful design methods to keep pace.

One approach to addressing the design methodologies, is to adopt the so-called reusability feature to boost design productivity. In the past years the primitive design units evolved from transistors to gates, finite state machines, and processor cores. The network on chip paradigm offers this attractive property for the future and will be able to close the productivity gap. From interconnection links level, wires are becoming increasingly the bottleneck, making transistors play second role. As results, wires dominate performance figures, power consumption and area utilization [1]. At the system level, this means that data transfer on chips is gaining more importance than computation. Yet, deep submicron effects like cross coupling [23], noise [24], [18], and transient errors [25] require extensive knowledge to control, while exploiting the most up-to-date technologies. Due to these physical effects, the clock signal will need numerous clock cycles to traverse the chip, and the clock distribution tree will give the major part of the power consumption and will make synchronous communication infeasible. Signal integrity will also be compromised due to increased RC effects, inductance, and cross-coupling capacitances [19]. Synchronization and clock skew problems in future Gigascale systems can be avoided by using the Globally Asynchronous Locally Synchronous (GALS) scheme.

Network-on-Chips (NoCs) [6], [7], [9] are becoming an attractive option for solving the aforementioned difficulties and problems. NoC is a scalable architectural platform with huge potential to handle growing complexity and can provide easy re-configurability. The basic idea is that processors are connected via a packet switched communication network on a single chip similar to the way computers are connected to internet. A chip based on NoC interconnection consists of several network clients (e.g. processors, memories, and custom logic) which are connected to a network that routes packets between them. Packet switching supports asynchronous transfer of information. It provides extremely high bandwidth by distributing the propagation delay across multiple switches, thus pipelining the signal transmission. In addition, the NoC offers several promising features. First, it transmits packets instead of words. Dedicated address line like in bus systems are not necessary since the destination address of a packet is part of the packet. Second, transmission can be conducted in parallel if the network provides more than one transmission channel between a sender and a receiver. Thus, unlike bus-based system on chip, NoC presents theoretical infinite scalability, facile IP core
During the last few years, several research groups adopted various concepts from conventional parallel and distributed (Internet) computing world and investigated various design issues related to NoCs [6], [7], [8], [9], [10], [5], [12]. The majority performed theoretical analysis or software simulation. Few researches investigated actual hardware implementation and analysis [2], [4], [3], [5].

This paper’s major contribution is investigating the design issues of a generic network on chip taking into account both computation and communication parts and placing them on the same footing. This orthogonalization of concerns is essential to the success of a re-use strategy as has been realized in recent years. This work is inserted in the architectural context and is part of BANC (Basic Network on Chip Network) project [21], which aims the development of new algorithms architectures and design methodology for the interconnection of future Gigascale Multicore in SoCs.

The rest of this paper is organized as follows. Section 2 gives NoCs design issues and related works. Section 3 analyses major NoCs design complexities. Section 4 gives our proposed BANC architecture. Section 5 gives some evaluation results, toward the design of efficient switch component for the BANC architecture. The last sections give the discussion and conclusion.

II. ON CHIP INTERCONNECTION DESIGN ISSUES

On chip interconnection networks use layered approaches, which are well suited to describe protocol functions that operate on data units at different levels of abstraction (in the form of streams, packets, bits or analog waveforms) and that are subject to various time granularity constraints. Each layer may include one or more closely related protocol functions, such as data fragmentation, encoding and synchronization. Similar to interconnection networks for conventional parallel computers, the NoC interconnection paradigm is also characterized by its topology, protocol, and flow control.

There are several decisions that should be made to design such system. These decisions should be made on communication protocol, switching style, network topology, clock synchronization method, signaling scheme, etc. We show in Figure 1 the major components that make a typical system based on NoC paradigm.

Various type of interconnect architectures for MCSoC architectures have been proposed so far. Most of them borrowed ideas from the area of parallel computing with the consideration of different set of constraints, such as power, complexity, etc. The common desires are low latency and high throughput. The latter depends on the flow control mechanism, which deals with the allocation of channel and buffer resources to packets as they traverse paths [13].

In the rest of this section, we will describe major challenges and design issues of systems based on packet switched interconnection network systems.

1) Nodes interconnection - Topology: The topology of a typical NoC system simply defines how the nodes are interconnected by links. Topologies can vary depending on system’s modules sizes and placements (functional requirements). There are several well known standard topologies on which an application can be mapped. We broadly classify them as direct and indirect topologies. In direct topology, each core is connected to a single switch as shown in Figure 2(a). For indirect topology, a set of cores are connected to a single switch as shown in Figure 2(b). Fat tree [8], folded torus [7], and regular mesh [9], [10], which belong to the second type have been already proposed. Lee found in [2] that considering the energy and area cost together, the hierarchical star topology is the most energy-efficient and cost-effective topology in general.

2) Packets movement - Protocol: There is a large protocol space to select from for NoCs. Circuit switching, packet switching, and wormhole switching are possible choices for NoC protocols [1]. These schemes can be mainly distinguished by their flow control methodologies. When these switching techniques are implemented in on-chip networks, they will have different performance along with different requirements on hardware resources.

In circuit switching, a physical path from the source to the destination is reserved prior to the transmission of data. Once a transmission starts, the transmission is not corrupted by other transmission since packets are not stored in buffer as in packet switching (discussed later). The advantage of circuit switching approach is that the network bandwidth is statically reserved for the whole duration of the data. Moreover, because circuit switching does not need packet buffers, area and power consumption can be reduced.

The overhead of this approach is that the setting up of an end-to-end path causes unnecessary delay. In summary, circuit switching can provide high performance but little flexibility.

Another alternative to circuit switching is so named packet switching scheme. Packet based communication has been brought to NoCs from the Internet world but loses the original advantage of reliability in the absence of dynamic routing. Currently, most of the proposals for routing in NoCs are based upon static routing mechanisms — XY-coordinate discipline. In packet based communication, data is divided into fixed length packets and whenever the source has a packet to be sent, it transmits the data. Every packet is composed of a control part, the header, and a data part (also named payload). Network
switches inspect the headers of incoming packets to switch the packet to the appropriate output port. In this scheme, the need for sorting entire packets in a switch makes the buffer requirement very high [1]. The last technique is the so called wormhole packets-switching [17], where each packet is further divided into flits (flow control unit) and the input and output buffers are expected to store only a few flits [17]. Therefore, the buffer space requirement in the switches can be small and compact compared to the packet switching scheme. The header flit reserves the routing channel of each switch, the biddy flits will then follow the reserved channel, and the tail flit will later release the channel reservation. The advantage of the wormhole routing is that it does not require the complete packet to be stored in the switch’s buffer while waiting for the header flit to route to the next stages. Thus, it requires much less buffer spaces. One packet may occupy several intermediate switches at the same time. Therefore, because of these advantages, wormhole routing is an ideal candidate switching technique for on-chip multiprocessor interconnects networks.

The disadvantage is that by allowing a message to occupy the buffers and channels, wormhole routing increases the possibility of deadlock. In addition, channel utilization is somehow decreased if a flit from a given packet is blocked in a buffer [4]. For clarity, we summarize the network latency in a wormhole routing with the following formula:

\[
\text{Latency} = T_{\text{channel}} \frac{L_{\text{packet}}}{W_{\text{channel}}} + T_{\text{channel}}D_{\text{path}}
\]  (1)

Where \(T_{\text{channel}}\) is the channel transmission time, \(L_{\text{packet}}\) is the length of the packet in bits, \(W_{\text{channel}}\) is the width of the channel in bits, and \(D_{\text{path}}\) is the length of the path between the source and destination nodes. The wormhole routing often reduces the effect of path length on communication latency. The deadlock problem in wormhole routing can be efficiently avoided by using virtual channel solution [14], [15]. Each physical channel is associated with several small FIFOs (Figure 3 (b)) rather than a single deep FIFO (Figure 3 (a)). 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. In addition to increasing throughput, VCs provide an additional degree of flexibility in allocation resources to packets in the network [15]. Adding VCs from control to a network makes more effective use of both physical channel and memory buffers. The only expense is a small amount of additional control logic [15].

3) Flow control: This is used only for dynamic routing. Flow control determines how resources, such as buffers and channels bandwidth are allocated and how packet collisions are resolved. Whenever the packet is buffered, blocked in place, dropped, or misrouted depends on the flow control strategy. A good flow control strategy should avoid channel congestion while reducing the latency.

4) Packet Size Selection: The packet size highly depends on the characteristics of the application being used (i.e. data or control dominated). If a message has to be split in too many small packets which have to be re-assembled at destination to obtain the original message, the resulted overhead will be too high. On the other case, if the packet is too large, the packet might block the link for too many cycles and potentially block other traffic with side effects on the performance of the whole system. Therefore, the correct packet size is also crucial to make optimum use of the network resources.

5) Virtual Channels Allocation: The allocation issue is considered when virtual channels are used. The VCs allocation scheme determines the appropriate channel output for a message at each intermediate switch nodes. As illustrated in Figure 4, each input port of the switch has a separate FIFO buffer corresponding to a VC.

When a given flit first arrives at an input port, its type is first decoded. If it is a header flit, then, according to its virtual channel number (VCID) field, it is stored in the corresponding...
virtual channel buffer. The routing circuit determines the output port to be taken by this flit and assigns the incoming flit to an available output virtual channel. When the body flits arrive, they are queued into the buffer of the input virtual channel and inherit the particular output virtual channel reserved by the header.

A. Interconnection Complexities

On-chip architecture greatly simplifies the design of the functional blocks because it decouples computation from communication. However, it makes the communication mechanism more complex. There are many performance depending parameters that should be determined: (1) the sizes of FIFO memories in each network interface, (2) the sizes of packet. In wormhole switching, small packets create less contention because they use routers during a small delay. The problem of this scheme is that each packet needs a header with routing information. So raising the number of packet increases the network traffic; (3) Functions mapping. When function partitioning is done, the positioning of functional unit must be done carefully in order to minimize routing path lengths; (4) Timing performances in a NoC are not easily predictable. Even if the bandwidth per link is high, traffic congestion in a node can create long latencies, which slows down the system.

III. BANC Specification and Building Blocks

The BANC platform is based on $S$ array, $d$ dimensional mesh architecture built by its dimension $d$ and array $S$. This leads the number of switches to be $S^d$. The $S^d$ switches are implemented on a $d$ dimensional, with $S$ switches on each dimension. The total number of implemented cores (C) is:

$$C_{mesh} = S^d(d - 1)$$  \hspace{1cm} (2)

The total bandwidth is obtained by:

$$B_{mesh} = 2S^db$$  \hspace{1cm} (3)

Where $b$ is the one directional bandwidth. We have to note that since this architecture achieves the simple connectional scheme with the complexity order $O(S^d)$, the shortest path routing algorithm is mostly applied to it. Each tile is composed of a resource (core), which can be a processor (conventional or QC-2 [26]), a memory, an FPGA, a custom made hardware block or any other IP that complies with the interface of the BANC. To attach more cores having different interfaces to a BANC network terminal, one needs to use smart wrappers (the network interface) to make the protocol adaptation. Similar to other proposed NoC architectures, the important building block of BANC is the switch, which is embedded onto each tile and is connected to the four neighboring tiles and its local core via channels as shown in Figure 5. Each channel consists of two bi-directional point-to-point links between two routers or a router and a local processing core (PC).

The switch architecture has great impact on the costs and on the performance of the whole network. Each switch has a unique address in the network. To simplify routing on the network this address is expressed in $XY$ coordinates, where $X$ represents the horizontal position and $Y$ the vertical position. BANC architecture uses wormhole packet switching, and messages are sent by means of packets (several flits). Therefore, the switching has low latency, saves memory buffers and, with appropriate routing algorithm, communication deadlock can be avoided.

A flit (flow control unit) is the smallest unit over which the flow control is performed. In BANC, as in the wormhole packet switched NoC, a flit equals the physical channel (phit) size.

For packets routing, BANC uses the simple deterministic $XY$ routing. Each switch (named SmartRoute) in the BANC is identified by a pair of coordinates $(X_{id}, Y_{id})$ as shown in figure 6. For example, the left top switch in the figure has coordinates $X_0 = 0, Y_0 = 0$ and the right bottom switch has $X_3 = 3, Y_3 = 3$.

When a packet is sent by a core attached to a switch port, the master (sender) must include, in the packet header, the coordinates of the switch at which the destination core (slave) is attached. The coordinate is used by each switch in the packet path, which compares the destination address with its own address, so that it can determine the output channel of the
A. Communication

BANC is based on the message passing communication model. Cores communicate by sending and receiving request and response signals. Every core (resource) has a unique address and is connected to the BANC via a switch. It communicates with the switch through its network adapter. Hence, any resource can be plugged into the network if it is equipped with an adapter. It is possible to add additional protocols on top of the transport layer to provide for instance a virtual shared memory abstraction, which will help the programmers.

The BANC’s network interface specifies four protocol layers: (1) Physical layer, (2) data link layer, (3) network layer, and (4) Transport layer. The switch to switch interfaces implements only the three lower protocol layers as shown in Fig 7.

B. BANC Packet format

The packet format has seven fields. One of them is reserved. The header flit is the one in which P_BEG equals 1, and it is always the packet header. This header is composed by routing bits and RES (reserved) fields. The routing bits are information used by each SmartRoute (switch) to perform the packet routing. That is, the coordinates of the destination core. RES is reserved for future expansion, i.e., for the implementation of protocols above the network layer (e.g., information for packet reordering), and it is not processed by the routers. Data (Payload), which has an unlimited length, comes immediately after the header flit. The last payload flit has P_END equals 1, which means this flit as the packet trailer. In Fig 8 (a), (b) the header and the data packets format are shown respectively.

C. Switching Policy

In BANC architecture, the on-chip cores interconnection is arranged as a mesh of switches. Each switch has five inputs and five outputs (see Figure 5). One input/output pair is for communication with the resource. The remaining four pairs are connected to the surrounding switches. The destination direction of a node is simply the direction to the destination switch of which the packet aims. When a challenge of direction occurs, the simplest method is to let one input choose first and then consequently let the remaining packets in some predefined order choose. As we mentioned, all cores within BANC communicate with each other with the help of intelligent switches.

1) Switch Architecture: The BANC’s switch hardware is simple crossbar circuit. It has 5 special FIFOs with 8x32-bits each and performs all the control flow and routing functions. The crossbar allows 5 different data to be routed at the same time. Every output port has an associated arbiter using simple Round-Robin scheduling scheme, with no priority. The arbiter’s main task is to grant data paths by evaluating how
many available positions the FIFO has and the data size of all requesting FIFOs. The switch architecture is shown in Figure 9. BANC switch ports include two unidirectional channels, each one with its data, framing and flow control signals. The typical values for data field are 8, 16, or 32 bits. The flow control bits are used to validate data at the channel and to acknowledge (ACK) the received data.

The operation of a switch is to perform one or more tasks depending on the format of the flit. If the flit contains a header, the processing sequence is as follows: 1) input arbitration, 2) routing, and 3) output arbitration. In the case of body flits, switch traversal replaces the routing process since the routing decision based on the header information is maintained for the subsequent body flits.

The routing operation consists of four major tasks: (1) Receiving flits from a neighbor node, (2) transmitting flits to a neighboring node, (3) deciding the channel through which a flit must be forwarded (routing), and (3) host resource NI, which involves assembling of flits into messages (whole packet) and disassembling a message into flits.

2) Network Adapter: The network adapter provides the conversion of the packet-based communication of the BANC to the higher-level protocol that cores use. It implements, via connections, high-level services, such as transaction ordering, throughput and latency guarantees, and end-to-end flow control. The network adapter also implements adapters to existing on-chip protocols, such as AXI, OCP and DTL, to seamlessly connect existing IP modules to the NoC. It also decouples computation from communication.

Decoupling computation from communication is key in managing the complexity of designing chips with billions of transistors, because it allows the IP modules and the interconnect to be designed independently [20]. This function is achieved by allowing the network services at the transport level, or above in the ISO-OSI reference model [22]. The NA in BANC consists of a core interface at the core side and a network interface at the network side.

IV. DISCUSSION

Before the real implementation of the BANC network on chip, we need estimation of several communication and computation components design parameters. In our earlier work [26], we proposed and designed a novel architecture (mainly related to the computation component part), which has several promising features needed in future MCSoCs systems, such as, low power, and low hardware complexity. Here in this work, we turn our focus into the communication part and particularly into the buffer design, which is the important part of the switch. To do so, we used a ns-2 simulator from Berkeley [11]. The simulated system is an architecture consisting of 50 components (25 cores and 25 switches) organized as a 5x5 mesh grid. The connections between nodes consist of duplex (simultaneously transfer in both ways) links with adjustable bandwidth and delay. There is a queue (FIFO) for temporary storing of packets at each port. We used droptail mechanism to handle packets overflow, which means that the lastly arrived packet is dropped. We also used random number generation (RNG), to select the X and y coordinates of each source and destination randomly.

Figure 10 shows the effect of buffer size on packet drop probability. Packet drop probability defines the probability of a packet being lost in the network due to heavy traffic and limited buffer capacity in the switches. When a packet arrives at a switch, it must first go into FIFO buffer. If the buffer is empty, the packet will be cast to the output port connected with the next hop of this packet immediately. Otherwise, this packet must wait in the FIFO until there are not other packets prior to it, and then be cast out. In another case, if the FIFO buffer is full and one packet arrives, it will be dropped and lost forever.

The experiment shows that packet drop probability decreases when the buffer size increases. For higher traffic rates (greater or equals than 120Mbits/s), we found that it is not significant that the drop probability decreases with the buffer size increasing from 4 packet to 1024 packets.

![Fig. 10. Effect of buffer size on drop probability. For clarity, the buffer size is given in log2](image-url)
Fig. 11. Effect of communication load on drop probability.

Fig. 12. Packet delay and communication load over buffer sizes.

V. CONCLUSION

In this research work we presented design issues for on chip interconnection architecture for future Gigascale Multicore systems on chips. We also discussed several design issues of so named Basic Network on Chip (BANC architecture).

For the communication part, where our focus is mainly on buffer design in a switch, we made several experiments using software simulation. From the experiments we conclude the following: first, for less than half load, the drop probability is almost zero for buffer size of 8 packets in each switch. Second, the delay in queue is an important part for delay in message. Hence, delay in message is more sensitive to buffer size than communication load. Finally, the drop probability is more sensitive to communication load than to buffer size.

REFERENCES


[21] BANC: Basic Network on Chips Project: http://www2.sowa.is.uec.ac.jp/ ben/espoir/


