International Journal of Advanced Information Science and Technology (IJAIST) ISSN: 2319:268
Vol.2, No.12, December 2013 DOI:10.15693/ijaist/2013.v2i12.58-66

Network-on-Chip (NOC). A Survey Paper

Asnika S.
Dept. of Information Science & Engineering,
N.M.A.M.Institute of Technology,Nitte, India

Sahana Damale
Dept. of Computer Science & Engineering,
N.M.A.M.Institute of Technology,Nitte, India

Abstract*-- *The scaling of microchip technologies has enabled large scale systems-on-chip (SoC). Network-on-chip (NoC) research addresses global communication in SoC, involving (i) a move from computation-centric to communication-centric design and (ii) the implementation of scalable communication structures. This survey presents a perspective on existing NoC research. We define the following abstractions: system, network adapter, network, and link to explain and structure the fundamental concepts. First, research relating to the actual network design is reviewed. Then system level design and modeling are discussed. We also evaluate performance analysis techniques. The research shows that NoC constitutes a unification of current trends of intra chip communication rather than an explicit new alternative.

1. INTRODUCTION

Chip design has four distinct aspects: computation, memory, communication, and I/O. As processing power has increased and data intensive applications have emerged, the challenge of the communication aspect in single-chip systems, Systems-on-Chip (SoC), has attracted increasing attention. This survey treats a prominent concept for communication in SoC known as Network-on-Chip (NoC). As will become clear in the following, NoC does not constitute an explicit new alternative for intra chip communication but is rather a concept which presents a unification of on-chip communication solutions. In this section, we will first briefly review the history of microchip technology that has led to a call for NoC-based designs. With our minds on intra chip communication, we will then look at a number of key issues of large-scale chip design and finally show how the NoC concept provides a viable solution space to the problems presently faced by chip designers.

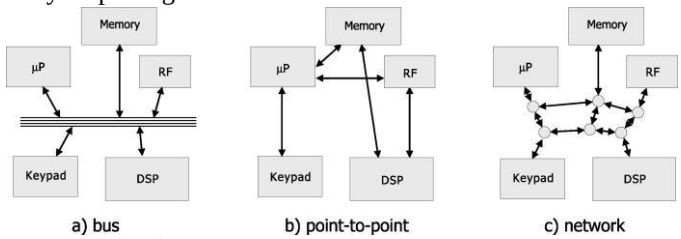

Fig. 1. Examples of communication structures in Systems-on-Chip. a) raditional bus-based communication, b) dedicated point-to-point links, c) a chip area network.

1.1. NoC in SoC

Figure 1 shows some examples of basic communication structures in a sample SoC, for example, a mobile phone. Since the introduction of the SoC concept in the 90s, the solutions for SoC communication structures have generally been characterized by custom designed ad hoc mixes of buses and point-to-point links. The bus builds on well understood concepts and is easy to model. In a highly interconnected multi core system, however, it can quickly become a communication bottleneck. As more units are added to it, the power usage per communication event grows as well due to more attached units leading to higher capacitive load. For multi master busses, the problem of arbitration is also not trivial. Table I summarize the pros and cons of buses and networks. A crossbar overcomes some of the limitations of the buses. However, it is not ultimately scalable and, as such, it is an intermediate solution. Dedicated point-to-point links are optimal in terms of bandwidth availability, latency, and power usage as they are designed especially for this given purpose. Also, they are simple to design and verify and easy to model. But the number of links needed increases exponentially as the number of cores increases. Thus an area and possibly a routing problem develops. From the point of view of design-effort, one may argue that, in small systems of less than 20 cores, an ad hoc communication structure is viable. But, as the systems grow and the design cycle time requirements decrease, the need for more generalized solutions becomes pressing. For maximum flexibility and scalability, it is generally accepted that a move towards a shared, segmented global communication structure is needed. This notion translates into a data-routing network consisting of communication links and routing nodes that are implemented on the chip. In contrast to traditional SoC communication methods outlined previously, such a distributed communication media scales well with chip size and complexity. Additional advantages include increased aggregated performance by exploiting parallel operation. From a technological perspective, a similar solution is reached: in DSM chips, long wires must be segmented in order to avoid signal degradation, and busses are implemented as multiplexed structures in order to reduce power and increase responsiveness. Hierarchical bus structures are also common as a means to adhere to the given communication requirements. The next natural step is to increase throughput by pipelining these structures. Wires become pipelines and bus-bridges become routing nodes. Expanding on a structure using these

elements, one gets a simple network. A common concept for segmented SoC communication structures is based on networks. This is what is known as Network-on-Chip (NoC)

Bus Pros & Cons			Network Pros & Cons
Every unit attached adds parasitic capacitance, therefore electrical performance degrades with growth.	-	+	Only point-to-point one-way wires are used, for all network sizes, thus local performance is not degraded when scaling.
Bus timing is difficult in a deep submicron process.	-	+	Network wires can be pipelined because links are point-to-point.
Bus arbitration can become a bottleneck. The arbitration delay grows with the number of masters.	-	+	Routing decisions are distributed, if the network protocol is made non-central.
The bus arbiter is instance-specific.	-	+	The same router may be reinstantiated, for all network sizes.
Bus testability is problematic and slow.	-	+	Locally placed dedicated BIST is fast and offers good test coverage.
Bandwidth is limited and shared by all units attached.	-	+	Aggregated bandwidth scales with the network size.
Bus latency is wire-speed once arbiter has granted control.	+	-	Internal network contention may cause a latency.
Any bus is almost directly compatible with most available IPs, including software running on CPUs.	+	-	Bus-oriented IPs need smart wrappers. Software needs clean synchronization in multiprocessor systems.
The concepts are simple and well understood.	+	-	System designers need reeducation for new concepts.

Table I. Bus-versus-Network Arguments

2. NOC BASICS

In this section, the basics of NoC are uncovered. First a component-based view will be presented, introducing the basic building blocks of a typical NoC. Then we will look at system-level architectural issues relevant to NoC-based SoC designs. After this, a layered abstraction-based view will be presented, looking at network abstraction models, in particular, OSI and the adaption of such for NoC. Using the foundations established in this section, we will go into further details of specific NoC research in Section 3.

2.1. A Simple NoC Example

Figure 2 shows a sample NoC structured as a 4-by-4 grid which provides global chip level communication. Instead of busses and dedicated point-to-point links, a more general scheme is adapted, employing a grid of routing nodes spread out across the chip, connected by communication links. For now, we will adapt a simplified perspective in which the NoC contains the following fundamental components.

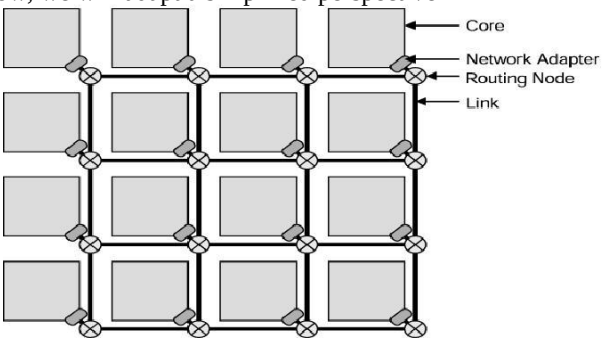

Fig. 2. Topological illustration of a 4-by-4 grid structured NoC, indicating the fundamental components.

—*Network adapters* implement the interface by which *cores* (IP blocks) connect to the NoC. Their function is to decouple computation (the cores) from communication (the network).

—*Routing nodes* route the data according to chosen protocols. They implement the routing strategy.

—*Links* connect the nodes, providing the raw bandwidth. They may consist of one or more logical or physical channels.

Figure 2 covers only the topological aspects of the NoC. The NoC in the figure could thus employ packet or circuit witching or something entirely different and be implemented using asynchronous, synchronous, or other logic. In Section 3, we will go into details of specific issues with an impact on the network performance.

2.2. Architectural Issues

The diversity of communication in the network is affected by architectural issues such as system composition and clustering. These are general properties of SoC but, since they have direct influence on the design of the system-level communication infrastructure, we find it worthwhile to go through them here. Figure 3 illustrates how system composition can be categorized along the axes of *homogeneity* and *granularity* of system cores. The figure also clarifies a basic difference between NoC and networks for more traditional parallel computers; the latter have generally been homogeneous and coarse grained, whereas NoC-based systems implement a much higher degree of variety in composition and in traffic diversity. *Clustering* deals with the localization of portions of the system. Such localization may be logical or physical. Logical clustering can be a valuable programming tool. It can be supported by the implementation of hardware primitives in the network, for example, flexible addressing schemes or virtual connections. Physical clustering, based on preexisting knowledge of traffic patterns in the system, can be used to minimize global communication, thereby minimizing the total cost of communicating, power and performance wise.

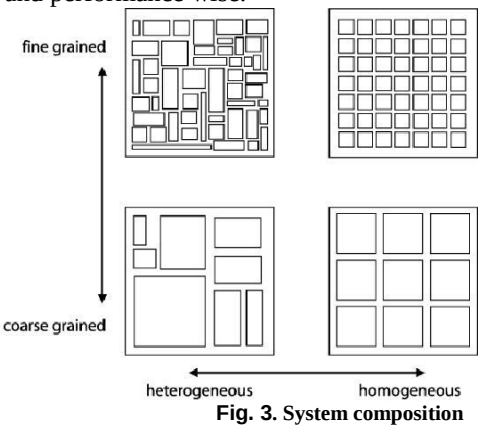

Fig. 3. System composition

Generally speaking, *reconfigurability* deals with the ability to allocate available resources for specific purposes. In relation to NoC-based systems, reconfigurability concerns how the

NoC, a flexible communication structure, can be used to make the system reconfigurable from an application point of view. A configuration can be established for example, by programming connections into the NoC. This resembles the reconfigurability of an FPGA, though NoC-based reconfigurability is most often of coarser granularity. In NoC, the reconfigurable resources are the routing nodes and links rather than wires. Much research work has been done on architecturally-oriented projects in relation to NoC-based systems. The main issue in architectural decisions is the balancing of flexibility, performance, and hardware costs of the system as a whole. As the underlying technology advances, the trade-off spectrum is continually shifted, and the viability of the NoC concept has opened up to a communication-centric solution space which is what current system-level research explores. At one corner of the architectural space outlined in Figure 5, is the Pleiades architecture and its instantiation, the Maia processor. A microprocessor is combined with a relatively fine-grained heterogeneous collection of ALUs, memories, FPGAs, etc. An interconnection network allows arbitrary communication between modules of the system. The network is hierarchical and employs clustering in order to provide the required communication flexibility while maintaining good energy-efficiency. At the opposite corner are a number of works, implementing homogeneous coarse-grained multiprocessors. In Smart Memories, a hierarchical network is used with physical clustering of four processors. The flexibility of the local cluster network is used as a means for reconfigurability, and the effectiveness of the platform is demonstrated by mimicking two machines on far ends of the architectural spectrum, the Imagine streaming processor and Hydra multiprocessor, with modest performance degradation. The global NoC is not described, however. In the RAW architecture, on the other hand, the NoC which interconnects the processor tiles is described in detail. It consists of a static network, in which the communication is preprogrammed cycle-by-cycle, and a dynamic network. The reason for implementing two physically separate networks is to accommodate different types of traffic in general purpose systems (see Section 4.3 concerning traffic characterization). The Eclipse is another similarly distributed multiprocessor architecture in which the interconnection network plays an important role. Here, the NoC is a key element in supporting a sofisticated parallel programming model.

3. NOC RESEARCH

In this section, we provide a review of the approaches of various research groups.

3.1. Network Adapter

The purpose of the network adapter (NA) is to interface the core to the network and make communication services transparently available with a minimum of effort from the core. At this point, the boundary between computation and communication is specified. As illustrated in Figure 4, the NA component implements a core interface (CI) at the core side and a network interface (NI) at the network side. The function of the NA is to provide high-level communication services to the core by utilizing primitive services provided by the network hardware.

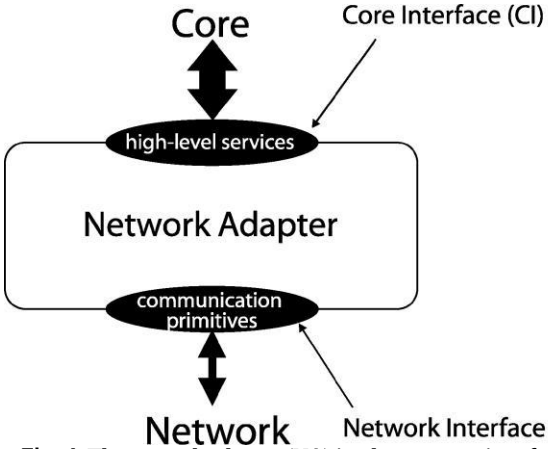

Fig. 4. The network adapter (NA) implements two interfaces, the core interface (CI) and the network interface (NI).

Thus the NA *decouples* the core from the network, implementing the network end-to-end flow control, facilitating a layered system design approach. The level of decoupling may vary. A high level of decoupling allows for easy reuse of cores. This makes possible a utilization of the exploding resources available to chip designers, and greater design productivity is achieved. On the other hand, a lower level of decoupling (a more network aware core) has the potential to make more optimal use of the network resources.

In this section, we first address the use of standard sockets. We then discuss the abstract functionality of the NA. Finally, we talk about some actual NA implementations which also address issues related to timing and synchronization.

3.1.1. Sockets.

The CI of the NA may be implemented to adhere to a SoC socket standard. The purpose of a socket is to orthogonalize computation and communication. Ideally a socket should be completely NoC implementation agnostic. This will facilitate the greatest degree of reusability because the core adheres to the specification of the socket alone, independently of the underlying network hardware. One commonly used socket is the *Open Core Protocol* (OCP). The OCP specification defines a flexible family of memory-mapped, core-centric protocols for use as a native core interface in on-chip systems. The three primary properties envisioned in OCP include (i) architecture independent design reuse, (ii) feature-specific socket implementation, and (iii) simplification of system verification and testing. OCP addresses not only dataflow signaling, but also uses related to errors, interrupts, flags and software flow control, control and status, and test. Another proposed standard is the *Virtual Component Interface* (VCI) used in the SPIN.

3.1.2. NA Services.

Basically, the NA provides *encapsulation* of the traffic for the underlying communication media and *management* of services provided by the network. Encapsulation involves handling of end-to-end flow control in the network. This may include global addressing and routing tasks, reorder buffering and data acknowledgement, buffer management to prevent network congestion, for example, based on credit, packet creation in a packet-switched network, etc. Cores will contend for network resources. These may be provided in terms of service quantification, for example, bandwidth and/or latency guarantees. Service management concerns setting up circuits in a circuit-switched network, bookkeeping tasks such as keeping track of connections, and matching responses to requests. Another task of the NA could be to negotiate the service needs between the core and the network.

3.2. Network Level

The job of the network is to deliver messages from their source to their designated destination. This is done by providing the hardware support for basic communication primitives. A well-built network, should appear as a logical wire to its clients. An on-chip network is defined mainly by its topology and the protocol implemented by it. Topology concerns the layout and connectivity of the nodes and links on the chip. Protocol dictates how these nodes and links are used.

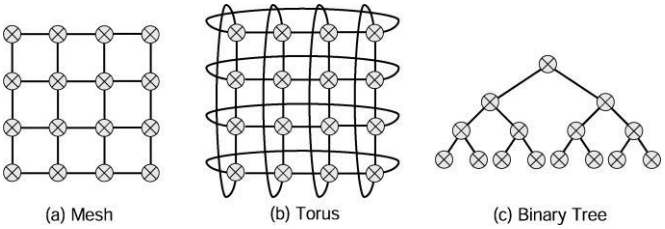

Fig. 5. Regular forms of topologies scale predictably with regard to area and power. Examples are (a) 4-ary 2-cube mesh, (b) 4-ary 2-cube torus and (c) binary (2-ary) tree.

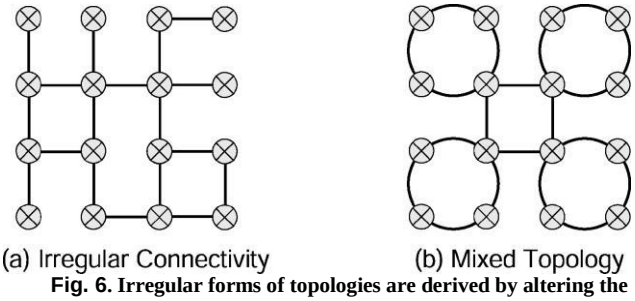

Fig. 6. Irregular forms of topologies are derived by altering the connectivity of a regular structure such as shown in (a) where certain links from a mesh have been removed or by mixing different topologies such as in (b) where a ring coexists with a mesh.

3.2.1. Topology.

One simple way to distinguish different regular topologies is in terms of *k-ary n-cube* (grid-type), where *k* is the degree of each dimension and *n* is the number of dimensions (Figure 5), for multicomputer networks. The *k-ary tree* and the *k-ary n-dimensional* fat tree are two alternate regular forms of networks explored for NoC. The network area and power consumption scales predictably for increasing size of regular forms of topology. Most NoCs implement regular forms of network topology that can be laid out on a chip surface (a 2-dimensional plane) for example, k-ary 2-cube, commonly known as grid-based topologies. The Octagon NoC is an example of a novel regular NoC topology. Its basic configuration is a ring of 8 nodes connected by 12 bidirectional links which provides two-hop communication between any pair of nodes in the ring and a simple, shortest-path routing algorithm. Such rings are then connected edge-to-edge to form a larger, scalable network. For more complex structures such as trees, finding the optimal layout is a challenge on its own right. Besides the form, the nature of links adds an additional aspect to the topology. In k-ary 2-cube networks, popular NoC topologies based on the nature of link are the mesh which uses bidirectional links and torus which uses unidirectional links. For a torus, a folding can be employed to reduce long wires. In the NOSTRUM NoC, a folded torus is discarded in favor of a mesh with the argument that it has longer delays between routing nodes. Figure 5 shows examples of regular forms of topology. Generally, mesh topology makes better use of links (utilization), while tree-based topologies are useful for exploiting locality of traffic. Irregular forms of topologies are derived by mixing different forms in a hierarchical, hybrid, or asymmetric fashion as seen in Figure 6. Irregular forms of topologies scale

3.2.2. Protocol.

The protocol concerns the strategy of moving data through the NoC. We define switching as the mere transport of data, while routing is the intelligence behind it, that is, it determines the path of the data transport. The following are the aspects of protocol commonly addressed in NoC research, are discussed.

—*Circuit vs packet switching.* Circuit switching involves the circuit from source to destination that is setup and reserved until the transport of data is complete. Packet switched traffic, on the other hand, is forwarded on a per-hop basis, each packet containing routing information as well as data.

—*Connection-oriented vs connectionless.* Connection-oriented mechanisms involve a dedicated (logical) connection path established prior to data transport. The connection is then terminated upon completion of communication. In connectionless mechanisms, the communication occurs in a dynamic manner with no prior arrangement between the sender and the receiver. Thus circuit switched communication is always connection-oriented, whereas packet switched communication may be either connection-oriented or connectionless.

—*Deterministic vs adaptive routing.* In a deterministic routing strategy, the traversal path is determined by its source and destination alone. Popular deterministic routing schemes for NoC are source routing and X-Y routing (2D dimension order routing). In source routing, the source core specifies the route to the destination. In X-Y routing, the packet follows the rows first, then moves along the columns toward the destination or vice versa. In an adaptive routing strategy, the routing path is

decided on a per hop basis. Adaptive schemes involve dynamic arbitration mechanisms, for example, based on local link congestion.

—*Delay vs loss.* In the delay model, datagrams are never dropped. This means that the worst that can happen is that the data is delayed. In the loss model, datagrams can be dropped. In this case, the data needs to be retransmitted. The loss model introduces some overhead in that the state of the transmission, successful or failed, must somehow be communicated back to the source. There are, however, some advantages involved in dropping datagram, for example, as a means of resolving network congestion.

—*Central vs distributed control.* In centralized control mechanisms, routing decisions are made globally, for example, bus arbitration. In distributed control, most common for segmented interconnection networks, the routing decisions are made locally. The most common forwarding strategies are store-and-forward, wormhole, and virtual cut-through. These will now be explained. Table II summarizes the latency penalty and storage cost in each node for each of these schemes.

Store-and-forward. Store-and-forward routing is a packet switched protocol in which the node stores the complete packet and forwards it based on the information within its header. Thus the packet may stall if the router in the forwarding path does not have sufficient buffer space

Wormhole. Wormhole routing combines packet switching with the data streaming quality of circuit switching to attain a minimal packet latency. The node looks at the header of the packet to determine its next hop and immediately forwards it. The subsequent flits are forwarded as they arrive. This causes the packet to *worm* its way through the network, possibly spanning a number of nodes, hence the name. The latency within the router is not that of the whole packet. A stalling packet, however, has the unpleasantly expensive side effect of occupying all the links that the worm spans. In Section 3.2.3, we will see how virtual channels can relieve this side effect at a marginal cost.

— *Virtual channels (VCs).* VCs are the sharing of a physical channel by several logically separate channels with individual and independent buffer queues. Generally, between 2 and 16 VCs per physical channel have been proposed for NoC. Their implementation results in an area and possibly also power and latency overhead due to the cost of control and buffer implementation. There are however a number of advantageous uses. Among these are:

—A*voiding deadlocks.* Since VCs are not mutually dependent on each other, by adding VCs to links and choosing the routing scheme properly, one may break cycles in the resource dependency graph.

Protocol	Per router cost		Stalling
	Latency	Buffering	
store-and-forward	packet	packet	at two nodes and the link between them
wormhole	header	header	at all nodes and links spanned by the packet
virtual cut-through	header	packet	at the local node

Table II. Cost and Stalling for Different Routing Protocols

—*improving performance.* VCs can generally be used to relax the inter resource dependencies in the network, thus minimizing the frequency of stalls.

—*virtual wires.* This refers to the use of network message-passing services to emulate direct pin-to-pin connection.

—*complex operations.* Complex functionality such as test-and-set issued by a single command across the network can be used to provide support for, for example, semaphores.

3.3. Link Level

Link-level research regards the node-to-node links. These links consist of one or more channels which can be either virtual or physical. In this section, we present a number of areas of interest for link level research: synchronization, implementation, reliability, and encoding.

3.3.1. Synchronization.

For link-level synchronization in a multiclock domain SoC, have presented a mixed-time FIFO design. The FIFO employs a ring of storage elements in which tokens are used to indicate full or empty state. This simplifies detection of the state of the FIFO (full or empty) and thus makes synchronization robust. In addition, the definitions of full and empty are extended so that full means that 0 or 1 cell is unused, while empty means only 0 or 1 cells is used. This helps in hiding the synchronization delay introduced between the state detection and the input/output handshaking. The FIFO design introduced can be made arbitrarily robust with regards to meta stability as settling time and latency can be traded off. With the emerging of the GALS concept of *globally asynchronous locally synchronous* systems, implementing links using asynchronous circuit techniques is an obvious possibility. A major advantage of asynchronous design styles relevant for NoC is the fact that, apart from leakage, no power is consumed when the links are idle. Thus, the design style also addresses the problematic issue of increasing power usage by large chips. Another advantage is the potentially very low forward latency in uncongested data paths leading to direct performance benefits. Examples of NoCs based on asynchronous circuit techniques are CHAIN, MANGO, ANoC, and QNoC. Asynchronous logic incorporates some area and dynamic power overhead compared with synchronous logic due to local handshake control. The 1-of-4 encodings discussed in Section 3.3.4, generalized to 1-of-N, is often used in asynchronous links. On the other hand, resynchronization of an incoming asynchronous transmission is also not trivial. It costs both time and power, and bit errorsmay be introduced.

3.3.2. Implementation Issues.

As chip technologies scale into the DSM domain, the effect of wires on link delays and power consumption increase. Aspects and effects on wires of technology scaling are presented,

covers the issues specifically from a NoC point-of-view, projecting the operating frequency and size of IP cores in NoC-based SoC designs for future CMOS technologies down to 0.05 μm. In the following, we will discuss a number of physical level issues relevant to the implementation of on-chip links.

Wire segmentation. At the physical level, the challenge lies in designing fast, reliable and low power point-to-point interconnects, ranging across long distances. Since the delay of long on-chip wires is characterized by distributed RC charging, it has been standard procedure for some time to apply segmentation of long wires by inserting *repeater* buffers at regular intervals in order to keep the delay linearly dependent on the length of the wire.

Pipelining. Partitioning long interconnects into pipeline stages as an alternative to wire segmentation is an effective way of increasing throughput. The flow control handshake loop is shorter in a pipelined link, making the critical loop faster. This is at the expense of latency of the link and circuit area since pipeline stages are more complex than repeater buffers. But the forward latency in an asynchronous pipeline handshake cycle can be minimized to a few gate delays so, as wire effects begin to dominate performance in DSM technologies, the overhead of pipelining as opposed to buffering will dwindle. Completion detection was employed at each stage to generate acknowledge signals which were then used to control the precharging and evaluation of the dynamic nodes. The result was a very high throughput of up to 1.2 GDI/s (giga data items per second) for single rail designs, in a 0.6 μmCMOS technology. A congestion signal traveling backwards through the channel compresses the data in the channel, storing it in the latches until the congestion is resolved. Thus a back pressure flow control scheme was employed without the cost of full pipeline elements.

Low swing drivers. In an RC charging system, the power consumption is proportional to the voltage shift squared. One way of lowering the power consumption for long on-chip interconnects is by applying low-swing signaling techniques which are also widely used for off-chip communication lines. Basically the power usage is lowered at the cost of the noise margin. However, a differential transmission line (2 wires), on which the voltage swing is half that of a given single-ended transmission line, has differential mode noise characteristics comparable to the single-ended version. This is so because the voltage difference between the two wires is the same as that between the single-ended wire and a mid-point between supply and ground. As an approximation, it uses only
half the power, however, since the two wires working at half the swing each consume one-fourth the power. The common mode noise immunity of the differential version is also greatly improved, and it is thus less sensitive to crosstalk and ground bounces, important sources of noise in on-chip environments as discussed in the reliability section that follow.

3.3.3. Reliability.

Designing global interconnects in DSM technologies, a number of communication reliability issues become relevant. Noise sources which can have an influence on this are mainly crosstalk, power supply noise such as ground bounce, electromagnetic interference (EMI), and intersymbol interference. Crosstalk is becoming a serious issue due to decreasing supply voltage, increasing wire to wire capacitance, increasing wire inductance (e.g., in power supply lines), and increasing rise times of signaling wires. The wire length at which the peak crosstalk voltage is 10% of the supply voltage decreases drastically with technology scaling and, since the length of global interconnects does not scale with technology scaling, this issue is especially relevant to the implementation of

NoC links. Power supply noise is worsened by the inductance in the package bonding wires, and the insufficient capacitance in the on-chip power grid. The effect of EMI is worsening as the electric charges moved by each operation in the circuit is getting smaller, making it more susceptible to external influence. Intersymbol interference, that is, the interference of one symbol on the following symbol on the same wire, is increasing as circuit speeds go up. *3.3.4. Encoding.* Using encoding for on-chip communication has been proposed; the
most common objective is to reduce power usage per communicated bit, while maintaining high speed and good noise margin.

4 NETWORK ANALYSIS

The most interesting and universally applicable parameters of NoC are *latency*, *bandwidth*, *jitter*, *power consumption*, and *area usage*. Latency, bandwidth and jitter can be classified as performance parameters, while power consumption and area usage are the cost factors. In this section, we will discuss the analysis and presentation of results in relation to these parameters.

4.1. Performance Parameters and Benchmarks

Specifying a single one of the performance parameters previously introduced is not sufficient to confer a properly constrained NoC behavior. The following example illustrates this. Given a network during normal operation, it is assumed that the network is not overloaded. For such a network, all data is guaranteed to reach its destination when

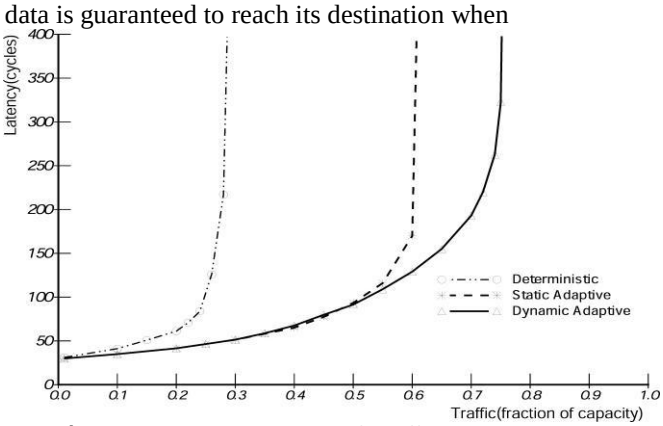

Fig. 7. Latency vs. network load for different routing schemes

employing a routing scheme in which no data is dropped (see Section 3.2.2, delay routing model). This means that, as long as the capacity of the network is not exceeded, any transmission is guaranteed to succeed (any required bandwidth is guaranteed). However, nothing is stated concerning the transmission latency whichmay well be very high in a network operated near full capacity. As shown in Figure 7, the exact meaning of which will be explained later, the latency of packets rise in an exponential manner as the *network load* increases. The exact nature of the network load will be detailed later in this section. It is obvious that such guarantees are not practically usable. We observe that the bandwidth specification is worthless without a bound on the latency as well. This might also be presented in terms of a maximum time window within which the specified bandwidth would always be reached, that is, the jitter of the data stream (the spread of the latencies). Jitter is often a more interesting parameter in relation to bandwidth than latency as it describes the temporal evenness of the data stream. Likewise, a guaranteed bound on latency might be irrelevant if the bandwidth supported at this latency is insufficient. Thus latency, bandwidth, and jitter are closely related. Strictly speaking, one should not be specified without at least one of the others. At a higher abstraction level, performance parameters used in evaluating multicomputer networks in general have been adopted by NoC researchers. These include *aggregated bandwidth*, *bisection bandwidth*, *link utilization*, *network load*, etc. The aggregate bandwidth is the accumulated bandwidth of all links, and the bisection bandwidth is the minimum collective bandwidth across links that, when cut, separate the network into two equal set of nodes. Link utilization is the load on the link compared with the total bandwidth available. The network load can be measured as a fraction of the network capacity, as normalized bandwidth. The network capacity is the maximum capacity of the network for a uniform traffic distribution, assuming that the most heavily loaded links are located in the network bisection. For highly complex systems, such as full-fledged computer systems including processor(s), memory, and peripherals, the individual parameters may say little about the overall functionality and performance of the system. In such cases, it is customary to make use of benchmarks. NoC-based systems represents such complexity, and benchmarks would be natural to use in its evaluation. Presenting performance in the form of benchmark results would help clarify the effect of implemented features in terms of both performance benefits (latency, jitter, and bandwidth) and implementation and operation costs (area usage and power consumption). Benchmarks would thus provide a uniform plane of reference from which to evaluate different NoC architectures. At present, no benchmark system exists explicitly for NoC, but its development is an exciting prospect. This is a set of benchmarks that has been developed for the performance evaluation of highly parallel supercomputers which mimic the computation and data movement characteristics of large scale computational fluid dynamics applications. It is questionable, however, how such parallel computer benchmarks can be used in NoC as the applications in SoCs are very different. In particular, SoC applications are generally highly heterogeneous, and the traffic patterns therein likewise.

4.2. Presenting Results

Generally it is necessary to simplify the multidimensional performance space. One common approach is to adjust a single aspect of the design, while tracking the effect on the performance parameters. An example is tracking the latency of packets, while adjusting the bandwidth capabilities of a certain link within the network, or the amount of background traffic generated by the test environment. In Section 5.2.1, we will give specific examples of simple yet informative ways of communicating results of NoC performance measurements. Since the NoC is a shared, segmented communication structure wherein many individual data transfer sessions can take place in parallel, the performance measurements there in, not only on the traffic being measured therein, but also on the other traffic in the network, the *background traffic*. The degree of background traffic is often indicated by the network load as described earlier. Though very simple, this definition makes valuable sense in considering a homogeneous, uniformly loaded network. One generally applicable practical method for performance evaluation is thus generating a uniform randomly-distributed background traffic so that the network load reaches a specified point. Test packets can then be sent from one node to another, according to the situation that one desires to investigate, and the latencies of these packets can be recorded (see example (i) in Section 5.2.1). Evenly distributed traffic, however, may cloud important issues of the network performance. The adaptive protocol resulted in a significant improvement of throughput over the deterministic one for nonuniform traffic but had little effect on performance with uniformly distributed traffic. The reason for this is that the effect of adaptive protocols is to even out the load to avoid hotspots, thus making better use of the available network resources. If the bulk load is already evenly distributed, there is no advantage. Also traffic parameters, like number of packets and packet size, can have a great influence on performance, for example, in relation to queueing strategies in nodes. There are manyways to approach the task of presenting test results. The performance space is a complex, multidimensional one, and there are many pitfalls to be avoided in order to display intelligible and valuable information about the performance of a network.

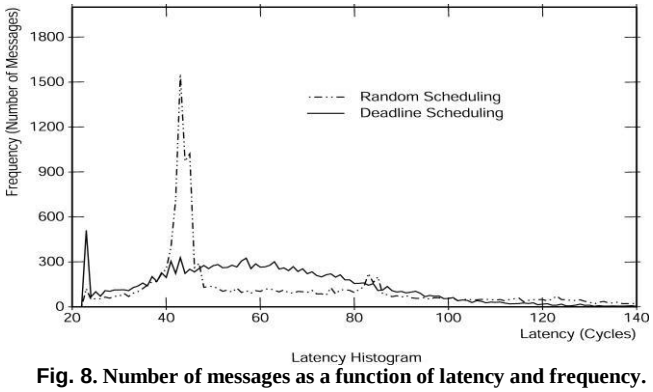

Fig. 8. Number of messages as a function of latency and frequency.

Often the presented results fail to show the interesting aspects of the network. It is easy to get lost in the multitude of possible combinations of test parameters. This may lead to clouding (or at worst failure to properly communicate), the relevant aspects of the research. Though the basis for performance evaluation may vary greatly, it is important for researchers to be clear about the evaluation conditions, allowing others to readily and intuitively grasp the potential of a newly developed idea and the value of its usage in NoC. *5.2.1. Examples.* We will now give some specific examples that we find clearly communicate the performance of the networks being analyzed. What makes these examples good are their simplicity in providing a clear picture of some very fundamental properties of the involved designs. (i) *Average latency vs. network load.* Figure 7 is a figure from the article, showing how the average latency of the test data grows exponentially as the background traffic load of the network is increased. In the presented case, the *throughput saturation point*, the point at which the latency curve bends sharply upwards, is shifted right as more complex routing schemes are applied. This corresponds to a better utilization of available routing resources. The article does not address the question of cost factors of the implementation.

(ii) *Frequency of occurrence vs. latency of packet.* Displaying the average latency of packets in the network may work well for establishing a qualitative notion of network performance. Where more detail is needed, a histogram or similar graph showing the distribution of latencies across the delay spectrum is often used with great effect. Figure 8, taken from the article, shows the effect of *random scheduling* and *deadline scheduling*. Random scheduling schedules the packets for transmission in a random fashion, while deadline scheduling prioritize packets according to how long they have been waiting (oldest-packet-first). It is shown how the choice of scheduling affects the distribution of latencies of messages.

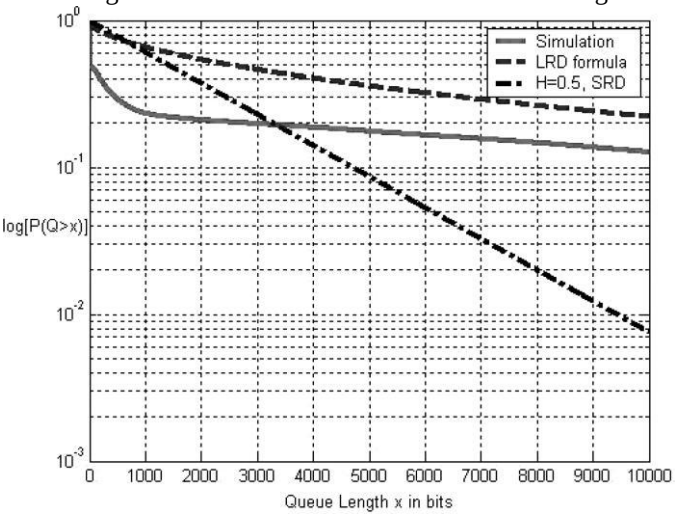

Fig. 9. The probability of queue length exceeding buffer size.

such a latency distribution graph is also used to display how a scheduling scheme provides hard latency bounds in that the graph is completely empty beyond a certain latency. (iii) *Jitter vs. network load.* The jitter of a sequence of packets is important when dimensioning buffers in the network nodes. High jitter (bursty traffic) needs large buffers to compensate in order to avoid congestion resulting in suboptimal utilization of routing resources. This issue is especially relevant in multimedia application systems with large continuous streams of data. In this work, statistical mathematical methods are used to analyze the traffic distribution. Figure 9, taken from the article, explores the use of two different models based on stochastic processes for predicting the probability that the queue length needed to avoid congestion exceeds the actual buffer size in the given situation. The models displayed in the figure are LRD (Long Range Dependent) or *self-similar*, and SRD (Short Range Dependent) or *Markovian* stochastic processes. In the figure, these models are compared with simulation results. The contributions of the paper include showing that LRD processes can be used effectively to model the bursty traffic behavior at chip level, and the figure shows how indeed the predictions of the LRD model comes closer to the simulation results than those of the SRD model.

4.3. Cost Factors

The cost factors are basically power consumption and area usage. A comparative analysis of cost of NoC is difficult to make. As is the case for performance evaluation, no common ground for comparison exists. This would require different NoC being demonstrated for the same application which is most often not the case. Hence a somewhat broad discussion of cost in terms of area and power cost is presented in this section. The power consumption of the communication structure in large single-chip systems is a major concern, especially for mobile applications. As discussed earlier, the power used for global communication does not scale with technology scaling, leading to increased power use by communication relative to power use by processing. In calculating the power consumption of a system, there are two main terms: (i) power per communicated bit and (ii) idle power. Depending on the traffic characteristics in the network, different implementation styles will be more beneficial with regards to power usage. The effects on power consumption of scaling a design were seen, and a bus design was compared with torus connected grid design (both synchronous and asynchronous implementations). Asynchronous implementation styles (discussed in Section 3.3.1), are beneficial for low network usage since they have very limited power consumption when idle, but use more power per communicated bit due to local control overhead. Technology scaling, however, leads to increased leakage current, resulting in an increasing static power use in transistors. Thus the benefit of low idle power in asynchronous circuits may dwindle. From a system-level perspective, knowledge of network traffic can be used to control the power use of the cores. Interest has been expressed in investigating centralized versus distributed power management (DPM) schemes. Centralized power managers (PM) are a legacy in bus-based systems. Since NoC is most often characterized by distributed routing control, naturally distributed PMs. In both of these studies, conceptually there is a node-centric and network-

centric PM. The node-centric PM controls the powering up or down of the core. The network-centric PM is used for overall load-balancing and to provide some estimations to the node-centric PM of incoming requests, thus masking the core's wake-up cost by precognition of traffic. This type of power management is expected to be favored to reduce power consumption in future NoCs. The results, show that, with only node PM, the power saving range from a factor of 1.5 to 3 compare to no power managers. Combining dynamic voltage scaling withDPMgives overall saving of a factor of 3.6. The combined implementation of node and network-centric management approaches shows energy savings of a factor of 4.1 with the performance penalty reduced by a minimum 15% compared to node-only PM. For experiments done on 2D mesh with minimal path routing, energy savings of 44% are reported when executing complex multimedia benchmarks. A design constraint of NoC less applicable to traditional multicomputer networks lies in the area usage. A NoC is generally required to take up less than 5% of the total chip area. For a 0.13 μm SoC with one network node per core and an average core size of 2×2 mm (approximately 100 cores on a large chip), this corresponds to 0.2 mm2 per node. One must also remember that the NA will use some area, depending on the complexity of the features that it provides. Trade-off decisions which are applicable to chip design in general and not particular to NoC are beyond the scope of this survey. At the network level, many researchers have concluded that buffering accounts for the major portion of the node area, hence wormhole routing has been a very popular choice in NoCs (see Section 3.2.2). As examples of an area issue related to global wires, introducing *fat wires*, that is, the usage of wide and tall top-level metal wires for global routing, may improve the power figures but at the expense of area.

CONCLUSION

NoC encompasses a wide spectrum of research, ranging from highly abstract software related issues, across system topology to physical level implementation. In this survey, we have given an overview of activities in the field. We have first stated the motivation for NoC and given an introduction of the basic concepts. In order to avoid the wide range of topics elevant to large scale IC design in general, we have assumed a view of NoC as a subset of SoC. From a system-level perspective, NoC is motivated by the demand for a well structured design approach in large scale SoCs. A modularized design methodology is needed in order to make efficient use of the ever increasing availability of on-chip resources in terms of the number of transistors and wiring layers. The main job of the NoC designer of the future will be to dimension and structure the network according to the communication needs of the SoC. At present, an interesting challenge lies in specifying ways to define these needs.

REFERENCES

AGARWAL, A. 1999. The Oxygen project—
Raw computation. *Scientific American*, 44–47.

AGGARWAL, A. AND FRANKLIN, M. 2002. Hierarchical interconnects for on-chip clustering. In *Proceedings of the 6th International Parallel and Distributed Processing Symposium (IPDPS)*. IEEE Computer Society, 602–609.

BAINBRIDGE, J. AND FURBER, S. 2002. CHAIN: A delay insensitive chip area interconnect. *IEEE Micro 22*, 5 (Oct.) 16–23.

CATTHOOR, F., CUOMO, A., MARTIN, G., GROENEVELD, P., RUDY, L., MAEX, K., DE STEEG, P. V., AND WILSON, R. 2004.

CHAPIRO, D. 1984. Globally-asynchronous locally-synchronous systems. Ph.D. thesis (Report No. STANCS-84-1026) Stanford University.

DALLY, W. J. 1990. Performance analysis of k-ary n-cube interconnection networks. *IEEE Trans. Comput. 39*, 6 (June) 775–785.

DALLY, W. J. 1992. Virtual-channel flow control. *IEEE Trans. Parall. Distrib. Syst. 3*, 2 (March) 194–205. GINOSAUR, R. 2003. Fourteen ways to fool your synchrononizer. In *Proceedings of the 9th International Symposium on Asynchronous Circuits and Systems*. IEEE, 89– 96