CCIT Report #565
December 2005

Zooming in on Network-on-Chip Architectures*

Israel Cidon
Dept. of Electrical Engineering
Technion, Haifa 32000, Israel
cidon@ee.technion.ac.il

Idit Keidar
Dept. of Electrical Engineering
Technion, Haifa 32000, Israel
idish@ee.technion.ac.il

ABSTRACT

The aim of this paper is to expose the networking community to the concept of *network-on-chip (NoC)*, an emerging field of study within the VLSI realm, in which networking principles play a significant role, and new network architectures are in demand. Networking researchers will find new challenges in exploring solutions to familiar problems such as network design, routing, and quality-of-service, in unfamiliar settings under new constraints. We present a new classification of chip architectures into three categories with different requirements from their NoCs. In order to stimulate some specific research directions, we highlight research problems arising in each of these categories, focusing on routing and resource allocation (e.g., capacity assignment). We provide initial solution directions to example problems.

1. INTRODUCTION

As VLSI technology becomes smaller, and the number of modules on a chip multiplies, on-chip communication solutions are evolving in order to support the new inter-module communication demands. Traditional solutions, which were based on a combination of shared-buses and dedicated module-to-module wires, have hit their scalability limit, and are no longer adequate for sub-micron technologies [4, 12, 8, 23, 14, 15]. Current chip designs incorporate more complex multi-layered and segmented interconnection buses [1, 16, 30]. More recently, chip architects have begun employing on-chip network-like solutions [12, 13, 8, 24, 2, 17, 5]. This evolution of on-chip interconnects may evoke feelings of *déjà vu* among networking old-timers. We believe that the considerations that have driven data communication from shared buses to packet-switching networks (spatial reuse, multi-hop routing, flow and congestion control, and standard interfaces for design reuse, etc.) will inevitably drive VLSI designers to use these principles in on-chip interconnects. In other words, we can expect the future chip design to incorporate a full-fledged *network-on-a-chip (NoC)*, consisting of a collection of links and routers and a new set of protocols that govern their operation. In Section 2, we survey the reasons for the inevitable shift to NoCs in the VLSI world, while exposing the most important requirements from a NoC. We note that although some recent papers have begun to design such on-chip network architectures, the field is still in its infancy, and many challenges have yet to be tackled.

In designing a NoC, one has to address all the classical networking issues. Addressing and routing schemes need to be devised in order to allow packets traversing the same links to be routed to diverse destinations. Names meaningful to applications (such as memory and I/O addresses) need to be translated into routing efficient labels. Since the timely delivery of certain types of traffic (or signals) on the chip is crucial for performance, support for multiple quality-of-service (QoS) requirements is also essential [5, 10, 28]. Similarly, a NoC should support network level congestion control in order to accommodate excessive traffic conditions. Where congestion control is employed, fairness issues need to be considered as well. One also needs to address reliability in the face of communication soft errors that may corrupt transmitted data [28, 32]; this can be done using a combination of error-correction codes and retransmission mechanisms.

Since problems of this type have been extensively studied in the networking realm, as well as for off-chip interconnection networks [9], one may be tempted to employ well-developed networking/interconnection solutions in the NoC context. Nevertheless, a direct adaptation of network protocols to NoCs is impossible, due to the different communication requirements, cost considerations, and architectural constraints. The primarily considerations in VLSI are minimizing power dissipation and area. This has a number of implications on NoC design. First, NoC components should be extremely simple, so as to allow implementing them with a small number of logic gates and to expend as little energy as possible. In addition, power considerations render shortest-path routes highly desirable, while area considerations dictate the use of small routing tables.

Beyond the distinctive cost considerations, the requirements from a NoC also differ from their off-chip counterparts. For example, on-chip network topologies are quite restricted– they are laid onto planar layers (in silicon and/or metal), and are therefore often organized as (possibly partial) grids. Thus, elaborate layouts like high-dimension hypercubes and butterflies, which are often employed in interconnection networks [9], are not cost-effective for NoCs. Moreover, NoC topologies are fixed throughout their lifetimes. That is, they do not need to support the dynamic addition or removal of network-attached modules. Furthermore, the NoC is synthesized anew for each design [27, 5, 3], eliminating the need for standard network protocols; i.e., there is no advantage in backward compatibility of NoC protocols and architectures employed in a new chip designs with those used in previous designs, beyond the use of standard network *interfaces* to allow the reuse of modules, called *IP cores*, across

*This research is partially supported by Intel Corporation and Semiconductor Research Corporation.

chip designs. This means that the future chip designer will be able to select the best NoC architecture among a plurality of architectures offered by the chip synthesis tools, customized to his specific requirements. Another crucial aspect of on-chip network design is meeting strict QoS requirements for distinct types of in-chip traffic, such as interrupt signals or fetching instructions and data from caches to processors.

Finally, the NoC design process dramatically differs from that of classical networks, allowing a new important dimension of freedom— the ability to alter the physical layout of the network routers and links along with the chip module placements. This enables the designer to optimize the NoC geography and resource allocation according to traffic and layout constraints. Moreover, if traffic requirements are known ahead of time (as is typical for many special purpose SoCs), the NoC's topological layout and capacity allocation can be optimized to this traffic with any desired routing protocol; this eliminates the need for a high cost load-balanced routing protocol.

The aforementioned differences between the on-chip environment and classical network settings imply that NoC design requires the development of new solutions. Hence, NoC is a promising research field, which can have a large practical impact on the future of VLSI, and where networking expertise will be valuable. Unlike the Internet "impasse" described in [21], the NoC presents a fertile ground where frustrated network architects may harness their abilities to design not one, but many diverse network architectures!

In order to advance this field, there is a need for crisp definitions of models, requirements, costs, and problems. Since different chips may require different NoCs, a classification of NoC models and problems that arise in each category is also of essence. In Section 3, we identify three categories of silicon systems, and observe that each category has inherently different requirements, for which very different NoC solutions are appropriate. In order to make the discussion concrete, we formally define example research problems in all categories. In particular, we formulate routing and link capacity problems for each category, highlighting the different requirements that these problems face in different system types. We describe preliminary solution directions, which are the subject of work-in-progress.

2. WHY NOC AND WHY NOW?

The first driving force behind the transition to NoC-based solutions is the inadequacy of current-day VLSI inter-chip communication design methodology for the deep sub-micron chip manufacturing technology. Chip design in the last decade consisted of designing separate functional modules, and interconnecting them using standard shared buses as well as a "spaghetti" of inter-module wires for critical or long connections. The technological progress of the silicon industry follows Moore's Law, doubling the number of gates every 18 months by shrinking the technology dimensions proportionally. However, whereas shrinking gates reduces their intrinsic latencies and dynamic power dissipation, the same is not true for interconnection wires. In fact, the decrease in wire dimensions increases their resistance, while decreased inter-wire spacing increases their capacitance, wire coupling delays [31], and crosstalk noises. This increase in line capacitance, resistance, and inter-wire cross-talk increases the latency of signal propagation with the advance of the technology. In order to preserve wire speeds and voltage noise margins, today's long wires (including bus lines) require the periodic insertion of repeaters [20], which in turn consume more dynamic and leakage power. Magen et al. [18] show that in a modern microprocessor, 50% of the power dissipation is due to the (long) wires. These phenomena intensify with time [15, 14, 23], as Ho et al. [15] write: "global wire delays scale worse than gates, by one to three orders of magnitude." This trend favors the use of short wires and the overall reduction of the total wire length in the chip. The network-on-a-chip concept naturally embodies both ideas, by building a highly-utilized and shared interconnection network with short links connected by routers.

In addition, a NoC-based interconnect achieves better scalability than traditional solutions in terms of the number of modules on a chip. Recent analysis [4] shows that the power and area of a NoC based on a grid of routers connected by short wires scales linearly with the chip density (number of modules), whereas common solutions, namely buses, segmented buses, and point to point links, exhibit super-linear growth in both dynamic power and area. Note that in the case of a bus, the increase in the number of modules not only increases the wire length, but also introduces a major waste of energy due to the inherent broadcast nature of buses, (which is only partially mitigated by the segmented bus idea).

Another key aspect factoring in the future of VLSI is the well-known chip *design productivity gap* [25]. It is a common assumption that the productivity of the designer increases every year by 21%, while the design complexity increases by 58% (following Moore's Law). This problem has been causing the number of custom chip designs to decrease annually. This situation resembles the software productivity crisis of the eighties, which has led to a paradigm shift in software engineering, emphasizing modular design, component reuse, object orient programming, standard inter-module communication interfaces, etc. A similar trend in VLSI calls for the development of highly reusable modules and the availability of third party IP cores. In order to enable such a component-oriented design, the module interface needs to be standardized. In particular, a module's interface should be independent of the chip size and of the number of modules on the chip. NoCs naturally lend themselves to such standard interfaces, since, in contrast to buses, a larger and faster network does not imply a faster (or wider) module interface. The non-standard interfaces used in custom wiring render automatic synthesis virtually impossible, whereas the NoC paradigm allows for a fully synthesizable solution, which can be optimized automatically for each chip design.

Finally, NoC adoption is also driven by the distributed nature of modern chip architectures, which introduce on-chip peer-to-peer communication patterns. Unlike traditional chip designs, which had rigid master-slave or pipelined communication patterns, the next generation of chip designs will incorporate multiple autonomous intelligent modules with a rich collection of communication services among them. The level of parallelism in chips is on the rise. There are already commercial examples of Chip Multi Processors (CMP), e.g., IBM's Cell, Xilinx's FPGA with multiple PowerPC cores, etc. Other examples are Systems-on-Chip (SoCs) for hand-held devices, which incorporate multiple functional processors like digital RF processors, GPS, graphic accelerator, 2D/3D video accelerator, voice and video codecs, digital cameras, and a general-purpose processor running an

operating system (e.g., the TI OMAP 2420 and the Qualcomm MSM 7600 chip sets). A NoC provides support for message-passing in such chips, and also allows for spatial reuse, whereby communication among various modules can occur concurrently over a distributed collection of wires. Moreover, by having modules communicate over a network, they can be separated into different clock and voltage domains, which enables independently slowing down or even shutting down unused parts of the system in order to reduce heat dissipation and conserve battery life.

All in all, we believe that the shift to NoC-based architectures is inevitable, and is likely to become the main trend in VLSI in years to come. We further emphasize that the main requirements from a NoC are conserving power and area, which will be manifested in designs consisting of short wires interconnected by simple routers, and energy-conserving protocols. For example, the need to reduce area and power is likely to promote the use of wormhole routing [9], whereby each packet is divided into small pieces called *flits*, and flits of the same packets progress consequently from a first router to a second router as soon as a flit buffer in the second router is available. This technique enables a large reduction in the minimal required buffer space compared to standard store-and-forward packet switching. Similarly, power conservation is likely to result in employing a minimal energy routing protocol, (which may or may not be identical to shortest path routing), and area conversation dictates the use of a simple, preferably table-free routing mechanisms.

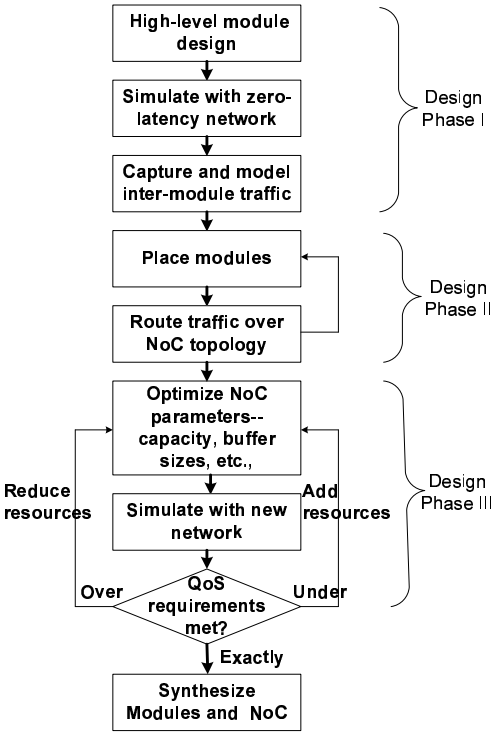


Figure 1: SoC network-on-chip design process.

3. CLASSIFYING NOC MODELS

In this section, we introduce our new classification of silicon chip designs. Chips are classified according to how much is known in advance about their functionality. As we explain below, this knowledge has a huge impact on the design of their respective NoCs.

The first category, *Systems-on-Chips (SoCs)* (cf. Section 3.1), encompasses custom-made designs for chips with a particular pre-known purpose, e.g., a multi-channel 3G base station, a video camera, or a printer/fax/scanner system. In such chips, the network usage is known a priori.

The second category, *Field-Programmable Gate Arrays (FPGAs)* (cf. Section 3.2), consists of multi-purpose chips that include generic hardware resources like logic arrays, flip-flops, RAM, processors and special purpose IP cores (Ethernet, USB, DSP core, etc.). Such chips can be *configured* using a programmable interconnect grid into specific systems. The design of such systems (and hence the design of their interconnects) leaves a large degree of freedom for the user (FPGA application designer/programmer). In this case, the NoC that is part of the FPGA infrastructure (or part of the interconnect grid) can also be configured according to the application designer's specification.

Finally, our last category, *Chip Multi Processors (CMPs)* (cf. Section 3.3), captures general-purpose chip designs supporting parallel processing, where the chip and interconnect usage is unpredictable, and only determined at run time.

3.1 Systems on Chip

The distinguishing property of the System-on-Chip type NoC is the assumed ability to accurately predict the network usage patterns before the network is laid out. Since a special purpose SoC has a specific functionality, this functionality can be accurately simulated, and the inter-module communication patterns can be inferred at design time. Traditional SoC designs synthesize direct point to point wires meeting the communication needs between pairs of modules. Similarly, a SoC's network can be synthesized specifically to satisfy these needs. Virtually all network parameters, including network layout, link capacity allocation, buffer sizes, packet headers, routing tables, and the number of service classes, can be custom-synthesized to meet the particular SoC's requirements.

Naturally, automated design tools are essential for designing the NoC with all its parameters for each SoC design. Such tools, in turn, need be based on certain design methodologies, and need to implement optimization algorithms within their framework. This is where the research challenges lie.

Bolotin et al. [5] suggest a design process for SoCs similar to the one illustrated in Figure 1, consisting of the following three phases. First, identifying the QoS requirements of the target SoC. Second, customizing the network by an appropriate module placement and a minimal gate-count cost, static, shortest path routing function. Note the feedback cycle between the placement of modules and the network design. Finally, performing network load balancing by an appropriate link bandwidth allocation so that the congestion in the network is reduced and multi-class QoS requirements of each communication flow are satisfied. The network's optimization process can be iterated until the QoS requirements are perfectly matched. We now discuss design phases two (routing) and three (resource allocation) in more detail.

3.1.1 Routing

A typical generic NoC architecture can be based on worm-

hole routing over a grid topology (possibly a partial grid), with shortest-path routing (in terms of Manhattan distance on the grid) [8]. Note that in a pre-defined topology, shortest-path routes may lead to uneven load distribution across links. In the SoC-based NoC, not only this does not pose a problem, it is even desirable for a number of reasons. First, the energy gain from routing over shortest-paths is significant. Second, traffic aggregation over a small number of high-capacity links results in lower latencies as compared to routing over multiple routes of the same overall length and capacity. A large degree of link sharing on some links can render other links and routers unnecessary, saving area and static power on the chip. Therefore, a shortest-path routing scheme that also aggregates paths is highly desirable.

In a regular mesh it is easy to accomplish low gate-count cost shortest path routing, by employing a simple variation of dimension order routing [17] such as XY. XY is a tableless routing discipline whereby each packet is routed first in an X direction and then along the perpendicular dimension. This is an attractive solution, since (i) it is shortest-path, and hence minimizes the energy spent per information unit transfer; and (ii) the usage of such a simple function dramatically reduces the number of gates as compared to routing table-based or source-route schemes. The fact that XY routing favors specific paths can be easily dealt with by moving resources from underutilized links to these preferred links (in the third design phase).

However, in the SoC environment, practical NoC topologies become irregular meshes because of modules' shape and size variability in VLSI layouts, and the need to physically separate between the modules and the NoC infrastructure. In [6], alternatives to the gate-count efficient shortest path XY routing are examined for a SoC-based NoC with an irregular mesh topology. Since all considered algorithms are static shortest path schemes, which all have equal link power dissipation, the comparative metric is the total implementation gate-count.

There are two simple routing alternatives for irregular meshes: (i) source routing (SR), where the packet carries a sequence of routing instructions; and (ii) distributing routing (DR), whereby the destination is looked up at each intermediate router. In general, both SR and DR make extensive use of routing tables (RTs). DR tables are located at each router. They are indexed by the packet destination address and contain output port values. SR tables are located at each source. They are also indexed by packet destination addresses, but they contain sequences of routing commands, one for each hop along the routing path. Naïve RT implementations have as many entries as there are nodes in the network. However, this is inefficient, since an all-to-all communication pattern is very unlikely and the actual set of destinations used at each source is typically a small fraction of the number of modules.

Nevertheless, the hardware cost of empty table entries can be avoided. More gate-efficient efficient implementations of SR and DR may use a reduced-size ROM, or simple Boolean logic implementing an equivalent routing function. In either case, only the necessary table entries for each node need be implemented. The reduced ROM implementation is equivalent to a two-level implementation of a routing function by a Programmable Logic Array (PLA).

The total gate count and the power dissipated in these tables can be estimated by the size of the tables, since the total capacitance is proportional to the number of entries and the size of an entry. Thus, the important factor to optimize in such routing schemes is the overall size of all routing tables. Formally, we define the following optimization problem:

Problem 1 (Minimal Hardware Cost Routing). *Given an irregular mesh topology, implement a shortest-path routing scheme with a minimum number of entries in all routing tables system-wide.*

Bolotin et al. [6] present two techniques to further improve the cost of both SR and DR. The XY-Deviation Table (XYDT) Routing method reduces the cost of DR, by relying on a default dimension ordering routing scheme (e.g., XY), and storing only entries that *deviate* from the default path. Each router holds a *deviation table*, so that an entry in the deviation table towards destination D exists only if the next hop from this router to D deviates from the next hop calculated by the default (XY or any other dimension ordering) routing function. We assume that packets carry the XY coordinates of the destination. When a packet enters a router, its next hop is looked up in the table. If it is found, it is routed according to the table. Otherwise, the default hardware function calculates the exit port for that packet. Note that XYDT is a generic scheme, not a single solution. In [6], specific assignments of deviation points are given, so as to minimize the total number of routing table entries.

Their second hardware-efficient routing scheme, Source Routing for Deviation-Points (SRDP), is a method for reducing the size of SR headers stored at the sources. Like XYDT, it combines a default fixed routing function (for example, XY) with lists of routing steps that may deviate from the default. SRDP designates a fixed collection of routers as *deviation points (DPs)*. These are nodes through which the direction of at least one routing path deviates from the default scheme. An SR packet header includes a list of tags, which are routing commands, one for each DP node on the traversed path. The size of an SRDP tag is two bits for a DP node that implements all ports, and less in cases when some ports are missing. Only DP routers examine these tags in order to determine how to route the packet; other routers only implement the default function. As before, SRDP is a general algorithm framework, not one specific algorithm. In [6] optimal selection of DPs is studied.

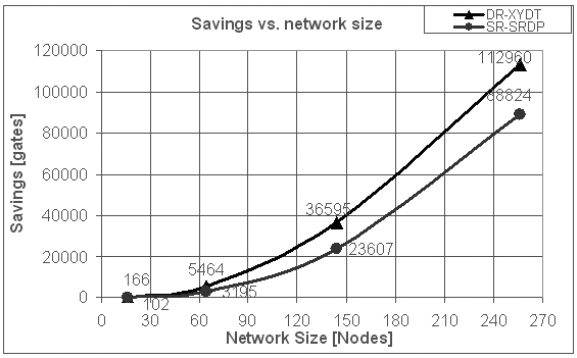


Figure 2: Scaling of cost savings with XYDT versus DR and SRDP versus SR.

Figure 2 shows the scaling of cost savings achieved by XYDT versus DR, and by SRDP versus SR. It simulates

typical NoCs with a number of nodes growing from 9 to 256. About 40% of the routers are missing in each NoC, and about 10% of the nodes are hotspots. The probability of each node to communicate with each hotspot is 0.5, and the probability to communicate with a non-hotspot node is 0.1. The curve with triangles shows the saving of XYDT versus traditional DR and the circled curve shows the saving of SRDP versus traditional SR. The graph clearly shows that the savings in routing costs grow rapidly (super-linearly) with the size of the network. In all points, the relative savings obtained by XYDT and SRDP are around 90% and 60%, respectively.

We are currently studying additional gate-efficient routing schemes for irregular meshes. Further research in this area should address not only missing nodes in the mesh topology, but also the insertion of long serial router-to-router wires, which bypass intermediate routers [19]. Such long wires can reduce the power dissipation of long distance traffic. Nevertheless, the communication cost over a wire does not increase linearly with its length. It is therefore interesting to study the SoC routing problem in the presence of such wires.

3.1.2 Capacity allocation

Having determined the traffic patterns and designed the routing scheme, we turn to the third design phase, assigning capacities to links. In this phase, the load on each link is precomputed, and each link is allocated a capacity appropriate for its load. Given a statistical characterization of the traffic, one must design the network capacity, buffer spaces, and policies, so as to meet the traffic's QoS requirements in terms of end-to-end delay (see Figure 1, Design Phase III). The QoS latency requirements can be stated statistically as well (e.g., 99% of the traffic must incur a latency of no more than 50 clock cycles). Thus, we identify *capacity allocation* as an important research problem in SoC, which is inter-related with buffer allocation.

Recently, Walter et al. [29] have tackled a NoC specific instance of the capacity allocation problem. In order to allocate the minimal amount of capacity (in terms of inter-router wires speed and width) they develop a new analysis methodology for wormhole networks under general flow distributions, variable link capacities, and multiple virtual channels. Such an analysis serves as a building block in an iterative optimization process where link capacity is gradually reduced until latency requirements are tightly met. At this point, more detailed simulations are conducted to finalize the capacity plan. The minimal number of wires for each link is selected as well as the minimal speed for a given amount of wires. Reducing the speed allows the designer to scale down the voltage to each router (a technique known as "voltage scaling") and consequently the dynamic and static power dissipation. In a typical DVD SoC design, their algorithm yields a reduction of roughly 30% in the overall network capacity as compared to an unoptimized grid with uniform capacities across all links. Many additional flavors of the problem have yet to be addressed, e.g., with different traffic shapes, different QoS requirements, and different buffer considerations.

3.2 Field-Programmable Gate Arrays

Our second category of silicon systems is FPGAs. These are multi-purpose chips whose functionality is determined when they are *configured* (or programmed) to perform a specific task. Current-day FPGAs are mainly comprised of programmable elements (gate arrays, flip-flops, RAM, etc.) with some embedded special-purpose IP cores such as processors and network interfaces, and wired internal routing.

As technology scales, the number of logic units (gate arrays, flip-flops, etc.) will render such a "flat" FPGA chip design unmanageable for programmers and optimization programs. We envision a future FPGA architecture that is organized hierarchically; that is, the chip is divided into high-level *regions*, interconnected by a NoC [11]. There are two types of regions: *configurable regions (CRs)*, comprised of general purpose programmable elements, and *functional regions (FRs)*, each performing a pre-specified task (e.g., general purpose processor, Ethernet interface, PCI interface, USB interface, DSP core, etc.). A configurable region can be organized like a current-day FPGA, with wired internal routing. Each CR may have several access points to the network, which can be configured to work together or apart, thus allowing multiple network-attached components to be programmed in a single CR. Regions are connected to the network using *Configurable Network Interfaces (CNIs)*. An example of such a high-level architecture is illustrated in Figure 3.

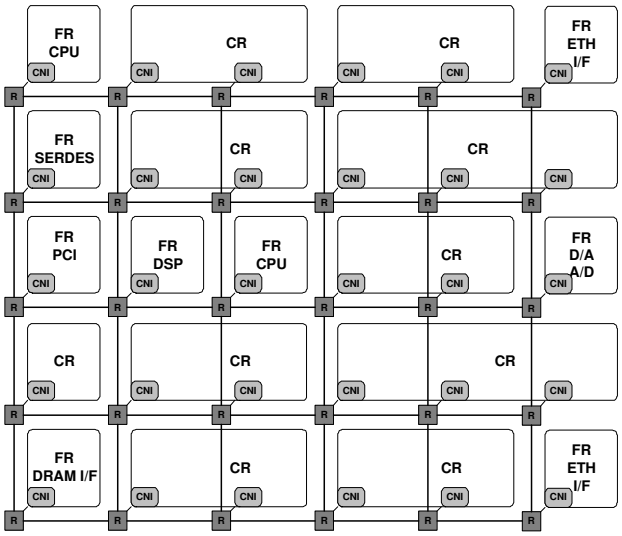


Figure 3: Example FPGA NoC architecture.

An FPGA's NoC is designed in two phases: (i) the *layout phase*, which occurs when the FPGA chip is designed, and (ii) the *configuration phase*, occurring when a specific system design is programmed into the FPGA interconnect grid. The core design made in the former should be flexible enough so as to allow for a large variety of configurations during the latter. Unlike in the SoC model, little is known about traffic patterns when the network is laid out, and hence, custom optimizations are impossible. In the configuration phase, the traffic patterns become available, but the physical wires are already in place. Since little is known about traffic requirements in advance, it is reasonable to design the NoC as a *uniform communication grid*, where all link capacities are identical.

The interplay between the two design phases defines the degree of freedom that the layout phase allows. Consider, for example, router logic. At one extreme, routers, includ-

ing routing tables, can be fully embedded in silicon during the layout phase. This offers the most compact and energy-efficient general-purpose router design, but may result in excess logic, as some routers and table entries may be redundant for certain traffic patterns. In order to reduce this waste of resources, it is possible to decouple routing tables from the core router logic, and implement them from standard configurable memory. It is further possible for *all* router logic to be "soft", i.e., implemented from general-purpose logic gates at the configuration phase, allowing for custom networks as in a SoC, but at a larger per-router cost in performance, power, and area.

Our architecture [11] employs a combination of hard and soft logic. The network infrastructure, including metal wires and hard-coded routers is laid out in silicon. In order to allow for maximum flexibility, the NoC infrastructure accommodates multiple routing schemes and a large variety of traffic patterns. To this end, we allow network interfaces to be soft. Simple routing schemes, like XY, can employ small interfaces, whereas more elaborate source-routing schemes may have the interfaces store large routing tables.

Let us now revisit the capacity allocation problem in the context of FPGA. Despite the impossibility of predicting the exact network usage, the layout phase can leverage knowledge about typical communication patterns. For example, by studying many configurations, one can observe that the most stringent bottleneck issues occur when heavy traffic is directed to one or more hotspots (high demand regions) on the chip. Thus, the main priority of the layout phase becomes providing for high throughput to a few hotspots. More formally, a collection of hotspot locations (given by their coordinates on the network grid), a hotspot-to-module flow F, and a routing policy, jointly induce a *communication pattern* where the flow from each module to each hotspot is F, and this flow traverses the links chosen by the routing algorithm. Note that if routing is symmetric, then F can be seen as the bidirectional flow between a module and a hotspot. Thus, the hotspot model is very general, and captures various common communication patterns. For example, a hotspot can be an interface to external communication or memory, a master module that communicates with a number of slaves, a dispatcher that forwards requests from multiple masters to multiple slaves [22], etc. We therefore focus on traffic patterns involving one, two, or three hotspots.

Matters are complicated by the fact that the number of hotspots and their locations on the chip can vary at configuration time. Hence, the layout should provide a *design envelope* [11] covering multiple designs. For example, the design envelope for all possible locations of k hotspots, is a capacity assignment that exceeds the communication pattern for every given choice of k hotspots.

An FPGA NoC architecture need not necessarily cater for all possible hotspot locations. Rather, the design may restrict the placement freedom of the configuration phase, and may restrict potential locations for hotspots. Consider, for example, the case of a single hotspot. One hotspot with a given throughput can be served at minimum cost if placed in the middle of the chip. However, given that off-chip communication (e.g., to memory) is often a bottleneck, the layout should also allow the hotspot to be located near one end of the chip. Hence, one can plan an architecture for the design envelope where one hotspot can be placed either in the middle of the chip or in the middle of one of the edges. In general, given a collection of restricted hotspot locations for k hotspots, the routing algorithm and capacity allocation should then accommodate the desired design envelope. Formally, we consider the following problem [11]:

Problem 2 (Uniform allocation for envelope). *Given a network grid, a number k of hotspots, a set of constraints on allowed hotspot locations, and a hotspot-to-module flow F, find a link capacity C and a routing scheme, so that for every choice of k different hotspot locations satisfying the constraints, C exceeds the flow routed by the induced communication pattern on each link.*

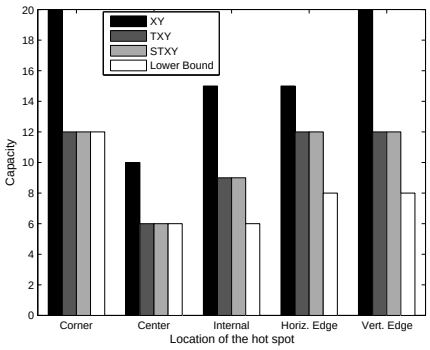
Obviously, the required capacity is tightly coupled with the routing scheme employed. Note that since we consider uniform grids, XY routing is no longer appropriate, since it causes a significant imbalance among links. Yet, similar to XY, the chosen routing algorithm should achieve shortest-path routing and low implementation cost. To this end, we examine the *toggle XY algorithm (TXY)* [11, 26], in which every router alternates between the XY and Y-X routing schemes. The only addition to the simple XY router is a single packet-header bit, indicating which scheme should be used to route the arriving packet, which is flipped at the source each time a packet is sent. This approach reduces the required capacity by up to 40% in some cases.

However, one problem with this approach is that sending packets from the same source via multiple routes may cause packets to arrive out-of-order, which requires buffering and re-ordering at the receiving end. Since such buffering involves increased hardware and power costs, we propose the *source-toggle XY algorithm (STXY)*, which uses XY or YX based on the position of the packet's source (similar to a chess board), rather than on a per-packet basis. Thus, all packets from a given source to a given destination traverse the same path, and no re-ordering is required.

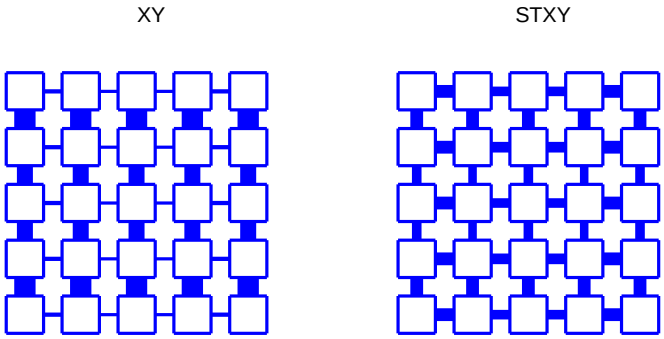
We turn to observe the design envelopes satisfying some example constraints under the different routing schemes. Let us first examine the case of one hotspot. We normalize the graphs for $F = 1$ unit. Figure 4(a) shows the required per-link capacity (in a uniform grid) for the three routing algorithms, and five different constrained locations on a 25 module (5x5) grid. These capacities are contrasted with a theoretical lower bound on the link capacity required to accommodate the envelopes. We see that TXY and STXY achieve virtually identical results, and both are significantly better than XY. Since STXY eliminates the need for buffering, it is therefore the preferred scheme. We further observe that the minimum required capacity is achieved when the hotspot is located in the center.

We turn to examine how much overall capacity is wasted by the fact that we chose a uniform-capacity grid. Figure 4(b) illustrates, on a 5x5 grid, the per-link capacity required to accommodate an envelope of all possible locations for a single hotspot, with XY and STXY routing. We see that whereas with XY routing, much higher capacity is required for vertical links than for horizontal ones, the capacity requirements of STXY are almost uniform, and hence, one does not waste resources by allocating a uniform network grid.

Figure 5 examines how the required capacity of accommodating a single hotspot scales with the grid size. We see that TXY and STXY scale much better than XY.



(a) Maximum link capacity.



(b) Envelope of all possible hotspot locations.

Figure 4: Required capacity for accommodating one hotspot on a 5x5 grid.

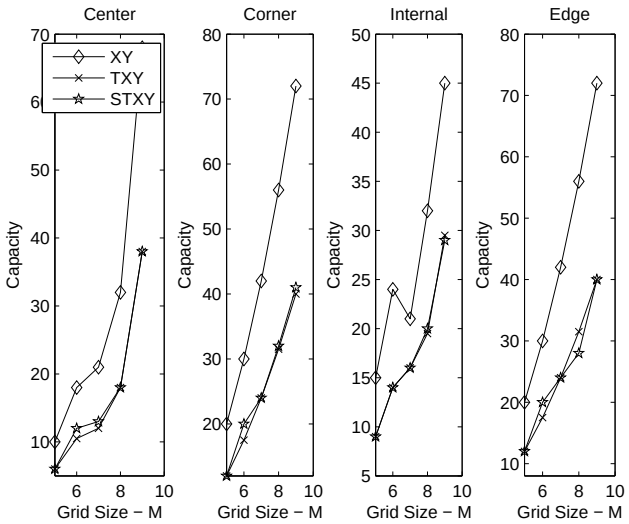


Figure 5: Scalability of capacity requirements for one hotspot at various locations.

In Figure 6, we examine the required capacity for accommodating multiple hotspots. We observe that the most important factor impacting the capacity requirements is the proximity of the hotspots to one another: If the hotspots are close together, then a large amount of traffic will need to reach their area. With two hotspots, the only significant drop in required capacity occurs when we rule out adjacent locations. Again, we see that STXY and TXY are superior to XY routing.

We are continuing our investigation of FPGA architectures, further optimizing routing and capacity allocation for multiple hotspots. We are also conducting a formal analysis of the requirements and proposed solutions. Future work should address network layouts that are not complete grids (as in Section 3.1.1 above), as well as richer communication patterns.

3.3 Chip Multi Processors

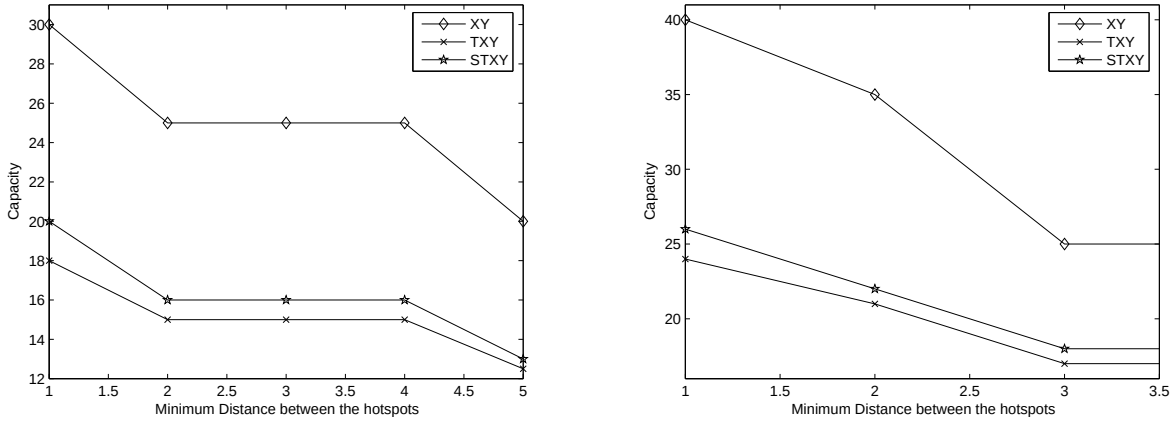
In a general-purpose CMP, the traffic pattern is completely unpredictable until run-time. In this regard, many of the challenges resemble ones we normally see in traditional networks or interconnection networks, e.g., static versus dynamic routing, congestion control, connection rate fairness, etc. Unfortunately, traditional mechanisms for dealing with these issues may be prohibitively expensive to implement in silicon. On the other hand, the relatively small dimension of the complete network, combined with the fact that the entire chip is often controlled by a single operating system, makes the problems amenable to centralized solutions.

But before developing such mechanisms, we need to design the network layout for the chip. Consider a CMP chip organized as a grid of computation units, memories, and other devices. Since all processing units are peers, and play the same role, we can expect a symmetric communication pattern among the peers. Although we cannot accurately predict the traffic pattern, our best bet for a CMP is to plan for a uniform one. That is, we shall assume that every module communicates with every other model at the same maximum rate F. We thus need to design a network layout accommodating a rate of F between every pair of modules.

Unlike in typical SoC designs, this all-to-all traffic pattern includes a substantial number of long-distance flows. Having all the long-distance traffic traverse all intermediate routers will be quite costly. In this situation, we would therefore like to bypass some routers en route to remote destinations [7]. That is, we can designate wires that traverse a given router, and others that bypass it. On the one hand, traversing routers allows multiple modules to share wires (spatial reuse). On the other hand, bypassing routers reduces the latency, energy consumption [20], and router gate costs. This tradeoff can be captured using a cost function, whereby the overall cost of a network consists of the total wire length times a constant, α, plus the number of router connections times a constant, β. This defines the following interesting cost optimization problem [7]:

Problem 3 (Router bypass layout). *Given a grid, wires with a fixed capacity C, and known traffic flow requirements $F_{i,j}$ among all pairs of modules i, j on the grid, lay out wires of capacity C between pairs of routers so as to minimize the cost function.*

An algorithmic solution to this problem will attempt to "fill" all wires to their maximum capacity, C, while inducing a minimal number of connections. For example, consider the



(a) Two hotspots.

(b) Three hotspots.

Figure 6: Required capacity for accommodating multiple hotspots.

case of symmetric flows, $F = 1$, among all pairs of modules. One can logically organize modules into clusters of size $\sqrt{C}$, and connect each pair of clusters using a direct link, i.e., a wire that does not traverse any routers. Since each module in one cluster needs to communicate with each module in the other, a single wire of capacity C exactly accommodates the communication requirements between the two clusters. Within each cluster, all modules are directly connected to a central point (module) via which all inter-cluster traffic is routed. We call this layout *Stars-Mesh*, since each cluster is organized as a star topology, and the entire chip is a fully-connected mesh of such stars. The division into clusters is based on proximity in the grid, e.g., by dividing the grid into squares of the appropriate size. Such an example layout on a 4x4 mesh with clusters of four modules is depicted in Figure 7.

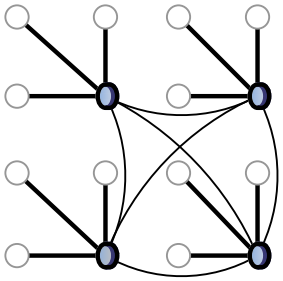


Figure 7: Stars-Mesh NoC layout for CMP with symmetric flows, 2x2 clusters.

We show that the clustering solution is asymptotically optimal for symmetric flows under the considered cost function [7]. More specifically, we first prove the following lower bound on network cost for an m by n grid:

$$\alpha \cdot \frac{m^3n^2+m^2n^3-m^2n-mn^2}{3C} \quad + \quad \beta \cdot \max\left(mn-1, \frac{mn(mn-1)}{C}\right)$$

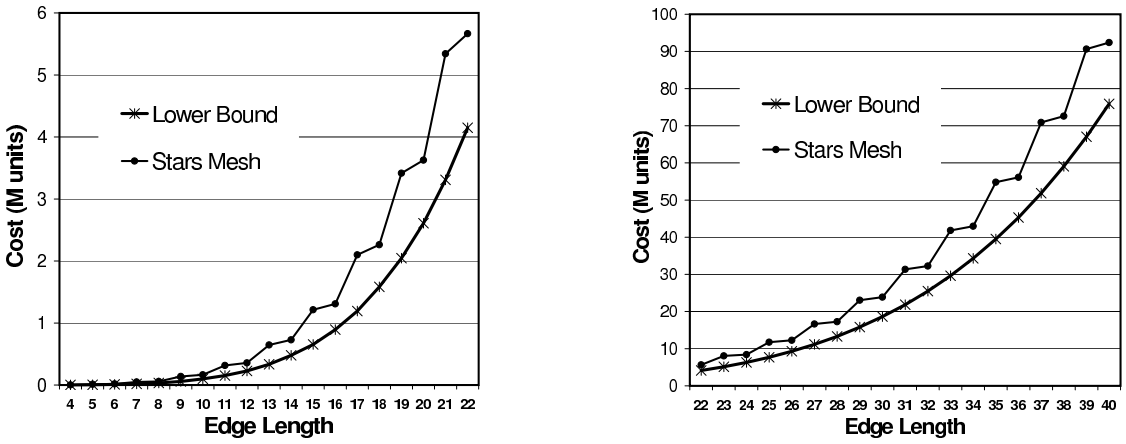
Note that, asymptotically, the dominant factor is the overall wire length, which is multiplied by α. Since the number of modules is $N = nm$, we get an asymptotic growth of $O(N^{2.5})$ in the cost. Second, we prove the following upper bound on the cost of the clustering algorithm described above, showing that the clustering algorithm is asymptotically optimal:

$$\alpha \cdot \frac{m^3n^2 + m^2n^3}{3C} + \mathrm{O}\left(m^2n^2 + m^3n + mn^3\right)$$

In Figure 8 we see the actual cost of the clusters algorithm for increasing grid sizes. Figure 8(a) focuses on grid sizes ranging from 4x4 to 22x22, and Figure 8(b) examines larger grids, ranging from 22x22 to 40x40. As only square grids are examined, $n = m = \sqrt{N}$. We examine the case that $C = 16$, and each cluster is a 2x2 square of four modules. The figure compares the actual algorithm cost against the theoretical lower and upper bounds. It shows that beyond being asymptotically optimal, the algorithm's cost is also close to the theoretical lower bound. In case C is not a perfect square, the actual results are not as close to the lower bound, but some heuristics can be used to improve its actual cost. We are currently developing such heuristics and optimizations.

Note that although the clustering solution addresses symmetric flows, Problem 3 is more generally stated for any specified flows, not necessarily symmetric ones. An interesting extension of this work is to find adequate solutions for various non-symmetric flow requirements. One interesting example to consider is a somewhat similar situation, where most modules communicate symmetrically with each other, but in addition, there is a hotspot (such as access to external memory) to which there is heavier traffic from all modules.

Unfortunately, the theoretical study above shows that a chip-wide all-to-all communication pattern does not scale well as the number of modules grows. While specific optimizations can yield efficient solutions for small to medium sized chips, ultimately, they will not suffice. Moreover, the main conclusion from the asymptotic study above is that long (inter-cluster) wires dominate the cost. Hence, for large chips, chip designs that encourage more local traffic and less long distance traffic are essential. We believe that the approach to designing such chips should be hierarchical, whereby the chip is divided into regions, each including



(a) Small grids (b) Large grids

Figure 8: Actual cost achieved by clustering algorithm versus theoretical bounds, $C = 16$, $\alpha = C$, $\beta = 3C$.

processing units and memory modules. The operating system must be aware of these regions, so that related threads can be scheduled to run on processors residing in the same region. In this scenario, there is very little inter-region communication, and the all-to-all communication problem as above needs to be solved only within each confined region.

4. CONCLUSIONS

One aim of this paper has been to introduce the network-on-a-chip concept to the networking community. We have argued that the future of VLSI will most likely focus on NoC-based system architectures, driven by considerations like scalability, energy-efficiency, and design productivity. Today's VLSI designers are approaching the NoC tentatively, repeating incremental steps similar to the evolution steps of interconnected networks – from point to point links to a collection of shared buses, to segmented buses, and to full-fledged networks. We believe that networking experts will be able to contribute to this evolution. We have emphasized the major differences between NoCs and off-chip interconnection networks, which render traditional network solutions not directly applicable to NoCs.

Our second aim has been to make the field more accessible for networking researchers, by highlighting specific problems that need to be addressed. To this end, we have classified potential NoC-based systems into three categories– SoCs, FPGAs, and CMPs– and have outlined the architectural characteristics and requirements of each. We have illustrated the unique characteristics of each model by examining common problems, namely routing and resource allocation, in all models. In this context, we have described initial solution ideas, and listed some specific future research problems that ought to be tackled.

5. ACKNOWLEDGMENTS

The authors thank Evgeny Bolotin, Roman Gindin, Ran Ginosar, Zvika Guz, Avinoam Kolodny, Ehud Shavit, and Isask'har Walter for helpful discussions and for many of the initial results presented here with their permission.

6. REFERENCES

[1] Arm inc. AMBA specification revision 2.0, May 1999.

[2] L. Benini and G. De Micheli. Networks on chips: A new SoC paradigm. *IEEE Computer*, 35(1):70–80, 2002.

[3] D. Bertozzi, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. De Micheli. NoC synthesis flow for customized domain specific multiprocessor systems-on-chip. *IEEE Trans. on Parallel and Dist. Systems,*, 16(2):113–129, 2005.

[4] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny. Cost considerations in network on chip. *Integration, Special Issue on NoC*, 38(1):19–42, Oct. 2004.

[5] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny. QNoC: QoS architecture and design process for network on chip. *J. Systems Architecture*, 50(2–3):105–128, Feb. 2004.

[6] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny. Routing in irregular meshes. TR CCIT 554, Department of Electrical Engineering, Technion, Sept. 2005.

[7] I. Cidon, I. Keidar, and E. Shavit. NoC Layout for CMPs. In preparation, 2005.

[8] W. J. Dally and B. Towles. Route packets, not wires: on-chip interconnection networks. In *DAC*, pages 684–689, 2001.

[9] W. J. Dally and B. Towles. *Principles and Practices of Interconnection Networks*. Morgan Kaufmann, 2004.

[10] T. Felicijan and S. B. Furber. An asynchronous on-chip network router with quality-of-service (QoS) support. In *SOCC*, Sept. 2004.

[11] R. Gindin, I. Cidon, and I. Keidar. NoC Architecture for Future FPGAs. In preparation, 2005.

[12] P. Guerrier and A. Greiner. A generic architecture for on-chip packet-switched interconnections. In *DATE*, pages 250–256, Mar. 2000.

[13] A. Hemani, A. Jantsch, S. Kumar, A. Postula, J. Oberg, M. Millberg, and D. Lindqvist. Network on a chip: An architecture for billion transistor era. In *IEEE NorChip*, 2000.

[14] R. Ho, K. Mai, and M. Horowitz. The future of wires. *Proceedings of the IEEE*, 89(4):490–504, Apr. 2001.

[15] R. Ho, K. Mai, and M. Horowitz. Managing wire scaling: A circuit perspective. In *IEEE Interconnect Technology Conference*, June 2003.

[16] IBM. The coreconnect bus architecture, 1999.

[17] S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A. Hemani. A network on chip architecture and design methodology. In *ISVLSI*, 2002.

[18] N. Magen, A. Kolodny, U. Weiser, and N. Shamir. Interconnect-power dissipation in a microprocessor. In *SLIP'04*, Feb. 2004.

[19] A. Morgenshtein, I. Cidon, A. Kolodny, and R. Ginosar. Comparative analysis of serial vs. parallel lings in NoC. In *International Symposium on System-on-Chip*, pages 185–188, Nov. 2004.

[20] A. Morgenshtein, I. Cidon, A. Kolodny, and R. Ginosar. Low-leakage repeaters for NoC interconnects. In *ISCAS*, pages 600–603, May 2005.

[21] L. Peterson, S. Shenker, and J. Turner. Overcoming the Internet impasse through virtualization. In *HotNets III*, Nov. 2004.

[22] A. Radulescu and K. Goossens. *Communication services for networks on chip*, pages 193–213. Marcel Dekker, 2004.

[23] V. Raghunathany, M. B. Srivastavay, and R. K. Guptaz. A survey of techniques for energy efficient on-chip communication. In *DAC*, pages 900–905, 2003.

[24] E. Rijpkema, K. Goossens, and P. Wielage. A router architecture for networks on silicon. In *Progress 2001, 2nd Workshop on Embedded Systems*, Veldhoven, the Netherlands, Oct. 2001.

[25] *ITRS report*. Semiconductor Industry Association, 1999.

[26] D. Seo, A. Ali, W.-T. Lim, N. Rafique, and M. Thottethodi. Near-optimal worst-case throughput routing for two-dimensional mesh networks. In *ISCA*, 2005.

[27] S. Stergiou, F. Angiolini, S. Carta, L. Raffo, D. Bertozzi, and G. De Micheli. Xpipes Lite: A synthesis oriented design library for networks on chips. In *DATE*, 2005.

[28] P. Vellanki, N. Banerjee, and K. S. Chatha. Quality-of-service and error control technfiques for mesh-based network-on-chip architectures. *Integration*, 38:353–382, 2005.

[29] I. Walter, Z. Guz, I. Cidon, R. Ginosar, and A. Kolodny. Efficient link capacity and QoS design for wormhole network-on-chip. In *DATE*, 2006.

[30] D. Wingard. Micronetwork-based integration of SoCs. In *DAC*, June 2001.

[31] X. Zhang. Coupling effects on wire delay - challenges in deep submicron vlsi design. *IEEE Circuits and Devices Magazine*, 12:12–18, Nov. 1996.

[32] H. Zimmer. Fault modelling and error-control coding in a network-on-chip. Masters thesis, Laboratory of Electronics and Computer Systems, KTH, Sweden, Dec. 2002.