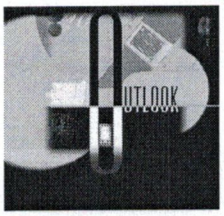
Networks on Chips: A New SoC Paradigm



On-chip micronetworks, designed with a layered methodology, will meet the distinctive challenges of providing functionally correct, reliable operation of interacting system-on-chip components.

Luca Benini
University of Bologna

Giovanni De Micheli
Stanford University

System-on-chip (SoC) designs provide integrated solutions to challenging design problems in the telecommunications, multimedia, and consumer electronics domains. Much of the progress in these fields hinges on the designers' ability to conceive complex electronic engines under strong time-to-market pressure. Success will rely on using appropriate design and process technologies, as well as on the ability to interconnect existing components—including processors, controllers, and memory arrays—reliably, in a plug-and-play fashion.

By the end of the decade, SoCs, using 50-nm transistors operating below one volt, will grow to 4 billion transistors running at 10 GHz, according to the International Technology Roadmap for Semiconductors. The major challenge designers of these systems must overcome will be to provide for functionally correct, reliable operation of the interacting components. On-chip physical interconnections will present a limiting factor for performance and, possibly, energy consumption.

Silicon technologies face other challenges. Synchronization of future chips with a single clock source and negligible skew will be extremely difficult, if not impossible. The most likely synchronization paradigm for future chips—globally asynchronous and locally synchronous—involves using many different clocks. In the absence of a single timing reference, SoC chips become distributed systems on a single silicon substrate. Global control of the information traffic is unlikely to succeed because the system needs to keep track of each component's states. Thus, components will initiate data transfers autonomously, according to their needs. The global communication pattern will be fully distributed, with little or no global coordination.

As SoC complexity scales, capturing the system's functionality with fully deterministic operation models will become increasingly difficult. As global wires span multiple clock domains, synchronization failures in communicating between different domains will be rare but unavoidable events.[1] Moreover, energy and device reliability concerns will impose small logic swings and power supplies, most likely less than one volt. Electrical noise due to crosstalk, electromagnetic interference, and radiation-induced charge injection will likely produce data errors, also called upsets. Thus, transmitting digital values on wires will be inherently unreliable and nondeterministic. Other causes of nondeterminism include design components with a high level of abstraction and coarse granularity and distributed communication control.

Focusing on using probabilistic metrics such as average values or variance to quantify design objectives such as performance and power will lead to a major change in design methodologies. Overall, SoC design will be based on both deterministic and stochastic models. Creating complex SoCs requires a modular, component-based approach to both hardware and software design.

Based on the premise that interconnect technology will be the limiting factor for achieving SoCs' operational goals, we postulate that the layered design of reconfigurable micronetworks, which exploits the methods and tools used for general networks, can best achieve efficient communication on SoCs.

Wiring Delays

Projections for future silicon technologies show that chip size will scale up slightly while gate delays decrease compared to wiring delays. A simple computation shows that delays on wires that span the chip will extend longer than the clock period. This trend is a trivial consequence of the finite propagation speed of electromagnetic waves, which is $v = (0.3/\sqrt{\epsilon})$ mm per second in a homogeneous medium with relative permittivity ϵ. In 50 nm technology, the projected chip die edge will be around 22 mm, with a clock frequency of 10 GHz.

Thus, the delay for a signal traversing the chip diagonally will be approximately 100 picoseconds, or one clock period, in the ideal case that $\epsilon = 1$. A lower bound of two clock periods applies to general media with $\epsilon > 1$.[1] Obviously, signal propagation on real-life interconnections is much slower than this lower bound, and optimistic predictions estimate propagation delays for highly optimized global wires—taking wire sizing and buffering into account—to be between six and 10 clock cycles for chips made using 50 nm technology.[2]

References

1. D. Sylvester and K. Keutzer, "A Global Wiring Paradigm for Deep Submicron Design," *IEEE Trans. CAD/ICAS*, Feb. 2000, pp. 242-252.
2. R. Ho, K. Mai, and M. Horowitz, "The Future of Wires," *Proc. the IEEE*, Apr. 2001, pp. 490-504.

NEW DESIGN APPROACH

Network engineers have already gained experience with using stochastic techniques and models for large-scale designs. We propose borrowing models, techniques, and tools from the network design field and applying them to SoC design.

We view a SoC as a micronetwork of components. The network is the abstraction of the communication among components and must satisfy quality-of-service requirements—such as reliability, performance, and energy bounds—under the limitation of intrinsically unreliable signal transmission and significant communication delays on wires. We propose using the micronetwork stack paradigm, an adaptation of the protocol stack shown in Figure 1,[2] to abstract the electrical, logic, and functional properties of the interconnection scheme.

SoCs differ from wide area networks in their local proximity and because they exhibit less nondeterminism. Local, high-performance networks—such as those developed for large-scale multiprocessors—have similar requirements and constraints. Some distinctive characteristics, such as energy constraints and design-time specialization, are unique to SoC networks, however.

Whereas computation and storage energy greatly benefit from device scaling, which provides smaller gates and memory cells, the energy for global communication does not scale down. On the contrary, as the "Wiring Delays" sidebar indicates, projections based on current delay optimization techniques for global wires[3] show that global on-chip communication will require increasingly higher energy consumption. Hence, minimizing the energy used for communications will be a growing concern in future technologies. Further, network traffic control and monitoring can help better manage the power that networked computational resources consume. For example, the clock speed and voltage of end nodes can vary according to available network bandwidth.

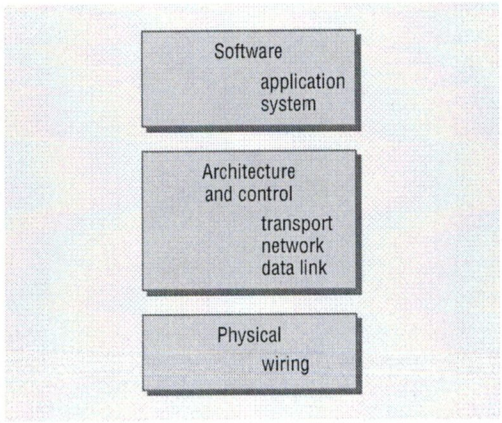


Figure 1. Protocol stack from which the micronetwork stack paradigm can be adapted. Bottom up, the layers span increasing design abstraction levels.

Another facet of the SoC network design problem, design-time specialization, raises many new challenges. Macroscopic networks emphasize general-purpose communication and modularity. Communication network design has traditionally been decoupled from specific end applications and is strongly influenced by standardization and compatibility constraints in legacy network infrastructures. In SoC networks, these constraints are less restrictive because developers design the communication network fabric on silicon from scratch. Thus, only the abstract network interface for the end nodes requires standardization. Developers can tailor the network architecture itself to the application, or class of applications, the SoC design targets.

We thus envision a vertical design flow in which every layer of the micronetwork stack is specialized and optimized for the target application domain. Such an application-specific on-chip network-synthesis paradigm represents an open and exciting research field. Specialization does not imply complete loss of flexibility, however. From a design standpoint, network reconfigurability will be key in providing plug-and-play component use because the components will interact with one another through reconfigurable protocols.

ON-CHIP SIGNAL TRANSMISSION

In addition to facilitating high-speed communication, reducing the voltage swing also has a beneficial effect on power dissipation.

Wires are the physical realization of communication channels in SoCs and, for our purposes, buses function as wire ensembles. Intensive research[3-5] into on-chip wiring has resulted in the commercial development of several physical design tools to support automated wiring. Nevertheless, coping with global wires that span significant distances, such as those beyond one millimeter, requires a paradigm shift.

Most likely, the reverse-scaled global wires will be routed on the top metal layers provided by the technology. Wiring pitch and width increase in higher wiring levels so that wires at top levels can be much wider and thicker than low-level wires.[5] Increased width reduces wire resistance, even considering the skin effect, while increased spacing around the wire prevents capacitance growth. At the same time, inductance effects increase relative to resistance and capacitance. As a result, future global wires will function as lossy transmission lines,[1] as opposed to today's lumped or distributed resistance-capacitance models.

In addition to facilitating high-speed communication, reducing the voltage swing also has a beneficial effect on power dissipation. Reduced-swing, current-mode transmission requires careful receiver design, with good adaptation to line impedance and high-sensitivity sensing, possibly with the help of sense amplifiers.

When using current technologies, most chip developers assume that electrical waveforms always carry correct on-chip information. Guaranteeing error-free information transfer at the physical level on global on-chip wires will become more difficult for several reasons.[6] Signal swings will be reduced and noise—due to crosstalk, electromagnetic interference, and other factors—will have increased impact. Thus, it will not be possible to abstract the physical layer of on-chip networks as a fully reliable, fixed-delay channel. At the micronetwork stack layers atop the physical layer, noise is a source of local transient malfunctions. An upset is the abstraction of such malfunctions. Upset probability can vary over different physical channels and over time.

In current designs, wiring-related effects are undesirable parasitics, and designers use specific, detailed physical techniques to reduce or cancel them. A well-balanced design should not try to achieve ideal wire behavior at the physical layer because the corresponding cost in performance, energy efficiency, and modularity may be too high. Physical-layer design should find a compromise between satisfying competing quality metrics and providing a clean and complete abstraction of channel characteristics for the micronetwork layers above.

MICRONETWORK ARCHITECTURE AND CONTROL

The architecture specifies the interconnection network's topology and physical organization, while the protocols specify how to use network resources during system operation. Whereas both micronetwork and general network design must meet performance requirements, the need to satisfy tight energy bounds differentiates on-chip network implementations.

Interconnection network architectures

On-chip networks relate closely to interconnection networks for high-performance parallel computers with multiple processors, in which each processor is an individual chip. Like multiprocessor interconnection networks, nodes are physically close to each other and have high link reliability. Further, developers have traditionally designed multiprocessor interconnections under stringent bandwidth and latency constraints to support effective parallelization.[7] Similar constraints will drive micronetwork design.

Shared-medium networks. Most current SoCs have a shared-medium architecture, which has the simplest interconnect structures. In this architecture, all communication devices share the transmission medium. Only one device can drive the network at a time. These networks support broadcast as well, an advantage for the highly asymmetric communication that occurs when information flows from few transmitters to many receivers. Within current technologies, the backplane bus is the most common example of an on-chip, shared-medium structure. This convenient, low-overhead interconnection handles a few active bus masters and many passive bus slaves that only respond to bus master requests.

We need bus arbitration mechanisms when several processors attempt to use the bus simultaneously. A bus arbiter module performs centralized arbitration in current on-chip buses. A processor seeking to communicate must first gain bus mastership from the arbiter. Because this process implies a control transaction and communication performance loss, arbitration should be as fast and rare as possible.

Together with arbitration, the response time of slow bus slaves may cause serious performance losses because the bus remains idle while the mas-

ter waits for the slave to respond. To minimize the bandwidth consumption, developers have devised split transaction protocols for high-performance buses. In these protocols, the network releases bus mastership upon request completion, and the slave must gain access to the bus to respond, possibly several bus cycles later. Thus, the bus can support multiple outstanding transactions.

Obviously, bus masters and bus interfaces for split-transaction buses are more complex than those for simple atomic-transaction buses. For example, developers chose a 128-bit split-transaction bus for the Lucent Daytona chip,[8] a multiprocessor on a chip that contains four 64-bit processing elements that generate transactions of different sizes. To improve bus-bandwidth utilization and minimize the average latency caused by simultaneous requests, the bus partitions large transfers into smaller packets.

Although well understood and widely used, shared-medium architectures have seriously limited scalability. The bus-based organization remains convenient for current SoCs that integrate fewer than five processors and, rarely, more than 10 bus masters. Energy inefficiency is another critical limitation of shared-medium networks. In these architectures, every data transfer is broadcast, meaning the data must reach each possible receiver at great energy cost. Future integrated systems will contain tens to hundreds of units generating information that must be transferred. For such systems, a bus-based network would become a critical performance and power bottleneck.

Direct and indirect networks. The direct or point-to-point network overcomes the scalability problems of shared-medium networks. In this architecture, each node directly connects to a limited number of neighboring nodes. These on-chip computational units contain a network interface block, often called a router, that handles communication and directly connects to neighboring nodes' routers. Direct interconnect networks are popular for building large-scale systems because the total communication bandwidth also increases when the number of nodes in the system increases.

The Raw Architecture Workstation (RAW) architecture[9] is an example of a direct network implementation derived from a fully programmable SoC consisting of an array of identical computational tiles with local storage. Full programmability means that the compiler can program both the function of each tile and the interconnections among them.

The term RAW derives from the "raw" hardware's full exposure to the compiler. To accomplish programmable communication, each tile has a router. The compiler programs the routers on all tiles to issue a sequence of commands that determines exactly which set of wires connect at every cycle. Moreover, the compiler pipelines the long wires to support high clock frequency.

Energy inefficiency is a critical limitation of shared-medium networks.

Indirect or switch-based networks offer an alternative to direct networks for scalable interconnection design. In these networks, a connection between nodes must go through a set of switches. The network adapter associated with each node connects to a switch's port. Switches themselves do not perform information processing—they only provide a programmable connection between their ports, setting up a communication path that can change over time.[7] Significantly, the distinction between direct and indirect networks is blurring as routers in direct networks and switches in indirect networks become more complex and absorb each other's functionality. As the "Virtex II FPGA" sidebar indicates, some field-programmable gate arrays are examples of indirect networks on chips.

Hybrid networks. Introducing a controlled amount of nonuniformity in communication-network design provides several advantages. Multiple-backplane and hierarchical buses are two notable examples of the many heterogeneous or hybrid interconnection architectures that developers have proposed and implemented. These architectures cluster tightly coupled computational units with high communication bandwidth and provide lower bandwidth intercluster communication links. Because they use a fraction of the communication resources and energy to provide performance comparable with homogeneous, high-bandwidth architectures, energy efficiency is a strong driver toward using hybrid architectures.[10]

Micronetwork control

Using micronetwork architectures effectively requires relying on protocols—network control algorithms that are often distributed. Network control dynamically manages network resources during system operation, striving to provide the required quality of service. Following the micronetwork stack layout shown in Figure 1, we describe the three architecture-and-control layers—data link, network, and transport—from the bottom up.

Data link layer. The physical layer is an unreliable digital link in which the probability of bit upsets is non-null. Data-link protocols increase the reliability of the link, up to a minimum required level,

Virtex II FPGA

Most current field-programmable gate arrays consist of a homogeneous fabric of programmable elements connected by a switch-based network. FPGAs can be seen as the archetype of future programmable SoCs: They contain many interconnected computing elements. Current FPGA communication networks differ from future SoC micronetworks in granularity and homogeneity.

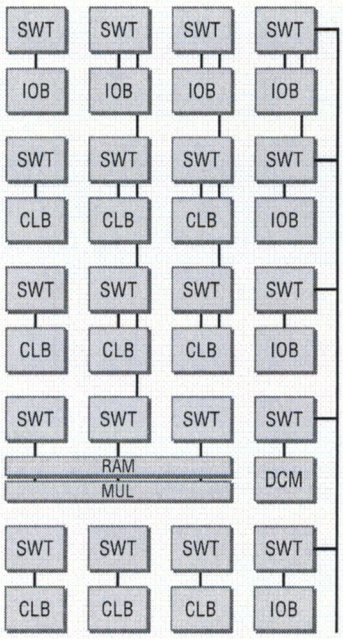


Figure A. Xilinx Virtex II, a field-programmable gate array architecture that exemplifies an indirect network over a heterogeneous fabric.

Processing elements in traditional FPGAs implement simple bit-level functional blocks. Thus, communication channels in FPGAs are functionally equivalent to wires that connect logic gates. Because future SoCs will house complex processing elements, interconnects will carry much coarser quantities of information. The different granularity of computational elements and communication requirements has far-reaching consequences for the complexity of the network interface circuitry associated with each communication channel. Interface circuitry and network control policies must be kept extremely simple for FPGAs, while they can be much more complex when supporting coarser-grain information transfers. The increased complexity will introduce greater degrees of freedom for optimizing communication as well.

The concept of dynamically reconfiguring FPGAs applies well to micronetwork design. SoCs benefit from programmability on the field to match, for example, environmental constraints. This programmability also lets runtime reconfiguration adapt, for example, to a varying workload. Reconfigurable micronetworks exploit programmable routers, switches, or both. Their embodiment may leverage multiplexers whose control signals are set—as with FPGAs—by configuration bits in local storage.

For example, Figure A shows the Xilinx Virtex II FPGA with various configurable elements to support reconfigurable digital-signal-processor design. The internal configurable rectangular array contains configurable logic blocks (CLBs), random access memories (RAMs), multipliers (MUL), switches (SWT), I/O buffers (IUB), and dynamic clock managers (DCM). Routing switches facilitate programmable interconnection. Each programmable element connects to a switch matrix, allowing multiple connections to the general routing matrix. Values stored in static memory cells control all programmable elements, including the routing resources. Thus, Virtex II exemplifies an indirect network over a heterogeneous fabric.

under the assumption that the physical layer by itself is not sufficiently reliable.

In a shared-medium network, contention creates an additional error source. Contention resolution, fundamentally a nondeterministic process, is an additional noise source because it requires synchronization of a distributed system. In general, synchronization can virtually eliminate nondeterminism at the price of some performance loss. For example, centralized bus arbitration eliminates contention-induced errors in a synchronous bus but the slow bus clock and bus request-and-release cycles impose a substantial performance penalty.

Packetizing data deals effectively with communication errors. Sending data on an unreliable channel in packets makes error containment and recovery easier because the packet boundaries contain the effect of errors and allow error recovery on a packet-by-packet basis. Using error-correcting codes that add redundancy to the transferred information can achieve error correction at the data link layer. Packet-based error-detection and -recovery protocols that have been developed for traditional networks, such as alternating-bit, go-back-*N*, and selective repeat, can complement error correction.[2] Several parameters in these protocols, such as packet size and number of outstanding packets, can be adjusted to achieve maximum performance at a specified residual error probability, within given energy consumption bounds, or both.

Network layer. This layer implements end-to-end delivery control in network architectures with many communication channels. In most current on-chip networks, all processing elements connect to the same channel: the on-chip bus, leaving the network layer empty. However, when a collection of links connects the processing elements, we must decide how to set up connections between successive links and route information from its source to the final destination. Developers have studied these

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.