30

Networks on Chips: Energy-Efficient Design of SoC Interconnect

Luca Benini
University of Bologna

Terry Tao Ye
Giovanni De Micheli
Stanford University

30.1 Introduction 30-1
30.2 Micro-Networks: Architectures and Protocols 30-2
Physical Layer • Data Link, Network, and Transport Layers • Software Layers
30.3 Energy-Efficient Micro-Network Design 30-6
Physical Layer • Data-Link Layer • Network Layer
30.4 Conclusions 30-14
References 30-14

30.1 Introduction

The challenge of designing systems on chip (SoCs) is related to both the large scale of integration and the small transistor features in the upcoming silicon technologies. Moreover, SoCs will be applied in many embedded systems, where reliability of operation and low-energy consumptions are key figures of merit.

It is a common belief that SoCs are designed using preexisting components, such as processors, controllers, and memory arrays. Design methodologies have to support component reuse in a plug-and-play fashion to be effective.

We think that the most critical factor in system integration will be related to the communication scheme among components. The implementation of on-chip communication largely affects the system correctness, reliability, and energy consumption. Indeed, technology trends foresee an increase in device density and frequency of operation, which both correlate to higher power consumption. Voltage down-scaling will mitigate the energy cost at the expenses of reduced signal integrity. Thus, future system designs will be based on a balancing act between performance, reliability, and energy consumption. This chapter will analyze this trade-off in the domain of on-chip component interconnection.

The challenges for on-chip interconnect stem from the physical properties of the interconnection wires. Propagation delays on global wires — spanning a significant fraction of the chip size — will carry signals whose propagation delay will exceed the clock period. Thus, signals on global wires will be pipelined. At the same time, the switched capacitance on global wires will constitute a significant fraction of the dynamic power dissipation. Moreover, estimating delays accurately will become increasingly harder, as wire geometries may be determined late in the design flow. Thus, the need for latency insensitive design is critical. The most likely synchronization paradigm for future chips is globally asynchronous locally synchronous (GALS), with many different clocks.

0-8493-1941-2/05/$0.00+$1.50

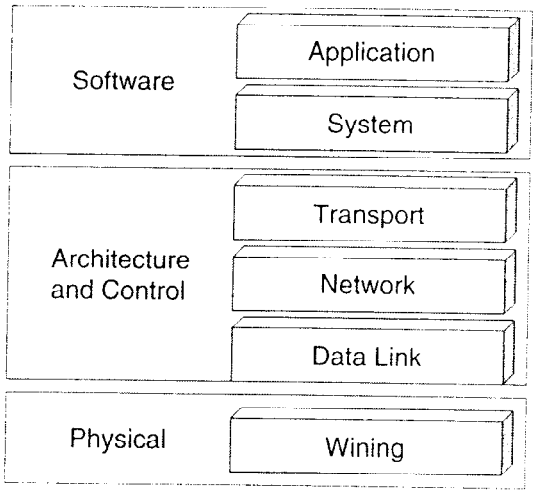

FIGURE 30.1 Micro-network stack.

SoC design will be guided by the principle of consuming the least possible power. This requirement matches the need of using SoCs in portable battery-powered electronic devices and of curtailing thermal dissipation, which can make chip operation infeasible or impractical. Energy considerations will impose small logic swings and power supplies, most likely below 1 V. Electrical noise due to crosstalk, electromagnetic interference (EMI), and radiation-induced charge injection (soft errors) will be likely to produce data upsets. Thus, the mere transmission of digital values on wires will be inherently unreliable.

To cope with these problems, we use network design technology to analyze and design SoCs modeled as micro-networks of components. The SoC interconnect design analysis and synthesis is based upon the micro-network stack paradigm, which is an adaptation of the protocol stack [30] (Figure 30.1) used in networking. This abstraction is useful for layering micro-network protocols and separating design issues belonging to different domains.

SoCs differ from wide-area networks because of local proximity and because they exhibit much less nondeterminism. In particular, micronetworks have a few distinctive characteristics, namely, energy constraints, design-time specialization, and low communication latency. This chapter addresses specifically the first problem.

Whereas computation and storage energy greatly benefits from device scaling (smaller gates, smaller memory cells), the energy for global communication does not scale down. On the contrary, projections based on current delay optimization techniques for global wires [15,28,29] demonstrate that global communication on chip will require increasingly higher energy consumption. Thus, communication-energy minimization will be a growing concern in future technologies. Furthermore, network traffic control and monitoring can help in better managing the power consumed by networked computational resources. For instance, clock speed and voltage of end nodes can be varied according to available network bandwidth. The emphasis on energy minimization creates a sleuth of novel challenges that have not been addressed by traditional high-performance network designers.

Design-time specialization is another facet of the SoC network design. Whereas macroscopic networks emphasize general-purpose communication and modularity, in SoCs networks, these constraints are less restrictive, because most on-chip solutions are proprietary. This degree of freedom can be used effectively to design low-energy communication schemes.

30.2 Micro-Networks: Architectures and Protocols

Much literature is available about architectures for macroscopic networks and, more specifically, for single-chip multi-processors [6,12,17]. These architectures can be classified by their topology, structure, and parameters. The most common on-chip communication architecture is the shared medium architecture, as exemplified by the shared bus. Unfortunately, bus performance and energy consumption are deeply penalized by the scaling up of the number of end nodes. Point-to-point architectures, such as

mesh, torus, and hypercube, have been demonstrated to scale up despite a higher complexity in their design. Examples of recent micro-network architectures include Octagon [17], which is a direct network (i.e., with routers attached to the end node) consisting of eight nodes arranged as the vertices of an octagon and connected via the octagon sides and diameters. Octagon has the properties that any two nodes can be reached in two hops, and that more octagons can be connected by sharing an end node. The Nostrum network is a two-dimensional indirect mesh network (i.e., with routers separate from end nodes) [19]. The network performs the routing function and acts as the network interface for each node processor as well. Another example of a recent micro-network is SPIN [12], which is also an indirect network, and is built with a 4-ary fat-tree topology.

Communication in any given architecture is regulated by protocols, which are designed in layers. We analyze next specific issues related to the different layers of abstraction outlined in the micro-network stack in a bottom-up way.

30.2.1 Physical Layer

Global wires are the physical implementation of the communication channels. Physical layer signaling techniques for lossy transmission lines have been studied for a long time by high-speed board designers and microwave engineers [2,10].

Traditional rail-to-rail voltage signaling with capacitive termination, as used today for on-chip communication, is definitely not well suited for high-speed, low-energy communication on future global interconnects [10]. Reduced swing, current-mode transmission, as used in some processor-memory systems, can significantly reduce communication power dissipation while preserving speed of data communication.

Nevertheless, as the technology trends lead us to use smaller voltage swings and capacitances, the upset probabilities will rise. Thus, the trend toward faster and lower-power communication may decrease reliability as an unfortunate side effect. Reliability bounds as voltages scale can be derived from theoretical (entropic) considerations [14] and can be measured by experiments on real circuits.

We conjecture that a paradigm shift is needed to address the aforementioned challenges. Current design styles consider wiring-related effects as undesirable parasitics, and try to reduce or cancel them by specific and detailed physical design techniques. It is important to realize that a well-balanced design should not over-design wires so that their behavior approaches an ideal one, because the corresponding cost in performance, energy-efficiency, and modularity may be too high. Physical layer design should find a compromise between competing quality metrics and provide a clean and complete abstraction of channel characteristics to micro-network layers above.

30.2.2 Data Link, Network, and Transport Layers

The data-link layer abstracts the physical layer as an unreliable digital link, where the probability of bit upsets is nonnull (and increasing as technology scales down). Furthermore, reliability can be traded off for energy [14]. The main purpose of data-link protocols is to increase the reliability of the link up to a minimum required level, under the assumption that the physical layer by itself is not sufficiently reliable.

An additional source of errors is contention in shared-medium networks. Contention resolution is fundamentally a nondeterministic process, because it requires synchronization of a distributed system, and for this reason it can be considered as an additional noise source. Generally, nondeterminism can be virtually eliminated at the price of some performance penalty. For instance, centralized bus arbitration in a synchronous bus eliminates contention-induced errors, at the price of a substantial performance penalty caused by the slow bus clock and by bus request/release cycles.

Future high-performance, shared-medium on-chip micro-networks may evolve in the same direction as high-speed local area networks, where contention for a shared communication channel can cause errors, because two or more transmitters are allowed to concurrently send data on a shared medium. In this case, provisions must be made for dealing with contention-induced errors.

An effective way to deal with errors in communication is to packetize data. If data is sent on an unreliable channel in packets, error containment and recovery is easier, because the effect of errors is contained by packet boundaries, and error recovery can be carried out on a packet-by-packet basis. At the data link layer, error correction can be achieved by using standard error correcting codes (ECC) that add redundancy to the transferred information. Error correction can be complemented by several packet-based error detection and recovery protocols. Several parameters in these protocols (e.g., packet size and number of outstanding packets) can be adjusted depending on the goal to achieve maximum performance at a specified residual error probability and/or within given energy consumption bounds.

At the network layer, packetized data transmission can be customized by the choice of switching and routing algorithms. The former establishes the type of connection while the latter determines the path followed by a message through the network to its final destination. Popular packet switching techniques include store-and-forward, virtual cut-through, and wormhole. When these switching techniques are implemented in on-chip networks, they will have different performance metrics along with different requirements on hardware resources.

- Store-and-forward (SAF) routing inspects each packet's content before forwarding it to the next stage. While SAF enables more elaborated routing algorithms, (e.g., content-aware packet routing), it introduces extra packet delay at every router stage. Furthermore, SAF also requires a substantial amount of buffer spaces because the switches need to store multiple complete packets at the same time. Because on-chip storage resources (i.e., static random access memory (SRAMs) and dynamic random access memory (DRAMs)) are very expensive in terms of area and energy consumption, SAF approaches are not appropriate for on-chip communications.
- Virtual cut-through (VCT) routing can forward a packet to the next stage before its entirety is received by the current switch. Therefore, VCT switching reduces the store-and-forward delays. When the next stage switch is not available, however, the entire packet still needs to be stored in the buffers of the current switch.
- Wormhole routing was originally designed for parallel computer clusters [11] because it achieves the minimal network delay and requires fewer buffers. In wormhole routing, each packet is further segmented into flits (flow control unit). The header flit reserves the routing channel of each switch, the body flits will then follow the reserved channel, and the tail flit will later release the channel reservation.

One major advantage of wormhole routing is that it does not require the complete packet to be stored in the switch while waiting for the header flit to route to the next stages. Wormhole routing not only reduces the store-and-forward delay at each switch, but it also requires much less buffer space. Because of these advantages, wormhole routing is an ideal candidate switching technique for on-chip interconnect networks [7].

In wormhole routing, one packet may occupy several intermediate switches at the same time. Thus, it may block the transmission of other packets. Deadlock and livelock are the potential problems in wormhole routing schemes [9,11].

At the transport layer, algorithms deal with the decomposition of messages into packets at the source and their assembly at destination. Packetization granularity is a critical design decision, because the behavior of most network control algorithms is very sensitive to packet size. Packet size can be application-specific in SoCs, as opposed to general networks. In general, flow control and negotiation can be based on either deterministic or statistical procedures. Deterministic approaches ensure that traffic meets specifications, and provide hard bounds on delays or message losses. The main disadvantage of deterministic techniques is that they are based on worst cases, and they generally lead to significant under-utilization of network resources. Statistical techniques are more efficient in terms of utilization, but they cannot provide worst-case guarantees. Similarly, from an energy viewpoint, we expect deterministic schemes to be more inefficient than statistical schemes, because of their implicit worst-case assumptions.

30.2.3 Software Layers

Current and future systems on chip will be highly programmable, and therefore their power consumption will critically depend on software aspects. Software layers comprise system and application software. The system software provides us with an abstraction of the underlying hardware platform, which can be leveraged by the application developer to exploit the hardware's capabilities safely and effectively.

Current SoC software development platforms are mostly geared toward single microcontroller with multiple coprocessor architectures. Most of the system software runs on the control processor, which orchestrates the system activity and farms off computationally intensive tasks to domain-specific coprocessors. Micro-controller–coprocessor communication is usually not data-intensive (e.g., synchronization and reconfiguration information), and most high-bandwidth data communication (e.g., coprocessor–coprocessor and coprocessor–IO) is performed via shared memories and direct memory access (DMA) transfers. The orchestration activities in the micro-controller are performed via runtime services provided by single-processor real time operating systems (RTOSs) (e.g., VxWorks, Micro-OS, and Embedded Linuxes), which differentiate from standard operating systems in their enhanced modularity, reduced memory footprint, and support for real-time-scheduling and bounded time-interrupt service times.

Application programming is mostly based on manual partitioning and distribution of the most computationally intensive kernels to data coprocessors (e.g., very long instruction word (VLIW) multimedia engines, digital signal processors, etc.). After partitioning, different code generation and optimization toolchains are used for each target coprocessor and the control processor. Hand-optimization at the assembly level is still quite common for highly irregular signal processors, while advanced optimizing compilers are often used for VLIW engines and fine-grained reconfigurable fabrics. Explicit communication via shared memory is usually supported via storage classes declarations (e.g., noncacheable memory pages) and DMA transfers from and to shared memories are usually set up via specialized system calls which access the memory-mapped control registers of the DMA engines.

Even from this cursory analysis, the poor scalability in a network on chip (NoC) setting of current software abstractions and runtime environments is evident. In our view, the most critical issues are the following:

- Confining the OS onto a single centralized micro-controller is a sensible choice for small-to-medium scale and asymmetric multi-processing architectures, but this choice is bound to create a performance bottleneck and significant power overhead as architectures become more symmetric and scale up in complexity and parallelism, resulting in a significant energy inefficiency. This is because all centralized control functions and policies will require communication (often under tight real-time constrains) to all peripheral processors. Even worse, a centralized OS would need to continuously collect information on all system components to maintain an updated system state snapshot. The cost in performance and power of system control and monitoring is significant and could either lead to an over budgeting of NoC resources (e.g., dedicated control channels) if quality-of-service guarantees (e.g., bounded control message delivery delay) must be provided, or to uncertain and unreliable operation in case of a best-effort network service.
- The manual and ad-hoc partitioning and workload distribution procedure is too slow and error-prone in parallel, large-scale applications and target architectures. Furthermore, the lack of communication analysis tools may lead to highly inefficient task mappings. From a performance viewpoint, a communication sub-optimal task mapping leads to reduced throughput and/or high latency. The energy implications can be even more serious, because in many cases reduced performance is caused by local congestion, which is a high-occupancy condition for network resources and implies high power consumption. Thus, energy efficiency decreases quadratically (high power and low performance).
- Current programming styles are based on a shared memory paradigm, which is quite natural and well suited for tightly coupled, small-scale clusters. Unfortunately, shared memory abstraction tends to hide the cost and unpredictability of communication, which are destined to grow in an

NoC setting. Furthermore, DMA burst transfers, often advocated as a mean to increase throughput in memory transfers, do increase the risk of starvation and the variance in delivery time of short, sporadic messages. From an energy viewpoint, we need to raise the level of awareness of programmers on the energy cost in accessing shared memories, especially when a single memory is shared among many multiple processors. Such a cost is only in part due to pure memory array access. Significant overheads are associated with communication and contention resolution.

In our view, software issues are among the most critical and less understood in NoC. We believe that the full potential of on-chip networks can be effectively exploited only if adequate software abstractions and programming aids are developed to support them.

30.3 Energy-Efficient Micro-Network Design

This section delves into a few specific instances of energy-efficient, micro-network design problems. In most cases, we also outline specific solutions that have been proposed in the literature, even though it should be clear that many design issues are open and significant progress in this area is expected in the near future.

30.3.1 Physical Layer

At the physical layer, low-swing signaling is actively investigated to reduce communication energy on global interconnects [36]. In the case of a simple CMOS driver, low-swing signaling is achieved by lowering the driver's supply voltage V_{dd}. This implies a quadratic dynamic power reduction (because $P_{dyn} = K V_{dd}^2$). Unfortunately, swing reduction at the transmitter complicates the receiver's design. Increased sensitivity and noise immunity are required to guarantee reliable data reception. Differential receivers have superior sensitivity and robustness, but they require doubling the bus width. To reduce the overhead, pseudo-differential schemes have been proposed, where a reference signal is shared among several bus lines and receivers, and incoming data is compared against the reference in each receiver. Pseudo-differential signaling reduces the number of signal transitions, but it has reduced noise margins with respect to fully differential signaling. Thus, reduced switching activity is counterbalanced by higher swings and determining the minimum-energy solution requires careful circuit-level analysis.

Dynamic voltage scaling has been recently applied to busses [26,33]. In Worm et al. [33], the voltage swing on communication busses is reduced, even though signal integrity is partially compromised. Encoding techniques are used to detect corrupted data that is retransmitted. The retransimission rate is an input to a closed-loop DVS control scheme, which sets the voltage swing at a trade-off point between energy saving and latency penalty (due to data retransmission).

Another key physical-layer issue is synchronization. Traditional on-chip communication has been based on the synchronous assumption, which implies the presence of global synchronization signals (i.e., clocks) that define data sampling instants throughout the chip. Unfortunately, clocks are extremely energy-inefficient, and it is a well-known fact that they are responsible for a significant fraction of the power budget in digital integrated systems. Thus, postulating global synchronization when designing on-chip micronetworks is not an optimal choice from the energy viewpoint. Alternative on-chip synchronization protocols that do not require the presence of a global clock have been proposed in the past [3,37], but their effectiveness has not been studied in detail from the energy viewpoint.

30.3.2 Data-Link Layer

At the data-link layer, a key challenge is to achieve the specified communication reliability level with minimum energy expense. Several error recovery mechanisms developed for macroscopic networks can be deployed in on-chip micronetworks, but their energy efficiency should be carefully assessed in this context. As a practical example, consider two alternative reliability-enhancement techniques: error-cor-

recting codes and error-detecting codes with retransmission. A set of experiments involved applying error correcting and detecting codes to an AMBA bus and comparing the energy consumption in four cases [4]:

1. Original unencoded data
2. Single-error correction
3. Single-error correction and double-error detection
4. Multiple-error detection

Hamming codes were used. Note that in case 3, a detected double error requires retransmission. In case 4, using (n,k) linear codes, there are $(2^n - 2^k)$ error patterns of length n that can be detected. In all cases, some errors may go undetected and be catastrophic. Using the property of the codes, it is possible to map the mean time to failure (MTTF) requirement into bit upset probabilities, and thus comparing the effectiveness of the encoding scheme in a given noisy channel (characterized by the upset probability) in meeting the MTTF target.

The energy efficiency of various encoding schemes varies: we summarize here one interesting case, where three assumptions apply. First, wires are long enough so that the corresponding energy dissipation dominates encoding/decoding energy. Second, voltage swing can be lowered until the MTTF target is met. Third, upset probabilities are computed using a white Gaussian noise model [13]. Figure 30.2 gives the average energy per useful bit as a function of the MTTF (which is the inverse of the residual word error probability). In particular, for reliable SoCs (i.e., for MTTF = 1 year), multiple-error detection with retransmission is demonstrated as more efficient than error-correcting schemes. We refer the reader to Bertozzi et al. [4] for results under different assumptions.

Another important aspect affecting the energy consumption is the media access control (MAC) function. Currently, centralized time-division multiplexing schemes (also called centralized arbitration) are widely adopted [1,8,32]. In these schemes, a single arbiter circuit decides which transmitter accesses to the bus for every time slot. Unfortunately, the poor scalability of centralized arbitration indicates that this approach is likely to be energy-inefficient as micronetwork complexity scales up. In fact, the energy cost of communicating with the arbiter, and the hardware complexity of the arbiter itself scales up more than linearly with the number of bus masters.

Distributed arbitration schemes as well as alternative multiplexing approaches, such as code division multiplexing, have been extensively adopted in shared-medium macroscopic network, and are actively investigated for on-chip communication [34]. Research in this area is just burgeoning, however, and significant work is needed to develop energy-aware media-access-control for future micronetworks.

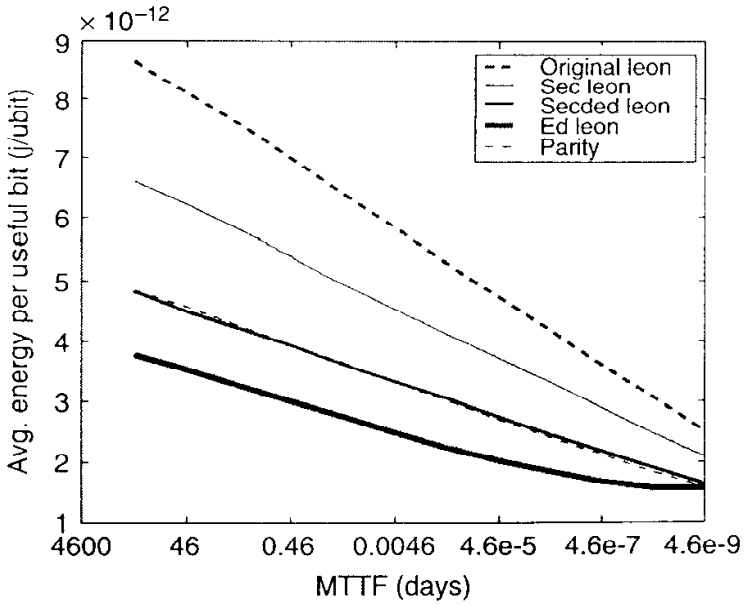

FIGURE 30.2 Energy efficiency for various encoding schemes.

30.3.3 Network Layer

Switching and routing for on-chip micro-networks affect heavily performance and energy consumption, whereas contention plays an important role. On one hand, contention delays packet transmission. On the other hand, resolving contention requires packets to be stored temporarily on the storage elements (on-chip SRAMs or DRAMs), which will increase power consumption significantly.

30.3.3.1 Contention-Look-Ahead Routing

A contention-look-ahead routing scheme is the one where the current routing decision is helped by monitoring the adjacent switches, thus possibly avoiding or reducing blockages and contention in the coming stages.

A contention-aware routing scheme is described in Nilsson [23]. The routing decision at every node is based on the "stress values" (the traffic loads of the neighbors) that are propagated between neighboring nodes. This scheme is effective in avoiding "hot spots" in the network. The routing decision steers the packets to less congested nodes.

To solve the contention problems in the wormhole routing schemes, we propose a contention-look-ahead routing algorithm that can "foresee" the contention and delays in the coming stages using a direct connection from the neighboring nodes. We use a two-dimensional mesh on-chip multiprocessor network to further explain and implement this routing algorithm. The processors are connected directly to each other in a tile array formation, similar to that proposed by Dally and Toles [7]. Each processor tile performs packet routing and arbitration independently. The major difference from Nilsson [23] is that information is handled in flits, and thus packets with large or variable sizes can be handled with limited input buffers. Furthermore, because it avoids contention between packets and requires much less buffer usage, the proposed contention-look-ahead routing scheme can greatly reduce the network power consumption.

30.3.3.2 Wormhole Contention-Look-Ahead Algorithm

At every intermediate stage, there may be many alternate routes to go to the next stage. We call the route that always leads the packet closer to the destination a profitable route. Conversely, a route that leads the packet away from the destination is called misroute [11] (Figure 30.3). In mesh networks, profitable routes and misroutes can be distinguished by comparing the current node ID with the destination node ID.

Profitable routes will guarantee a shortest path from source to destination. Nevertheless, misroutes do not necessarily need to be avoided. Occasionally, the buffer queues in all available profitable routes are full, or the queues are too long. Thus, detouring to a misroute may lead to a shorter delay time. Under these circumstances, a misroute may be more desirable.

Any packet entering an intermediate switch along a path finds a set C of output channels to exit. As an example, for a two-dimensional mesh, C = {North, South, East, West}. We further partition C into profitable routes P and misroutes M. We define the buffer queue length of every profitable route $p \in P$ as Q_p. Similarly, we define the buffer queue length of every misroute $m \in M$ as Q_m.

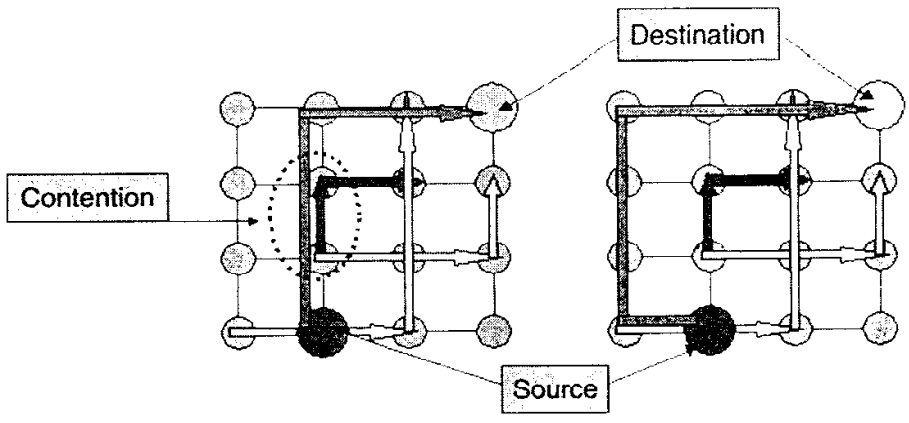

FIGURE 30.3 Profitable route and misroute.

Assume the flit delay of one buffer stage is D_B, and the flit delay of one switch stage is DS. The delay penalty to take a profitable and a misroute is defined as D_{profit} and $D_{misroute}$, respectively, in the following equation (Equation 30.1).

$$D_{profit} = min(D_B \times Q_p) \forall p \in P \tag{30.1}$$

$$D_{misroute} = min(D_B \times Q_m + 2D_S) \forall m \in M \tag{30.2}$$

In a mesh network, when a switch routes a packet to a misroute, the packet moves away from its destination by one switch stage. In the subsequent routing steps, this packet needs to get back on track and route one more stage back toward its destination. Therefore, the delay penalty for a misroute is $2 \times D_S$. The delay D_S can be estimated beforehand, and, without loss of generality, we assume the same D_S value for all switches in the network.

If all profitable routes are available and waiting queues are free, the packet will use profitable routing decision. If the buffer queues on all of the profitable routes are full or the minimum delay penalty of all the profitable routes is larger than the minimum penalty of the misroutes, it is more desirable to take the misroute (Equation 30.3):

$$(D_{profit} \leq D_{misroute})(Q_p \leq Q_{p_{max}} \forall p \in P)?ProRoute:Misroute \tag{30.3}$$

where Q_{pmax} is the maximum buffer queue length (buffer limit). This routing algorithm is heuristic, because it can only "foresee" one step ahead of the network. It provides a local best solution but does not guarantee the global optimum.

30.3.3.3 Network Power Consumption

This routing scheme was simulated with RSIM, a multiprocessor instruction level simulator, using 16 reduced instruction set computer (RISC) processors connected in a 4×4 (4-ary 2-cube) mesh network. Control wires that deliver the input queue information to the adjacent switches also connect adjacent processors.

The contention-look-ahead routing algorithm is compared with dimension-ordered routing — a routing scheme that always routes the packets on one dimension first, upon reaching the destination row or column, then switch to the other dimension until reaching the destination. Dimension-ordered routing is deterministic and guarantees shortest path, but it cannot avoid contention. The comparison is performed on four benchmarks: quicksort, fft, lu, and sor. These benchmarks are ported from Stanford SPLASH suite [27] and running on the RSIM simulation platform.

On-chip network power consumption comes from three contributors:

1. The interconnect wires
2. The buffers
3. The switch logic circuits

A network power consumption estimation technique is proposed by Ye et al. [35], and we will use it in the experiments.

The contention-look-ahead routing will reduce the power consumption on the buffers because it can "foresee" the contention in the forthcoming stages and shorten the buffer queue length. Figure 30.4(a) presents the averaged buffer power reduction of different benchmarks. The reduction is more significant under larger buffer sizes. This is because larger buffers will consume more power, and the power consumption is more sensitive with contention occurrence.

The power consumption on the interconnect can be estimated by counting the average number of hops a packet travels from source to destination. Dimension-ordered routing always steers the packets along the shortest path. In comparison, our proposed routing scheme may choose the misroute when contention occurs. Therefore, the contention-look-ahead routing has larger average hop count per

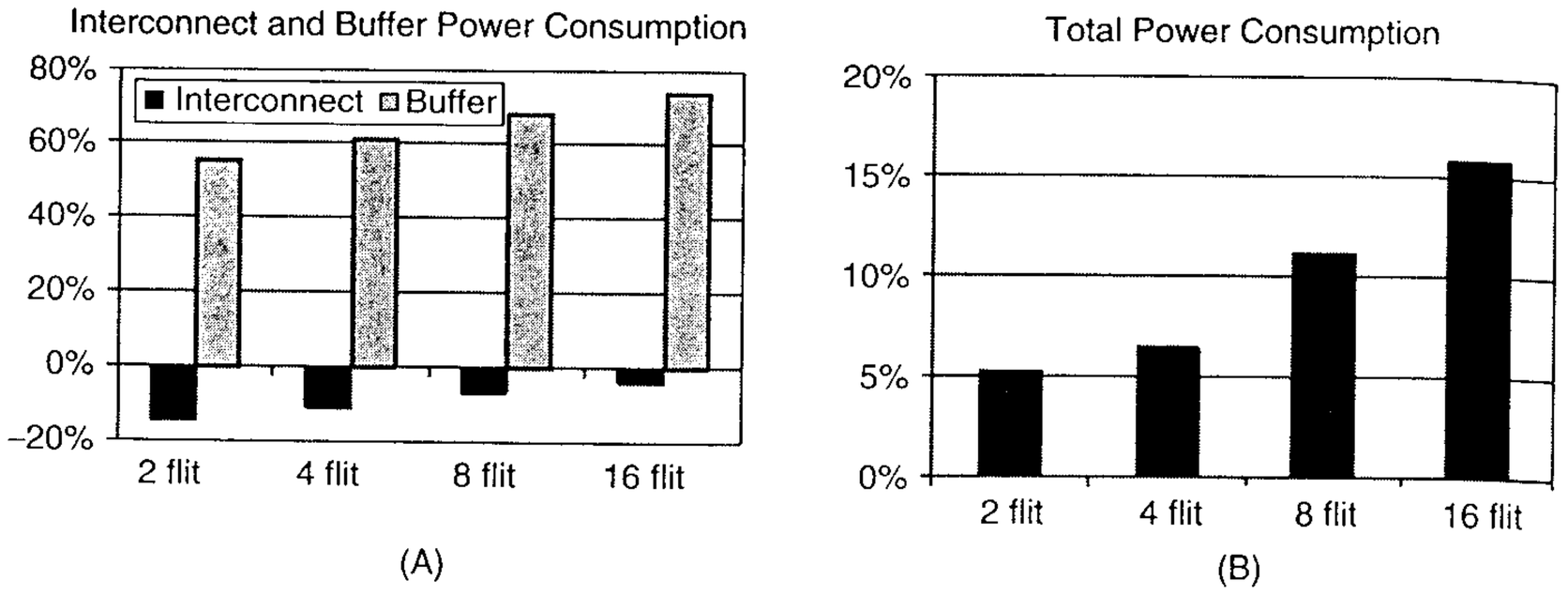

FIGURE 30.4 Power consumption comparison on interconnect wires and buffers.

packet than the dimension-ordered routing, and consequently consumes more power on the interconnects. Figure 30.4(A) depicts the proposed routing scheme consumes more power (presented as negative values) with smaller buffer size, this is because smaller buffer sizes will cause more contention and induce more misroutes.

The contention-look-ahead routing switch needs more logic gates than dimension-ordered routing. From synopsys power compiler simulation, the proposed switch circuit consumes about 4.8% more power than dimension-ordered switch. Combining the power consumption on the interconnects and buffers, the total network power consumption is depicted in Figure 30.4(B). It presents the total network power reduction compared with dimension-ordered routing. The reduction is more significant with larger buffer sizes (15.2% with 16-flit buffers).

30.3.3.3.1 Transport Layer

Above the network layer, the communication abstraction is an end-to-end connection. The transport layer is concerned with optimizing the usage of network resources and providing a requested quality of service. Clearly, energy can be considered as a network resource or a component in a quality-of-service metric. An example of transport-layer design issue is the choice of information decomposition into packets or flits, as well as the choice of packet size. Energy efficiency can be heavily impacted by this decision. Next, we will use the shared-memory multi-processor system on chip (MPSoC) as a case study to analysis the packet size trade-offs both qualitatively and quantitatively.

A typical shared-memory MPSoC architecture is illustrated in Figure 30.5. The MPSoC power consumption originates from three sources:

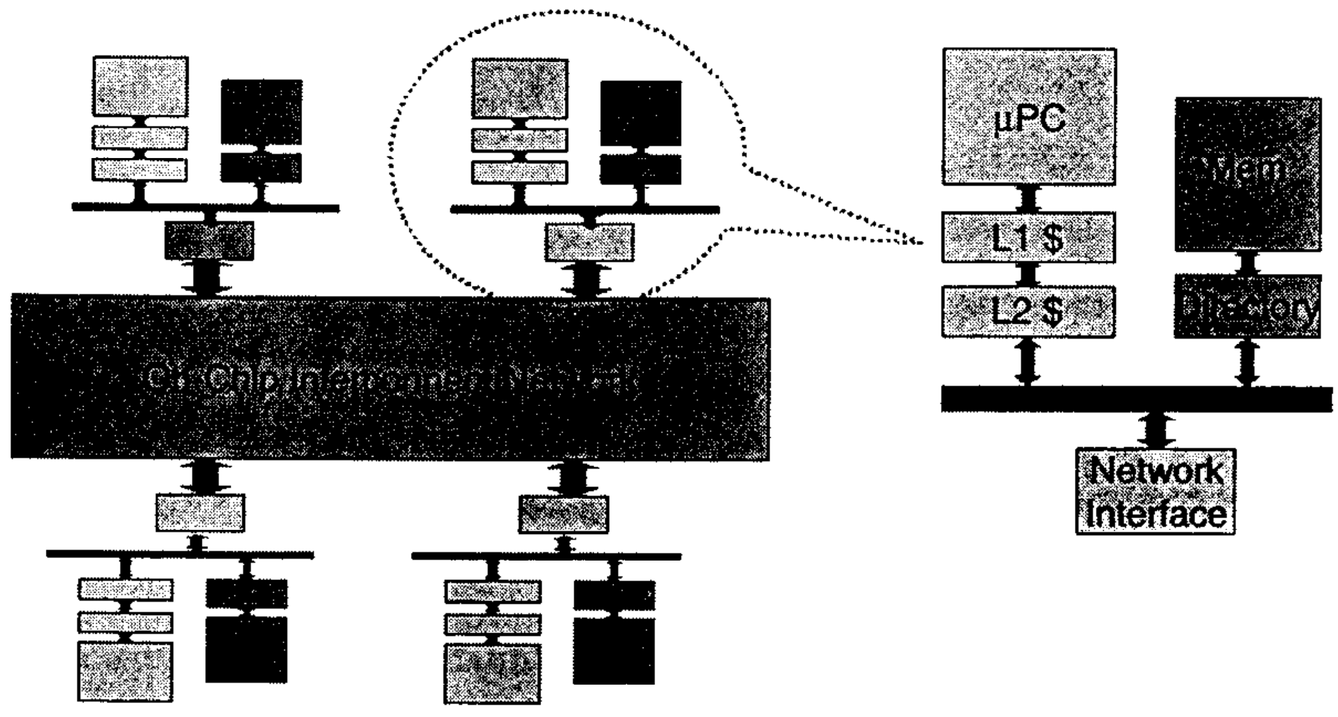

FIGURE 30.5 MPSoC architecture.

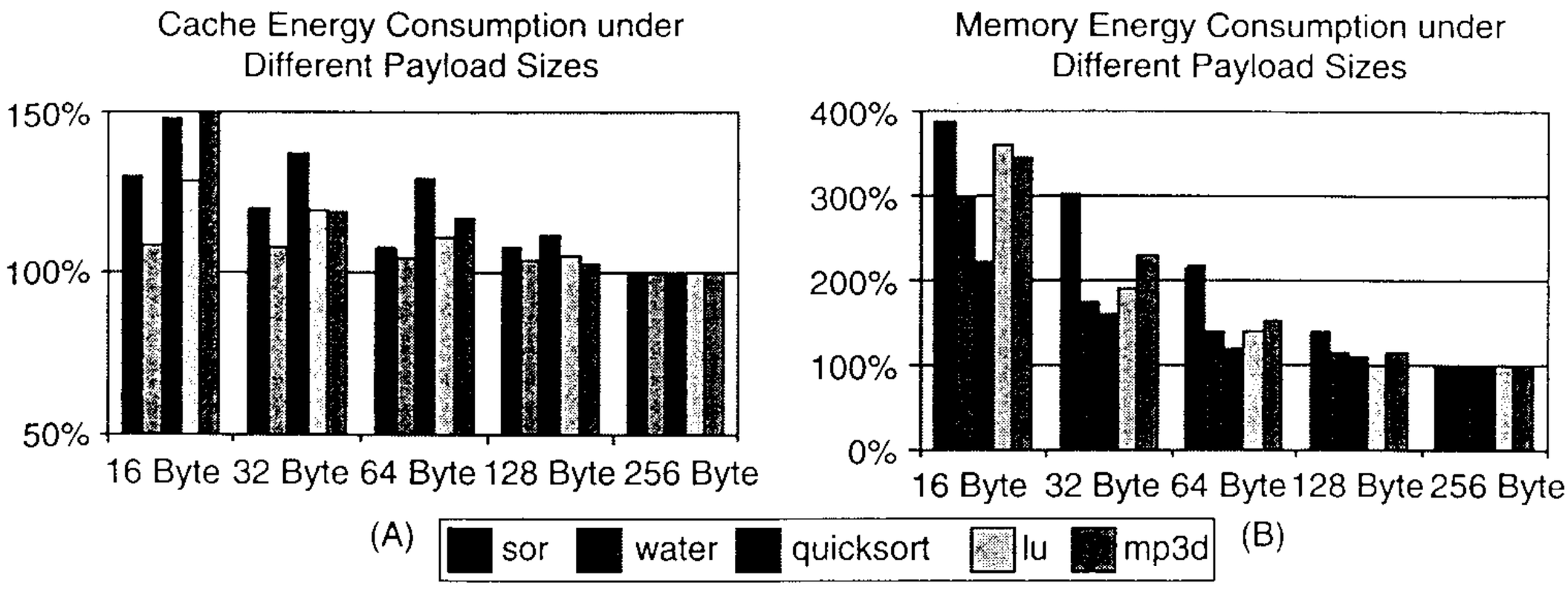

FIGURE 30.6 Cache and memory energy decrease as packet payload size increases.

1. The node processor power consumption
2. The cache and shared memory power consumption
3. The interconnect network power consumption

We will start first from the cache and memory analysis.

30.3.3.4 Cache and Memory Power Consumption

Whenever there is a cache miss, the cache block content needs to be encapsulated inside the packet payload and sent across the network. In shared-memory MPSoC, the cache block size correlates with the packet payload size. Larger packet sizes will decrease the cache miss rate, because more cache content can be updated in one memory access. Consequently, both cache energy consumption and memory energy consumption will be reduced. This relationship can be observed in Figure 30.6. It depicts the energy consumption by cache and memory under different packet sizes. The energy in the figure is normalized to the value of 256 Bytes, which achieves the minimum energy consumption.

30.3.3.5 Interconnect Network Power Consumption

The power consumption of packetized dataflow on MPSoC network is determined by three factors. The effects of these factors are summarized and listed next:

1. The number of packets on network. Packets with larger payload size will decrease the cache miss rate and consequently decrease the number of packets on the network. This effect can be observed in Figure 30.7(A). It gives the average number of packets on the network (traffic density) at one clock cycle. As the packet size increases, the number of packets decreases accordingly.

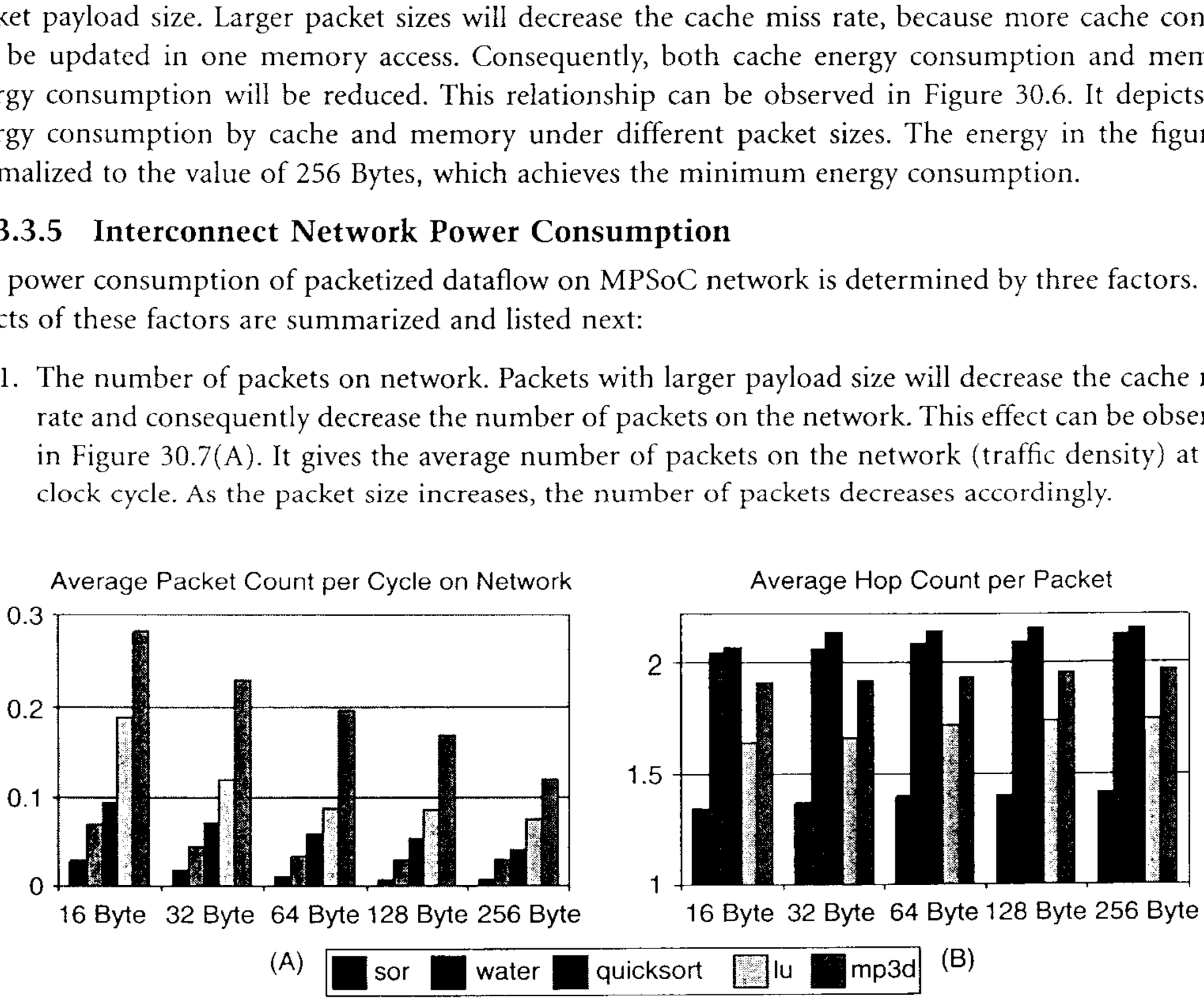

FIGURE 30.7 Packet count and hop count per packet under different payload sizes.

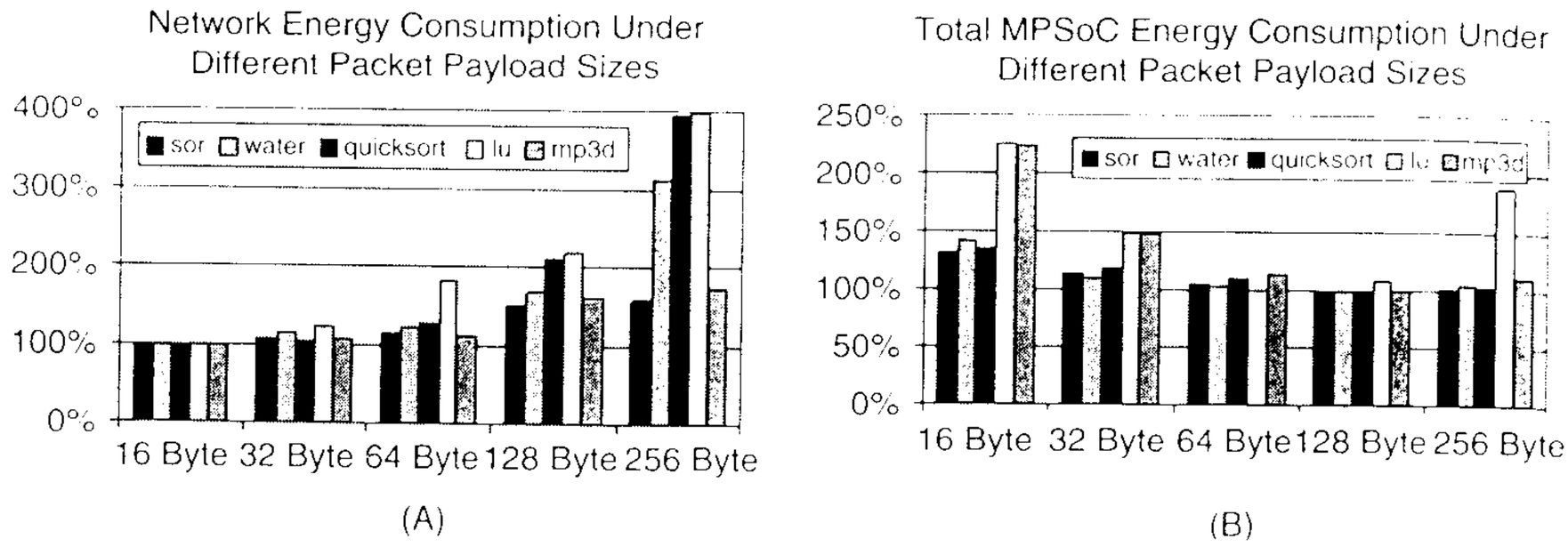

FIGURE 30.8 Network and total MPSoC energy consumption under different packet payload sizes.

2. The energy consumed by each packet on one hop. Larger packet size will increase the energy consumed per packet, because there are more bits in the payload.
3. The number of hops each packet travels. Larger packets will occupy the intermediate node switches for a longer time, and cause other packets to be rerouted to longer paths. This leads to more contention that will increase the total number of hops needed for packets traveling from source to destination. Figure 30.7(B) illustrates the effect. As packet size (payload size) increases, average hop count per packet increases as well.

Actually, increasing the cache block size will not decrease the cache miss rate proportionally. Therefore, the decrease of packet count cannot compensate for the increase of energy consumed per packet caused by the increase of packet length. Larger packet size also increases the hop counts on the datapath. Figure 30.8(A) presents the combined effects of these factors. The values are normalized to the measurement of 16 Bytes. As packet size increases, energy consumption on the interconnect network will increase.

The total energy dissipated on MPSoC comes from noncache instructions (instructions that do not involve cache access) of each node processors, the caches and the shared memories as well as the interconnect network. The overall results are given in Figure 30.8(B). From this figure, we can see that the total MPSoC energy will decrease as packet size increases. When the packets are too large, however, as in the case of 256 Bytes in the figure, the total MPSoC energy will increase. This is because when the packet is too large, the increase of interconnect network energy will outgrow the decrease of energy on cache and memories. In our simulation, the noncache instruction energy consumption does not change significantly under different packet sizes.

30.3.3.5.1 Application and System Layer

As hinted in Section 30.2, software layers are critical for the NoC paradigm shift, especially when energy efficiency is a requirement. As outlined in the previous sections, NoCs have the potential for overcoming many of the energy bottlenecks of current integrated architectures (i.e., globally shared communication and storage blocks), but only if programming abstractions, development tools, and system software help programmers understand communication-related costs and how to cope with them.

From a high-level application viewpoint, multi-processor SoC platforms can be viewed as networks of computing nodes equipped with local storage. Computation and storage are highly energy efficient if confined to the local resources within a node. Communication cost should be made explicit throughout all steps of the code development flow. Software analysis tools should help designers in identifying communication bottlenecks and code optimizers should heavily emphasize communication cost reduction. Many effective techniques have been devised in the area of parallel programming for large-scale supercomputers, and there is good potential for leveraging these experiences. It is important, however, to point out three key differences:

1. Target MPSoC architectures are much more heterogeneous than general-purpose parallel computers.

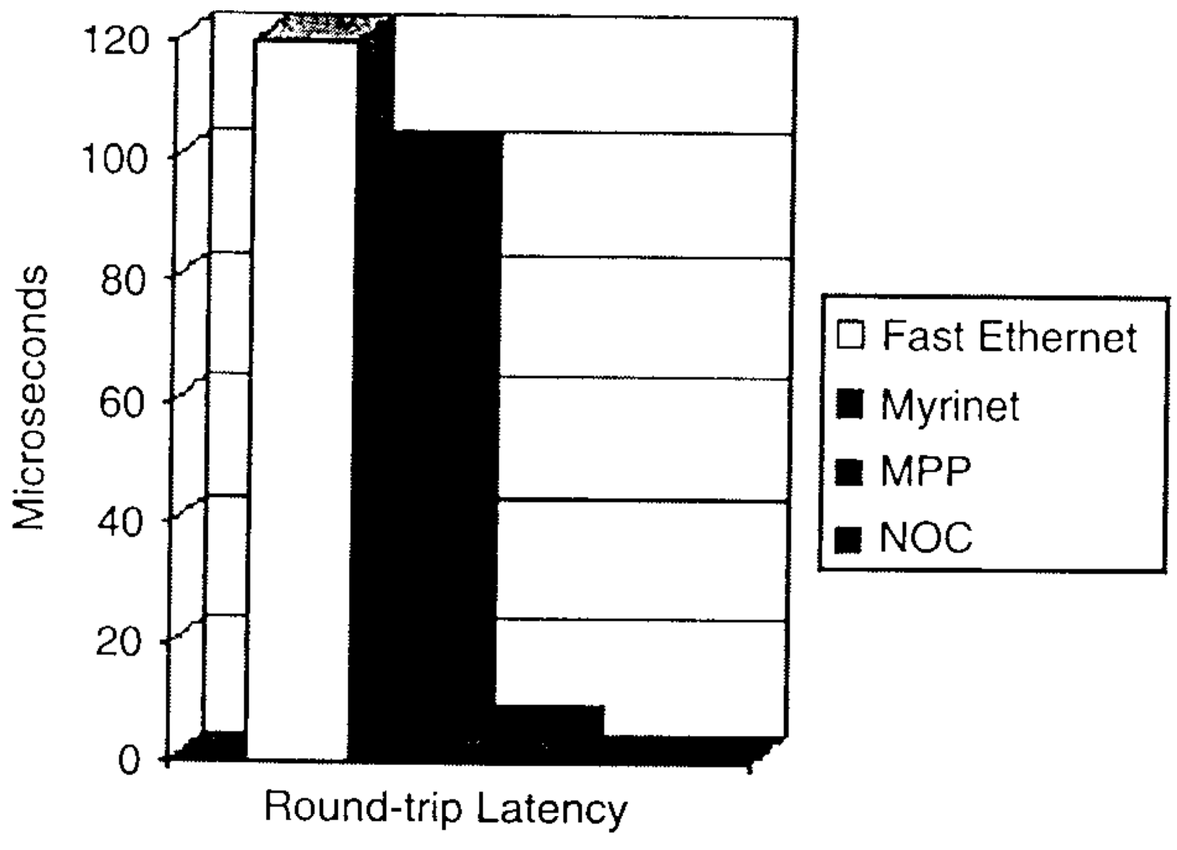

FIGURE 30.9 Interconnect latency for different parallel machines.

2. Physical link latency, albeit significant, is not nearly as dominant in on-chip networks as it is in macroscopic parallel computers (see Figure 30.9).
3. Energy constraints are extremely tight, while they have never been significant in traditional parallel machines.

The following paragraphs outline four critical areas for the evolution of energy efficient software layers in current and future NoCs. We survey seminal contributions and identify critical needs.

1. Programming abstractions. Developing adequate abstractions for NoC programming is a critical objective. Dataflow programming abstractions, such as streams [18] and Kahn networks [22], are based on a model of computation that matches very well NoC architectures. With these abstractions, communication is made explicit starting from the early steps of application development, because data flow is explicitly represented. At these levels, energy efficiency can be pursued by minimizing redundant communication, and by carefully balancing local computation and communication costs. A critical need in this area is the definition of hardware platform dependent high-level metrics, such as energy per local operation and energy per transmitted bit, which can help in first-cut exploration of the communication vs. computation trade-off during algorithm development. Unfortunately, even though many digital signal processing and multimedia applications are developed starting from dataflow models, numerous legacy applications use more traditional programming styles, where tasks are not clearly decoupled and communication is implicitly performed through memory. Leveraging the existing code basis, without compromising performance and energy efficiency, is today an open challenge.
2. Task-level analysis and optimization. A number of interesting opportunities are open for high-level optimization tools, which can help designers mapping data-flow specifications onto target hardware platforms. Consider, for instance, task splitting and merging (i.e., distributing the computation performed by a task among two or more computational nodes and collapsing two or more tasks onto the same node), task allocation, as well as communication splitting and merging over available physical NoC links. Even though a few of these problems have been explored in preliminary works [16,25], we critically need high-level energy models and analysis tools to explore techniques for increasing energy efficiency. It is important to acknowledge that energy-optimal solutions can differ significantly from performance-optimal ones in this context. Consider, for instance, a situation where available computational resources are used to achieve marginal performance benefits (e.g., a task is speculatively executed), at a price of significantly increased power consumption.

3. Code optimization. Classical code optimization, at the single task level, will still hold an important place in future NoC software development. In this area we view as critical further developments of two types of code optimizers: tools for parallelism extraction from a single task or a legacy applications [31]; tools that reduce the memory footprint and improve access locality for both code and data [24]. The first class of tools represents enabling technology that enables the reuse of legacy software as well as task splitting. The second class has a critical role in reducing "implicit" communication (as opposed to "explicit" data flow communication) from/to large background memories, which ensues because of large working sets that do not fit into local node memory. Another critical area in code optimization is the development of highly efficient communication primitives, possibly with significant dedicated hardware support. The latency and energy consumption associated with software handling of communication primitives (e.g., message send or receive) in traditional parallel programming libraries are simply unacceptable in an NoC setting [5].
4. Distributed operating systems. Intuitively, the operating system support NoC operation cannot be centralized. Truly distributed embedded OSes are required [5,6] to create a scalable runtime system. In addition to traditional functions (i.e., scheduling, interrupt handling), the NoC OS should natively support power management. End-nodes (processing elements) in SoC micronetworks will most likely be power-manageable "voltage islands" [20], with individually controllable clock speeds and supply voltages. One of the key tasks of the system software will be to control the voltage islands power states. We can envision a network-centric approach, where components send messages to neighbors to request state changes [21]. Such requests are originated and serviced at the system software levels. For example, an image processor can be required to raise its service levels before receiving a stream of data. In this case, the system software supports policies that accept requests from other components and perform transitions according to such requests.

30.4 Conclusions

The challenges of designing SoCs in 50- to 100-nm technologies available in the second part of this decade include coping with design complexity and providing reliable, high-performance operation and minimizing energy consumption. Starting from the observation that interconnect technology will be the limiting factor for achieving the operational goals, we envisioned a communication-centric view of design. We focused on energy efficiency issues in designing the communication infrastructure for future SoCs. We described several open problems at various layers of the communication stack, and we outlined basic strategies to effectively tackle the energy efficiency challenge for on-chip communication networks.

References

[1] P. Aldworth, System-on-a-chip bus architecture for embedded applications, *IEEE Int. Conference on Comput. Design*, pp. 297–298, 1999.

[2] H. Bakoglu, *Circuits, Interconnections, and Packaging for VLSI*, Addison-Wesley, Reading, MA, 1990.

[3] W.J. Bainbridge and S.B. Furber, Delay insensitive system-on-chip interconnect using 1-of-4 data encoding, *IEEE Int. Symp. on Asynchronous Circuits and Syst.*, March 2001, pp. 118–126.

[4] D. Bertozzi, L. Benini, and G. De Micheli, Low power error resilient encoding for on-chip data busses, *Proc. Int. Conf. on Design and Test Europe*, Paris, France, March 2000, pp. 102–109.

[5] D. Bertozzi, F. Poletti, L. Benini, and A. Bogliolo, Performance analysis of arbitration policies for SoC communication architectures, *J. Design Automation for Embedded Sys.*, Vol. 8, June–Sept. 2003, pp. 189–210.

[6] W.O. Cesario, D. Lyonnard, G. Nicolescu, Y. Paviot, S. Yoo, L. Gauthier, M. Diaz-Nava, and A.A. Jerraya, Multiprocessor SoC platforms: a component-based design approach *IEEE Design andTest of Comput.*, Vol. 19 No. 6, Nov.–Dec., 2002

[7] W. Dally and B. Towles, Route packets, not wires: on-chip interconnection networks, *Proc. 38th Design Automation Conf.*, June 2001, pp. 684–689.

[8] B. Cordan, An efficient bus architecture for system on chip design, *IEEE Custom Integrated Circuits Conf.*, May 1999, pp. 623–626.

[9] W.J. Dally and H. Aoki, Deadlock-free adaptive routing in multicomputer networks using virtual channels *IEEE Trans. on Parallel and Distributed Syst.*, April 1993, pp. 466–475.

[10] W. Dally and J. Poulton, *Digital System Engineering*, Cambridge University Press, New York, 1998.

[11] J. Duato, S. Yalamanchili, and L. Ni, *Interconnection Networks: An Engineering Approach*, IEEE Computer Society Press, Washington, D.C., 1997.

[12] P. Guerrier and A. Greiner, A generic architecture for on-chip packet-switched interconnections, *Proc. Int. Conf. on Design Automation and Test in Europe*, March 2000, pp. 250–256.

[13] R. Hegde and N. Shanbhag, Toward Achieving Energy Efficiency in Presence of Deep Submicron Noise, *IEEE Trans. on VLSI Syst.*, pp. 379–391, Vol. 8, No. 4, August 2000.

[14] R. Hegde and N. Shanbhag, Toward achieving energy efficiency in presence of deep submicron noise, *IEEE Trans. on VLSI Syst.*, pp. 379–391, Vol. 8, No. 4, August 2000.

[15] R. Ho, K. Mai, and M. Horowitz, The future of wires, *Proc. IEEE*, April 2001, pp. 490–504.

[16] J. Hu and R. Marculescu, Energy-aware mapping for tile-based NOC architectures under performance constraints, *Proc. ASP Design Automation Conf.*, Jan. 2003, pp. 233–239.

[17] F. Karim, A. Nguyen, and S. Dey, On-chip communication architecture for OC-768 network processors, *Proc. 38th Design Automation Conf.*, June 2001, pp. 678–683.

[18] B. Khailany et al. Imagine: Media Processing with Streams, IE*EE Micro* vol. 21, no. 2, pp. 35–46, 2001.

[19] S. Kumar, A. Jantsch, J. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrij, and A. Hemani, A network on chip architecture and design methodology, *Proc. IEEE Computer Society Annual Symp. on VLSI*, April 2002, pp. 105–112.

[20] D. Lackey, P. Zuchowski, T. Bednar, D. Stout, S. Gould and J. Cohn, Managing power and performance for systems on chip design using voltage islands, *ICCAD – Int. Conf. on Computer-Aided Design*, Nov. 2002, pp. 195–202.

[21] A. Laffely, J. Liang, P. Jain, N. Weng, W. Burleson, and R. Tessier, Adaptive systems on a chip (aSoC) for low-power signal processing, *35th Asilomar Conf. on Signals, Syst., and Comput.*, Nov. 2001, pp. 1217–1221.

[22] P. Lieverse, P. van der Wolf, K. Vissers, and E. Deprettere, A methodology for architecture exploration of heterogeneous signal processing systems *J. VLSI Signal Process. for Signal, Image and Video Technol.*, Vol. 29, No. 3, pp. 197–207, 2001.

[23] E. Nilsson Design and implementation of a hot-potato switch in a network on chip, M.S. thesis, Department of Microelectronics and Information Technology, Royal Institute of Technology, Stockholm, Sweden, June 2002.

[24] P. R. Panda, N. D. Dutt, A. Nicolau, F. Catthoor, A. Vandecappelle, E. Brockmeyer, C. Kulkarni, and E. de Greef, Data memory organization and optimizations in application-specific systems, *IEEE Design and Test of Comput.*, Vol. 18, No. 3, May–June 2001.

[25] A. Pinto, L Carloni, and A. Sangiovanni-Vincentelli, Constraint-driven communication synthesis,' *Design Automation Conf.*, June 2002, pp. 783–788.

[26] L. Shang, L.-S. Peh, and N.K. Jha, Dynamic voltage scaling with links for power optimization of interconnection networks, *HPCA — Proc. Int. Symp. on High-Performance Computer Architecture*, Anaheim, CA, February 2003, pp. 91–102.

[27] J.P. Singh, W. Weber, and A. Gupta, SPLASH: Stanford parallel applications for shared-memory *Computer Architecture News*, Vol. 20, No. 1, March 1992, pp. 5–44.

[28] D. Sylvester and K. Keutzer, A global wiring paradigm for deep submicron design, *IEEE Trans. on CAD/ICAS*, Vol. 19, No. 2, February 2000, pp. 242–252.

[29] T. Theis, The future of Interconnection Technology, *IBM J. Res. and Dev.*, Vol. 44, No. 3, May 2000, pp. 379–390.

[30] J. Walrand and P. Varaiya, *High-Performance Communication Networks*, Morgan Kaufman, San Francisco, 2000.

[31] M. Wolfe, *High-Performance Compilers for Parallel Computing*, Addison-Wesley, Reading, MA, 1995.

[32] S. Winegarden, A bus architecture centric configurable processor system, *IEEE Custom Integrated Circuits Conf.*, May 1999, pp. 627–630.

[33] F. Worm, P. Ienne, P. Thiran, and G. De Micheli, An Adaptive low-power transmission scheme for on-chip networks, *ISSS, Proc. Int. Symp. on System Synthesis*, Kyoto, Japan, October 2002, pp. 92–100.

[34] R. Yoshimura, T. Koat, S. Hatanaka, T. Matsuoka, and K. Taniguchi, DS-CDMA wired bus with simple interconnection topology for parallel processing system LSIs, *IEEE Solid-State Circuits Conf.*, January 2000, pp. 371.

[35] T.T. Ye, L. Benini, and G. De Micheli, Packetized on-chip interconnect communication analysis for MPSoC, *Proc. on Design Automation and Test in Europe*, March 2003, pp. 344–349.

[36] H. Zhang, V. George, and J. Rabaey, Low-swing on-chip signaling techniques: effectiveness and robustness, *IEEE Trans. on VLSI Syst.*, Vol. 8, No. 3, pp. 264–272, June 2000.

[37] H. Zhang, M. Wan, V. George, and J. Rabaey, Interconnect architecture exploration for low-energy configurable single-chip DSPs, *IEEE Computer Society Workshop on VLSI*, April 1999, pp. 2–8.