# Computation and Communication Refinement for Multiprocessor SoC Design: A System-Level Perspective

RADU MARCULESCU, UMIT Y. OGRAS, and NICHOLAS H. ZAMORA
Carnegie Mellon University

Continuous advancements in semiconductor technology enable the design of complex systems-on-chips (SoCs) composed of tens or hundreds of IP cores. At the same time, the applications that need to run on such platforms have become increasingly complex and have tight power and performance requirements. Achieving a satisfactory design quality under these circumstances is only possible when both computation and communication refinement are performed efficiently, in an automated and synergistic manner. Consequently, formal and disciplined system-level design methodologies are in great demand for future multiprocessor design. This article provides a broad overview of some fundamental research issues and state-of-the-art solutions concerning both computation and communication aspects of system-level design. The methodology we advocate consists of developing abstract application and platform models, followed by application mapping onto the target platform, and then optimizing the overall system via performance analysis. In addition, a communication refinement step is critical for optimizing the communication infrastructure in this multiprocessor setup. Finally, simulation and prototyping can be used for accurate performance evaluation purposes.

Categories and Subject Descriptors: B.3.3 [**Memory Structures**]: Performance Analysis and Design Aids; C.4 [**Computer Systems Organization**]: Performance of Systems; C.5.4 [**Computer System Implementation**]: VLSI Systems; B.7.1 [**Integrated Circuits**]: Types and Design Styles—*VLSI (very large scale integration)*

General Terms: Design, Performance

Additional Key Words and Phrases: Embedded systems, energy optimization, performance analysis, Markov chains, communication, traffic, systems-on-chip, networks-on-chip, prototype

## 1. INTRODUCTION

Unrelenting advancements in semiconductor technology enable the integration of a myriad of intellectual property (IP) blocks on the same chip. General purpose CPUs, DSPs, ASICs, I/O and networking devices, and shared embedded

This research was supported by Marco GSRC, SRC 2004-HJ-1189, NSF CCR-00-93104.
Authors' address: R. Marculescu, U. Y. Ogras, N. H. Zamora, Department of Electrical and Computer Engineering, Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213; email: {radum,uogras,nhz}@ece.cmu.edu.

memory modules on a single chip are now not only possible but necessary to maintain competitive advantage. Along with these technological advancements, we are witnessing an explosive growth in demand for consumer electronics functionality. These demands result in a push to design portable devices and appliances with only a few months in time-to-market constraints.

Systems-on-chips (SoCs) implementing text, speech, and video processing applications are becoming overwhelmingly complex, with digital and analog parts coexisting on the same chip. This trend puts enormous pressure on tool designers to integrate all these parts seamlessly in terms of system synthesis, verification, simulation, and testing. Complementary techniques for system validation, including formal methods and system-level design tools, are therefore crucial in the design of complex embedded systems.

Speaking in general terms, a primary goal of system-level design is to minimize the development time and nonrecurrent engineering costs, subject to constraints on performance and functionality of the system. To succeed, both module reuse and system flexibility are key components. At the very heart of the system-level approach is a set of models derived for both applications and platforms, and techniques that can be used to optimize the system at hand. This approach is known as the Y-Chart scheme [Lieverse et al. 2001] (Figure 1).

Referring to Figure 1, the *application models* include the workload characterization which is typically expressed in probabilistic terms. These application models must be scalable and flexible enough for quick analysis. Furthermore, it is also crucial to capture the key behavior of the application in the model in order to have confidence in the predicted results. For instance, the application considered herein (namely, MPEG-2 video) is characterized by soft real-time constraints; this implies that occasionally missing deadlines is perfectly acceptable. In real-time systems where the behavior is characterized by hard real-time constraints (e.g., automotive or safety-critical systems) a different set of challenges is posed, both in modeling and analysis; these issues are covered in a companion paper [Eles and Pop, this issue].

From an implementation perspective, the term *platform* is used for a family of heterogeneous architectures that need to satisfy a set of architectural constraints imposed to allow reuse of hardware and software components [Ferrari and Sangiovanni-Vincentelli 1999]. The platform description may come with some low-level information from designers, depending on the targeted level of accuracy for this type of evaluation. For instance, a platform model can abstract away the cache or other memory effects in order to keep the complexity of the model under control. This is very much the case in Section 2.1, where we focus primarily on the possible interactions between different parts of the application, buffer occupancy, etc. At the same time, the platform model can emphasize communication (as opposed to computation) aspects where communication volume, packet rates, buffer size, etc., represent the information of interest. We make such a distinction, for instance, in Section 3.1, where we try to capture the impact of topology on the overall behavior of the network. Ultimately, from the platform perspective the models we build need to enable accurate analysis in a relative sense, but also need to be simple and flexible enough to allow quick

modifications. For instance, to facilitate a systematic design space exploration the platform models should support a wide range of programmable IPs, including fixed processor cores, configurable processors, and reconfigurable fabrics [Mishra et al. 2006].

Returning to Figure 1, once the appropriate models for the application and platform become available, the application is *mapped* onto the target architecture and performance analysis is used to determine whether or not the chosen application-architecture combination satisfies the imposed design constraints. Several formalisms (e.g., queueing networks, Petri nets, process algebra, etc.) can be used to carry out this analysis step. In this survey, we discuss a Markovian approach based on stochastic automata networks (SANs) to determine in a systematic manner the possible power-performance tradeoffs in an MPEG-2 application. We also compare the results of the SAN analysis with those obtained when self-similarity is taken into account. These effects are quite important, particularly when the buffer size is critical; this is precisely the case, for instance, for portable devices. If the design constraints are not met, then the mapping process is reiterated with a different set of parameters until the desired result is obtained. From this perspective, the performance analysis needs to be tractable to allow for cutting down time in the design cycle. This enables fast design space exploration and finding good quality solutions within a short and predictable time budget.

Once the results obtained at this level of abstraction are satisfactory, we need to consider the communication refinement step. This has become increasingly important in recent years due to the richness of on-chip computational devices which places tremendous demands on the communication resources as well. While traditional bus-based or point-to-point (P2P) connections are the default choices, new communication architectures have been recently proposed for SoC design to provide truly scalable communication capabilities. From this perspective, the network-on-chip (NoC) architecture, where regularity is enforced, has the large advantage of providing more predictability and better scalability (both in performance and power consumption) compared to "classical" communication schemes. In this article, we discuss several issues related to communication infrastructure design and optimization, as well as the effects of the communication paradigm on overall network behavior. Interestingly enough, the application mapping process in the NoC setup mirrors the mapping process in the processor-bound context, although this time the set of constraints is completely different. Again, performance analysis is at the very heart of the overall optimization process.

Referring to the methodology in Figure 1, we note that simulation and often prototyping are used for performance evaluation purposes. While the design automation community is very familiar with simulation (e.g., Spice, Cadence, Matlab, etc.) as an effective means for design space exploration, simulation (and prototyping) cannot offer guarantees about the design at hand. For example, simulating an architecture which executes a single application can give the designer a good idea of what to expect in terms of system response and device utilization. However, it is difficult in most simulation frameworks to identify what the worst-case system response (runtime) is for that particular

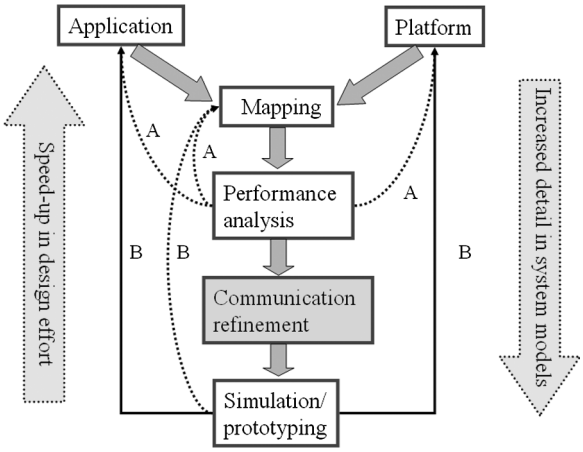


Fig. 1. The Y-Chart scheme. Design loops taking paths marked *A* take little design effort compared to loops marked *B* and therefore can be iterated many more times.

architecture. Further, both simulation and prototyping can be very expensive to achieve and using them in an optimization loop is practically impossible due to the excessive run and design times. Consequently, it is critical to integrate performance analysis into the early stages of system design. Designers may be less familiar with various analysis formalisms, but being primarily based on stochastic models, such tools compute the metrics of interest in times which are orders of magnitude faster than simulation (typically, they range from a few seconds to tens of minutes, depending on the type and complexity of the model used). As such, reiterating at the end of the performance analysis step (loop *A* in Figure 1) is much more desirable than doing it after simulation or prototyping.

In the remainder of this article, we address application and platform modeling, application to architecture mapping and performance analysis in Section 2. This is the logical sequence of steps in any rigorous system-level design methodology. Next, we address some issues regarding the communication refinement and performance evaluation steps and present their impact on overall system performance in Sections 3 and 4, respectively. Indeed, while selecting a platform for a target application involves extensive analysis, as on-chip multiprocessor systems gain widespread use, we need to focus more on the communication capabilities of possible HW platforms. Finally, we summarize our main ideas in Section 5.

## 2. MODELING, MAPPING, AND PERFORMANCE ANALYSIS

The growing interest in designing multiprocessor systems on the same chip brings concurrency and communication to the forefront of the design process. To begin with, the *system description* can be initially given in an informal manner, as outlined in Figure 2 [Gotz et al. 1993]. This informal specification needs to be later translated into a formal description (e.g., finite state machines, process algebra, etc.) which is precise in meaning and amenable to performance

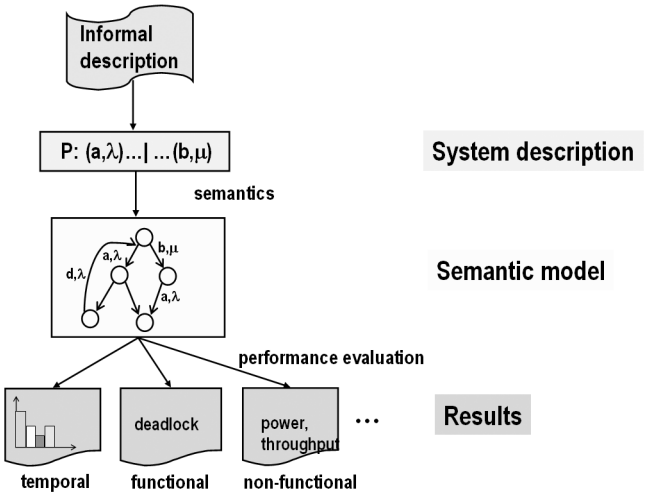


Fig. 2. A semantic model is created from an informal description of an application by obtaining stochastic rates which characterize the application. The model can then be analyzed to obtain different metrics which are not easy to obtain using simulation techniques alone.

analysis. Usually, such a representation is based on states and transitions among states which are caused by various actions in the system. The *formal* model should represent an abstraction of the system behavior which is sufficiently accurate yet tractable for analysis and verification purposes [Hillstone 1996]. The idea is to build a model at the highest level of abstraction, free of any architectural constraints, which captures the entire parallelism available at application-level. This model needs to capture the essential properties of the application (such that its performance can be accurately evaluated) assuming infinite physical (i.e., computation- and communication-bound) resources.

Once this *functional* modeling part is completed, a formally defined semantics is used to obtain a *quantitative* model that captures the dynamic properties of the system. This new semantic model typically exposes the flow and control data between system components, so it can be used to extract performance metrics that can guide the system optimization process in a meaningful way. Translating the functional model into a quantitative one may involve obtaining worst-case conditions or stochastic rates for application behavior based on the expected input workload. Estimating such parameters inherently introduces uncertainty and modeling errors when compared to real behavior. Therefore, knowing which behaviors should be included in the model and how to estimate these behaviors is critical to maintaining an accurate performance model.

Concurrent systems can be modeled in a variety of ways each providing different tradeoffs in terms of specification flexibility and convenience for performance evaluation. For example, queueing theory [Trivedi 1982] provides a compact notation at the expense of limited expressiveness of the language. The customers representing jobs in the system flow through the network of service centers (i.e., queues) connected together in a network. The parameters of interest are the arrival processes to the queues, the service rates, the capacity of the

buffers, and the queueing discipline (e.g., first-come-first-serve). Compared to queueing networks, Petri nets [Peterson 1981] can represent a larger class of systems, although they face a state explosion problem even for relatively simple systems.

Unlike queueing networks or Petri nets, process algebras such as the calculus of communicating systems (CCS) [Milner 1989] and the algebra of communicating systems [Bergstra and Klop 1985] characterize the system by its active components and the communication between them in a compositional manner. Compositional approaches model a system as a collection of smaller, more manageable components. Therefore, *compositionality* is a crucial strategy for tackling complex applications in a systematic way. At the same time, in order to simplify the performance analysis, most performance models assume that each action in the system is characterized by an exponential distribution determining its duration. In other words, when enabled, an activity with rate $r$ will delay for a period determined by the distribution $1 - e^{-rt}$. This is a very important assumption, as it allows the use of powerful results from Markov chain theory for the performance analysis of concurrent systems. Other choices for the distribution may lead to semiMarkov or more general models, but the mathematical results available in such cases are significantly more complicated.

Another formalism that has been proposed to model a set of communicating processes for streaming applications is data flow process networks, such as Kahn process network (KPN) [Lieverse et al. 2001; Kahn et al. 1974; Lee and Parks 1995] and synchronous data flow (SDF) [Lee and Messerschmitt 1987; Sriram and Bhattacharyya 2001; Geilen et al. 2005]. In the KPN model of computation, the processes communicate asynchronously using P2P connections over unbounded FIFOs. On the other hand, SDFs model the processes with fixed data generation and consumption rates. SDFs can nicely evaluate worst-case situations of complex systems; this is analogous to identifying the critical path in a processor design. Care should be exercised, however, since worst-case times could be one or two orders of magnitude larger than the actual values. As pointed out in Kim and Shin [1996], while scheduling tasks based on worst-case execution times can guarantee meeting their timing requirements, it may lead to severe underutilization of CPUs. For this reason, SDFs should be used in a design optimization loop only if the worst-case behavior of an application is of primary importance.

Our approach towards performance analysis is based on stochastic automata networks (SANs). SANs are powerful Markovian models suitable for modeling communicating processes which appear naturally in highly concurrent applications [Plateau et al. 1991; Stewart et al. 1995]. Later, we show how the application and platform models can be used together in the mapping step (Figure 1) to analyze different power-performance tradeoffs.

### 2.1 Stochastic Analysis

SANs model applications and architectures at process-level, where the processes communicate and interact such that they define what the underlying

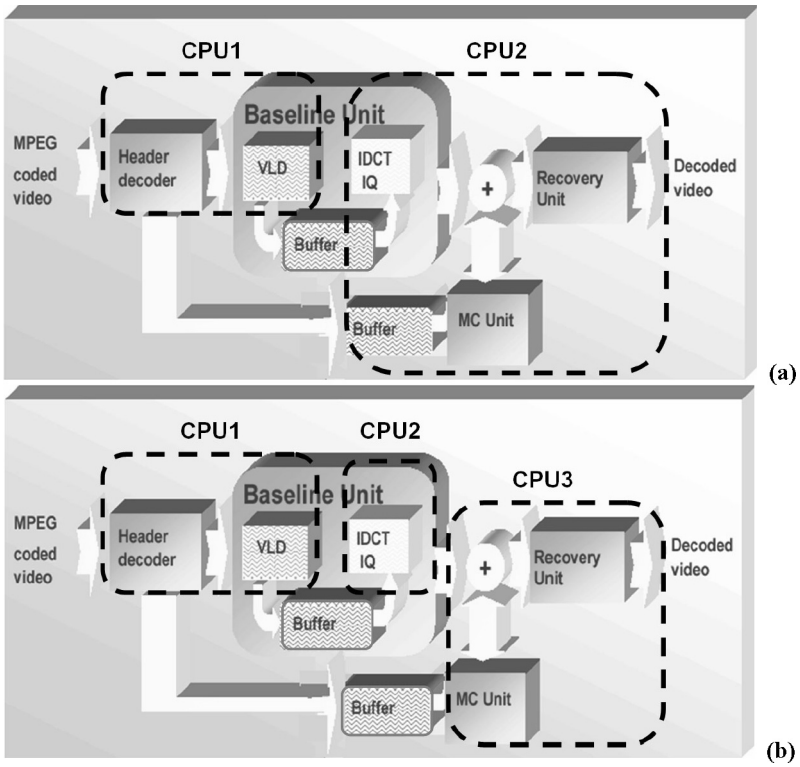


Fig. 3. The implementation of an MPEG-2 decoder using (a) a two-processor and (b) a three-processor platform.

system should do without a precise idea of how it is going to be implemented. The objective of SAN analysis is the computation of the stationary probability distribution to a target system described as stochastic automata that operate more or less independently. The interaction between the automata is captured by using both *synchronizing* and *functional* transitions. A synchronizing transition in one automaton is a transition which forces a transition to occur in one or more other automata simultaneously. Synchronizing events affect the global system by altering the state of at least one automaton. Functional transitions, on the other hand, are nonzero transition rates for one automaton when another automaton is in a particular state. As opposed to constant-rate, or non-functional, transitions, the functional rates in one automaton are affected by the state of another automaton, and so these transitions can model interactions effectively. Functional transitions affect the global system only by changing the rate of state changes of a single automaton. For more details on SANs, the reader is referred to Plateau and Fourneau [1991] and Stewart [1994].

2.1.1 *Application and Platform Modeling.* To make the discussion more concrete, we consider an MPEG-2 decoder application (Figure 3) which consists of a baseline unit, a motion compensation (MC) unit, a recovery unit, and the associated buffers. In turn, the baseline unit consists of a VLD (variable

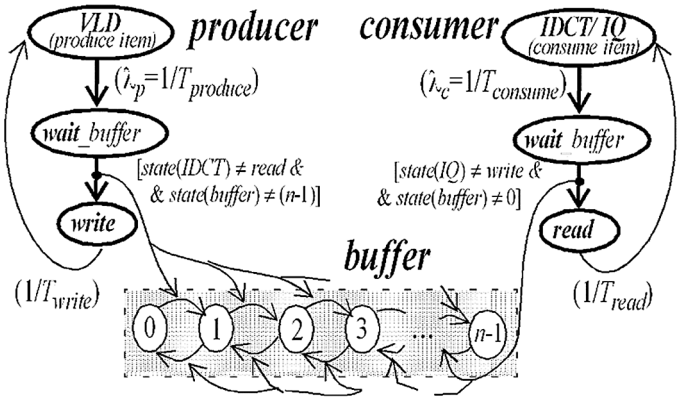


Fig. 4. MPEG-2 application. The application models are void of any architectural details (Figure taken from Nandi and Marculescu [2001]).

length decoder), an IQ/ IZZ (inverse quantization/inverse zigzag) module, IDCT (inverse discrete cosine transform) modules, and the buffer.

Suppose now that we want to explore different possible implementations of this application running on a multiprocessor system with two or three CPUs, as shown in Figure 3. Using the SAN analysis, we need to determine the system steady-state regime by calculating the state probabilities associated with the main modules, the buffers' occupancy, average power consumption, etc. To this end, the first step is to create a semantic model for the target application using the SAN formalism, as shown in Figure 2. These models should be void of any architectural details of the target implementation platform. The application is modeled as a process graph with concurrently active processes which communicate and interact with each other. This process graph translates into a network of automata, while the communication between processes translates to synchronizing transitions. Figure 4, for instance, illustrates a simple *producer-consumer* SAN model for the VLD and IDCT blocks in an MPEG-2 decoder. It is very important to note that, under the assumption of exponentially distributed sojourn times, this network of automata is characterized by an identical Markov chain [Stewart 1994]. This is a very important property, since it enables the use of powerful results from Markov chain theory for performance analysis of concurrent systems.

Working in parallel with designing the application models, the system-level designer can build models that represent abstract behavioral descriptions of the architectural building blocks. These blocks typically consist of several computational resources such as programmable cores or dedicated hardware units, communication resources such as high-performance shared buses, and memory resources such as RAMs memory modules and FIFO buffers. The details of the application are omitted from the architectural models [Nandi and Marculescu 2001]. As such, the SAN model provides the highest level of abstraction and is not confined to any particular hardware-software combination. This design philosophy enjoys the benefits of what is called the *separation of concerns* between application and architecture [Ferrari and Sangiovanni-Vincentelli 1999].

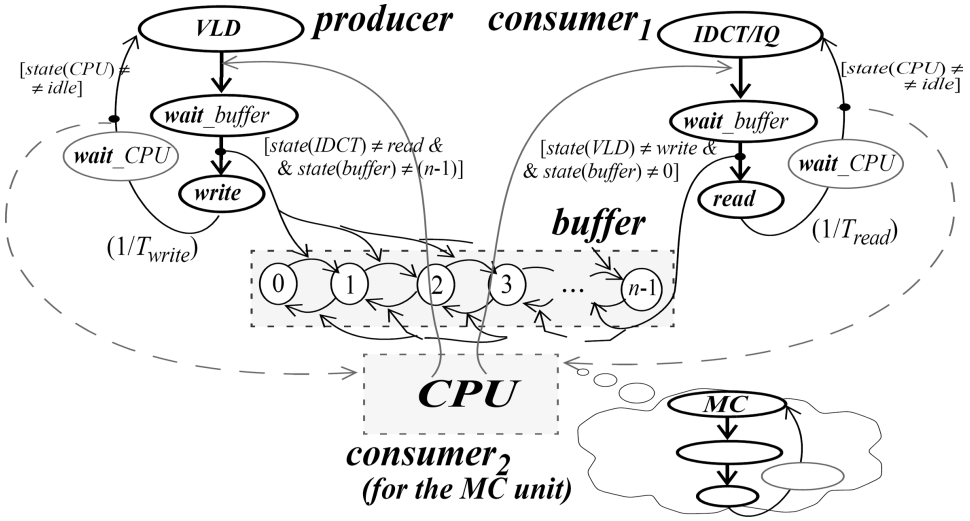


Fig. 5. Example mapping of MPEG-2 application onto a single CPU architecture. New wait states and synchronization signals have been added due to the limitations of HW resources (Figure taken from Nandi and Marculescu [2001]).

This separation enables reuse of both application and architecture models and facilitates an exploratory design process in which application models are subsequently mapped onto architectural models [Yang et al. 2004].

2.1.2 *Application Mapping for Multiprocessor Platforms.* Having the system-level application and platform models available enables the designer to map the application onto a family of architectures (platform) and evaluate the system performance. This evaluation indicates whether the choice of the platform and set of design parameters satisfies the required level of design quality, which could be measured in terms of performance figures, power consumption, fault-tolerance, etc., as shown in Figure 2. Furthermore, if necessary, the mapping process can be reiterated with a different set of parameters and/or target architectures (using design loops marked with *A* in Figure 1) until convergence to the desired design quality.

One way to do the mapping is to add synchronization transitions between the application and architecture models and modify the transition rates of the application based on the actual capabilities of the architectural components. Mapping our simple *VLD-IDCT/IQ* processes in Figure 4 onto a platform with a single CPU is illustrated in Figure 5. Because these processes[1] now have to share the same CPU, some of the local transitions become synchronizing transitions (e.g., the local transitions with rates $1/T_{\text{produce}}$ or $1/T_{\text{consume}}$ become synchronized). Moreover, some new states (e.g., *wait_CPU*) have been introduced to model the new synchronization relationship [Nandi and Marculescu 2001]. This mapping method has the advantage of retaining flexibility, as the

[1]For simplicity, the second consumer process for the Motion Compensation (MC) unit was not explicitly represented in this figure.

architecture and application models still remain separate from each other. This allows the designer to painlessly replace current models with a new application or architectural model in the future.

Once the models are built and the mapping is complete, the application-architecture model is finally analyzed for different input parameters (this is the "Performance analysis" step in Figure 1). While model evaluation is a challenging problem, *analytical* performance evaluation presents additional challenges. To this end, the SAN formalism can be used to build an analytical framework where lengthy profiling simulations for predicting power and performance figures can be avoided. In essence, the SAN analysis involves solving the equation $\pi Q = 0$, where $\pi$ is the steady-state probability distribution that needs to be found via analysis, and $Q$ is the global generator matrix which includes all local and synchronizing transition rates for each automaton executing concurrently. Together with this equation, a key constraint is that the probabilities for being in every possible state must add up to 1. This is expressed as the normalization equation $\pi e = 1$, where $e$ is a column vector such that $e^T = (1, 1, \ldots, 1)$. Determining steady-state probabilities we can ascertain, for example, how utilized a communication buffer or CPU is for a particular application-architecture combination and input trace characteristics [Nandi and Marculescu 2001].

SAN analysis is particularly valuable for multimedia systems where many simulation runs are typically required to gather relevant statistics for average-case behavior. Considering that 5 minutes of compressed MPEG-2 video needs roughly a few Gbits of input vectors to simulate, the impact of having an analysis tool to quickly provide performance estimates becomes evident. Another major advantage of SANs over other formalisms, such as Petri nets, is that the state space explosion problem associated with Markov models is partially mitigated by the fact that the global generator matrix $Q$ is not stored, nor even generated [Plateau and Fourneau 1991; Stewart 1994]. Instead, the state transition information is represented by a number of smaller matrices (one for each automaton belonging to the system) which are combined using tensor algebra. By using vector matrix multiplications, all relevant information is determined without explicitly forming the global matrix.

In summary, this kind of stochastic analysis allows media systems designers to explore architectures more rapidly and to estimate the impact of different design choices more robustly. Ultimately, this enables systems designers to optimally trade off performance metrics and multimedia quality.

### 2.2 Rate Analysis

While SAN analysis is essentially an average-case approach, rate analysis provides bounds for worst-case conditions [Maxiaguine et al. 2004]. For example, depicted in Figure 6 is the same multimedia application where the input stream enters a processing element (PE), gets processed by the PE, and is then transmitted to another PE over a communication channel for further processing. A feasible mapping of this multimedia application onto architecture should ensure that each buffer between the PEs does not overflow and that the play-out buffer, which is read by the real-time client, does not underflow at any point

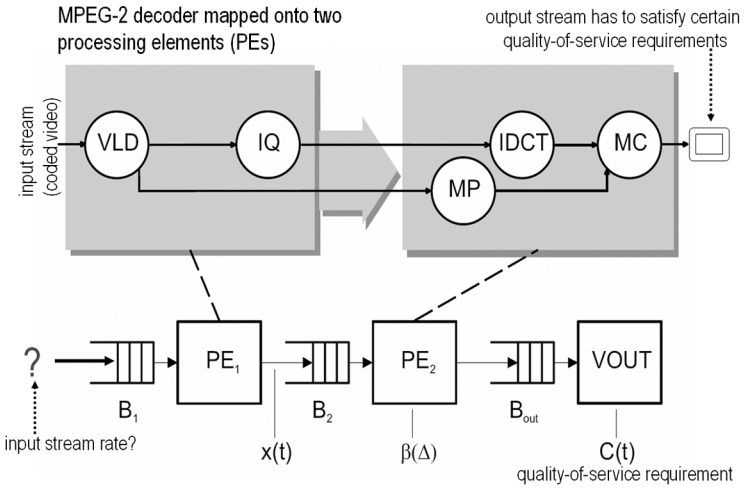


Fig. 6. Illustration of the rate analysis problem using the MPEG-2 decoder. The input stream near the bottom of the figure is difficult to predict, and so a rate analysis mechanism is needed (Figure taken from Maxiaguine et al. [2004]).

in time. The bursty nature of the input stream and the variation in execution time of stream processing applications make the input rates hard to predict.

Referring to Figure 6, the problem at hand is to compute the feasible rates of the input streams, given the output consumption rate of a stream (e.g., by the video output device), the service offered by the two processors (determined by the frequency of the processors and the scheduling policy implemented on them), and the properties of the streams (processor cycle requirement for each macroblock and its variability). If the input rates are too high, the buffers will overflow. On the other hand, too low a rate will lead to play-out buffer underflow and hence, loss of video output quality. Therefore, there is only a specific range of inputs to consider and this range is a function of the service provided by the processors and buffer sizes. The rate analysis problem is important since it helps designers to choose a suitable architecture and to determine the output quality that can be supported.

## 2.3 Traffic Analysis

The SAN approach presented in Section 2.1 is based on a Markovian model for probability distributions. While the Markovian assumption is a convenient mathematical simplification, it is not always applicable in practice, particularly for video applications. For example, it can be shown that the multimedia traffic generated at macroblock-level exhibits self-similar (or long-range) phenomena [Beran 1994]. This means that the correlations between the processed macroblocks do *not* necessarily decay exponentially, but rather obey a power law characterized by the Hurst parameter. As such, if used for buffer sizing, Markovian models would not produce accurate results.

To be concrete, Figure 7 shows a possible mapping of the IP cores which implements the MPEG-2 decoder using a platform where communication among the

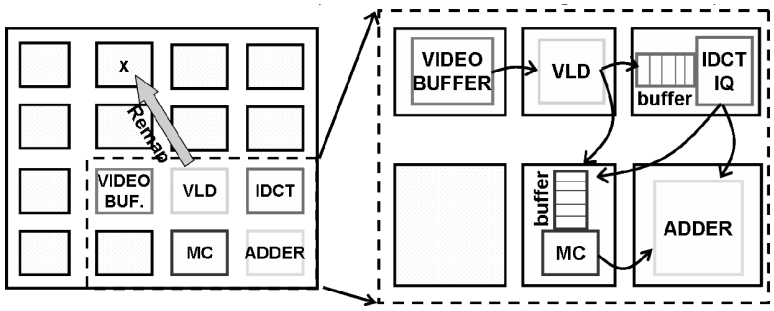


Fig. 7. Mapping of an MPEG-2 decoder to a NoC platform such that the cores that communicate with each other are placed into neighboring tiles (Figure taken from Varatkar and Marculescu [2004]).

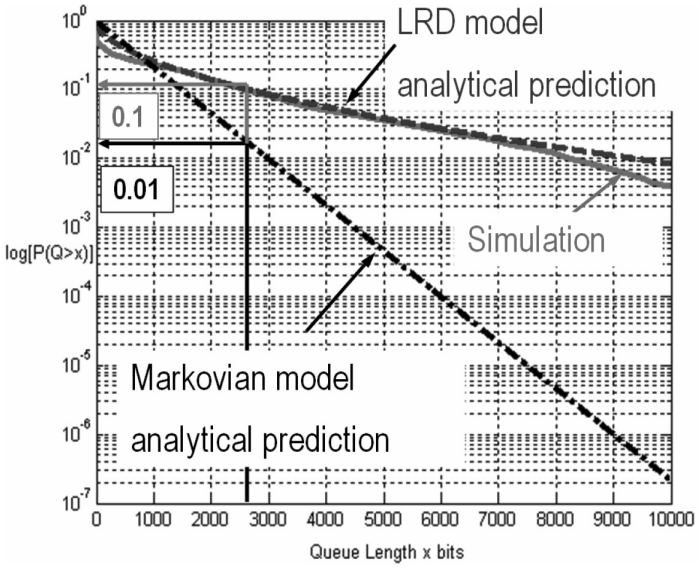


Fig. 8. The variation of packet dropping probability (log scale) as a function of the queue length (linear scale) estimated by a Markovian model, an LRD model, and simulation (Figure taken from Varatkar and Marculescu [2004]).

IPs is realized via the NoC approach. The analysis of multiple traces taken between the VLD-MC and VLD-IDCT/IQ modules reveals that the traffic exhibits long-range dependencies, thus, the rate of the autocorrelation function decays much more slowly than the exponential decay [Varatkar and Marculescu 2004].

Since self-similar processes are so different from Markovian processes, this observation has significant consequences for the design of multimedia systems. For example, using a Markovian analysis technique to determine the size of the buffers highlighted in Figure 7 will significantly underestimate buffer overflow probabilities. This is because the actual variance of the network traffic is much larger than predicted by Markovian models.

In Figure 8, the Y axis gives the probability of overflowing a buffer $Q$ as a function of its size. The precise mathematical relationship was derived by

Norros [1994]; it relates the buffer overflow probability to the Hurst parameter which characterizes the burstiness in the data. For instance, if the buffer size is chosen arbitrarily to be 2.8K, then Markovian analysis would predict a buffer overflow probability of about $10^{-2}$. On the other hand, nonMarkovian analysis based on the Norros formula would predict an overflow probability about one order of magnitude higher, which matches the simulation results very well, as can be observed in Figure 8. Sizing the buffer based on such an erroneous (former) prediction would drastically affect the capability of the video decoder, since the buffer will overflow too often to maintain a reasonable quality of the video. For this reason, it is important to investigate nonMarkovian analysis techniques and find suitable ways to deal with the video traffic affected by long-range dependencies.

To summarize, formal approaches for modeling and analysis are important in system-level design. Despite many years of research, building and solving the stochastic model remains the main challenge. The models used in system-level analysis should be aimed at the highest level of abstraction; they should exploit the separation of concerns between function-architecture, and between computation-communication. If successful, system-level analysis can provide quick feedback to the designer about possible power-performance tradeoffs. The results of this analysis can be used at the microarchitectural-level of abstraction for further optimization.

## 3. COMMUNICATION REFINEMENT

In its most abstract form, the application can be modeled as a set of communication processes free of any architectural constraints, as illustrated in Figure 9(a) and discussed in Section 2. As we move the level of abstraction down, the communication mechanism needs to be taken into account since the behavior of the processes depends on the behavior of the communication channel. More specifically, the behavior and constraints imposed by the platform need to be integrated into the system model. For example, communication can take place over an ideal channel connecting two processes with finite buffers, as shown in Figure 9(b). A concrete instance of such a communication scenario would be a bus architecture, or two processes communicating over a Point-to-Point (P2P) link. We can lower the level of abstraction further down by considering a more detailed model for the channel itself. For instance, the communication can take place over a lossy wireless channel, as shown in Figure 9(c), where the error model plays a significant part. The important thing is that the communication refinement process makes the effects of the communication mechanism and constraints imposed by the available resources more salient.

A system-level analysis for platforms that are designed to support portable encoding/decoding devices that interact and communicate over a wireless channel is presented in Marculescu et al. [2001]. The technique models all components, namely, encoder, decoder, and communication channel, using the SAN approach. This allows us to estimate the degradation in performance (i.e., QoS) seen at the decoder-level as a function of channel behavior and enables analytical calculations of power-performance metrics which are significantly faster

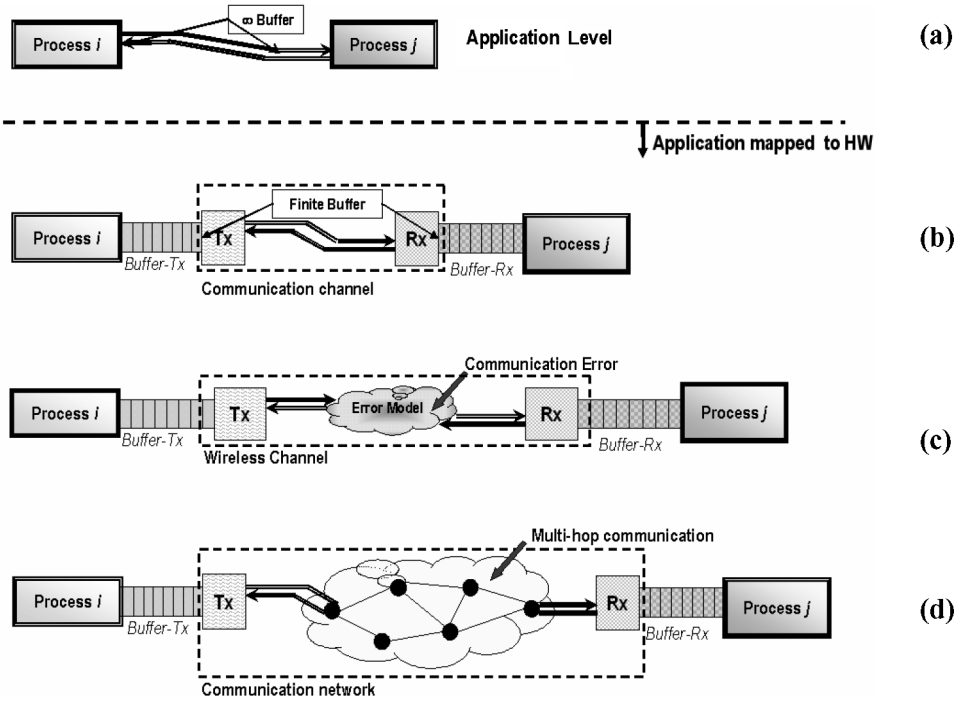


Fig. 9. (a) Simple two-process model with infinite communication buffer, (b) replacing infinite buffers with finite ones, (c) adding errors, and (d) communicating through a multihop network.

than time-consuming profiling simulations. This kind of analysis shows, for instance, that the communication may become too costly for some encoding rates; this is because the design constraints that should be satisfied become too severe in terms of power consumption and/or end-to-end latency. Under such conditions, a data encoding rate cannot be chosen arbitrarily. On the contrary, the affordable rate is imposed by channel behavior. Obviously, determining this data rate using simulations is extremely difficult.

All the communication channels discussed so far (see Figure 9(a)–(c)), model P2P links. This means that the sender and receiver are directly connected to each other via a communication channel. A more general case to consider is that of a *network* connecting several processes. For instance, the process $i$ on the left side of Figure 9(d) can be an application which accesses a remote database, modeled by process $j$, over a multihop (wired or wireless) network. Another example would be to consider the *VLD* core in Figure 7 as being remapped to an arbitrary tile labeled $X$; then, the communication between the *VLD* and *IDCT* modules needs to take place across a multihop network, similar to the scenario shown in Figure 9(d).

The most general case corresponds to having a set of nodes (which encapsulate several communicating processes) connected in a network (Figure 10). Under such a scenario, multiple distributed applications can run concurrently while sharing the same network. Hence, different processes from the same application can communicate across the network, as discussed in the previous

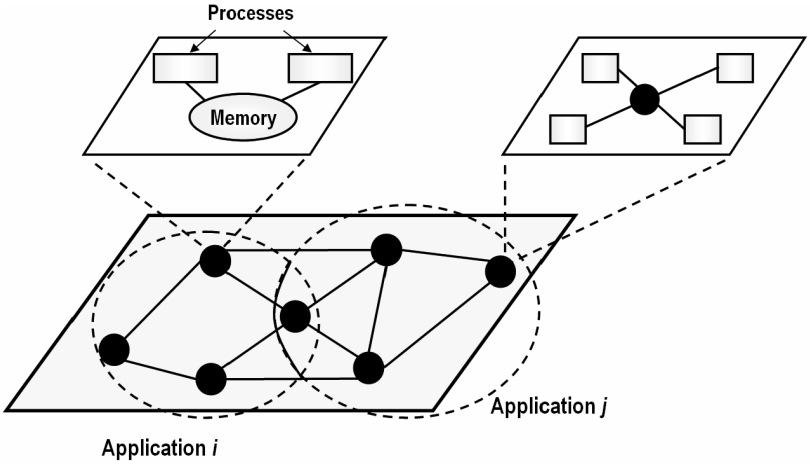


Fig. 10. An on-chip network running two applications. Each node in the network executes several processes in parallel.

case. If we compare Figures 5, 9, and 10, we can clearly see the increasing importance of the communication part and its subtle relationship with computation. Indeed, refining the communication medium from the simple buffer in Figure 5 to the multihop network in Figure 9 (or 10) changes the nature of the design problem [Sangiovanni-Vincentelli et al. 2006]. Moreover, the power and performance figures of the system depend critically on the efficiency of the communication infrastructure. For this reason, *communication-centric design* plays an increasingly important role in system-level design; this is the focus of the following sections.

### 3.1 Communication Infrastructure Modeling and Optimization

Traditionally, bus-based communication has been the most common choice for both off-chip and on-chip designs. Indeed, industrial standards such as AMBA [Flynn 1997], CoreConnect [IBM 2006], and OCP [Open Core Protocol International Partnership 2006] have been successfully deployed in a wide range of systems. This is mainly due to the low costs and simple, well-understood concepts that govern bus-based design. Unfortunately, the buses are not scalable. Each new component connected to the bus shares the same limited bandwidth and adds more parasitics which deteriorate the interconnect delay; this becomes a major problem in the nanoscale domain. Furthermore, the power-hungry nature of buses makes them unsuitable for portable and other low-power applications.

On the other hand, P2P communication architectures bring the utmost of communication performance by providing dedicated channels to all communicating IP pairs. However, P2P architectures do not scale well due to their large number of communication interfaces and dedicated links. In contrast to these methods, the NoC approach emerged as a promising solution to on-chip communication problems [Hemani et al. 2000; Dally and Towles 2001; Benini and De Micheli 2002; Jantsch and Tenhunen 2003].

NoC architectures consist of heterogeneous cores such as CPUs, DSPs, video processors, embedded memory blocks, and application-specific components. The cores are connected through an interconnection network. Hence, communication between the nodes is achieved by routing packets rather than wires. As a result, NoC represents a highly scalable communication architecture. Moreover, the modularity and standard network interfaces of the NoC approach facilitate reusability across many different applications, and thus, make NoC an attractive platform for SoC integration.

Formal descriptions of NoC architectures, as well as analysis methodologies similar to the application models presented in Section 2, are needed to tackle the NoC design problem. For example, effective performance and energy models are essential, for optimization purposes, where long simulation times cannot be tolerated. A layered protocol stack has been successfully applied in the context of macronetworks to expedite the network design [Bertsekas and Gallager 1987]. While some ideas from this domain can be transferred to NoCs, it should be noted that, as opposed to NoCs, macronetworks aim at general purpose communication. As such, several fundamental problems in NoC design (e.g., application mapping, application-specific topology optimization, etc.) simply do *not* appear in the macronetwork domain; this is because the macronetwork design is typically decoupled from driver applications. Therefore, a formal description and specific algorithms reflecting the NoC design space can point out salient issues and expedite research towards finding and/or improving solutions to the existing problems [Ogras et al. 2005].

For a given class of applications, the NoC architectures have the potential to be fully structured, general purpose, or customized. The first enables preoptimized interconnections and well-controlled electrical parameters. On the other hand, homogeneous structures may not satisfy the strict requirements of heterogeneous SoC traffic, which typically varies widely across different application domains. The optimization process can explicitly consider performance in a variety of contexts such as partial [Ogras and Marculescu 2005a] or complete [Ogras and Marculescu 2005b; Pinto and Sangiovanni-Vincentelli 2003; Srinivasan et al. 2004; Jalabert et al. 2004] topology customization, buffer space allocation [Hu et al. (to appear); Saastamoinen et al. 2003], network channel [Lin and Pileggi 2002; Morganstein et al. 2004] and floorplan design [Ye and De Micheli 2003]. Interestingly enough, imposing regularity constraints (e.g., mesh-based architectures) causes the topology synthesis problem to degenerate into the application mapping problem, as described in the previous section. Two relevant optimization problems, namely, network topology optimization and buffer space allocation, are discussed next.

3.1.1 *Network Topology.* Network topologies are graphs whose properties determine the information dissemination capability of the network. For instance, the diameter of a 2D mesh network grows linearly with each dimension. This results in long internode distance, hence, poor performance for large networks. On the other hand, fully customized networks, such as in Figure 11, can provide significant improvements in terms of performance, power consumption, and area.

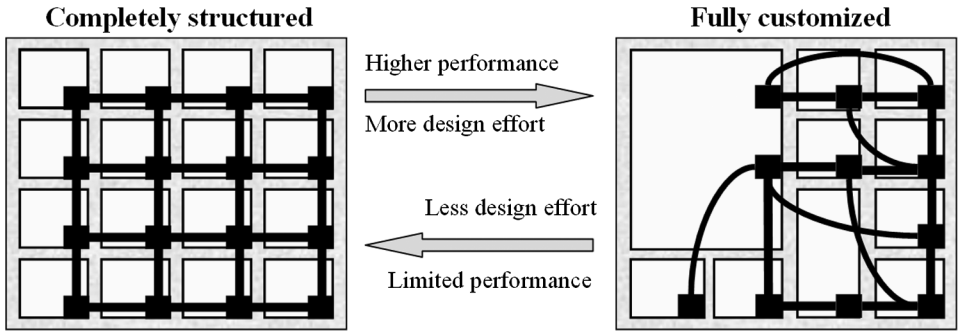


Fig. 11. Structured and fully customized communication architectures. Regular structures provide greater scalability but often worse performance, while fully customized ones require more design effort but yield better performance results.

The topology synthesis problem can be cast in very general terms. While it is true that a large number of different communication patterns can be observed in any network, certain generic communication primitives such as gossiping (i.e., all-to-all communication), broadcasting (i.e., one-to-all), and multicasting (i.e., one-to-many) are encountered more frequently. The optimal topologies for these common communication primitives have been well-studied in the past [Hedetniemi et al. 1988]. Therefore, these communication primitives can be used as an *alphabet* to characterize any given communication patterns in real applications [Ogras and Marculescu 2005b]. Hence, solving the communication synthesis problem reduces to decomposing arbitrary application graphs into combinations of basic communication primitives.

The decomposition step consists of identifying the subgraphs in the input graph that are isomorphic to one of the graphs (i.e., primitives) in the communication library. A sample graph decomposition is shown in Figure 12. Each path from the root of the tree to one of the leaves is a valid decomposition which has a cost proportional to the communication energy consumption of its implementation. Once a set of decompositions is found using a branch-and-bound algorithm, the decomposition with the minimum cost is selected as the solution (e.g., leftmost branch in Figure 12). Also, after the best decomposition is obtained, the basic graphs are replaced by their optimal implementations in a predefined library and finally, the customized topology is obtained by gluing them together under imposed design constraints.

A constraint-driven communication synthesis approach based on P2P communication specifications is presented in Pinto and Sangiovanni-Vincentelli [2003]. The resulting architecture consists of optimized channels obtained by merging or separating the original P2P links. In Srinivasan et al. [2004], a linear programming-based approach is proposed for synthesizing application-specific NoC architectures such that the power consumption is optimized subject to performance constrains. Fully customized topologies improve overall network performance at the expense of a penalty in the structured wiring which is nonetheless one of the main advantages offered by regular on-chip networks. A trade-off between these two extremes can be achieved by application-specific long-range

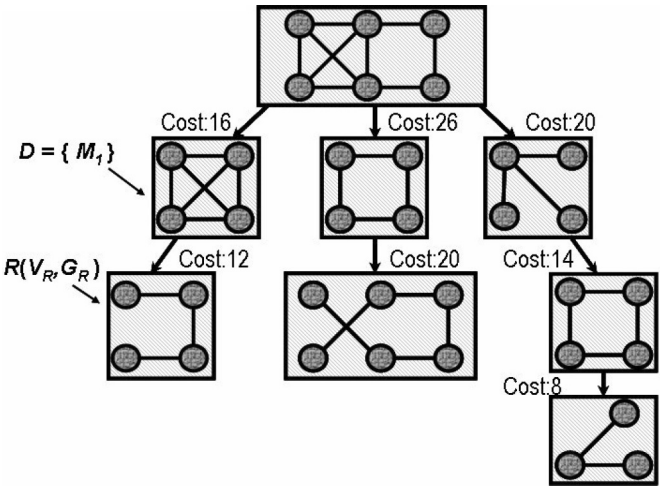


Fig. 12. Illustration of the graph decomposition algorithm. The cost of each decomposition is used to select the final solution (Figure taken from Ogras and Marculescu [2005b]).

link insertion to regular architectures [Ogras and Marculescu 2006a]. This approach boosts the network performance while only minimally affecting the network regularity and area.

Finally, using fully customized topologies or imposing long-range links on top of regular topologies may result in links with varying lengths and performance figures. In order to decouple high-level design problems from timing and synchronization problems, latency-insensitive design techniques can be used. The latency-insensitive design [Carloni and Sangiovanni-Vincentelli 2002] maintains the simplicity of synchronous designs while separating communication from computation in a system where several computational processes exchange data by means of communication channels.

3.1.2 *Buffer Sizing.* The input buffers in routers represent a substantial portion of the on-chip overall area. At the same time, buffering space significantly affects network performance. The current practice in NoC design is to allocate the buffering resources uniformly across each router in the design (i.e., all the buffers in the network have equal depth). This makes sense only if all the buffers are equally utilized. However, due to traffic variations, some parts of the network are clearly utilized much more than others. Therefore, the buffering resources should be allocated judiciously so as to optimize the application performance subject to the resource constraints, as illustrated in Figure 13.

Towards this end, an automated buffer space assignment for structured grid-like NoC architectures is discussed in Hu and Marculescu [2004a] and Hu et al. [To appear]. This technique starts with a network configuration where all the buffers have a depth of 1. Then, using sophisticated queueing models for routers, it determines the most likely channel to become congested as a function of the architectural (e.g., network topology, packet service time at routers, etc.) and application-specific (e.g., injection rate at each IP, probability distribution of

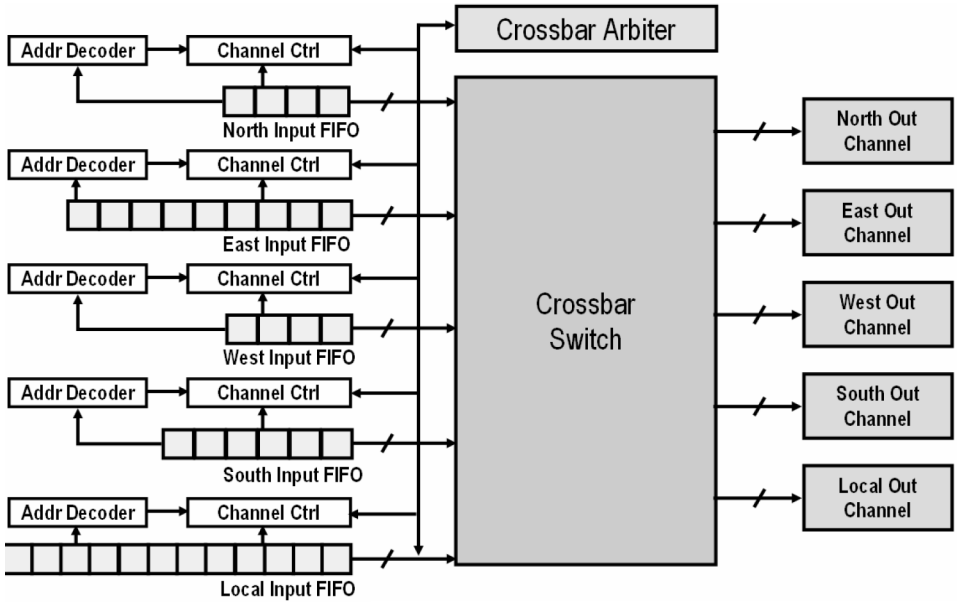


Fig. 13. Illustration of nonuniform buffer space allocation for on-chip routers.

packet destinations, etc.) parameters. After that, the size of the buffer connected to the most congested channel is increased and the procedure repeated until the total buffering space is used up.

Other studies about on-chip buffers include Saastamoinen et al. [2003] and Chandra et al. [2004]. The study in Saastamoinen et al. [2003] analyzes the properties of on-chip buffers and reports gate-level area estimates and buffer utilization across the network. The authors in Chandra et al. [2004] investigate the impact of FIFO sizing on the interconnect throughput for single-source, single-sink interconnect scenarios.

### 3.2 Communication Paradigm

Designing the communication infrastructure involves only the static aspect of network-based communication. The actual paths followed by packets are governed by the routing strategy adopted in the network. The choice of routing decision has a significant impact on network performance and power consumption [Duato et al. 2002; Glass and Ni 1992; Ni and McKinley 1993; Shang et al. 2006] as well as area [Hu and Marculescu 2004]. Since more complicated routing strategies will typically result in larger designs. In addition to this, proper design of a routing algorithm can enable fault-tolerant communication [Dumitras and Marculescu 2003].

A common choice for on-chip routing is *deterministic routing*, which means that the routing decision is fixed, given the source-destination pair. An alternative is *adaptive* routing, where the routers base their routing decisions on the level of congestion in downstream routers as well as on the source-destination pairs. Deterministic routing algorithms are simpler to implement compared

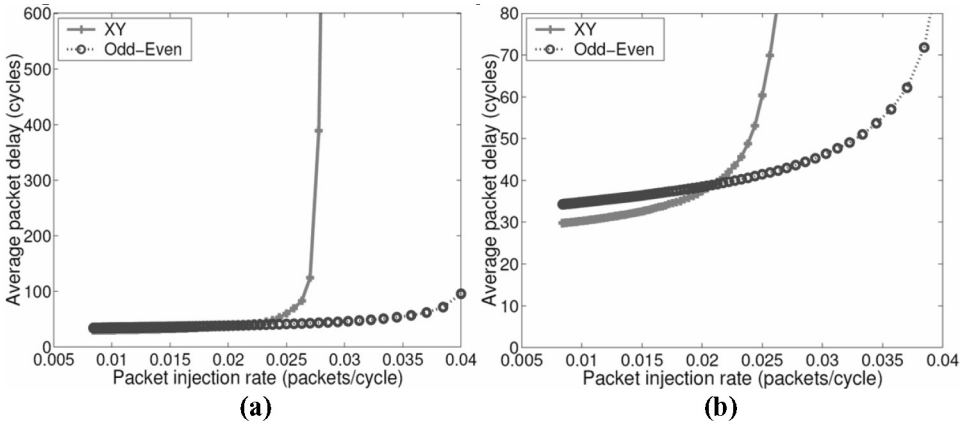


Fig. 14. Performance comparison of deterministic (XY) and adaptive (odd-even) routing for a $6 \times 6$ 2D mesh network under transpose traffic pattern. Both plots show that odd-even routing can sustain higher traffic rates than XY, but plot (b) demonstrates that at low traffic rates, XY routing outperforms odd-even routing (Figure taken from Hu and Marculescu [2004]).

to their adaptive counterparts. Typically, they result in lower area overhead and shorter latency values when the network is not congested. Nevertheless, as the congestion level in the network increases, the performance of deterministic algorithms deteriorates rapidly; this is because the routers cannot adapt to changing traffic conditions. Adaptive algorithms, on the other hand, have more degrees of freedom in making the routing decision since alternative paths can be used in case of severe congestion. This alleviates the accumulation of messages in a certain region of the network and allows for a higher message throughput.

The tradeoff between deterministic and adaptive algorithms in terms of performance is illustrated in Figure 14. Although more difficult to implement, adaptive routing algorithms (e.g. odd-even routing[2] in Chiu [2000]) can achieve larger saturation throughput values compared to deterministic ones (e.g. XY routing in Duato et al. [2002]). This tradeoff suggests that deterministic and adaptive algorithms can be combined to achieve higher performance at all levels of traffic congestion.

The approach proposed in Hu and Marculescu [2004] achieves this goal by dynamically switching between adaptive and deterministic routing modes as a function of the network congestion level. Routers in the network continuously monitor the traffic congestion level in neighboring routers. Unless one of the neighboring routers asserts the congestion flag, the router uses deterministic routing to obtain minimum latency. If the congestion flag is set, it switches to an adaptive mode and makes a smart choice in order to avoid congested links. As a result, routers in the network switch dynamically between deterministic and adaptive modes to obtain the best possible performance. Experimental studies performed using synthetic and multimedia traffic demonstrate that

[2]The Odd-Even routing prohibits East-North and East-South turns at any tile located in an even column, and North-West and South-West turns at any tile in an odd column.

the resulting dynamic routing algorithm performs well at low *and* high traffic levels.

To give some insight, better performance can be achieved via combined deterministic/adaptive routing because the routers in both deterministic and adaptive modes actually coexist in the network. To be more specific, even at low injection rates, some of the routers in the network may experience transient congestion. During these intervals, a few routers can switch to an adaptive mode and prevent messages from blocking the network links and getting stuck in the routers. By the same token, as the network congestion increases, the number of routers in the adaptive mode and the duration they remain in the adaptive mode increases. Nevertheless, some routers in the less congested regions of the network may switch to a deterministic mode and enjoy faster operation. Hence, this heterogeneity results in better performance compared to that using purely adaptive routing algorithms.

### 3.3 Application Mapping to NoC Architectures

When the network architecture is fixed *a priori* or its optimization process is already completed, the job of the designer is to map the target application to the NoC architecture. To solve the mapping problem, we first need a description of the target application. While it may come in different flavors as described in Section 2, a common choice is to describe the target application by a set of communicating processes. When the description is given at task-level, the solution to a task scheduling problem can lead to very interesting algorithms that statically schedule both communication transactions and computation tasks onto heterogeneous NoC architectures under soft or hard real-time constraints [Hu and Marculescu 2005a].

When the task and communication scheduling/binding are fixed using predefined IPs, the application can be modeled as a directed graph where the vertices represent cores and the arcs characterize the communication relationship between them. For instance, the arcs can be annotated by the communication volume and bandwidth requirements for the connection. In this case, the problem is to determine how to topologically map the selected IPs onto the network such that certain metrics of interest are optimized [Hu and Marculescu 2005b; Murali and De Micheli 2004a, 2004b; Ascia et al. 2004].

Network architecture can also be modeled as a graph where the vertices represent network tiles and the arcs characterize the direct links between them. Similar to the application graph, the architecture graph is annotated with information such as the maximum bandwidth of the links, average energy consumption of sending one bit of data, etc. Given the application and platform characterizations, the mapping problem consists of assigning the cores implementing the application to tiles in the architecture such that a certain objective (e.g., performance, energy consumption, fault-tolerance) is achieved subject to different design constraints (e.g., area, bandwidth, latency), as illustrated in Figure 15.

The authors in Hu and Marculescu [2005b] present a mapping technique for regular NoCs aimed at minimizing communication energy consumption while exploiting different routing algorithms. The technique is based on a

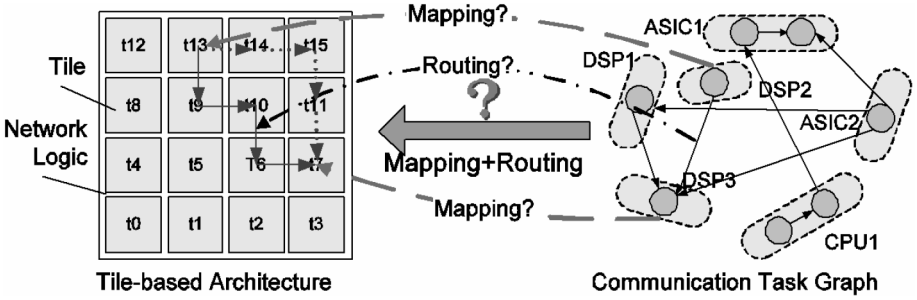


Fig. 15. Target HW platform (a tile-based NoC) and application description are shown. Mapping of IP cores to network tiles and different routing choices are also illustrated (Figure taken from Hu and Marculescu [2005b]).

branch-and-bound algorithm that explores the design space by both topologically mapping IP cores to network tiles and generating a routing path for each communicating pair. Once a feasible mapping is obtained, the communication energy consumption of the resulting system is calculated using the average energy consumed to transport one bit from one node to another [Ye et al. 2002]. Finally, the mapping and corresponding routing path allocation which minimizes communication energy consumption are selected as the best solutions.

Another mapping algorithm that optimizes performance under bandwidth constraints is presented in Murali and De Micheli [2004a]. A multiobjective mapping to mesh-based NoC architectures is discussed in Ascia et al. [2004] and mapping of heterogeneous cores with an embedded floorplanner is presented in Murali and De Micheli [2004b]. Mapping techniques can also be used to reduce the potential hotspots and obtain a thermally-balanced design. For example, a genetic algorithm is proposed to design a thermally-balanced design while minimizing communication cost via placement in Hung et al. [2004].

### 3.4 Stochastic Communication

As we move deeper into the DSM domain, one of the unavoidable consequences is the increasing failure rate in the design. Failures occur mainly due to an increasing number of timing violations and soft-error rates, and the sensitivity of DSM circuits to neutron and alpha radiation [Constantinescu 2002]. Besides this, transient faults and silent data corruption can have a negative effect on circuit behavior [Horst et al. 1993; Constantinescu 2001]. Therefore, it is desirable to inject some architectural and system-level fault-tolerance into the system so as to relax the "100% correctness" requirement for VLSI circuits, and reduce the design and verification costs [Dumitras and Marculescu 2003; Bertozzi et al. 2002; Maly 2001; Semiconductor Association 2003].

A fault-tolerant communication in a network can be achieved through a probabilistic broadcasting scheme called on-chip stochastic communication [Dumitras and Marculescu 2003]. In this approach, the network nodes consist of an IP core for doing computation and a stochastic router, as shown in Figure 16(a). The routers are composed of send/receive buffers, cyclic redundancy check (CRC) hardware, and random number generators.

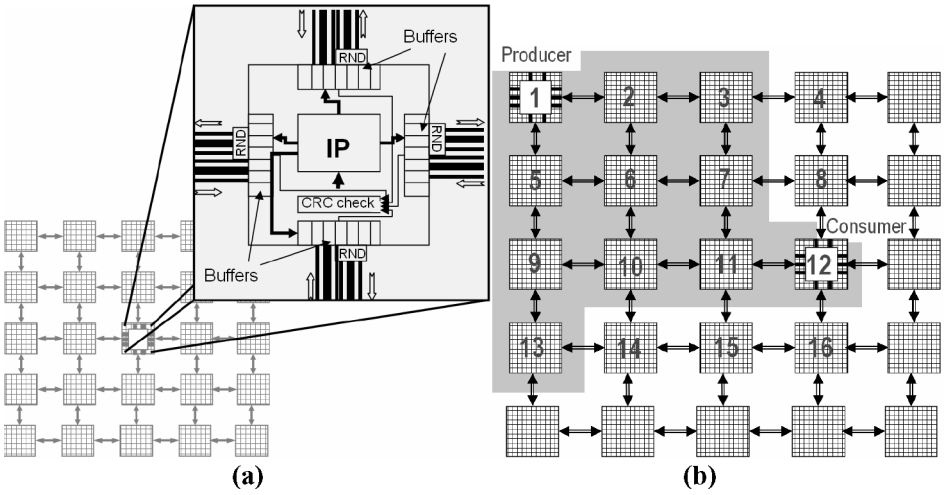


Fig. 16. (a) The on-chip stochastic router design and (b) information dissemination using stochastic communication on a 2D grid.

Suppose we want to exploit router stochastic behavior to transmit messages from the source node 1 (the "Producer" in Figure 16(b)) to the destination node 12 ( "Consumer"). In such a setup, information dissemination takes place across multiple paths, so several nodes within a larger region (Figure 16(b)) will become aware of the newly transmitted message. Upon receiving a new message, the routers first check whether or not the message is corrupted; this check is done using the CRC hardware available at each node in the network. In case of error, the packet is safely discarded because the same message is sent many times to different directions in any case. Otherwise, the message is written to the send/receive buffers. All output ports read a message from this buffer and forward it to their downstream routers with probability $p > 0$. This way, the messages simply diffuse through the network, similar to the proliferation of an epidemic in a large population [Bailey 1975]. When a message reaches its destination, it is forwarded to the local IP, as illustrated in Figure 16(b).

In order to avoid overpopulating the network, a time-to-live (TTL) constant is assigned to every message upon generation. The TTL value is decremented at every hop and when the TTL becomes equal to 0, the message is simply ignored and removed from the network. The forwarding probability $p$ and the TTL are also used to tune the tradeoff between performance and energy consumption. Further details of stochastic communication and experimental evaluations can be found in Dumitras and Marculescu [2003]; alternative approaches are discussed in [Pirretti et al. 2004; Manolache et al. 2005].

To summarize, communication-based design is a fundamental paradigm shift which is becoming increasingly important for both on- and off-chip communication. A holistic approach to the network paradigm involves understanding the theoretical basis (e.g., stochastic modeling and analysis), essential properties (e.g., structure, dynamics), and metrics (e.g., energy, fault-tolerance) which are relevant to designing different networks. At the same time, a deeper

integration of physical-level and system-level issues is needed in order to better understand the complex behavior of future SoCs.

## 4. SIMULATION AND PROTOTYPING FOR PERFORMANCE EVALUATION

Referring to the design methodology in Figure 1, loop *A* is most likely iterated a large number of times in order to solve a constrained optimization problem such as minimizing energy consumption with performance constraints. Therefore, the speed of the analysis techniques and the type of metrics they can provide are more important than absolute accuracy. On the other hand, when the communication refinement stage is completed, the system has already evolved towards a more stable state. At this point, more accurate, yet time-consuming, performance evaluation techniques are used. Longer execution times can be tolerated at this stage since only a small number of iterations are typically expected for the *B* loop in Figure 1. Absolute accuracy, on the other hand, becomes crucial in this case because any iteration during later stages of the design results in major penalties in both time and cost. For this reason, systematic approaches for accurate system evaluation are needed. Simulation and prototyping are two possible means towards this end.

*Simulation* models the system under study at a level of detail which is intractable for analytical approaches. In turn, simulation provides: (*i*) more accurate estimations; and (*ii*) a flexible framework to obtain metrics beyond the reach of analysis. For instance, queueing network analysis becomes easily intractable when considering nonMarkovian traffic and finite buffers in the system. On the other hand, simulations for various parameters and network configurations can easily capture such effects. As examples of simulation-based approaches, the Metropolis [Balarin et al. 2003] and SPADE [Lieverse et al. 2001] frameworks provide capabilities for design exploration of heterogeneous systems by following the Y-Chart paradigm discussed in Section 1. Metropolis provides a simulation environment based on a metamodel with precise semantics. This metamodel is general enough not only to support existing models of computation, but also to accommodate new ones. The metamodel supports functionality and architecture specification and simulation and formal verification, as well as the mapping of functionality to architectural resources. In SPADE the application is modeled as a network of KPNs and since this choice provides an executable model, it can be directly simulated to obtain output traces. The traces capture both the computation and communication workloads imposed on system resources. SPADE provides abstract architecture models developed using a library of generic and extendable building blocks. Finally, the application is mapped to the architecture and trace-based simulations are performed for performance evaluation.

*Prototyping* provides even more accurate results compared to simulation. However, it requires a significant effort and lacks flexibility, since the prototype is tailored towards a specific system [Ogras et al. 2006; Lee et al. 2006; Adriahantenaina and Greiner 2003; Bartic et al. 2003]. Having a prototype available enables the designer to make evaluations beyond the capabilities of simulation. For instance, simulation can be used for power consumption evaluation only

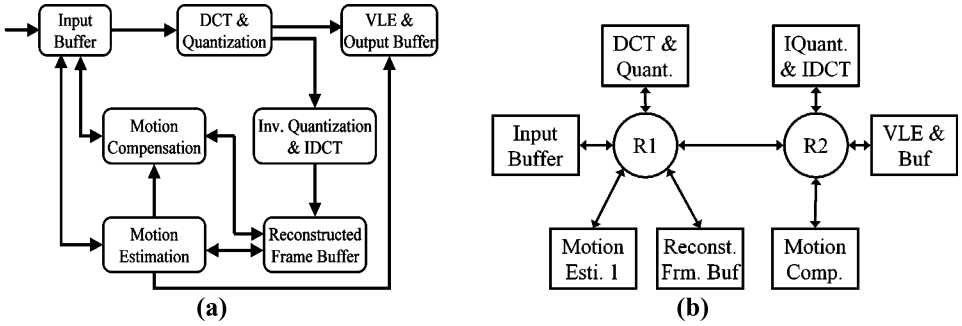


Fig. 17. MPEG-2 encoder implementation using (a) P2P- and(b) NoC-based architectures.

after developing power models for all components in the system and estimating the workload. All assumptions and approximations made during this process deteriorate the accuracy of the results. In addition, simulations cannot be used for area estimation. On the other hand, prototypes can successfully address all of these problems.

An FPGA-based NoC prototype is introduced in Lee et al. [2006]. The authors present complete P2P- and NoC-based implementations of the MPEG-2 encoder in Figure 17. The NoC-based implementation of the encoder achieves a 47 frames/sec encoding rate for a CIF frame of size $352 \times 288$, while the P2P architecture provides a throughput of 48 frames/sec. By performing direct measurements on the FPGA prototype and using real video clips, the authors show the real benefits of using the NoC approach in terms of power consumption and area, particularly as the system size becomes larger. Other examples of prototyping include Adriahantenaina and Greiner [2003] and Bartic et al. [2003]. In the former, the authors present the SPIN interconnect architecture and implement a 32-port network architecture supporting best-effort service. In the latter, a flexible FPGA-based NoC design that consists of instruction set processors and reconfigurable components is presented.

To summarize, simulation and prototyping are very important tools for systems design. They complement formal approaches for analysis and optimization purposes. Together, they offer the accuracy and flexibility the designer needs in order to find the best design tradeoffs.

## 5. CONCLUSION

In this article, we have discussed some fundamental research issues concerning both computation and communication aspects in system-level design. We have shown that as applications become more complex, concurrency and communication play increasingly important roles in the design process. In particular, highly concurrent communicating models demand truly scalable communication architectures. As shown, refining the communication medium from a simple bus to a multihop network requires a completely new design methodology. Towards this end, we presented a communication-centric approach where abstract application and platform models are used to optimize the system via performance analysis. Both computation and communication refinement steps have been

addressed and state-of-the-art solutions were discussed. Finally, simulation and prototyping were considered as possible means for pushing the limits of analytical approaches for performance analysis.

ACKNOWLEDGMENTS

The authors would like to thank all current and previous members of System Level Design Group (SLD) at Carnegie Mellon University. Their contribution to bringing many of these ideas to life was essential. Discussions with Professor Alberto Sangiovanni Vincentelli of UC Berkeley and the members in the component/communication-based design and heterogeneous systems teams in GSRC are also greatly acknowledged.

REFERENCES

ADRIAHANTENAINA, A. AND GREINER, A. 2003. Micro-network for SoC: Implementation of a 32-Port SPIN network. In *Proceedings of the Design Automation and Test in Europe Conference*.

ASCIA, G., CATANIA, V., AND PALESI, M. 2004. Multi-objective mapping for mesh-based NoC architectures. In *Proceedings of the International Conference on Hardware/Software Codesign and System Synthesis*.

BAILEY, N. 1975. *The Mathematical Theory of Infectious Diseases*, 2nd ed. Charles Griffin, London.

BALARIN, F., WATANABE, Y., HSIEH, H., LAVAGNO, L., PASSERONE, C., AND SANGIOVANNI-VINCENTELLI, A. 2003. Metropolis: An integrated electronic system design environment. *IEEE Computer 36*, 4, 45–52.

BARTIC, T. A., MIGNOLET, J.-Y., NOLLET, V., MARESCAUX, T., VERKEST, D., VERNALDE, S., AND LAUWEREINS, R. 2003. Highly scalable network on chip for reconfigurable systems. In *Proceedings of the International Symposium on System-on-Chip*.

BENINI, L. AND DE MICHELI, G. 2002. Networks on chips: A new SoC paradigm. *IEEE Comput. 35*, 1.

BERAN, J. 1994. *Statistics for Long-Memory Processes*. Chapman & Hall, London.

BERGSTRA, J. A. AND KLOP, J. W. 1985. Algebra of communicating processes with abstraction. *Theoretical Comput. Sci. 37*, 1.

BERTOZZI, D., BENINI, L., AND DE MICHELI, G. 2002. Low power error resilient encoding for on-chip data buses. In *Proceedings of the Design Automation and Test in Europe Conference*.

BERTSEKAS, D. AND GALLAGER, R. 1987. *Data Networks*. Prentice-Hall, Upper Saddle River, N.J.

CARLONI, L. P. AND SANGIOVANNI-VINCENTELLI, A. L. 2002. Coping with latency in SoC design. *IEEE Micro (Special Issue on System on Chip) 22*, 5.

CHANDRA, V., XU, A., SCHMIT, H., AND PILEGGI, L. 2004. An interconnect channel design for high performance integrated circuits. In *Proceedings of the Design Automation and Test in Europe Conference*.

CHIU, G.-M. 2000. The odd-even turn model for adaptive routing. *IEEE Trans. Parallel Distributed Syst. 11*, 7, 729–738.

CONSTANTINESCU, C. 2001. Dependability analysis of a fault-tolerant processor. In *Proceedings of the Pacific Rim International Symposium on Dependable Computing*.

CONSTANTINESCU, C. 2002. Impact of deep submicron technology on dependability of VLSI circuits. In *Proceedings of the International Conference on Dependable Systems and Networks*.

DALLY, W. AND TOWLES, B. 2001. Route packets, not wires: On-Chip interconnection networks. In *Proceedings of the 38th ACM IEEE Design Automation Canference*.

DUATO, J., YALAMANCHILI, S., AND NI, L. M. 2002. *Interconnection Networks: An Engineering Approach*. Morgan Kaufmann, San Fransisco, Calif.

DUMITRAS, T. AND MARCULESCU, R. 2003. On-Chip stochastic communication. In *Proceedings of the Design Automation and Test in Europe Conference*.

FERRARI, A. AND SANGIOVANNI-VINCENTELLI, A. 1999. System design: Traditional concepts and new paradigms. In *Proceedings of the International Conference on Computer Design*.

FLYNN, D. 1997. AMBA: Enabling reusable on-chip designs. *IEEE Micro 17*, 4, 20–27.

GEILEN, M. C. W., BASTEN, T., AND STUIJK, S. 2005. Minimising buffer requirements of synchronous dataflow graphs with model checking. In *Proceedings of the ACM IEEE Design Automation Conference*.

GLASS, C. J. AND NI, L. M. 1992. The turn model for adaptive routing. *J. ACM 41*, 5, 874–902.

GOTZ, N., HERZOG, U., AND RETTELBACH, M. 1993. Multiprocessor and distributed system design: The integration of functional specification and performance analysis using stochastic process algebras. In *Tutorial Proceedings of the 16th International Symposium on Computer Performance Modelling, Measurement and Evaluation*, vol. 729. Lecture Notes in Computer Science, Springer Verlag, New York.

HEDETNIEMI, S. M., HEDETNIEMI, S. T., AND LIESTMAN, A. L. 1988. A survey of gossiping and broadcasting in communication networks. *Networks 18*, 4, 319–359.

HEMANI, A., JANTSCH, A., KUMAR, S., POSTULA, A., OBERG, J., MILLBERG, M., AND LINDQVIST, D. 2000. Network on a chip: An architecture for billion transistor era. In *Proceedings of the IEEE NorChip Conference*.

HILLSTONE, J. 1996. *A Compositional Approach to Performance Modelling*. Cambridge University Press, Cambridge Mass.

HORST, R., JEWETT, D., AND LENOWSKI, D. 1993. The risk of data corruption in microprocessor based systems. In *Proceedings of the 23rd International Symposium on Fault-Tolerant Computing*.

HU, J. AND MARCULESCU, R. 2004. DyAD-Smart routing for networks-on-chip. In *Proceedings of the Design Automation Conference*.

HU, J. AND MARCULESCU, R. 2004a. Application-specific buffer space allocation for Networks-on-Chip router design. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*. San Jose, CA.

HU, J. AND MARCULESCU, R. 2005a. Communication and task scheduling of application-specific Networks-on-chip. *IEE Proceedings Comput. Digital Techniques. 152*, 5, 643–651.

HU, J. AND MARCULESCU, R. 2005b. Energy- and performance-aware mapping for regular NoC architectures. *IEEE Trans. Comput. Aided Des. Integrated Circuits Syst. 24*, 4.

HU, J., OGRAS, U. Y., AND MARCULESCU, R. System-level buffer allocation for application-specific Networks-on-Chip router design. *IEEE Trans Comput. Aided Des. Integrated Circuits Syst*. To appear.

HUNG, W., ADDO-QUAYE, C., THEOCHARIDES, T., XIE, Y., VIJAKRISHNAN, N., AND IRWIN, M. J. 2004. Thermal-aware IP virtualization and placement for networks-on-chip architecture. In *Proceedings of the IEEE Conference on Computer Design*.

IBM CORECONNECT. 2006. http://www.chips.ibm.com/products/powerpc/cores.

JALABERT, A., MURALI, S., BENINI, L., AND DE MICHELI, G. 2004. $\times$ pipesCompiler: A tool for instantiating application specific networks on chip. In *Proceedings of the Design Automation and Test in Europe Conference*.

JANTSCH, A. AND TENHUNEN, H. (EDS.). 2003. *Networks-on-Chip*. Kluwer, Hingham, Mass.

KAHN, G. 1974. The semantics of a simple language for parallel programming. In *Information Processing*, J. L. Rosenfeld, ed. Stockholm, Sweden.

KIM, J. AND SHIN, K. G. 1996. Execution time analysis of communicating tasks in distributed systems. *IEEE Trans. Comput. 45*, 5, 572–579.

LEE, E. AND MESSERSCHMITT, D. 1987. Synchronous dataflow. In *Proc. IEEE 75*, 9, 1235–1245.

LEE, E. AND PARKS, T. M. 1995. Dataflow process networks. *Proc. IEEE 83*, 5.

LEE, H. G., OGRAS, U. Y., MARCULESCU, R., AND CHANG, N. 2006. Design space exploration and prototyping for on-chip multimedia applications. In *Proceedings of the Design Automation Conference*.

LIEVERSE, P., WOLF, P., VISSERS, K., AND DEPRETTERE, E. 2001. A methodology for architecture exploration of heterogeneous signal processing systems. *J. VLSI Signal Processing Syst. Signal, Image Video Technol. 29*, 3.

LIN, T. AND PILEGGI, L. T. 2002. Throughput-Driven IC communication fabric synthesis. In *Proceedings of the International Conference on Computer-Aided Design*.

MALY, W. 2001. IC design in high-cost nanometer technologies era. In *Proceedings of the 38th Conference on Design Automation*.

MANOLACHE, S., ELES, P., AND PENG, Z. 2005. Fault and Energy aware communication mapping with quaranteed latency for applications implemented on NoC. In *Proceedings of the Design Automation Conference*.

MARCULESCU, R., NANDI, A., LAVAGNO, L., AND SANGIOVANNI-VINCENTELLI, A. 2001. System-Level power/performance analysis of portable multimedia systems communicating over wireless channels. In *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*.

MAXIAGUINE, A., KUENZLI, S., CHAKRABORTY S., AND THIELE, L. 2004. Rate analysis for streaming applications with on-chip buffer constraints. In *Proceedings of the Asia and South Pacific Design Automation Conference*.

MILNER, R. 1989. *Communication and Concurrency*. Prentice-Hall, Englewood Cliffs, N.J.

MISHRA, P. SHRIVASTAVA, A., AND DUTT, N. Architecture description language (ADL-) Driven software toolkit generation for architectural exploration of programmable SOCs. *ACM Trans. Design Autom. Elect. Syst. 11*, 3, 626–658.

MORGENSTEIN, A. 2004. Comparative analysis of serial vs. parallel links in networks on chip. In *Proceedings of the International Symposium on Systems on Chip.*

MURALI, S. AND DE MICHELI, G. 2004a. Bandwidth-Constrained mapping of cores onto NoC architectures. In *Proceedings of the Design Automation and Test in Europe Conference*.

MURALI, S. AND DE MICHELI, G. 2004b. SUNMAP: A tool for automatic topology selection and generation for NoCs. In *Proceedings of the Design Automation Conference*.

NANDI, A. AND MARCULESCU, R. 2001. System-Level power/performance analysis for embedded systems design. In *Proceedings of the 38th ACM/IEEE Design Automation Conference*.

NI, L. M. AND MCKINLEY, P. K. 1993. A survey of wormhole routing techniques in direct networks. *IEEE Trans. Comput. 26*.

NORROS, I. 1994. A storage model with self-similar input. *Queueing Syst. 16*, 3–4, 387–396.

OGRAS, U. Y., HU, J., AND MARCULESCU, R. 2005. Key research problems in NoC design: A holistic perspective. In *Proceedings of the 3rd IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthses*.

OGRAS, U. Y. AND MARCULESCU, R. 2005a. Application-Specific network-on-chip architecture customization via long-range link insertion. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*.

OGRAS, U. Y. AND MARCULESCU, R. 2005b. Energy- and performance-driven NoC communication architecture synthesis using a decomposition approach. In *Proceedings of the Design Automation and Test in Europe Conference*.

OGRAS, U. Y., MARCULESCU, R., LEE, H. G., AND CHANG, N. 2006. Communication architecture optimization: Making the shortest path shorter in regular networks-on-chip. In *Proceedings of the Design Automation and Test in Europe Conference*.

OGRAS, U. Y. AND MARCULESCU, R. 2006a. It's a small world after all; NoC performance optimization via long link insertion. *IEEE Trans. Very Large Scale Integrat. Syst.* (Special Section Hardware/Software Codesign and System Synthesis.) *14* (July).

OPEN CORE PROTOCOL INTERNATIONAL PARTNERSHIP (OCP-IP). 2006. OCP datasheet. http://www.ocpip.org.

PETERSON, J. L. 1981. *Petri Net Theory and the Modeling of Systems*. Prentice-Hall, Englewood Cliffs, N.J.

PINTO, A. AND SANGIOVANNI-VINCENTELLI, A. 2003. Efficient synthesis of networks on chip. In *Proceedings of the 21st International Conference on Computer Designe*.

PINTO, A., BONIVENTO, A., SANGIOVANNI-VINCENTELLI, A. L., PASSERONE, R., AND SGROI, M. 2006. System level design paradigms: Platform-based design and communication synthesis. *ACM Trans. Design Autom. Elect. Syst., 11*, 3, 537–563.

PIRRETTI, M. 2004. Fault tolerant algorithms for network-on-chip interconnect. In *Proceedings of the IEEE Computer Society Annual Symposium on VLSI*.

PLATEAU, B. AND ATIF, K. 1991. Stochastic automata network for modeling parallel systems. *IEEE Trans. Softw. Eng. 17*, (Oct.).

PLATEAU, B. AND FOURNEAU, J. M. 1991. A methodology for solving Markov models of parallel systems. *J. Parallel Distrib. Comput. 12*, 4, 370–387.

Pop, P. and ELES, P. 2006. Analysis and optimization of communication-dominated real-time embedded systems. *ACM Trans. Design Autom. Elect. Syst. 11*, 3, 593–625.

SAASTAMOINEN, I., ALHO, M., AND NURMI, J. 2003. Buffer implementation for proteo network-on-chip. In *Proceedings of the International Symposium on Circuits and Systems*.

SEMICONDUCTOR ASSOCIATION. 2003. The International Technology Roadmap for Semiconductors (ITRS).

SHANG, L., PEH, L. S., AND JHA, N. K. 2006. POWERHERD: A distributed scheme for dynamic satisfying peak power constraints in interconnection networks. *IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst. 25*, 1, 92–110.

SRINIVASAN, K., CHATHA, K. S., AND KONJEVOD, G. 2004. Linear programming based techniques for synthesis of network-on-chip architectures. In *Proceedings of the IEEE International Conference on Computer Design*.

SRIRAM, S. AND BHATTACHARYYA, S. S. 2001. *Embedded Multiprocessors Scheduling and Synchronization*. Marcel Dekker, New York.

STEWART, W. 1994. *An Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, N.J.

STEWART, W., ATIF, K., AND PLATEAU, B. 1995. The numerical solution of stochastic automata networks. In *European J. Operational Research 86*, 503–525.

TRIVEDI, K. S. 1982. *Probability and Statistics with Reliability, Queueing, and Computer Science Applications*. Prentice-Hall., Englewood Cliffs, N.J.

VARATKAR, G. AND MARCULESCU, R. 2004. On-Chip traffic modeling and synthesis for MPEG-2 video applications. *IEEE Trans. VLSI 12*, 1, 108–119.

YANG, G., SANGIOVANNI-VINCENTELLI, A., WATANABE, Y., AND BALARIN, F. 2004. Separation of concerns: Overhead in modeling and efficient simulation techniques. In *Proceedings of the ACM 4th International Conference on Embedded Software*.

YE, T. T., BENINI, L., AND DE MICHELI, G. 2002. Analysis of power consumption on switch fabrics in network routers. In *Proceedings of the Design Automation Conference*.

YE, T. T. AND DE MICHELI, G. 2003. Physical planning for multiprocessor networks and switch fabrics. In *Proceedings of the IEEE International Conference on Application-Specific Systems, Architectures, and Processors*.

Received February 2006; accepted May 2006