On-Chip Communication Architecture Exploration: A Quantitative Evaluation of Point-to-Point, Bus, and Network-on-Chip Approaches

HYUNG GYU LEE and NAEHYUCK CHANG
Seoul National University
and
UMIT Y. OGRAS and RADU MARCULESCU
Carnegie Mellon University

Traditionally, design-space exploration for systems-on-chip (SoCs) has focused on the computational aspects of the problem at hand. However, as the number of components on a single chip and their performance continue to increase, a shift from computation-based to communication-based design becomes mandatory. As a result, the communication architecture plays a major role in the area, performance, and energy consumption of the overall system. This article presents a comprehensive evaluation of three on-chip communication architectures targeting multimedia applications. Specifically, we compare and contrast the network-on-chip (NoC) with point-to-point (P2P) and bus-based communication architectures in terms of area, performance, and energy consumption. As the main contribution, we present complete P2P, bus-, and NoC-based implementations of a real multimedia application (i. e. the MPEG-2 encoder), and provide direct measurements using an FPGA prototype and actual video clips, rather than simulation and synthetic workloads. We also support the experimental findings through a theoretical analysis. Both experimental and analysis results show that the NoC architecture scales very well in terms of area, performance, energy, and design effort, while the P2P and bus-based architectures scale poorly on all accounts except for performance and area, respectively.

Categories and Subject Descriptors: B.4.3 [**Input/Output and Data Communications**]: Interconnections (Subsystems)—*Topology (e.g., bus, point-to-point)*

The authors acknowledge the support of the Gigascale Systems Research Center, one of five research centers funded under the Focus Center Research Program, a Semiconductor Research Corporation Program. This work was also supported in part by the International Research Internship Program of the Korea Research Foundation (KRF).

Authors' addresses: H. G. Lee and N. Chang, School of Computer Science and Engineering, Seoul National University, 599 Kwanak Rd., Seoul, Korea; email: {hglee,naehyuck}@cselab.snu.ac.kr; U. Y. Ogras and R. Marculescu, Department of Electrical and Computer Engineering, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213; email: {radum,uogras}@ece.cmu.edu.

General Terms: Design, Measurement, Performance

Additional Key Words and Phrases: Networks-on-chip, point-to-point, system-on-chip, MPEG-2 encoder, FPGA prototype

ACM Reference Format:

Lee, H. G., Chang, N., Ogras, U. Y., and Marculescu, R. 2007. On-Chip communication architecture exploration: A quantitative evaluation of point-to-point, bus, and network-on-chip approaches. ACM Trans. Des. Autom. Electron. Syst. 12, 3, Article 23 (August 2007), 20 pages. DOI = 10.1145/1255456.1255460 http://doi.acm.org/10.1145/1255456.1255460

1. INTRODUCTION

The increasing number of IP cores that can be integrated on a single chip enables implementation of complex applications using the system-on-chip (SoC) approach. The huge communication demands of these applications and the abundant computation power available on chip put tremendous pressure on the communication architecture. Consequently, scalable communication architectures are needed for efficient implementation of future systems.

Traditionally, two types of on-chip communication schemes have been considered, namely, point-to-point (P2P) and bus-based communication architectures. P2P communication architectures can provide the utmost in communication performance at the expense of dedicated channels among all the communicating IP pairs. However, these architectures suffer from lack of scalability in terms of high complexity, cost, and design effort. On the other hand, bus-based architectures can connect a few tens of IP cores in a cost-efficient manner by reducing the design complexity and eliminating the dedicated wires required by P2P communication architectures. However, bus-based architectures still fail to satisfy the requirements of future applications mainly due to lack of scalability, both in terms of energy and performance. Indeed, from an implementation standpoint, a bus-based design would clearly provide lower performance figures due to its limited bandwidth capabilities. Moreover, the large capacitive load of the bus drivers results in large delays and energy consumption in the interconnected wires [Wolkotte et al. 2005]; this makes the bus-based solution inappropriate for implementing a complex design such as an MPEG-2 encoder.

In contrast to these methods, the network-on-chip (NoC) approach emerged as a promising solution to on-chip communication problems [Benini and De Micheli 2002; Dally and Towles 2001; Jantsch and Tenhunen 2003; Hemani et al. 2000]. NoC communication architectures connect the processing and storage resources via a network. Therefore, communication among various cores is realized by generating and forwarding packets through the network infrastructure. By eliminating the (ad hoc) global wires, the NoC approach provides the following advantages:

—*Scalability.* Since communication among different nodes is achieved by routing packets, a large number of cores can be connected without using (long) global wires. Moreover, the network bandwidth scales nicely with the number of cores in the design. Hence, the NoC paradigm provides a highly scalable communication architecture.

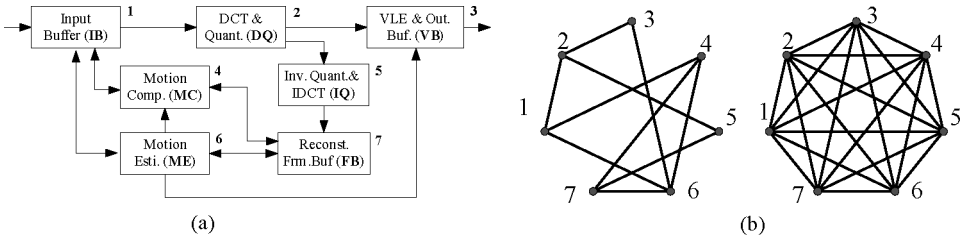

Fig. 1. (a) MPEG-2 encoder implementation and its; (b) graph representation using a P2P communication architecture and a complete graph of the same size.

—*Design reuse.* The modularity of the NoC approach offers a great potential for reusing the network routers and other IP cores. The routers, interconnect, and lower-level communication protocols can be designed, optimized, and verified only once and reused subsequently in a large number of products. Likewise, the IP cores complying with the network interface can be reused across many different applications.

—*Predictability.* The structured nature of global wires facilitates well-controlled and optimized electrical parameters. In turn, the controlled parameters allow for aggressive signaling circuits which can reduce the power dissipation and propagation delay significantly.

While NoCs recently gained a significant momentum, there are no complete NoC implementations of *real applications* reported to-date. Therefore, there is a need for both experimental and theoretical evidence to show that NoC architectures can indeed outperform their bus and P2P counterparts. To remedy this situation, the goal of this article is to demonstrate the viability of the NoC approach using a concrete hardware implementation running a real multimedia application. Towards this end, this work presents a complete MPEG-2 encoder design using the NoC approach and compares it with P2P and nonsegmented bus-based designs running the same application. The MPEG-2 encoder has been selected as the driver application, since it covers a rich *class* of multimedia applications where similar considerations apply from an implementation standpoint. For instance, basic JPEG, motion-JPEG, and MPEG-1 encoders can all be implemented using a similar architecture and set of IP cores. We also note that, due to the small number of point-to-point connections in its architecture, the MPEG-2 encoder lends itself to a P2P implementation, as shown in Figure 1(a). Indeed, the link-to-node ratio in the MPEG-2 implementation shown in Figure l(b) is only 1.5, while the same ratio is found to be 3.0 for a complete graph of the same size, even when all the links are unidirectional. It should be noted that for many other applications where a subset of cores communicate with all remaining nodes, the overhead incurred by dedicated channels of the P2P architecture is significant. Similarly, the performance of bus architectures drops quickly due to the large number of communicating cores. As a result, the conclusions derived herein with respect to the benefits of the NoC architecture compared to the P2P and bus architectures are rather conservative, so they shed light on a wide range of practical scenarios.

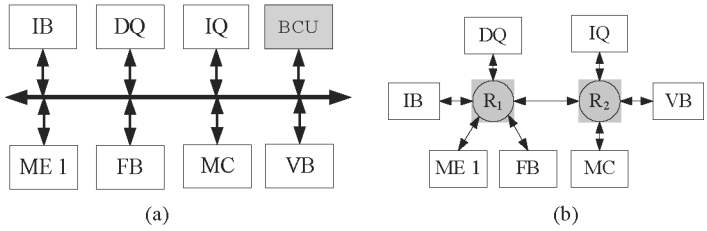

Fig. 2. (a) Bus and; (b) NoC-based implementations of the MPEG-2 encoder with one ME module.

1.1 Overall Approach and Article Contribution

We first design the computational resources that are needed for the implementation of the MPEG-2 encoder (e.g., discrete cosine transformation, motion estimation, and variable-length encoding modules) using Verilog HDL. Next, the cores are connected to each other using P2P links, a bus, and an NoC architecture, as shown in Figure l(a) and Figure 2. For the bus and NoC implementations, we also design a bus control unit (ECU) and an on-chip router, respectively. Once the designs are complete, they are evaluated in terms of area, performance, energy, and power consumption. In addition to these standard metrics, we are also interested in analyzing the *scalability* of these communication architectures. For this reason, we increase the parallelism of the encoder by duplicating the motion estimation (ME) module, which is the true performance bottleneck for this multimedia application. This way, we increase the number of cores in the design and perform evaluations with 1, 2, 4, and 8 motion estimation modules;[1] we refer to the design with one ME as the *baseline implementation.*

The P2P, bus, and NoC architectures are first compared analytically using the area, performance, and power consumption values obtained for each module individually. Besides providing insight about the performance of the communication architectures, the analytical approach enables us to project results for larger designs. Finally, the analytical results are verified against real measurements on the FPGA prototype based on a Xilinx XC2V3000 platform.

According to our measurements, the NoC-based MPEG-2 implementation with one *ME* module achieves a 45.6 *frames*/*sec* encoding rate for a common intermediate format (CIF) frame of size 352 $\times$ 288, while P2P and bus architectures achieve 46.5 *frames*/*sec* and 36.2 *frames*/*sec,* respectively. At the same time, the area of the NoC design is about 3.5% smaller than the P2P counterpart, and 4.2% larger than the bus architecture. These values show that even for the baseline implementation, the NoC architecture performs as well as the P2P architecture, while having a smaller area overhead. At the same time, the real benefits of using the NoC approach are observed when we analyze the scalability of these designs as a function of the number of cores. More specifically, when the motion estimation module, which performs the most computationally expensive task, is replicated, the area occupied by the P2P implementation grows

[1]This corresponds to having MPEG-2 encoder implementations with a total of 7, 8, 10, and 14 cores, respectively.

abruptly. The bus architecture, on the other hand, scales well in terms of area, but it suffers from an energy/performance standpoint. Unlike both of these approaches, the NoC implementation incurs only a modest area overhead while keeping up with the performance increase achieved by the P2P implementation. Finally, the NoC-based implementation also has better scalability in terms of energy consumption compared to P2P and bus-based implementations.

1.2 Article Organization

The remainder of this work is organized as follows: Section 2 reviews related work. Section 3 presents the details of the P2P-, bus-, and NoC-based implementations of the MPEG-2 encoder. Detailed area, performance, and power consumption comparisons are provided in Sections 4, 5, and 6, respectively. Finally, our conclusions appear in Section 7.

2. RELATED WORK

The networks-on-chip communication paradigm is introduced and motivated in Benini and De Micheli [2002], Dally and Towles [2001], Jantsch and Tenhunen [2003], and Hemani et al. [2000]; several key research issues in NoC design are discussed in Ogras et al. [2005]. While it is intuitively accepted that NoCs can provide scalable communication with small area overhead, this fact has not been justified to-date by concrete NoC implementations of real applications. In addition, most theoretical studies rely mainly on simulation to justify such findings, so the network traffic used in such studies is either synthetic or approximating a real application via synthetic traffic generators. For example, several studies in [Hu and Marculescu 2005; Murali et al. 2006; Ogras and Marculescu 2005; Srinivasan et al. 2004] present design methodologies for application mapping and NoC topology synthesis and demonstrate the effectiveness of these methodologies via simulation. The authors of Kim et al. [2005] present an MPEG-4 performance evaluation for a CDMA-based implementation of a NoC, but their design mimics the MPEG-4 traffic using a random traffic generator; this is clearly problematic, given the complex nature of multimedia traffic [Varatkar and Marculescu 2004]. Finally, the work presented in Bolotin et al. [2004] compares the P2P, bus, and NoC communication architectures assuming a uniform traffic distribution.

On the other hand, several studies consider implementation issues in NoCs [Adriahantenaina and Greiner 2003; Lee et al. 2004; Liang et al. 2004]. However, these studies do not consider the target application, but assume synthetic traffic patterns instead. For instance, the authors present in Adriahantenaina and Greiner [2003] the SPIN interconnect architecture and implement a 32-port SPIN architecture supporting best-effort traffic using a 0.13μm process. Similarly, a low-power on-chip network implementation with a hierarchical star topology is presented in Lee et al. [2004]. Finally, the architectural support for compile-time scheduling for on-chip communication is presented in Liang et al. [2004]. This approach optimizes data transfer that can be determined at compile-time using scheduling, and provides software-based dynamic routing. However, the implementation was not tested using a real driver application.

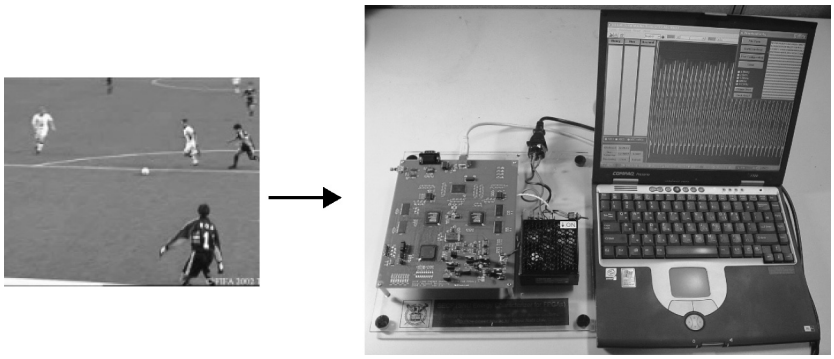

Fig. 3. The FPGA prototype running the MPEG-2 encoder as demonstrated at the *UBooth, Design Automation Conference* in July 2006.

Energy and power models for NoCs are discussed in several studies. In Wang et al. [2002], the authors develop a power model for routers and introduce a power-performance simulator for interconnect networks. Similarly, the authors in Ye et al. [2002] analyze the energy per bit consumed from source to destination and propose a system-level model for NoCs. However, none of these studies considers the power consumption of a complete system implemented using the NoC approach. In [Xu et al. 2004], the authors compare different NoC architectures and a bus architecture using a smart camera SoC as a driver application. The authors adopt a telecommunication network simulator and utilize real traffic traces to evaluate the designs under study. Similarly, the researchers in Wolkotte et al. [2005] present energy models and compare NoC and bus communication architectures. However, the results are obtained using only synthetic workloads.

In contrast to previous work, we present a *complete* NoC implementation of an MPEG-2 encoder as the driver application. The processing and storage elements, as well as the on-chip routers, are all implemented using Verilog HDL. Then, the functionality of the design is validated using an FPGA-based prototype and real video clips which cover a wide range of static and dynamic (e.g., sports) images, as shown in Figure 3. Furthermore, the same target application is also implemented using bus and P2P architectures prototyped on an FPGA board. Finally, accurate area, performance, and energy consumption measurements are obtained directly from the prototype for each implementation.

3. MPEG-2 ENCODER IMPLEMENTATION

The basic MPEG-2 encoder implementation using the P2P approach is depicted in Figure 1. It consists of 7 modules: (1) input buffer (IB); (2) DCT and quantization (DQ); (3) variable length encoder and output buffer (VE); (4) motion compensation (MC); (5) inverse quantization and inverse DCT (IQ); (6) motion estimation (ME); and (7) reconstructed frame buffer (FB). In our implementation, each intra frame I is followed by 3 predicted (P) frames.

The P2P architecture obviously enables the fastest possible communication due to the existence of dedicated channels between all the communicating

Table I. Area Comparison for Individual Cores in MPEG-2

	w/o Wrapper		in P2P Arch.			in Bus Arch.			in NoC Arch.		
Core	*Slices*	*BRAM*	*# of NIs*	*Slices*	*BRAM*	*# of NIs*	*Slices*	*BRAM*	*# of NIs*	*Slices*	*BRAM*
Input Buffer (IB)	103	1	3	37	1	1	290	1	1	266	1
DCT and Quantization (DQ)	2,449	1	3	2,682	1	1	2,633	1	1	2,615	1
Inverse Quantization and Inverse DCT (IQ)	3,875	2	2	4,018	2	1	3,971	2	1	3,950	2
Reconstructed Frame Buffer (FB)	770	76	3	1,066	76	1	958	76	1	933	76
Motion Estimation (ME)	473	8	4	897	8	1	662	8	1	634	8
Motion Compensation (MC)	430	2	2	733	2	1	620	2	1	598	2
Variable-Length Encoder and Output Buffer (VB)	765	0	3	1,094	0	1	946	0	1	934	0

The number of slices and BRAMs are in a Xilinx Virtex2 3000 FPGA. The columns labeled "w/o Wrapper" give the core area without any wrapper, while the remaining columns give the area of the cores implemented using P2P, bus, and NoC architectures, respectively.

modules. On the other hand, the utilization of dedicated channels is low, since most of the time, the links among various modules are idle. For instance, we measured an average utilization of the P2P links of only 4% using our MPEG-2 encoder prototype.

As opposed to this, bus and NoC communication architectures enable link sharing among the communicating cores. For instance, the bus and NoC implementations for this encoder need 1 and 8 links, respectively, in the network, as opposed to 10 links needed in the P2P version (see Figure 2). Each link can be either bidirectional or unidirectional, depending on the functionality of the connected modules. Obviously, sharing links may cause extra communication delay compared to the P2P implementation, but the performance degradation is negligible for the NoC implementation, while the bus implementation shows significant performance degradation, as shown in Section 5.

Since we use the same set of IP cores to implement all three communication architectures, we discuss next the design of individual processing elements.

3.1 Design of the Processing Elements

Each processing element is implemented using synthesizable Verilog. The area occupied by the individual cores in the design is summarized in Table I. The second column in Table I shows the number of slices that a core takes in a Xilinx XC2V3000 FPGA when implemented without any wrapper (i.e., network interface (NI)).[2] However, before using the core in a real design, we need to add a wrapper such that it can successfully communicate with other nodes in the network. The wrapper for the P2P communication architecture has a simple flow control and I/O buffers. Therefore it requires only 113 slices. On the other hand, the wrapper for the bus communication has to perform the bus request and release before and after packet sending. Likewise, the interface used in the

[2]Communication interfaces in all implementations are referred to as network interfaces (NIs).

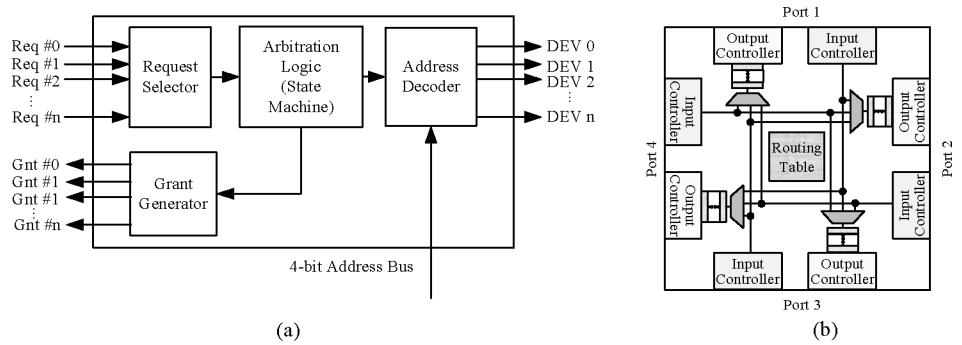

Fig. 4. Block diagram of the: (a) bus control unit used in the bus; and (b) on-chip router in the NoC implementation.

NoC implementation has to perform packetization and depacketization operations. Hence, the wrappers used for bus and NoC implementations take more resources than those used in the P2P implementation. More specifically, the wrapper module used in the bus implementation takes 190 slices, while the wrapper used in the NoC implementation requires 161.

The remaining columns in Table I show the area of cores (i.e., number of slices and BlockRAMs) with the wrapper area included. Although one wrapper for the P2P architecture is smaller than its bus and NoC counterparts, the cores instantiated for the P2P architecture actually have larger area, since the modules have more than one dedicated interface. On the other hand, for bus and NoC implementations, each module has only one interface which connects it directly to the shared bus and router, respectively. The number of NIs required for each implementation is given in Table I. We note that the slice numbers in Table I do not simply come from summing the area of the core without the NI and the area of the NI itself; they come directly from the FPGA synthesis tool *after* optimization. So, there exist small differences between the analytical calculation and real measurements due to the optimization step during the NI integration process in FPGA synthesis tools.

3.2 Bus Control Unit Design (BCU)

In the bus-based design, each module can send and receive packets only through an arbitration logic implemented in the BCU. The arbitration should take place at the start of every new packet; this unit determines which module has the right to access the bus. The block diagram of the BCU is depicted in Figure 4(a). It consists of a *request selector, a grant generator, arbitration logic,* and an *address decoder.* At the first step, the bus control unit accepts the bus request signals from each module using dedicated channels and decides which request can be accepted based on the arbitration policy. After that, it sends the grant signal to the winning module. We use a priority-based arbitration scheme and the arbitration process requires 3 clock cycles. The BCU occupies 67 slices on the Xilinx XC2V3000 FPGA when 7 modules are connected to it; this area changes slightly when the number of connected modules increases.

Table II. Area Taken by Routers in a NoC Implementation

Router Type	No. of Slices
3-port router	219
4-port router	304
5-port router	397
6-port router	503

Synthesis is performed for a Xilinx XC2V3000 FPGA.

3.3 Router Design

The key component of the NoC implementation is the on-chip router. The block diagram of the on-chip router employed in our design is shown in Figure 4(b). Due to the moderate buffer requirements, the router implements wormhole routing. The network channel width and flit size is set to 16 bits. Each packet in the network contains one block in the current frame, namely, 8×8 pixels, each represented by 16 bits. Consequently, the packets are divided into 64 body flits, and 1 header flit which carries the address information. The FIFO buffers implemented at the output ports of the router have a depth of 16 flits.[3] Therefore, only a part of the packet can be stored in a single queue. Finally, the router takes 4 cycles to process the header flit. After a routing decision is made and the header flit put to the output channel, the remaining flits follow the header flit in a pipelined fashion.

Depending on the network topology, we use routers with 3, 4, 5, or 6 ports. The area occupied by these routers, as a function of the number of ports, is summarized in Table II. Although none of these routers is optimized for area, their area overhead is smaller than the overhead incurred by the dedicated links and network interfaces of the P2P implementation; we discuss this issue in more detail in Section 4.

4. EVALUATION OF DESIGN AREA

In this section, we compare the areas occupied by the complete MPEG-2 encoder implemented with P2P, bus, and NoC communication architectures. Besides the absolute value of area for the baseline designs in Figures 1 and 2, we also analyze how the area scales with increasing numbers of cores. For this reason, we also implement a version of the MPEG-2 encoder which has two separate ME modules.

As shown with dotted lines in Figure 5(a), adding one more ME module to the P2P architecture requires four extra links and eight extra network interfaces. In addition, the IB, MC, FB, and VB modules have to be modified to allow the integration of the second ME module. On the other hand, the impact of this additional core on the bus and NoC implementations is only *local,* since we need to add only one more link from the newly inserted module to the bus/router and

[3]A buffer depth of 16 words is selected, since the minimum depth of the block RAMS in XC2V3000 FPGA is 16. Even if the depth is forced to be smaller, the resource utilization in the target FPGA remains the same.

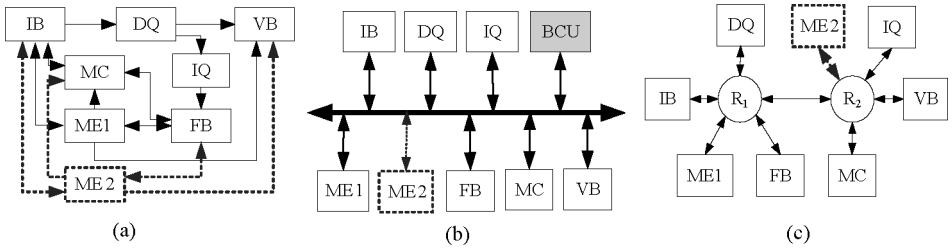

Fig. 5. Implementation of MPEG-2 encoder with 2 MEs using: (a) P2P; (b) bus and (c) NoC communication architectures.

one extra port inside the router (see Figures 5(b) and 5(c)). As a result, the effort of adding additional modules in order to increase the design parallelism and the penalty in area are both much smaller for bus and NoC designs.

4.1 Analytical Comparison of Area

In this section, we analytically estimate the area of each implementation and then use these calculations to compare the P2P, bus, and NoC implementations. We analyze the area occupied by logic components such as cores and network interfaces separately from that occupied by links because the interconnection mechanism of FPGAs is quite different from the one in a real silicon implementation due to the reconfiguration capability of FPGAs. The area occupied by computation and communication resources in the design is estimated by using the area of the individual modules, such as processing elements and all network interfaces.

In the following, N_C denotes the total number of cores in the design, $c_i (i = 1, \ldots, N_C)$ represents core i, while $A(.)$ gives the area of its arguments. Finally, NI_{P2P} NI_{bus}, and NI_{NoC} stand for network interfaces which correspond to the P2P, bus, and NoC designs, respectively. Using this notation, the area of the P2P implementation can be found

$$A_{P2P} = \sum_{i=1}^{N_C} A(c_i) + 2N_L \cdot A(NI_{P2P}), \tag{1}$$

where N_L is the number of links in the design. Similarly, the area of the bus- and NoC-based designs is expressed as

$$A_{bus} = \sum_{i=1}^{N_C} A(c_i) + N_C \cdot A(NI_{bus}) + A(BCU) \tag{2}$$

$$A_{NoC} = \sum_{i=1}^{N_C} A(c_i) + N_C \cdot A(NI_{NoC}) + \sum_{i=1}^{N_R} A(R_{NoC_i}), \tag{3}$$

where $R_{NoC_i} (i = 1, \ldots, N_R)$ denotes router i, and N_R is the total number of routers in the network.

The analytical estimation for the area of P2P, bus, and NoC-based implementations with 1, 2, 4, and 8 ME modules is shown in Figure 6(a). As we can see, the P2P implementation is consistently larger than the bus and NoC

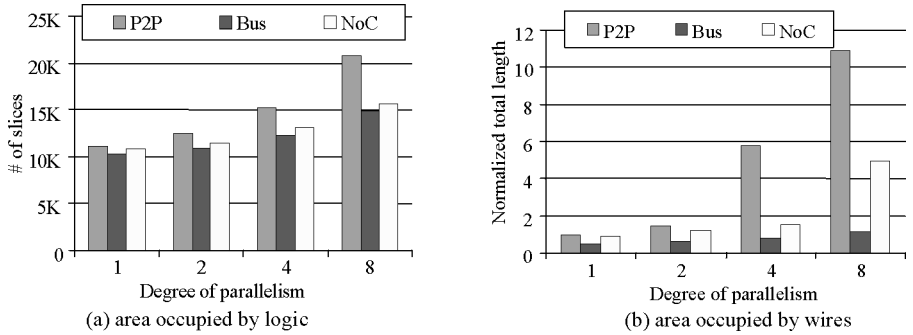

Fig. 6. Area comparisons of the MPEG-2 encoder implemented using P2P, bus, and NoC architectures for increasing levels of parallelism.

Table III. Comparison of Analytical and Measured Area (number of slices) for P2P, Bus, and NoC Communication Architectures

	P2P		Bus		NoC	
	1 ME	2 ME	1 ME	2 ME	1 ME	2 ME
Measurement	10,812	11,750	10,012	10,537	10,442	11,031
Analytical	11,125	12,502	10,262	10,925	10,794	11,428
Difference (%)	2.9	6.4	2.5	3.7	3.4	3.6

implementations. More importantly, the area of the P2P implementation scales considerably worse. For example, the P2P version is more than 24.7% larger than the NoC version for the implementation with 8 ME modules, while the difference in performance is only about 4.4%. This shows that P2P architectures scale poorly in terms of area and that they are not suitable for designs involving a large number of cores. On the other hand, NoC-based implementation scales as well as bus-based with an increasing number of cores. For instance, the NoC version has about 4.6% overhead compared to the bus version, even for the 8 ME implementation.

Unlike logic, the interconnect area is difficult to estimate with simplified models, since it depends on logic placement and space complexity. Hence, instead we use an in-house communication-aware floorplanner for P2P, bus, and NoC implementations to determine the total length of all wires [Hu et al. 2002]. The scaling of the wiring area for each implementation is depicted in Figure 6(b). This figure clearly shows that the wiring area of the P2P implementation scales poorly (similar to the area of logic), while the area of the NoC implementation scales considerably better.

4.2 Experimental Comparison of Area

In addition to our analytical estimations obtained using the area of individual components, we present next the area of the complete design measured using the FPGA prototype. The measured area for the P2P implementation with a single ME is 10,812 slices, as shown in Table III; this is about 2.9% smaller than our analytical estimation. This small difference is due to the optimization process performed at the borders of cores during synthesis of the complete design.

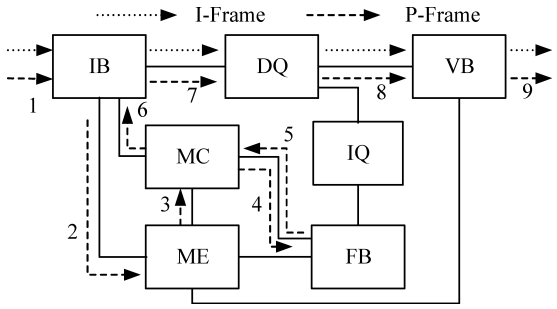

Fig. 7. Critical paths for I-Frame and P-Frame encodings are shown with dotted and dashed lines, respectively.

Table III summarizes both the analytically estimated and truly measured values for all P2P, bus, and NoC implementations with 1 and 2 ME modules. We observe that the measured values are always within 7% of the analytical predictions.

To summarize, the bus-based implementation of the encoder occupies the smallest area. The NoC implementation of the same application has only a small overhead compared to the bus implementation, but scales equally well. In contrast, the P2P implementation scales poorly.

5. PERFORMANCE EVALUATION

In this section, we develop an analytical model to estimate the encoder throughput, which is the performance metric of interest. Then, we compare the throughputs of the P2P, bus, and NoC designs using this model and measured data.

5.1 Analytical Comparison of Performance

Since all P2P, bus, and NoC implementations support pipelined and parallel execution, the encoder throughput is determined by the latency of the critical path in the data flow, as illustrated in Figure 7. More precisely, the execution time of the bottleneck module on the critical path is given by

$$T_{critical} = max\{T_i\} \;\; i \in S_C, \tag{4}$$

where S_C is set of computational nodes and communication links on the critical path, and T_i is the latency of ith node or link. From MPEG-2 functionality, it follows that for I-frames, the critical path is given by $S_C = \{T_{IB} \rightarrow T_{DQ} \rightarrow T_{VB}\}$. Similarly, the critical path for the predicted P-frames can be expressed as $S_C = \{T_{IB} \rightarrow T_{ME} \rightarrow T_{MC} \rightarrow T_{FB} \rightarrow T_{MC} \rightarrow T_{IB} \rightarrow T_{DQ} \rightarrow T_{VB}\}$. Since the critical path for P-frames is significantly longer, we next only consider the performance analysis of the P-frame encoding.

Among modules on the critical path for the P-frame encoding, the ME module requires the largest computational time. Hence, its latency value directly determines the throughput of the system.

$$Throughput = \frac{1}{T_{ME} + L_{Data}} \tag{5}$$

Here, T_{ME} is the time required for motion estimation, and L_{Data} is the time for receiving data, N_{data} from the IB and FB modules. We note that T_{ME} is the same for all designs, while L_{Data} varies. So the performance differences among the three implementations come from the performance difference in the communication channel. In the P2P architecture, there is no contention between the modules; indeed, in this case, each module has a dedicated channel, so all modules can fully use the original link bandwidth W. More precisely, for the P2P architecture, L_{Data} is given by N_{Data}, divided by the link bandwidth W.

$$L_{Data}^{P2P} = \left\lceil \frac{N_{Data}}{W} \right\rceil \tag{6}$$

On the other hand, for the bus implementation all modules use only one shared link. Hence, they compete with each other to use the shared bus. As a result, the link bandwidth that the source i.e., IB and FB modules can use decreases as the number of competing module increases. So L_{Data} for the bus can be calculated as follows.

$$L_{Data}^{Bus} = L_R + \left\lceil \frac{N_{Data}}{\rho_{Bus} W} \right\rceil, \tag{7}$$

where L_R is the arbitration delay in the BCU and ρ_{Bus} is the ratio of the data volume generated by the IB and FB modules to that generated by all remaining modules.

Similarly, for NoCs, we calculate the communication time using the latency formula for wormhole routing [Duato et al. 2002].

$$L_{Data}^{NoC} = H \cdot L_R + \left\lceil \frac{N_{Data} - W}{\rho_{NoC} W} \right\rceil, \tag{8}$$

where $H$is the hop count between source and destination, L_R is the time it takes to route the header flit, and ρ_{NoC} is the ratio of the data volume generated by the IB and FB modules to that generated by all remaining modules. The first term in this equation gives the time it takes to route the header flit, while the second gives the latency of the remaining flits, since they all follow the header flit in a pipelined manner.

The throughput of these implementations as a function of the number of ME modules in the design is plotted in Figure 8. Using Eq. (6), the throughput of the P2P implementation is found to be 47.0 frames/sec for a CIF frame of size 352×288. From analytical calculations, the throughput of the corresponding bus and NoC implementations is 38.9 and 46.4 frames/sec, respectively. We note that, based on analysis, the NoC implementation achieves a throughput very close to the P2P version, which provides the ultimate communication performance.

As explained in the previous section, the bottleneck module in both designs is the ME module. For this reason, by duplicating this module, we expect a significant improvement in the throughput. The encoder implementation with two ME modules shows that this is indeed the case. Specifically, according to Eq. (6), the throughput of the P2P implementation almost doubles if a second ME module is added (see Table IV). The corresponding improvement in performance of the NoC implementation is about 93%; this is comparable to the P2P implementation.

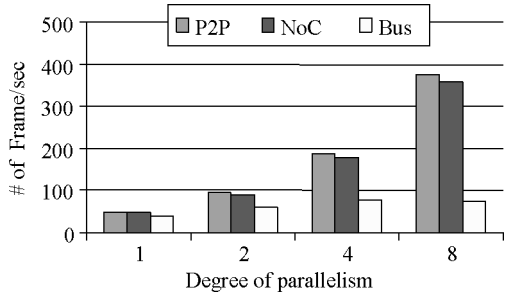

Fig. 8. Throughput comparison of various MPEG-2 encoder implementations.

Table IV. Analytical vs. Measured Throughput (frame/sec per CIF frame)

	P2P		Bus		NoC	
	1 ME	2 ME	1 ME	2 ME	1 ME	2 ME
Measurement	46.5	87.4	36.2	59.1	45.6	83.2
Analytical	47.0	93.9	38.9	60.2	46.4	89.6
Difference (%)	1.0	7.4	2.1	2.0	1.8	7.2

The throughput of all implementations as a function of increasing number of ME modules is shown in Figure 8. We observe that the NoC implementation scales as well as the P2P in terms of throughput, while the bus-based implementation scales poorly. However, one should note that beyond a certain degree of parallelism, the communication itself becomes the bottleneck and therefore the NoC performance saturates. It is possible to eventually stretch the performance beyond this point by customizing the network topology [Ogras and Marculescu 2005; Srinivasan et al. 2004].

5.2 Experimental Comparison of Performance

The accuracy of the performance estimates in Eqs. (6)–(8) are validated against measured data on the FPGA prototype. The analytical throughput estimations are in good agreement with the measured ones, as summarized in Table IV. The *measured throughput* of the P2P implementation with one ME module is 46.5 frames/sec, which is very close to the analytical estimation. Similarly, the measured throughput of the NoC implementation (45.6 frames/sec is very close to that of the P2P implementation, as predicted analytically. On the other hand, the bus-based implementation reaches only 36.2 frame/sec, as shown in table.

We also measured the throughput of the encoders with 2 ME modules, as summarized in Table IV. The throughput of the P2P implementation rises to 87.4 frames/sec, which implies about an 88% improvement. Similarly, the throughput of the NoC design goes up to 83.2 frames/sec, showing a comparable improvement; this justifies our analytical estimations.

To summarize, the P2P implementation achieves the best performance and its throughput scales very well as the number of cores increases. The NoC-based implementation has a slightly smaller throughput, but it scales equally well with the P2P-implementation. In contrast to the P2P- and NoC-based implementations, the throughput of the bus-based implementation scales poorly.

Combined with the area comparisons, we conclude that the NoC approach provides a good tradeoff between area and performance.

6. ENERGY AND POWER CONSUMPTION EVALUATION

As battery-powered devices become popular, energy issues gain more importance. For this reason, this section evaluates the energy and power consumptions of the MPEG-2 encoder implemented using P2P, bus, and NoC communication architectures.

The activity-based model $P = \alpha CV^2 f$ is the most commonly used model for power estimation at all levels of abstractions. However, as the complexity of systems increases, obtaining accurate C and α values becomes difficult. Therefore, we use a system-level approach to reduce the simulation time and complexity of the work involved, as described next.

6.1 Analytical Evaluation of Energy and Power Consumption

The total system energy consumption can be divided into two components: *computational* energy consumption E_{COMP} and *communication* energy consumption E_{COMM}. In turn, the computational energy can be expressed as

$$E_{COMP} = \sum_{i=1}^{N_C} E(c_i), \tag{9}$$

where $E(c_j)$ denotes the energy consumption of any component c_i, and N_C represents the total number of computational modules. Communication energy, on the other hand, consists of energy dissipated in the link L, network interface NI, and arbitration logic (e.g., arbiters in the bus implementation and routers in the NoC implementation), namely,

$$E_{COMM} = \sum_{i=1}^{N_L} E(L_i) + \sum_{i=1}^{N_{NI}} E(NI_i) + \sum_{i=1}^{N_R} E(R_i), \tag{10}$$

where N_L and N_{NI} represent the number of links, and network interfaces in the network, respectively. The number of arbiters in the bus implementation and of routers in the NoC implementation are both denoted by N_R. Since the P2P implementation does not have any arbiter, N_R is equal to zero in the P2P implementation.

Depending on the operating mode, the IP cores are characterized by different power consumption values. Therefore, we can analyze the energy consumption of any individual module $E(c_j)$ as follows:

$$E(c_i) = \sum_{j=1}^{N_{LP}} (\pi_j P_j) \cdot t, \tag{11}$$

where N_{LP} is the number of distinct power states, π_j is the probability that the module is in state j during execution time t, and P_j is the power consumption which characterizes the jth power state.[4] The same calculation can be also used

[4]Here, the power dissipation during the transitions among different power modes is neglected, since this represents a small fraction of the energy consumption in regular power states.

Table V. Power Consumption (mW@100MHz) for Each Computational Node In the Network

Node	Power in *idle* Mode	Power in *active* Mode
IB	20	35
DQ	321	936
IQ	353	1,465
FB	183	299
ME	128	290
MC	60	87
VB	141	364

Table VI. Power Consumption (mW@100MHz) over the Communication Channel

Resource	Mode	Power Consumption (mW@100MHz)		
		P2P	Bus	NoC
Interface	*Idle*	33	29	38
	Receive	76	76	74
	Send	79	83	77
	Receive + Send	96	NA	100
Router (Arbiter)	*Idle*	NA	6	87
	Active		8	140 (1 port)
				152 (2 port)
				190 (3 port)
				232 (4 port)
Link	*Shortest*	12		
	Middle	16		
	Longest	19		

to calculate the energy consumption of the communication components (i.e., L, NI, and R).

In order to achieve accurate results, we measure the power consumption of all individual IPs and communication components using a cycle-accurate energy measurement tool based on the technique presented in [Lee et al. 2005]. Then, these individual power values are used to compute the power consumption of the entire system.

As seen from Eq. (11), having accurate power consumption estimates is critical for the system energy characterization. Toward this end, we use two levels (i.e., *idle* and *active*) of power states to characterize the individual computational nodes, as summarized in Table V. We note that the power consumption of the ME module dissipates less power than those of the DQ and IQ modules. However, the ME module consumes much more energy than these two modules because its operation time is about ten times longer.

For the communication components, we use a larger number of power states, as summarized in Table VI. To accurately compute the link power consumption, we use an in-house communication-aware floorplanner to determine the length of all links. Three typical values (which are classified as shortest, middle, and longest) are shown in Table VI. After that, we measure the corresponding link power consumption in the FPGA prototype by varying the link length. These measured values are later used to estimate the link power consumption. Finally, we characterize the power consumption of the router, as shown in the last

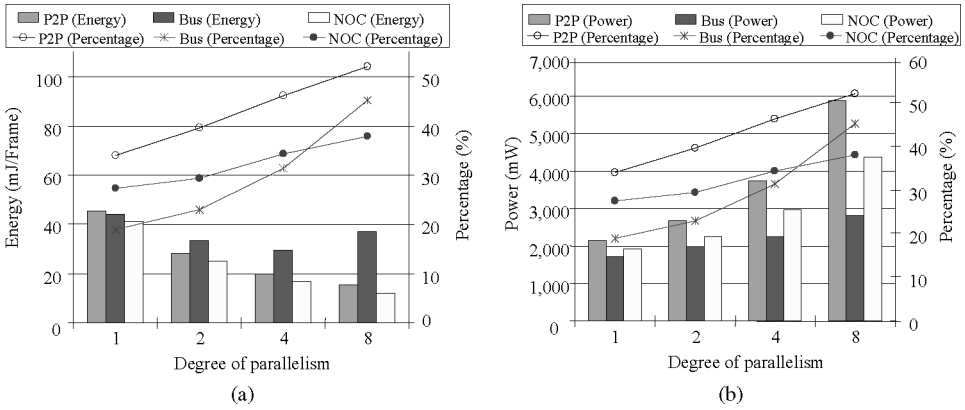

Fig. 9. (a) Energy consumption; and (b) power consumption (mW@ 100MHz) as a function of the degree of parallelism for P2P, bus, and NoC implementations.

column of Table VI. The table also shows the router power consumption when the number of active ports varies from one to four.

From a methodology standpoint, we first obtain the utilization and operation modes of all components in the design; this is done by dynamically profiling the detailed Verilog simulations obtained from encoding real video clips. Then, we measure the energy consumption values of each module and use these values to estimate the energy and power consumption of the entire design using Eqs. (9)–(11).

Figure 9(a) shows that the energy consumption for the NoC implementation is consistently smaller than its P2P and bus counterparts for different levels of parallelism. The P2P implementation has larger energy consumption, since it has more interfaces and links than its NoC counterpart. On the other hand, the energy consumption of the bus-based implementation is large due to the longer time needed to encode real data (i.e., smaller throughput compared to P2P- and NoC-based implementations). The longer encoding time of the bus-based implementation also results in a smaller power consumption, as shown in Figure 9(b).

On the other hand, the power consumption of the P2P implementation is larger than the power consumed by the NoC, even for the baseline implementation. Furthermore, we observe that the NoC design scales better in terms of power consumption compared to the P2P implementation, as shown in Figure 9(b). The power consumption of the P2P architecture scales poorly as the degree of parallelism increases, since the P2P implementation requires a significantly larger number of additional links and network interfaces to accommodate the addition of extra ME modules. More specifically, the NoC-based implementation consumes up to 42% less power compared to the P2P for an implementation involving 8 ME modules. This corresponds to about 17% of the total power consumption, which is quite important when optimizing portable systems.

Another issue is the contribution of communication energy consumption to the overall energy consumption. For this reason, we plot the percentage of the

Table VII. Analytical vs. Measured Energy Consumption (μJ/frame per CIF frame)

	P2P		Bus		NoC	
	1 ME	2 ME	1 ME	2 ME	1 ME	2 ME
Measurement	42.8	27.3	41.2	30.5	37.6	23.4
Analytical	45.5	28.5	44.3	33.7	41.2	24.5
Difference (%)	6.4	4.3	7.7	10.3	9.4	6.8

communication energy and power consumption in Figures 9(a) and 9(b), respectively. For baseline implementations, the communication power consumption of P2P, bus, and NoC designs represents 34%, 27%, and 19% of the total power consumption, respectively. As we can see in the figure, these percentages increases rapidly for P2P and bus-based implementations as a function of the degree of parallelism. This means that communication energy becomes more significant for larger designs. On the other hand, the percentage of communication energy (and power) in the NoC implementation increases much more slow. As a result, the NoC approach clearly provides a more scalable solution than P2P and bus-based architectures in terms of energy and power consumption.

6.2 Experimental Comparison of Energy and Power Consumption

The accuracy of the analytical energy estimations obtained using Eqs. (9)–(11) is validated against the measured data on the FPGA, similar to the area and performance comparisons in Section 4.2 and Section 5.2. For better accuracy, the energy consumption of complete designs is measured using a cycle-accurate energy measurement tool based on the technique presented in [Lee et al. 2005].

As shown in Table VII, the analytical estimations are in good agreement with the measured values. More precisely, the energy consumed per encoding a single CIF frame is 42.8 μJ for the P2P implementation with 1 ME module. Similarly, the bus-based implementation with 1 ME module consumes 41.2 μJ. On the other hand, the NoC-based implementation requires 37.6 μJ/frame, which is about 9% smaller than the bus-based implementation.

We also measured the energy consumption of video encoders built with 2 ME modules, as summarized in Table VII. For this case, the P2P, bus, and NoC-based consume 27.3, 30.5, and 23.4 μJ/frame, respectively. In terms of scalability, we observe that the difference between the NoC-based and other implementations becomes more pronounced as we increase the number of ME modules. To be more specific, the NoC version consumes about 23% less energy compared to bus-based and 14% less energy compared to P2P implementations.

Finally, the leakage energy is mostly a function of the target FPGA device, since we did not perform any specific optimizations to minimize the leakage power consumption. Using our energy measurement tool, the leakage power consumption of the target FPGA device is found to be slightly more than 20 mW.

To summarize, the NoC-based implementation achieves the smallest energy consumption per frame, and offers the best scalability. The P2P implementation suffers from needing a large number of communication interfaces and dedicated

links, while the bus-based implementation suffers from excessively long encoding times.

7. CONCLUSION

Integrating an increasingly large number of IP cores on the same chip makes the design of communication architectures for future SoCs a challenging problem. As a result, design-space exploration with emphasis on the communication aspects becomes crucial. Towards this end, this article presented a comprehensive evaluation of P2P, bus-based, and NoC communication architectures targeting multimedia applications. Our study involves: (1) a concrete proof of an FPGA-based NoC implementation running an MPEG-2 encoder application, and P2P and bus-based implementation counterparts of the same application; (2) analytical estimations of area, performance, and energy consumption which are backed up by real measurements on our FPGA prototype. Through analytical estimations and direct measurements on the FPGA prototype, we support the following conclusions:

—The performance of the NoC-based implementation is very close to that of the P2P for the same application. Moreover, the scalability analysis based on duplicating the bottleneck module in the MPEG-2 design shows that the performance of the NoC design scales as well as the P2P, while the bus-based implementation scales much more poorly.

—In terms of area, the NoC scales as well as the bus-based implementation. However, the P2P implementation does not scale well due to the overhead involved in redesigning the interfaces. Moreover, the design effort for adding new cores to an existing design is much smaller for the NoC case as compared to P2P.

—The energy consumption of the NoC-based implementation is smaller than both P2P and bus-based implementations and it scales much better with the number of extra ME modules added to the base design.

In summary, our study supports the idea that the NoC design scales very well in terms of area, performance, power/energy consumption, and overall design effort, while the P2P architecture scales poorly on all accounts except performance. By contrast, the bus-based architecture scales poorly in terms of performance and energy consumption.

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.* 11128–11129.

BENINI, L. AND DE MICHELI, G. 2002. Networks on chips: A new SoC paradigm. *IEEE Comput. 35*, 1, 70–78.

BOLOTIN, E., CIDON, L., GINOSAR, R., AND KOLODNY, A. 2004. Cost considerations in network on chip. *Integration, VLSI J. 38*, 1.

DALLY, W. AND TOWLES, B. 2001. Route packets, not wires: On-Chip interconnection networks. In *Proceedings of the Design Automation Conference.* 684–689.

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

JANTSCH, A. AND TENHUNEN, H. 2003. *Networks on Chip.* Kluwer, Hingham, MA.

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.* 166–173.

HU, J., DENG, Y., AND MARCULESCU, R. 2002. System-Level point-to-point communication synthesis using floorplanning information. In *Proceedings of the Asia and South Pacific Design Automation Conference.*

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

KIM, M., KIM, D., AND SOBELMAN, G. E. 2005. MPEG-4 performance analysis for a CDMA network-on-chip. In *Proceedings of the International Conference on Communications Circuits and Systems.*

LEE, H., LEE, K., CHOI, Y., AND CHANG, N. 2005. Cycle-Accurate energy measurement and characterization of FPGAs. *Analog Integrated Circ. Signal Process. 42*, 3.

LEE, K, LEE, S., KIM, S., CHOI, H., KIM, D., KIM, S., LEE, M., AND YOO, H. 2004. A 51mW 1.6GHz on-chip network for low-power heterogeneous SoC platform. In *Proceedings of the International Solid-State Circuits Conference.*

LIANG, J., LAFFELY, A., SRINIVASAN, S., AND TESSIER, R. 2004. An architecture and compiler for scalable on-chip communication. *IEEE Trans. VLSI Syst. 12*, 7.

MURALI, S., COENEN, M., RADULESCU, A., GOOSSENS, K., AND DE MICHELI, G. 2006. Mapping and configuration methods for multi-use-case networks on chips. In *Proceedings of the Asia and South Pacific Design Automation Conference.* 146–151.

OGRAS, U. Y. AND MARCULESCU, R. 2005. 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., 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 Synthesis.*

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.*

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

XU, J., WOLF, W., HENKEL, J., CHAKRADHAR, S., AND LVET, T. 2004. A case study in networks-on-chip design for embedded video. In *Proceedings of the Design, Automation and Test in Europe Conference.*

WANG, H., ZHU, X., PEH, L., AND MALIK, S. 2002. Orion: A power-performance simulator for interconnection networks. In *Proceedings of the 35th Annual International Symposium on Micro-Architecture.*

WOLKOTTE, P. T., SMIT, G. J. M., KAVALDJIEV, N., BECKER, J. E., AND BECKER, J. 2005. Energy model of networks-on-chip and bus. In *Proceedings of the International Symposium on System-on-Chip.*

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.*

Received September 2006; revised February 2007; accepted March 2007