On-Chip Stochastic Communication*

Tudor Dumitraş and Radu Mărculescu
Electrical and Computer Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213
{tdumitra,radum}@ece.cmu.edu

Abstract

As CMOS technology scales down into the deep-submicron (DSM) domain, the costs of design and verification for Systems-On-Chip (SoCs) are rapidly increasing due to the inefficiency of traditional CAD tools. Relaxing the requirement of 100% correctness for devices and interconnects drastically reduces the costs of design but, at the same time, requires that SoCs be designed with some system-level fault-tolerance. In this paper, we introduce a new communication paradigm for SoCs, namely stochastic communication. The newly proposed scheme not only separates communication from computation, but also provides the required built-in fault-tolerance to DSM failures, is scalable and cheap to implement. For a generic tile-based architecture, we show how a ubiquitous multimedia application (an MP3 encoder) can be implemented using stochastic communication in an efficient and robust manner. More precisely, up to 70% data upsets, 80% packet drops because of buffer overflow, and severe levels of synchronization failures can be tolerated while maintaining a low latency.

1. Introduction

As modern systems-on-chip (SoCs) are becoming extremely complex, the Intellectual Property (IP) based design has become widely accepted as a reuse paradigm. This methodology makes a clear distinction between *computation* (the tasks performed by the IPs) and *communication* (the interconnecting architecture and the communication protocols used). Currently, the traditional CAD tools and methodologies are very inefficient in dealing with a large number of IPs placed on a single chip.

A recently proposed platform for the on-chip interconnects are the networks-on-chip (NoCs) [6], where the IPs are placed on a rectangular grid of tiles and the communication between modules is implemented by a stack of networking protocols. However, the problem of defining such *communication protocols* for NoCs has not been addressed so far. Also it does not seem to be an easy matter, as the constraints are very tight and the important resources used in traditional data networks are not available at chip level.

*Research supported by NSF CCR-00-93104, DARPA/Marco Gigascale Research Center (GSRC), and SRC 2001-HJ-898.

Furthermore, as the complexity of designs increases and the technology scales down into the deep-submicron (DSM) domain, devices and interconnect are subject to new types of malfunctions and failures that are harder to predict and avoid with the current SoC design methodologies. These new types of failures are impossible to characterize using deterministic measurements so, in the near future, *probabilistic* metrics, such as average values and variance, will be needed to quantify the critical design objectives, such as performance and power [12].

Relaxing the requirement of 100% correctness in operation drastically reduces the costs of design but, at the same time, requires SoCs be designed with some degree of system-level *fault-tolerance* [15]. Distributed computing has dealt with fault-tolerance for a long time; unfortunately most of those algorithms are unlikely to be useful, because they need a lot of resources which are not available on-chip. Generally, simple deterministic algorithms do not cope very well with random failures [11]. On the other hand, the cost of implementing full-blown *adaptive routing* [13] for NoCs is prohibitive because of the need of very large buffers, lookup tables and complex shortest-path algorithms.

To address these unsolved problems, the present paper introduces a new communication paradigm called *on-chip stochastic communication*. This is similar, although *not* identical, to the proliferation of an epidemic in a large population [3]. To explain our approach, we place ourselves in a NoC context, where the network is composed of tiles organized as a rectangular grid, and the IPs are placed on these tiles. The IPs communicate using a probabilistic broadcast scheme, similar to the *randomized gossip protocols* [10]. If a tile has a packet to send, it will forward the packet to a randomly chosen subset of the tiles in its neighborhood. This way, the packets are *diffused* from tile to tile to the entire NoC. Every IP then selects from the set of received messages only those that have its own ID as the destination.

This simple algorithm achieves many of the desired features of the future NoCs. As shown below, the algorithm provides:

- *separation between computation and communication*, as the communication scheme is implemented in the network logic and is transparent to the IPs;
- *fault-tolerance* since a message can still reach its destination

despite severe levels of DSM failures on the chip;

- *extremely low latency* since this communication scheme does not require retransmissions to deal with incorrect data;
- *low production costs* because the fault-tolerant nature of the algorithm eliminates the need for detailed testing/ verification;
- *simplified design* since NoCs are easy to replicate;
- *design flexibility* since it provides a mechanism to tune the tradeoff between performance and energy consumption.

In summary, we propose a fault-tolerant solution for the NoC communication which lowers the production costs, simplifies the design, and can be customized for various applications and architectures, while maintaining high levels of performance.

This paper is organized as follows: at first, we review previous work relevant to this paper. In Section 3 we introduce our approach and the metrics for performance evaluation. In Section 4 we show experimental results that clearly indicate the great potential of this approach. We conclude by summarizing our main contribution.

2. Previous Work

One of the important mathematical challenges of this century has been developing analytical models for the proliferation of diseases. The earliest research in this direction [3] showed that, under certain conditions, epidemics are spread *exponentially fast*. Much later, [7] proposed to use a probabilistic broadcast algorithm, the *randomized gossip*, for the lazy update of data objects in a database replicated at many sites and proved that this scheme behaves very much like the spreading of an epidemic. Several networking protocols are based on the same principles, as the gossip protocols proved to be very *scalable* and to maintain a *steady throughput* [4, 2]. Recently, these types of algorithms were applied to networks of sensors [8]. Their ability to limit communication to local regions and to support light-weight protocols, while still accomplishing their task, is appealing to applications where power, complexity and size constraints are critical.

In this paper, we demonstrate that this paradigm can be successfully applied to SoC communication as well, especially for regular NoC architectures. The experimental results on a popular multimedia application (an MP3 encoder) show the effectiveness of our approach in a practical setup. However, bringing this new communication paradigm closer to silicon, raises specific challenges as: defining a fault-tolerant communication protocol, modeling data upsets and their impact on NoC communication, assessing the impact of synchronization faults between different clock domains on performance, etc. Our paper is an attempt to address all these issues, outlining a possible solution. From this perspective we believe that our ideas open an important research direction with deep implications in the design methodology of NoCs.

3. Description of Approach

Traditionally, on-chip interconnects are considered to be 100% reliable and hardware components do not check received data for integrity. However, this is not realistic since such upsets can occur in DSM circuits; specifically:

- the high transistor density makes cross-talk noise and high field effects impossible to control;
- as routing a single clock tree for large circuits is difficult and leads to high power consumption, modern chips have several clock domains and may be designed globally-asynchronous, locally-synchronous (GALS) [5]. For such designs the communication can be affected by synchronization errors;
- as the area available on-chip is reduced, the buffer sizes are also reduced, so overflows can easily occur.

Traditional acknowledgement / retransmission schemes, widely used in large-scale networks, can thus lead to severe loss of performance and even deadlock. We propose a new paradigm for on-chip communication, which is described next.

3.1. Stochastic Communication: Basics

We propose *on-chip stochastic communication*, which implements NoC communication using a *probabilistic broadcast* algorithm. The behavior of such an algorithm is similar to the spreading of a rumor within a large group of friends. Assume that, initially, only one person in the group knows the rumor. Upon learning the rumor, this person (the initiator) passes it to someone (confidant) chosen at random. At the next round, both the initiator and the confidant, if there is one, select independently of each other someone else to pass the rumor to. The process continues in the same fashion; namely, everyone informed after t rounds passes the rumor at the $(t+1)^{\text{th}}$ round to someone selected at random independently of all other past and present selections. Such a scheme is called a *gossip algorithm* and is known in biology to model the spreading of an epidemic [3].

Let $I(t)$ be the number of people who have become aware of the rumor after t rounds ($I(0) = 1$) and let $S_n = \min\{t : I(t) = n\}$ be the number of rounds until n people are informed. In order to know how fast this algorithm spreads the rumor, we have to estimate S_n. A fundamental result states that $I(t)$ is very close to its deterministic approximation defined by a difference equation:

$$I(t+1) = n - [n - I(t)]\, e^{-\frac{I(t)}{n}}, \qquad I(0) = 1 \qquad (1)$$

and $S_n = \log_2 n + \ln n + O(1)$ as $n \to \infty$ with probability 1 [14]. Therefore, after $O(\log_2 n)$ rounds (n represents the number of nodes) all the nodes have received the message *with high probability (w.h.p.)*[1] [10]. For instance, in Fig. 1, in less than 20 rounds as many as 1000 nodes can be reached.

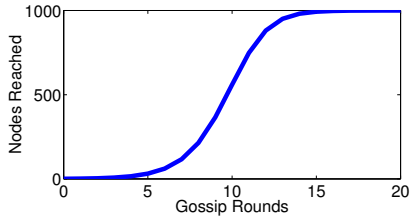

Fig. 1. Gossip in a fully connected network

[1]The term *with high probability* means with probability at least $1 - O(n^{-\alpha})$ for some positive constant α.

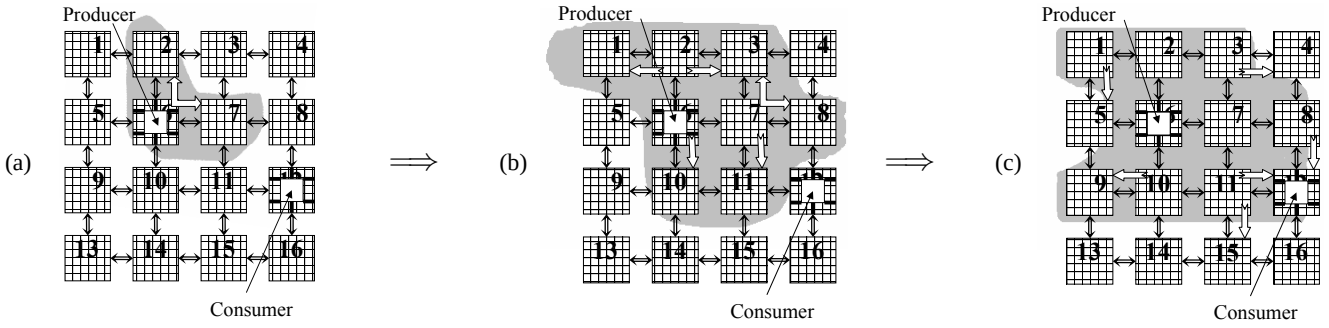

Fig. 2. Producer – Consumer application implemented on a stochastically communicating NoC

Conversely, after t rounds the number of people that have become aware of the rumor is an exponential function of t, so this algorithm spreads rumors *exponentially fast*.

3.2. What is specific to NoCs?

The analogy we make for the case of on-chip networks is that the tiles of the NoC are the "gossiping friends" and the packets transmitted between modules are the "rumors" (see Fig. 3). Since any friend in the original gossip setup is able to gossip with anyone else in the group, the above analysis can be applied directly to the case of a fully connected network, like the one in Fig. 3-a). However, because of its high wiring demands, such a fully connected architecture is *not* a reasonable solution for SoCs. Therefore, for NoCs, we have to consider a grid-based topology (Fig. 3-b) since this is much easier and cheaper to implement on silicon. Although the theoretical analysis in this case is an open research question, our experimental results show that the messages can be spread explosively fast among the tiles of the NoC for this topology as well. To the best of our knowledge, this is the first evidence that gossip protocols can be applied to SoC communication as well.

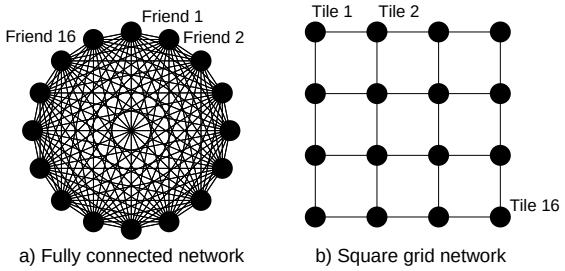

Fig. 3. Topologies for a 16 node network

In the following sections, we will describe algorithm for on-chip stochastic communication and show how it can be used for a real multimedia application (an MP3 encoder).

3.3. Stochastic Communication for NoCs

3.3.1. Example: Producer – Consumer. To better explain the essence of our approach, a simple example of a Producer – Consumer application is shown in Fig. 2. On an NoC with 16 tiles the Producer is placed on tile 6 and the Consumer on tile 12. Suppose the Producer needs to send a message to the Consumer. Initially the Producer sends the message to a randomly chosen subset of its neighbors (ex. tiles 2 and 7 in Fig. 2-a). At the second gossip round, tiles 6, 2, 7 (the Producer and the tiles that have received the message during the first round) forward it in the same manner. After this round, eight tiles (6, 2, 7, 1, 3, 8, 10, 11) become aware of the message and are ready to send it to the rest of the network. At the third gossip round, the Consumer finally receives the packet from tiles 8 and 11. Note that:

- the Producer needs *not* know the location of the Consumer, the message will arrive at the destination *w.h.p.*.
- the message reaches the Consumer *before* the full broadcast is completed; that is the message reaches the Consumer before reaching tiles 14 and 16.

The potential of this communication scheme can be even better appreciated when we evaluate its performance in the presence of failures. For instance, suppose the packet transmitted by tile 8 is affected by a data upset. The Consumer will discard it, as it receives from tile 11 another copy anyway. The presence of such an upset is detected by implementing on each tile an error detection / multiple transmission scheme. The key observation is that, at chip-level, bandwidth is *less* expensive than in traditional macro-networks, because of high-speed buses and interconnection fabrics which can be used for the implementation of an NoC.

3.3.2. Features and Fault Tolerance. The functioning of this architecture with an error detection / multiple transmissions mechanism in place uses the fact that during transmission, packets are protected against data upsets by a cyclic redundancy code (CRC). If an error is discovered, then the packet will be simply discarded. Because a packet is retransmitted many times in the network, the receiver does *not* need to ask for retransmission, as it will receive the packet again anyway. This is why stochastic communication can sustain low-latencies (and therefore high throughput levels) even under severe levels of failures. Furthermore, CRC encoders and decoders are easy to implement in hardware, as they only require one shift register [2].

In our implementation, the links are assumed to be *active*; that is, they have some control logic that can choose randomly whether or not an incoming packet is to be transmitted. This design choice is *not* vital for the performance of the algorithm; the random decision could be implemented in the tiles instead of the links, if so desired. This further emphasizes the separation between computation and communication in the on-chip networks based on our approach.

We also note that, since a message might reach its destination be-

fore the broadcast is completed, the spreading could be terminated even earlier in order to reduce the number of messages transmitted in the network. This is important because this number is directly connected to the bandwidth used and the energy dissipated (see Section 3.5). To do this, we assign a *time to live* (TTL) to every message upon creation and decrement it at every hop until it reaches 0; then the message can be garbage-collected. After a certain threshold, the latency does not improve if the TTL is increased. As seen in Fig. 4, the energy consumption increases almost linearly with the TTL, which suggests that the TTL can be set at the smallest level that ensures the completion of the tasks.

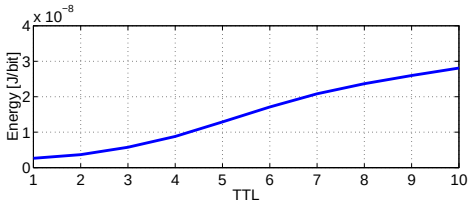

Fig. 4. Dependence of energy on the TTL

3.3.3. The Algorithm. Our algorithm is explained in Figures 5 and 6 (with standard set theory notations). A tile forwards the packet that is available to be sent to all four output ports, then every active link makes a random decision (with probability p) whether or not to transmit the message to its outbound tile (Fig. 6). In Section 4 we will show how this probability can be used to tune the tradeoff between performance and energy consumption.

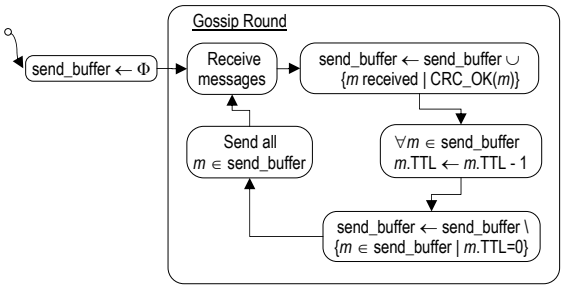

Fig. 5. Operations executed by every node

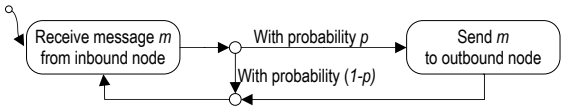

Fig. 6. Operations executed by every link

The evolution of the broadcast is shown in the left part of Fig. 7. The spread is slow in the beginning, but it soon reaches a stage of explosive growth and the whole network becomes aware of the message after a small number of rounds. The shape of these curves is similar for all the values of the transmission probability p. Note the similarity of these results with the ones predicted by theory (Fig. 1). In the right part of Fig. 7, we compare the energy dissipation for two different values of the parameter p. As expected, the algorithm with $p = 0.5$ reduces the power consumption to half the energy needed by the flooding technique (i.e. when $p = 1$).

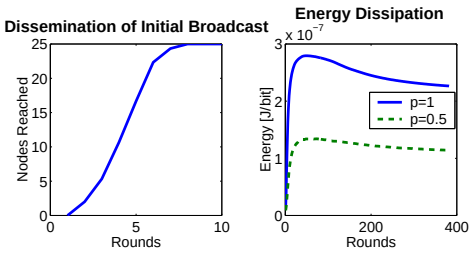

Fig. 7. Performance of the algorithm

3.4. Performance Metrics

A *broadcast round* is the time interval in which a tile has to finish sending all its messages to the next hops; this will usually take several clock cycles. The optimal duration of a round (T_R) can be determined using Eq. 2:

$$T_R = \frac{N_{packets/round}\, S}{f} \tag{2}$$

where f is the maximum frequency of any link, $N_{packets/round}$ is the average number of packets that a link sends during one round (which is application-dependent), and S is the average packet size. Note that, as we *do not* store multiple copies of the same packet in a buffer (see Fig. 5), $N_{packets/round}$ will be proportional to the number of *unique* messages in the network and this will keep the buffers small.

As we will show in Section 4, the fast dissemination of rumors in this algorithm makes it possible to achieve very low latencies. This protocol spreads the traffic onto all the links in the network, reducing the chances that packets are delayed because of congestion. This is especially important in multimedia applications, where a sustainable constant bit rate is highly desirable. Furthermore, the fact that we don't store or compute the shortest paths (as dynamic routing does) makes this algorithm *computationally lightweight*, simpler and easier to customize for every application and interconnection network.

The following parameters are relevant to our analysis:

- The *number of broadcast rounds needed* is a direct measure of the inter-IP communication *latency*;
- The *total number of packets* sent in the network shows the bandwidth required by the algorithm and can be controlled by varying the message TTL;
- The *fault-tolerance* evaluates the algorithm's resilience to abnormal functioning conditions in the network;
- The *energy consumption*, computed with Eq. 3.

3.5. Energy Metrics

In estimating this algorithm's energy consumption, we take into consideration the total number of packets sent in the NoC, since these transmissions account for the switching activity at network level. This is expressed by Eq. 3, where $N_{packets}$ is the total number of messages generated in the network, S is the average size of one packet (in bits) and E_{bit} is the energy consumed per bit:

$$E_{total} = E_{computation} + E_{communication} = E_{computation} + N_{packets}\, S\, E_{bit} \quad (3)$$

$N_{packets}$ can be estimated by simulation, S is application-dependent, and E_{bit} is a parameter from the technology library. As shown in Eq. 3, the total energy consumed by the chip will be influenced by the activity in the computational cores as well ($E_{computation}$). Since we are trying to analyze here the performance and properties of the *communication scheme*, estimating the energy required by the computation is not relevant to the present paper. This can be added, however, to our estimations from Section 4 to get the combined energy.

3.6. Failure Modeling

Several failure models have been identified in the traditional networking literature [9]. However, as some of these models are not very relevant to the on-chip networks, we have developed a fault model that is well suited to the SoC context. Specifically, have the following types of failures:

- *data upsets*: because of crosstalk and electromagnetic interference, the most common type of failures in DSM circuits will be data transmission errors (also called upsets) [15]. Simply stated, if the noise in the interconnect causes a message to be scrambled, a data upset error will occur;
- *buffer overflows*: they are directly reflected in the number of packets dropped during the algorithm.
- *synchronization errors*: as modern circuits span multiple clock domains (as in GALS architectures [5]) and can function at different voltages and frequencies (as in the "voltage and frequency island"-based architectures advocated by IBM), *synchronization errors* are very hard to avoid. In our experiments, we have adopted a tile-based architecture in which every tile has its own clock domain. The actual duration of a round is normally distributed around the optimal value T_R (see Eq. 2), with a variance of σ^2_{synchr}. Whenever the lengths of a round in two neighboring tiles will not match, a synchronization error will occur.

Summarizing, this fault model has the following parameters:

- p_{upset}: probability a packet is scrambled because of a *data upset*;
- $p_{overflow}$: probability a packet is dropped because of *buffer overflow*;
- σ_{synchr}: standard deviation error of the duration of a round (T_R), which indicates the level of *synchronization errors*.

4. Experimental Results

Stochastic communication can have a wide applicability, ranging from parallel SAT solvers and multimedia applications to periodic data acquisition from non-critical sensors. In this section, we will demonstrate how stochastic communication can be used for the implementation of a complex multimedia application (an MP3 encoder). For this purpose, we have developed a simulator using the Parallel Virtual Machine (PVM)[2] and implemented our algorithm, as described in Section 3. In our simulations, we assume the failure model described in Section 3.6. As realistic data about failure patterns in regular SoCs are currently unavailable, we *exhaustively* explore here the parameter space of our failure model.

Another important parameter we vary is p, the probability that a packet is forwarded over a link (see Section 3.1). For example, if we set the forwarding probability to 1, then we have a completely deterministic algorithm which floods the network with messages (every tile always sends messages to all its four neighbors). This flooding algorithm is optimal with respect to latency, since the number of intermediate hops between source and destination is always equal to the Manhattan distance, but extremely inefficient with respect to the bandwidth used and the energy consumed. In contrast, stochastic communication allows us to trade off energy and performance by varying the probability of transmission p.

4.1. The Target Application: MP3 Encoder

MP3 encoders and decoders are ubiquitous in modern embedded systems, from PDAs and cell phones to digital sound stations, because of the excellent compression rates and their streaming capabilities. We have developed a parallel version of the LAME MP3 encoder [1] (shown in Fig. 11) and introduced the code in our stochastically communicating simulator. The following results are based on this experimental setup.

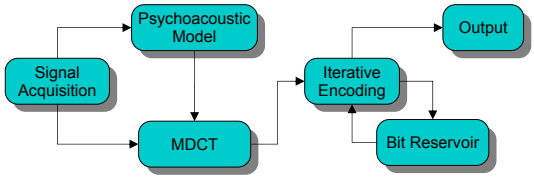

Fig. 11. The modules of the MP3 encoder

4.1.1. Latency. The latency is measured as the number of rounds MP3 needs to finish encoding and it is influenced by several parameters: the value of p (probability of forwarding a message over a link), the levels of data upsets in the network, the buffer overflows and the synchronization errors. Fig. 8 shows how latency varies with the probability of retransmission p and with the probability of upsets p_{upset}. As expected, the lowest latency is when $p = 1$ (the messages are always forwarded over a link) and $p_{upset} = 0$ (when there are no upsets in the network), and increases when $p \to 0$ and $p_{upset} \to 1$ to the point where the encoding of the MP3 cannot finish because the packets fail to reach their destination too often. Note that p_{upset} is dependent on the technology used, while p is a parameter that can be used directly by the designer in order to tune the behavior of the algorithm.

4.1.2. Energy Dissipation. We have estimated the energy dissipation of our algorithms, using Eq. 3. We can see from Fig. 9 that the energy dissipation increases almost linearly with the probability of re-sending p. This is because the energy is proportional to the total number of packets transmitted in the network

[2] PVM is a programming platform which simulates a message passing multi processor architecture.

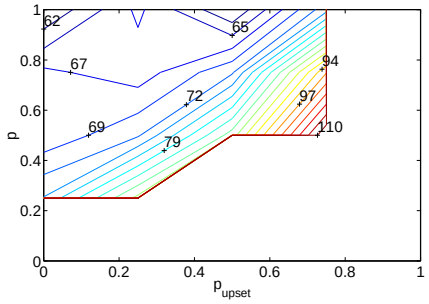

Fig. 8. Latency

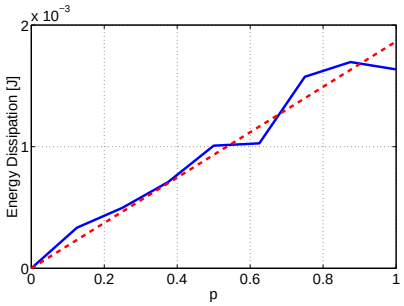

Fig. 9. Energy Dissipation

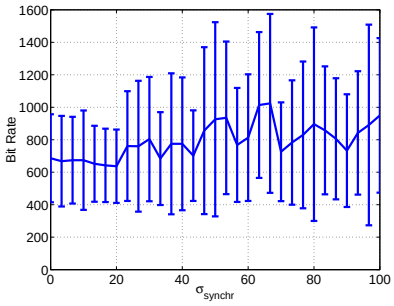

Fig. 10. Impact of synchronization errors

and this number is controlled by p. As we can see from Fig. 8, high values of p mean low latency, but they also mean high energy dissipation. This shows the importance of this parameter, which can be used to tune the tradeoff between performance and energy dissipation, making the stochastic communication flexible enough to fit the needs of a wide range of applications.

4.1.3. Fault-Tolerance. Different types of failures have different effects on the performance of stochastic communication. The levels of buffer overflow do not seem to have a big impact on latency (see left part of Fig. 12). However the encoding will not be able to complete if these levels are too high ($>$ 80%, as in point A in Fig. 12), because one or several packets are lost and none of the tiles has a copies of them. On the other hand, data upsets seem to have little influence on the chances to finish encoding. However, upsets do have an impact on the latency, especially if $p_{upset} > 0.7$ (as seen in Fig. 8).

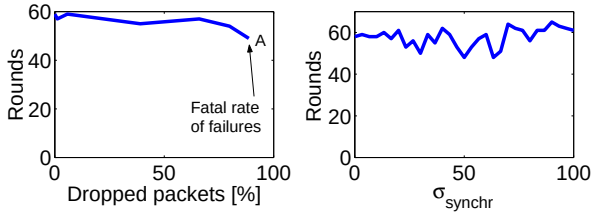

Fig. 12. Rounds needed to finish encoding

The influence of synchronization errors on the output bit rate and latency is shown in Fig. 10 (together with the jitter) and the right part of Fig. 12. Note that even very important synchronization error levels do not have a big impact on the bit rate, the jitter or the latency. This proves that our method tolerates extremely well very high levels of synchronization errors.

These results show that stochastic communication has a very good behavior in time with respect to latency and energy dissipation. In the worst-case (very unlikely for typical levels of failures in NoCs), the protocol will not deliver the packet to the destination; therefore this communication paradigm is better suited for applications which tolerate small levels of information loss, such as our MP3 encoder. In general, real-time streaming multimedia applications have requirements that match very well the behavior of our new communication paradigm, as they can deal with small information losses as long as the bit rate is steady. The results show that our MP3 encoder tolerates very high levels of failures with a graceful quality degradation. If, however, the application requires strong reliability guarantees, these can be implemented by a higher level protocol built on top of the stochastic communication.

5. Conclusion

In this paper, we emphasized the need for on-chip fault-tolerance and proposed a new communication paradigm based on stochastic communication. This approach takes advantage of the large bandwidth that is available on chip to provide the needed system-level tolerance to synchronization errors, data upsets and buffer overflows. At the same time, this method offers a low-latency, low-cost and easy to customize solution for on-chip communication. We believe that our results suggest the big potential of this approach and they strongly indicate that further research in this area would lead to vital improvements of the SoC design.

References

[1] The LAME project. http://www.mp3dev.org/mp3/.

[2] A. Leon-Garcia, I. Widjaja. *Communication Networks*. McGraw, 2000.

[3] N. Bailey. *The Mathematical Theory of Infectious Diseases*. Charles Griffin and Company, London, 2nd edition, 1975.

[4] K. Birman et. al. Bimodal Multicast. *ACM Transactions on Computer Systems*, 17(2):41–88, 1999.

[5] D. M. Chapiro. *Globally Asynchronous Locally Synchronous Systems*. Ph.D. thesis, Stanford University, 1984.

[6] W. Dally and B. Towles. Route packets, not wires: On-chip interconnection networks. In *Proc. of 38th DAC*, June 2001.

[7] A. Demers et. al. Epidemic algorithms for replicated database maintenance. In *Proc. ACM Symp. on Princ. of Distrib. Comp.*, 1987.

[8] D. Estrin et. al. Next century challenges: Scalable coordination in sensor networks. In *Proc. of the MOBICOM*, August 1999.

[9] V. Hadzilacos and S. Toueg. A modular approach to fault-tolerant broadcasts and related problems. Technical Report TR94-1425, 1994.

[10] R. Karp et. al. Randomized rumor spreading. In *Proc. IEEE Symp. on Foundations of Comp. Sci.*, 2000.

[11] F. T. Leighton et. al. On the fault tolerance of some popular bounded-degree networks. In *IEEE Symp. on Foundations of Comp. Sci.*, 1992.

[12] V. Maly. IC design in high-cost nanometer technologies era. In *Proc. of 38th DAC*, June 2001.

[13] L. M. Ni and P. K. McKinley. A survey of wormhole routing techniques in direct networks. *IEEE Computer*, 1993.

[14] B. Pittel. On spreading a rumor. *SIAM Journal of Appl. Math.*, 1987.

[15] Semiconductor Association. The International Technology Roadmap for Semiconductors (ITRS), 2001.