Hindawi Publishing Corporation
VLSI Design
Volume 2007, Article ID 95348, 17 pages
doi:10.1155/2007/95348

Research Article

Stochastic Communication: A New Paradigm for Fault-Tolerant Networks-on-Chip

Paul Bogdan, Tudor Dumitraş, and Radu Marculescu

Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15213-3890, USA

Received 12 December 2006; Accepted 6 February 2007

Recommended by Maurizio Palesi

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. Relaxing the requirement of 100% correctness for devices and interconnects drastically reduces the costs of design but, at the same time, requires SoCs to be designed with some degree of system-level fault-tolerance. Towards this end, this paper introduces a novel communication paradigm for SoCs, called *stochastic communication*. This scheme separates communication from computation by allowing the on-chip interconnect to be designed as a reusable IP and also provides a built-in tolerance to DSM failures, without a significant performance penalty. By using this communication scheme, a large percentage of data upsets, packet losses due to buffers overflow, and severe levels of synchronization failures can be tolerated, while providing high levels of performance.

1. INTRODUCTION AND NOVEL CONTRIBUTION

Nowadays, the application-specific integrated circuits (ASICs) have evolved into complicated systems-on-chip (SoCs), where dozens, and soon hundreds, of predesigned IP cores are assembled together to form large chips with complex functionality. Extensive research on how to integrate and connect these IPs is currently being conducted, but there are many open issues that are difficult to address within the framework of existing CAD tools and design methodologies.

Indeed, shrinking transistor dimensions, smaller interconnect features, and higher operating frequencies lead to a higher sensitivity of deep-submicron (DSM) circuits to neutron and alpha radiation, significantly higher soft-error rates, and an increasing number of timing violations [1]. These new types of failures are impossible to characterize using deterministic measurements and, thus, *probabilistic metrics*, such as average values and variances, are likely to be needed to quantify the critical design objectives, such as performance and power. It has become clear that, in order to reduce the cost of design and verification, the "100% correctness" requirement for VLSI circuits has to be relaxed [2, 3]. This means that, in the near future, circuits will be designed with some degree of architectural and system-level fault-tolerance [4–6].

Furthermore, as emphasized in the ITRS 2001 [3] it is very important, especially at the system-level, to separate the computation from communication, as they are orthogonal issues which should remain separate. A novel communication paradigm has to be developed in order to enable the separation between the design of the on-chip communication architecture and the design of the SoC main functionality.

Traditionally, the IP cores on a typical SoC are connected with a shared bus or a hierarchy of buses. When a bus needs to connect a large number of modules, its performance decreases drastically because of the contention for access to the shared medium. Therefore, this communication architecture is not well suited to future SoCs which may incorporate hundreds of communicating IPs. On-chip buses need to be supplemented or even replaced with a more scalable on-chip communication infrastructure [7].

A recently proposed design platform for the on-chip interconnect is the network-on-chip (NoC) architecture [8, 9], where the IPs are placed on a grid (see Figure 1) and the communication between tiles is implemented by a stack of networking protocols. Such regular structures are very attractive because they can offer well-controlled electrical parameters, which enable high-performance circuits by reducing latency and increasing bandwidth. From this perspective, the SoC design will resemble more the creation of large-scale

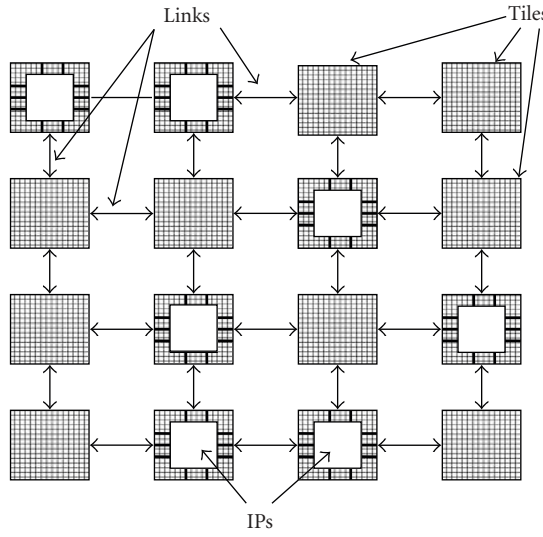


Figure 1: Network-on-chip (NoC).

communication networks rather than traditional IC design practice.

However, providing communication via NoCs is not an easy matter, as mitigating the effects of on-chip failures on the network communication remains largely an open question. Furthermore, the resources used in traditional networks in order to achieve fault-tolerance are not easily available in VLSI chips. For instance, the static routing approach transmitting messages along a fixed path would fail if even a single tile or link on the path becomes faulty. Generally speaking, the deterministic algorithms do not behave well in the presence of random failures [10, 11]. On the other hand, implementing adaptive dynamic routing for on-chip networks is prohibitive because of the need for very large buffers, lookup tables, and complex shortest-path algorithms [12].

Perhaps the greatest challenge introduced by the advent of new technologies is the shift from design determinism to *design uncertainty* [13, 14]. Failures that occur in the DSM technologies can only be characterized by stochastic models, as they are either nondeterministic in nature or too complex to be described by simplistic models. Therefore, the on-chip communication has to be aware of this inherent nondeterminism induced by the DSM technologies.

1.1. Contributions of this paper

The research presented in this paper addresses the issue of on-chip fault-tolerant communication. Towards this end, we introduce a novel communication paradigm, called *on-chip stochastic communication*. In an NoC such as the one in Figure 1, the IPs can communicate using a probabilistic broadcast scheme, very similar to the *randomized gossip* protocols used in databases or sensor networks [15, 16]. More precisely, if a tile has a message that needs to be transmitted, this will be forwarded to a randomly chosen subset of the tiles in the neighborhood. Hence, the messages are *diffused* through the network. At the same time, every IP selects from the set of received messages, only the messages whose destination field is identical to the ID of the tile. The behavior of this communication scheme is similar, but *not* identical, to the proliferation of an epidemic in a large population[1] [17].

This approach achieves many desired features of future SoCs. As shown later in the paper, the algorithm provides the following.

(i) *Separation between computation and communication*, as the communication scheme is implemented in the network logic and is transparent to the IPs.
(ii) *Fault-tolerance* since a message can still reach its destination despite severe levels of DSM failures.
(iii) *Extremely low latency* since this communication scheme does not require retransmission of corrupted data.
(iv) *Low production costs* because the fault-tolerant nature of the algorithm eliminates the need for detailed testing/verification.
(v) *Design flexibility* since it provides a mechanism to tune the tradeoff between performance and energy consumption.

We also note that the proposed on-chip stochastic communication is accompanied with a theoretical justification. The analytical model allows us to explain the essence of stochastic communication and determine nodes coverage which is later validated via simulation.

1.2. Structure of this paper

The remainder of this paper is organized as follows: Section 2 reviews related work. Section 3 introduces a fault model for NoCs which captures the typical errors that may appear in a DSM circuit. Section 4 describes the on-chip stochastic communication algorithm meant to work in a failure-prone environment. Section 5 presents the analytical treatment of the novel communication paradigm. Section 6 presents the experimental results obtained for a simple two-dimensional FFT application working on a flat NoC. Finally, in Section 7 we outline the limitations of the proposed model. We then conclude in Section 8 by summarizing our main contribution.

2. RELATED WORK

The starting point of on-chip stochastic communication is the epidemics theory and gossip algorithms. The epidemics theory has its origin in the Bernoulli's work who tried to prove that the exposure of healthy people to smallpox may contribute to their immunization and decrease the mortality rate [18]. Generally speaking, the *simple* epidemic models characterize the infection process in a finite population of susceptible (S) and infected (I) individuals. The velocity of infection process is given by the product between the num-

[1] This analogy explains intuitively the high performance and robustness of this protocol in the presence of failures.

ber of susceptibles and infected individuals [17, 18]. Besides these two types of individuals, the *general* epidemics model, also called *SIR* model, introduces the removal of individuals (R) which designates an infected person that is removed either by immunization or by death.

In contrast to epidemics, the rumor spreading theory takes into account an additional type of interaction that may happen between a spreader (similar to an infected individual) and a stifler (viewed as a removal). While in epidemics the stifler corresponds to an isolated individual, in rumor spreading it influences directly the dissemination process.

The first complete stochastic model for rumor spreading was introduced by Daley and Kendall in [19]. They divided the entire population of individuals in three categories: spreaders (S), ignorants (I), and stiflers (R). The *spreader* designates an individual (or node) who disseminates the rumor or forwards the message according to a fixed probabilistic rule. An *ignorant* represents an individual who is not aware of the rumor contents yet. A *stifler* refers to an individual who is aware of the rumor, but decides to stop its dissemination.

According to the epidemics modeling, the interaction between a spreader and an ignorant leads to a state with two spreader individuals at a rate proportional to the number of the spreader and ignorant populations (i.e., SI). Similarly, the interaction between a spreader and a stifler may lead to a new stifler at a rate proportional to the number of the spreader and stifler subpopulations (i.e., SR). Moreover, in the case of an interaction between two spreaders, it may happen that one of them becomes a stifler at a rate proportional with the frequency of interactions between individuals belonging to the same subpopulation (i.e., $0.5S(S-1)$).

An alternative approach, called the Maki-Thompson rumor model, is presented in [18]. The main difference between the Daley-Kendall and Maki-Thompson models is that in the later, the interaction between two spreaders results in a new stifler at a rate double than the one in the first approach (i.e., $S(S-1)$). A unified treatment of the rumor spreading process was recently proposed by Pearce [20].

All these approaches inspired the so-called *gossip protocols* for information dissemination in computer systems or sensor networks [15, 16, 21]. They are very attractive for applications that require *localized* communication. A distant ancestor is the USENET news protocol, NNTP [22], developed in early 80s. The news servers running the NNTP protocol exchange updates with the neighboring servers without knowing the entire set of hosts that run NNTP worldwide. Thus, a new message that has been sent to a newsgroup will propagate from server to server until it is known by all of them. An interesting property of this protocol is that the servers are *not* required to know about all other servers, and yet they are able to broadcast the updates to the entire group. This reduces drastically the bandwidth required for the broadcast, which makes this protocol more scalable than most traditional distributed algorithms.

Demers et al. in [15] propose randomized gossip protocols for the lazy update of data objects in databases replicated at many sites. In that paper, the authors show how the gossip-based communication is related to the propagation of epidemics and develop a family of gossip-based multicast protocols. It can be shown that, the updates spread *exponentially* fast among the replicated instances of the database, and the broadcast is accomplished with only a few retransmission rounds.

Several networking protocols, such as the Internet Muse protocol [23], the Scalable Reliable Multicast [24], and the Xpress Transfer Protocol [25] were based on the same principles. Birman et al. in [21] have shown that for the gossip-based multicast protocols there is a high probability that *almost all or almost none* of the players will receive the broadcast, as opposed to the stronger *all or none* guarantee of the classical distributed algorithms. Therefore, these protocols are best suited for applications that can tolerate a small percentage of message losses, but need to be scalable and have a steady throughput.

More recently, these types of algorithms have been applied to the networks of sensors [16]. Their ability to limit the communication to local regions and support light-weight protocols is appealing to applications where power, complexity, and size constraints are critical. Although promising from the practical standpoint, the claims about the practicality of the proposed algorithm are not supported by an analytical framework.

We argue that this communication paradigm can be successfully applied to the SoC design as well, especially for the NoC type of architecture as in Figure 1. Preliminary results investigating the basic paradigm and hardware implementation were presented in [5, 26]. Recently, Manolache et al. in [27] proposed a method to reduce the number of broadcast messages and improve the application response time. In terms of the analytical modeling, while the on-chip stochastic communication resembles epidemics and rumor spreading, it is highly dependent on the topology on which the message diffusion takes place [28]. These topological considerations dictate the calculations of the transition probabilities in our analytical model and help explain the nature of on-chip stochastic communication and interactions that take place. Finally, the analytical model validation is done by evaluating nodes coverage.

3. FAULT MODELING FOR NoCs

Moore's law has been valid for the past three decades. Presently, the advances in wiring and manufacturing technology, as well as the device scaling below the 100 nm threshold, seem to allow Moore's law to continue only for a few more years. Indeed, shrinking transistor dimensions, smaller interconnect features, and higher operating frequencies lead to a higher sensitivity of DSM circuits to neutron and alpha radiation, significantly higher soft error rates, and an increasing number of timing violations [1]. Dependability modeling shows that complex VLSI circuits can be seriously impacted by transient faults and silent data corruption [29, 30]. Furthermore, in order to reduce the costs of design, manufacturing, and verification and make such technologies affordable not only for the highest volume products, the "100% correctness" requirement for VLSI circuits has to be relaxed.

The faults that may appear in NoCs are either transient or permanent. The *transient faults*, also known as *data upsets* or soft errors, are caused by fluxes of neutron and alpha particles, power supply and interconnect noise, electromagnetic interference, or electrostatic discharge. They represent the most common problem for future VLSI circuits. Simply stated, if noise in the interconnect causes a message to be scrambled, a data upset will occur; these faults are subsequently characterized by a probability P_{upset}. As predicted in [1], the rate of occurrence increases as technology scales down into the DSM domain.

Permanent faults reflect irreversible physical changes in the structure of the circuit. They make recovery very hard or even impossible. However, while these errors occur infrequently [31] and do not pose a serious threat to the mass production of VLSI chips, this may change for future nanotechnologies [32].

Another common failure is when a message is lost because of buffer overflow. These faults are known as send/receive omissions [33] or *buffer overflows*. The occurrence rate of these faults is modeled by the probability P_{overflow}.

For complex SoCs, there are additional, more subtle, error modes that can appear. The high coupling capacities of the interconnect and tighter integration favor the Miller effect. As such, it becomes very difficult to achieve predictable delays. Furthermore, as modern circuits span multiple clock domains (e.g., GALS architectures [34]), and can function at different voltages and frequencies (as for "voltage/frequency island"-based architectures [35]), the communication between domains has to go through a special interface with mixed clocks [36]. Due to the special handshake needed before transferring data, the latency in communication may increase and *synchronization errors* appear. In our experiments, every tile has its own clock domain. In this architecture, synchronization errors are normally distributed with a standard deviation σ_{synchr}.

In summary, our fault model depends on the following parameters:

(i) P_{tiles} and P_{links}: probability that a tile/link is affected by a *permanent failure*,
(ii) P_{upset}: probability that a packet is scrambled because of a *data upset*,
(iii) P_{overflow}: probability that a packet is dropped because of *buffer overflows*,
(iv) σ_{synchr}: standard deviation error of the duration of a round (T_R) which indicates the magnitude of *synchronization errors*.

4. THE IDEA OF ON-CHIP STOCHASTIC COMMUNICATION

Traditionally, data networks dealt with fault-tolerance by using complex algorithms, like the Internet Protocol or the ATM Layer [23]. However, these algorithms require many resources that are not available on chip. In addition, they are not always able to guarantee a low latency which is vital for SoCs. For example, in these protocols, the packets are protected by a *cyclic redundancy code* (*CRC*) which is able to detect if a packet contains correct or corrupted data. If an error is detected, the receiver asks for the retransmission of the scrambled messages. This method is known as the *automatic retransmission request (ARQ)*; it has the disadvantage that it increases the communication latency. Another approach is the *forward error correction* (*FEC*), where the errors are corrected directly by the receiver by using an error correction scheme, like the Reed-Solomon code. *FEC* is appropriate when a return channel is not available, as in deep-space communications or in audio CD recordings. *FEC*, however, is less reliable than *ARQ* and incurs significant additional processing complexity.

Based on these considerations, we propose a fast and computationally lightweight scheme for the on-chip communication, based on an error-detection/multiple-transmissions scheme. The key observation behind our strategy is that, at chip level, the bandwidth is less expensive than in traditional networks; this is due to the existing high-speed buses and interconnect fabrics which can be used to implement NoCs. In the following subsections, we illustrate the idea of stochastic communication and then describe the mathematical model of on-chip stochastic communication which is built on this very principle.

4.1. Example of a Producer-Consumer application

In Figure 2, we give the example of a generic Producer-Consumer application. 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 (e.g., tiles 2 and 7 in Figure 2(a)). At the second gossip round, tiles 6, 2, and 7 forward it in the same manner. After this round, eight tiles (i.e., 6, 2, 7, 1, 3, 8, 10, and 11 in Figure 2(b)) 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 (see Figure 2(c)). Note that

(i) the Producer needs *not* know the location of the Consumer and the message will arrive at the destination *with high probability (w.h.p.)*;[2]
(ii) the message reaches the Consumer *before* the full broadcast is completed. For instance, by the time the Consumer gets the message, tiles 13–16 have not received the message yet.

The most appealing feature of this algorithm is its excellent performance in 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 of the same packet anyway. The presence of such upsets is detected by implementing an error-detection scheme on each tile. Bandwidth is less expensive compared to traditional networks, so we can afford more packet transmissions to simplify the communication scheme and guarantee low latencies.

[2] This means with probability at least $1 - O(n^{-\alpha})$ for any constant $\alpha > 0$.

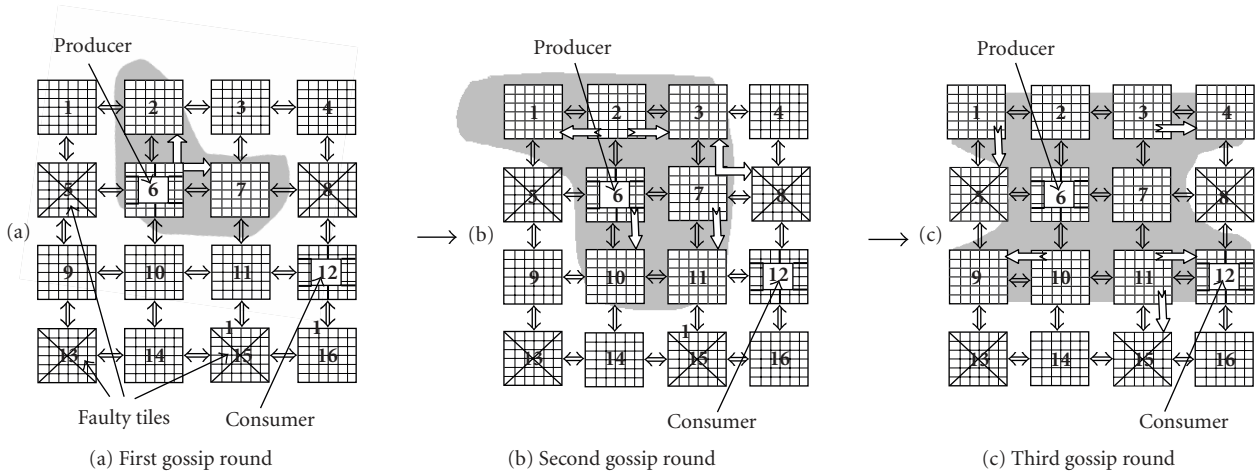


Figure 2: Producer-Consumer application in a stochastically communicating NoC.

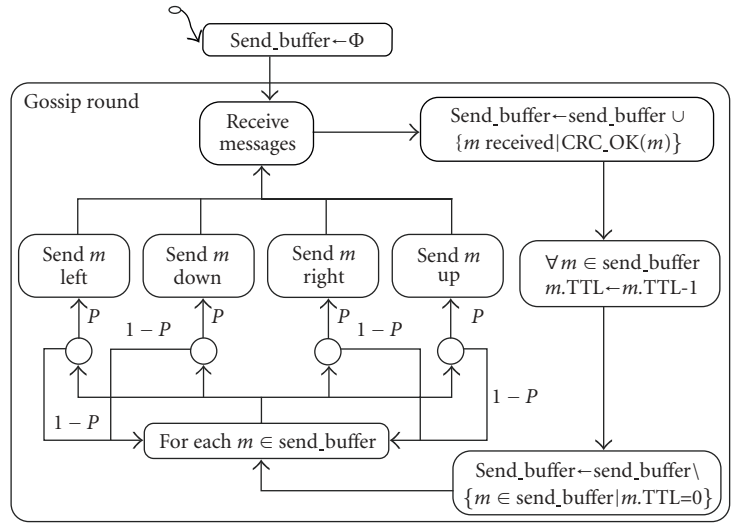


Figure 3: The basic algorithm for stochastic communication.

4.2. A generic algorithm for stochastic communication

The above example assumes that individual tiles can detect if data transmissions are affected by upsets. This is possible by protecting the packets with a CRC code. If an error is discovered, then the packet is 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 even under severe levels of failures.

Our proposed algorithm is presented in Figure 3 (with standard set theory notations) and it is executed concurrently by all nodes in the network. A tile forwards the packet that is available for sending to all four output ports, and then a decision with probability P is made whether or not to transmit the message to the next tile. In practice, a gossip round takes several clock cycles. We also note that, since a message can reach its destination before the broadcast is completed, the spreading can be terminated even earlier in order to reduce the number of messages transmitted in the network. This is important because it is directly related to the bandwidth used and energy dissipated (see Section 5.2). To this effect, we assign a *time-to-live* (*TTL*) to every message upon creation and decrement it at every hop until it reaches value 0 and so the message is garbage-collected. In Section 6, we show how the probability P and the *TTL* can be used to tune the trade-off between performance and energy consumption.

4.3. The hardware interface

A typical tile of such an NoC is shown in Figure 4. The IP core is placed in the center of the tile. On the four edges of the tile, there exist buffers to hold the messages that are sent and received by the IP. A *CRC* decoding circuit checks all the

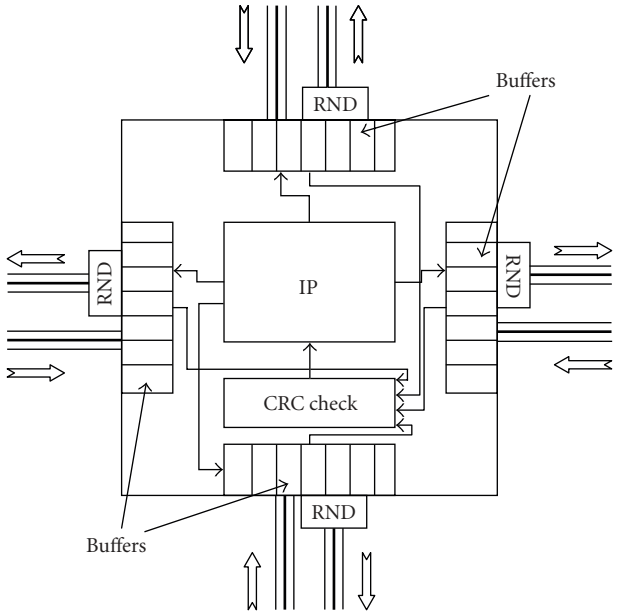


Figure 4: A typical tile of a stochastically communicating NoC.

received packets and when an error is discovered, the message is discarded before being fed into the IP. The tile keeps a list of messages that have to be sent in an output buffer. The messages received during the last round and the new messages generated by the IP core are constantly added to the list. However, if a message is already present, a duplicate message will *not* be inserted. Consequently, even if the message is received a second time from one of the tiles in the neighborhood, only *one copy* is kept in the send buffer and sent to the neighbors during the next round. However, before being actually transmitted over a link, some messages are randomly dropped by a specialized circuit. The selected threshold voltage determines the probability P that a message is forwarded over a link. This is how the aforementioned random subset of neighbors is actually selected.

4.4. Relationship between stochastic communication and rumor spreading

With this technique, the NoC communication becomes similar to the dissemination of a rumor within a large group of friends [37]. Assume that, initially, only one person in the group knows the rumor (see Figure 5(a)). Upon learning the rumor, this person (initiator) passes it to a group of friends (considered confidants if they spread the rumor) randomly chosen. At the next round, both the initiator and the confidants, if there is at least one, independently of each other, select someone else to pass the rumor to. The process continues in the same fashion until everyone is informed or there are no spreaders of the rumor in the entire population.

By analogy with rumor spreading, the NoC tiles are the "gossiping friends," while packets transmitted between them become the "rumors" (see Figure 5(c)). Since in the social network any "friend" is able to communicate with anyone else in the same group (see Figure 5(b)), communication in this setup may lead to the small world phenomenon [38]. From a silicon implementation perspective, however, the topology in Figure 5(b) does *not* represent a viable solution due to its huge wiring demands. Therefore, for NoCs, we consider a grid-based topology (Figure 5(c)), as this is easier and cheaper to implement in silicon.

To analyze the square grid in Figure 5(c)), we clearly need a new method which takes into account the network topology. This is achieved by considering a stochastic model where transition probabilities are dictated by the network topology.[3] This aspect distinguishes the small world approach from the NoC stochastic communication as it will be detailed next. Besides topology, we need to also consider buffer overflows, links and/or nodes failures. Although deriving a good theoretical model for rumor spreading across a grid is an open research question, to the best of our knowledge, our effort represents the first approach that describes both qualitatively and quantitatively the on-chip stochastic communication.

5. ON-CHIP STOCHASTIC COMMUNICATION THEORY

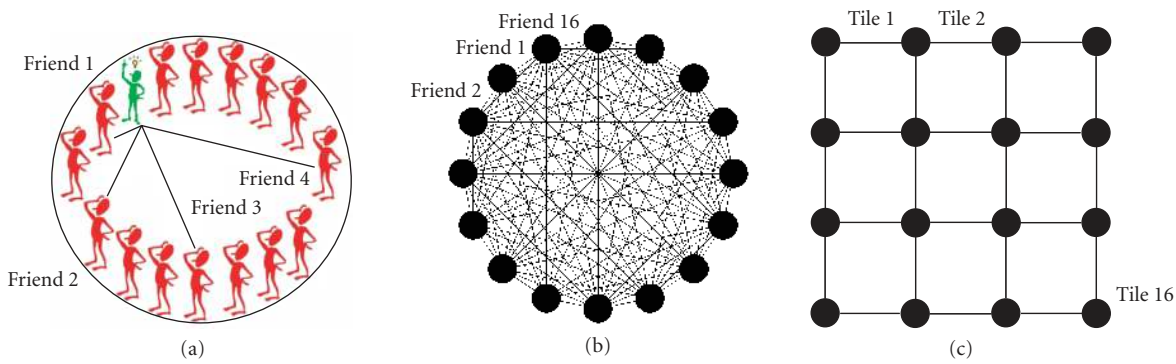
5.1. Basic node interactions and problem formulation

Starting from theory of epidemic and rumor dissemination modelling [17, 19, 37], we consider a grid network of N^2 nodes able to communicate as in Figure 1. Following the model of epidemics theory, we classify the entire population into spreader, ignorant, and stifler nodes. The *spreader* represents any node that is aware of the message and decides to disseminate it. We denote by $S(t)$ the number of spreader nodes at time t. The *ignorant* defines a node that is not included in the communication area yet and it is denoted by $I(t)$. The *stifler* node refers to the case in which a node aware of the message chooses to cease its dissemination (either the link is faulty or the node is dropping the packet). The initial number of spreader and ignorant nodes is denoted by $S(0) = N_1$ and, respectively, $I(0) = N_2$. The number of stifler nodes is obtained by subtracting the spreader and ignorant populations from the total number of nodes: $R(0) = N^2 - N_1 - N_2$. Since we are dealing with a closed population of N^2 nodes, we can write the following conservation law, for time $t \geq 0$:

$$S(t) + I(t) + R(t) = N^2. \tag{1}$$

The process $\{(S, I, R)(t) : t > 0\}$ evolves over a finite state space so that it is completely described by a continuous time Markov chain, for which the initial state and the infinitesimal transition probabilities are specified. While the number of stiflers is simply $R(t) = N^2 - S(t) - I(t)$, it is sufficient to deal with stochastic processes $S(t)$ and $I(t)$ in order to ensure the system evolution. Since we deal with three types of

[3] As we can see in Figure 5(c), each spreader node may communicate directly with at most four neighbors.



(a) Social network (b) Fully connected network (c) Square grid network

Figure 5: Different topologies illustrating social and technological networks.

populations, using the decomposition of a pathway into elementary reactions or interactions [39], we can distinguish the following types of interactions.

(a) Spreader-ignorant interaction

In a mesh topology, each spreader node can be surrounded by one, two, three, or four ignorant nodes. This can contribute to an increase with one, two, three, or four new spreaders. Using the law of mass action as in [17], we can write the associated transition probability in the infinitesimal time interval $(t, t+h)$ as follows:

$$\begin{aligned} P\{S(t+h) = s+k,\ I(t+h) = i-k \mid S(t) = s,\ I(t) = i\} \\ = \alpha_k(t)sih + O(h), \quad k = 1,2,3,4, \end{aligned} \tag{2}$$

where $\alpha_1(t)$, $\alpha_2(t)$, $\alpha_3(t)$, $\alpha_4(t)$ are *time varying* rates that characterize the interaction between any spreader and one, two, three, and four ignorant neighbors. More precisely, starting with $s-1$ spreaders and $i+1$ ignorants (i.e., initial state $(s-1, i+1)$), we end up with s spreaders and i ignorants (i.e., final state (s,i)) at a rate proportional with $\alpha_1(t)(s-1)(i+1)$ (i.e., $k=1$).

We note that the α_k parameters in (2) are directly related to the forwarding probability P in the algorithm in Figure 3. More precisely, $\alpha_k = \binom{4}{k}P^k(1-P)^{4-k}$. This illustrates the direct connection between the formalism developed in this section and the "knobs" that control the stochastic communication in Figure 3.

(b) Spreader-spreader interaction

Another type of reaction is the interaction between two spreaders that are neighboring nodes. In this situation, it can happen that either both spreader nodes stop the packet diffusion process with a rate $\beta_2(t)$, or one node ceases the dissemination while the second node continues to forward packets in the network with a rate $\beta_1(t)$. The situations can describe the behavior of a link between two congested nodes which is characterized by P_{overflow}. We do not consider the situation when *both* nodes continue the packet dissemination since this can be viewed as a consequence of the spreader-ignorant interaction mentioned above. The corresponding transition probability for these two cases is

$$\begin{aligned} P\{S(t+h) = s-m,\ I(t+h) = i \mid S(t) = s,\ I(t) = i\} \\ = \beta_m(t)sh + O(h), \quad m = 1,2. \end{aligned} \tag{3}$$

(c) Spreader-stifler interaction

The third type of interaction we consider is between a spreader and a stifler. The result of this interaction consists in altering the state of the spreader node with a rate $\alpha_5(t)$. More precisely, a faulty node, also called stifler, may corrupt the received packet and disseminate it, affecting the state of other nodes. The associated transition probability is given by

$$\begin{aligned} P\{S(t+h) = s-1,\ I(t+h) = i \mid S(t) = s,\ I(t) = i\} \\ = \alpha_5(t)s(N^2 - s - i)h + O(h). \end{aligned} \tag{4}$$

(d) No interaction

If there is no change in the continuous time Markov chain, the corresponding transition probability has the form:

$$\begin{aligned} P\{S(t+h) = s,\ I(t+h) = i \mid S(t) = s,\ I(t) = i\} \\ = 1 - \left\{ \sum_{k=1}^{4} \alpha_k(t)i + \sum_{m=1}^{2} \beta_m(t) + \alpha_5(t)(N^2 - s - i) \right\} sh \\ - O(h). \end{aligned} \tag{5}$$

The parameters that characterize the above interactions were introduced for capturing both permanent and transient errors. For $s+i \le N_1+N_2$ and $i \le N_2$ we define

$$P\{S(t+h) = s,\ I(t+h) = i \mid S(t) = N_1,\ I(t) = N_2\} = P(s,i,t) \tag{6}$$

which stands for the probability that, at time t, we have s spreader nodes and i ignorant nodes. If needed, the number of stifler nodes can be easily obtained by subtracting

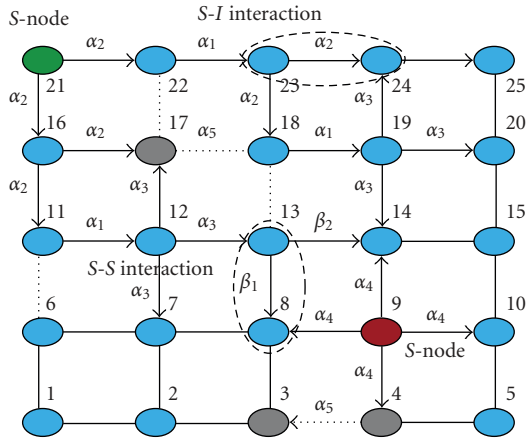


FIGURE 6: Stochastic dissemination in a 5×5 mesh network. The dotted lines show damaged links. The lines with arrows represent packet communication.

from the entire population the number of spreader and ignorant nodes. The packet dissemination process can be then described by the following forward Kolmogorov equation:

$$\begin{aligned}\frac{dP(s,i,t)}{dt} &= \sum_{k=1}^{4}\alpha_k(t)(s-k)(i+k)P(s-k,i+k,t)\\ &+\sum_{k=1}^{2}\beta_k(t)(s+k)P(s+k,i,t)\\ &+\alpha_5(t)(s+1)\left(N^2-s-i-1\right)P(s+1,i,t)\\ &-s\left[\sum_{k=1}^{4}\alpha_k(t)i+\sum_{k=1}^{2}\beta_k+\alpha_5(t)\left(N^2-s-i\right)\right]P(s,i,t).\end{aligned} \tag{7}$$

Next, we give some intuition about the stochastic packet dissemination using a 5×5 mesh network (see Figure 6). We set the nodes 9 and 21 to be the sources for packet dissemination. Source 21 can follow the spreader-ignorant interaction and send packets to its neighbors (i.e., nodes 16 and 22) with rate α_2. Further, nodes 16 and 22 can forward their packets with rates α_2 and α_1, respectively. Node 22 sends packets with rate α_1 (instead of α_2) because the dotted link between nodes 22 and 17 is assumed to be damaged. We observe that node 17 does not send any packet and becomes a stifler with rate α_5. Subsequently, nodes 11, 12, and 13 behave similarly as they describe new spreader-ignorant interactions.

By the same token, the links between the neighboring nodes 6, 11 and 13, 18 are considered damaged. Source 9 can send a packet to its neighbors with probability α_4. Later on, nodes 8, 13, and 14 try to disseminate their packets. If the network is overloaded, we may notice new interactions. For instance, it can happen that only one of the nodes 8 or 13 disseminates its packets; this transition probability (β_1) is shown on the link between them (spreader-spreader interaction). Moreover, if nodes 13 and 14 are blocked due to congestion, then we end up with two nodes blocking the dissemination process (β_2).

Next, we illustrate the derivation of the closed-form solution for the packet diffusion process [28]. First, we use the probability generating functions

$$f_i(x,t)=\sum_{s=0}^{\infty}P(s,i,t)x^s,\quad |x|\le 1,\ 0\le i\le N_2 \tag{8}$$

to rewrite (7) as a transport equation,

$$\begin{aligned}\frac{\partial f_i(x,t)}{\partial t} &= \sum_{k=1}^{4}\alpha_k(t)(i+k)\frac{\partial f_{i+k}(x,t)}{\partial x}+\alpha_5(t)x(x-1)\frac{\partial^2 f_i(x,t)}{\partial x^2}\\ &+\left\{\sum_{k=1}^{2}\beta_k(t)\left[\frac{1}{x^{k-1}}-x\right]+\alpha_5(t)\left(N^2-i-1\right)(1-x)\right.\\ &\left.-x\sum_{k=1}^{4}\alpha_k(t)i\right\}\frac{\partial f_i(x,t)}{\partial x}.\end{aligned} \tag{9}$$

Furthermore, we can construct a vector

$$F(x,t)=\left[f_{N_2}(x,t),f_{N_2-1}(x,t),\ldots,f_1(x,t),f_0(x,t)\right]^T \tag{10}$$

from all the probability generating functions and obtain the following nonlinear differential equation:

$$\frac{\partial F(x,t)}{\partial t}=A(x,t)\frac{\partial F(x,t)}{\partial x}+B(x,t)\frac{\partial^2 F(x,t)}{\partial x^2}, \tag{11}$$

where the functions A and B are given by

$$A(x,t)=\begin{bmatrix} a(N_2) & 0 & \cdots & 0\\ b(N_2-1) & a(N_2-1) & \cdots & 0\\ c(N_2-2) & b(N_2-2) & \cdots & 0\\ d(N_2-3) & c(N_2-3) & \cdots & 0\\ e(N_2-4) & d(N_2-4) & \cdots & 0\\ \cdots & \cdots & \cdots & \cdots\\ 0 & 0 & \cdots & a(0)\end{bmatrix} \tag{12}$$

$$B(x,t)=\alpha_5(t)x(x-1) \tag{13}$$

with the following components:

$$\begin{aligned}a(i) &= \sum_{k=1}^{2}\beta_k(t)\left(\frac{1}{x^{k-1}}-x\right)\\ &+\alpha_5(t)\left[\left(N^2-i\right)(1-x)-1\right]-\sum_{k=1}^{4}xi\alpha_k(t)\end{aligned} \tag{14}$$

$$b(i)=\alpha_1(t)(i+1)x^2 \qquad c(i)=\alpha_2(t)(i+2)x^3$$

$$d(i)=\alpha_3(t)(i+3)x^4 \qquad e(i)=\alpha_4(t)(i+4)x^5.$$

Generally speaking, it is difficult to find out the solution of the above-mentioned differential equation in matrix form. We briefly summarize the methods for finding the solution in time and frequency domains for the case in which A and B are time-independent. This leads to the following equation:

$$\frac{\partial F(x,t)}{\partial t}=A(x)\frac{\partial F(x,t)}{\partial x}+B(x)\frac{\partial^2 F(x,t)}{\partial x^2}. \tag{15}$$

5.1.1. Time domain investigation

To solve (15) in time domain, we use the method of separating variables. Consequently, the base solution will have the following form $F(x,t) = X(x)T(t)$, where X is a real function and T is a temporal scalar function. Consequently, the parabolic differential equation can be written as follows:

$$LX(x) = A(x)D^2X(x) + B(x)DX(x) = \frac{dT(t)}{T(t)}\frac{X(x)}{T(x)}. \quad (16)$$

Hence, by introducing the eigenvalue λ in the following equation $dT(t)/dt = -\lambda T(t)$, we obtain $T(t) = e^{-\lambda t}$. The general solution will be given by the following formula:

$$F(x,t) = \sum_{\lambda} c_\lambda X_\lambda(x) e^{-\lambda t}, \quad (17)$$

where X_λ is the eigenvector associated with the eigenvalue λ of the Sturm-Liouville differential operator L (i.e., $L = A(x)D^2 + B(x)D$).

5.1.2. Frequency domain investigation

To find out the solution of the forward Kolmogorov equation (7), we use the method described in [18, 37]. Using the Laplace transform for the probability generating function associated with the forward Kolmogorov equation, we reach the following relation for $0 \le i \le N_2$:

$$\begin{aligned}\theta f_i(x,s) - \delta_{N_2} x^{N_2} = {} & \left\{\sum_{k=1}^{2}\frac{\beta_k}{x^{k-1}} + \alpha_5\left(N^2 - i - 1\right)\right\}\frac{\partial f_i(x,s)}{\partial x} \\ & - x\left[\sum_{k=1}^{4} i\alpha_k + \sum_{k=1}^{2}\beta_k\alpha_5\left(N^2 - i\right)\right]\frac{\partial f_i(x,s)}{\partial x} \\ & + \sum_{k=1}^{4}\alpha_k(i+k)x^{k+1}\frac{\partial f_{i+k}(x,s)}{\partial x} \\ & + \alpha_5 x(x-1)\frac{\partial^2 f_i(x,s)}{\partial x^2}.\end{aligned} \quad (18)$$

In order to solve the differential equation, we apply the Laplace transform to the column vector introduced in (10) which leads to:

$$B(x)\frac{\partial^2 F(x,s)}{\partial x^2} + A(x)\frac{\partial F(x,s)}{\partial x} - sF(x,s) = -x^{N_2}E_{N_2+1}, \quad (19)$$

where $A(x)$ is given by (12), and $B(x) = \alpha_5 x(x-1)I_{N_2+1}$. The symbol I_{N_2+1} represents the identity matrix of dimension $N_2 + 1$. If we differentiate the above equation with respect to variable x, and use the notation

$$F_d = F_d(x,s) = \frac{\partial^d F(x,s)}{\partial x^d}, \quad (20)$$

then (19) takes the form

$$\begin{aligned}B(x)F_{d+2} + \left[z(d)B^{(1)}(x) + A(x)\right]F_{d+1} & \\ + \left[w(d)B^{(2)}(x) + z(d)A^{(1)}(x) - sI_{N_2+1}\right]F_d & \\ + w(d)A^{(2)}(x)F_d = -\frac{N_2!x^{N_2-d}}{(N_2-d)!}E_{N_2+1} &\end{aligned} \quad (21)$$

with $z(d) = d$ and $w(d) = d(d-1)/2$, for $d \ge 0$. Considering that the probability generating functions $f_i(x,s)$ are polynomial functions of variable x of degree N_1+N_2-i and using the Taylor expansion of $F(x,s)$, we get the following relations:

$$F(x,s) = F_0 + \sum_{d=1}^{N_1+N_2+2}\frac{(x-x_0)^d}{d!}\cdot\frac{\partial^d F(x,s)}{\partial x^d}, \quad (22)$$

where F_0 represents the initial condition, and the intermediate Taylor factors are given by

$$\frac{\partial^d F(x,s)}{\partial x^d} = \left[\prod_{d=0}^{N_1+N_2} A_d \begin{bmatrix} F_1 \\ F_0 \\ 0 \end{bmatrix}\right]_{N_2+1} + \sum_{d=0}^{N_1+N_2}\prod_{k=d+1}^{N_1+N_2} A_k B_d \quad (23)$$

with the following coefficients:

$$\begin{aligned} B_d &= \begin{bmatrix} -\dfrac{N_2!x^{N_2-d}}{(N_2-d)!}E_{N_2+1} \\ 0 \\ 0 \end{bmatrix}, \\ A_d &= \begin{bmatrix} A_d^1 & A_d^2 & A_d^3 \\ I_{N_2+1} & O_{N_2+1} & O_{N_2+1} \\ O_{N_2+1} & I_{N_2+1} & O_{N_2+1} \end{bmatrix}, \\ A_d^1 &= B^{-1}(x)\left[z(d)B^{(1)}(x) + A(x)\right]A_d^3 = w(d)B^{-1}(x)A^{(2)}(x), \\ A_d^2 &= B^{-1}(x)\left[w(d)B^{(2)}(x) + z(d)A^{(1)}(x) - \theta I_{N_2+1}\right]. \end{aligned} \quad (24)$$

The symbols I_{N_2+1} and O_{N_2+1} stand for the identity and zero matrices, respectively, of dimension $N_2 + 1$.

In summary, (22) governs the evolution of packet dissemination on a grid. This equation lies at the heart of on-chip stochastic communication. More precisely, it captures the transition probabilities of the Markov process associated with the SIR interactions, as well as the effects induced by the network topology during the stochastic dissemination process.

5.2. Performance evaluation

The behavior of the stochastic communication scheme can be characterized based on a small number of parameters which are detailed in this section.

5.2.1. Nodes coverage

An interesting metric that can be derived from the proposed analytical framework is the *coverage* metric (i.e., the number of reached nodes as a function of time). To analyze the coverage metric for the mesh network, we get inspiration from systems biology [39]. As such, we construct a set of rate equations that describe the rumor dissemination process on a mesh network. The ordinary differential equation modeling starts from the following set of possible reactions and their

rate coefficients:

$$\begin{aligned}
S+I &\longrightarrow S+S \text{ with rate } \alpha_1 \quad (1,-1,0) \text{ with rate } \alpha_1,\\
S+2I &\longrightarrow S+2S \text{ with rate } \alpha_2 \quad (2,-2,0) \text{ with rate } \alpha_2,\\
S+3I &\longrightarrow S+3S \text{ with rate } \alpha_3 \quad (3,-3,0) \text{ with rate } \alpha_3,\\
S+4I &\longrightarrow S+4S \text{ with rate } \alpha_4 \quad (4,-4,0) \text{ with rate } \alpha_4,\\
S+S &\longrightarrow S+R \text{ with rate } \beta_1 \quad (-1,0,1) \text{ with rate } \beta_1,\\
S+S &\longrightarrow R+R \text{ with rate } \beta_2 \quad (-2,0,2) \text{ with rate } \beta_2,\\
S+R &\longrightarrow R+R \text{ with rate } \alpha_5 \quad (-1,0,1) \text{ with rate } \alpha_5,
\end{aligned} \tag{25}$$

where S denotes the number of spreaders, I designates the number of ignorants, and R represents the number of stiflers. Whenever a reaction takes place, the number of individuals in each population evolves as shown in the right-hand side of (25). For example, if the third reaction takes place with rate α_3, then the expression $(3,-3,0)$ shows that the number of spreader nodes (S) increases by 3 individuals, the number of ignorants (I) decreases by the same amount and the number of stiflers (R) remains unchanged. More precisely, if we start from (S,I,R) configuration and we follow the third reaction, then we end up in the $(S+3, I-3, R)$ state.

The reaction velocity of each equation can be obtained by using the following rationale: given the reaction $mX + nY \rightarrow qZ$ that takes place at rate k, the reaction velocity is $(1/m)(d[X]/dt) = (1/n)(d[Y]/dt) = k[X]^m[Y]^n$. Further, we have the following relations:

$$\begin{aligned}
\frac{[dS/dt]}{1} &= \frac{[dI/dt]}{-1} = \alpha_1[S][I],\\
\frac{[dS/dt]}{1} &= \frac{[dI/dt]}{-2} = \alpha_2[S][I]^2,\\
\frac{[dS/dt]}{1} &= \frac{[dI/dt]}{-3} = \alpha_3[S][I]^3,\\
\frac{[dS/dt]}{1} &= \frac{[dI/dt]}{-4} = \alpha_4[S][I]^4,\\
\frac{[dS/dt]}{-1} &= \frac{[dR/dt]}{1} = \beta_1[S]^2[R],\\
\frac{[dS/dt]}{-2} &= \frac{[dR/dt]}{2} = \beta_2[S]^2[R],\\
\frac{[dS/dt]}{-1} &= \frac{[dR/dt]}{1} = \alpha_5[S][R].
\end{aligned} \tag{26}$$

This is a compact way of defining the velocity of each reaction. For instance, the first relation in (26) states that the small variation in the number of spreaders over a short time interval (i.e., dS/dt) is proportional with $\alpha_1 SI$; this needs to be accounted for since one ignorant becomes a spreader. At the same time, the variation in the number of ignorants over a short time interval (i.e., dI/dt) is proportional to $-\alpha_1 SI$ since the ignorant population losses one member. It is important to note that all the interactions in (26) are *independent* and only *one* such interaction occurs in any given time interval dt. Based on these relations, we can write the following rate equations to describe how the ensemble changes at any point in time:

$$\begin{aligned}
\frac{ds}{dt} &= \sum_{k=1}^{4} k\alpha_k s i^k - \sum_{k=1}^{2} k\beta_k s^2 - \alpha_5 sr,\\
\frac{di}{dt} &= -\sum_{k=1}^{4} k\alpha_k s i^k,\\
\frac{dr}{dt} &= \sum_{k=1}^{2} k\beta_k s^2 + \alpha_5 sr.
\end{aligned} \tag{27}$$

The variations in number of spreaders, ignorants, and stiflers over time are obtained by taking into account the *superposition* of all the above-mentioned individual effects. Thus, if the third reaction in (25) takes place, for instance, then the number of spreaders increases by three units (therefore, $k = 3$ in (27) and the first term becomes $3\alpha_3 SI^3$) since three ignorants are turned into spreaders. At the same time, the same quantity needs to be subtracted from the number of ignorants variation in (27) (i.e., $-3\alpha_3 SI^3$). We note that the semantics of these relations is similar to that used in process algebra of concurrent communicating systems [40].

Considering, for instance, (27) and the following initial configuration on a 10×10 grid: one spreader, one stifler, $N^2 - 2$ ignorants (i.e., 98 ignorants), and setting the parameters $\beta_1 = \beta_2 = \alpha_5 = 0$ (i.e., assuming neither damaged links, nor nodes), we obtain the time evolution in Figure 7, as a function of P. The conclusion from this plot is twofold: on one hand, there exists a region of exponential growth in the number of spreader nodes; this is also later shown experimentally by simulation in Section 6.1.2. On the other hand, in absence of faults or upsets, the number of spreader nodes reaches a maximum point after which it does not change significantly with time.

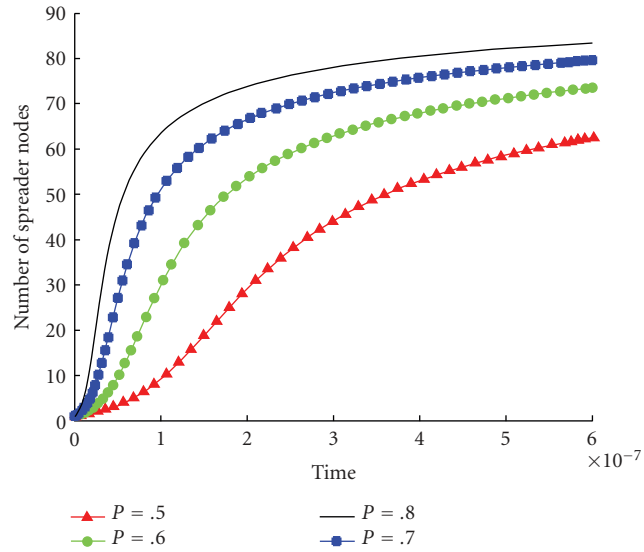


Figure 7: Time evolution of spreader nodes with and without faults.

In order to investigate the impact of faults or upsets on the coverage metric, we increase the values of the fault parameters (i.e., β_1, β_2, α_5) and use the analytical approximation, in (27). We compare these results with the case in which there are neither damaged links, nor nodes. As we can see in Figure 8 ($P = .7$), the number of nodes that become aware of the packet dissemination reaches a maximum point in both cases. However, in presence of faults, the number of reached nodes is significantly smaller. This can be explained with the spreader-spreader and spreader-stifler interactions introduced in Section 5.1 (i.e., (3) and (4)).

5.2.2. Broadcast rounds

A *broadcast round* is the time interval in which a tile has to finish sending all its messages to the next hops; this usually takes several clock cycles. The optimal duration of a round (T_R) can be determined using (28)

$$T_R = \frac{S \cdot N_{\text{packets/round}}}{f}, \tag{28}$$

where f is the maximum frequency of any link, $N_{\text{packets/round}}$ is the average number of packets a link sends during one round (this is application dependent), and S is the average packet size.

As shown in Section 6, the fast dissemination of rumors makes it possible to achieve very low latencies. While XY routing generates traffic along a single path, stochastic communication spreads the traffic uniformly across all links in the network, thereby reducing the chances that packets are delayed due to local congestion. Under the proposed protocol, congestion occurs only when the entire network becomes saturated. This is especially important in multimedia applications, for instance, where a sustainable constant bit-rate is highly desirable. Furthermore, the fact that we do not store or compute the shortest paths (as in dynamic routing) makes this algorithm computationally lightweight, simpler and easier to customize for every application and interconnection network. We also note that since the stochastic communication algorithm uniformly spreads the traffic across the entire network, the proposed algorithm does not suffer from deadlock. Indeed, the packets can reach their destinations even when using finite buffers if the injection rates are sufficiently small.

5.2.3. Energy metrics

To estimate the energy consumption of this algorithm, we use (29)

$$\begin{aligned} E_{\text{total}} &= E_{\text{computation}} + E_{\text{communication}} \\ &= E_{\text{computation}} + N_{\text{packets}} \cdot S \cdot E_{\text{bit}}, \end{aligned} \tag{29}$$

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 per link. The parameter N_{packets} can be estimated by simulation, S is application-dependent, and E_{bit} can be derived from the technology library.

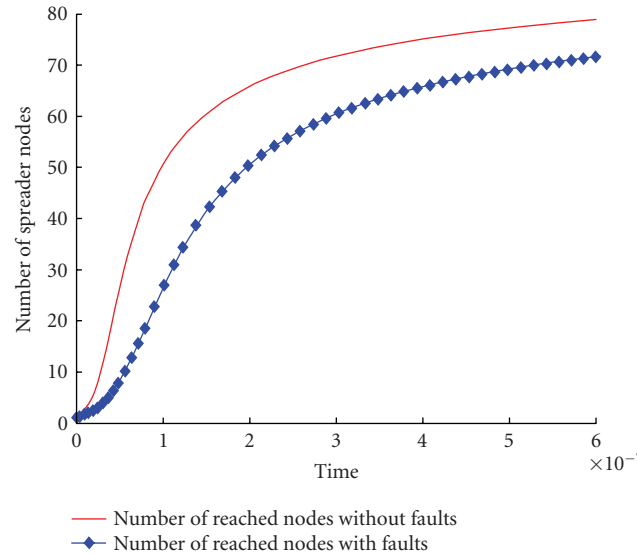


Figure 8: Time evolution of spreader nodes with and without faults.

As shown in (29), the total energy consumed is also influenced by the computational cores ($E_{\text{computation}}$). Since our focus is on the performance of the *communication scheme*, it is not necessary to estimate precisely the energy required by computation. However, this can be added, to our estimations in Section 6 in order to get the combined energy values.

In summary, the parameters relevant to our analysis are:

(i) *nodes coverage*, which is a measure of message dissemination in the network (similarly to molecules diffusion in gases),
(ii) *number of broadcast rounds* needed, which is a direct measure of the inter-IP communication latency,
(iii) *total number of packets* sent in the network, which indicates the bandwidth required by the algorithm; it can be controlled by varying the message TTL value,
(iv) *fault-tolerance*, which evaluates the algorithm resilience to abnormal conditions in the network,
(v) *energy consumption*, which is computed with (29) (as detailed in Section 6).

6. PRACTICAL CONSIDERATIONS AND EXPERIMENTAL RESULTS

Stochastic communication can be beneficial to many application areas ranging from parallel SAT solvers and multimedia applications to periodic data acquisition from noncritical sensors. Starting from a computational task graph, the fault-tolerant scheme distributes the independent tasks and operations to different nodes (i.e., information dissemination towards slave nodes) similar to a network of sensors. Later on, the results are collected at the master node through multiple paths. This idea is better emphasized by our first example, the two-dimensional Fast Fourier Transform (FFT) (see Figure 9). The inherent parallelism of the FFT algorithm is exploited by computing the partial results at the slave nodes.

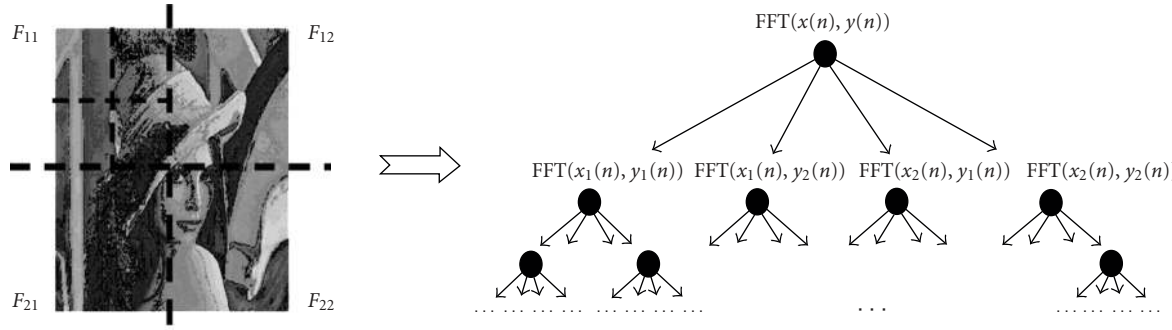


Figure 9: Parallel scheme for computing FFT2.

The stochastic communication scheme ensures that the master node ultimately receives these results and is able to return the desired FFT value even for high data error rates.

Since realistic data about failure patterns in regular SoCs are not easily available, we explore the entire parameter space of our fault model (i.e., $P_{\text{upset}} \in [0, 1]$). Another important parameter we vary is P, the probability that a packet is forwarded over a link (see Section 4.1). Stochastic communication allows us to tune the trade-off between energy and performance by varying the probability of transmission (P) between 0 and 1.

6.1. Case study: the 2D Fast Fourier Transform

Given the structure of the Discrete Fourier Transform of N samples, a recursive divide-and-conquer algorithm can be used to compute the FFT of N samples (see Figure 9). This reduces the number of operations to $O(N \log_2 N)$. Because of its widespread use in engineering (especially in image and multimedia processing), we have focused on the *two-dimensional FFT* algorithm (FFT2) applied to both XY dimensions. Every node in the tree from Figure 9 represents a parallel process. The leaves compute the FFT on a small number of samples and send the intermediate results back to the root which assembles the final result.

We map this application onto a 4×4 NoC running the stochastic communication algorithm in Section 4. To increase the tolerance to permanent tile failures, the IPs can be duplicated (as for the Master-Slave computations). Our models simulate grids of 16–25 tiles, which is relevant to nowadays designs, but the gossip algorithms scale extremely well so stochastic communication can be applied to much larger designs.

For this case study we have considered a restricted version of the fault model from Section 3, which includes only the most cited failure modes in the current literature [33] permanent failures of tiles and links (with probabilities P_{tiles} and P_{links}) and *data upsets* (with probability P_{upset}). The random failures are distributed uniformly in time during the length of the simulation (see Figure 10 where we consider a 4×4 NoC).

In our framework, the *packet* (not the flit) represents the basic unit of information and as such, packet reordering is not a real issue. More precisely, in our experiments, while packets may arrive at destination in an out-of-order fashion, they can be easily reordered based on the information contained in their header. As such, the experiments reported in this paper do not suffer from the out-of-order arrival problem.

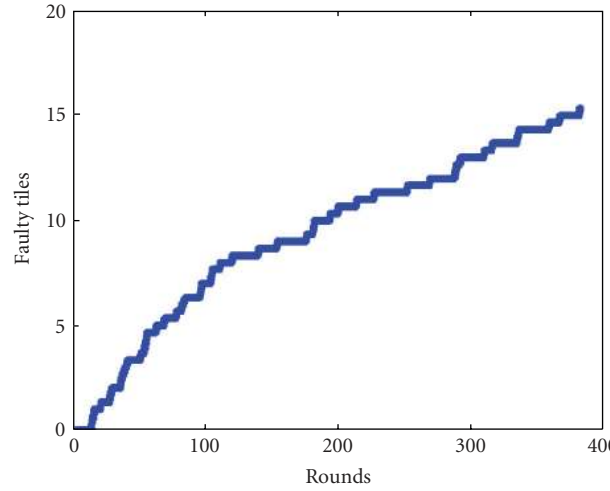


Figure 10: Distribution of failures during simulation.

We evaluate the *latency*, *the energy dissipation*, and the *fault-tolerance* of stochastic communication.For consistency reasons, all the results presented in this section are averages obtained after several simulations. We compare here four versions of the stochastic communication, obtained for different values of the parameter P:

(i) the network *flooding*, which is a deterministic algorithm where the tiles send the messages to *all* their neighbors all the time ($P = 1$);
(ii) three versions of the stochastic communication algorithm in Section 4, which uses different probabilities to transmit the message across the links; namely, we use $P = .75$, $P = .5$, and $P = .25$.

The reason for comparing our protocol against the flooding algorithm is because the latter is the simplest possible example of a deterministic broadcast protocol. The flooding

Table 1: Stochastic communication in an NoC with 6.25% defective tiles and 8.33% defective links.

	Initial broadcast (no. of rounds)				FFT2 finished (no. of rounds)				Total number of packets				Average energy ($\times 10^{-8}$ J/bit)			
	Flood	Stoch.comm.(P)			Flood	Stoch.comm.(P)			Flood	Stoch.comm.(P)			Flood	Stoch.comm.(P)		
	($P = 1$)	.75	.5	.25	($P = 1$)	.75	.5	.25	($P = 1$)	.75	.5	.25	($P = 1$)	.75	.5	.25
$P_{\text{upset}} = 0$	5	7	—	16	5	6	9	16	471	481	619	558	2.261	1.924	1.651	0.836
$P_{\text{upset}} = 0.2$	5	9	12	30	5	7	11	16	444	559	811	508	2.129	1.917	1.768	0.761
$P_{\text{upset}} = 0.4$	9	11	—	—	5	13	16	20	387	1417	1289	486	1.858	2.616	1.934	0.583
$P_{\text{upset}} = 0.6$	19	11	17	38	10	16	18	39	1226	1997	1172	1700	2.942	2.995	1.562	1.046
$P_{\text{upset}} = 0.8$	22	18	46	85	23	25	34	84	4158	2734	2080	3562	4.339	2.624	1.468	1.018

Table 2: Impact of failures on latency of stochastic communication with $P = .5$ (number of rounds to finish: initial broadcast/FFT2).

	$P_{\text{tiles}} = 0$				$P_{\text{tiles}} = .125$				$P_{\text{tiles}} = .25$			
	$P_{\text{links}} =$				$P_{\text{links}} =$				$P_{\text{links}} =$			
	0	.083	.166	.25	0	.083	.166	.25	0	.083	.166	.25
$P_{\text{upset}} = 0$	8/6	9/9	15/8	—/9	8/10	7/7	13/13	—/11	6/8	—/—	—/14	—/—
$P_{\text{upset}} = 0.2$	11/10	10/8	—/12	—/12	9/12	—/10	23/19	—/19	8/—	—/20	—/17	—/—
$P_{\text{upset}} = 0.4$	9/11	16/14	—/14	—/25	15/10	15/16	—/16	—/—	—/11	—/37	—/9	—/22
$P_{\text{upset}} = 0.6$	15/18	21/13	—/22	—/25	43/15	—/36	21/12	—/—	24/16	—/16	—/17	—/—
$P_{\text{upset}} = 0.8$	33/27	46/33	83/37	—/69	24/53	—/49	35/46	—/—	—/—	42/—	—/39	—/35

algorithm is also optimal with respect to latency; that is, the number of intermediate hops between source and destination is always equal to the Manhattan distance between the two tiles. We show that our protocol has a latency close to this optimum. The flooding is, however, extremely inefficient to implement with respect to the bandwidth and energy consumed. The stochastic communication enables various trade-offs between energy and performance figures by varying the values of the transmission probability P.

A summary of our results is given in Tables 1 and 2. For example, the second row in Table 1 shows that, when the probability of an unsuccessful transmission is $P_{\text{upset}} = .2$, depending on the algorithm used, the broadcast of the first message takes between 5–30 rounds to reach all the tiles, FFT2 is computed after 5–16 rounds, there are 444–811 packets transmitted in the network, with an average energy consumption of 7.61–21.29 nJ per message bit.[4] The second row of Table 2 shows the number of rounds needed to finish the initial broadcast/the computation of FFT2, for different levels of tile and link failures, when $P_{\text{upset}} = .2$.

To give an idea about the average communication energy consumption of the stochastic communication algorithm, we compare the energy consumption of the deterministic XY routing, against the energy of the proposed approach when there are no defective tiles or links in the network. We consider a random mapping which is not optimal for either stochastic or deterministic cases. Moreover, we assume that for a nonzero probability of data upsets (i.e., $P_{\text{upset}} > 0$) in the deterministic case, a number of $(P_{\text{upset}} \times N_{\text{packets}})$ packets need to be retransmitted during the next communication round.[5] The packet retransmission is done in n communication rounds, where n is determined from the following condition $(P_{\text{upset}})^{\mathbf{n}} \leq \varepsilon$ and ε depends on the allowed tolerance to missing FFT2 values. In absence of data upsets (i.e., $P_{\text{upset}} = 0$), the deterministic XY routing scheme consumes 7.2 nJ energy which is smaller than the energy consumption needed in flooding (i.e., 17.22 nJ) and stochastic communication (i.e., 18.34 nJ for $P = .75$, 11.94 nJ for $P = .5$ and 7.83 nJ for $P = .25$). In presence of data upsets, the deterministic routing scheme requires several retransmissions and the energy consumption is given by $E_0/(1 - P_{\text{upset}})$. For instance, for a $P_{\text{upset}} = .4$, the communication energy consumption of the deterministic case is 12 nJ, while the energy consumption of the stochastic communication varies from 22.29 nJ for $P = .75$ to 16.64 nJ for $P = .5$ and 10.48 nJ for $P = .25$. Interestingly enough, for higher error rates (e.g., $P_{\text{upset}} = .8$), the communication energy consumption in the deterministic case is 36 nJ, which is higher than the stochastic communication energy consumption which varies from 33.55 nJ for $P = .75$ to 19.85 nJ for $P = .5$ and 7.72 nJ for $P = .25$. Thus, the stochastic communication algorithm proves to work well and consume less energy at higher data upset rates, while the deterministic case seems to imply a nonlinear increases with P_{upset}.

[4] The total energy dissipation can be obtained by multiplying these figures with the packet size S. In our experiments, we have used $S = 364$ bits.

[5] A communication round refers to the number of cycles needed until the first FFT2 value is computed.

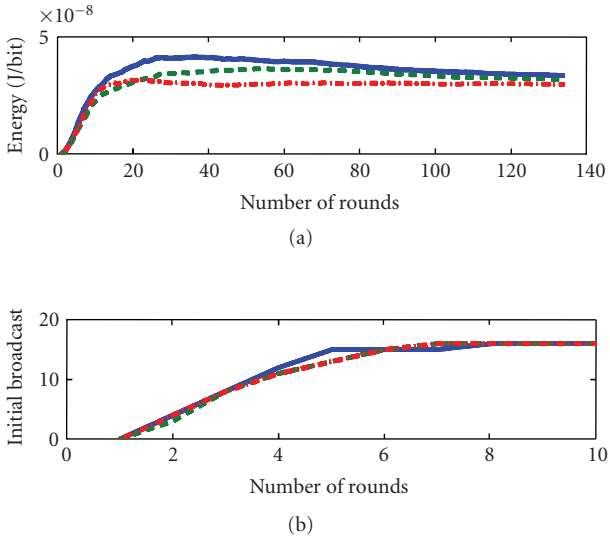


FIGURE 11: Three different runs of the algorithm.

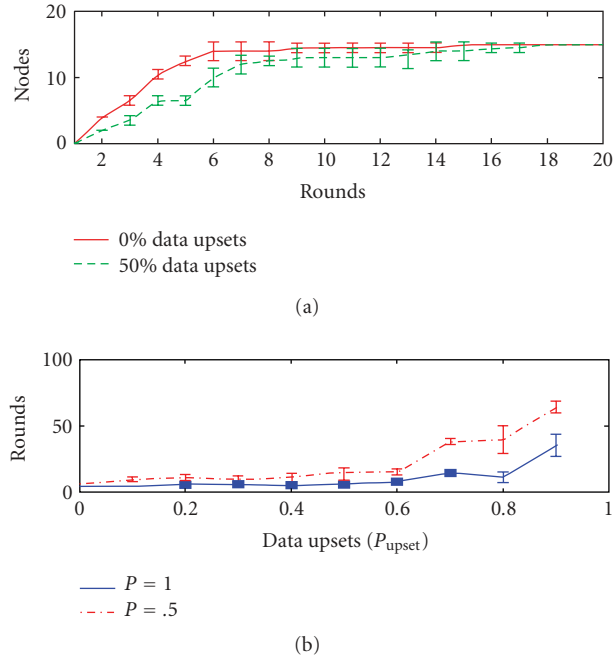


FIGURE 12: Coverage and latency of the stochastic communication.

6.1.1. *Repetability of results*

Our experiments show that, for several runs of the protocol, the parameter estimations are consistent and do not fluctuate much around the mean value. To support this claim, we present in Figure 11 a comparison for the evolution of the energy dissipation across the entire simulation, and the number of rounds needed to complete the application.

6.1.2. *Latency*

From the communication point of view, the FFT2 application has two phases.

(i) First, the initial message, containing the raw image samples, has to reach all of the leaf nodes,
(ii) Second, the computed results have to come back to the root node, which will assemble the final result.

The evolution of the first phase, namely the spread of the initial broadcast, is shown in the upper part of Figure 12. As we can see, the spread is relatively slow in the beginning, but soon reaches a stage of explosive growth and so the entire network becomes aware of the message after a small number of rounds. This behavior is reasonably close to the one predicted by the theoretical model in Section 5.2.1 (see Figure 8). In both graphs, the data upsets influence the number of nodes reached. The presence of transmission failures slows down the spreading (dashed line in Figure 12 and the blue line in Figure 8), but the message still reaches most of the network tiles.

The end of the second phase, namely the time when all the partial results reach the root node, marks the completion of the application and is illustrated in the lower part of Figure 12. We note that stochastic communication with probability of transmission $P = .5$ (when up to 50% of the sent packets are corrupted because of data upsets) has a latency very close to the optimal one which corresponds to the flooding algorithm. The standard deviation (shown by the error bars in Figure 12) is small which proves that the stochastic communication has indeed a *stable behavior* and the results can be reproduced across several runs of the algorithm.

Last but not least, these numbers depend on the mapping of IPs to tiles. In some cases, we notice that the replies may come back *before* the full broadcast of the original message reaches all the tiles of the network. This indicates that the application mapping phase has to take into account the communication performance in order to obtain an efficient design [41].

6.1.3. *Energy dissipation*

We have estimated the energy dissipation of our algorithm using (29). We do not include the energy consumed during computation and so, all the results presented here reflect solely the performance of the NoC stochastic communication.

In the top part of Figure 13 we compare the energy consumption of flooding ($P = 1$) and stochastic communication (with $P = .5$) algorithms. While stochastic communication has a latency close to optimal (as explained above), its energy dissipation is about half of the one of the flooding algorithm. This is because the energy is proportional to the total number of messages generated in the network (see Section 5.2.3), which is controlled by the probability of transmission P. This observation indicates that, by modifying parameter P, the designer can tune the trade-off between the performance and the energy dissipation of the communication architecture.

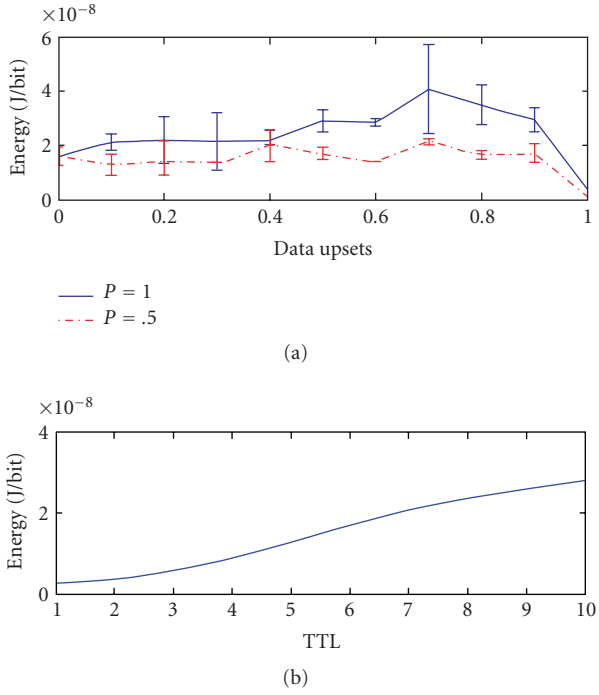


Figure 13: Energy dissipation of the stochastic communication.

The energy dissipation drops to 0 in the extreme case when $P_{\text{upset}} \approx 1$ because all the packets are corrupted and therefore, they are not retransmitted anymore. We also note that the energy dissipation of the stochastic communication also has a smaller variance across several runs of the protocol.

Further energy savings can be achieved by stopping the spreading of messages after a fixed number of rounds. This can be done by assigning messages a finite TTL. For a small TTL, neither the broadcast of the initial message nor the application itself can complete. However, after a certain threshold, the latency does not improve if the TTL is increased. The lower half of Figure 13 shows that the energy consumption increases almost linearly with the TTL. This suggests that the TTL can be set to the smallest value which guarantees tasks completion, in order to keep the energy consumption at minimum.

6.1.4. Fault-tolerance

Different types of failures have different effects on the performance of stochastic communication (see Table 2). The levels of defective links and tiles do not seem to have a big impact on latency. However, if a significant number of permanent tile and link failures occurs during the early stages of the algorithm (fortunately, this is actually not likely with modern manufacturing technologies, as explained in Section 3), the applications would fail because too many important modules are not working or entire regions of the NoC may become isolated.

The computation of FFT2 succeeds in many cases, for instance even with 12.5% faulty tiles and 16.67% faulty links,

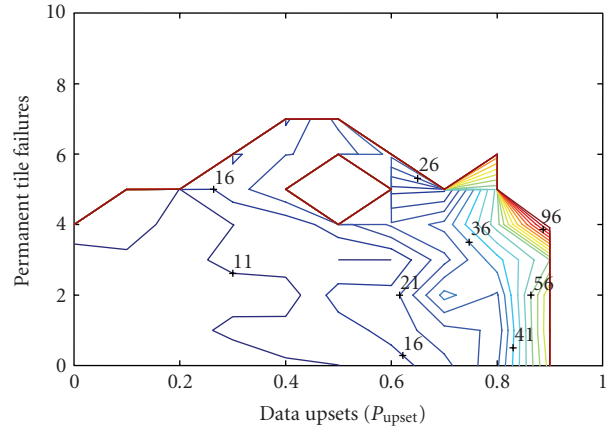


Figure 14: Impact of permanent failures and data upsets on latency.

FFT2 succeeds in almost all runs (see Table 2). The initial broadcast is especially affected by the number of defective links, which can disconnect a region of the chip from the network and prevent those tiles from sending/receiving messages. However, due to resource duplication, even in these cases the computation of FFT2 may succeed, if there are enough resources left that can communicate with each other.

On the other hand, data upsets seem to have little influence on the probability that the applications will be able to terminate; however, upsets do have a serious impact on latency, especially if $P_{\text{upset}} > .5$. Figure 14 illustrates how the defective tiles and data upsets influence the latency of stochastic communication, again on a 4×4 NoC. As shown, FFT2 cannot finish with more than 8 permanently failed tiles. Generally speaking, data upsets will not cause the communication to fail completely. As shown in Figure 14, for less than 4 stopped tiles (this represents 25% faulty tiles), the algorithm eventually terminates with levels of data upsets as high as 90%, even if it requires close to 100 rounds to do so.

6.2. More complex applications

Another application of stochastic communication that was extensively studied in [26] is the MP3 encoder implemented in a "voltage/frequency island" architecture [35]. For this application, some regions of the NoC run at different voltages and even at different frequencies without a significant performance penalty. Indeed, certain parts of the chip, like memories or control logic, do not require as high a voltage as the processor; by placing them on different voltage islands, the total power consumed by the design can be significantly reduced.

As shown in [26], the level of buffer overflows does not seem to have a big impact on latency. Moreover, the synchronization errors do not prevent the application from terminating; however, they do cause a great variability in the latency. Data upsets also seem to have little influence on the chances to finish encoding, but have a significant impact on the latency, especially if $P_{\text{upset}} > .7$.

7. LIMITATIONS AND EXTENSIONS

The on-chip stochastic communication approach is inspired by epidemic models where communication is seen as a consequence of interactions between individuals from various subpopulations (i.e., spreaders, ignorants, and stiflers). These interactions are defined in Section 5.1, in accordance to the restrictions imposed by the topology. Also, the mathematical model captures the situation in which due to buffer overflow, link failures or nodes malfunctions, the dissemination process can be stopped. Further, the mathematical model in Section 5.2.1 retrieves the evolution of coverage of spreader nodes which is also illustrated in the experiments (Section 6.1.1).

A similar behavior can be obtained if the Euler-Maruyama stochastic method is used for evaluating the evolution of the number of spreader nodes over time [42]. Although it seems attractive for its simplicity, this method may not serve as a prediction tool for latency and energy values because it does not capture the spatial relationship between the network nodes. Future analysis should take into account the variation of stochastic communication in space; this can provide insight on the covered surface by the packet dissemination process.

We plan to extend the present work in several directions. One possibility is to address the issue of task mapping in the context of stochastic communication. The main challenge here comes from the difficulty involved in precise TTL evaluation. The TTL value is not only dependent on the distance between source and destination, but also on the network diameter and node buffering resources. This remains an open problem.

Other possible extensions include investigating the tradeoff between TTL and packet transmission probability for different network topologies. Another important extension is to accommodate the proposed approach with a wormhole switching technique and consider the problem of out-of-order packet delivery.

Finally, the problem of CRC code design should be also considered in the NoC context. In our framework, a packet is treated as the basic unit of information. The packet header does not include the routing path information since it is retransmitted many times over different paths. Instead, we assume that each packet is protected by a CRC code. While there are several criteria for characterizing the optimality of the CRC codes, two properties are of particular interest, namely, the Hamming distance and the burst error detection capability. A CRC code of degree d can detect a burst error whose length is smaller than its degree. For instance, Koopman and Chakravarty report in [43] that the USB-5 (i.e., Universal Serial Bus protocol) is optimum for data words of 11-bit long and BERs of 1e-6. Nevertheless, the literature (e.g., [9, 44]) reports that the routing tag consists of at least 16 bits and the required number of bits increases with network size. As such, the use of CRC codes may bring some benefits in reducing the number of header bits. However, the detailed analysis that would help us determine the optimal CRC codes is beyond the purpose of the present work and it is left for future work.

8. CONCLUSIONS

This paper proposed stochastic communication as a novel SoC communication paradigm. This approach takes advantage of the large bandwidth available on chip in order to provide the needed system-level tolerance to synchronization errors, data upsets, and buffer overflows. A theoretical framework which accounts for network topology was proposed to analyze the packet dissemination and interactions between nodes.

Experimental results show that this approach is very scalable, robust, and has a low latency, while keeping the energy consumption at reasonable levels. At the same time, this method simplifies the design process by offering a low-cost and easy to customize solution for on-chip communication. Last but not least, stochastic communication enables various hybrid architectures that may be used to enable on-chip diversity.

ACKNOWLEDGMENTS

The authors thank Nick Zamora and Umit Ogras of System Level Design group at CMU for their insightful comments on the stochastic communication approach. Also, the authors would like to thank Sam Kerner for help with the experimental part in the initial stages of this project. This Research was supported by SRC 2004-HJ-1189 and Marco GSRC.

REFERENCES

[1] C. Constantinescu, "Impact of deep submicron technology on dependability of VLSI circuits," in *Proceedings of the International Conference on Dependable Systems and Networks (DNS '02)*, pp. 205–209, Washington, DC, USA, June 2002.

[2] W. Maly, "IC design in high-cost nanometer-technologies era," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 9–14, Las Vegas, Nev, USA, June 2001.

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

[4] D. Bertozzi, L. Benini, and G. De Micheli, "Low power error resilient encoding for on-chip data buses," in *Proceedings of Design, Automation and Test in Europe Conference and Exhibition (DATE '02)*, pp. 102–109, Paris, France, March 2002.

[5] T. Dumitraş, S. Kerner, and R. Marculescu, "Towards on-chip fault-tolerant communication," in *Proceedings of the Asia and South Pacific Design Automation Conference (ASP-DAC '03)*, pp. 225–232, Kitakyushu, Japan, January 2003.

[6] T. Valtonen, T. Nurmi, J. Isoaho, and H. Tenhunen, "Interconnection of autonomous error-tolerant cells," in *Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS '02)*, vol. 4, pp. 473–476, Phoenix, Ariz, USA, May 2002.

[7] H. G. Lee, U. Y. Ogras, R. Marculescu, and N. Chang, "Design space exploration and prototyping for on-chip multimedia applications," in *Proceedings of the ACM/IEEE 43rd Design Automation Conference (DAC '06)*, pp. 137–142, San Francisco, Calif, USA, July 2006.

[8] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 684–689, Las Vegas, Nev, USA, June 2001.

[9] A. Jantsch and H. Tenhunen, *Networks on Chip*, Kluwer Academic Publishers, Norwell, Mass, USA, 2003.

[10] L. Gasieniec and A. Pelc, "Adaptive broadcasting with faulty nodes," *Parallel Computing*, vol. 22, no. 6, pp. 903–912, 1996.

[11] T. Leighton, B. Maggs, and R. Sitaraman, "On the fault tolerance of some popular bounded-degree networks," in *Proceedings of the 33rd IEEE Annual Symposium on Foundations of Computer Science*, pp. 542–552, Pittsburgh, Pa, USA, October 1992.

[12] L. M. Ni and P. K. McKinley, "A survey of wormhole routing techniques in direct networks," *Computer*, vol. 26, no. 2, pp. 62–76, 1993.

[13] G. De Micheli, "Robust system design with uncertain information," in *The Asia and South Pacific Design Automation Conference (ASP-DAC '03) Keynote Speech*, Kitakyushu, Japan, January 2003.

[14] T. Karnik, S. Borkar, and V. De, "Sub-90nm technologies—challenges and opportunities for CAD," in *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD '02)*, pp. 203–206, San Jose, Calif, USA, November 2002.

[15] A. Demers, D. Greene, C. Hauser, et al., "Epidemic algorithms for replicated database maintenance," in *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing*, Vancouver, British Columbia, Canada, August 1987.

[16] D. Estrin, R. Govindan, J. Heidemann, and S. Kumar, "Next century challenges: scalable coordination in sensor networks," in *Proceedings of the 5th Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '99)*, pp. 263–270, Seattle, Wash, USA, August 1999.

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

[18] D. J. Daley and J. Gani, *Epidemics Modelling: An Introduction*, Cambridge University Press, Cambridge, UK, 1999.

[19] D. J. Daley and D. G. Kendall, "Stochastic rumours," *IMA Journal of Applied Mathematics*, vol. 1, no. 1, pp. 42–55, 1965.

[20] C. E. M. Pearce, "The exact solution of the general stochastic rumour," *Mathematical and Computer Modelling*, vol. 31, no. 10, pp. 289–298, 2000.

[21] K. P. Birman, M. Hayden, O. Ozkasap, Z. Xiao, M. Budiu, and Y. Minsky, "Bimodal multicast," *ACM Transactions on Computer Systems*, vol. 17, no. 2, pp. 41–88, 1999.

[22] B. Kantor and P. Lapsley, "Network News Transfer Protocol," RFC 977, February 1986. http://www.w3.org/Protocols/rfc977/rfc977.

[23] K. Lidl, J. Osborne, and J. Malcolm, "Drinking from the firehose: multicast USENET news," in *Proceedings of the USENIX Winter Technical Conference*, San Francisco, Calif, USA, January 1994.

[24] S. Floyd, V. Jacobson, C.-G. Liu, S. McCanne, and L. Zhang, "A reliable multicast framework for light-weight sessions and application level framing," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 784–803, 1997.

[25] XTP Forum, "Xpress Transfer Protocol Specification Revision 4.0," March 1995.

[26] T. Dumitraș and R. Marculescu, "On-chip stochastic communication," in *Proceedings of the Design, Automation and Test in Europe Conference and Exhibition (DATE '03)*, pp. 790–795, Munich, Germany, March 2003.

[27] S. Manolache, P. Eles, and Z. Peng, "Fault and energy-aware communication mapping with guaranteed latency for applications implemented on NoC," in *Proceedings of the 42nd Design Automation Conference (DAC '05)*, pp. 266–269, ACM Press, Anaheim, Calif, USA, June 2005.

[28] P. Bogdan and R. Marculescu, "A theoretical framework for on-chip stochastic communication analysis," in *Proceedings of the 1st International Conference on Nano-Networks (NANONETS '06)*, Lausanne, Switzerland, September 2006.

[29] C. Constantinescu, "Dependability analysis of a fault-tolerant processor," in *Proceedings of Pacific Rim International Symposium on Dependable Computing*, pp. 63–67, Seoul, South Korea, December 2001.

[30] R. Horst, D. Jewett, and D. Lenoski, "The risk of data corruption in microprocessor-based systems," in *Proceedings of the 23rd International Symposium on Fault-Tolerant Computing (FTCS-23 '93)*, pp. 576–585, Toulouse, France, June 1993.

[31] T.-T. Y. Lin and D. P. Siewiorek, "Error log analysis: statistical modeling and heuristic trend analysis," *IEEE Transactions on Reliability*, vol. 39, no. 4, pp. 419–432, 1990.

[32] C. Constantinescu, "Trends and challenges in VLSI circuit reliability," *IEEE Micro*, vol. 23, no. 4, pp. 14–19, 2003.

[33] V. Hadzilacos and S. Toueg, "A modular approach to fault-tolerant broadcasts and related problems," Tech. Rep. TR94-1425, Department of Computer Science, Cornell University, Ithaca, NY, USA, May 1994.

[34] D. M. Chapiro, *Globally-asynchronous locally-synchronous systems*, Ph.D. thesis, Stanford University, Stanford, Calif, USA, 1984.

[35] D. E. Lackey, P. S. Zuchowski, T. R. Bednar, D. W. Stout, S. W. Gould, and J. M. Cohn, "Managing power and performance for system-on-chip designs using voltage islands," in *Proceedings of IEEE/ACM International Conference on Computer Aided Design (ICCAD '02)*, pp. 195–202, San Jose, Calif, USA, November 2002.

[36] T. Chelcea and S. M. Nowick, "Robust interfaces for mixed-timing systems with application to latency-insensitive protocols," in *Proceedings of the 38th Design Automation Conference (DAC '01)*, pp. 21–26, Las Vegas, Nev, USA, June 2001.

[37] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry, "Stochastic Models for Social Processes," in *Proceedings of the 6th Annual ACM Symposium on Principles of Distributed Computing, Vancouver*, John Wiley & Sons, British Columbia, Canada, August 1987.

[38] D. J. Watts, *Small Worlds, the Dynamics of Networks between Order and Randomness*, Princeton University Press, Princeton, NJ, USA, 1999.

[39] D. J. Wilkinson, *Stochastic Modelling for Systems Biology*, Chapman & Hall Press, London, UK, 2006.

[40] R. Milner, *Communication and Concurrency*, Prentice-Hall, Upper Saddle River, NJ, USA, 1989.

[41] J. Hu and R. Marculescu, "Energy- and performance-aware mapping for regular NoC architectures," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 24, no. 4, pp. 551–562, 2005.

[42] P. E. Protter, *Stochastic Integration and Differential Equations*, Springer, Berlin, Germany, 2004.

[43] P. Koopman and T. Chakravarty, "Cyclic Redundancy Code (CRC) polynomial selection for embedded networks," in *Proceedings of the International Conference on Dependable Systems and Networks (DSN '04)*, pp. 145–154, Florence, Italy, June-July 2004.

[44] J. Duato, S. Yalamanchili, and L. M. Ni, *Interconnection Networks: An Engineering Approach*, Morgan Kaufmann Publishers, San Francisco, Calif, USA, 2002.