To be published in Proceedings 22nd IEEE International Symposium on Defect and Fault Tolerance in VLSI Systems, Roma Sept 26-28 2007

Production Yield and Self-Configuration in the Future Massively Defective Nanochips

Piotr Zając (a,b), Jacques Henri Collet (a)
(a) LAAS – CNRS, University of Toulouse, France
(b) Department of Microelectronics and Computer Science, Technical University of Lodz, Al. Politechniki 11, 93-590 Lodz, Poland
{pzajac, jacques.collet}@laas.fr

Abstract

We address two problems in this work, namely, 1) the resilience challenge in the future chips made up of massively defective nanoelements and organized in replicative multicore architectures and 2) the issue of preserving the production yield. Our main suggestion is that the chip should be self-configuring at the architectural level, enabling with almost no external control mechanisms, core mutual-test to isolate the defective core and self-configuration of communications to discover the routes in the defective network. Our contribution is a systematic study of the dependence of the production yield versus the core failure probability (possibly as high as 0.4) in several networks with different node connectivity ranging from 3 to 5. The result is obtained in terms of a probabilistic metrics to warrant that a minimal fraction of nodes can be contacted by the input-output port for participating to the processing.

1. Introduction

Thanks to size reduction and technology advances, chips with an extremely large number of transistors, say typically several hundreds billions transistors, will become feasible in the next decade. While this trend is prone to result in substantial performance improvements, two fundamental challenges are to be faced [1], namely, the control of architectural complexity and the increase of defective elements in the physical layer, as extreme downsizing inevitably results in increased inter- and intra-device variability [2,3] and ultimately in massively defective technologies, thus impairing the production yield of such nanochips.

The suitable response to architectural complexity is *replication* and even *massive replication*. Actually, it is nothing new because nature has been applying the replication for million years in all the fields, in biology, chemistry, crystals, etc… Replication solves the *physical* complexity and enables redundancy although it generates new issues to predict the collective behavior and (or) the programming. In this context, one may expect that, in near future, the number of cores and the organization of processor chips will evolve significantly beyond the symmetric multiprocessor architectures such as those of today's popular bi, or quadricore processor chips. The recent announcement for a 80-core chip by Intel [4], already paves the way forward. In this work, we shall consider large processor arrays including typically from hundred to thousand cores. In our opinion, the long-term response to the increasing rate of defective elements is *self-organization* that may be viewed as a new adaptative fault-tolerance technique. Similar principles are already applied in the circuit layer, especially for what concerns memory chips. Most advanced techniques consist in providing spare elements (lines, rows or words) in order to dynamically replace some defective elements. Indeed, techniques have been proposed that not only cope with production defects but also with faults

occurring at runtime. Such techniques are primarily meant to achieve a high yield, which may require a significant overhead. For example, in [5], it is shown that for a 1Mb-chip and a cell defect ratio of 3%, a near 100% yield can be achieved, but at the cost of close to 100% overhead.

In this work, we consider reconfiguration and self-organization at the *architectural* level. By reconfiguration and self-organization at the architectural level, we mean:

a) Self-diagnosis of cores in the array through mutual tests. The underlying idea is to split valid and defective cores in disconnected zones with as little as possible external control, as it would become unrealistic to consider diagnosing all cores and routes in a very complex chip via some external equipment. Alone, such an approach would require increasingly — perhaps prohibitively — high effort and cost in manufacturing and testing.
b) Self-configuration of communication routes.
c) Self-shutdown of the cores which cannot take part to the processing, i.e., which are defective or inaccessible.
d) Self-adaptative management of execution redundancy and allocation at runtime.

Steps *a-c* must be executed at startup, and possibly periodically at runtime. The expected benefits from such self-configuring capabilities are i) Improving operation resilience in general processing systems including a fraction of defective cores, and ii) enabling a smooth degradation of the chip performance as a function of the number of defective cores. This way, it would be possible to continue using the processing resources available onchip even when faults occurring in operation will impair some additional cores. The performance would be simply gradually reduced accordingly (as aging process in real life!). In this paper, we focus on steps *a-b*. In Section 2, we briefly introduce the different network topologies that we consider to change the node connectivity from 3 to 5 because the connectivity is a crucial parameter to maintain communications in defective networks. Section 3 describes how to conduct self-diagnosis through mutual tests; in particular, we show that mutual tests enable splitting the chip into disconnected zones of consistent cores such that all cores are good or all are defective in a zone. In Section 4, we study the efficiency of the routing discovery to assess how many valid cores can be reached via a flooding protocol as a function of the node failure probability and the node connectivity. We study the production yield in Section 5. Finally, Section 6 provides some concluding remarks.

2. Architectural framework

We briefly describe here the architectural framework. Figure 1a depicts an example of such a framework with 9x5 nodes organized in a 2D-mesh network and including one single input/output port (IOP). Each node is made up of a processing core associated to a router, which forwards messages to enable intercore communications. Lines connecting the routers materialize inter-router links. In this figure, blackened cores depict defective ones; 9 cores are identified as such: C_{52}, C_{58}, C_{43}, C_{31}, C_{32}, etc. This would correspond to a massively defective chip featuring a core failure probability of about 20%. Also, dashed lines identify logically disconnected links. Perhaps, one of the most important network parameters (regarding the capability of maintaining communications in the presence of a large fraction of defective cores) is the node connectivity. Thus, we consider in Figures 1b to 1d three other networks with respective connectivity C ranging from 3 to 5. They are built from the hexagonal, the hyper-cube and the torus

topologies. The torus topology is similar to the 2D-mesh except that there is no border so that all nodes have exactly the same connectivity C=4.

We concentrate in what follows on permanent core failures. The core failure probability P_F is an adjustable parameter in our study that will be varied typically from 10 to 40%. The consideration of permanent link failures (in addition to the core failures) is perfectly integrable in our proposal of self-configuration, but the scope of this paper. The problem that we consider in sections 3 and 4 is how diagnosing cores and discovering valid routes at startup (i.e., when the chip is powered on), as it is not known which cores are defective and which communication routes are available to exchange messages between nodes.

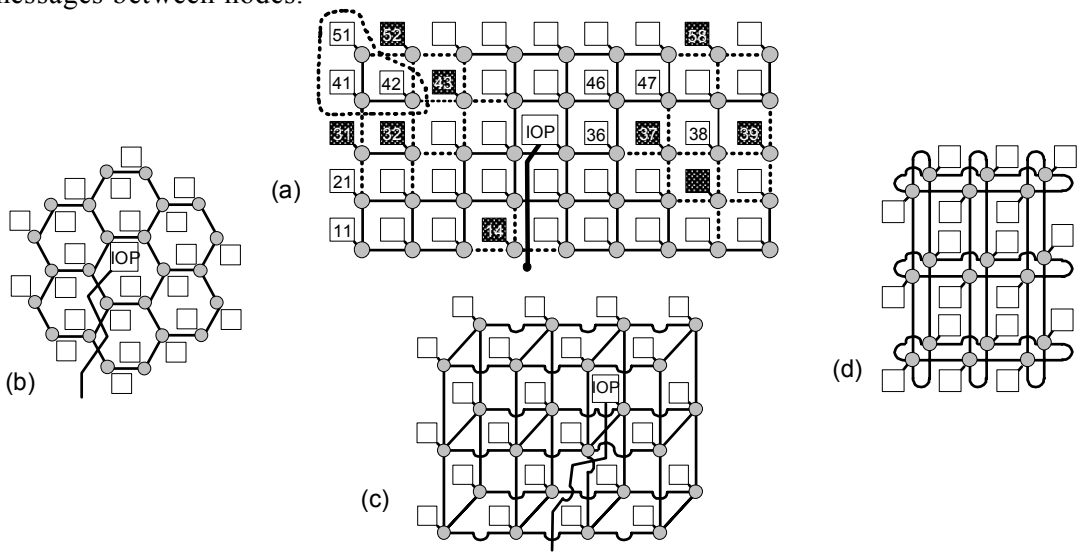

Figure 1. a: 9x5 node network architecture (connectivity C= 4, 45 nodes). b: Hexagonal architecture (C=3, 24 nodes); c: Hypercube (C=5, 24 nodes); d: Torus (C=4, 18 nodes)

3. Self-diagnosis via mutual test

Chip self-diagnosis is based on mutual-tests. In this approach [6], each core in the network executes *separately* the diagnosis program that we assume to be locally stored in a flash-like memory. Then, it compares its own generated data with those generated by each direct neighbor. The mutual-diagnosis approach exclusively relies on the properties of good (i.e., not-defective) cores, as the operation of defective cores is unpredictable. If a core is good and if the test data generated by a neighbor are different, it stops all communications with this one. Note that this disconnection is no physical disconnection, but contrarily a logical disconnection in that a good-core simply stops exchanging data with any adjacent defective core. *The disconnection mechanism splits the core array in several simple-connected zones (SCZ) which are mutually exclusive*. In this approach, the SCZ enclosing the IOP (denoted as the MSCZ) is especially important, because all cores outside this zone are lost for the processing. For sake of clarity, let us refer to Figure 1a. For instance, C_{36} will conclude that C_{37} is faulty and it will disable all communications with it. All disabled links appear as dashed. Note that the dotted line in the top left corner encloses the cluster of good cores (C_{51}, C_{41}, C_{42}), which are disconnected from the MSCZ and therefore cannot take part to the processing. Of course, the presence of inaccessible nodes becomes all the more problematic as the fraction of failing cores increases in the network, as it will be shown in section 4.

However, it persists a pernicious problem in that we do not know the true state of the cores in the MSCZ, although we know that all cores in a SCZ are in the same state, i.e., all are good or all are defective. Of course, the respective probabilities of the two situations are very different, since the probabilities of having *n* good or *n* defective cores are $(1 - P_F)^n$ or P_F^n respectively. To remove this ambiguity, we suggest executing one external diagnosis of the *sole* IOP to make sure that it works correctly. Ultimately, the self-diagnosis is not completely autonomous and requires a minimal external test to validate in an unquestionable way the MSCZ.

4. Self-configuration of communications

The stage that we consider here consists in discovering the routes connecting the IOP to good cores. Remember that valid routes are unknown, because the topology of the MSCZ is *a priori* unknown and because it might be necessary to move around (clusters of) defective nodes. Nevertheless, these routes are crucial for chip operation, as the IOP will use them to allocate incoming tasks to idle cores. The basic idea is that the IOP should include a *buffer storing the valid routes* (VRB) to the idle cores, so that the IOP allocates the incoming processes by searching a core in this buffer. Of course, this buffer is initially empty. Consequently, the route discovery described below must be executed at startup to initialize the VRB. Route discovery is achieved by means of a *contract net protocol* (CNP) based on the exchange of two messages [7]. It can be typically decomposed in the three following phases:

- Request phase: First, the IOP emits a *request message* (RM). We consider a broadcast diffusion, where each node forwards each incoming RM to all links, except the incoming link [8], so-called hot-potatoe forwarding (HPF). Flooding protocols are successful in discovering routes (as shown below) and enable moving around the disconnected zones, i.e., around zones including cores diagnosed as faulty in the mutual-test procedure depicted in the previous paragraph. The trick here is that each router forwarding a RM adds in this one the routing executed locally, so that during propagation, the RM stores the followed route by storing the successive routings. For instance, in a topology with 4 links per node as in Figure 1a, one may consider the following coding ID_{North}=00, ID_{East}=01, ID_{South}=10, and ID_{West}=11. A node adding 0111 to the route field of the RM tells that the message came in from East and was forwarded to West. Note also that no absolute node addressing is needed in this communication approach.

- Acknowledgment phase: Each valid node receiving the RM sends an *acknowledgment message* (AM) back to the emitter. The AM simply follows the RM route in the opposite direction, which dramatically limits the number of reemissions. Globally, the number of AMs returning to the IOP is as large as the number of contacted nodes in the MSCZ.

- Valid Route Buffer initialization: The IOP therefore collects the AM messages and stores the routes in the VRB.

We study in the rest of this section the efficiency of the *request phase*, i.e., the capability of contacting cores in a faulty network via the flooding mechanism. The efficiency of the route discovery mechanism (RDM) is studied as a function of the fraction of defective nodes and for the four network topologies displayed in Figure 1. The RM flooding was simulated using the multiagent simulator MASS. A detailed and comprehensive documentation of MASS is available from the URL http://www.laas.fr/~collet together with the possibility of downloading the simulator. Briefly, MASS is a Windows® application developed at LAAS, which calculates the temporal evolution of any system that can be described in terms of coupled state

automata (SA). The full description of MASS is beyond the scope of this paper. MASS was successfully used in multiagent simulations, communication, prey-predator games. In the route discovery studies considered here, each node is represented by a SA, which forwards incoming messages. All nodes are activated in the asynchronous mode through a global scheduler. The principle of each simulation is as follows: 1) The simulator randomly generates a fraction of NP_F of holes in the N-core network. Holes represent the faulty cores logically inhibited following the mutual-test process; 2) The IOP emits a RM, which is broadcast across the defective network, following the HPF; 3) The program calculates the number n of nodes receiving the RM. The array $table[N_{MAX}]$ is created and the entry $table[n]$ is incremented every time exactly n nodes received the RM; 4) This procedure is repeated typically 1.000 times. At the end of the simulation, the entry $table[i]$ determines the number of times that i nodes were exactly reached. For instance, $table[26]$=110 would mean that the message reached exactly 26 nodes in 110 simulations out of 1000. The probability to reach 26 nodes is therefore estimated as p(26)=110/1000. Figures 2 and 3 below make it possible to appreciate the effectiveness of the RDM as a function of the node connectivity and versus the fraction of defective nodes in the network. Figure 2 shows the simulation results for the hexagonal and square (2D-mesh) networks and Figure 3 for the torus and the hypercube.

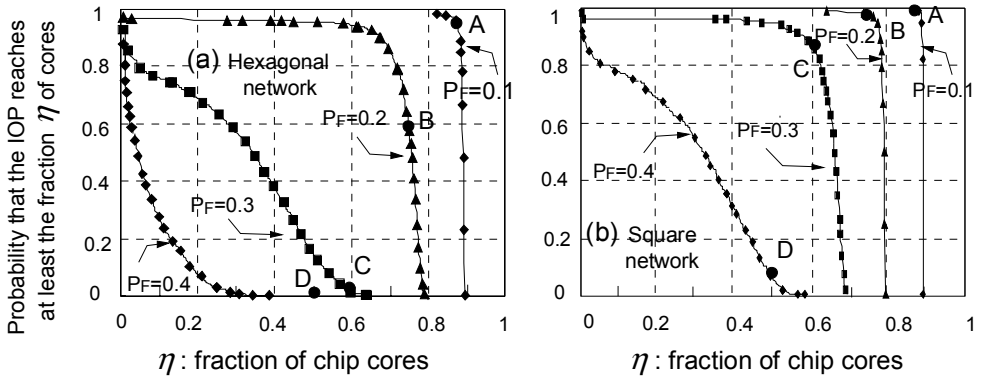

Figure 2. Probability that the IOP reaches at least a fraction η of cores in a hexagonal (a) or a 2D-mesh topology (b) versus the fraction of defective nodes. The network is made up of 450 cores.

Concretely, these figures show the *probability that the IOP reaches at least the fraction η of cores* in a large network comprising 450 cores. The simplest way to explain this figure is undoubtedly to consider particular points. So, let us consider points *A* with abscissa X=0.88 in the four figures corresponding to weakly defective arrays with 10% of defective cores (i.e., P_F=0.1). X=0.88 means here that about 2% of the cores are lost for the processing because they cannot be reached. The ordinate of each point A determines the probability that the IOP reaches at least 88% of the cores. Clearly, the efficiency of the RDM may be considered as satisfactory in all topologies, as the probability is about 0.95 even in the hexagonal topology with the lowest node connectivity (Figure 2a). Now, let us consider the points B in the four figures, which display the probability that at least 75% of the nodes can be reached by the IOP when the network includes 20% of defective nodes. The only case which becomes problematical is that of the hexagonal network (Figure 2a), as the probability reduces to 0.6. The appreciation of the RDM when the network includes 30% of defective nodes is displayed by the four points C. It is very low in the hexagonal network (see Figure 2a). The two networks with connectivity 4 (i.e., the 2D mesh Figure 2b and the torus Figure 3a) may reasonably be considered as satisfactory up to this fraction of defective nodes, as the probability that the IOP reaches at least 60% of the nodes is larger or close to 0.9. However,

40% of the nodes are lost for the processing, corresponding to 30% of defective nodes and 10% of inaccessible nodes!

Now let us analyze points D in the four figures. They show the probability to discover at least 50% of the nodes in the extreme case when approximately 40% of the nodes are defective. Consequently, 50% of the nodes would be lost for the processing (40% because they are defective and 10% being inaccessible). The sole network which can be envisioned as shown in Figure 3b is the hypercube because increasing the node connectivity to 5 raises to about 0.93 the probability to reach at least 50% of the nodes in the chip.

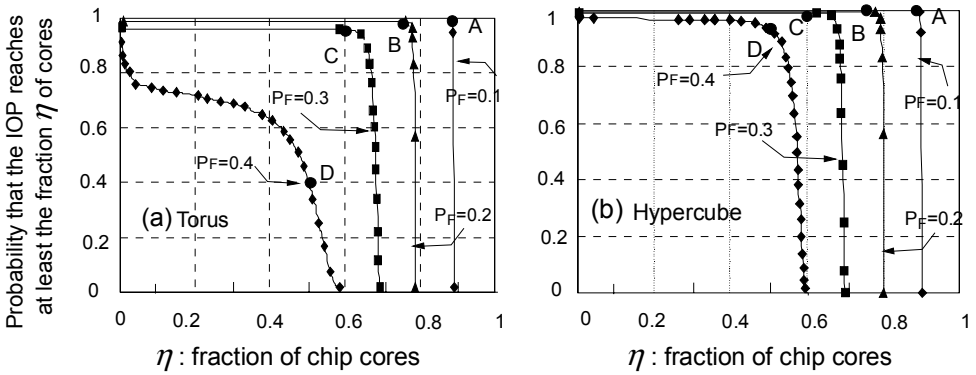

Figure 3. Probability that the IOP reaches at least a fraction η of cores in a torus (a) or an hypercube (b) versus the fraction of defective nodes. The network is made up of 450 cores.

Discussion: In the present state of the study, one may wonder whether it is realistic to consider that the node failure probability P_F could reach 0.4, i.e., that 40% of the nodes could be defective. In fact, the crucial remark is that *the node failure probability P_F does not depend on the sole reliability of the nanotechnology under consideration*. The basic question of deciding whether 10, 20 or 40% of nodes could be defective is also a design problem related to the following issue: On one side, it is very attractive to increase the node complexity, because it increases the node processing power and limits the constraint of parallelizing the execution of tasks across the network. However, on the other side, increasing the node complexity raises the failure probability P_F, thus increases the fraction of defective nodes and consequently the difficulties to establish valid routes in the network as shown above. Therefore, one must consider the node failure probability P_F as resulting from an open and complex design tradeoff between the desirable top complexity of the core (the more complex, likely the better), the protection overhead to maintain P_F under the values deduced from this study, and ultimately the acceptable network connectivity to maximize the processing power. What is sure, it is that the increase of the failure rate of the nanoelements will inexorably force to reduce the size of cores (to limit the fraction of defective cores and to preserve the communications as shown above) and ultimately leads to the parallelization barrier, i.e., to the very complex (and open) issue of maintaining the efficiency of the processing, in spite of the constraint of massively parallelizing the applications and distributing the processing to weak nodes in the network.

5. Probabilistic production yield

We study in this section the production yield (PY), which is a crucial parameter in any massively defective technology. The first possible idea to validate a chip might consist in counting the number of cores responding to the RM emitted by the IOP, and in selecting the chips which enable contacting at least the predefined fraction η of cores. In this approach, the PY can be directly derived from Figures 2 and 3. Simply, multiply the ordinate of any point

by the probability $(1-P_F)$, as the IOP must be also fault free. For instance, if we consider point C (X=0.6, Y≈0.95) in Figure 3a (i.e., corresponding to the torus topology when the failure probability is P_F=0.3), we conclude that the PY would be approximately 0.66 when selecting chips which enable contacting at least 60% of all nodes. Now, this approach has two drawbacks. First, it only considers the average number of defective nodes, and completely ignores their positions in the network, in particular in the vicinity of the IOP. Second, it is clear that all the traffic concentrates around the IOP and that the failure of nodes close to the IOP (especially the adjacent neighbors) will dramatically limit the communication bandwidth, when contrarily, the defective nodes far from the IOP will have little effect. In first approximation, it is expected that the communication bandwidth will be approximately proportional to the number of valid nodes adjacent to the IOP. So, in the framework of the PY analysis, it is necessary to sort and validate the chips depending on the number of valid adjacent nodes. The probability $P_L(k,n_C,P_F)$ that the IOP is good (i.e., fault-free) and that at most k nodes (out of n_C connected to it) are defective reads:

Eq. 1
$$P_L(k,n_C,P_F)=(1-P_F)\sum_{i=0}^{k}\binom{n_C}{i}(1-P_F)^{n_C-i}P_F^i$$

$P_L(k,n_C,P_F)$ is a kind of locality factor, which takes into account the state of the environment around the IOP. In first approximation, the PY is the product of this locality factor by the probability $P(\eta,P_F)$ (displayed in Figures 2 and 3) to reach at least the fraction η of nodes in the network under consideration, i.e. [1],

Eq. 2
$$PY(k,n_C,\eta)=P_L(k,n_C,P_F)P(\eta,P_F)$$

Figures 4 and 5 display the application of Eq. 2 to estimate the production yield in a 450-node network, organized in the topologies previously considered in Figure 1.

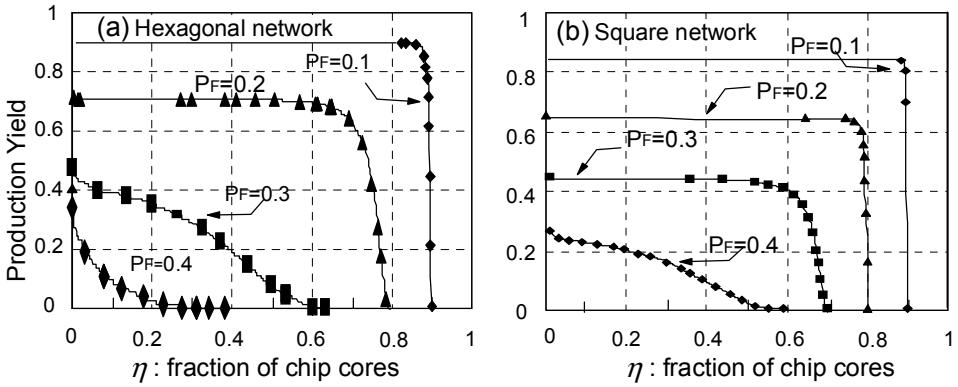

Figure 4. Estimation of the production yield $P(1,n_C,\eta)$ when selecting chips having at most one defective node adjacent to the IOP. (a): Hexagonal network; (b) Square network.

At most one defective node adjacent to the IOP is tolerated for chip validation. In other words, at least 2 out of 3 nodes adjacent to the IOP are not defective in the hexagonal network, 3 out of 4 in the square lattice or in the torus, and 4 out of 5 in the hypercube. The two figures show the degradation of the PY in the highest connectivity network because this approach (i.e., assuming at most one defective adjacent node) increases the selectivity of the

[1] Eq. 2 neglects the correlation between the locality state around the IOP and the capability to reach a fraction of nodes in the network. However, we conducted several additional simulations for the 2D-mesh, considering P_F ranging from 0.1 to 0.4 and analyzing the influence of the number of valid nodes adjacent to the IOP on the route discovery efficiency. There is almost no change up to P_F=0.3 with the results displayed in Figures 2 and 3. The correction is at worst a few percents when P_F=0.4. Consequently, the presentation that we follow with Eq. 2 has the advantage of being simple but it slightly underestimates PY when the node failure probability is larger that 0.3.

validation rule in the hypercube with respect to the torus, and in the torus with respect to the hexagonal network.

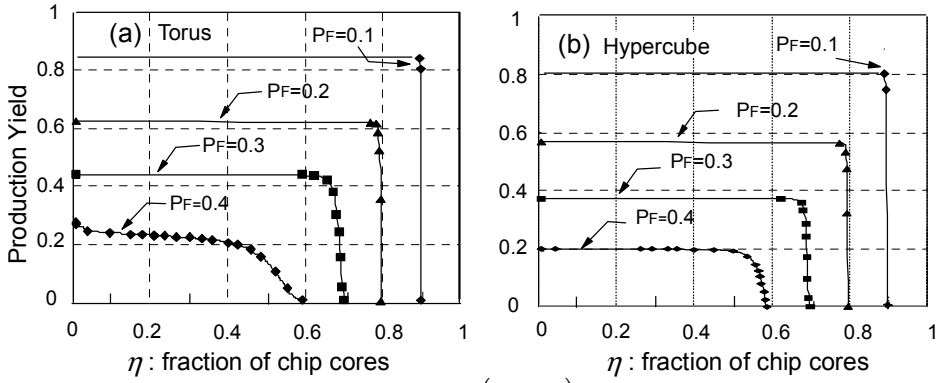

Figure 5. Estimation of the production yield $P(1,n_C,\eta)$ when selecting chips having at most one defective node adjacent to the IOP (a): Torus; (b) Hypercube.

The results are different if one calculates the PY for the different topologies assuming the same minimum communication bandwidth of the IOP, i.e., the same minimum number of valid adjacent nodes. Figure 6 shows such a PY calculation for the torus and the hypercube topologies assuming that at least 2 adjacent nodes are valid. Note that this condition slightly increases the chip sorting selectivity with respect to that considered in Figures 2 and 3, which implicitly assume that, at least, one node adjacent to the IOP is good. We observe in Figure 6 an increase of the PY with respect to Figure 5 due to the increase of the locality function $PY(k,n_C,\eta)$. That is no surprise. For instance in the hypercube, the locality function increases because it is obviously easier to find at least 2 good adjacent nodes out of 5 as in Figure 6b rather than 4 out of 5, as in Figure 5b. Figures 4-6 make it possible to appreciate the effectiveness of the RDM and consequently the YD versus the fraction of defective nodes in the network. It turns out that the square network constitutes a good choice when less than 20% of cores are defective. When P_F is between 0.2 and 0.3, the torus is a better choice regarding the network complexity. The hypercube should be privileged when the node failure probability reaches 0.4.

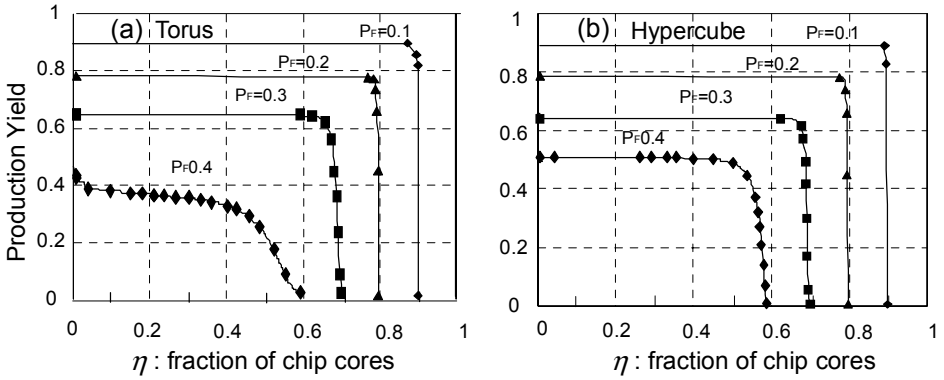

Figure 6. Estimation of the production yield in the torus and hypercube topologies when selecting chips having at least two good nodes adjacent to the IOP.

6. Conclusion

We have described a self-configuration methodology to tolerate defective nodes in chips made up of massively defective nanoelements and organized in replicative multi-core architectures. The idea behind self-configuration is that the chip should become autonomous and adaptative to preserve its resilience, and that as little external

interventions as possible should be involved to control the start up phase and the subsequent operation. We only described the self-diagnosis of cores through mutual tests, and the self-configuration of communications through a contract network protocol, which consists in discovering the routes by broadcasting a request message across the network and in storing the routes of the responding nodes in the VRB of the IOP.
We studied the efficiency of the route discovery mechanism as a function of the fraction of defective nodes in the network considering several topologies selected to change the node connectivity from 3 (in the hexagonal network) to 5 (in the hypercube). The results are described in terms of a probabilistic metrics to warrant that *at least a minimal fraction of valid nodes can be contacted by the IOP* for participating to the processing (see Figures 2 and 3). In Figures 4-6, we also studied the production yield depending on the number of valid nodes directly adjacent to the IOP, as in first approximation, it is expected that the IOP communication bandwidth should be proportional to this number. Of course, the larger is the IOP connectivity, the less disturbing is the influence of the defective adjacent nodes. We studied in particular in Figure 6 the production yield versus the node connectivity when at least two adjacent nodes are good (i.e., fault free) to warrant the same minimal bandwidth. When P_F is between 0.2 and 0.3, the torus is the better choice. The hypercube should be privileged when the node failure probability reaches 0.4.

Ultimately, let us add this comment: The insights gained from this study rest on the assumption that the self diagnosis is perfect, i.e., that the fault coverage of test vectors is complete during the mutual-test process described in Section 3. Now, it is well known that it is generally not true, especially if the core in each node implements a complex micro-architecture. Thus, it will be safe to consider that any core deemed as fault-free by the mutual test could be ultimately faulty, especially in massively defective technologies, when P_F is as high as 0.2 or 0.3. Consequently, a safe and conservative position will consist in implementing another fault-tolerance layer at runtime, based on the redundant execution of the applications among the available nodes that the IOP can contact using the routes stored in its VRB. Additionally, this fault tolerance layer is mandatory for coping with the transient faults occurring at runtime.

Acknowledgment: It is our pleasure to thank Jean Arlat and Yves Crouzet (from LAAS-CNRS) for a critical reading of the manuscript and for their support and encouragements during the development of this work.

7. References

[1] R. I. Bahar, D. Hammerstrom, J. Harlow, W. H. Joyner, C. Lau, D. Marculescu, A. Orailoglu, and M. Pedram, "Architectures for Silicon Nanoelectronics and Beyond," Computer, vol. 40, no. 1, pp. 25-33, January 2007.

[2] A. J. Bhavnagarwala, X. Tang, J. D. Meindl, "The Impact of Intrinsic Device Fluctuations on CMOS SRAM Cell Stability", IEEE Journal on Solid-State Circuits, vol. 36, no. 4, pp. 658–665, April 2001.

[3] S. Roy, A. Azenov, "Intrinsic Parameter Fluctuations in Nano-scale CMOS," *Science*, vol. 309, no. 5733, pp. 388-390, July 2005.

[4] S. Vangal *et al.*, "An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS " in Proc. IEEE International Solid-State Circuits Conference (ISSCS-2007), San Francisco, CA, USA, 2007, (IEEE CS Press).

[5] M. Nicolaïdis, N. Achouri, L. Anghel, "A Diversified Memory Built-In Self-Repair Approach for Nanotechnologies," in Proc. 22nd IEEE VLSI Test Symp (VTS'2004), Napa Valley, CA, USA, 2004, pp. 313-318, (IEEE CS Press).

[6] L.E. Laforge, K. Huang, V.K. Agarwal, "Almost Sure Diagnosis of Almost Every Good Elements," IEEE Trans. on Computers, vol. 43, no. 3, pp295-305, 1994.

[7] R.G. Smith, "The Contract Net Protocol: High-level Communication and Control in a Distributed Problem Solver," IEEE Trans. on Computers, vol. 29, pp. 1104-1113, 1980.

[8] Y.K. Dalal, and R.M. Metcalfe, "Reverse Path Forwarding of Broadcast Packets," Communications of the ACM, vol. 21, no. 12, pp.1040-1048, 1978.