CONSTRUCTING SPARE SHARING NETWORKS FOR RELIABILITY ENHANCEMENT OF SCALABLE SYSTEMS

Soroush Khaleghi, Wenjing Rao

University of Illinois at Chicago
Department of Electrical and Computer Engineering
Chicago, Illinois, USA

ABSTRACT

Future systems based on nano-scale devices will be highly susceptible to operational faults. When hardware redundancy is used to boost reliability, through the repair-based approaches, redundancy is most effectively utilized if any spare unit can be made available to replace any faulty unit. However, such fully connected network of spare sharing demands dreadfully expensive interconnection when system size and complexity scale up. Consequently, while spare sharing provides flexibility, the sharing must be limited due to interconnection constraints.

In this paper, we study the reliability issue of systems with shared spare units, under limited interconnection constraints. Particularly, we focus on the construction of the "spare sharing network" to boost system reliability. The proposed methodology connects the vulnerable units to the exploitable existing spares, thus strengthening the entire system without adding extra spares. Simulation results confirm that the proposed methodology achieves a significant reliability boost at very small cost of additional interconnects.

Index Terms— Reliability, Fault tolerance, Nano-scale devices, Hardware redundancy, Spare sharing

1. INTRODUCTION

Next generation of Signal Processing Systems (SiPS) can benefit from nano-scale devices to achieve significantly better performance and energy efficiency at lower cost. However, the minute scale of nano devices makes them highly susceptible to operational faults [1], [2], [3]. Therefore, one of the critical limitations facing all future systems (based on emerging nanoelectronic devices or progressively scale-down CMOS ones) is unreliability.

Hardware redundancy techniques such as fault masking and repair-based schemes are typically employed to boost reliability. In the fault masking approaches, such as N-Modular Redundancy (NMR) [4], multiple copies of a component perform the same task to produce a single output through a majority vote. These schemes are very expensive for high fault rates, as is the case of nano systems [5]. Repair-based approaches rely on fault detection and standby spare units. After detecting the presence of a fault, a subsequent repair process ensures the correctness of the system by replacing the faulty unit with a standby spare unit [6], [7], [8], [9]. Compared to the fault masking schemes, repair-based approaches require significantly less hardware overhead, particularly when a few spare units could be shared among many functional units for replacement [5], [10]. For repair-based approaches to work, a system needs to support reconfiguration in two ways:

1. The interconnections must be reconfigurable, so that spare units can redirect the I/O channels of the faulty unit: such a reconfigurability can be supported at various hierarchical levels. Among the emerging nanoelectronic devices, reconfigurabilities are inherently supported by many candidates [11], [12], [13]. At the logic level, regular structures, such as PLAs, support the reconfiguration to implement any boolean function. More generally, configurable systems, such as FPGAs, contain a number of configurable logic blocks (CLBs) and programmable interconnects, which are utilized to implement a reconfigurable digital circuit. At the system level, Network-on-chip (NoC) interconnection techniques support reconfigurability.
2. The functionality of a spare unit must be compatible to the functional units to be replaced: such a compatibility is supported at various hierarchical levels: at the arithmetic level, most components such as multipliers and adders are made of many identical and replaceable units; at the processor architecture level, several homogeneous computational units including ALUs are employed; at the system level, such as Multi-Processor System-on-Chip (MPSoC) systems, faulty processors can be replaced by spare processors [14].

Apparently, reliability can be most effectively achieved from full redundancy sharing. In other words, if every spare unit can be made available to replace every functional unit, this maximum flexibility makes the system most reliable. Although implementation of spare units are becoming progressively cheap, interconnection and communication are becoming prohibitively expensive [1], [2]. Therefore, spare sharing

THIS WORK IS SUPPORTED BY NSF GRANT 1149661.

has to be limited under strict interconnection constraints. In addition, spare units can be shared only among interchangeable units. For instance, a spare Ripple Carry Adder might be used to replace a faulty Carry Look-Ahead Adder, yet it cannot replace a faulty multiplier.

Most recent studies of spare-sharing fault tolerance techniques aim at the *optimal* fault tolerate systems. This assumption, however, leads to an enormous complexity for large systems, which limits the application of the approaches to a small range of highly simple or regular system structures [15], [14], [16]. Furthermore, they usually do not consider the limited interconnection constraints. Therefore, having a low-cost design technique that can be applied to any topology (with limited interconnection constraints) is necessary for future scalable systems.

In this paper, we present a framework of constructing reliable systems based on limited spare sharing among functional units. A methodology is proposed to boost the reliability of any given spare sharing network. The proposed approach works by identifying a set of criteria to pinpoint the "most vulnerable" functional unit and the "most exploitable" spare unit in a network. Connecting the units together then boosts the entire system reliability.

The rest of this paper is organized as follows: Section II motivates the research work and presents the preliminaries on reliability models. In section III, the proposed methodology for boosting reliability is presented. Section IV shows the simulation results and section V concludes the paper.

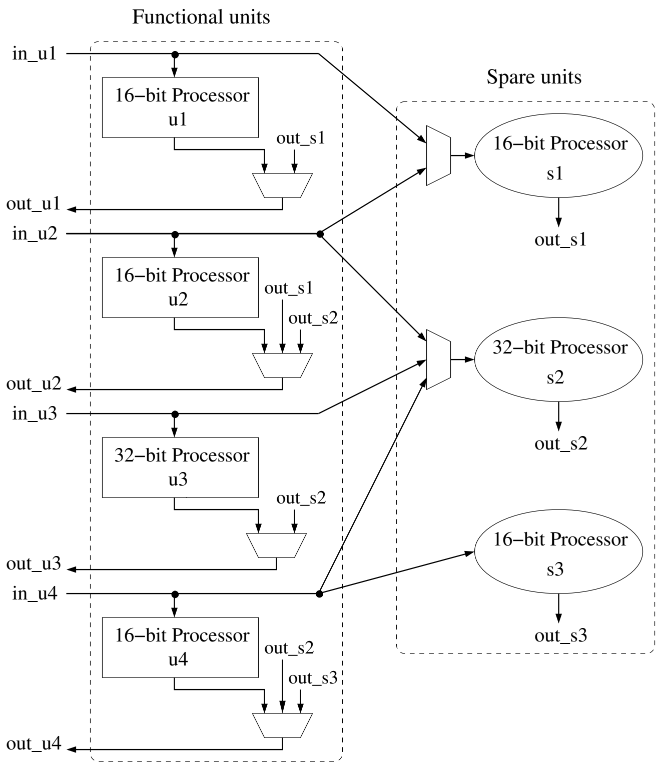

Fig. 1: Example of a system with three spare processors shared among four functioning ones

2. MOTIVATIONS AND PRELIMINARIES

In this paper, we denote the original components in a system as *functional units*, and the standby ones, which are capable of replacing faulty functional units, as *spare units*.

Figure 1 shows a Multi-Processor System with built-in spare units ($s1, s2, s3$), that can be used to replace specific functional units ($u1, u2, u3, u4$). In this example, if a fault occurs at $u2$ (a 16-bit processor), then it can be replaced by either $s1$ (a 16-bit processor) or $s2$ (a 32-bit processor). In order to replace a faulty functional unit with a spare unit, the spare needs to cover the operation of the faulty functional unit and redirect the I/O of the functional unit. This can be done by additional MUXes, as is shown in Figure 1. Apparently, the overhead (in terms of MUX cost and interconnection cost) increases rapidly as the range of sharing expands.

2.1. System Model

For a system consisting of a functional unit set, a spare unit set, and the corresponding replacement relationships, it can be uniquely represented by *bipartite graph* $BG(N_U, N_S, E)$, where N_U and N_S represent the functional unit set and the spare unit set, respectively. Each edge $e = (u, s) \in E$, where $u \in N_U$ and $s \in N_S$, indicates that the corresponding functional unit u can be replaced by spare unit s. From the bipartite graph model, it is clear that the amount of redundancy (number of spare units) embedded in the system is depicted by $|N_S|$. The limited spare sharing of the system, imposed by interconnect constraint and functionality, is represented by the topology of the network. Specifically, the fan-out degree of a spare unit node (denoted by $d(s)$) illustrates that the spare unit s can replace any one of the $d(s)$ connected functional units; the fan-out degree of a functional unit u (denoted by $d(u)$) represents the number of its accessible spare units. The overall number of edges in the bipartite graph, $|L|$, depicts the interconnection complexity of the system.

Figure 2(a) shows the bipartite graph representation of the example in Figure 1. Squares and circles illustrate functional units and spare units respectively. Replacing a faulty functional unit with an accessible spare unit is translated in the bipartite graph model as follows: after the repair process, the allocated spare unit will be removed with all of its associated edges, and the faulty functional unit node can be marked as fault-free again. Figure 2(b),(c),(d) illustrates a repair process. The spare replacement in 2(b) shows a *successful repair*, in which faulty $u4$ is replaced by $s3$, as is shown in Figure 2(c). Figure 2(d) depicts an unrepairable fault sequence: after the second fault occurs at $u3$, no spare unit is available for $u3$, and thus no successful repair exists for this fault. In general, when a faulty functional unit has no accessible spare units, no repair process exists, and a *system failure* occurs.

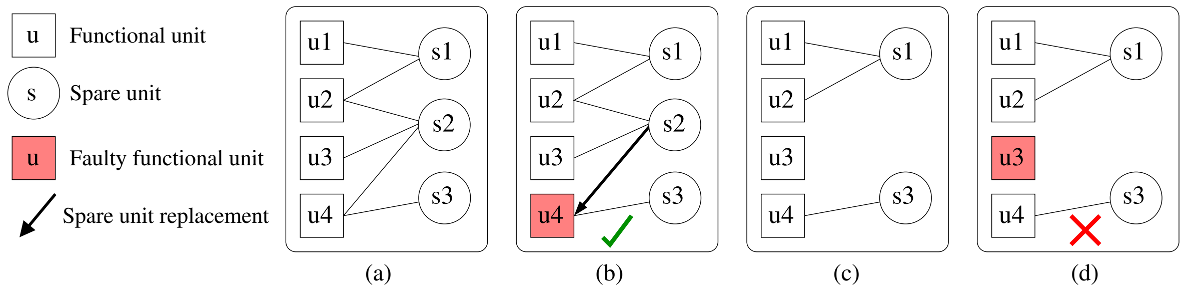

Fig. 2: Examples of bipartite graph representation and repair process: (a): the spare sharing network in Figure 1 (b): a successful repair after one fault (c): the remaining system after the successful repair (d): system failure: unrepairable after the second fault happening at $u3$

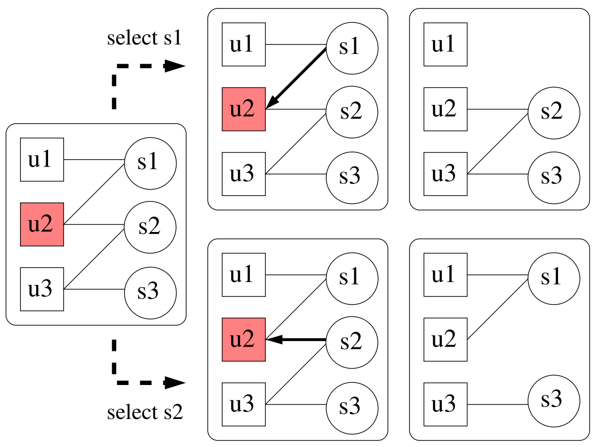

Fig. 3: Example of different resultant systems according to distinct spare unit selections: the repair process performed at bottom guarantees the remaining system to survive the next fault, but the one at top might fail if the next fault happens at $u1$.

2.2. Reliability Model

There are three main factors, contributing to the reliability of a system:

1. *Fault Sequence*: A system might survive or fail, depending on the fault occurrences. For example, in Figure 2(c), a fault occurring at $u3$ cannot be repaired (Figure 2(d)); however, if a fault happens at $u1$ or $u2$, the system can be repaired by $s1$.
2. *Network Topology*: The reliability of a system heavily depends on the amount of redundancy (number of spare units), and the spare sharing network topology.
3. *Repair Process Algorithm*: Each functional unit can be replaced by any of its connected spare units upon fault occurrences. Thus, whether or how long a system can survive, depends on the spare unit selection made upon every fault occurrence. Figure 3 shows an example of two different resultant systems for the same fault, according to different selections made for spare unit.

To determine which of the resultant systems in Figure 3 is more reliable, we adopt the reliability model of [17]. This model essentially measures the survival probability of a system, under a random sequence of faults with repair options.

Definition: The reliability of a system under f faults, $RE(BG, f)$, is defined by the percentage of repairable cases over all the $|N_U|^f$ possible fault occurrence patterns.

For this reliability model we assume: 1) independent fault occurrence on each functional unit with equal probability, and 2) sequential occurrences of faults, i.e. one repair is performed after every single fault occurrence. For f faults occurring sequentially on a set of $|N_U|$ functional units, there is a total number of $|N_U|^f$ equally possible fault occurrence patterns. For any of the $|N_U|^f$ fault occurrence patterns, it is defined as repairable if at least one successful repair sequence exists for it [17].

Plotting the entire reliability curve $RE(BG, f)$ is computationally impractical. However, it is possible to deduce a number of characteristics [17]:

- *Monotonically decreasing*: The reliability of a system decreases as the number of fault increases.
- *0-point*: Since each faulty functional unit is replaced by a spare unit, the maximum number of faults that a system can tolerate is no more than the number of spare units, i.e. Reliability function always becomes 0 when the number of fault occurrences is greater than $|N_S|$.
- *100%-point*: A system can always survive the next fault as long as all functional units have at least one connected spare unit. Therefore, if the number of fault occurrences is less than the minimum fan-out degree in the functional unit node set, reliability remains at 100%.

In order to boost system reliability, two approaches can be performed:

1. *Network Construction*: This approach concerns the topology of bipartite graph, i.e. the number of spare units, and the sharing structure.
2. *Spare Selection Algorithm*: This approach targets the repair process. The reliability of a system can be enhanced based on carefully designed spare unit selection algorithm, upon each fault occurrence. Various spare unit selection algorithms have been suggested in [17], [15], [14].

3. RELIABILITY ENHANCEMENT VIA NETWORK CONSTRUCTION

In this paper, a network construction technique is proposed; involving adding extra interconnects between existing functional units and spare units. Basically, the following question is to be asked: given a small budget in the form of adding a few extra connections, how should the extra connections be placed, such that system reliability is boosted to the largest extent? To do so, we develop a set of criteria to pinpoint: 1) the most vulnerable functional unit, and 2) the most exploitable spare unit, of a given network.

3.1. Motivation

Adding an extra connection to a functional unit has the obvious benefit of expanding access to more spare units. The question is then which functional unit can benefit the most. Naturally, the functional units with the least accessible spare units are more likely to benefit from such extra connectivity. According to the reliability function, the whole system fails if any faulty functional unit has no more accessible spare units. Therefore, any extra budget to strengthen one of the functional units should go to the most vulnerable one (with the minimum accessible spare units).

The effect of adding an extra connection to a spare unit is not as clearly beneficial as it is for a functional unit. On one hand, it makes a spare unit accessible by more functional units, which entails a benefit. On the other hand, this "expansion of service" for the specific spare unit has a side effect on the group of functional units originally connected to this spare unit. Since the spare unit will have one more functional unit to support, this originally supported group of functional units will have reduced chance of support from this spare. To select the spare unit with the least side effect as such, an already "popular" spare unit should not be picked, because a wide neighborhood of functional units will then be affected. However, "popularity" is not the sole criterion. Sometimes, a spare unit is "crucial" or "urgently needed" for some functional unit(s), because this functional unit does not have much access to other spare units. Therefore, both "popularity" and "urgency" of a spare unit need to be considered as criteria.

3.2. Spare unit selection: the most exploitable one

Based on the above discussion, we have two considerations for selecting a spare unit to add one extra connection.

1. *Popularity*: The number of functional units that each spare unit can replace is denoted by the fan-out degree of its corresponding node in the bipartite graph representation. Let $d(s)$ denotes the *degree* of a spare unit node s.
2. *Urgency*: If a spare unit serves only "well supported" functional units (with accessibility to many spare units), then the "urgency" for this particular spare unit is low, and the side effect of picking this "non-urgent" spare unit is also low. Such "urgency" of a spare unit can be captured by the minimum degree among the spare unit's connected functional units. In a bipartite graph $BG(N_U, N_S, E)$, the adjacent node set $adj(s)$ denotes the set of nodes connected to node $s : adj(s) = \{u : (u, s) \in L\}$. Let $d \downarrow (N)$ denotes the minimum degree among a node set N: $d \downarrow (N) = min\{d(x), \forall x \in N\}$ For a spare unit s, $d \downarrow (adj(s))$ then denotes the number of accessible spare units for the least supported functional unit in the neighborhood of s. This parameter can show whether a spare unit is "urgently" needed. In other words, if $d \downarrow (adj(s1))$ is large for some spare unit $s1$, it shows that even the "weakest" functional unit in its neighborhood has a lot of accessible spare units. On the other hand, if $d \downarrow (adj(s2))$ is small for some spare unit $s2$, it indicates that at least one of the functional units in its neighborhood has a very small number of accessible spare units, thus is relying heavily on $s2$ to survive.

According to our discussion on the criteria of popularity and urgency, clearly, the most exploitable spare unit is the one that is neither urgently needed (with a large $d \downarrow (adj(s))$) nor popular (with a small $d(s)$). By the same logic, the least priority goes to the spare unit that is both urgently needed and popular. Since system reliability is dominated by the most vulnerable functional unit, the second priority should be to avoid picking an urgently needed spare unit. If $d \downarrow (adj(s))$ is large for a popular spare unit, functional units in its neighborhood are strong enough and do not need it urgently. This makes it a good candidate for extending the accessibility to a new (vulnerable) functional unit. Figure 4 provides the priority ranking for spare unit selection based on two criteria, namely the $d \downarrow (adj(s))$ and $d(s)$ parameters. The principles shown in Figure 4 is implemented by:

1. ranking all the spare units based on urgency ($d \downarrow (adj(s))$) and choose the largest one(s).
2. using popularity ($d(s)$) as a tie-breaker to choose the least popular spare unit among the top ranked ones in 1.

3.3. Functional unit selection: the most vulnerable one

We have motivated in Section 3.1 that the most "vulnerable" functional unit (with the least number of accessible spare units) should be selected for the most gain in reliability boost. This translates into the functional unit with the minimum fan-out degree.

In a sparse network, it often occurs that there exist many functional units tied with the same minimum fan-out degree. In such cases, a careful observation reveals a secondary criterion, which can be used as an effective tie-breaker. Figure 5 shows an example of two functional units, tied in degree, each with only one accessible spare unit: one being less popular ($s2$ in Figure 5 (a)) than the other one ($s1$ in Figure 5 (b)). In this case of deciding which between $u1$ and $u2$ should

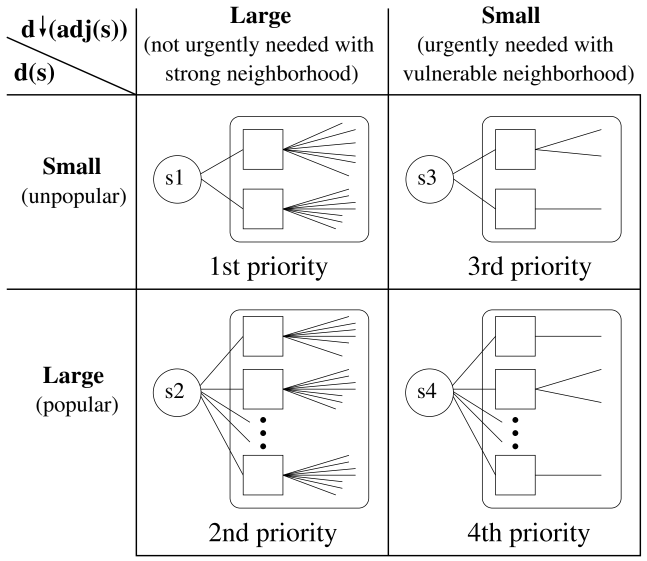

Fig. 4: Spare units selection priorities based on popularity and urgency

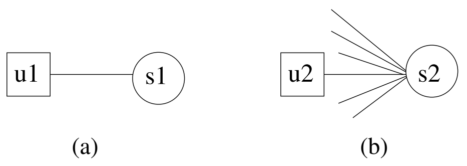

Fig. 5: Example of functional unit selection tie-breaker: (a): $u1$ has an unpopular spare (mostly to itself) (b): $u2$ has a popular spare (that might be used to replace many other functional units)

receive some extra "help", $u2$ is a better choice, because increasing the number of accessible spare units for $u2$ will relieve the urgent demand from $u2$ to $s2$, thus indirectly benefit all the neighboring functional units of $s2$. Such a secondary repercussion of benefit is more significant in the case of $u2$ (because $s2$ has a larger neighborhood), therefore, it should be selected over $u1$. In other words, among all tied functional units $(\forall u | d(u) = d \downarrow (N_U))$, the one with the largest $d \downarrow (adj(u))$ should be selected. Here, $d \downarrow (adj(u))$ essentially indicates the popularity of spare units in the neighborhood of functional unit u.

4. SIMULATION RESULTS

In order to evaluate the reliability gain, we implement the proposed scheme on networks with 15 functional units, 10 spare units, and initialized with 20 connections. The 20 initial connections are placed randomly between the functional units and the spare units. All results are the averages 100 runs of network generation. The reliability of every system is evaluated by the percentage of repairable sequences over 10,000 randomly generated fault sequence patterns, where random choices are made during the repair process, i.e. faulty functional units are replaced by one of their accessible spare units selected randomly.

Figure 6 shows the reliability of the original network and the reliability after adding 5 extra edges to the original network using various methods: 1) spare units and functional units are selected randomly, 2) the most exploitable spare units and random functional units are selected, 3) random spare units and the most vulnerable functional units are selected, and 4) the most exploitable spare units and the most vulnerable functional units are selected. As can be seen intuitively, even randomly added connections can boost reliability over the original network, especially after a number of faults until system failure.

More importantly, the selection of the extra edges affects reliability boost significantly. As it can be seen in Figure 6, selecting the most exploitable spare units (and random functional units) can improve the reliability by up to 15% compared to the original network, and by up to 5% compared to the case of fully randomly selected edges. Selecting the most vulnerable functional units (and random spare units) results in reliability improvements of up to 30% compared to the original network (without extra edges) and of up to 20% compared to the case of fully randomly selected edges. It turns out that functional unit selection is more crucial than spare unit selection, because preserving the most vulnerable functional unit is vital to avoid system failure.

The largest reliability boost is obtained by selecting both the most vulnerable functional units and the most exploitable spare units. In this case, the reliability improvement is of up to 35% compared to the original network, and of up to 25% compared to the fully random selection. The average reliability in this case is 1.75 fold higher than the average reliability of the original network, and 1.4 fold higher than the average reliability of the fully random selection case.

Figure 7 illustrates the cost efficiency of the proposed approach. This graph shows that, by carefully selecting the functional units and spare units, only a few extra edges (in this case, 5) are needed to achieve significant reliability gain, which is more than 70% of that in the original network. A random selection based network construction, however, would need 3 times the cost (15 extra edges in this case) to reach comparable reliability results.

5. CONCLUSION

We have studied the problem of boosting the reliability of future scalable systems with shared spares. Interconnect constraints is taken into consideration such that the spares can only be shared in a limited way. A low cost methodology is proposed to boost the reliability of any given network by adding a very small number of extra connections to expand spare sharing. We developed a set of criteria to pinpoint the most vulnerable functional units and the most exploitable spare units in the network, so that connecting them together

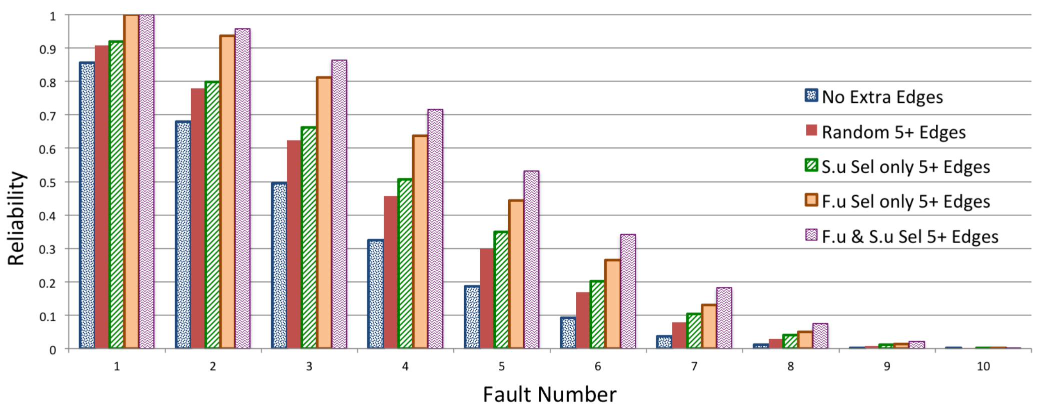

Fig. 6: Reliability boost evaluation of networks with 15 functional units, 10 spare units, and 20 initial edges

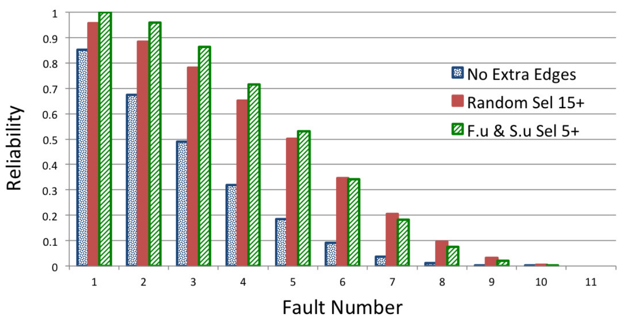

Fig. 7: Cost efficiency of the proposed methodology

will significantly increase the system reliability. Simulation results confirm that the proposed methodology is highly effective and cost-efficient.

6. REFERENCES

[1] ITRS, *International Technology Roadmap for Semiconductors Emerging Research Devices*, 2010.

[2] P. Beckett and A. Jennings, "Towrads nanocomputer architecture," in *Asia-Pacific Computer System Architecture Conference*, 2002, pp. 141–150.

[3] M. Forshaw, R. Stadler, D. Crawley, and K. Nikolic, "A short review of nanoelectronic architectures," *Nanotechnology*, pp. 220–223, 2004.

[4] J. Han and P. Jonker, "A system architecture solution for unreliable nanoelectronic devices," *IEEE Transactions on Nanotechnology*, vol. 1, no. 4, pp. 201–208, December 2002.

[5] K. Nikolic, A. Sadek, and M. Forshaw, "Architectures for reliable computing with unreliable nanodevices," in *IEEE-NANO*, 2001, pp. 254–259.

[6] J. R. Heath, P. J. Kuekes, G. S. Snider, and S. Williams, "A defect tolerant computer architecture: Opportunities for nanotechnology," *Science*, vol. 280, pp. 1716–1721, June 1998.

[7] M. B. Tahoori, M. Momenzadeh, J. Huang, and F. Lombardi, "Defects and faults in quantum cellular automata at nano scale," in *VTS*, 2004, pp. 291–296.

[8] H. Naeimi and A. DeHon, "A greedy algorithm for tolerating defective crosspoints in nanopla design," in *International Conference on Field-Programmable Technology*, 2004, pp. 49–56.

[9] H. Naeimi and A. DeHon, "Seven strategies for tolerating highly defective fabrication," *IEEE Design and Test of Computers*, vol. 22, no. 4, pp. 306–315, 2005.

[10] C. H. Lee, M. A. Perkowski, D. V. Hall, and D. S. Jun, "Self repairable eplds ii: Advanced self repairing methodology," *IEEE Congress on Evolutionary Computation*, pp. 616–623, 2001.

[11] C. P. Collier, E. W. Wong, M. Belohradsky, F. M. Raymo, J. F. Stoddart, P. J. Kuekes, R. S. Williams, and J. R. Heath, "Electronically configurable molecular-based logic gates," *Science*, vol. 285, pp. 391–394, July 1999.

[12] Y. Huang, X. Duan, Y. Cui, L. J. Jauhon, K. Kim, and C. M. Lieber, "Logic gates and computation from assembled nanowire building blocks," *Science*, vol. 294, pp. 13131317, November 2001.

[13] Y. Luo, C. P. Collier, J. O. Jeppesen, K. A. Nielsen, E. DeIonno, G. Ho, J. Perkins, H. Tseng, T. Yamamoto, J. F. Stoddart, and J. R. Heath, "Two-dimensional molecular electronics circuits," *ChemPhysChem*, vol. 3, pp. 519–525, 2002.

[14] L. Zhang, Y. Han, Q. Zu, and X. Li, "Defect tolerance in homogeneous manycore processors using core-level redundancy with unified topology," *Design, Automation and Test in Europe*, vol. 25, pp. 891–896, March 2008.

[15] J. P. Hayes, "A graph model for fault-tolerant computing systems," *IEEE Transactions on computers*, vol. 25, pp. 875–884, September 1976.

[16] M. D. Derk and L. S. Debrunner, "Reconfiguration for fault tolerance using graph grammars," *ACM Transactions on Computer Systems*, vol. 16, pp. 41–54, February 1998.

[17] W. Rao, A. Orailoglu, and K. Marzullo, "Locality aware redundancy allocation in nanoelectronic systems," *IEEE International Symposium on Nanoscale Architectures*, pp. 24–31, June 2008.