Spare Sharing Network Enhancement for Scalable Systems

Soroush Khaleghi and Wenjing Rao
ECE Department, University of Illinois at Chicago, IL 60607, USA
Email: skhale4@uic.edu, wenjing@uic.edu

***Abstract*—Future systems based on nano-scale devices will provide great potentials for scaling up in system complexity, yet at the same time be highly susceptible to operational faults. While spare units can be used to enhance reliability through the repair-based approaches, redundancy is most effectively utilized if any spare unit can be used to replace any faulty unit. However, such a scheme demands a fully connected network of spare sharing, and will be dreadfully expensive when system size and complexity scale up. In this paper, we focus on scalable systems with spare units shared under interconnect constraints in a limited way. Particularly, we propose an approach to add a small number of connections in the "spare sharing network" to boost system reliability. The proposed methodology identifies and connects the vulnerable units to the exploitable spares, thus strengthening the entire system at low cost. Simulation results confirm that the proposed methodology boosts reliability at the small cost of a few additional interconnections.**

I. Introduction

As feature sizes continue to shrink with the advances in nanotechnologies, future systems are expected to scale up in complexity and computational capabilities. 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 nano-electronic 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-based 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 deemed very expensive as fault rates grow, as is the case of nano systems [5]. Repair-based approaches, on the other hand, rely on fault detection and standby spare units. After detecting the presence of a fault, a subsequent repair process replaces the faulty unit with a standby spare unit [6], [7], [8], [9]. As the energy efficiency of devices is not scaling along with integration capacity, the area of the chip that cannot be powered is increasing. This future trend, known as *Dark Silicon*, makes it more plausible to employ repair-based schemes with standby spares at no extra hardware cost in future systems [10].

Repair-based approaches become most efficient when a few spare units can be shared among many functional units, i.e., each spare unit can be used to replace any of multiple functional units [5], [11]. Such a "spare sharing" approach requires the functionality of a spare unit to be compatible with the functional units to be replaced. 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. Such a compatibility is either ensured by employing redundant units, or via reconfigurability, for example in the case of Field Programmable Gate Arrays (FPGAs) and Multi-Processor System-on-Chips (MPSoCs) [12]. In addition to compatible functionality requirement, the interconnections must also be reconfigurable, so that spare units can redirect the I/O channels of the faulty unit. Such a reconfiguration is either supported by various built-in mechanisms in configurable systems, such as FPGAs, or by using additional Multiplexers (MUXes) in a general way, for Network-on-Chip (NoC) and System-on-Chip (SoC) architectures [12]. Among the emerging nanoelectronic devices, general reconfiguration capabilities are inherently supported by many device candidates [13], [14], [15].

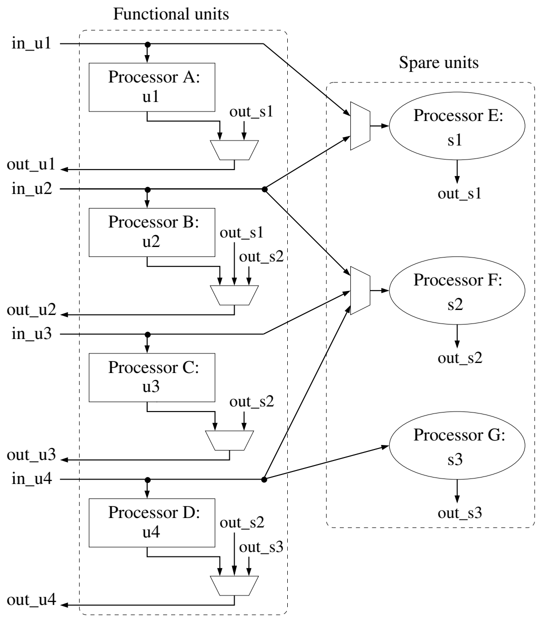
If every spare unit can be made available to replace every functional unit, such a maximum flexibility will result in the highest possible reliability. Although the implementation of spare units are becoming progressively cheap as systems scale up, interconnection and communication are becoming prohibitively expensive [1], [2]. Hence, spare sharing has to be limited under strict interconnection constraints, in addition to the limited functionality compatible units.

Most previous studies of repair-based schemes aim at achieving the *optimal* fault tolerance for a small range of highly simple or regular system structures. Interconnection constraints are not considered as a general limit, but rather at a specific topology such as mesh, ring, or tree structures [16], [12], [17]. Therefore, having a low-cost design technique that can be applied to any topology of sparse interconnection is the goal of our study.

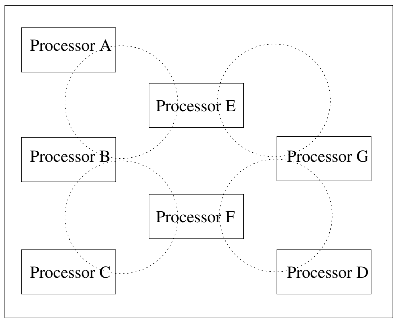
In this paper, we present a methodology of boosting system reliability based on the limited spare sharing constraints. To enhance the reliability of any given spare sharing network, the proposed approach works by: 1) identifying a set of criteria to pinpoint the "most vulnerable" functional unit and the "most exploitable" spare unit in a network; and 2) adding a few extra connections between the identified units, to achieve the maximum gain of reliability boost for the entire system.

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.

This work is supported by NSF Grant 1149661.



(a) Spare sharing implementation



(b) Interconnection constraint: each processor can be replaced by processors within a specific range: D can be replaced by G and F

Fig. 1. Example of a system with three standby spare processors (E,F,G) shared among four functioning ones (A,B,C,D) in a limited way, due to the interconnection constraints.

II. Motivations and preliminaries

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(a) shows an example of a Multi-Processor System with three built-in spare units ($s1, s2, s3$), that can be used to replace the four functional units ($u1, u2, u3, u4$) based on the limited sharing network topology, perhaps imposed by interconnection constraints, shown in Figure 1(b). For example, if a fault occurs at $u2$ (processor B), it can be replaced by either spare unit $s1$ (Processor E) or $s2$ (Processor F), yet it cannot be repaired by any of the functional units. 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 is facilitated by the MUXes, as is shown in Figure 1(a). Apparently, the overhead (in terms of MUX cost and interconnection cost) increases rapidly as the number of interconnects in the sharing network increases. Apparently, a different selection of four functional units (such as A, B, E, F) and three spare ones (such as C, D, G) would result in a different spare sharing network.

A. 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 a *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 the spare unit s. From the bipartite graph model, it is clear that the amount of redundancy embedded in the system is depicted by $|N_S|$ (number of spare units). The limited spare sharing of the system, imposed by interconnection constraint and functionality, is represented by the edges of the network. Specifically, the fan-out degree of a spare unit s (denoted by $d(s)$) illustrates that 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 accessible spare units for u. The overall number of edges in the bipartite graph, $|E|$, approximately depicts the interconnection complexity of the system.

Figure 2(a) shows the bipartite graph representation of the example in Figure 1, with squares and circles illustrating functional units and spare units respectively. Replacing a faulty functional unit with an accessible spare unit is represented 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) illustrate 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: no spare unit is available for $u3$ after the second fault, 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.

B. Reliability Model

Essentially, the reliability of a system is determined jointly by three major factors:

1) *Fault Sequence*: A system might survive or fail, depending on the specific sequence of fault occurrences. For example, in Figure 2(c), a fault occurring at $u3$ cannot be repaired, as is shown in (Figure 2(d)); however, if a fault occurs 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 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 two distinct selections made for spare unit.

To determine which of the resultant systems in Figure 3 is more reliable, we adopt the reliability model of [18]. Reliability is measured by the survival probability of a system, for all the possible fault occurrence sequences with repair options.

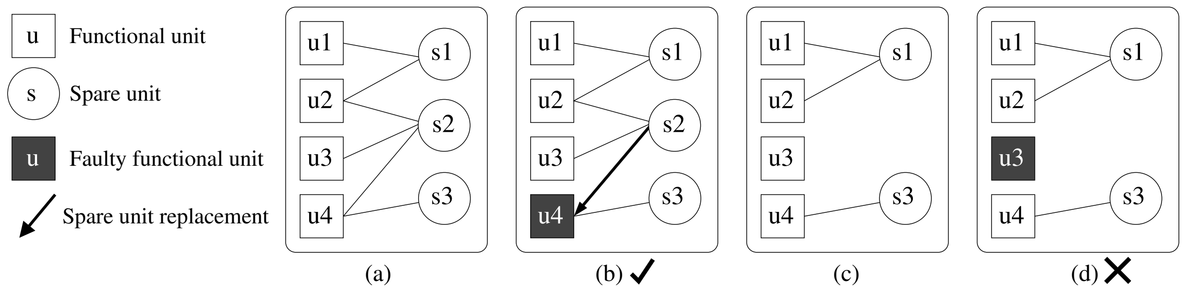


Fig. 2. Examples of bipartite graph representation and repair process: (a) the spare sharing network in Figure 1. (b) a successful repair after one fault. (c) the remaining system after the successful repair. (d) system failure: unrepairable after the second fault happening at $u3$.

With 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 the $|N_U|$ functional units, there is a total number of $|N_U|^f$ equally possible fault occurrence patterns (including repeated fault occurrence onto the same functional unit after repair). 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 [18].

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

Plotting the entire reliability curve $RE(BG, f)$ for a given network is generally impractical, due to the huge number of possible fault sequences. However, it is possible to deduce a number of characteristics of the reliability curve [18]:

1) *Monotonically decreasing*: the reliability of a system decreases monotonically as the number of fault increases.
2) *100%-point*: a system can always survive the next fault as long as all functional units have at least one connected spare unit. Consequently, if the number of fault occurrences is less than the minimum fan-out degree in the functional unit set, reliability remains at 100%.

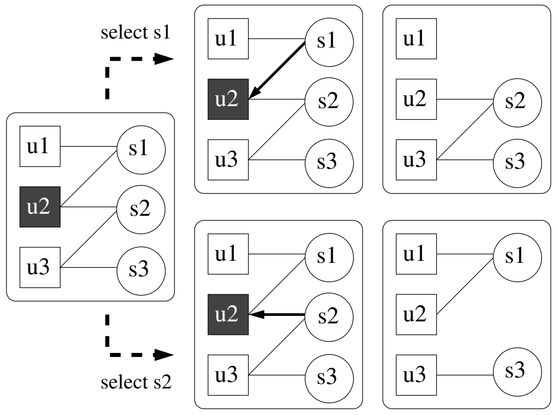


Fig. 3. Example of different resultant systems according to distinct spare unit selections: the repair process performed at bottom guarantees the remaining system to survive the next fault, but the one at top might fail if the next fault happens at $u1$.

3) *0-point*: because 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 drops to 0 when the number of fault occurrences is greater than $|N_S|$.

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

1) *Network Construction* concerns the topology of bipartite graph, i.e., the number of spare units and the sharing structure, and is the main focus of this paper.
2) *Spare Selection Algorithm* targets the repair process. The reliability of a system can be enhanced based on a carefully designed spare unit selection algorithm, upon each fault occurrence. Various spare unit selection algorithms are suggested in [12], [16], [18].

III. NETWORK ENHANCEMENT APPROACH

The reliability of a system depends heavily on the "weakest" links. According to the reliability model, the whole system fails if any functional unit cannot be repaired. Consequently, the best way to construct a network would be to distribute all edges evenly in a regular manner among functional units and spare units. This way, all the links are "equally strong". A high-order *ring* structure, for which all the functional units have a same fan-out degree, and so do all the spare units, can offer such an "optimal" reliability for a given number of edges. However, due to several interconnection constraints such as wiring area, routing delays, power consumption and functionality match, it is not always possible or desirable to have a perfect ring structure network. For this work, we focus on the general framework of improving the reliability of any arbitrary network, with sparse sharing of spare units.

The proposed network enhancement technique involves adding extra interconnections between existing functional units and spare units. Basically, the following question is to be asked: given a small budget (in the form of a few extra connections), how should the extra connections be added, 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. The main contribution of the paper is to present a priority list of adding extra edges, for providing the most reliability gains.

A. Motivations

From the perspective of a functional unit, adding an extra connection has the obvious benefit of expanding access to more spare units. The question is then which functional unit within a given network 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 from the perspective of a spare unit is not as clearly as in the case for a functional unit. On the one hand, it makes a spare unit accessible to more functional units, which indicates an enhancement to reliability. On the other hand, this "expansion of service" for the specific spare unit has a side effect on the group of functional units that are originally connected to this spare unit: 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 of this kind, 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 (set of) functional unit(s) does not have much access to other spare units. Therefore, both "popularity" and "urgency" of a spare unit need to be considered.

B. Spare unit selection: the most exploitable one

Based on the above discussion, we have two criteria 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 its fan-out degree: we use $d(s)$ to denote the fan-out degree of a spare unit node s.
2) *Urgency*: If a spare unit serves only "well supported" functional units (with accessibility to many other 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 functional units connected to this spare unit.
In a bipartite graph $BG(N_U, N_S, E)$, the adjacent node set $adj(s)$ denotes the set of nodes connected to node $s : adj(s) = \{u : (u, s) \in E\}$. Let $D \downarrow (N)$ denote the minimum degree among a node set N: $D \downarrow (N) = min\{d(x), \forall x \in N\}$. For a spare unit s, then, $D \downarrow (adj(s))$ denotes, for the least supported functional unit in the neighborhood of s, the number of its accessible spare units. This criterion 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 the neighborhood of $s1$ has a lot of other 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 very limited access to other spare units other than $s2$, thus is relying heavily on $s2$ to survive. If a choice has to be made between $s1$ and $s2$, $s1$ should be picked over $s2$ to support an extra function unit.

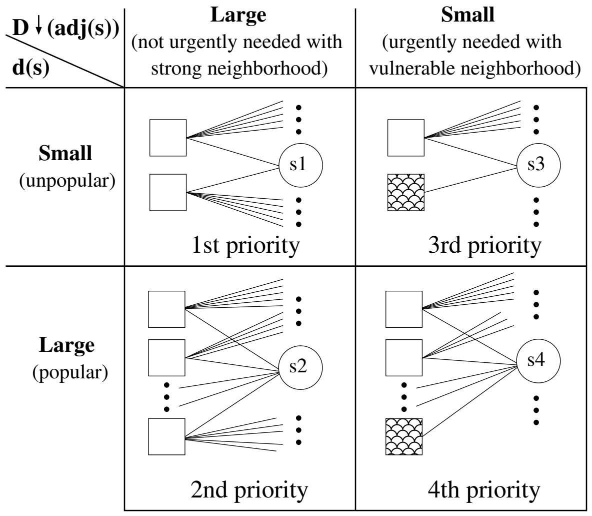


Fig. 4. Spare units selection priorities based on popularity and urgency (functional units filled with the pattern are the most vulnerable ones).

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 the two criteria, namely the $D \downarrow (adj(s))$ and $d(s)$. 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).

C. Functional unit selection: the most vulnerable one

We have motivated in Section III-A 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.

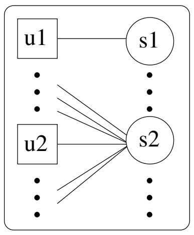
In a sparse network, it often occurs that there exist many functional units tied with the same minimum fan-out degree. In such cases, a careful observation reveals a secondary criterion, which can be used as an effective tie-breaker. Figure 5 shows an example of two functional units ($u1$ and $u2$),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

IV. 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 randomly placed between the functional units and the spare units. All results are the averages over 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. We do not impose any specific repair algorithm, 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 both selected randomly, 2) only the most exploitable spare units are selected, to be connected to a random selection of functional units, 3) random spare units are connected to the most vulnerable functional units, and 4) both the most exploitable spare units and the most vulnerable functional units are selected to be connected.

As can be seen in Figure 6, even randomly added connections can boost reliability over the original network, especially at the later stage of fault occurrence sequences. More importantly, the selection of the extra edges affects reliability boost significantly. Figure 6 shows that, selecting the most exploitable spare units (to "help" a set of random functional units) can improve the average reliability by 48% compared to the original network (without extra edges), and 12% compared to the case of fully randomly selected edges. Selecting the most vulnerable functional units (and random spare units) results in better average reliability improvements by 103% compared to the original network and 54% compared to the case of fully randomly selected edges. Apparently, functional unit selection is more crucial than spare unit selection, because preserving the most vulnerable functional unit is vital to avoid system failure.



u1 has an unpopular spare (s1) mostly to itself –> the extra edge only benefits u1

u2 has a popular spare (s2) that might be used used to replace many other functional units –> the extra edge benefits all of them

Fig. 5. Example of functional unit selection tie-breaker.

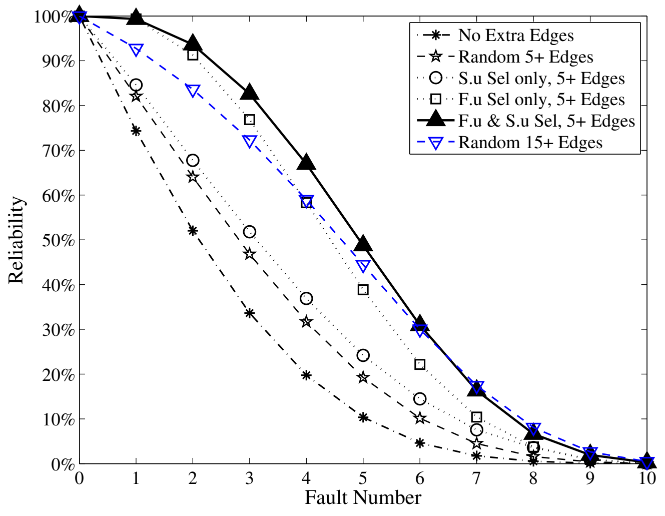


Fig. 6. Reliability boost evaluation of networks with 15 functional units, 10 spare units, and 20 initial edges.

The largest reliability boost is obtained by selecting both the most vulnerable functional units and the most exploitable spare units. In this case, the average reliability improvement is 127% compared to the original network and 70% compared to the fully random selection.

Figure 6 also illustrates the cost efficiency of the proposed approach. It 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 twice (2.27 times) of that in the original network. A random selection based network enhancement, however, would need 3 times the cost (15 extra edges in this case) to reach comparable reliability results.

Figure 7 depicts a spectrum of approaches for building a network with 15 functional units, 10 spare units and 45 edges. At one end of the spectrum, the network is constructed entirely randomly, while at the other end, the proposed methodology is used to add all the edges, one at a time. Two other networks in the middle are shown by starting with 35 random edges plus adding 10 edges using the proposed methodology, and 30 random edges plus 15 selected edges, respectively. These approaches are compared against a network constructed with the highest reliability, when all the 45 edges are distributed evenly in a regular manner, forming a high-order ring structure. As it can be seen, the most reliability boost is resulted when moving from random approach to the one of adding a few edges selectively. This graph illustrates that, in order to have a reliable network, it is not necessary to build the network from scratch; even a very small number of edges added using the proposed criteria, at the end of a randomly built network can elevate reliability to a level that is comparable to a network with the optimal reliability (evenly distributed edges of ring structure). Therefore, during the design process of a spare sharing network, few edges in the last phase as suggested by the proposed methodology seems to be practical to enhance the system reliability significantly.

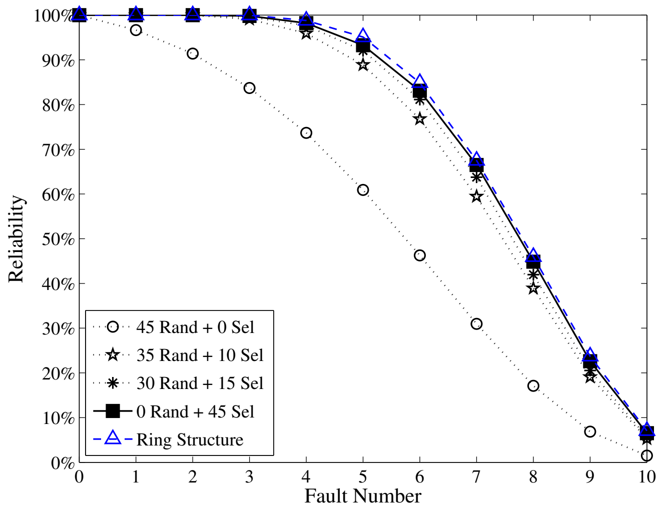


Fig. 7. Various approaches for building a network with 15 functional units, 10 spare units, and 45 edges.

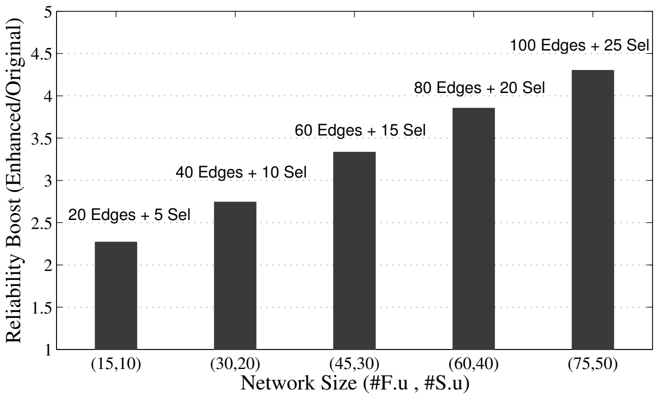


Fig. 8. Scalability of the proposed methodology (the number of initial and extra edges for the first network is 20 and 5, respectively).

Figure 8 verifies the scalability of the proposed methodology, i.e., the larger the network size, the more gain can be achieved by the proposed methodology. It shows that the reliability boost of the enhanced network over the original network increases as system size scales. As can be seen in Figure8, by scaling the network size (the number of functional units, spare units, initial edges, and extra edges) the proposed approach can elevate the average reliability (over the faults) more significantly for the larger systems.

V. Conclusion

We have studied the problem of boosting the reliability of future scalable systems with shared spares. Interconnection constraint is taken into consideration for scalable systems, 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 will significantly increase the system reliability. Simulation results confirm that the proposed methodology is highly effective and cost-efficient in reliability boost for scalable systems.

References

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

[2] P. Beckett and A. Jennings, “Towrads nanocomputer architecture,” pp. 141–150, 2002.

[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,” pp. 254–259, 2001.

[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,” pp. 291–296, 2004.

[8] H. Naeimi and A. DeHon, “A greedy algorithm for tolerating defective crosspoints in nanopla design,” pp. 49–56, 2004.

[9] ——, “Seven strategies for tolerating highly defective fabrication,” *IEEE Design and Test of Computers*, vol. 22, no. 4, pp. 306–315, 2005.

[10] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, “Dark silicon and the end of multicore scaling,” *IEEE Micro*, vol. 32, no. 3, pp. 122–134, 2012.

[11] 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.

[12] 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.

[13] 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.

[14] 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, p. 13131317, November 2001.

[15] 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.

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

[17] 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.

[18] 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.