TABLE II
COMPARISON OF THE DETERMINISTIC AND THE HYBRID ATPG APPROACH

circuit	determin. approach		hybrid approach		speed up
	det. fault coverage	CPU-time	det. fault coverage	CPU-time	
C432	99.35 %	3:46	99.35 %	1:13	3.1
C499	98.87 %	15:45	100 %	1:54	8.3
C880	100 %	6:10	100 %	20	17.9
C1355	98.86 %	1:15:08	100 %	3:42	20.2
C1908	99.76 %	40:57	100 %	1:27	28.1
C2670	99.35 %	5:38:30	99.7 %	3:21:56	1.7
C3540	99.56 %	3:26:29	99.73 %	1:57:18	1.8
C5315	100 %	2:34:35	100 %	24:49	6.2
C6288	100 %	11:15:30	100 %	12:58	52.1
C7552	99.87 %	33:26:35	99.99 %	11:24:42	3

CPU-time are considerable. The average speedup obtained for the ten benchmark circuits was 14.2. The acceleration of test pattern computation is maximal for random pattern susceptible faults, but even for random pattern resistant circuits a reduction by a factor of 1,7 is obtained.

VII. CONCLUSIONS

A hybrid ATPG system has been outlined and a general criterion for switching from fault simulation to algorithmic test pattern computation is derived. The criterion is based on a model of online monitoring the simulation process and estimating the fault detection probabilities. A prototype of the hybrid ATPG system was implemented on an Apollo DN3000. Compared to a conventional ATPG system, better coverage and shorter test generation times were obtained. The speedup of the ATPG process is maximal for random pattern susceptible circuits but even for circuits with lots of random pattern resistant faults the hybrid ATPG system is superior. The average speed up was 14.2 for the implemented prototype system.

REFERENCES

[1] S. Funatsu, N. Wakatsuki, and T. Arima, "Test generation systems in Japan," in *Proc. 12th Design Automat. Symp.*, vol. 6, June 1975, pp. 114–122.

[2] E. B. Eichelberger and T. W. Williams, "A logic design structure for LSI testability," in *Proc. 14th Design Automat. Conf.*, 1977, pp. 462–468.

[3] H. Ando, "Testing VLSI with random access scan," in *Proc. COMPCON*, 1980, pp. 50–52.

[4] J. P. Roth, W. G. Bouricius, and P. R. Schneider, "Programmed algorithms to compute tests to detect and distinguish between failures in logic circuits," *IEEE Trans. Electron. Comput.*, vol. EC-16, pp. 567–580.

[5] P. Goel, "An implicit enumeration algorithm to generate tests for combinational circuits," *IEEE Trans. Comput.*, vol. C-30, pp. 215–222, Mar. 1981.

[6] H. Fujiwara and T. Shimono, "On the acceleration of test generation algorithms," *IEEE Trans. Comput.*, vol. C-32, pp. 1137–1144, 1983.

[7] G. F. Pfister, "The Yorktown Simulation Engine," in *Proc. 19th DAC*, 1982, pp. 31–54.

[8] T. Sasaki, N. Koike, K. Omori, and K. Tomita, "HAL: A block level hardware logic simulator," in *Proc. 20th Design Automat. Conf.*, 1983, pp. 150–156.

[9] T. Blank, "A survey of hardware accelerators used in computer aided design," *IEEE Design Test*, vol. 1, no. 3, pp. 21–39, 1984.

[10] S. Köppe and C. W. Starke, "Logiksimulation komplexer Schaltungen für sehr große Testlängen," NTG-Fachtagung "Groß-integration," 18.-20.3.1985, Baden-Baden, NTG-Fachberichte, Band 87, pp. 73–80.

[11] N. Ishiura, H. Yasuura, T. Kawasata, and S. Yajima, "High-speed logic simulation using a vector processor," in *Proc. VLSI 85 Int. Conf.*, 26.–28.8.1985, Tokyo, pp. 67–76.

[12] J. A. Waicukauski, E. B. Eichelberger, D. Forlenza, E. Lindblom, and T. McCarthy, "Fault simulation for structured VLSI," *VLSI Syst. Design*, pp. 20–32, Dec. 1985.

[13] W. Daehn and M. Geilert, "Fast fault simulation by compiler driven single fault propagation," in *Proc. Int. Test Conf. 1987*, Washington, DC, Sept. 1–3, 1987, pp. 286–292.

[14] J. L. Carter, S. Dennis, V. S. Iyengar, and B. K. Rosen, "ATPG by random pattern simulation" in *Proc. ISCAS '85*, pp. 683–686.

[15] M. H. Schulz, E. Trischler, and T. M. Sarfert, "SOCRATES: A highly efficient automatic test pattern generation system," in *Proc. Int. Test Conf. 1987*, Washington, DC, Sept. 1987, pp. 1016–1027.

[16] M. R. Mercer, V. D. Agrawal, and C. M. Roman, "Test generation for highly sequential scan-testable circuits through logic transformation," in *Proc. Int. Test Conf. 1981*, Philadelphia, PA, Oct. 27–29, 1981, pp. 561–565.

[17] H. Chernoff and L. E. Moses, *Elementary Decision Theory*. New York: Wiley, 1959.

[18] H.-J. Wunderlich, "PROTEST: A tool for probabilistic testability analysis," in *Proc. 22nd Design Automat. Conf.*, 1985, pp. 204–211.

A Modular Fault-Tolerant Binary Tree Architecture with Short Links

Adit D. Singh and Hee Yong Youn

***Abstract*— In this paper, we present a new modular fault-tolerant binary tree architecture which we show to be more effective in overcoming both operational faults as well as fabrication defects than earlier approaches. Furthermore, for practical size trees of up to eight levels, we also show how our design can be efficiently laid out in VLSI with very short interconnections. Thus, our design is very well suited to the monolithic implementation of large binary tree architectures. For board level multichip designs, we present a hybrid scheme, combining our new design with the SOFT approach, that shows better reliability than either design alone.**

***Index Terms*— Binary tree, fault tolerance, layout, modular, reconfiguration, VLSI.**

Manuscript received May 21, 1987; revised September 25, 1989. This work was supported in part by the National Science Foundation under Grants MIP 8808325 and MIP-9009643.

A. D. Singh is with the Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA 01003.

H. Y. Youn was with the Department of Computer Science, University of North Texas, Denton, TX 76203. He is now with the Department of Computer Science Engineering, The University of Texas at Arlington, Arlington, TX 76019.

IEEE Log Number 9042298.

I. Introduction

The VLSI implementation of complete binary tree architectures has been the focus of much research interest in recent years [1]–[5]. Because many projected applications, e.g., dictionary machines [6] and database systems [7], require significantly more silicon area than can be fabricated defect-free with viable yield, several of the proposed designs include a fault tolerance capability, achieved by employing redundant processors, switches, and interconnections. It is expected that this can be used to overcome fabrication defects at manufacture [8]–[11], [18], and perhaps also operational faults in the field.

However, any fault-tolerant design developed to overcome fabrication defects such as an entire binary tree architecture realized on a single large area (even wafer scale) VLSI circuit must satisfy the following two requirements. 1) The fault tolerance mechanisms must make efficient use of the redundant resources so that the benefits of fault tolerance are not lost to additional faults due to significantly increased circuit area. This implies that the design should attempt to minimize the likelihood that available spare elements anywhere in the array are unable to functionally replace failed elements because of interconnect limitations, thereby ensuring a high level of fault tolerance with a minimum number of spares. 2) The restructured interconnections lengths following reconfiguration must be quite short to minimize performance degradation. Since it is not known *a priori* as to which links will have to be restructured to bypass failed nodes, for adequate performance all interconnections must be implemented with powerful driver circuits capable of driving the worst case restructured interconnections with acceptable delay. If restructured interconnections are long, this can impose very substantial area, power, and delay penalties on the design.

While recently proposed designs for tree architectures are becoming increasingly more efficient in their use of redundancy, they have not been as effective in meeting requirement 2. This is because all the fault-tolerant VLSI designs proposed so far employ a layout approach based on the H-tree layout [5]. This layout approach results in varying length links between nodes, as can be seen in Fig. 1, with the root node having the longest connections to its children. Link lengths decrease by half for each successive lower level in the tree. It is clear from Fig. 1 that the link connecting two brother nodes at level 2 required by the SOFT design [13], which is also employed in [14], has a length equal to half the edge of the entire layout. For proposed large area and wafer scale designs, this could easily be a centimeter or more, perhaps as much as 4 or 5 cm. Note that in a typical VLSI technology an interconnection 1 cm long has a capacitance comparable to that for a bonding pad, and requires substantial area, power, and delay for the driving circuits. Even without the increase in interconnection lengths due to restructuring, the H-tree layout is not suitable for such large area designs because of the large lengths (and hence delay) of the upper level links particularly those connecting the root node. This is exactly contrary to the requirement for high bandwidth at the root node of most tree architectures since the root experiences the maximum communication traffic.

In this paper, we present a new modular fault-tolerant binary tree architecture which we show to be more effective in overcoming both operational faults as well as fabrication defects than earlier approaches. Furthermore, for practical size trees of up to eight levels we also show how our design can be efficiently laid out in VLSI with very short interconnections, using the near optimal layout for binary tree architectures [17] we have recently reported. Thus, our design is very well suited to the VLSI implementation of large binary tree architectures. For board level multichip designs, we also present a hybrid scheme, combining our new design with the SOFT approach, that shows better reliability than either design alone.

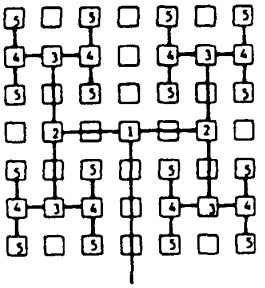

Fig. 1. A five-level tree with H-tree layout.

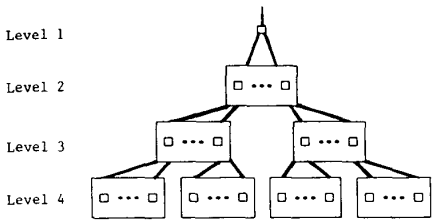

Fig. 2. Proposed fault-tolerant modular tree architecture.

The rest of this paper is organized as follows. In Section II, we present the new modular fault-tolerant architecture, along with a reconfiguration algorithm and switch design to support the reconfiguration. Section III presents area efficient layouts for embedding our modular tree architecture in VLSI. Section IV analyzes the reliability and yield for our designs, and compares them to earlier modular designs. We compare our design to the SOFT and Cluster Proof [14] schemes in Section V. Also in Section V we present a new hybrid design that combines our design with the SOFT scheme to obtain additional reliability improvement in systems with low levels of redundancy. We conclude the paper in Section VI.

II. The Proposed Fault-Tolerant Modular Tree Architecture

Fig. 2 presents the proposed new fault tolerant tree architecture for a four-level modular level. Observe that except for the root node, the design comprises identical modules which are themselves connected as a complete binary tree. (The interconnection between the modules can be regarded as a single link when considering the module tree.) For consistency, we refer to the levels in the module tree beginning at level 2 so that the levels in both the overall computational tree and the module tree are the same. Each module nominally realizes two brother nodes of the overall computational tree and therefore must contain at least two of the processing elements (PE's) that are located at each node of the tree architecture. However, for fault tolerance each module can also contain any number of additional extra spare PE's. These cannot only be used to overcome PE failures in that module, but can also be "lent" as needed to neighboring modules. By the same token, a module can "borrow" an operational spare PE from a neighboring module to meet its own needs, or even to satisfy a loan request from a different neighboring module which cannot be satisfied from its own resources. Thus, a module can sometimes still lend a PE by borrowing from another module even when it itself does not have any unused operational PE's. It is this flexible sharing of redundant PE's that is responsible for the effectiveness of our fault-tolerant design.

While a completely general form of this proposed modular redundancy approach would allow an arbitrary number of spare PE's in

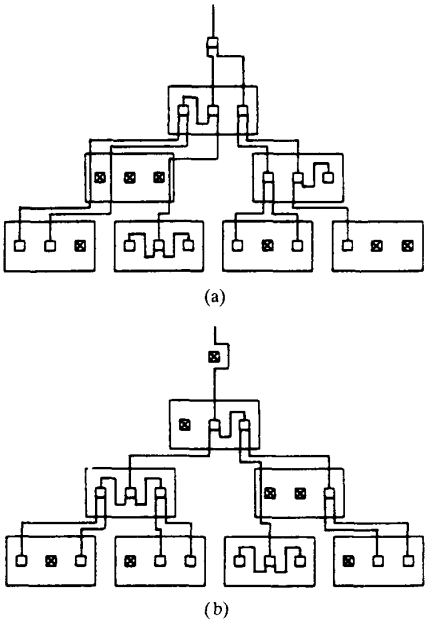

Fig. 3. Two examples of tree reconfiguration with 50% PE redundancy. (Note: Faulty PE's are marked ⊠.) (a) With nonfaulty root PE. (b) With faulty root PE.

TABLE I
POSSIBLE LOGICAL TRANSFERS OF PE'S BETWEEN A MODULE AND ITS THREE ADJACENT MODULES

Adjacent Module	Configurations
parent(p)	lend, borrow, no transfer
left child(lc)	lend, borrow, no transfer
right child(rc)	lend, borrow, no transfer

each module, and any number of PE's to be borrowed or lent between modules, the interconnection to realize such a capability would be prohibitively complex. We therefore limit this logical (virtual) transfer between any two adjacent modules to a single PE. We do, however, allow any number of extra spare PE's to be placed in each module. In Section II-B, we shall present a switch and interconnection structure that can realize this capability with no increase in complexity over the switch designs in [12] and [13].

Table I summarizes the types of logical PE transfers that can take place between a module and its three adjacent neighboring modules. Note again that to limit the complexity of the restructuring interconnect we allow only a single PE to be logically transferred between any two adjacent modules. Table I indicates that a total of $3 \times 3 \times 3 = 27$ different configurations are possible. However, not all these 27 configurations may be meaningful for a given design. For example, the configuration where a module lends a PE to each one of its three neighbors requires that there must be at least five PE's in that module, i.e., a PE redundancy level of 150%. Such a configuration would never be valid for a design that has only a single spare PE (50% PE redundancy) in each module. (Note that a valid configuration requires each module to realize its two brother nodes in the overall computational tree.) While space limitations prevent us from illustrating all the 27 configuration, some typical examples for a design with 50% PE redundancy (1 spare PE in each module) are shown in Fig. 3.

A. Reconfiguration Algorithm

To realize the desired complete binary tree architecture, each module must be configured so that it implements two brother nodes of the overall computational tree as shown in Fig. 2. In general, this will entail the logical lending and borrowing of PE's between adjacent modules. To ensure a successful reconfiguration (if one is possible), this logical transfer of PE between modules must be optimized globally, in the context of the entire tree. We now present a simple algorithm to achieve this.

Algorithm 2.1

1) Begin at $i = l$ (l is the lowest level of the module tree.)
2) At the level i, configure each module to realize its two brother nodes by using as many PE's borrowed from modules at a lower level as possible. (This is to allow one of the module's own PE's to be loaned up the tree if possible. Note that modules at the lowest level in the tree cannot borrow PE's from a lower level.) PE's may be borrowed from a module higher up in the tree only as a last resort.
3) $i = i - 1$
4) Repeat steps 2–4 until the root node is reached. If at any stage a module cannot be configured to realize its two brother nodes we have a failed tree.

Fig. 3 illustrates how the above algorithm optimally configures the modules, band finally the desired binary tree architecture.

B. Switch Structure

Fig. 4 shows a possible switch design for the proposed scheme with one spare PE in each module. Observe that the switch structure is quite regular with respect to the PE's and can be easily extended to allow a higher level of redundancy in the module. Note also that the design averages 11 switches per PE for nonleaf nodes and only three switches per PE for the leaf nodes, giving an overall average of about seven switches per PE. This is about the same complexity as the switch structures in [12] and the SOFT [13] design.

Fig. 4(c) shows how the configurations in Fig. 3(a) can be realized by this switch structure, with active links shown in bold to highlight the reconfigured interconnections. Switch settings to realize all the valid configuration for a three PE module with 50% module redundancy are shown in Table II.

III. VLSI LAYOUT

We have argued in the introduction that it is critical for large area fault-tolerant tree architectures to have short interconnections, and therefore a layout strategy different from the H-tree design. In this section, we present area efficient layouts for embedding modular tree architectures with up to 127 nodes (seven levels) in a planar two-dimensional array of modules with very short (minimal length) interconnections. We further prove that trees larger than seven levels cannot be embedded without increasing the maximum interconnection length.

Fig. 5 shows our layouts for trees of five, six, and seven levels [17]. A five-level tree can be obtained by combining two four-level trees as shown in Fig. 5(a). Observe that except for two nodes in the five-level tree, any two directly connected nodes are physical neighbors (including diagonal neighbors) in our layout. We do not distinguish between the lengths of adjacent and diagonal connections, taking them both to be of unit length. This is because in practice the actual lengths will depend on where the input and output ports are located on the node layouts; the two interconnections will generally be of comparable lengths. Note also that our layouts are highly area efficient in the sense that almost all the area is utilized by active

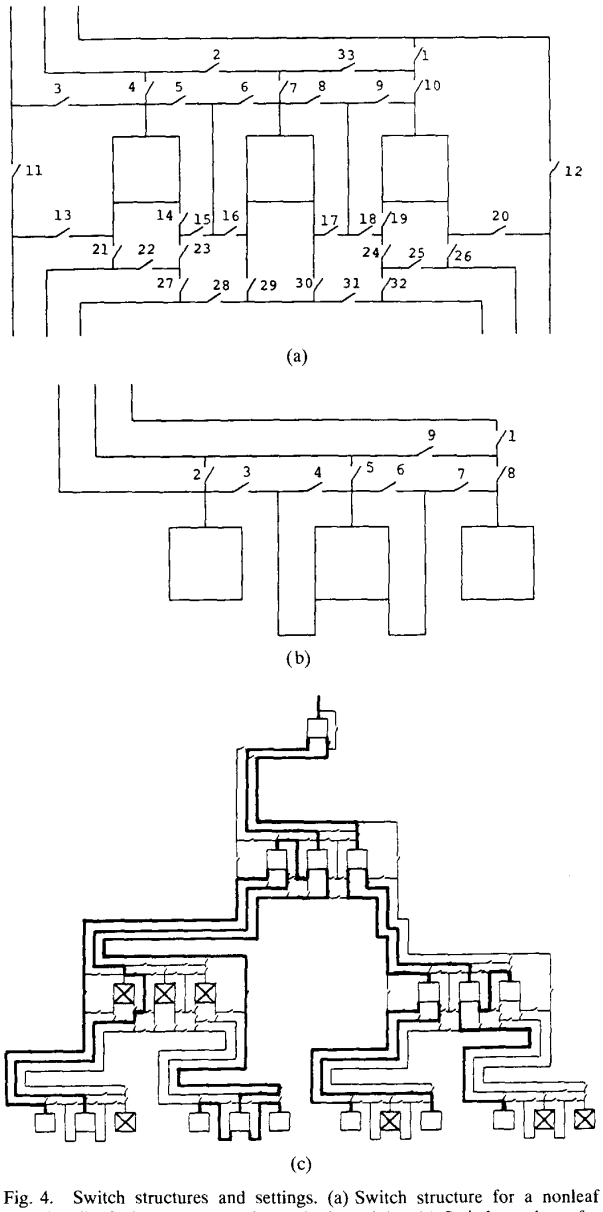

Fig. 4. Switch structures and settings. (a) Switch structure for a nonleaf module. (b) Switch structure for a leaf module. (c) Switch settings for Fig. 3(a).

modules (in all cases over 95%). In contrast, only about 50% of the modules in a H-tree layout are active.

We now prove that trees with more than seven levels cannot be embedded with unit length interconnections.

Theorem 3.1: At most a seven-level tree can be embedded in a two-dimensional rectangular array such that any two directly connected nodes in the tree are adjacent or diagonal neighbors.

Proof: Consider the embedding of an eight-level tree in a square array. Since there are $2^8 - 1 = 255$ tree nodes, the square array must be at least 16 units on a side as shown in Fig. 6. Also, all but one of the modules in the figure must be utilized as tree nodes. Suppose that the root node is in the middle of the array, at position 1 in Fig. 6. Then the distance from the root to the leaf node located in position 2 or position 3 must be at least eight units. But an eight-level tree with only unit length interconnects has a root to leaf distance of only seven units. Thus, an eight-level tree cannot be embedded with unit length interconnects with the root in position 1. It is readily seen that moving the root node away from the middle of the array only increases the worst case root to leaf distance. The same holds if a (nonsquare) rectangular array is used for the embedding. We therefore conclude that an eight-level tree cannot be embedded in a rectangular array with unit length interconnects.

Since the number of nodes in a binary tree increases as 2^l with the number of levels l, while the number of nodes in square array increases only as n^2 with edge dimension n, it is obvious that larger trees also cannot be embedded with unit length interconnections. Q.E.D.

While the above theorem suggests that very large tree architectures must necessarily have longer interconnects, in a practical sense, at the current state of technology its impact is minimal. Because of the complexity of the restructuring switch structure for all fault-tolerant designs, a defect-tolerant VLSI implementation of such architectures only makes sense if the nodes are relatively large. Node sizes in the range 0.05–0.5 square centimeter have been discussed in the literature [12], [14]. Even at the low end of this range, a square centimeter of silicon will only hold 20 PE's, not even allowing for any interconnect area. Because we propose implementing a tree of modules as shown in Fig. 2, and each module contains two tree nodes (plus spares), a seven-level module tree will realize a computational tree with eight levels (255 nodes). Including the spares, a 50% redundant design will entail implementing almost 400 PE's on a wafer. Relatively large size trees up to 12 level can also be efficiently laid out using a scheme reported in [17]. This scheme minimizes the maximum edge length, while utilizing all PE's in the array.

IV. Reliability and Yield Analysis

The modular fault-tolerant design presented in this paper is aimed at two distinct applications: 1) enhancing the reliability of conventional binary tree architectures, and 2) improving yield for single chip VLSI implementations of binary trees. We now evaluate the effectiveness of our design for these applications. Throughout this section we shall assume that the failure rate for links and switches is negligible compared to the PE's. Thus, only PE failures are considered. While this is an optimistic assumption, it greatly simplifies the analysis. This assumption has also been made in the analysis of earlier designs [3], [7], [12]–[14]. It can be argued that if the PE's themselves are relatively complex, then the failure probabilities for the links will indeed be small. Moreover, many link failures can be modeled as failures in the connected PE's.

A. Reliability Evaluation

We analyze the reliability of our fault-tolerant tree structure by considering the reconfiguration process and evaluating the probability of obtaining a fully configured fault-free computational tree. Let $R_p = e^{-\lambda t}$ be the reliability function for individual PE's in the modules. Consider Fig. 2 again. The reconfiguration algorithm in the previous section begins with modules at the bottom of the tree and works its way up to the root node. Whenever possible, the modules are configured by borrowing PE's from lower down in the tree. Only if a module cannot be configured in this way, a PE from a module higher up in the tree used.

Let us consider the possible logical transfer of a PE between an arbitrary module and the module above it *after the lower module has been configured.* Information regarding whether or not the lower module can lend a PE, or needs to borrow a PE, and the success of the configuration process so far can be represented in four states.

TABLE II
SWITCH SETTINGS FOR VALID CONFIGURATION WITH ONE SPARE PE IN A MODULE.

No. of good PEs	Configuration: lend to	Configuration: borrow from	Switches to be closed
3	–	–	(4,21,14,23,27)(1,10,19,24,32,26)
	p	–	(2,7,16,5,17,9)(21,14,23,27)(19,24,32,26)
	lc	–	(2,7,16,5,30,28)(13,14,23,22)(1,10,19,24,32,26)
	lc,rc	p	(11)(4,21,14,23,27)(1,33,7,29,31,17,9)(19,24,25,20)
	p,lc	rc	(1,10,19,18,8,20)(16,5,30,28)(13,14,23,22)
2	–	–	(3,21,14,23,27)(2,7,29,31,17,18,24,25)
	p	lc	(4,13,14,15,6)(29,31,17,18,24,25)
	lc	p	(11)(4,21,14,23,27)(1,33,7,29,31,17,18,24,25)
	lc	rc	(2,7,16,5,30,28)(13,14,23,22)(12)
1	p	lc,rc	(2,7,16,15,23,22,17,18,24,25)
	lc	p,rc	(11)(2,7,16,15,23,22,30,28)(12)
	–	p	(3,5,6,16,15,23,22,30,28)(12,2,7,8,18,24,25)
	–	lc	(11)(2,7,29,31,17,18,24,25)
0	–	lc,rc	(11)(12)
	–	p,lc	(11)(12,2,7,8,18,24,25)

(Note: Only one of any pair of symmetric configurations involving left and right child modules is shown.)

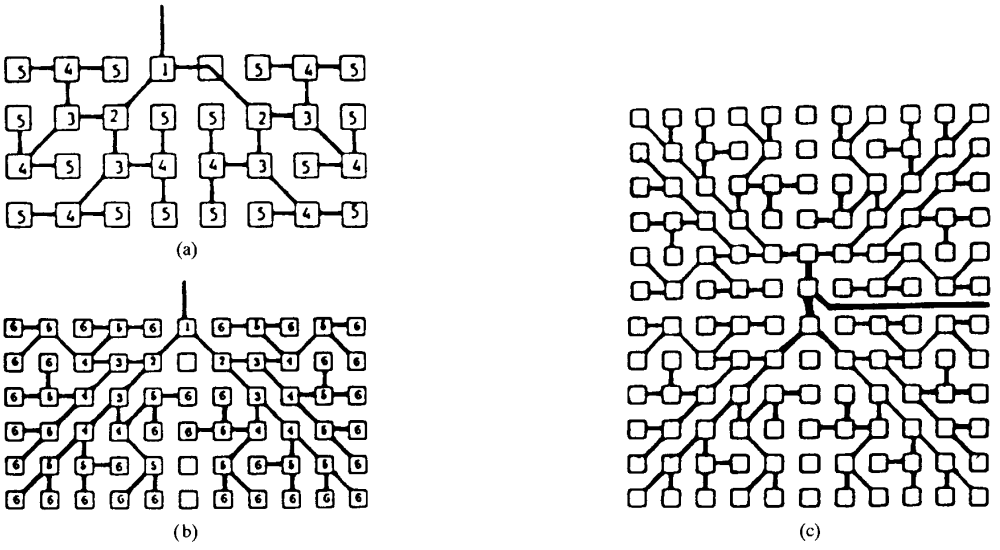

Fig. 5. Proposed compact layout. (a) Five-level tree. (b) Six-level tree. (c) Seven-level tree.

State 1: The module can lend a PE
State 2: The module cannot lend but does not need to borrow a PE
State 3: The module needs to borrow a PE
State 4: Failed tree because the module or one below it in the module tree could not be successfully configured.

Let the probability of an arbitrary module at level i being in State j be $P_j^{(i)}$, for $j \in \{1,2,3,4\}$. Then using simple combinational arguments, we can develop the following recurrence equations for these state probabilities one level higher in the tree:

$$P_1^{(i-1)} = R_p^3\left\{\left(P_2^{(i)}\right)^2 + 2P_1^{(i)}P_3^{(i)} + 2P_1^{(i)}P_2^{(i)} + \left(P_1^{(i)}\right)^2\right\}$$

(3 good PE's in module plus 0, 1 or 2 borrowed from below.)

$$+ 3R_p^2(1-R_p)\left\{2P_1^{(i)}P_2^{(i)} + \left(P_1^{(i)}\right)^2\right\}$$

(2 good PE's in module plus 1 or 2 borrowed from below.)

$$+ 3R_p(1-R_p)^2\left(P_1^{(i)}\right)^2$$

(1 good PE's in module plus 2 borrowed from below.)

It can be similarly seen that

$$P_2^{(i-1)} = R_p^3\left(2P_2^{(i)}P_3^{(i)}\right) + 3R_p^2(1-R_p) \cdot \left\{\left(P_2^{(i)}\right)^2 + 2P_1^{(i)}P_2^{(i)}\right\} + 3R_p(1-R_p)^2 \cdot \left(2P_1^{(i)}P_2^{(i)}\right) + (1-R_p)^3\left(P_1^{i}\right)^2$$

$$P_3^{(i-1)} = R_p^3\left(P_3^{(i)}\right)^2 + 3R_p^2(1-R_p)\left(2P_2^{(i)}P_3^{(i)}\right) + 3R_p(1-R_p)^2\left\{\left(P_2^{(i)}\right)^2 + 2P_1^{(i)}P_3^{(i)}\right\} + (1-R_p)^3\left(2P_1^{(i)}P_2^{(i)}\right). \quad (14)$$

Also, because in a l level tree there are no modules below level l, we obtain

$$P_1^{(l+1)} = 0, \qquad P_2^{(l+1)} = 1, \qquad P_3^{(l+1)} = 0$$

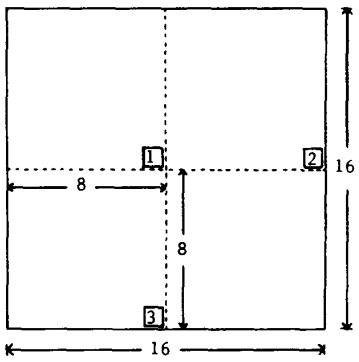

Fig. 6. Figure for Theorem 3.1.

$$\text{and} \quad P_4^{(l+1)} = 0.$$

Using the above equations we can recursively evaluate the state of a module at each level in the tree, until the root node in the module tree (level 2) is reached. (Recall that the root node in the module tree is at level 2 in Fig. 2). Then

$$R_{\text{sys}} = R_p\left(P_1^{(2)} + P_2^{(2)}\right) + (1 - R_p)P_1^{(2)}.$$

This equation can be used to compute system reliability for the proposed scheme with 50% redundant PE's. Some plots for five and eight-level binary trees with $\lambda = 0.01$ are shown in Fig. 7. Using the same approach, system reliability for fault-tolerant trees with other levels of redundancy can also be computed. We shall present some of these results later in the yield analysis section.

The plots in Fig. 7 also show the reliability for the modular scheme in [12] (referred to here as the old modular scheme) which was shown to be better than all earlier designs except for the SOFT design. It is clear that our new modular scheme is significantly better. Unfortunately we cannot plot the reliability of the SOFT scheme here for a direct comparison because no generalized analytical means for evaluating it has been developed yet. We shall, however, compare the two schemes in detail later in Section V.

B. Yield Analysis

We next evaluate the effectiveness of the proposed fault-tolerant tree architecture in enhancing the yield of a large VLSI tree implementation. We assume here that the restructuring switches can be reliably set using some electrical or laser programming technology [15], [16].

Let p_s be the PE survival probability, i.e., the probability is that a PE is failure-free following manufacture. For analytical tractability we assume that PE failures are independent. This implies a Poisson defect distribution on the wafer, although in practice, some clustering of defects is observed which could make our analysis somewhat optimistic. Setting $p_s = R_p$ we can directly obtain the tree yield (probability that the computational tree is failure free) using the reliability analysis presented earlier in this section. Results for five and eight-level trees are plotted in Fig. 8. Fig. 8 also shows yields for the old modular scheme in [12], and for the proposed scheme with 100% PE redundancy. These latter results were obtained by directly extending the analysis in this section.

While the plots in Fig. 8 indicate significantly better tree yields for our designs (in fact the new scheme with 50% redundancy outperforms the old modular scheme with 100% redundancy), they do not reflect the true silicon yield because they are not normalized with respect to the extra area required by the redundant designs. To make

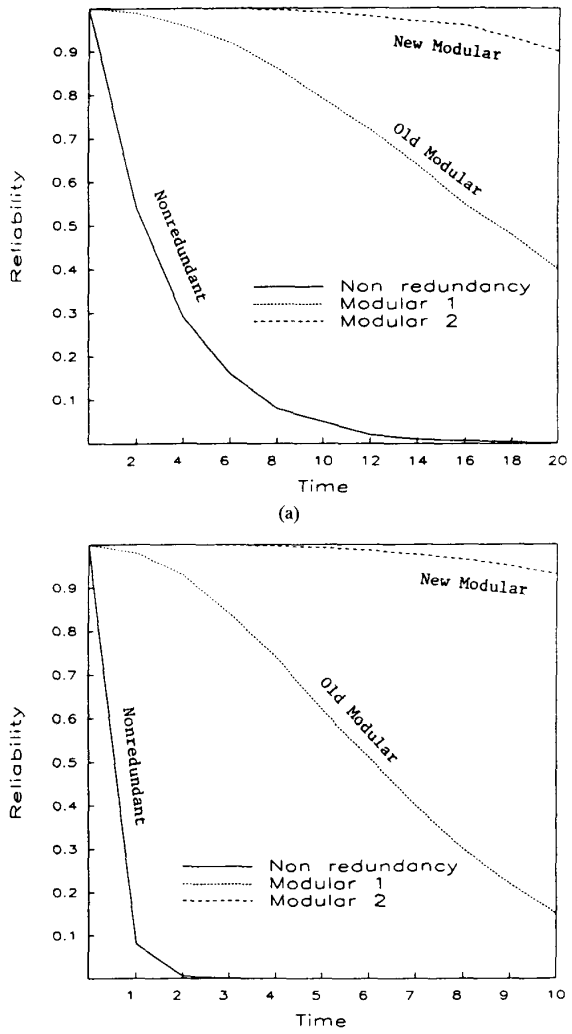

Fig. 7. Reliability plots for $\lambda = 0.01$: (a) Five-level tree. (b) Eight-level tree.

a more meaningful comparison, we use the silicon area utilization factor (SAUF), defined in [9] to be

$$\text{SAUF} = \frac{\text{Number of PE's in the computational tree} \times \text{tree yield}}{\text{expected number of good PE's on the chip}}.$$

This factor, sometimes also called *harvest*, measures the fraction of failure free PE's that are actually utilized (on the average) in implementing the array on silicon. Clearly it measures the efficiency of silicon utilization. In considering SAUF it is important to recognize that for large structures with a significant PE failure probability, SAUF cannot be expected to be very high. Because of interconnection limitations, such structures cannot always utilize all failure free PE's. Also, since a tree of fixed size is desired, if there are too few good PE's on a given chip due to statistical variations, the whole chip must be discarded. These factors lead to SAUF inefficiencies.

SAUF plots for the proposed design for five-level (31 nodes) and eight-level (255 nodes) trees with 50% and 100% PE redundancy are shown in Fig. 9. As expected, the new modular scheme is much better for all values of p_s, the PE survival probability. For the

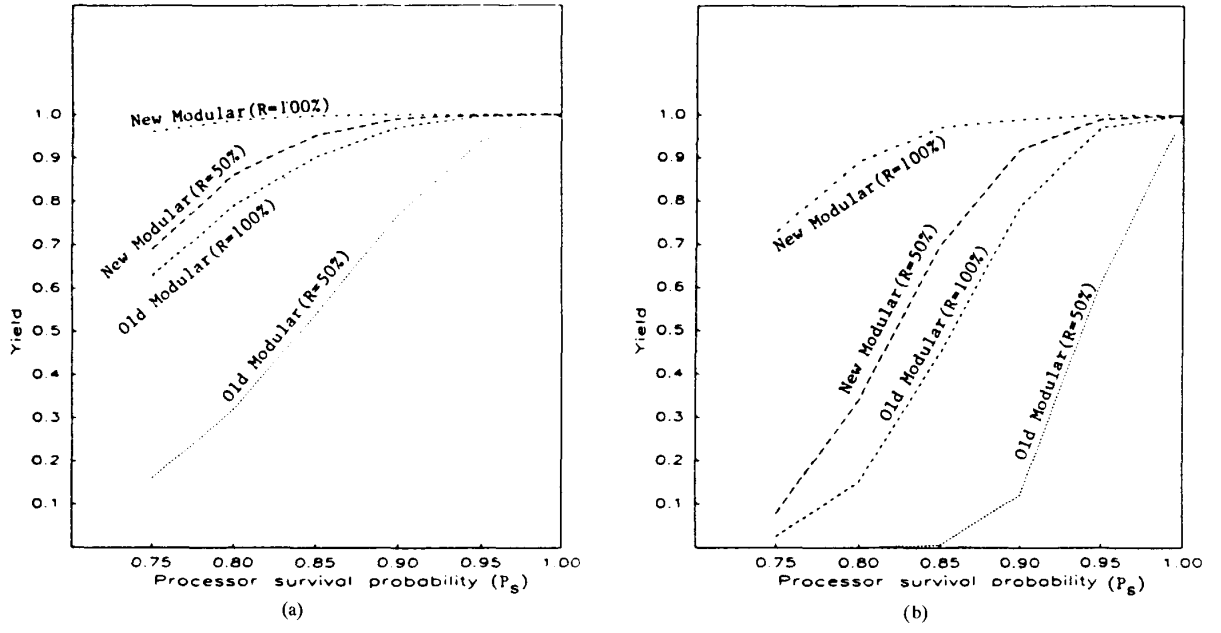

Fig. 8. Yield plots (R = Redundancy). (a) Five-level tree. (b) Eight-level tree.

smaller 31 node tree in Fig. 9(a), a SAUF of better than 50% can be achieved by using the optimum level of redundancy even for relatively large PE's with p_s down to about 0.6. The larger tree with 255 computational nodes (and additional redundant PE's) can be expected to have smaller PE's, and here a SAUF of better than 50% can be achieved for p_s down to 0.75. Note that a multichip design would require additional area for bonding pads and off-chip driver circuits. Thus, our single chip (wafer) implementations are almost as efficient in silicon utilization as multichip designs, while providing substantial size, power, and performance benefits.

V. Comparison to Other Designs

The modular scheme that we have presented in this paper is clearly a complete generalization of earlier modular designs [3], [12]. The improvement in system reliability, yield, and SAUF when compared to these earlier designs is very substantial as was seen in Figs. 7–9. We first compare our design to SOFT [13] scheme.

A. The SOFT Scheme

It is more interesting to observe that our scheme can also, in a sense, be considered to be a generalization of the highly effective but nonmodular SOFT design recently proposed by Lawrie and Fuchs [13]. To understand this, we briefly review the SOFT scheme which is illustrated with 25% PE redundancy in Fig. 10(a). In this design, brother nodes are connected together at all levels of the tree. The spare PE's are connected to leaf nodes from adjacent subtrees. Reconfiguration is achieved by shifting up one of the children of the failed node to replace it. The displaced child is in turn replaced by one of its own children, and this shifting up of nodes is propagated down all the way to a leaf node that has a spare to replace it.

Fig. 10(b) illustrates how this SOFT scheme can be interpreted to be a modular design with spare PE's located between the leaf modules. However, because of the interconnection limitations of the SOFT design, each module can only borrow PE's from the modules below it, and lend a PE to the module above it. The SOFT design does not either allow a module to borrow spares from a module above it in the module tree or to lend PE's to modules below it as our design does. In our design, we allow each module to both lend and borrow from all three adjacent modules.

It is generally expected that for moderate-sized PE's, a redundancy level of 50–100% may be required to optimize silicon utilization and provide the highest effective yield. However, the SOFT approach does not readily lend itself to redundancy levels of greater than 25%. When more than 25% redundancy is required, our design can thus more effectively restructure the tree architecture by putting the required number of spare PE's in each module and efficiently sharing them between adjacent modules. Thus, the advantages of using the new modular fault-tolerant design presented in this paper over the SOFT scheme to overcome fabrication defects in large area VLSI tree architectures are substantial. In addition to a higher expected yield, we have shown that the new design (for practical sized trees) can be laid out with short, minimum length interconnection which will result in very significant area, power, and performance benefits. Being nonmodular, the SOFT design cannot as effectively employ our layout approach.

The advantages of our new design over the SOFT design for operational reliability improvement for a board level (multichip) design are less clear cut. In such designs, a relatively low level of redundancy is generally used because many simultaneous failures are not expected. While our design does display a better yield for implementing large tree architecture in a single chip, locating spares in subtrees as in the SOFT design appears more natural when the level of redundancy is small. For such designs we propose a hybrid scheme combining our design to the SOFT approach to obtain a new design with even better reliability than either scheme alone.

B. A New Hybrid Scheme

Fig. 11 illustrates the hybrid scheme for a five-level tree with four spares. This has switches in the modules to allow the more generalized borrowing as in our modular scheme, but has no redundancy in

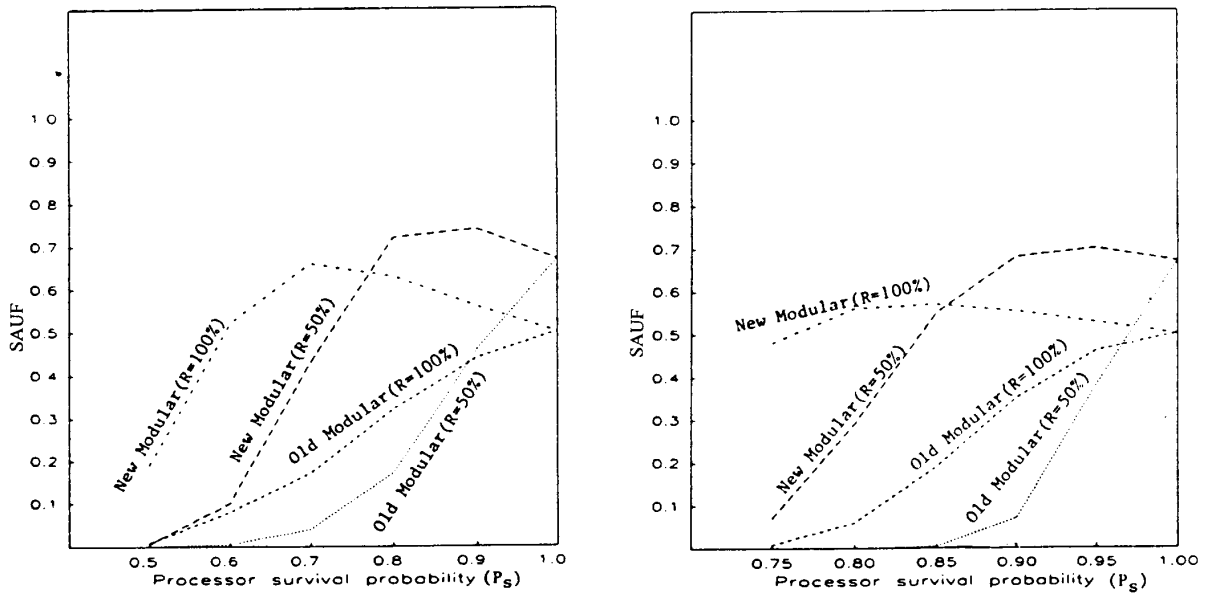

Fig. 9. SAUF plots (R = Redundancy). (a) Five-level tree. (b) Eight-level tree.

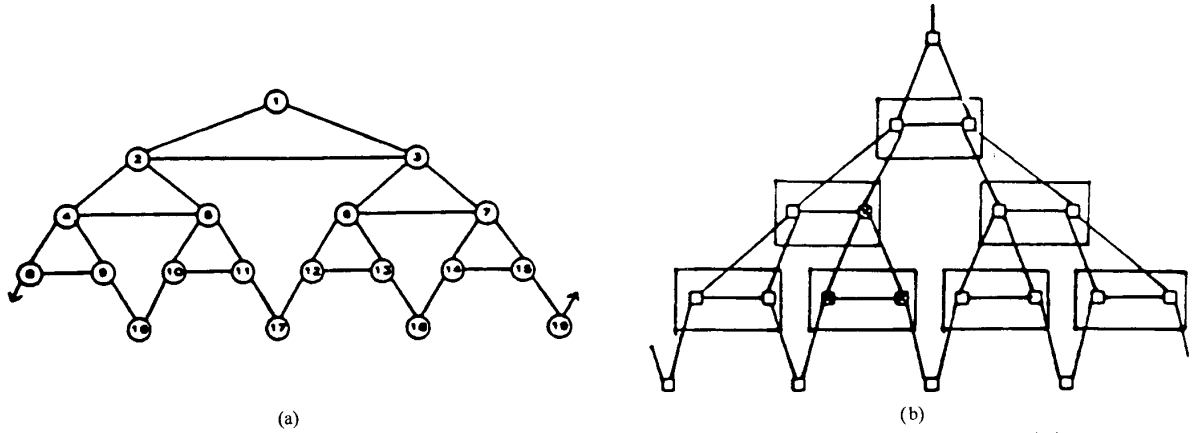

Fig. 10. Comparison to SOFT scheme. (a) SOFT scheme with 25% redundancy. (b) SOFT scheme as a modular design.

the modules. The spare PE's are connected to the leaf modules as in the SOFT scheme. In addition, adjacent leaf modules from different subtrees that do not share a spare are also directly connected together, again as in the SOFT design.

That such a hybrid design displays better reliability than either design individually can be seen by considering the left subtree partition as shown in the Fig. 11. In the SOFT design at most three faults can be tolerated in the left subtree if all the spares are good. This is because even if there are no faults in the right subtree, there is no mechanism available to use spare 4 in the left subtree. Similarly, in a purely modular design using the scheme presented in this paper, the left subtree is also limited to three spares. This is because in such a design spares would be located in the leaf modules and would not be directly shared between the leaf modules. The left subtree would receive its share of two out of the four spares distributed in some of its leaf modules, and could use only one of the spares in the right subtree by "propagating" it through the common module at level 2. The hybrid scheme allows both these mechanisms for sharing spares between the two subtrees, and thus allows the subtrees to share all spares instead of the limited ones by either scheme individually. When the total number of spare PE's in the system is small, such as in Fig. 11, then this additional sharing of spares is likely to have a significant impact on reliability. On the other hand when the level of redundancy in each subtree is high, as in defect tolerant VLSI designs with 50–100% redundancy, the additional sharing of one more PE's between subtrees will only minimally impact yield. For such implementations the modular design with its better layout characteristics will generally be superior.

C. The Cluster Proof Scheme

In concluding this section we must also mention the Cluster Proof scheme proposed in [14]. This design combines a SOFT scheme in the upper half of the tree with a fully connected replacement capability from a pool of spares in the lower subtrees. Because of this full

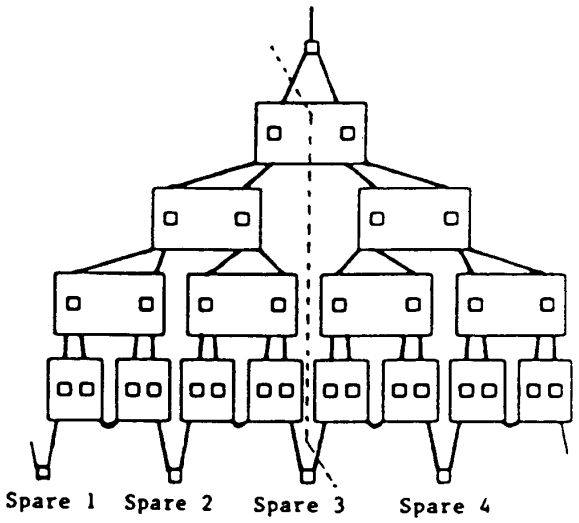

Fig. 11. Hybrid scheme.

replacement capability, the reliability and yield for this Cluster Proof scheme can be expected to be better than the SOFT design. However, the restructured interconnection lengths in this design including the lower subtree are relatively long and the switch structure is more complex. In applications where a somewhat lower bandwidth may be acceptable in the lower half of the tree architecture, the Cluster Proof approach combined with our new modular scheme for the upper half tree may offer a viable design.

VI. Concluding Remarks

We have presented a modular fault-tolerant architecture for complete binary trees that is particularly well suited for large area VLSI implementation. Our design displays the best yield enhancement of all known designs, and for the first time allows a VLSI layout with short interconnections, critical for large area designs. In conjunction with a laser programming technique such as that developed at MIT Lincoln Labs [15], we believe that large binary tree VLSI architecture can now be successfully realized. However, for reliability improvement in board level multichip designs with relatively low levels of redundancy, we suggest a new hybrid scheme, combining our new design with the recently proposed SOFT approach, that shows better reliability than either scheme alone.

References

[1] A. Despain and D. Patterson, "X-tree: A tree structured multiprocessor computer architecture," in *Proc. 5th Annu. Symp. Comput. Architecture,* Apr. 1978, pp. 144–151.

[2] D. Gordon, I. Koren, and G. M. Silberman, "Embedding tree structures in fault tolerant VLSI hexagonal arrays," *IEEE Trans. Comput.,* vol. C-33, no. 1, pp. 104–108, Jan. 1984.

[3] A. S. M. Hassan and V. K. Agarwal, "A fault tolerant modular architecture for binary trees," *IEEE Trans. Comput.,* vol. C-35, no. 4, pp. 356–361, Apr. 1986.

[4] J. P. Hayes, "A graph model for fault tolerant computing systems," *IEEE Trans. Comput.,* vol. 25, no. 9, pp. 875–884, Sept. 1976.

[5] E. Horowitz and A. Zorat, "The binary tree as an interconnection network: Applications to multiprocessor systems, and VLSI," *IEEE Trans. Comput.,* vol. C-30, no. 4, pp. 247–253, Apr. 1981.

[6] T. A. Ottman, A. L. Rosenberg, and L. J. Stockmeyer, "A dictionary machine (for VLSI)" *IEEE Trans. Comput.,* vol. C-31, pp. 892–898, Sept. 1982.

[7] M. A. Bonncelli *et al.,* "A VLSI tree machine for relational data bases," in *Proc. 10th Annu. Symp. Comput. Architecture,* June 1983, pp. 67–73.

[8] A. Rosenberg, "The Diogenes approach to testable fault tolerant networks of processors," *IEEE Trans. Comput.,* vol. C-32, pp. 902–910, Oct. 1983.

[9] A. D. Singh, "Intersitial redundancy: An area efficient fault tolerance scheme for large area VLSI processor arrays," *IEEE Trans. Comput.,* vol. 37, pp. 1398–1410, Nov. 1988.

[10] J. D. Ullmann, *Computational Aspects of VLSI.* Baltimore, MD: Computer Science Press, 1984.

[11] C. S. Raghavendra, A. Avizienis, and M. Ercegovac, "Fault-tolerance in binary tree architectures," *IEEE Trans. Comput.,* vol. C-33, pp. 568–572, June 1984.

[12] A. D. Singh, "A reconfigurable modular fault tolerant binary tree architecture," in *Proc. 15th Annu. Symp. Fault Tolerant Comput.,* June 1987, pp. 298–303.

[13] M. B. Lowrie and W. K. Fuchs, "Reconfigurable tree architectures using subtree oriented fault tolerance," *IEEE Trans. Comput.,* vol. C-36, pp. 1172–1182, Oct. 1987.

[14] M. C. Howells and V. K. Agarwal, "Yield and reliability enhancement of large area binary tree architectures," in *Proc. 15th Annu. Symp. Fault Tolerant Comput.,* June 1987, pp. 290–295.

[15] J. I. Raffel *et al.,* "A wafer-scale digital integrator using restructurable VLSI," *IEEE J. Solid-State Circuits,* vol. SC-20, pp. 399–406, Feb. 1985.

[16] D. C. Shaver, "Electron-beam customization, repair, and testing of wafer-scale circuits," *Solid State Technol.,* pp. 135–139, Feb. 1984.

[17] H. Y. Youn and A. D. Singh, "On implementing large binary tree architectures in VLSI and WSI," *IEEE Trans. Comput.,* vol. C-38, pp. 526–537, Apr. 1989.

[18] ——, "An efficient channel routing algorithm for defective arrays," in *Proc. Int. Conf. Comput.-Aided Design,* Nov. 1989, pp. 432–435.