# Generation of Minimal Vertex Covers for Row/Column Allocation in Self-Repairable Arrays

Michael D. Smith, *Member, IEEE*,
and Pinaki Mazumder, *Senior Member, IEEE*

***Abstract*—This paper lays foundations for an approach to on-chip row/column allocation that exploits certain properties offered by laterally connected networks of simple threshold devices. As a sample application, it is demonstrated how electronic implementations of these networks can be used as the basis for effective memory array repair systems that require little hardware overhead.

***Index Terms***—Self-repair, redundant memory, embedded memory, neural network, vertex cover problem.

## 1 Introduction

The decade of the 1980s saw phenomenal advances in VLSI circuit technology, as shrinking feature width and increasing die size allowed unprecedented levels of integration. These advances, unfortunately, also rendered fabrication processes more susceptible to impurity-related manufacturing defects, even as greater circuit complexity left many subsystems "embedded," where they cannot be observed externally or controlled directly. As prospects for further technological development reach previously unthinkable levels, it is becoming necessary to develop *built-in* systems which can automatically repair partially faulty integrated circuits.

This paper lays theoretical foundations for an approach to on-chip spare allocation in regular VLSI structures such as memories. In high-density DRAM chips, manufacturers commonly include spare rows and columns of memory cells so that rows and columns in which faulty cells are detected can be replaced. Repair by the replacement of rows and columns of memory cells, rather than by replacement of individual cells or entire memory sub-arrays, offers a good compromise between simplicity of reconfiguration hardware and efficient utilization of spare memory. This reconfiguration strategy, however, gives rise to a difficult optimization problem. Given a scattering of faulty cells in a two-dimensional rectangular array, determining whether the array can be repaired using a limited number of spare rows and columns is NP-complete [1].

The problem of efficient spare row and column allocation in redundant memories has been widely studied [1], [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], and a variety of applicable algorithms are present in the literature. To be considered viable for the purpose of on-chip spare allocation, however, an algorithm must satisfy two challenging and mutually conflicting design requirements. First, it must lend itself to easy implementation using very little hardware, incurring a minimal penalty in terms of silicon area overhead, and second, the algorithm must be capable of rapidly generating high-quality solutions to spare allocation problem instances. Previously reported algorithms for spare row and column allocation, most of which were designed for incorporation into dedicated repair systems on integrated circuit production lines, generally fail to meet one or both of these criteria. In contrast to these existing methods of spare allocation, the approach presented in this paper offers the combined advantages of high execution speed, simplicity of implementation, and high rates of successful spare allocation in problem instances for which solutions exist.

Section 2 of this paper begins the theoretical discussion by describing a graph-theoretic model for the problem of array repair through row and column replacement, relating the problem to that of finding a constrained vertex cover in an undirected bipartite graph. Some relevant properties of laterally connected threshold devices are reviewed briefly in Section 3. Section 4 introduces a class of threshold device networks which is proven applicable to a generalized form of the vertex cover problem, and Section 5 concludes the theoretical discussion with an examination of certain convergence issues. Section 6 demonstrates how electronic implementations of the proposed networks can be applied to the problem of spare allocation in embedded memories. Simulation results indicate that these networks can be made to provide consistently feasible solutions to the computationally intractable spare allocation problem.

## 2 Graph-Theoretic Definitions and Model

We begin by defining some common terms from graph theory, regarding the vertex covers of undirected graphs.

Definition 1. *Given an undirected graph $G$ consisting of a set of vertices $V$ and a set of edges $E$, i.e., $G = (V, E)$, a subset $V_c$ of $V$ is called a* vertex cover *of $G$ iff every edge in $E$ has at least one endpoint in $V_c$.*

Definition 2. *Given some subset $R$ of $V$, a vertex cover $V_c$ of $G$ is said to be* minimal *with respect to $R$ iff there exists no vertex $i$ in $R \cap V_c$ such that $V_c - \{i\}$ is a vertex cover of $G$.*

The problem of finding a vertex cover whose cardinality is less than some specified constant is known as the vertex cover problem, and a vertex cover which contains the least number of vertices necessary to cover a given graph is considered optimal. For the purposes of this paper it will prove useful to define a new term, the generalized vertex cover (GVC) problem, as follows.

Definition 3. *Let $G = (V, E)$ be an undirected graph, and let $R_1, R_2, \ldots, R_D$ be disjoint subsets of its vertices. Given integer constants $C_1, C_2, \ldots, C_D$ we define the* generalized vertex cover problem *for $G$ with respect to $R_1, R_2, \ldots, R_D$ to be that of finding a vertex cover $V_c$ of $G$ such that the cardinality of $R_i \cap V_c$ is less than $C_i$ for $1 \le i \le D$.*

An optimal solution to the generalized vertex cover problem is a vertex cover with the property that

$$\sum_{i+1}^{D} \left| R_i \cap V_c \right|$$

is a minimum, where $|R_i \cap V_c|$ denotes the cardinality of $R_i \cap V_c$.

Finally, we define the connection matrix associated with an undirected graph, a construction which will prove useful in later analyses.

Definition 4. *Given an undirected graph $G = (V, E)$ in which $V$ has cardinality $N$, let the* connection matrix **C** *associated with $G$ be the $N$-by-$N$ matrix whose element $c_{ij}$ equals 1 iff an edge connects vertices $i$ and $j$, i.e., iff $e(i, j) \in E$, or equals 0 otherwise.*

It is assumed that the $N$ vertices of $G$ are numbered 1 through $N$ in some arbitrary manner. Note that, by definition, the connection matrix associated with an undirected graph must be symmetric. Furthermore, if no edge in $G$ connects any vertex with itself, then the graph is said to possess no self-loops and the main diagonal of its associated connection matrix consists entirely of 0s.

• *The authors are with the Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122. E-mail: mazum@eecs.umich.edu.*

*Manuscript received Aug. 23, 1993; revised Feb. 8, 1995.*
*For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number C95137.*

An instance of the spare row/column allocation problem consists of a two-dimensional rectangular array of cells, some of which are designated faulty, and two integers providing, respectively, the number of spare rows and the number of spare columns available with which to make repairs. Any subset of the rows and columns in such an array constitutes a *spare allocation scheme*. In the context of memory stuck-at fault repair, we define a spare allocation scheme to be *valid* if and only if its constituent rows and columns, taken together, contain every faulty cell in the array at hand; a valid spare allocation scheme is said to be *minimal* if and only if there does not exist any proper subset of its constituent rows and columns which is itself valid. We define a valid spare allocation scheme to be *feasible* if and only if the number of designated rows and the number of designated columns each obey their respective upper bounds as set forth under the particular problem instance at hand. Finally, we consider *optimal* any feasible spare allocation scheme which designates for replacement the minimum total number of rows and columns.

Fig. 1 illustrates these concepts. Assuming there exist four spare rows and four spare columns with which to make repairs, the diagram in Fig. 1a represents an instance of the spare allocation problem. Fig. 1b indicates those rows and columns designated for replacement under one possible spare allocation scheme. This solution is valid, minimal, feasible, and optimal with respect to the simple problem instance at hand.

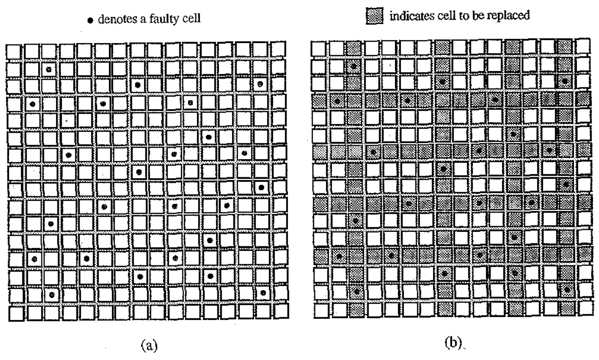


Fig. 1. A faulty array and a valid spare allocation scheme.

The problem of spare row/column allocation in a two-dimensional rectangular array can be modeled as that of computing a constrained vertex cover in an undirected bipartite graph [1]. Consider an array of $M_1$ by $M_2$ cells, containing a defect pattern in which faults occur in $m_1$ or fewer distinct rows and $m_2$ or fewer distinct columns, where $m_1 \leq M_1$ and $m_2 \leq M_2$. If a graph on $m_1 + m_2$ vertices is constructed such that each row and each column of the array corresponds to exactly one vertex, and such that an edge connects vertices $i$ and $j$ if and only if a faulty cell lies at the intersection of the corresponding row and column, then it is easily proven that valid spare allocation schemes for the faulty array exhibit a one-to-one correspondence with vertex covers of the associated graph. Applying the graph-theoretic definition provided earlier, furthermore, it is clear that an instance of the spare allocation problem is easily represented as a special case of the generalized vertex cover problem, in which $D = 2$, the subset $R_1$ consists of all vertices corresponding to array rows, $R_2$ consists of all vertices corresponding to array columns, and the constants $C_1$ and $C_2$ are determined by the number of available spare rows and spare columns, respectively. It is worth mentioning that this graph-theoretic model, and hence the results described in this paper, apply only to arrays of two dimensions.

As an example, consider the faulty 1,024-by-1,024 array illustrated in Fig. 2a. Fig. 2b depicts an associated bipartite graph in which vertices 1 through 4 correspond, respectively, to rows 42, 118, 629, and 823 of the array, and vertices 5 through 8 correspond, respectively, to columns 37, 125, 225, and 921. Every vertex cover of this graph can be mapped to a valid spare allocation scheme for the original array.

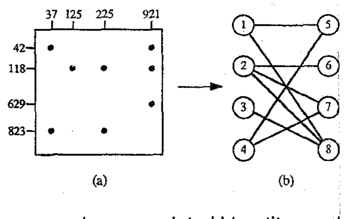


Fig. 2. A faulty array and an associated bipartite graph.

## 3 Relevant Properties of Laterally Connected Threshold Devices

A threshold device network of the form considered throughout this paper consists of highly interconnected simple processing elements, which are numbered 1 through $N$ in some arbitrary manner. The current state $s_i$ of threshold device $i$ at any given time is either 0 or 1, and the current state $S$ of the system is the binary vector determined collectively by the states of all $N$ devices.

Given a network in the current state $S$, its next state $S'$ is brought about by updating the state of exactly one of its constituent threshold devices. Thus, $S$ and $S'$ are separated by a maximum Hamming distance of one. The next state $s_i$ of device $i$ is determined by

$$s_i' = \begin{cases} 1 \text{ if } F_i(S) > 0 \\ 0 \text{ if } F_i(S) < 0 \\ s_i \text{ otherwise} \end{cases} \quad \text{where} \quad F_i(S) = \sum_{j=1}^{N} w_{ij} s_j + b_i$$

at the time the device is clocked. Each $w_{ij}$ is a constant multiplicative weight between the output of threshold device $j$ and the input of threshold device $i$, and $b_i$ is a constant bias term. Fig. 3 presents a high-level schematic of a general threshold device network. Threshold devices are "laterally connected" in the sense that the input to each device is a function of the outputs from every other device in the network.

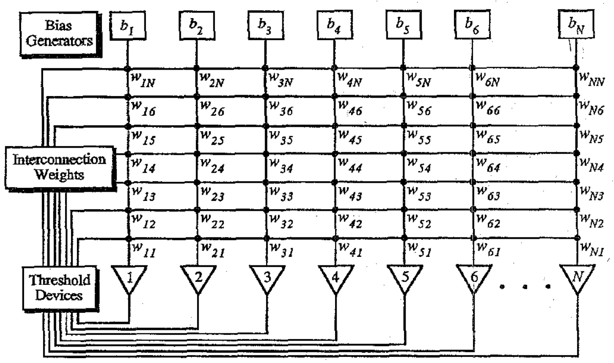

Bias Generators
Interconnection Weights
Threshold Devices
$b_1$ $b_2$ $b_3$ $b_4$ $b_5$ $b_6$ $b_N$
$w_{1N}$ $w_{2N}$ $w_{3N}$ $w_{4N}$ $w_{5N}$ $w_{6N}$ $w_{NN}$
$w_{16}$ $w_{26}$ $w_{36}$ $w_{46}$ $w_{56}$ $w_{66}$ $w_{N6}$
$w_{15}$ $w_{25}$ $w_{35}$ $w_{45}$ $w_{55}$ $w_{65}$ $w_{N5}$
$w_{14}$ $w_{24}$ $w_{34}$ $w_{44}$ $w_{54}$ $w_{64}$ $w_{N4}$
$w_{13}$ $w_{23}$ $w_{33}$ $w_{43}$ $w_{53}$ $w_{63}$ $w_{N3}$
$w_{12}$ $w_{22}$ $w_{32}$ $w_{42}$ $w_{52}$ $w_{62}$ $w_{N2}$
$w_{11}$ $w_{21}$ $w_{31}$ $w_{41}$ $w_{51}$ $w_{61}$ $w_{N1}$
1 2 3 4 5 6 . . . N

Fig. 3. Schematic of a general threshold device network.

If the matrix $W$ of interconnection weights is symmetric and possesses no nonzero terms on its main diagonal, i.e., if $w_{ij} = w_{ji}$ and $w_{ii} = 0$ for $1 \leq i, j \leq N$, then the value of the function

$$E(S) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} s_i w_{ij} s_j - \sum_{i=1}^{N} b_i s_i$$

is guaranteed to decrease monotonically as the state of a threshold device network evolves. Stable states of a network, those states in which the next state $s_i'$ of threshold device $i$ equals $s_i$ for $1 \le i \le N$, exhibit a one-to-one correspondence with the local minima of this "energy function." A network of laterally connected threshold devices whose stable states represent potential solutions to an optimization problem can be used as a simple means for computing these solutions [12].

## 4 Threshold Device Networks for Generalized Vertex Cover Problems

The primary theoretical contributions of this paper are expressed in three theorems, each of which contributes to the eventual development of methodologies for effective on-chip spare row/column allocation in VLSI arrays. The purpose of this section is to identify and analyze a class of threshold device networks which is proven applicable to the GVC problem.

Consider a network of threshold devices whose energy function is of the form

$$E_{VC}(S) = -A\sum_{i=1}^{N}\sum_{j=1}^{N} s_i c_{ij} s_j$$

where each $c_{ij}$ is determined by the connection matrix of some undirected graph $G$ with no self-loops, and where $A$ is any positive constant. If a correspondence is established between device $i$ of the network and vertex $i$ of $G$ for $1 \le i \le N$, and if a subset $V_0$ is defined of which vertex $i$ is considered a member whenever $s_i = 0$, then we have the following lemma.

LEMMA 1. *Let $H$ be a network of laterally connected threshold devices whose energy function is given by $E_{VC}$. A necessary and sufficient condition for any state to be a local minimum of $E_{VC}$, and hence a stable state of $H$, is that its associated vertex set $V_0$ be a vertex cover of $G$.*

PROOF OF NECESSITY. Suppose there exists some state $S$ of $H$ such that

1) the state is a local minimum of $E_{VC}$, and
2) its associated $V_0$ is not a vertex cover of $G$.

Recalling the correspondence established between threshold devices and vertices, and between connection matrix elements and graph edges, condition 2 implies that there must be some $i$ and $j$ such that $c_{ij} = s_i = s_j = 1$. Switching either $s_i$ or $s_j$ from 1 to 0, however, results in a decrease of system energy by $2A$, a strictly positive value. Thus $S$ is not a local minimum of $E_{VC}$, and condition 1 is violated.

PROOF OF SUFFICIENCY. Inspection of $E_{VC}$ reveals that its globally minimal value is 0. This minimal value is achieved iff $s_i = 0$, or $s_j = 0$, or both, whenever $c_{ij} = 1$. As every state $S$ which represents a vertex cover must satisfy this condition, every such state is a local minimum of $E_{VC}$. □

Now consider a laterally connected network whose $N$ threshold devices are conceptually divided into $K$ arbitrary subsets, $L_1$ through $L_K$, such that each device is a member of exactly one subset and no subset is empty. Let us denote by $z_i$ the number of threshold devices in set $L_i$ which, at any given time, are in state 0. Suppose that the network energy function is given by

$$E_{MVC}(S) = E_{VC}(S) + \sum_{i=1}^{K} B_i z_i^2 .$$

Establishing the usual correspondence between vertices and threshold devices, and defining $V_0$ as before, we state the following lemma.

LEMMA 2. *Let $H$ be a network of laterally connected threshold devices whose energy function is given by $E_{MVC}$, and whose parameters $A$ and $B_i$ are chosen so as to obey the inequalities*

$$A > \frac{B_i(2|L_i| - 1)}{2} \text{ and } B_i \ge 0 \text{ for } 1 \le i \le K.$$

*A necessary condition for any state of $H$ to be stable is that its associated $V_0$ be a vertex cover of $G$.*

PROOF. Suppose there exists a stable state $S$ of $H$ for which $V_0$ is not a vertex cover of $G$. Then it must be true that

1) some edge in $G$ has neither of its endpoints in $V_0$ and
2) system energy cannot be reduced by rectifying this situation.

We demonstrate that our choice of network parameters ensures conditions 1 and 2 cannot simultaneously be satisfied.

Condition 1 implies that $z_i < |L_i|$ for some $i$ between 1 and $K$. Suppose that switching one of the remaining $|L_i| - z_i$ threshold devices in subset $L_i$ from state 1 to state 0 will have the effect of including in $V_0$ an endpoint of $n$ edges whose other endpoints did not previously lie in $V_0$. Doing so will prevent $2n$ nonzero connection matrix elements from contributing to the value of the energy function. Such a change in the state of $H$ results in a net decrease in energy iff

$$-A(2\,n) + B_i\left[(z_i + 1)^2 - z_i^2\right] < 0$$

$$\text{or, in other words, } A > \frac{B_i(2\,z_i + 1)}{2\,n}$$

In the limiting case when only one member of $L_i$ is in state 1, and switching the state of this device eliminates the contributions of two nonzero connection matrix elements, this inequality reduces to

$$A > \frac{B_i(2|L_i| - 1)}{2}.$$

Maintaining this inequality across every parameter of $H$ guarantees that conditions 1 and 2 are incompatible. Hence, stable states of $H$ are associated with vertex covers of $G$. □

Every stable state of a threshold device network satisfying the requirements of Lemma 2 represents some vertex cover of the network's associated graph $G$. It is not the case, however, that every vertex cover of $G$ is represented by some stable state. We assert the following theorem.

THEOREM 1. *Let $H$ be a network of laterally connected threshold devices which fulfills the requirements of Lemma 2, and let $U$ denote the union of all sets $L_i$ of threshold devices for which $B_i > 0$. A necessary and sufficient condition for any state of $H$ to be stable is that its associated vertex set $V_0$ be a vertex cover of $G$ which is minimal over the set of vertices corresponding to $U$.*

PROOF OF NECESSITY. Suppose there exists some state $S$ of $H$ such that

1) the state is stable, and
2) it does not represent a vertex cover of $G$ which is minimal over the set of vertices corresponding to $U$.

Condition 1 and Lemma 2 guarantee that $S$ does in fact represent a vertex cover of $G$. Condition 2 implies that there exists some vertex $i$ in $V_0$ whose elimination from the set would yield a new set which is also a vertex cover, and Lemma 1 dictates that the value of $E_{VC}$ cannot change as a result. The elimination of vertex $i$ from $V_0$ is effected by switching $s_i$ from 0 to 1, a process which results in a reduction of system energy given by

$$B_i[z_i^2 - (z_i - 1)^2]$$

assuming that device $i$ is a member of set $L_i$, and that $z_i$ members of this set are in state 0 when $H$ is in state $S$. As this reduction is a strictly positive value whenever device $i$ is a member of $U$, it must be true that $S$ is not a minimum of $E_{MVC}$. Hence, $S$ is not a stable state of $H$.

PROOF OF SUFFICIENCY. Suppose $H$ is currently in some state $S$ which represents a vertex cover that is minimal over the vertices corresponding to $U$. The next state of $H$ can differ from $S$ in one of three ways. Either

1) the state of some device is switched from 1 to 0, or
2) the state of some device which does not belong to $U$ is switched from 0 to 1, or
3) the state of some device which belongs to $U$ is switched from 0 to 1.

Lemma 1 guarantees that $S$ is a local minimum of $E_{VC}$. Since the sum of the remaining terms in $E_{MVC}$ never decreases as the number of 0s in $S$ increases, option 1 cannot result in a reduction of system energy. Nor can option 2, since any device which is not a member of $U$ must belong to some set $L_i$ for which $B_i = 0$. Because option 3 inevitably yields a new state which does not represent a vertex cover of $G$, it must lead to an increase in system energy, as made evident in the proof of Lemma 2. We conclude that $S$ is a local minimum of $E_{MVC}$ and hence a stable state of $H$. □

Clearly, a vertex cover which is nonminimal over some vertex set $R$ can never be an optimal solution to the GVC problem with respect to sets $R_1, R_2, ..., R_D$ if $R$ is a subset of $R_1 \cup R_2 \cup ... \cup R_D$. The threshold device network of Theorem 1 is naturally applicable to the GVC problem, and a simple method for mapping any specific problem instance onto such a network is apparent. For a graph $G$ with any given choice of disjoint vertex sets, one simply constructs a network whose energy function is of the form of $E_{MVC}$, where each $c_{ij}$ is defined by the connection matrix associated with $G$, and where some set of threshold devices corresponds to each vertex set over which vertex covers are to be minimized. A positive $B_i$ value is chosen for each such set of threshold devices, while $B_i$ values for any remaining sets are left at 0, and the constant $A$ is chosen as per the specification of Theorem 1.

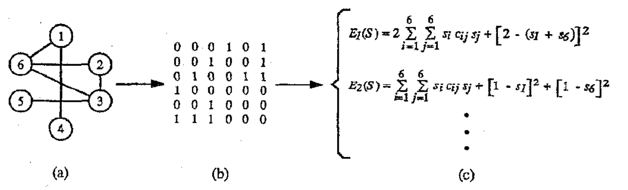

1
2
3
4
5
6
0 0 0 1 0 1
0 0 1 0 0 1
0 1 0 0 1 1
1 0 0 0 0 0
0 0 1 0 0 0
1 1 1 0 0 0
$E_1(S) = 2\sum_{i=1}^{6}\sum_{j=1}^{6} s_i c_{ij} s_j + [2 - (s_1 + s_6)]^2$
$E_2(S) = \sum_{i=1}^{6}\sum_{j=1}^{6} s_i c_{ij} s_j + [1 - s_1]^2 + [1 - s_6]^2$
(a)
(b)
(c)

Fig. 4. An undirected graph, its connection matrix, and energy functions of networks that find minimal covers with respect to vertices 1 and 6.

Aspects of the mapping process are illustrated in Fig. 4. Given the graph depicted in Fig. 4a, suppose it is desired to generate vertex covers which are minimal with respect to the set of vertices consisting of vertex 1 and vertex 6. The graph's connection matrix, shown in Fig. 4b, is used to determine each $c_{ij}$ in the network energy functions of Fig. 4c. Function $E_1$ results from the mapping technique described above. Function $E_2$, the product of an alternative mapping process, gives rise to identical stable states and serves to demonstrate that the mapping from a given instance of the GVC problem to a network of threshold devices is not generally unique.

## 5 Algorithms for Network Convergence

Given any instance of the GVC problem, it is now possible to construct a network of laterally connected threshold devices whose stable states represent potentially optimal solutions. Nevertheless, the applicability of such a network to spare allocation is limited without some means to ensure that stable states are encountered within a reasonable length of time. This section describes threshold device clocking procedures which, for networks that fulfill the requirements of Theorem 1, guarantee convergence to a stable state in a maximum number of steps.

Theorem 2 describes a procedure which can be preset, in the sense that an inflexible sequence can be established in which devices are to be clocked. It is presented below without proof.

THEOREM 2. *Let $H$ be a network of laterally connected threshold devices which fulfills the requirements of Theorem 1, let $U$ denote the union of all sets $L_i$ of threshold devices for which $B_i > 0$, and let $S$ be any initial state. If each threshold device in $H$ is clocked exactly once in any arbitrary order, after which each threshold device in the set $U$ is again clocked exactly once in any arbitrary order, then the resulting state $S'$ of $H$ will be stable.*

It should be noted that a similar scheme was derived in [13] for a less general class of networks.

One drawback to the procedure of Theorem 2 is that it disregards information present in the current state of the network which can be used to inform the choice of which network state to select next. Theorem 3 describes an alternative procedure which makes use of this information.

THEOREM 3. *Let $H$ be a network of size $N$ which fulfills the requirements of Theorem 1, and whose constants obey the inequality*

$$A > B_i(2\,|L_i| - 1) \text{ for } 1 \le i \le K.$$

*Define $U$ in the usual manner, and for each threshold device $i$ in $H$, define a function $G_i$ such that*

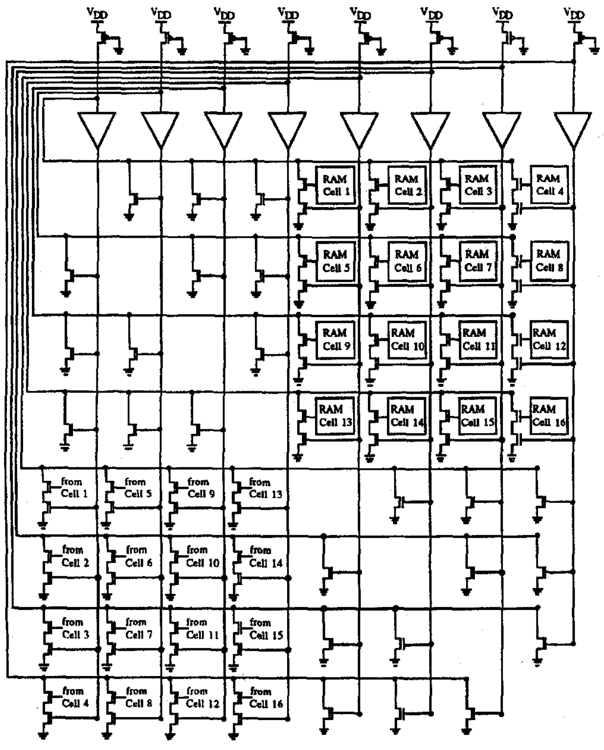
$$G_i(S) = \begin{cases} F_i(S) \text{ if } F_i(S) > 0 \text{ and } s_i = 0 \\ -F_i(S) \text{ if } F_i(S) < 0 \text{ and } s_i = 1 \\ 0 \text{ otherwise.} \end{cases}$$

Suppose a clocking procedure is established under which threshold device $i$ in $H$ is clocked iff $G_i$ is greater than or equal to $G_j$ for all other $j \neq i$, with ties broken in some arbitrary manner. Then, convergence to a stable state is guaranteed in no more than $N + |U| - 1$ steps.

Intuitively, the function $G_i$ quantifies the encouragement received by a threshold device $i$ to switch its state. It is easily shown that the procedure of Theorem 3 effectively leads a threshold device network along the path of gradient descent in $E_{MVC}$.

## 6 On-Chip Spare Allocation in Embedded Memories: A Sample Application

Section 2 established that the problem of spare row/column allocation can be modeled as a special case of the generalized vertex cover problem. Theorem 1 went on to demonstrate that, given any instance of the GVC problem, a network of laterally connected threshold devices exists which possesses a unique stable state representing each minimal solution to the instance at hand. Furthermore, given a specified network architecture, it is clear that different problem instances can be accommodated by simply modifying the connection matrix upon which the network's energy function is based. It follows from these results that a programmable network of an appropriate size could provide the basis for an optimization system capable of devising spare allocation schemes for any pattern of faults likely to occur in a given array. This section demonstrates how Theorems 2 and 3, when applied to appropriate network architectures, can offer a high probability of encountering stable states which represent *feasible* spare allocation schemes. Electronic implementations of the resulting systems, though extremely simple, nevertheless offer a high rate of success in solving the computationally intractable spare allocation problem.



$$w_{ij}=\begin{cases}-2\,B_1\,(1-\delta_{ij}) \text{ if } i\le m_1 \text{ and } j\le m_1\\ (-2\,A\,c_{ij}) \text{ if } i\le m_1 \text{ and } j>m_1\\ (-2\,A\,c_{ij}) \text{ if } i>m_1 \text{ and } j\le m_1\\ -2\,B_2\,(1-\delta_{ij}) \text{ if } i>m_1 \text{ and } j>m_1\end{cases} \quad \text{and} \quad b_i=\begin{cases}B_1\,(2\,m_1-1) \text{ if } i\le m_1\\ B_2\,(2\,m_2-1) \text{ if } i>m_1\end{cases}$$

Fig. 5. An implementation of network $H_1$ for $m_1 = m_2 = 4$.

Employing the technique described in Section 4 to map an instance of the GVC problem onto a network of threshold devices, the spare allocation problem can be represented as follows. With faults expected to occur in $m_1$ or fewer distinct rows and $m_2$ or fewer distinct columns, construct a network $H_1$ of size $N$, where $N = m_1 + m_2$, and choose subsets $L_1$ and $L_2$ of cardinality $m_1$ and $m_2$, respectively. Let the threshold devices of set $L_1$ be numbered from 1 to $m_1$, let those of $L_2$ be numbered from $m_1 + 1$ to $N$, and establish a correspondence between device $i$ and row $i$ of the fault pattern for $1 \le i \le m_1$, and between device $i$ and column $i - m_1$ for $(m_1 + 1) \le i \le N$. Finally, let connection matrix elements be determined by a graph associated with the specific pattern of faults to be repaired. The energy function thus created is

$$E_1(S) = A\sum_{i=1}^{N}\sum_{j=1}^{N} s_i c_{ij} s_j + B_1\left[M_1 - \sum_{i=1}^{M_1} s_i\right]^2 + B_2\left[M_2 - \sum_{i=m_1+1}^{N} s_i\right]^2.$$

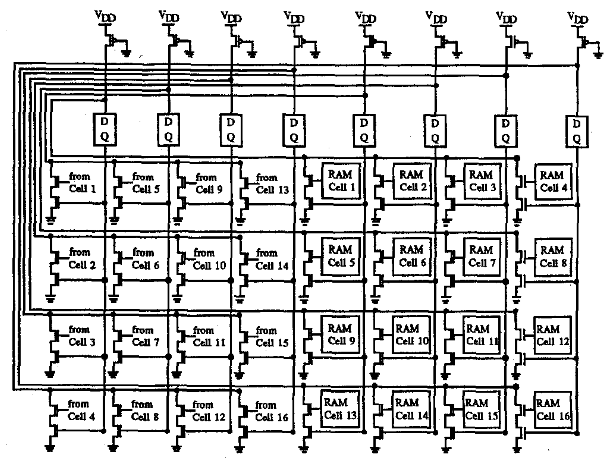
By expanding squared terms and relating the result to the general energy function of Section 3, values can be derived for network interconnection weights and bias terms. These values, along with an electronic implementation of the resulting network, are illustrated in Fig. 5.

If the initial state of $H_1$ is fixed to the $N$-vector consisting entirely of 1s, then the gradient descent procedure of Theorem 3 allocates spare rows and columns according to what, in effect, is a simple heuristic algorithm. Gradient descent allocates the first spare to cover some row or column containing the maximum number of faults. Spares are assigned thereafter in the same manner, with ties between any row and column broken in favor of that set from which fewer spares have already been allocated. The tendency to eliminate the maximum number of faults with each allocation may even be overridden if the number of spares allocated from one set greatly exceeds that of the other. Intuitively, the system will tend to preserve a balance between the number of spare rows and spare columns assigned, thereby preferring stable states which represent feasible spare allocation schemes.

Theorem 2 can be used as the basis for an even less complex alternative system. Since the preset updating procedure is unable to exploit any heuristic qualities present in a given energy function, however, it is worthwhile to construct a new energy function whose implied network $H_2$ is as simple as possible. Let us modify network $H_1$ by establishing $N$ threshold device subsets, $L_1$ through $L_N$, rather than the original $L_1$ and $L_2$. The resulting energy function

$$E_2(S) = A\sum_{i=1}^{N}\sum_{j=1}^{N} s_i c_{ij} s_j + B_1[1-s_1]^2 + B_2[1-s_2]^2 + \cdots + B_N[1-s_N]^2$$

gives rise to the interconnection weights and bias terms detailed in Fig. 6. An electronic implementation of network $H_2$, also shown in Fig. 6, is easily constructed using strictly digital components. Approximating the silicon layout area of an integrated circuit by the number of transistors it comprises, and given an expected maximum fault pattern size of 32 by 32 elements, a network of this kind would require only 0.29 percent of the total area of a 1 MBit DRAM.



$$w_{ij}=\begin{cases}0 \text{ if } i\le m_1 \text{ and } j\le m_1\\ (-2\,A\,c_{ij}) \text{ if } i\le m_1 \text{ and } j>m_1\\ (-2\,A\,c_{ij}) \text{ if } i>m_1 \text{ and } j\le m_1\\ 0 \text{ if } i>m_1 \text{ and } j>m_1\end{cases} \quad \text{and} \quad b_i=B_i$$

Fig. 6. An implementation of network $H_2$ for $m_1 = m_2 = 4$.

Consideration must now be given to a means of generating initial states for network $H_2$, and to a strategy for clocking its threshold devices. In order to ensure that neither row nor column replacements are preferred, and to increase the likelihood of devising feasible spare allocation schemes, it is reasonable to establish an updating protocol which alternates between threshold devices corresponding to rows and devices corresponding to columns. Initial states may be chosen in a random fashion to allow for thorough exploration of the search space, and repeated iterations performed until the network converges to a state representing a feasible spare allocation scheme.

Fig. 7 illustrates a paradigm for using a threshold device network to provide for on-chip spare allocation in embedded memory arrays. Faulty cells are located by BIST hardware, and the resulting pattern of faults is used to configure the interconnection weight matrix of an appropriate network. (It is important to note that, typically, fault distributions within large memories can be represented by compacted patterns which are many orders of magnitude smaller.) With programming completed, control hardware resets the threshold devices to some initial state, and begins an updating procedure which brings the network to convergence. The resulting stable state is evaluated to determine whether it represents a feasible solution to the problem instance at hand. If so, a signal is generated which directs reconfiguration hardware to repair the memory as per the suggested spare allocation scheme.

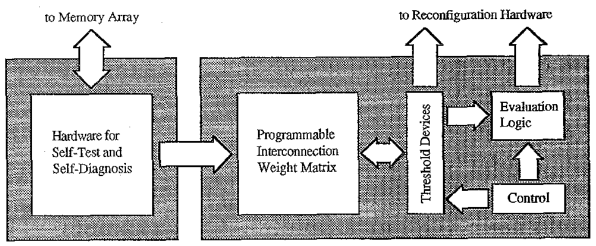

to Memory Array
to Reconfiguration Hardware
Hardware for Self-Test and Self-Diagnosis
Programmable Interconnection Weight Matrix
Threshold Devices
Evaluation Logic
Control

Fig. 7. Application of a threshold device network to spare allocation in embedded memories.

Tables 1 and 2 summarize the performance of the heuristic and iterative approaches, respectively, under variations of fault pattern size, number of available spare rows and columns, and average number of faulty elements per pattern. Fault patterns were generated randomly through a process which guarantees repairability. Table 3 summarizes the performance of Repair Most [10], the most viable alternative algorithm for on-chip spare allocation, when applied to the same sets of test instances. Tables 4 and 5 detail performance of the iterative approach as a function of the number of iterations completed. Here, "rate of success" refers to the percentage of problem instances for which the algorithm under test was capable of devising a feasible solution. It is assumed that the spare allocation systems under test are themselves fault free.

TABLE 1
PERFORMANCE OF THE HEURISTIC APPROACH,
SHOWING THE RATE OF SUCCESS FOR EACH CASE

| | 16-by-16 Fault Pattern | | | 32-by-32 Fault Pattern | | |
|---|---|---|---|---|---|---|
| | 4 Spare Rows and 4 Spare Columns | 6 Spare Rows and 6 Spare Columns | 8 Spare Rows and 8 Spare Columns | 8 Spare Rows and 8 Spare Columns | 12 Spare Rows and 12 Spare Columns | 16 Spare Rows and 16 Spare Columns |
| 10% Faulty | 98.6 | 100 | 100 | 99.8 | 99.8 | 99.8 |
| 20% Faulty | 100 | 99.4 | 99.0 | 100 | 100 | 99.8 |
| 30% Faulty | 100 | 100 | 98.0 | 100 | 100 | 99.8 |

TABLE 2
PERFORMANCE OF THE ITERATIVE APPROACH,
SHOWING THE RATE OF SUCCESS FOR EACH CASE

| | 16-by-16 Fault Pattern | | | 32-by-32 Fault Pattern | | |
|---|---|---|---|---|---|---|
| | 4 Spare Rows and 4 Spare Columns | 6 Spare Rows and 6 Spare Columns | 8 Spare Rows and 8 Spare Columns | 8 Spare Rows and 8 Spare Columns | 12 Spare Rows and 12 Spare Columns | 16 Spare Rows and 16 Spare Columns |
| 10% Faulty | 100 | 100 | 100 | 100 | 100 | 100 |
| 20% Faulty | 100 | 100 | 100 | 100 | 100 | 100 |
| 30% Faulty | 100 | 100 | 100 | 100 | 100 | 100 |

TABLE 3
PERFORMANCE OF REPAIR MOST,
SHOWING THE RATE OF SUCCESS FOR EACH CASE

| | 16-by-16 Fault Pattern | | | 32-by-32 Fault Pattern | | |
|---|---|---|---|---|---|---|
| | 4 Spare Rows and 4 Spare Columns | 6 Spare Rows and 6 Spare Columns | 8 Spare Rows and 8 Spare Columns | 8 Spare Rows and 8 Spare Columns | 12 Spare Rows and 12 Spare Columns | 16 Spare Rows and 16 Spare Columns |
| 10% Faulty | 71.0 | 83.4 | 94.4 | 94.8 | 85.6 | 91.2 |
| 20% Faulty | 99.2 | 80.0 | 78.4 | 100 | 99.2 | 92.8 |
| 30% Faulty | 100 | 97.2 | 80.6 | 100 | 100 | 98.9 |

TABLE 4
PERFORMANCE OF THE ITERATIVE APPROACH WHEN APPLIED
TO 16-BY-16 FAULT PATTERNS, SHOWING RATE OF SUCCESS
AS A FUNCTION OF NUMBER OF ITERATIONS

| | | Iterations | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80 | 90 | 100 |
| 4 Spare Rows and 4 Spare Columns | 10% Faulty | 77.6 | 89.4 | 93.0 | 94.8 | 96.2 | 97.2 | 97.2 | 98.0 | 98.8 | 99.0 |
| | 20% Faulty | 90.8 | 96.0 | 98.0 | 98.8 | 98.8 | 99.0 | 99.4 | 99.4 | 99.6 | 99.6 |
| | 30% Faulty | 97.4 | 99.2 | 99.8 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| 6 Spare Rows and 6 Spare Columns | 10% Faulty | 96.6 | 99.2 | 99.8 | 99.8 | 100 | 100 | 100 | 100 | 100 | 100 |
| | 20% Faulty | 73.2 | 84.8 | 90.0 | 91.6 | 93.2 | 94.6 | 95.0 | 96.0 | 96.6 | 97.6 |
| | 30% Faulty | 83.0 | 91.6 | 94.0 | 95.0 | 95.8 | 96.6 | 97.6 | 98.4 | 98.6 | 98.6 |
| 8 Spare Rows and 8 Spare Columns | 10% Faulty | 99.8 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| | 20% Faulty | 92.0 | 97.0 | 98.4 | 99.2 | 99.8 | 99.8 | 99.8 | 99.8 | 99.8 | 99.8 |
| | 30% Faulty | 78.2 | 88.0 | 92.0 | 95.2 | 96.4 | 96.8 | 97.4 | 98.0 | 98.4 | 98.6 |

TABLE 5
PERFORMANCE OF THE ITERATIVE APPROACH WHEN APPLIED
TO 32-BY-32 FAULT PATTERNS, SHOWING RATE OF SUCCESS
AS A FUNCTION OF NUMBER OF ITERATIONS

| | | Iterations | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 20 | 40 | 60 | 80 | 100 | 120 | 140 | 160 | 180 | 200 |
| 8 Spare Rows and 8 Spare Columns | 10% Faulty | 78.6 | 87.8 | 91.4 | 94.8 | 96.2 | 97.0 | 97.8 | 98.0 | 98.6 | 98.8 |
| | 20% Faulty | 97.0 | 98.8 | 99.8 | 99.8 | 99.8 | 99.8 | 100 | 100 | 100 | 100 |
| | 30% Faulty | 98.9 | 99.8 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| 12 Spare Rows and 12 Spare Columns | 10% Faulty | 82.8 | 93.0 | 96.4 | 96.6 | 97.8 | 98.0 | 98.4 | 98.6 | 98.8 | 99.0 |
| | 20% Faulty | 75.4 | 84.0 | 90.0 | 93.0 | 94.8 | 95.6 | 96.4 | 97.2 | 97.4 | 97.8 |
| | 30% Faulty | 93.4 | 96.4 | 97.8 | 98.8 | 98.8 | 98.8 | 98.8 | 98.8 | 99.2 | 99.2 |
| 16 Spare Rows and 16 Spare Columns | 10% Faulty | 99.4 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 | 100 |
| | 20% Faulty | 83.0 | 91.2 | 94.8 | 96.4 | 97.0 | 98.0 | 98.6 | 99.0 | 99.0 | 99.2 |
| | 30% Faulty | 72.2 | 82.0 | 87.2 | 89.2 | 91.6 | 92.8 | 93.0 | 94.0 | 94.4 | 95.0 |

## 7 CONCLUSION

This paper lays foundations for an approach to on-chip spare allocation in rectangular VLSI arrays, demonstrating how properly designed networks of simple threshold devices can be used as the basis for optimization systems which are simple and effective. It should be emphasized that the proposed hardware systems are *not* as powerful as some existing software-based spare allocation techniques. Such existing techniques, however, were developed for utilization by external repair equipment, and the algorithms employed typically require the full resources of a general-purpose digital computer. Hardware implementations would incur far too much area overhead to be considered viable for the purpose of built-in self-repair. In contrast, the systems developed in this paper achieve near perfect success in devising spare allocation schemes, using hardware whose complexity is negligible when compared with a VLSI array of any substantial size.

Earlier research along these lines [14] succeeded in developing threshold device networks which, intuitively, seemed applicable to the spare allocation problem, but which occasionally failed to yield valid spare allocation schemes in practice. While they do build upon these earlier results, the systems described in this paper include a number of major refinements which make their actual implementation for the purpose of built-in self-repair an immediate practical possibility. First and most importantly, the general network architectures developed are proven to possess stable states which represent only valid spare allocation schemes. Second, certain special instances of these networks can be implemented in a straightforward manner using strictly digital components. It deserves mention also that Theorem 1, as derived herein, is easily extended to the case of threshold device networks which operate in the analog domain [15], raising the possibility of alternative optimization system designs that offer much greater speed of operation than those described here.

## Acknowledgments

This research was supported in part by the U.S. National Science Foundation under Grant Number MIP-9013092.

## References

[1] S.-Y. Kuo and W.K. Fuchs, "Efficient spare allocation for reconfigurable arrays," *IEEE Design & Test*, vol. 4, pp. 24-31, Feb. 1987.

[2] D. Blough and A. Pelc, "A clustered failure model for the memory array reconfiguration problem," *IEEE Trans. Computers*, vol. 42, pp. 518-528, May 1993.

[3] M.-F. Chang, W.K. Fuchs, and J.H. Patel, "Diagnosis and repair of memory with coupling faults," *Proc. 1988 Int'l Conf. Computer-Aided Design*, pp. 524-527, Nov. 1988.

[4] J.R. Day, "A fault-driven, comprehensive redundancy algorithm," *IEEE Design & Test*, vol. 2, pp. 35-44, June 1985.

[5] R.C. Evans, "Testing repairable RAMs and mostly good memories," *Proc. 1981 Int'l Test Conf.*, pp. 49-55, Oct. 1981.

[6] N. Funabiki and Y. Takefuji, "A parallel algorithm for allocation of spare cells on memory chips," *IEEE Trans. Reliability*, vol. 40, pp. 338-346, Aug. 1991.

[7] R.W. Haddad and A.T. Dahbura, "Increased throughput for the testing and repair of RAMs with redundancy," *Proc. 1987 Int'l Conf. Computer-Aided Design*, pp. 230-233, Nov. 1987.

[8] N. Hasan and C.L. Liu, "Minimum fault coverage in reconfigurable arrays," *Proc. 1988 Int'l Symp. Fault-Tolerant Computing*, pp. 348-353, June 1988.

[9] F. Lombardi and W.K. Huang, "Approaches for the repair of VLSI/WSI DRAMs by row/column deletion," *Proc. 1988 Int'l Symp. Fault-Tolerant Computing*, pp. 342-347, June 1988.

[10] M. Tarr, D. Boudreau, and R. Murphy, "Defect analysis system speeds test and repair of redundant memories," *Electronics*, vol. 57, pp. 175-179, Jan. 12, 1984.

[11] C.-L. Wey and F. Lombardi, "On the repair of redundant RAMs," *IEEE Trans. Computer-Aided Design*, vol. 6, pp. 222-231, Mar. 1987.

[12] J.J. Hopfield and D.W. Tank, " 'Neural' computation of decisions in optimization problems," *Biological Cybernetics*, vol. 52, pp. 141-152, July 1985.

[13] Y. Shrivastava, S. Dasgupta, and S.M. Reddy, "Guaranteed convergence in a class of Hopfield networks," *IEEE Trans. Neural Networks*, vol. 3, pp. 951-961, Nov. 1992.

[14] P. Mazumder and J.-S. Yih, "A new built-in self-repair approach to memory yield enhancement by using neural-type circuits," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 124-136, Jan. 1993.

[15] J.J. Hopfield, "Neurons with graded response have collective computational properties like those of two-state neurons," *Proc. Nat'l Academy of Science USA*, vol. 81, pp. 3,088-3,092, May 1984.

# Hyperneural Network—An Efficient Model for Test Generation in Digital Circuits

Suresh Rai and Weian Deng

**Abstract**—This paper considers the problem of applying neural network for logic circuit testing and proposes an efficient method based on *hyperneural network* (HNN). The HNN uses an energy function that not only considers binary relations but also captures all higher order relations among $N$ neurons. We illustrate the hyperneural concept using two formulations. First, a constraint engery function is defined and the gate model is obtained. Second, the Hopfield network is reformulated to generate the gate level hyperneural model. The gate level HNNs are used to give a mathematical form to the digital circuit that, in turn, requires optimization techniques to solve the test generation problem. We have used ISCAS'85 benchmark circuits to illustrate the method. Results are compared with those obtained from PODEM, MODEM, and FAN.

**Index Terms**—Hyperneural model, logic circuit testing, neural network, NP-completeness, optimization, pseudo-Boolean programming, satisfiability problem.

## 1 Introduction

A Boolean difference (BD) approach defines a complete set of tests for a given fault and, thus, offers a unique advantage over path sensitization techniques [4], [9], [10]. It was disfavored because of the difficulty in the manipulation of algebraic forms using computers. Currently, reformulating BD between the unfaulted and faulted circuits using pseudo-Boolean programming [1], [5], Boolean satisfiability and implication graph [11], and neural network [3], [8] has renewed interest in it. They also help make the BD problem machine computable.

This paper extends the concept of neural networks (NN) modeling of digital circuits for test generation. The NN are interconnections of *neurons* that are simple computing elements and are characterized by a pseudo-Boolean quadratic function, called the *energy function* [3], [6]. Chakradhar et al.[3] and Fujiwara [8] use Hopfield network and the *basis* NN set for two input gates. Hopfield network of $N$ neurons describes only binary relations between neurons. Gates with more than two inputs need *hidden* neurons. Even two inputs XOR and XNOR gates require four neurons. Inclusion of an additional neuron doubles the search space. Finding a valid test set is, therefore, either increasingly hard or the network converges to an invalid solution. We propose here a new model, henceforth called *hyperneural network* (HNN), to overcome the difficulties with existing NN modeling techniques of logic gates. The proposed technique captures all (including binary) relations among $N$ neurons. Recently, Ortega et al. [13] have independently defined a similar function, known as generalized Hopfield network (GHN). They use GHN and spectral coefficients in the design of the extra modules of a concurrent testable circuit.

The lay out of the paper is as follows: Section 2 provides the background needed in later sections. We discuss two formulations to present the concept of HNN in Section 3. Section 4 describes the technique. Section 5 provides experimental results for some typi-

• *The authors are with the Electrical and Computer Engineering Department, Louisiana State University, Baton Rouge, LA 70803.*
*E-mail: suresh@gate.ee.lsu.edu.*

*Manuscript received September 1993; revised February 1995.*
*For information on obtaining reprints of this article, please send e-mail to: transactions@computer.org, and reference IEEECS Log Number C95139.*