JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING 12, 249–268 (1991)

Designing Fault-Tolerant Systems Using Automorphisms*

SHANTANU DUTT

Department of Electrical Engineering, University of Minnesota, Minneapolis, Minnesota 55455

AND

JOHN P. HAYES

Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, Michigan 48109-2110

This paper presents a general theory for modeling and designing fault-tolerant multiprocessor systems in a systematic and efficient manner. We are concerned here with structural fault tolerance, defined as the ability to reconfigure around faults in order to preserve the interconnection structure of a multiprocessor. We represent multiprocessor systems by graphs whose node sets denote processors and whose edge sets denote dedicated interprocessor links. The fault-tolerant design and reconfiguration process of a multiprocessor is modeled by graph automorphisms. This automorphism-based methodology also models some important practical design features not previously addressed, including applicability to any multiprocessor structure and any number of faults. Low redundancy and efficient reconfigurability are also addressed. We apply our approach directly to a class of regular multiprocessor graphs termed circulant. For noncirculant graphs we give an algorithm to construct their circulant edge supergraphs efficiently. An application of the theory to the design of fault-tolerant hypercube multiprocessors is described. The resulting designs are shown to be far superior to those proposed in previous work.

1. INTRODUCTION

Numerous computer architectures that use multiple processors to solve complex problems in parallel have been introduced recently. This particular architectural thrust has received impetus from developments in IC technology that make it possible to fabricate complex single processors or a few simple ones, along with their local memory and input/output (I/O) ports, on a single chip (very large scale integration or VLSI). Several commercial multiprocessors like the NCUBE/ten [13] and INTEL iPSC series of hypercube computers and the Connection Machine [14], which can accommodate hundreds or thousands of processors are currently available. With the introduction of large numbers of processors, the probability of failure increases significantly. Many critical applications that use multiprocessors need a very low failure rate. For example, airborne systems typically require a failure rate of the order of 10^{-10} failures per hour [15]. Hence, the design of fault-tolerant multiprocessor systems is an important issue.

This paper addresses the design of structurally fault-tolerant realizations of large-scale distributed-memory multiprocessors. *Structural fault tolerance* is defined as the ability to reconfigure around faults so that the reconfigured system is isomorphic to the original one. Such fault tolerance is necessary mainly to reduce the performance degradation of parallel algorithms that execute on specific classes of multiprocessor interconnection structures, such as trees or hypercubes. Since the interconnection structure changes when system components (processors and links) fail, it is necessary to reconfigure around faults in order to preserve the basic interconnection structure.

Fundamental to the reconfiguration process is the one-to-one mapping of faulty processors onto fault-free ones. This mapping process and the interconnection structure to which it applies are most often modeled mathematically by means of a graph G whose nodes and edges correspond to processors and interprocessor communication links, respectively. Faults in processors are then modeled by the removal of the corresponding nodes and the edges incident on them from G.

A number of fault-tolerant designs for specific multiprocessor architectures have been proposed on the basis of graph theoretic models. For example, Hayes [12] and Kwan and Toida [17] present designs for tolerating one and two faults, respectively, in systems whose interconnection structures form nonhomogeneous symmetric trees. Fault-tolerant designs for symmetric tree architectures appear in [22, 11, 20]; Banerjee *et al.* [2] give a fault-tolerant design scheme that tolerates one faulty cycle in the cube-connected-cycles architecture, while Lombardi *et al.* [19] present a scheme to reconfigure a faulty two-dimensional mesh-structured multiprocessor array by providing an extra row and column of

* This research was supported in part by the Office of Naval Research under Contract N00014 85 K 0531 and in part by a University of Michigan Rackham predoctoral fellowship.

0743-7315/91 $3.00

spare processors. Rosenberg *et al.* [24, 7] discuss the Diogenes method of obtaining a k-FT (k-fault-tolerant) realization of any graph G that can be decomposed into m outerplanar graphs by m bundles or stacks of wires. However, this method is very expensive in terms of total wire and layout area, especially when $m > 1$. In a related paper [8], we developed a design methodology based on a technique called node covering, for constructing optimal or near-optimal k-FT supergraphs of symmetric trees. The node-covering technique can implement an arbitrary degree of fault tolerance and is also generalizable to any graph G [9]. It also meets all the above criteria that are important in the design of fault-tolerant multiprocessors.

None of the preceding fault-tolerance schemes, except the Diogenes and node-covering methods, are generalizable to other architectures, and few can implement an arbitrary degree of fault tolerance. Most of the above design methods, except node covering, also fail to address several important practical aspects of fault-tolerant multiprocessors. These include:

1. Low redundancy. The system cost measured by the number of spare nodes and links should be minimized, for example, by using links that can be shared by appropriate switching mechanisms.

2. Fast, distributed, and incremental reconfiguration on the occurrence of faults. *Incremental reconfiguration* is the ability to reconfigure around any new set of faults without undoing any earlier reconfiguration done around previous faults. This reduces the down time of a system when faults occur. *Distributed reconfiguration* allows different parts of the system to reconfigure concurrently, which not only makes reconfiguration faster, but also avoids relying on a central controller for reconfiguration.

3. *Incremental design,* which is the ability to construct a system of high fault tolerance from one of lower fault tolerance, with little or no waste of hardware.

4. Amenability to local sparing, which we shortly define.

In this paper, we present a completely new and general design theory based on graph automorphisms that is capable of addressing all the above issues. The automorphic method is very suitable for multiprocessors that have a certain regularity in their structure (graphs), for example, hypercubes and meshes. Another salient feature of the automorphic methodology is that it is very versatile and allows for a number of variations in the design of fault-tolerant systems [9].

We first introduce the graph model used to represent multiprocessor systems and their faults and then review some graph automorphism concepts required in the sequel. A general discussion on incorporating fault tolerance in graphs using their automorphism group follows. Some properties of the subgraphs induced by cycles in an automorphism of G are derived; these properties lead to a simple fault-tolerant design scheme for an important class of graphs termed circulant. We then discuss an efficient implementation of the proposed k-FT supergraphs using switched links. We also present an algorithm for converting noncirculant graphs into circulant ones. This conversion enables the design method and the switch implementation for circulant graphs to be applied to noncirculant graphs. We apply the automorphic design method to a particular class of graphs that is widely used as multiprocessor interconnection topologies, viz., hypercubes, and compare the resulting fault-tolerant designs to previously proposed fault-tolerant hypercube designs [7, 23]. The paper concludes with a brief discussion on how the automorphic method can be used to implement local sparing.

2. PRELIMINARIES

The graph model used here to represent the interconnection structures of interest is an extension of the models proposed in [12, 28]. A graph $G(V, E)$ represents the structure of a multiprocessor system, where the node set $V(G)$ of G represents the processors in the system, and the edge set $E(G)$ of G represents unshared or dedicated interprocessor communication links. By *link* we mean the physical wires and interface circuits that connect two processors directly. $G(S)$ denotes the subgraph induced in G by the nodes in a subset S of $V(G)$. Unless otherwise specified, N and e denote the number of nodes and edges, respectively, of G; d denotes the node degree of G when it is regular (all nodes have the same degree) and the average node degree $\lceil 2e/N \rceil$ when G is not.

Two types of failures in a multiprocessor system are of interest, processor failures and link failures. In our model, a *link failure* corresponds to the deletion of an edge from G, while a *processor failure* corresponds to the removal of a node and all edges incident on it from G. If F is the set of faults (faulty nodes or edges) in G, then $G - F$ denotes the graph obtained by deleting the faults from G as explained above. We consider only processor failures in this research, since processors are more complex than links, and so have much higher failure probabilities; this is also the fault model used in most prior work. Link failure probability does not significantly affect the overall reliability of the system—links are generally passive components, while a typical processor consists of hundreds of thousands of active components. Figure 1a shows a graph G representing a three-dimensional hypercube, while Fig. 1b is the graph $G - F$, where F is the set of two faulty nodes shown darkened, and the edges incident on the faulty nodes are deleted.

G' is a *supergraph* of G if G is a subgraph of G' and $V(G') \supset V(G)$. G^e is an *edge supergraph* of G if G is a subgraph of G^e and $V(G^e) = V(G)$. A supergraph $G'[k, G]$ of G is a *k-fault-tolerant* realization of G if for any set F of k nodes of G', $G' - F$ contains a subgraph isomorphic to G. We use G' to denote either a generic fault-tolerant supergraph of G or a specific type of fault-tolerant supergraph as should be clear from the context. A *k-fault pattern* is any set of k faulty nodes. Thus $G'[k, G]$ can tolerate any f-fault pattern as long

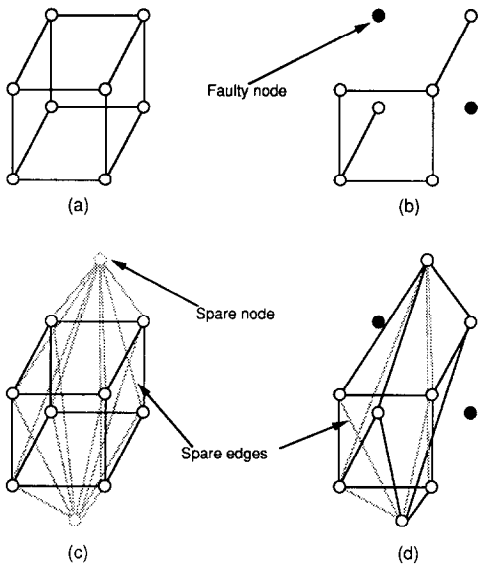

FIG. 1. (a) Graph G representing a three-dimensional hypercube. (b) $G - F$ for the set F of two faulty nodes. (c) A 2-FT supergraph $G'[2, G]$ of G. (d) Reconfiguration of G' around F.

as $f \leq k$. Instead of associating the set of spare nodes "globally" with the whole system, as in $G'[k, G]$, we can partition G into t subgraphs and associate t sets of spare nodes, each "locally" with one of the subgraphs. We term the design methods for the former systems *global sparing,* while those for the latter systems are termed *local sparing.* Local sparing can make the design and reconfiguration process simpler. This paper deals with the k-FT case only; an extension of the automorphic method for constructing local sparing designs is given in [9].

The goal of a fault-tolerant design is to preserve a certain structure G called the *basic graph;* the nodes and edges of G are *primary* nodes and edges. The fault-free subgraph of G' serving as the currently active graph isomorphic to G is called the *current graph.* The nodes and edges of the current graph G are *active* nodes and edges, respectively, while the rest of the nodes and edges of G' are *spare* or *redundant.* The following convention is used in all the figures. Spare nodes are shown as open circles with a light outline, spare edges as light lines, active nodes as open circles with thick outlines, active edges as thick lines, and faulty nodes as dark circles. Figure 1c shows a 2-FT supergraph of a three-dimensional hypercube, while Fig. 1d shows its reconfiguration around the two faulty nodes.

Next, we review some basic terminology pertaining to graph automorphisms [4, 29]. Two graphs G and H are said to be *isomorphic* if there is a bijection ϕ: $V(G) \rightarrow V(H)$ such that $\{x, y\} \in E(G)$ if and only if $\{\phi(x), \phi(y)\} \in E(H)$; ϕ is an *isomorphism* from G to H. An isomorphism from G to itself is called an *automorphism* of G. Basically an automorphism of G is a permutation of $V(G)$ that preserves adjacency. The set of all automorphisms of G forms a group under composition and is denoted by $\mathrm{aut}(G)$; $\mathrm{aut}(G)$ is said to *act* on $V(G)$. Two nodes x and y of G are said to be *similar* if there is an automorphism mapping x to y. Similarity is an equivalence relation (reflexive, symmetric, and transitive); the subsets (or equivalence classes) of nodes in the partition of $V(G)$ based on this equivalence relation are called *orbits.* Thus all nodes in an orbit O_i are similar, while any two nodes in different orbits are dissimilar. G is said to be *node-symmetric* if it has only one orbit, i.e., all its nodes are similar to one another. The graph of Fig. 2a is an example of a node-symmetric graph, and its only orbit O_1 contains all 20 nodes. At the other end of the spectrum, if the only automorphism of a graph G is the identity mapping, then G is said to be an *asymmetric* graph; it has $N = |V(G)|$ orbits, each containing a single node.

An action of $\mathrm{aut}(G)$ on $V(G)$ is said to be *k-transitive,* or transitive on ordered k-tuples of $V(G)$, if for any two ordered k-tuples $(x_1, x_2, \ldots, x_k)$ and $(y_1, y_2, \ldots, y_k)$ of

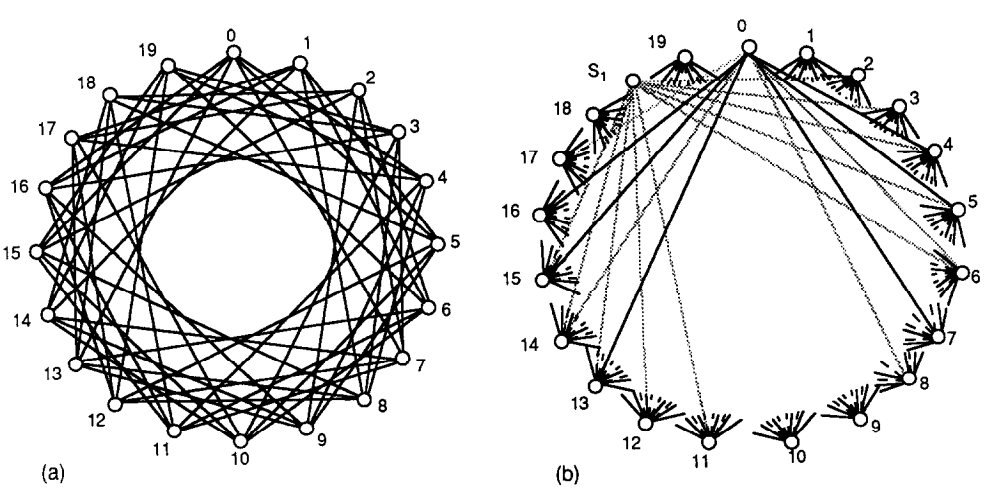

FIG. 2. (a) A circulant graph J with 20 nodes. (b) A 1-FT circulant supergraph J_1 of J with spare node s_0.

nodes of G, there is an automorphism mapping one k-tuple to the other. Thus aut(G) acts 1-transitively on a node-symmetric graph G. We now introduce the new concept of k-subtransitivity. An action of aut(G) on $V(G)$ is said to be *k-subtransitive* if for any two unordered subsets X and Y of $V(G)$ containing exactly k nodes, there is an automorphism mapping X to Y.

S_N is the set of all permutations of the generic set $\{0, 1, 2, \ldots, N-1\}$ of N elements; S_N is a group under composition. Any automorphism group of a graph with N nodes is a subgroup of S_N. Any automorphism α (or for that matter any permutation in S_N) can be expressed uniquely (up to ordering) as the product of disjoint cycles $(a_0, a_1, \ldots, a_{l-1})(b_0, b_1, \ldots, b_{m-1}) \cdots (x_0, x_1, \ldots, x_{p-1})$. This cyclic representation means that $\alpha(a_0) = a_1$, $\alpha(a_1) = a_2$, $\ldots$, $\alpha(a_{l-1}) = a_0, \ldots, \alpha(x_0) = x_1$, $\alpha(x_1) = x_2, \ldots, \alpha(x_{p-1}) = x_0$. For example, $\beta = (0, 4, 8, 12, 16)(1, 5, 9, 13, 17), (2, 6, 10, 14, 18)(3, 7, 11, 15, 19)$ is an automorphism of the graph J of Fig. 2a, and the mapping it specifies is $\beta(0) = 4$, $\beta(4) = 8$, $\beta(8) = 12$, $\beta(12) = 16$, $\beta(16) = 0$, which completes the first cycle, $\beta(1) = 5, \ldots, \beta(17) = 1$, completing the second cycle, etc. The reader might want to verify that β is in fact an automorphism—for example, it maps the two adjacent node pairs (0, 7) to the adjacent node pair (4, 11). If α is an automorphism of G, then so is each element of the set subgrp(α) $= \{e, \alpha, \alpha^2, \ldots, \alpha^{o(\alpha)-1}\}$, where $o(\alpha)$ is the least integer i such that $\alpha^i = e$, the identity permutation, and is called the *order* of α. Here, α^p is recursively defined as $\alpha^p = \alpha \circ \alpha^{p-1}$, where $\circ$ denotes function composition. For instance, $\alpha = (0, 1, 2, \ldots, 18, 19)$ is a single-cycle (containing only one cycle) automorphism of the graph J of Fig. 2a, while α^4 is the automorphism β specified above. subgrp(α) is a subgroup of aut(G) and is said to be *generated* by α.

Finally, if A is a set of integers, and r is an integer, then $A + r$ is defined as the set $\{a + r : a \in A\}$. $A \hat{+} r$ is defined as $A \cup (A + r)$.

3. AUTOMORPHISMS AND FAULT TOLERANCE

In this section, we relate graph automorphisms to the construction and reconfiguration of $G'[k, G]$. The basic principle that comes into play here is illustrated by Fig. 3. The fault-tolerant supergraph G' of G should be constructed so that each node of G has a number of other nodes, including some spare nodes, that are similar to it in G'. When a node u becomes faulty it can be mapped to, i.e., replaced by, a similar node v. At the same time, neighbors u_1, u_2, and u_3 of u in G are replaced by similar neighbors v_1, v_2, and v_3, respectively, of v in G'. Furthermore, v has to be replaced by some node w that is similar to it, etc., until a spare node s is used to replace a similar node x in this replacement sequence. The spare s is then mapped to the faulty node u. This sequence of replacements in a reconfiguration process like this can be represented as a mapping between the nodes by an automorphism α of G'. The faulty node is labeled as a spare node by α, while the remaining nonfaulty nodes are labeled as nodes of G and hence determine a fault-free subgraph isomorphic to G.

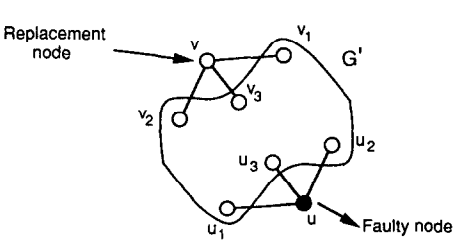

FIG. 3. Illustration of the basic principle.

More formally, we wish to map a spare node in G' to a faulty node using an automorphism of G'. This mapping relabels the faulty nodes as spares, and thus the rest of the nodes determine a fault-free copy of G. The problem now is to construct G' so that it has the requisite automorphisms. A direct way to do this is to add k spare nodes and edges to G so that the automorphism group of the resulting supergraph G' has the property that for every subset F of k nodes of $V(G')$, there is an automorphism α of G' that maps the set S of k spare nodes to F. Then, if F is a set of faulty nodes, α maps all the nodes of G to nonfaulty nodes of G'. This means that the graph induced by the nonfaulty nodes of G' contains a subgraph isomorphic to G. Reconfiguration can then be accomplished by identifying those nodes to which the nodes of G have been mapped by α and activating only those edges that correspond to the edges of G under this new mapping.

It can be easily shown that aut(G') can map a specified subset S of k spare nodes of G' to any other subset F of k nodes if and only if aut(G') is k-subtransitive. We have not found any characterization of k-subtransitive automorphism groups of graphs in the literature on graph automorphisms [4, 5, 1, 29]. The closest characterization is that of a k-transitive automorphism group. However, k-transitivity is a much stricter requirement than what we need, leading immediately to the following theorem.

THEOREM 1. *Let G' be a supergraph of G with spare nodes $\{s_0, s_1, \ldots, s_{k-1}\}$. If* aut($G'$) *is k-transitive, then G' is a k-FT supergraph of G and can be reconfigured around any k faulty nodes by an automorphism of G'.*

Thus one way of constructing G' is to require that aut(G') be k-transitive. It is easily shown that in this case, for $k \geq 2$, aut(G') quickly becomes $S_{|V(G')|}$, i.e., G' becomes the complete graph. This is obviously an excessively costly k-FT supergraph. Though requiring aut(G') to be k-subtransitive is a significantly weaker restriction than requiring aut(G') to be k-transitive, it means that every set of k nodes of G' has to induce isomorphic subgraphs in G', which offhand also seems to be a severe requirement. In fact, as stated in the

next theorem, a 2-FT supergraph G' of a nonempty graph constructed to meet this requirement must be complete.

THEOREM 2. *If G' is a 2-FT supergraph of a nonempty graph G on N nodes, such that* aut(G') *is 2-subtransitive, then $G' = K_{N+2}$, the complete graph on $N+2$ nodes.*

Proof. Since G is not the empty graph on N nodes, it has at least one pair $\{v_1, v_2\}$ of adjacent nodes. Let $\{s_1, s_2\}$ be the pair of spare nodes of G'. Then there is an $\alpha \in \text{aut}(G')$ such that $\alpha\{s_1, s_2\} = \{v_1, v_2\}$. This implies that s_1 and s_2 are adjacent nodes. For every pair of nodes $\{x_1, x_2\}$ of G' there is some automorphism mapping the pair of adjacent nodes $\{s_1, s_2\}$ to them. Hence every pair of nodes of G' are adjacent, implying that $G' = K_{N+2}$. ■

Surprisingly, it is much more difficult to generalize this result—we have not been able to do so—to the k-FT case, i.e., to prove that $G'[k, G] = K_{N+k}$, if G is not the empty graph on N nodes and if aut(G') is k-subtransitive. This is because for any graph H, the fact that aut(H) is k-subtransitive does not necessarily imply that aut(H) is i-subtransitive for any i, $2 \leq i \leq k-1$. However, it is easy to see that if aut(H) is transitive on k-ordered subsets of $V(H)$, then it is also transitive on i-ordered subsets of $V(H)$, for all i, $1 \leq i \leq k-1$. This could be the reason why group theorists, as far as we know, have dealt only with the k-transitivity of a permutation group P, and not with what seems to be the more natural phenomenon of the k-subtransitivity of P.

The above sufficient conditions of k-transitivity and k-subtransitivity to ensure k-fault tolerance assume that exactly k spare nodes are present. If more than k spare nodes are present in a k-FT supergraph, then it might be possible to relax these conditions substantially. To give an extreme example, if we have N spare nodes in a 1-FT supergraph, then its automorphism group need not be 1-transitive (or simply transitive) at all. The use of more than k spare nodes to achieve k-fault tolerance falls within the area of local-sparing designs alluded to earlier and discussed in greater detail in [9]. In this paper, we restrict our attention to k-FT designs using exactly k spare nodes. This is not an unreasonable restriction and, in many cases, for example, when the processors are complex and expensive, it is a desirable and practical one.

We apply the term *automorphic reconfiguration* to reconfiguration schemes that use automorphisms of the fault-tolerant supergraph to map the spare nodes to the faulty nodes. Hence, automorphic reconfiguration also maps the nodes of the basic graph G to nonfaulty nodes of the supergraph G' to obtain a nonfaulty copy of G. Similarly, fault-tolerant design schemes that construct $G'[k, G]$ with automorphic reconfiguration of G' as an objective are termed *automorphic designs.*

THEOREM 3. *If automorphic reconfiguration can be used to reconfigure a k-FT supergraph $G'[k, G]$ with k spare nodes around any k faults in G, then G' is node-symmetric.*

Proof. Suppose that G' is not node-symmetric. Let $O_1, O_2, \ldots, O_t$ be the orbits of G', where $t > 1$, and let k_i be the number of spare nodes in orbit O_i. There has to be some i for which $k_i < k$ and $|O_i| > k_i$. If t nodes in O_i fail, where $k_i < t \leq k$, then no automorphism of G' maps all spare nodes in O_i to all the faulty nodes in O_i. Hence, for any automorphism α, at least one of the active nodes of O_i is mapped to a faulty node of O_i, making reconfiguration unsuccessful. ■

Node symmetry is, of course, not sufficient to guarantee reconfiguration. With this in view, as well as the severity of the requirement of k-subtransitivity of aut(G'), we present an iterative automorphic design and reconfiguration scheme.

We first construct a node-symmetric supergraph G_1 of $G_0 = G$ with one spare node s_0. Since this means that aut(G_1) is transitive on 1-subsets of $V(G_1)$, G_1 is a 1-FT supergraph of G. We then construct a node-symmetric supergraph of G_2 of G_1, with one more spare node s_1. By the same reasoning as that used for G_1, $G_2 = G'[1, G_1] = G'[2, G]$, so we obtain a 2-FT supergraph of G. Continuing this way, we eventually obtain a node-symmetric supergraph G_k of G_{k-1}; $G_k = G'[1, G_{k-1}] = G'[k, G]$. Note that if H is an i-FT supergraph of G, and H' is a j-FT supergraph of H, then H' is also an $(i+j)$-FT supergraph of G.

An important point to make here is that aut(G_k) is only 1-transitive and not necessarily k-transitive or k-subtransitive. Thus, the iterative design procedure provides us with a means of constructing a k-FT supergraph whose automorphism group is only 1-transitive! In general, the degree of transitivity or subtransitivity of the automorphism groups of the k-FT supergraphs that we construct is of only minor interest.

The iterative reconfiguration strategy for $G_k = G'[k, G]$ is as follows. Let $\{s_0, s_1, \ldots, s_{k-1}\}$ be the spare nodes in G_k constructed as explained above, where s_i is the spare node used to construct a 1-FT node-symmetric supergraph G_{i+1} of G_i. Suppose that $t \leq k$ faults $\{f_0, f_1, \ldots, f_{t-1}\}$ occur. Considering f_{t-1} to be a fault in G_k, we use an automorphism that maps s_{k-1} to f_{t-1} to obtain a copy of G_{k-1} with fault set $\{f_0, \ldots, f_{t-2}\}$. We do the same with s_{k-2} and f_{t-2} in this copy of G_{k-1} (using an automorphism of G_{k-1}) to obtain a copy of G_{k-2} with fault set $\{f_0, \ldots, f_{t-3}\}$. By continuing in this manner, we obtain a nonfaulty copy of G_{k-t} which contains G. Note that this iterative automorphic reconfiguration scheme is inherently incremental.

4. CIRCULANT GRAPHS AND FAULT TOLERANCE

In this section, we use automorphisms to construct a 1-FT node-symmetric supergraph G_1 for a useful class of graphs termed circulant. We first characterize the structure of subgraphs induced in any graph G by a cycle in an automorphism of G. This characterization helps in defining the

structure of fault-tolerant supergraphs of circulant graphs in a regular way.

LEMMA 1. *Let $A = (a_0, a_1, \ldots, a_{t-1})$ be a cycle in an automorphism α of a graph G. For any $a_i \in A$, if $a_{i+j \pmod t}$ is adjacent to a_i then so is $a_{i-j \pmod t}$, where $-t \leq j \leq t$. Moreover, for every $a_r \in A$, the nodes $a_{r \pm j \pmod t}$ are adjacent to a_r.*

Proof. All subscript addition is taken mod t in this proof. Let $a_r \neq a_i$ be any other node in A. Then, α^{r-i} maps node a_i to a_r, as well as a_{i+j}, a neighbor of a_i to node $a_{i+j+r-i} = a_{r+j}$. Hence node a_{r+j} is a neighbor of a_r, for every $a_r \in A$. Now by taking a_{i-j} to be a_r, we see that a_{i-j} is adjacent to a_i, as required. ■

If $a_0, a_1, \ldots, a_{t-1}$ are represented by equidistant consecutive points on a circle C, then by Lemma 1, the neighbors of any node a_i are placed symmetrically on C about the diameter through a_i. We define the *circular distance* $\mathrm{cd}(u, v)$ between any two nodes u and v on C as one plus the number of nodes lying on the smaller arc of C subtended by chord (u, v). Figure 2a shows such a representation of the subgraph J induced by the cycle $(0, 1, \ldots, 19)$ of an automorphism of some graph G. The circular distance between nodes 18 and 2, for example, is 4. The circular distances in J of the neighbors of each node u from it are 4, 5, and 7 in the clockwise, as well as the counterclockwise directions. We call such a representation of the subgraph induced by a cycle A of an automorphism of G a (*symmetric*) *embedding* of A on a circle and denote it by $\mathrm{circ}(A)$.

$\mathrm{circ}(A)$ can be characterized by its *distance sequence* $\mathrm{ds}(A)$, which is the ordered set, increasing from left to right, of the circular distances between any node a_i in A and all its neighbors in one of the semicircles formed in $\mathrm{circ}(A)$ by the diameter through a_i. The distance sequence is well defined, since by Lemma 1, every node in $\mathrm{circ}(A)$ has neighbors at the same circular distances from it, in both the clockwise and the counterclockwise directions. Thus the distance sequence for the cycle of Fig. 2a is (4, 5, 7). If $|\mathrm{ds}(A)| = m$, then the degree of each node in $G(A)$ is either $2m - 1$ or $2m$. A maximal subsequence of consecutive integers in $\mathrm{ds}(A)$ is called a *block* of $\mathrm{ds}(A)$. Thus if $\mathrm{ds}(A) = (2, 3, 4, 7, 9, 10, 14, 15)$, then its blocks are [2, 3, 4], [7], [9, 10], and [14, 15].

In special cases, an automorphism α of G may consist of a single cycle. We then can talk of embedding α, and hence G, on a circle in the above sense. A graph G that has a cycle containing $N = |V(G)|$ nodes as an automorphism is called a *circulant graph* [6] and can be embedded on a circle in the above manner. The order in which to place the nodes of G on the circle is provided by the single-cycle automorphism α of G, in which case G is said to be embedded in a circle according to α. Such an embedding of G is denoted by $\mathrm{circ}_\alpha(G)$. However, when there is no confusion regarding which single-cycle automorphism of G is being used to embed G on a circle, we simply denote the embedding by $\mathrm{circ}(G)$. Moreover, by the distance sequence $\mathrm{ds}(G)$ of a circulant graph G, we implicitly mean the distance sequence of the single-cycle automorphism α of G according to which it is embedded on a circle. The graph J in Fig. 2a can also be viewed as a circulant graph with 20 nodes embedded in a circle according to the automorphism $(0, 1, 2, \ldots, 19)$, with $\mathrm{ds}(J) = (4, 5, 7)$.

An edge $\{u, v\}$ of G is said to *cross* a node w in $\mathrm{circ}(G)$ if w lies on the smaller arc of $\mathrm{circ}(G)$ subtended by chord (u, v). Thus in Fig. 2a, edge $\{2, 6\}$ crosses nodes 2, 3, 4, 5, 6. If $\{u, v\}$ is a diameter of $\mathrm{circ}(G)$, then it is said to cross all nodes encountered in the clockwise direction on the circle from u to v, if $u < v$, and in the counterclockwise direction otherwise. The circular distance of an edge $\{u, v\}$ in $\mathrm{circ}(G)$ is the circular distance between u and v and is denoted by $\mathrm{cd}(u, v)$. Hence in Fig. 2a, $\mathrm{cd}(2, 6) = 4$. It should be clear from our discussion that the circular distance of any edge in $\mathrm{circ}(G)$ is one of the integers in $\mathrm{ds}(G)$, and that for every d_j in $\mathrm{ds}(G)$, there are two edges of circular distance d_j incident on every node of G. The following lemma gives an obvious characterization of a circulant graph.

LEMMA 2. *A graph G is circulant if and only if it can be embedded in a circle.*

Proof. The necessity follows from the above discussion. Suppose that G can be embedded in a circle, and let $(v_0, v_1, \ldots, v_{N-1})$ be the order in which the nodes of G are arranged in $\mathrm{circ}(G)$. Then $\alpha = (v_0, v_1, \ldots, v_{N-1})$ is an automorphism G, since for every pair of nodes (v_i, v_{i+1}) $(\alpha(v_i) = v_{i+1})$, if v_{i+j} is a neighbor of v_i, by the definition of circle embedding, v_{i+j+1} is a neighbor of v_{i+1} $(\alpha(v_{i+j}) = v_{i+j+1})$. ■

A necessary condition for G to be circulant is that it is node-symmetric. A sufficient condition for G to be circulant, which is due to Turner [27], is that G is node-symmetric and that $|V(G)|$ is a prime. There are other node-symmetric graphs with a nonprime number of nodes that are also circulant. The cycle C_N with an arbitrary number N of nodes, its power graph C_N^t for any $1 < t < N$, and, of course, the complete graph K_N and the empty graph E_N are examples.

If G is a circulant graph, then it is regular with node degree d. Suppose G is embedded on a circle according to the single-cycle automorphism $\alpha = (v_0, v_1, \ldots, v_{N-1})$ in $\mathrm{aut}(G)$. Let its distance sequence be $(d_0, d_1, \ldots, d_{m-1})$, where $m = \lceil d/2 \rceil$. We now describe the construction of a node-symmetric circulant supergraph G_1 of G with one spare node s_0. Insert s_0 between any two nodes, say v_i and v_{i+1} on $\mathrm{circ}(G)$, and connect s_0 to the nodes of G as explained shortly. The circular distances of all edges of G that cross s_0 in $\mathrm{circ}(G_1)$ increase by one. This is shown in Fig. 2b for the graph of Fig. 2a, where the circular distances of edges (0, 16), (0, 15), and (0, 13) increase by one, while those of edges (0, 4), (0, 5), and (0, 7) remain the same as in $\mathrm{circ}(G)$. In general, for every d_i in $\mathrm{ds}(G)$, there is an edge $\{u, v\}$ with $\mathrm{cd}(u, v)$

$= d_i$, which crosses s_0. Hence $\mathrm{cd}(u, v)$ becomes $d_i + 1$ in $\mathrm{circ}(G_1)$. Similarly, for every d_i in the distance sequence of G, there is an edge $\{x, y\}$ such that $\mathrm{cd}(x, y) = d_i$ and $\{x, y\}$ does not cross s_0. Thus $\mathrm{circ}(G_1)$ has edges of G with circular distances d_i and $d_i + 1$ for each d_i in $\mathrm{ds}(G)$. Hence, to make G_1 a circulant, and thus node-symmetric, supergraph of G, we insert edges with circular distances in $\mathrm{ds}(G_1) - \mathrm{ds}(G)$ at each node of G, as well as edges with circular distances in $\mathrm{ds}(G_1)$ incident on the spare node s_0. Here $\mathrm{ds}(G_1)$ is given by $\{c_i$: either c_i or $c_i - 1$ is in $\mathrm{ds}(G)\} = \mathrm{ds}(G) \hat{+} \{1\}$. In Fig. 2b, edges have been added in this manner to construct J_1 for the graph J in Fig. 2a; for clarity, only the edges incident on nodes s_0 and 0 are shown. It can be seen that $\mathrm{ds}(J_1) = \mathrm{ds}(J) \hat{+} \{1\} = (4, 5, 6, 7, 8)$.

If there are b blocks in $\mathrm{ds}(G)$, then $|\mathrm{ds}(G_1)| = |\mathrm{ds}(G)| + b$, and hence the node degree of G_1 is $d + 2b$. To see this, let $D = [d_i, d_{i+1} = d_i + 1, \ldots, d_{i+j} = d_i + j]$ be a block in $\mathrm{ds}(G)$. In the corresponding block in $\mathrm{ds}(G_1)$, there are edges with circular distances d_i, $d_i + 1, \ldots, d_i + j, d_i + j + 1$. Hence exactly one integer is added to each block in $\mathrm{ds}(G)$ to make up the blocks of $\mathrm{ds}(G_1)$. Thus in the worst case, when all blocks in $\mathrm{ds}(G)$ have a single element, the cardinality of $\mathrm{ds}(G_1)$ is double that of $\mathrm{ds}(G)$, and hence the node degree of G_1 also doubles. In the best case, there is a single block in $\mathrm{ds}(G)$, and $|\mathrm{ds}(G_1)| = |\mathrm{ds}(G)| + 1$. As shown later in Theorem 6, this means that the node degree of G_1 is either $d + 1$ or $d + 2$.

If s_0 is inserted between v_i and v_{i+1} in $\mathrm{circ}(G)$, then $\beta = (v_0, v_1, \ldots, v_i, s_0, v_{i+1}, v_{i+2}, \ldots, v_{N-1})$ is an automorphism of G_1 containing a single cycle. This is the only automorphism that we need for reconfiguration purposes, since for any fault f in G_1, there is an automorphism β^t in the subgroup $\mathrm{subgrp}(\beta)$ of $\mathrm{aut}(G_1)$ generated by β, which maps s_0 to f. As explained earlier, β^t can be used to relabel the nodes of G_1 to reconfigure around fault f. Note that $\mathrm{subgrp}(\beta)$ is the set of all mappings required to reconfigure G_1 around any 1-fault, and that it is sufficient to store only the automorphism β to represent these mappings. Figure 4 shows the reconfiguration of J_1 around fault 17, using the automorphism $(0, 1, 2, \ldots, 10, s_0, 11, 12, \ldots, 19)^7$ to relabel the nodes of J_1. The nonfaulty copy of J is shown with dark edges, and the new labels of each node are shown near the node; the old labels are shown in italics and further away from the nodes.

We now describe the construction of the general k-FT circulant supergraph G_k of a circulant graph G. As explained in Section 3, we can construct G_k by iteratively constructing 1-FT supergraphs of G, $G_1, \ldots, G_{k-1}$ in the manner given in the preceding paragraph. However, we can also construct G_k directly from G by "flattening" the iterative process as shown in the algorithm SUPER_CIRCULANT in Fig. 5. The k-FT supergraph of G constructed by SUPER_CIRCULANT is denoted by $G'_{\mathrm{auto}}[k, G]$. Note that SUPER_CIRCULANT is an incremental design procedure, since an m-FT supergraph G_m of G can be constructed from a k-FT supergraph G_k of G, where $m > k$, by invoking SUPER_CIRCULANT with parameters $m - k$ and G_k. The supergraph G_m obtained in this case is isomorphic to the supergraph obtained by calling SUPER_CIRCULANT with parameters m and G.

Figure 6 shows part of a 3-FT graph $H_3 = G'_{\mathrm{auto}}[3, H]$ of the basic graph H, which has 18 nodes and distance sequence $(5, 6)$. For clarity, only edges incident on node 0 are shown. Note that the edges to nodes 5 and 6 constitute edges of H and are dark, while the edges to nodes 7, 8, and 9 are spare edges of H_3 and are light. Similarly, the edges to 13 and 12 constitute edges of H, since ignoring spare node s_2, it can be seen that $\mathrm{cd}(0, 13) = 5$, and $\mathrm{cd}(0, 12) = 6$ in $\mathrm{circ}(H)$.

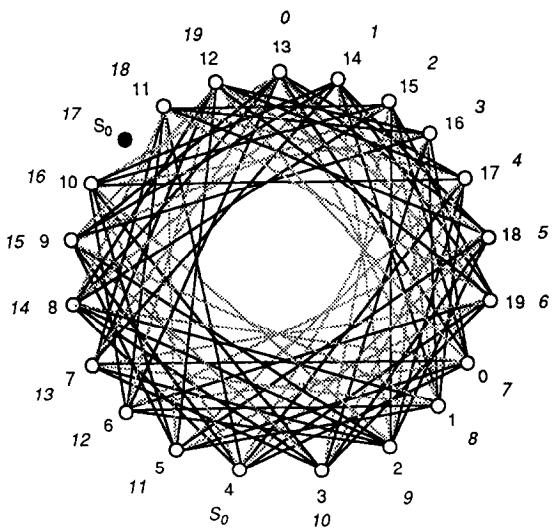

FIG. 4. Reconfiguration of the 1-FT supergraph J_1 of Fig. 2b around faulty node 17.

```
Procedure SUPER_CIRCULANT(k, G);
/* G is a circulant graph with N nodes, whose k-FT supergraph
G_k = G'_auto[k, G] with k spare nodes is constructed by SUPER_CIRCULANT */
begin
    Find a single-cycle automorphism α of G, and embed G in a circle according to α;
    ds(G_k) := ds(G);
    for i = 1 to k do ds(G_k) := ds(G_k) ∔ {1};
    Insert k spare nodes {s_0, s_1, ..., s_{k-1}} anywhere in circ(G);
    for each node v_i in G do
        for each d_j ∈ ds(G_k) − ds(G) do
            Add edges from v_i to nodes at circular distance d_j from it in
            the clockwise and counterclockwise directions;
    for each spare node s_i in G_k do
        for each d_j ∈ ds(G_k) do
            Add edges from s_i to nodes at circular distance d_j from it in
            the clockwise and counterclockwise directions;
    return(G_k);
end; /* Procedure SUPER_CIRCULANT */
```

FIG. 5. Procedure for constructing a k-FT supergraph of a circulant graph.

The edges to nodes 14, s_1, and 11 are spare, as cd(0, 14) = 4 and cd(0, 11) = 8 in circ(H).

Next we show how $G_k = G'_{\text{auto}}[k, H]$ can be automorphically reconfigured to $G_{k-t} = G'_{\text{auto}}[k-t, G]$ in the presence of $t \leqslant k$ faults. Without loss of generality, assume that the spare nodes $\{s_0, s_1, \ldots, s_{k-1}\}$ are all inserted on circ(G) between nodes v_i and v_{i+1} of G. Then, as explained earlier, $\alpha = (v_0, v_1, \ldots, v_i, s_0, s_1, \ldots, s_{k-1}, v_{i+1}, v_{i+2}, \ldots, v_{N-1})$ is an automorphism of G_k. Let the fault set be $F = \{f_0, f_1, \ldots, f_{t-1}\}$, and let t_{k-1} be the circular distance between node s_{k-1} and f_{t-1} in circ(G_k). Use automorphism $\beta_k = \alpha^{t_{k-1}}$ of G_k to map s_{k-1} to f_{t-1} and reconfigure to obtain a copy of $G_{k-1} = G'_{\text{auto}}[k-1, G]$ with $t-1$ faults. This copy of G_{k-1} can be identified by discarding the node labeled s_{k-1} under mapping β_k and by considering the other nodes in the circular order thus obtained. Under the new labeling this circular order is $(v_0, \ldots, v_i, s_0, \ldots, s_{k-2}, v_{i+1}, \ldots, v_{N-1})$, which also defines the ordering of circ(G_{k-1}). Discard the edges in circ(G_{k-1}) that are at circular distances d_i not in ds(G_{k-1}).

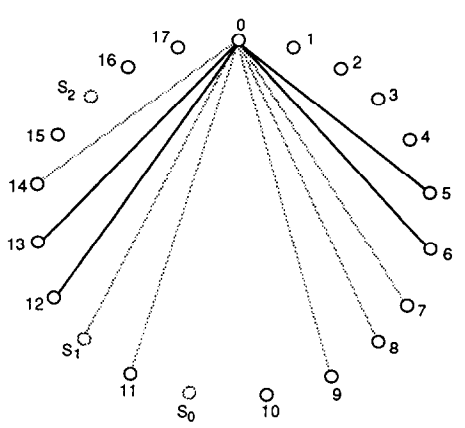

FIG. 6. Part of a 3-FT supergraph $H_3 = G'_{\text{auto}}[3, H]$ of the circulant graph H with 18 nodes and distance sequence (5, 6).

The rest of the edges and the nonfaulty nodes determine the required copy of G_{k-1}, while $\alpha_1 = (v_0, \ldots, v_i, s_0, \ldots, s_{k-2}, v_{i+1}, \ldots, v_{N-1})$ is the required automorphism of G_{k-1}. Again, let t_{k-2} be the circular distance between newly labeled node s_{k-2} and fault f_{k-2}. Use $\beta_{k-1} = \alpha_1^{t_{k-2}}$ to map s_{k-2} to f_{k-2}, and determine a copy of $G_{k-2} = G'_{\text{auto}}[k-2, G]$ with $t-2$ faults as described above. By repeating this procedure $t-2$ more times, we obtain a fault-free copy of G_{k-t}. When $l \leqslant k-t$ more faults occur, we can reconfigure our fault-free copy of G_{k-t} in the same manner to obtain a fault-free copy of G_{k-t-l} and similarly determine a fault-free subgraph G in it. Hence reconfiguration is incremental; as pointed out before, this is an inherent and valuable characteristic of all iterative automorphic reconfiguration schemes.

Figure 7a shows three faults in the supergraph H_3 of Fig. 6, while Figs. 7b, 7c, and 7d show the reconfiguration of H_3, H_2, and H_1 into a copy of H_2 with two faults, a copy of H_1 with one fault, and a fault-free copy of H, respectively. As before, the new labels of nodes in the reconfigured graphs are shown near the nodes, while the previous labels are shown further away and in italics. The preceding discussion leads to the following theorem.

THEOREM 4. *The supergraph $G'_{\text{auto}}[k, G]$ of a circulant graph G constructed by SUPER_CIRCULANT is a k-FT supergraph of G.*

The next two theorems give bounds on the node degree of $G'_{\text{auto}}[k, G]$ in terms of the number of blocks of ds(G), the node degree of G, and k.

THEOREM 5. *Let G be a circulant graph with node degree d and with b blocks in* ds(G). *Then the node degree of $G_k = G'_{\text{auto}}[k, G]$ is between $d + 2(b + k - 1)$ and $d + 2bk$.*

Proof. Let ds(G) = $(d_1, \ldots, d_t)$, and consider the ordered set $(g_1, \ldots, g_{t-1})$, where $g_i = d_{i+1} - d_i - 1$. Then, it can be seen from SUPER_CIRCULANT that $|\text{ds}(G_k)|$

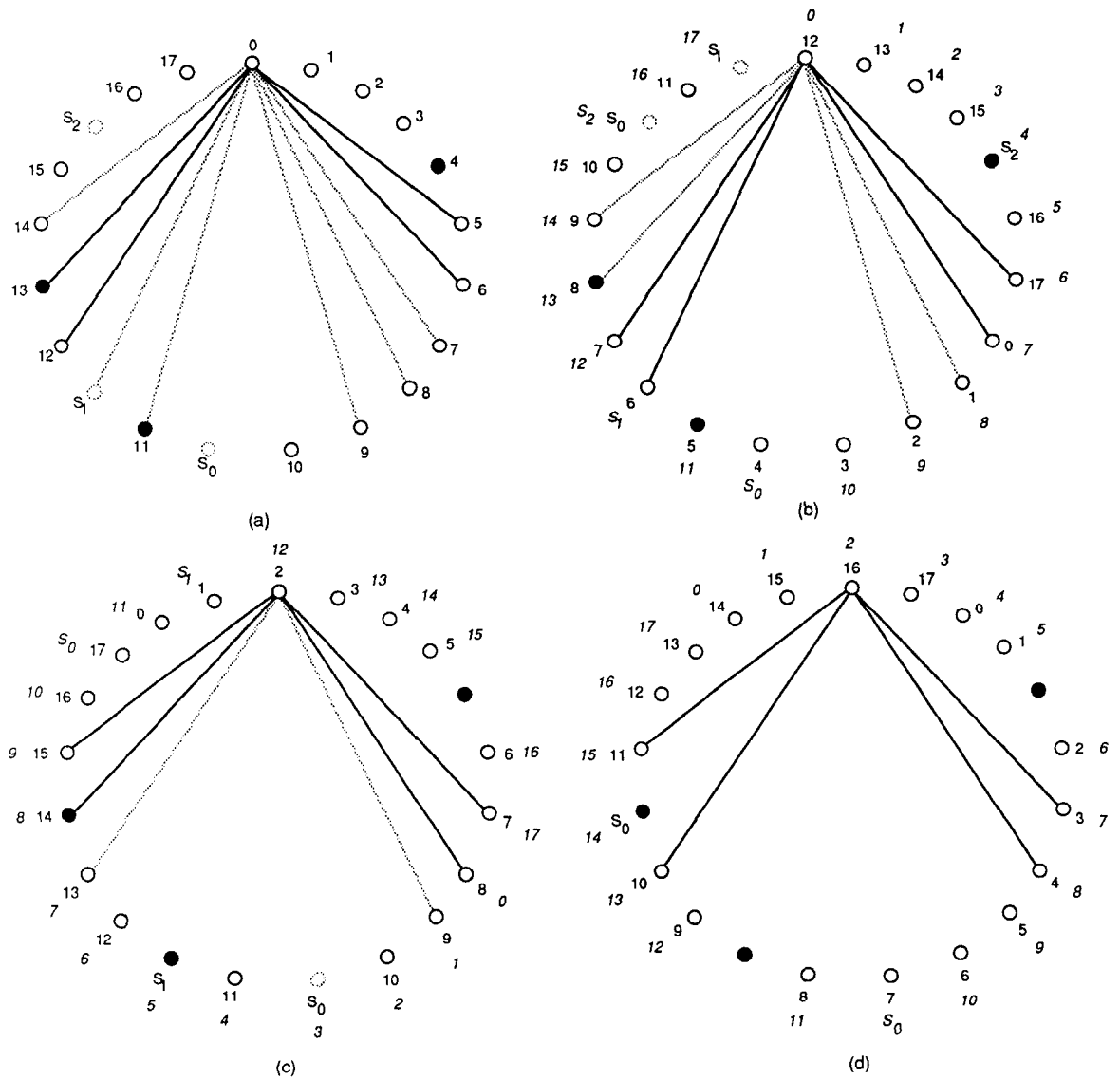

FIG. 7. (a) H_3 of Fig. 6 with three faults. (b) Reconfiguration to a copy of H_2 with two faults. (c) Reconfiguration to a copy of H_1 with one fault. (d) Reconfiguration to a fault-free copy of H.

$= |\mathrm{ds}(G)| + \sum_{i=0}^{b-2} \min(k, g_i) + k$; thus the node degree of G_k increases by twice that amount. In the worst case, each $g_i \geq k$ and bk integers are added to $\mathrm{ds}(G)$ to form $\mathrm{ds}(G_k)$. In the best case, each $g_i = 1$ and we have to add $b + k - 1$ integers to $\mathrm{ds}(G)$ in order to form $\mathrm{ds}(G_k)$. ■

In the worst case, the number of blocks $b = d/2$, and $G'_{\mathrm{auto}}[k, G]$ has node degree $(k+1)d$. Taking into account the fact that the reliability gain will ultimately level off as k increases, and that the value of k at which this happens is small compared to N, as suggested by empirical evidence, we only design k-FT supergraphs with $k \ll N$. (Local sparing is one solution to alleviating this reliability saturation problem, but that again is another topic). Hence it is extremely unlikely that the $G'_{\mathrm{auto}}[k, G]$'s of interest will be complete graphs. This is especially true when G is sparse, as are all multiprocessor graphs. For example, if G is a hypercube with $N = 2^n$ nodes, then $d = n$, and $G'_{\mathrm{auto}}[k, G]$ will be complete only if $k \approx N/\log N$, which is a very large value for k. A more reasonable range of values for k is in the neighborhood of $\log N$, and it is easy to see that $G'_{\mathrm{auto}}[k, G]$ will never be complete in such cases. We are not concerned any more with whether G' is complete, but only with whether it is efficient and practical to realize, which of course means that G' has to be reasonably sparse. In a later section, we discuss a hardware-efficient implementation of G' using switches and shared spare links that substantially reduces the link complexity.

THEOREM 6. *Let G be a circulant graph with a distance sequence containing one block. Then the node degree of G_k*

$= G'_{\text{auto}}[k, G]$ is between k and $2k$ more than the node degree of G.

Proof Outline. With $b = 1$, Theorem 5 requires $2k$ extra edges per node in G_k. However, this assumes that the largest integer d_{m-1} in $\text{ds}(G)$ is less than or equal to $\lfloor N/2 \rfloor - k/2$, where for simplicity, we assume that k is even. Suppose now that $d_{m-1} = \lfloor N/2 \rfloor - l$, where $k > 2l$. Then, it can be shown that the number of circular distances added to $\text{ds}(G)$ to form $\text{ds}(G_k)$ is $2l + (k - 2l)/2$. This represents an addition of $4l + k - 2l = k + 2l$ to the node degree of G_k over that of G. By taking $l = 0$, we obtain the lower bound stated in the theorem. ■

In any k-FT supergraph $G'[k, G]$ of a regular graph G (not necessarily circulant) with exactly k spare nodes, the node degree of each node should be at least $d + k$, where d is the node degree of G. Hence Theorem 6 implies that for the case where G is a circulant graph with one block and $\lfloor N/2 \rfloor$ is the largest circular distance in $\text{ds}(G)$, $G'_{\text{auto}}[k, G]$ is optimal with respect to the number of spare edges and the maximum node degree.

5. APPLICATION TO CYCLES

In this section, we apply the automorphic method to the simple case of cycle graphs. This application illustrates the automorphic method's ability to construct optimal k-FT supergraphs and also gives us an example of Theorem 6.

We denote a cycle graph with N nodes by C_N. Figure 8a shows a cycle graph C_{15} with 15 nodes. It is easily seen that C_N is a circulant graph, and if $L = (0, 1, \ldots, N-1)$ is the cycle in C_N, then $\alpha = L$ is a single-cycle automorphism of C_N. The *power graph* C'_N of a cycle graph C_N is defined as an edge supergraph of C_N in which each node u has edges $\{(u \pm i) \pmod N : 1 \leqslant i \leqslant t\}$ incident on it. The power-graph construction method has been well known for constructing k-FT supergraphs of cycle graphs [12]. In this method, the power graph C^{k+1}_{N+k} of the cycle C_{N+k} is constructed to obtain a k-FT supergraph of C_N. Note that if the automorphism $\alpha = L$ is used to construct a k-FT supergraph of C_N by invoking SUPER_CIRCULANT(k, C_N), then the resulting supergraph is also C^{k+1}_{N+k}. Note that the distance sequence of C_N when arranged on a circle according to α is (1), while that of C^{k+1}_{N+k} is $(1, 2, \ldots, k+1)$. Figure 8b shows the power graph C^3_{17}, which is a 2-FT realization of the cycle C_{15}. The number of spare edges per node in C^{k+1}_{N+k} is $2k$, which corresponds to the upper bound of Theorem 6—the distance sequence of C_N is a trivial single block.

Instead of embedding C_{15} according to $\alpha = (0, 1, \ldots, 14)$, as has been done in Fig. 8a, let us embed its nodes according to the automorphism $\beta = \alpha^2 = (0, 2, 4, 6, 8, 10, 12, 14, 1, 3, 5, 7, 9, 11, 13)$; see Fig. 8c. In this case, the distance sequence of C_{15} becomes $(\lfloor N/2 \rfloor = 7)$, and the two neighbors of each node are consecutive to each other. This

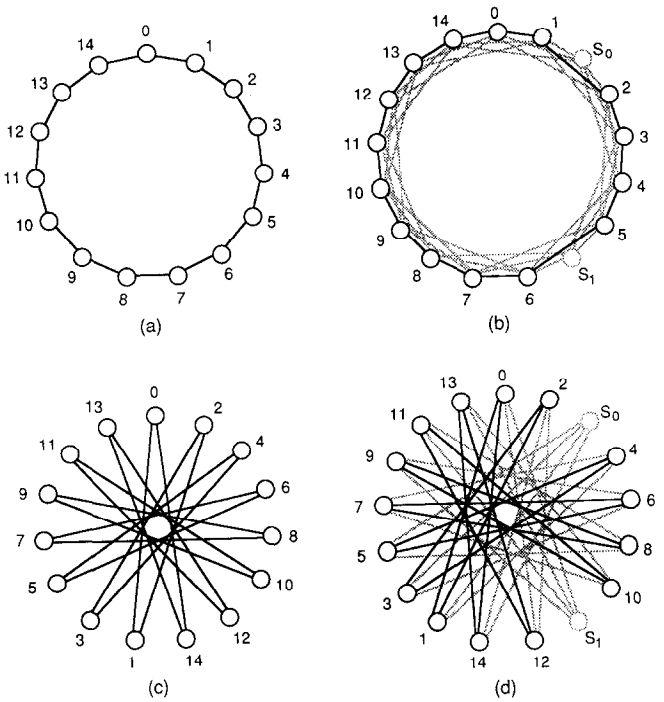

FIG. 8. (a) The cycle graph C_{15} with 15 nodes. (b) A 2-FT supergraph C^3_{17} of C_{15} constructed by the power-graph construction method. (c) An embedding of C_{15} on a circle according to the automorphism $\beta = (0, 2, 4, 6, 8, 10, 12, 14, 1, 3, 5, 7, 9, 11, 13)$. (d) The 2-FT supergraph C'_{17} of C_{15} constructed by SUPER_CIRCULANT $(2, C_{15})$ using automorphism β.

also satisfies the condition given the proof of Theorem 6 for a k-FT supergraph to have only k spare edges per node. On applying SUPER_CIRCULANT$(1, C_{15})$ with β as the automorphism, we obtain a 1-FT supergraph C'_{16} of C_{15} with distance sequence (7, 8). The circular distance 8 corresponds to a diameter in $\text{circ}(C'_{16})$, and hence each node has only one neighbor at circular distance 8. Thus only one spare edge per node has been added to construct C'_{16}. If we further apply SUPER_CIRCULANT$(1, C'_{16})$ to construct a 2-FT supergraph of C_{15}, then the distance sequence of the resulting supergraph C'_{17} remains (7, 8). However, an edge with circular distance 8 is no longer a diameter in $\text{circ}(C'_{17})$, and each node has two neighbors at circular distance 8. Thus we add one more edge per node to C'_{16} to construct C'_{17}, which is shown in Fig. 8d. Note that while there are four spare edges per node in the 2-FT supergraph C^3_{17} of C_{15} (Fig. 8b), there are only two spare edges per node in the optimal 2-FT supergraph C'_{17}. From the discussion in the previous section, it follows that C'_{17} is an optimal 2-FT supergraph of C_{15}. Further, SUPER_CIRCULANT(k, C_{15}) also constructs an optimal k-FT supergraph of C_{15} when automorphism β is used.

In general, to construct such optimal k-FT supergraphs of a cycle graph, we should use that single-cycle automorphism in subgrp(α) in which the two neighbors of a node are placed at circular distance $\lfloor N/2 \rfloor$ from it. Here α is the

automorphism whose cycle corresponds to the cycle L in C_N. Suppose the automorphism that accomplishes this is α^x; then x satisfies the following conditions: (1) x is relatively prime to N, and (2) either $\lfloor N/2 \rfloor x = 1 \pmod N$ or $\lfloor N/2 \rfloor x = -1 \pmod N$. Such an x may not exist for a given N. For example, x does not exist when N is even. However, in this case, it might be possible to use an automorphism α^x in which the two neighbors of each node in C_N are placed at clockwise and counterclockwise circular distance $N/2 - 1$. Besides the relative prime condition, x now has to satisfy either $(N/2 - 1)x = 1 \pmod N$ or $(N/2 - 1)x = -1 \pmod N$. If this is possible, then applying SUPER_CIRCULANT(k, C_N) with α^x as the automorphism results in a k-FT supergraph with $k + 1$ spare edges per node. Consider the cycle C_8, where $L = (0, 1, \ldots, 7)$ is the cycle in the graph; let $\alpha = L$. Then, α^3 is the automorphism (0, 3, 6, 1, 4, 7, 2, 5) in which the two neighbors of each node occur at a circular distance of $N/2 - 1 = 3$.

For really "bad" cycle graphs, even an automorphism that places neighbors at circular distance $\lfloor N/2 \rfloor - 1$ may not exist. Then, in a similar manner, we can search for an automorphism that places the neighbors of each node at a circular distance that is as close to $\lfloor N/2 \rfloor$ as possible.

The application of the automorphic method to cycle graphs shows the technique's versatility in terms of the usage different automorphisms to construct k-FT supergraphs with reduced complexities. For many cycle graphs it can construct optimal k-FT supergraphs.

6. SWITCH IMPLEMENTATION

We may be able to reduce the number of spare I/O ports and links of a fault-tolerant supergraph $G'[k, G]$ by judicious use of switches to switch a processor from one set of shared links to another when reconfiguration takes place. By a *switch* here we mean a hardware component that has m input and l output terminals and can be programmed to connect some subset of its inputs to some subset of its outputs. Recall that we assume that switch failure probabilities are negligible; hence a switch implementation of $G'[k, G]$ is as reliable as $G'[k, G]$. Typical examples of switches are multiplexers, demultiplexers, crossbar switches, and the routing switches introduced in [8]. There we discussed a switch implementation of k-FT trees using 2×2 routing switches. The fault-tolerant automorphic designs for circulant graphs proposed in this paper can also be efficiently implemented using shared links and demultiplexer switches. This switch implementation takes advantage of the circulant property of the k-FT supergraphs and the automorphic nature of the reconfiguration process.

In the following discussion, we assume that the nodes of G are $0, 1, \ldots, N-2, N-1$; that they are arranged on circ(G) in that order, i.e., according to the automorphism $(0, 1, \ldots, N-1)$ of G; and that the spare nodes of G_k are $s_0, s_1, \ldots, s_{k-1}$. Unless otherwise stated, all addition is modulo N. We have seen that for every edge $(u, u + d_i)$ with circular distance d_i in G, we effectively introduce spare edges incident on u with circular distances $d_i + 1, d_i + 2, \ldots, d_i + k$ in $G'_{\text{auto}}[k, G]$. When edge $(u, u + d_i)$ crosses $h \leq k$ faulty nodes, and u is nonfaulty, then the spare edge $(u, u + d_i + h)$ is used to replace $(u, u + d_i)$, since the former now realizes an edge with circular distance d_i incident on u (the h faulty nodes that it crosses are no longer part of the reconfigured system). Hence, none of the k spare edges corresponding to edge $(u, u + d_i)$ are ever used simultaneously in any k-fault set. We take advantage of this fact in our switch implementation of $G'_{\text{auto}}[k, G]$, which we denote by $\text{SW}_{\text{auto}}(k, G)$.

Figure 9 illustrates the basic idea of our switch implementation. The $k + 1$ edges in $G'_{\text{auto}}[k, G]$ that provide the necessary redundancy for edge $(u, u + d_i)$ with circular distance d_is in G are shown in Fig. 9a. In Fig. 9b, these edges are replaced by a single edge and a 1-to-$(k + 1)$ demultiplexer that can switch its single input to one of its $k + 1$ outputs to realize any of the original $k + 1$ edges.

Under fault-free conditions, the demultiplexer corresponding to each node u of G and circular distance $d_i \in \text{ds}(G)$ is programmed to connect u to the node $u + d_i$ that is at clockwise circular distance d_i from it in circ(G). The demultiplexers corresponding to spare nodes in $G'_{\text{auto}}[k, G]$ are disabled, so that all their output terminals are effectively disconnected. Thus, when there are no faults, the connections among the nodes of G in $\text{SW}_{\text{auto}}(k, G)$ are those of G, while the spare nodes of $G'_{\text{auto}}[k, G]$ are not connected to any other nodes.

A static rule for selecting f out of k spare nodes to configure into the system when $f \leq k$ nodes fail can be established. For example, here we assume that the subset of spare nodes $S(f) = \{s_{k-f}, \ldots, s_{k-1}\}$ will be configured into the system whenever f nodes fail. Suppose now that $f \leq k$ faults occur in G_k, that the edge from $(u, u + d_i)$ crosses f_1 faulty nodes, and that t_1 of the spare nodes crossed by $(u, u + d_i)$ are in $S(f)$. Then the edge $(u, u + d_i)$ has a circular distance $d_i + t_1 - f_1$ in the newly configured system, since it crosses exactly that many good nodes of this system. Assume that node u is not faulty. The demultiplexer switch corresponding to edge $(u, u + d_i)$ is reprogrammed so that u is connected to the node v making d_i the circular distance of edge (u, v) in the newly reconfigured system. Moreover, every spare node s_i that is configured into the new system is also connected to nodes that are at clockwise circular distances d_i from it, for every $d_i \in \text{ds}(G)$. The next theorem follows from the preceding discussion.

THEOREM 7. *The switch implementation* $\text{SW}_{\text{auto}}(k, G)$ *realizes* $G'_{\text{auto}}[k, G]$ *and has an incremental reconfiguration scheme that can reconfigure around any* $f \leq k$ *faulty nodes.*

To illustrate the feasibility and cost of the proposed

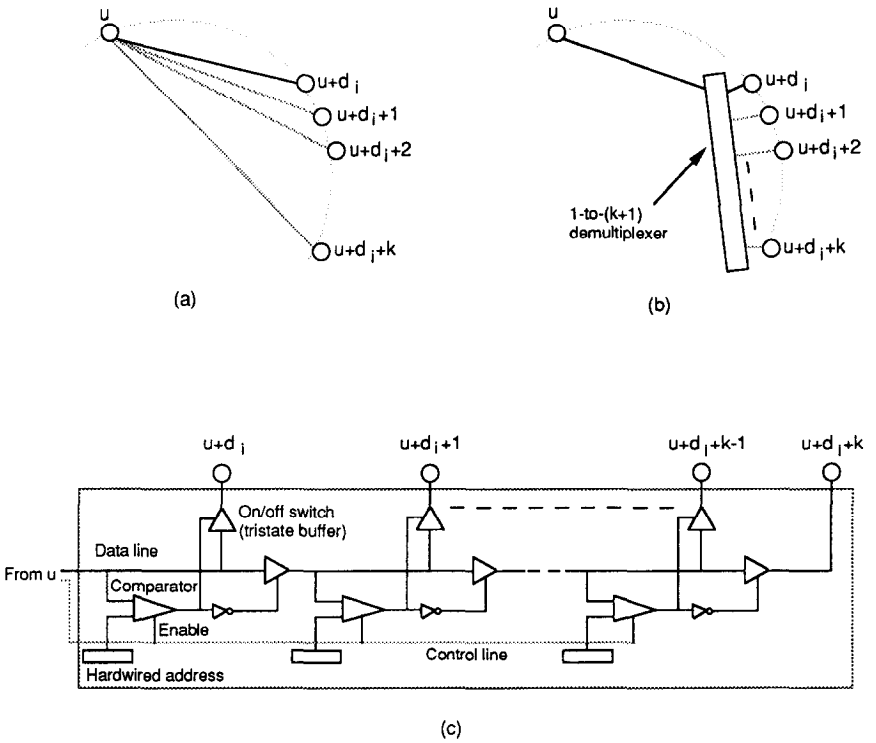

FIG. 9. (a) Spare edges in $G'_{\text{auto}}[k, G]$ corresponding to the edge $(u, u + d_i)$ in G. (b) Switch replacement of the spare edges employing a 1-to-$(k + 1)$ demultiplexer switch. (c) A possible logic design of the demultiplexer switch.

switching approach, a specific implementation at the logic level is outlined in Fig. 9c. A 1-to-$(k + 1)$ demultiplexer can be constructed from $k + 1$ comparators connected in a daisy-chain fashion, as shown. Each comparator is capable of activating exactly one output terminal and has a hardwired address corresponding to the node of G' connected to the output terminal it controls. When the node u controlling a demultiplexer sends it the address of the node to which the demultiplexer should make a connection, the address passes through the comparators until it reaches the one which registers a match with its hard-wired address. This comparator captures the address, activates its output terminal, and prevents the address from propagating further. The address sent to the demultiplexer needs only to be of length $\lceil \log(k + 1) \rceil$, since address resolution is among $k + 1$ nodes. Address transmission can be over the data lines as shown in Fig. 9c and requires time $\Theta(\log k)$.

Figure 10 shows a representative part of the proposed switch implementation of the 3-FT graph H_3 of Fig. 6. The edges in Fig. 10 are labeled by their circular distances. Corresponding to each edge of circular distance 5 are four links entering a node v from the output terminals of four demultiplexers. Since exactly one of these links is used to realize an edge between v and another node u at counterclockwise circular distance of 5 from v, all links can share a common bus connected to a single I/O port of v, as shown in the figure. Only one of these links is active at any time.

Figure 11a shows the symmetry and regularity of the overall switch implementation for a 2-FT supergraph of Q_4^2, which is a circulant edge supergraph (to be specified later) of a four-dimensional hypercube Q_4. Figure 11b gives a more

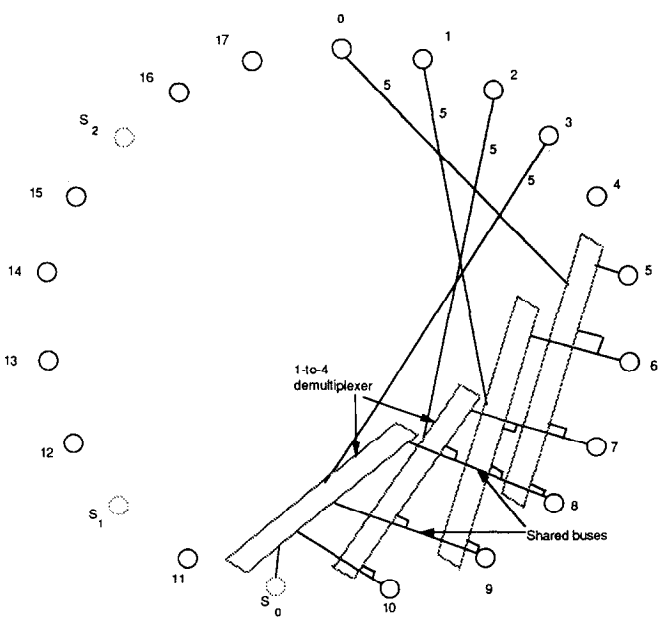

FIG. 10. Part of a switch implementation of the 3-FT graph H_3 of Fig. 6.

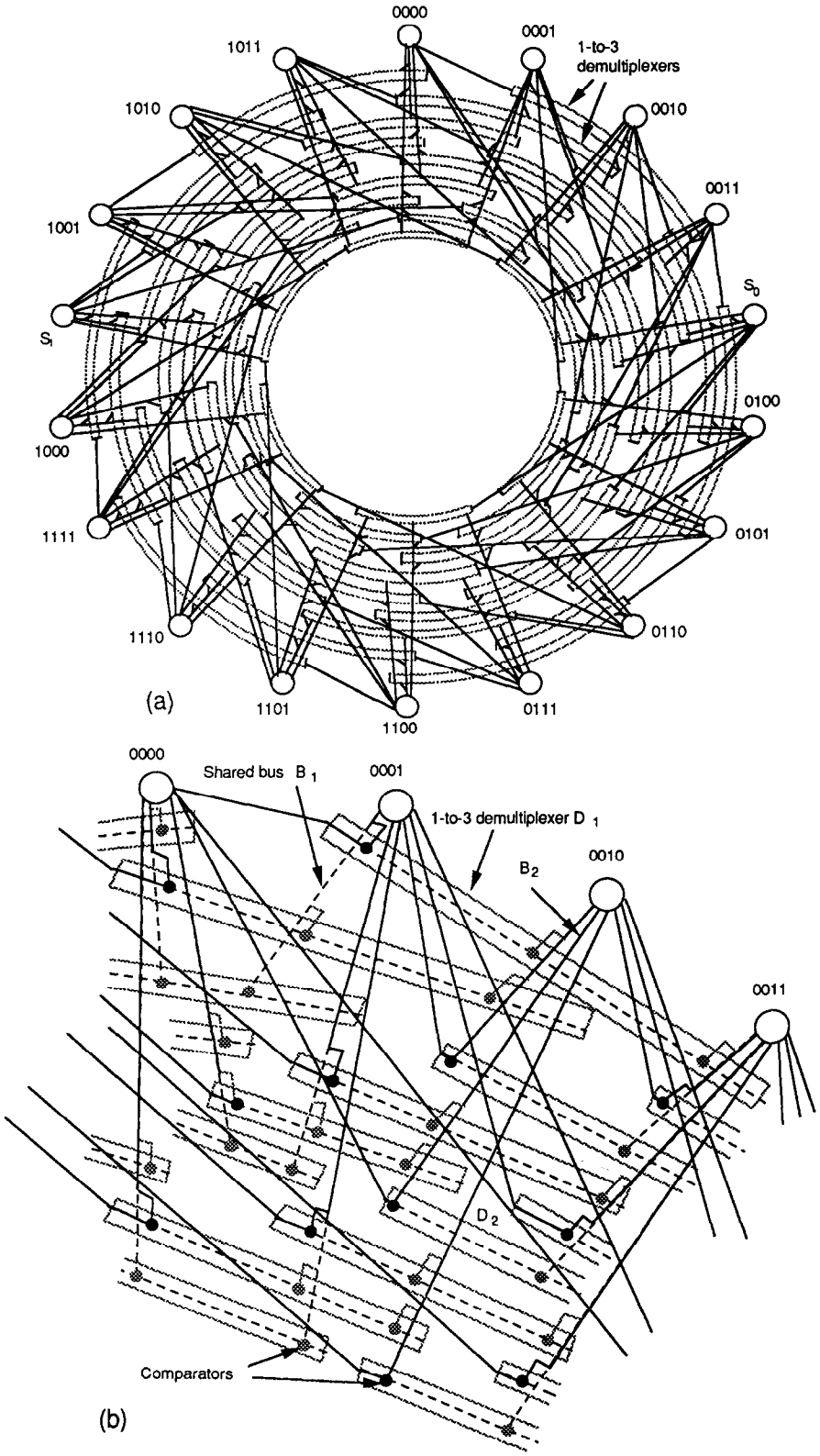

FIG. 11. (a) Switch implementation of the 2-FT supergraph $G'_{\text{auto}}[2, Q_4^*]$ of a four-dimensional hypercube. (b) An enlarged view of part of the switch mplementation.

detailed view of part of this switch implementation; the active comparators in each demultiplexer and the active links realizing the connections betwen nodes corresponding to the fault-free case are darkened; the inactive comparators are shown lightened, while the inactive links are either shown dashed or lightened. For instance, this darkening shows that the connection between nodes 0000 and 0001 is realized by demultiplexer D_1 activating its output link to bus B_1, while the connection between 0000 and 0010 is realized by demultiplexer D_2 activating its output link to bus B_2.

Though the switch implementation substantially reduces the wiring complexity of the fault-tolerant realization, one might argue that it reduces the overall reliability of the system. This is because switches, unlike the links they replace, are active components and thus have a higher probability of failure than links. However, the switches that we use are simple ones; each 1-to-$(k+1)$ demultiplexer of the type shown in Fig. 9c can be realized by using only $(k+1)(2 \times \log(k+1)+2)$ basic gates like OR and XOR. Thus the total gate complexity of all the switches in a k-FT implementation is $e(k+1)(2\log(k+1)+2)$. To use some actual numbers, consider the 2-FT four-dimensional hypercube of Fig. 11a. The total gate complexity of all the 48 switches used in this 2-FT realization is $48 \times 18 = 864$, which is a very small number compared to the hundreds of thousands of gates used in a single processor chip. Thus we can ignore the possibility of a fault occurring in the switches due to internal factors like fabrication defects and operational stuck-at-type faults. However, external factors like radiation and power line surges can still cause the switches to become faulty. It is generally feasible to model a switch failure (as well as a link failure) as the failure of the processor controlling it [9]. We can thus deal with switch and link failures at the system level and reconfigure around them in the rare cases that they do become faulty.

We now discuss how reconfiguration around processor faults can be achieved in our fault-tolerant switch implementation. We assume that the faulty processors have been detected using system-level fault-diagnosis techniques [21, 10, 16, 26] in either a centralized or distributed manner, and that the fault information has been disseminated among all nonfaulty processors. Hence, it is straightforward to keep track of the number of faults that have occurred up to the current time. Once this number exceeds k, the degree of fault tolerance, it will not be possible for the system to reconfigure structurally, and some other actions, which are outside the scope of this paper, will have to be initiated. Our concern here is with the mechanisms for structural reconfiguration, once faults have been detected. In the following, we refer only to those edges that emanate in the clockwise direction from each node—edges in the counterclockwise direction from a node are in the clockwise direction from their second end nodes and are thus accounted for. For distributed reconfiguration, each node has to know which faulty nodes are crossed by each of its edges in the clockwise direction, which is determined during the diagnosis phase. We also assume that each node knows which spares are crossed by these edges. In any fault situation with k or fewer faults, a nonfaulty node can determine in time $\Theta(k)$ the nodes to which each of its edges should connect. The complexity of $\Theta(k)$ arises from the time required to compute the number of nodes in $S(f)$, the set of spares to be configured into the system, which are crossed by an incident edge. Since every node has $\lceil d/2 \rceil$ incident edges in the clockwise direction, the total computation time required by each node is $\Theta(kd/2)$. As noted earlier, it takes $\Theta(\log k)$ time to transmit the address of the node to be connected to each demultiplexer. Since the address has to pass through $\Theta(k)$ comparators, the process of connecting u to the appropriate node takes time $\Theta(k)$, for each edge from u. Thus the total reconfiguration time is $\Theta(kd/2)$.

To recover completely from the effects of faults, the parallel processes that were executing on different processors will have to roll back to a previous checkpoint in order to reach a consistent state. Subsequently, these processes will have to be reassigned to new processors. For example, the process running previously on processor 2 will have to be transferred, along with its state information, to the processor relabeled as 2 after structural reconfiguration. This involves N one-to-one communications from every processor x to processor u, where x is the processor previously labeled as u, over the interconnection network. The time taken to accomplish the transfer of processes is a function of the amount of information to be transferred between each processor pair, and the interconnection network. For example, in a k-FT n-dimensional hypercube $\mathrm{SW}_{\mathrm{auto}}(k, Q_n)$, it is possible to accomplish the N transfers in $\Theta(\log N)$ hops using a pipelined store-and-forward scheme. Furthermore, if the interconnection network can be circuit switched, i.e., it is possible to create a single electrical connection between two nonadjacent nodes through the routing switches of intermediate nodes, then the N transfers can each be realized in a single hop using disjoint paths. In this case, however, the computation time required to set up these circuit-switched paths is $\Theta(\log N)$.

For every edge in G, the switch implementation requires one 1-to-$(k+1)$ demultiplexer. As described earlier, the $(k+1)$ switches in the demultiplexer of Fig. 9c can be $\Theta(\log k)$-bit comparators, whose hardware complexity is $\Theta(\log k)$. Hence the area/hardware complexity of each demultiplexer is $\Theta(k \log k)$. In this analysis, we assume that the nodes of G_k are arranged in some regular fashion in a two-dimensional region so that nodes that are close to one another on $\mathrm{circ}(G_k)$ are also physically near; it is not necessary for them to be on the perimeter of a circle. For example, if the nodes are arranged in rows in a rectangular area A, then they can be labeled in a snake-like manner along the rows as $0, 1, \ldots, N-1, s_0, \ldots, s_{k-1}$, which

conforms to their order on circ(G_k). In such a situation, each demultiplexer corresponding to an edge from u can be laid out so that each of its $k + 1$ comparators is close to the node that it connects to, as suggested by Fig. 10. For each d_i in ds(G) there will then be $k + 1$ comparators, each belonging to a different demultiplexer, near node v. Since the output terminals from each of these demultiplexers share a common bus connected to an I/O port of v, the total wire area required is dominated by these buses and can be shown to be $\Theta(kd^2/4)$ for each node. The area required by the demultiplexers is $\Theta((Nkd/2)\log k)$; thus the total redundant area/hardware complexity is $\Theta(\max(Nkd^2/4, (Nkd/2)\log k))$. In addition to reasonable complexity, another attractive feature of this switch implementation is that the degree of each node remains constant at d.

7. CYCLE MERGING

A natural way to extend the methodology developed in the previous sections to noncirculant graphs is to construct edge supergraphs that are circulant. In this section, we discuss a simple and efficient strategy to construct circulant edge supergraphs of noncirculant graphs. Let G be a noncirculant graph, and let $\alpha = C_1C_2 \cdots C_t$ be an automorphism of G with t cycles, where C_i is the cycle $(x_{i,0}, x_{i,1}, \ldots, x_{i,l_i-1})$. Merge these t cycles of G to form a single cycle $\alpha_1 = (x_{1,0}, \ldots, x_{1,l_1-1}, x_{2,0}, \ldots, x_{2,l_2-1}, \ldots, x_{t,0}, \ldots, x_{t,l_t-1})$ of length N, and add edges to G to construct an edge supergraph G^e of G, so that α_1 is an automorphism of G^e. We term this process *cycle merging*. To construct G^e, place the nodes of G on a circle $\mathcal{C}$ in the order that they appear in α_1, and compute the set D of circular distances d_i on $\mathcal{C}$ of each edge of G. Add edges between every pair of nodes at circular distances d_i, for each d_i in D. The resulting edge supergraph is G^e, with α_1 as one of its automorphisms and D as its distance sequence. The trick, however, is to arrange the cycles of α so that $|D|$ is as small as possible. For example, if the cycle $(x_{2,0}, \ldots, x_{2,l_2-1})$ is rotated clockwise i times to yield the cycle $(x_{2,l_2-i}, x_{2,l_2-i+1}, \ldots, x_{2,l_2-1}, x_{2,0}, x_{2,1}, \ldots, x_{2,l_2-i-1})$, which is then merged with the other cycles of α, a different and possibly smaller D may be obtained. Besides rotation of each cycle of α, the ordering of the cycles for merging also affects the final outcome.

Consider the graph K with 10 nodes shown in Fig. 12a; $\beta = (1, 2, 3, 4, 5)(6, 7, 8, 9, 10)$ is an automorphism of K. To construct a circulant edge supergraph of K, we merge the two cycles $C_1 = (1, 2, 3, 4, 5)$ and $C_2 = (6, 7, 8, 9, 10)$ of β. Figure 12b shows the two cycles arranged on a circle to yield the ordering $(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$ of the nodes of K; each edge is labeled by its circular distance. The set D of edge circular distances for this ordering is $\{1, 4, 5\}$. Now if C_2 is rotated clockwise by one node, and then merged with C_1, we obtain a circular ordering $(1, 2, 3, 4, 5, 10, 6, 7, 8, 9)$ of the nodes as shown in Fig. 12c. In this case, the set D $= \{1, 4\}$ is smaller than that obtained for the arrangement of Fig. 12b.

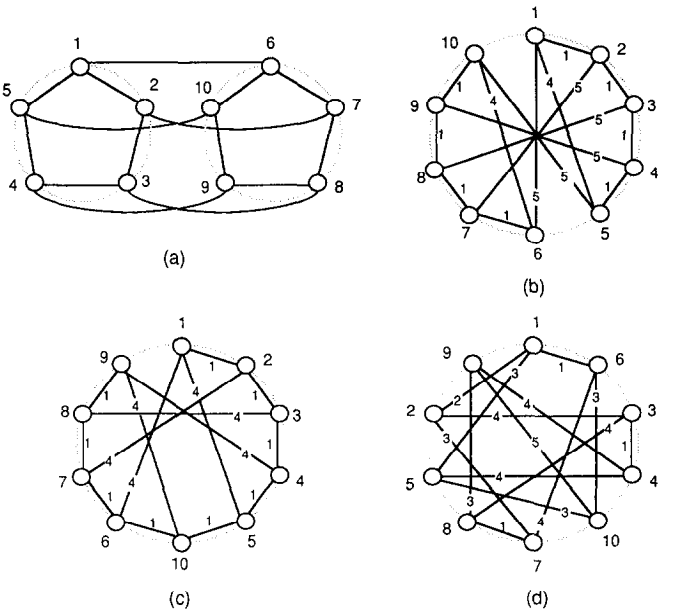

FIG. 12. (a) A graph K with an automorphism $\beta = (1, 2, 3, 4, 5)(6, 7, 8, 9, 10)$. (b) Merging of the cycles $(1, 2, 3, 4, 5)$ and $(6, 7, 8, 9, 10)$. (c) Merging of the cycle $(1, 2, 3, 4, 5)$ with the clockwise 1-rotation $(10, 6, 7, 8, 9)$ of cycle $(6, 7, 8, 9, 10)$. (d) An arbitrary circular ordering of the nodes of K.

Visualize the nodes of G laid out on $\mathcal{C}$, with the nodes in each C_i laid out in the order $x_{i,0}, \ldots, x_{i,l_i-1}$ and the cycles laid out in the order $C_1, \ldots, C_t$. Suppose, further, that there is an edge $(x_{i,0}, x_{j,p})$ in G between cycles C_i and C_j. Since α^s is an automorphism of G mapping $x_{i,0}$ to $x_{i,s}$, and $x_{j,p}$ to $x_{j,p+s}$, there are also edges $\{(x_{i,s}, x_{j,s+p})\}$ in G, where s is any integer, and the addition in the second subscript is mod l_i for the nodes in C_i and mod l_j for the nodes in C_j. Suppose, without loss of generality, that $l_i > l_j$. If $\text{cd}(x_{i,0}, x_{j,p}) = d_r$, then $\text{cd}(x_{i,s}, x_{j,s+p}) = d_r$, for $0 \leq s < l_j - p$, while $\text{cd}(x_{i,s}, x_{i,p+s}) = d_r - l_j$, for $l_j - p \leq s < 2l_j - p$, etc. Thus the set of edges $\{(x_{i,s}, x_{j,s+p})\}$ introduces more than one circular distance on $\mathcal{C}$. In general, it can be shown that the number of distinct circular distances corresponding to the edges between C_i and C_j is at most $d_{i,j} + d_{j,i}$, where $d_{h,l}$ is the number of edges in G between a node in C_h and the nodes in C_l. Note that $d_{h,l}$ is well defined, since the nodes in a cycle of an automorphism are similar and hence have the same number of edges to another cycle in that automorphism.

Suppose that instead of merging cycles to construct a circulant edge supergraph of G, some random merging scheme is used. This rearranges the nodes of G in an arbitrary cyclic order to yield a circulant G^e. Since there are a total of $d_{i,j}l_i$ edges between the nodes of C_i and C_j, potentially $d_{i,j}l_i$ distinct circular distances could be introduced in G^e by these edges. Refer to Fig. 12d, where the nodes of the graph K are arranged randomly on a circle in the order $(1, 6, 3, 4, 10, 7, 8, 5, 2,$

9); the corresponding set D of edge circular distances is $\{1, 2, 3, 4, 5\}$, and the resulting edge-circulant supergraph is complete. On the other hand, merging cycles introduces at most $d_{i,j} + d_{j,i}$ circular distances corresponding to the edges between C_i and C_j. For example, for the cycle-merging cases of Figs. 12b and 12c, only one circular distance is introduced by the intercycle edges. Hence, cycle merging is a good heuristic for constructing a circulant edge supergraph of G. The following result establishes a loose upper bound on the node degree of a circulant edge supergraph of G.

THEOREM 8. *Let G be a noncirculant graph and α an automorphism of G with t cycles $C_1, \ldots, C_t$. If the degree of each node in C_i is d_i, then the circulant edge supergraph G^e resulting from merging the cycles of α has node degree at most $2\sum_{i=1}^{t} d_i$. Thus if G is a regular graph with node degree d, then G^e has node degree at most $2dt$.*

Proof. Let the cycles of α be merged in any order. By our earlier discussion, we know that the number of distinct circular distances introduced by the edges between cycles C_i and C_j is at most $d_{i,j} + d_{j,i}$, when $i \neq j$, and exactly $d_{i,i}$, when $i = j$. In the worst case, the sets of circular distances $D_{i,j}$ introduced by the edges between each pair of cycles (C_i, C_j) are disjoint. Thus we can have a maximum of $\sum_{i=0}^{t-2} \sum_{j=i+1}^{t-1} (d_{i,j} + d_{j,i}) + \sum_{i=0}^{t-1} d_{i,i} = \sum_{i=0}^{t-1} d_i$ distinct circular distances in $\mathrm{ds}(G^e)$. This means that the maximum node degree in G^e is $2\sum_{i=0}^{t} d_i$, which is the contention of the theorem. ■

Let C_i be a cycle whose nodes are ordered as $(x_{i,0}, x_{i,1}, \ldots, x_{i,l_i-1})$. The ordered set $(x_{i,l_i-h}, x_{i,l_i-h+1}, \ldots, x_{i,l_i-1}, x_{i,0}, \ldots, x_{i,l_i-h-1})$ obtained by rotating C_i clockwise by h nodes is called the *clockwise h-rotation of C_i* and is denoted by C_i^h. A heuristic procedure MERGE_CYCLES for cycle merging, which rotates cycles and orders them to minimize the node degree of G^e, is given in Fig. 13. It is easy to see that MERGE_CYCLES is a superset of the cycle-merging scheme assumed in Theorem 8. It will thus perform at least as well as the lower bound (with respect to performance) stated in the theorem. The time complexity of this algorithm is $O(N^2 d_{\max})$, where $d_{\max}$ is the maximum node degree of G.

8. APPLICATION TO HYPERCUBE COMPUTERS

We now describe an application of the foregoing theory to hypercube architectures. A *hypercube* Q_n of dimension n is a graph with 2^n nodes, each of which can be labeled by a distinct binary vector of length n, so that there is an edge between two nodes u and v if and only if the labels of u and v differ in exactly one bit position. An automorphism of Q_n can be generated by any pattern of complementation and permutation of the bits in the label of each node [25]. For example, consider the labeling of Q_3 shown in Fig. 14a. Let us employ the permutation (1)(2, 3) of the dimensions of Q_3 (the leftmost bit position in an n-bit boolean vector represents dimension 1 and the rightmost represents dimension n) and complement bit 3 of each label after employing the above permutation. Then, label 000 remains 000 on permuting its bits according to (1)(2, 3) and becomes 001 on complementing bit 3. Similarly, label 001 after permutation

```
Procedure MERGE_CYCLES(G); /* This is a heuristic for constructing
a low-cost circulant edge-supergraph of a graph G using cycle merging */
begin
    Select an automorphism α = C1 C2 ... Ct of G;
    Compute the set Di,j of circular distances introduced by placing
    Ci adjacent to Cj, for each pair (Ci, Cj) of distinct cycles;
    Place the pair (Ci, Cj) adjacent to one another to minimize |Di,j|;
    L0 := Cj; L-1 := Ci; D := Di,j ∪ Di,i ∪ Dj,j;
    Not_Inserted := {C1, ..., Ct} - {Ci, Cj};
    while (Not_inserted ≠ ∅) do
        for each Cr in Not_Inserted do
            Let the inserted cycles be in the order L-h L-h+1 ... L0 L1 ... Lt;
            if h > t then p := t else p := -h;
            for each clockwise rotation Cr^i of Cr do
                Compute the set Dr^i of additional circular distances
                introduced by placing Cr^i adjacent to Lp;
            endfor;
            Dr := min{|Dr^i| : 0 ≤ i ≤ lr - 1};
        endfor;
        Select Cs for placing adjacent to Lp to minimize |Ds|;
        Place that Cs^i adjacent to Lp so that Ds = Ds^i;
        D := D ∪ Ds;
        Not_Inserted := Not_Inserted - {Cs};
    endwhile;
    From each node of G place edges to nodes at circular distances di
    in the clockwise and counterclockwise directions, for each di in D;
end. /* MERGE_CYCLES */
```

FIG. 13. A heuristic procedure for constructing a circulant edge supergraph of a noncirculant graph G.

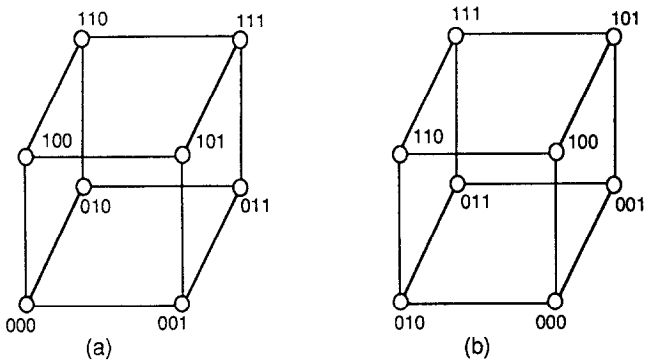

FIG. 14. (a) A labeling of Q_3. (b) A relabeling of Q_3 by the automorphism (000, 001, 011, 010)(100, 101, 111, 110).

according to (1)(2, 3) becomes 010, and on complementing, bit 3 becomes 011. If we continue this way, the permutation induced on the nodes of Q_3 in this manner is found to be (000, 001, 011, 010)(100, 101, 111, 110). This permutation preserves adjacency of the nodes of Q_3 and hence is an automorphism of Q_3. The new labeling of Q_3 according to this automorphism is shown in Fig. 14b.

The following theorem establishes that it is possible to construct a circulant edge supergraph of Q_n by cycle merging so that the node degree increases by exactly $n - 2$.

THEOREM 9. *There exists a circulant edge supergraph of Q_n in which each node has degree $2n - 2$.*

Proof Sketch. The proof, which is by construction, is as follows. Consider the automorphism $\alpha = C_1, \ldots, C_{2^{n-1}}$, where C_i is the cycle $(a_n a_{n-1} \cdots a_2 0, a_n a_{n-1} \cdots a_2 1)$ containing two nodes, and $a_n a_{n-1} \cdots a_2 0$ is the binary representation of the integer $2(i-1)$. The cycles of α are arranged on a circle in the order $C_1, \ldots, C_{2^{n-2}}, C_{3.2^{n-3}+1}, \ldots, C_{2^{n-1}}, C_{2^{n-2}+1}, \ldots, C_{3.2^{n-3}}$. Such an arrangement of the nodes of Q_4 is shown in Fig. 15a. It can be seen that the set D of circular distances in this case is $\{1, 2, 4\}$, and thus a circulant edge supergraph Q_4^e can be constructed with $\mathrm{ds}(Q_4^e) = D$, such that each node has degree 6; see Fig. 15b. For the general case Q_n, the set D of circular distances for the above-mentioned arrangement of the cycles of α is $\{2^i : 0 \leq i \leq n-2\}$; hence a circulant supergraph with distance sequence D and node degree $2(n-1)$ can be constructed. ■

The switch implementation of a 2-FT Q_4^e $G'_{\text{auto}}[2, Q_4^e]$ generated by SUPER_CIRCULANT is shown in Fig. 11. The distance sequence of Q_4^e is (1, 2, 4); thus the distance sequence of $G'_{\text{auto}}[2, Q_4^e]$ is $((1, 2, 4) \hat{+} \{1\}) \hat{+} \{1\} = (1, 2, 3, 4, 5) \hat{+} \{1\} = (1, 2, 3, 4, 5, 6)$, and its node degree is 12. However, note that the node degree in the switch implementation of $G'_{\text{auto}}[2, Q_4^e]$ reduces to 6, the node degree of Q_4^e. In general, assuming $k = 2^i$, where $i \leq n-3$, the node degree of $G'_{\text{auto}}[k, Q_n^e]$ constructed by SUPER_CIRCULANT is $2(n(k+1) - (k+1)\log k - 3)$. However, the switch implementation $\mathrm{SW}_{\text{auto}}[k, Q_n^e]$ reduces the node degree to $2n - 2$, while the redundant area/hardware required for the switch implementation of $G'_{\text{auto}}[k, Q_n^e]$ is $\Theta(Nk \log^2 N)$. Indeed, the node degree can be further reduced to n by using a two-to-one multiplexer to switch the two edges with circular distance 2^i, for $0 \leq i \leq n-3$, incident on node u, to one I/O port of u. This is possible, since exactly one edge with circular distance 2^i, either in the clockwise or the counterclockwise direction, is required at each node to realize Q_n, where $0 \leq i \leq n-3$; see Fig. 15a.

For the purpose of comparison, we now briefly outline a previous design for 1-FT hypercubes due to Rennels [23] that uses crossbar switches. In this approach, an n-dimensional hypercube Q_n is partitioned into 2^{n-m} disjoint Q_m's, where m is a constant between 1 and n. Two crossbar switches, a relay crossbar (RCB) and a connection crossbar (CCB), and a spare node are associated with each of these Q_m's. An RCB is a 2^m-input $(n-m)$-output crossbar switch, while a CCB is a $(2^m + n - m)$-input n-output crossbar switch. The 2^m inputs of the RCB corresponding to a Q_m A are connected to an I/O port of each node in A, while its $n - m$ outputs go to one input in each CCB of the Q_m's adjacent

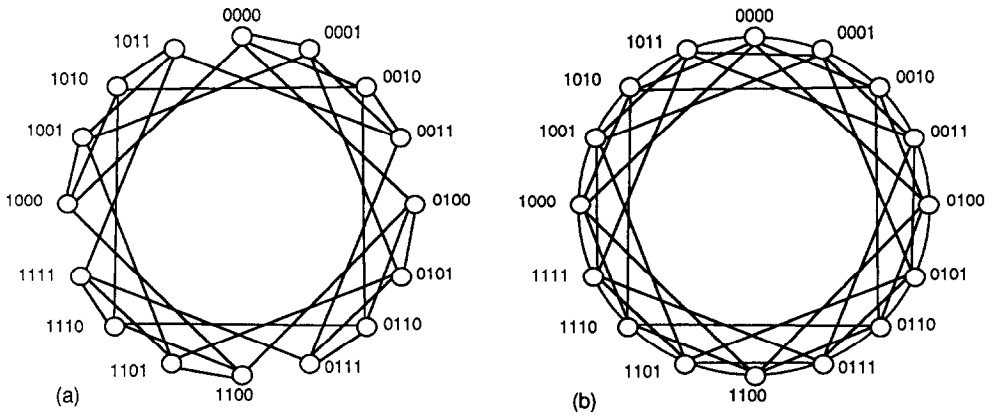

FIG. 15. (a) An arrangement of the nodes of Q_4 on a circle. (b) A circulant edge supergraph Q_4^e of Q_4.

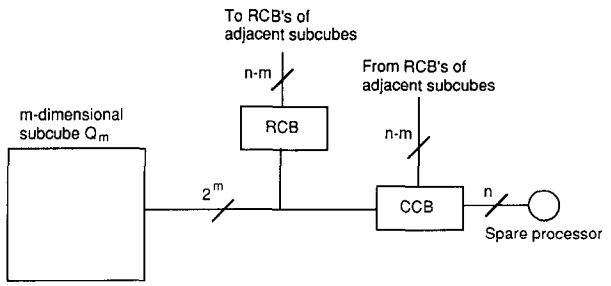

FIG. 16. Connections for an m-dimensional subcube of Q_n to realize a 1-FT Q_n due to Rennels [23].

to A. 2^m of the inputs to the CCB corresponding to A are also connected to the nodes of A, while the other $n - m$ inputs come from one output of the RCBs of the Q_m's adjacent to A. The n outputs of this CCB are connected to the spare node associated with A. These connections are shown for one of the Q_m's in Fig. 16. When a node u in Q_m A becomes faulty, A's CCB is programmed to connect the m neighbors of u in A to the spare node s_A associated with A. Each Q_m B adjacent to A contains one neighbor of u, which is connected to an output of B's RCB that is connected to an input of A's CCB. The CCB of A is further programmed to connect its $n - m$ inputs coming from the RCBs of the $n - m$ adjacent Q_m's to s_A. In this manner, all the neighbors of u in Q_n are connected to s_A to complete the reconfiguration process.

The area/hardware complexity of the crossbars is $\Theta(n^2 2^n) = \Theta(N \log^2 N)$, which is the total redundant complexity of our switch implementation for $k = 1$. It is difficult to analyze the redundant wire area required in this design, though it is likely to be more than $\Theta(N \log^2 N)$. Note that while we use only one spare node in our 1-FT design of Q_n, the 1-FT design in [23] uses 2^{n-m} spare nodes. Moreover, each node in our design has n I/O ports irrespective of k, while each node in the 1-FT design of [23] has $n + 1$ I/O ports.

Table I compares the automorphic design of fault-tolerant hypercubes to the previous fault-tolerant hypercube designs of Rennels [23] and the Diogenes approach [7, 24], which was mentioned in Section 1. Note that the Rennels design is 1-FT and hence is comparable with other designs only when $k = 1$. In the expression for the maximum edge length of the automorphic k-FT hypercube design, $e_{\max}$ is the maximum edge length in the nonredundant layout of Q_n. For an area-optimal layout of Q_n, $e_{\max}$ is $\Theta(N)$ [3]. Table I shows that the fault-tolerant hypercubes constructed by the automorphic design method are significantly better in most characteristics than the other two methods. In all other respects, this method is at least as good as the other methods. We estimate that in the best case, an n-input m-output crossbar switch can be programmed in a distributed manner in time $\Theta(\min(n, m))$. Hence, the reconfiguration time for the Rennels design is $\Omega(\log N)$.

9. CONCLUSIONS

We have presented a new approach to fault tolerance that uses the automorphisms of graphs to construct their k-FT supergraphs. We demonstrated various ways to exploit automorphisms including a cost-effective iterative scheme. We developed this iterative method for a class of graphs termed circulant and presented fault-tolerant design (SUPER_CIRCULANT) and reconfiguration algorithm that applies to such graphs. This scheme is simple and systematic and, unlike most previously published methods, provides for any arbitrary degree k of fault tolerance, as well as for incremental design. We showed that the automorphic method can construct optimal k-FT supergraphs for many cycle graphs. The k-FT supergraphs constructed by SUPER_CIRCULANT are amenable to an efficient and low-cost switch implementation $SW_{auto}(k, G)$. Further, a dis-

TABLE I

Comparison of the Automorphic Method to Previous Designs of k-FT n-Dimensional Hypercubes

Characteristics	Rennels [23]	Diogenes [7]	Automorphic
Spare nodes for 1-FT case	2^{n-m}	1	1
Generalizable to k faults	No	Yes	Yes
Redundant switch area	$\Theta(N \log^2 N)$	$\Theta(N(N + k))$	$\Theta((N/2)k \log N)$
Redundant wire area	Unknown	$\Theta(N(N + k)\log N)$	$\Theta(N^2 + Nk \log^2 N)$
I/O ports per node	$n + 1$	n	n
Maximum edge length	Unknown	$\Theta((N + k)\log N)$	$\Theta(e_{\max} + k \log N)$
Incremental design	N/A	No	Yes
Local sparing	N/A	No	Yes
Reconfiguration time	$\Omega(\log N)$	$\Theta(N(N + k))$	$\Theta(k \log N)$
Incremental reconfiguration	N/A	Yes	Yes
Distributed reconfiguration	Yes	No	Yes

Note. $e_{\max}$, the maximum edge length in a nonredundant layout of Q_n; N/A, not applicable.

tributed and incremental reconfiguration scheme with small time complexity ($\Theta(kd)$) was also given for this switch implementation. To apply our fault-tolerant design scheme and switch implementation to noncirculant graphs, we presented a technique called cycle merging to construct efficient circulant edge supergraphs of such graphs. The design methodology is thus applicable to general graphs, as illustrated for the hypercube Q_n, which is noncirculant for $n \geq 3$. Application of the automorphic approach to hypercubes also resulted in a fault-tolerant design significantly better than those presented in previous work [7, 23].

SUPER_CIRCULANT and cycle merging assume some knowledge of the automorphism group aut(G) of the original graph G. As demonstrated for the hypercube case, it is possible to deduce the relevant properties of aut(G), if G is sufficiently regular and symmetric; this is the case with most of the multiprocessor architectures that have been proposed. For example, we have also been able to characterize aut(G) fully, when G is a two-dimensional toroidal mesh. However, it is still desirable to be able to automate the automorphism-generation process, as well as develop algorithms to search for automorphisms with certain desirable properties like a small number of blocks (for single-cycle automorphisms), or a small number of cycles (for multiple-cycle automorphisms). We are pursuing these topics using results from computational group theory [18].

The automorphic method can also be extended to implement local sparing. Suppose that each subset V_i of a partition $\{V_1, \ldots, V_t\}$ of $V(G)$ corresponds to a cycle C_i of an automorphism α of G. Since the subgraph induced by a cycle of α is circulant, we can apply SUPER_CIRCULANT to each $G(C_i)$ to construct its k_i-FT supergraph (recall that $G(S)$ denotes the subgraph induced in G by the subset of nodes S), for some k_i. Finally, for edges between any two cycles C_i and C_j of α, spare edges have to be added so that the resulting design can tolerate k_i faults in each C_i; two ways in which this can be accomplished are discussed in [9]. Thus, implementing local sparing obviates the need to always construct circulant edge supergraphs of noncirculant graphs in order to apply automorphic design. This is especially useful when G is not "circulant-like," and thus the number of extra edges required to obtain its circulant edge-supergraph is large.

REFERENCES

1. Babai, L. On the abstract group of automorphisms. *Combinatorics, Proc. 8th British Combinatoric Conference,* Swansea, 1981, pp. 1–40.
2. Banerjee, P., Kuo, S. Y., and Fuchs, W. K. Reconfigurable cube-connected cycles architecture. *Proc. Sixteenth Fault Tolerant Computing Symposium,* June 1986, pp. 286–291.
3. Bhatt, S. N., and Leighton, F. T. A framework for solving graph layout problems. *J. Comput. System Sci.* **28** (1984), 300–343.
4. Biggs, N. *Finite Groups of Automorphisms.* Cambridge Univ. Press, Cambridge, 1971.
5. Biggs, N. *Algebraic Methods in Graph Theory.* Cambridge Tracts in Mathematics, Cambridge Univ. Press, Cambridge, 1974.
6. Boesch, F. T., and Tindell, R. Circulants and their connectivities. *J. Graph Theory* **8** (1984), 487–499.
7. Chung, F. R. K., Leighton, F. T., and Rosenberg, A. L. Diogenes: A methodology for designing fault-tolerant VLSI processor arrays. *Proc. Thirteenth Fault Tolerant Computing Symposium,* June 1983, pp. 26–31.
8. Dutt, S., and Hayes, J. P. On designing and reconfiguring k-fault-tolerant tree architectures. *IEEE Trans. Comput.* **C-39** (Apr. 1990), 490–503.
9. Dutt, S. Designing and reconfiguring fault-tolerant multiprocessor systems. Ph.D. thesis, Department of Electrical Engineering and Computer Science, University of Michigan, Aug. 1990.
10. Hakimi, S. L., and Amin, A. L. Characterization of connection assignment of diagnosable systems. *IEEE Trans. Comput.* (Jan. 1974), 86–88.
11. Hassan, A. S. M., and Agarwal, V. K. A modular approach to fault-tolerant modular tree architectures. *Proc. Fifteenth Fault Tolerant Computing Symposium,* June 1985, pp. 344–349.
12. Hayes, J. P. A graph model for fault tolerant computing systems. *IEEE Trans. Comput.* **C-25** (Sept. 1976), 875–883.
13. Hayes, J. P., *et al.* A microprocessor-based hypercube supercomputer. *IEEE Micro.* **6** (Oct. 1986), 6–17.
14. Hillis, W. D. *The Connection Machine.* MIT Press, Cambridge, MA, 1985.
15. Hopkins, A. L., Jr., Smith, T. B., III, and Lala, J. H. FTMP—A highly reliable fault-tolerant multiprocessor for aircraft. *Proc. IEEE* **66** (Oct. 1978), 1221–1239.
16. Kuhl, J. G., and Reddy, S. M. Distributed fault tolerance for large multiprocessor systems. *Proc. Seventh Annual International Symposium on Computer Architecture,* May 1980, pp. 23–30.
17. Kwan, C. L., and Toida, S. An optimal 2-fault tolerant realization of symmetric hierarchical tree systems. *Networks* **12** (1982), 231–239.
18. Leon, J. Computing automorphism groups of combinatorial objects. In Atkinson, M. D. (Ed.). *Computational Group Theory.* Academic Press, New York, 1984, pp. 321–337.
19. Lombardi, F., Negrini, R., Sami, M. G., and Stefanelli, R. Reconfiguration of VLSI arrays: A covering approach. *Proc. Seventeenth Fault Tolerant Computing Symposium,* June 1987, pp. 251–256.
20. Lowrie, M. B., and Fuchs, W. K. Reconfigurable tree architectures using subtree oriented fault tolerance. *IEEE Trans. Comput.* **C-36** (Oct. 1987), 1172–1182.
21. Preparata, F. P., Metze, G., and Chien, R. T. On the connection assignment problem of diagnosable systems. *IEEE Trans. Electron. Comput.* **EC-16** (Dec. 1967), 848–854.
22. Raghavendra, C. S., Avizienis, A., and Ercegovac, M. D. Fault tolerance in binary tree architectures. *IEEE Trans. Comput.* **C-33** (June 1984), 568–572.
23. Rennels, D. A. On implementing fault-tolerance in binary hypercubes. *Proc. Sixteenth Fault Tolerant Computing Symposium,* June 1986, pp. 344–349.
24. Rosenberg, A. L. The Diogenes approach to testable fault-tolerant arrays of processors. *IEEE Trans. Comput.* **C-32** (Oct. 1983), 902–910.
25. Saad, Y., and Schultz, M. H. Topological properties of hypercubes. *IEEE Trans. Comput.* **C-37** (July 1988), 867–872.
26. Sullivan, G. F. A polynomial time algorithm for fault diagnosability. *Proc. Twenty-Fifth Annual Symposium on Foundations of Computer Science,* Oct. 1984, pp. 148–156.
27. Turner, J. Point-symmetric graphs with a prime number of points. *J. Combin. Theory* **3** (1967), 136–145.

28. Yanney, R. M., and Hayes, J. P. Distributed fault recovery in multiprocessor networks. *IEEE Trans. Comput.* **C-35** (Oct. 1986), 871–879.
29. Yap, H. P. *Some Topics in Graph Theory.* Cambridge Univ. Press, Cambridge, 1986, pp. 96–97.

SHANTANU DUTT received the B.E. degree in electronics and communication engineering from the M.S. University of Baroda, India, in 1983, the M.Tech. degree in computer engineering from the Indian Institute of Technology, Kharagpur, India, in 1985, and the Ph.D. degree in computer science and engineering from the University of Michigan, Ann Arbor, in 1990. He is currently an assistant professor at the Department of Electrical Engineering, University of Minnesota, Twin Cities. He was awarded a National Merit Scholarship by the Government of India, a University Fellowship by the M.S. University of Baroda, and a Rackham Predoctoral Fellowship by the University of Michigan. His current technical interests include the design of fault-tolerant multiprocessor systems, parallel and distributed computing, computer architecture, and distributed fault diagnosis. He is a member of the IEEE Computer Society and the ACM Special Interest Group on Computer Architecture.

JOHN P. HAYES received the B.E. degree from the National University of Ireland in 1965 and the M.S. and Ph.D. degrees from the University of Illinois in 1967 and 1970, respectively, all in electrical engineering. He has been a professor in the EECS Department at the University of Michigan since 1982. He teaches and conducts research in the areas of computer-aided design and testing, computer architecture, and fault-tolerant computing. Hayes has authored several books, including *Computer Architecture and Organization* (McGraw-Hill, 2nd ed. 1988), and over 100 technical papers. He is a fellow of IEEE and a member of ACM and Sigma Xi.

Received September 30, 1990; accepted February 19, 1991