Reliable Butterfly Distributed-Memory Multiprocessors

Nian-Feng Tzeng, *Senior Member, IEEE*

Abstract— **Since the butterfly network possesses various attractive topological properties and its constituent node has a fixed degree, independent of the system size, interconnecting processors in accordance with the butterfly topology to construct a distributed-memory multiprocessor is advantageous, especially for a large sized system. Every butterfly node in a multiprocessor so constructed is a processor, not simply a switch. In this paper, we examine a reliable butterfly-based multiprocessor that preserves its full rigid butterfly configuration even in the presence of faults. The proposed butterfly parallel system can tolerate any single and many multiple node/link failures, giving rise to significantly improved reliability. Reconfiguration in response to an operational fault in our design is easy and may be performed in a distributed manner. A system after reconfiguration is ensured to provide the same high performance. Reliability results show that our design compares favorably with an earlier design. An extension to this reliable design is also addressed.**

Index Terms—**Butterfly network systems, distributed-memory multiprocessors, fault-tolerance, reconfiguration procedures, reliability evaluation.**

I. Introduction

THE interconnection topology of a large distributed-memory multiprocessor is critically important to overall system performance. The butterfly network, like the binary cube, has many attractive features, including, among others, simple routing, low diameter, and good support for communication patterns generated by numerous algorithms. In addition, every node in a butterfly network requires a constant degree, independent of the system size. The property of a fixed degree is extremely advantageous, because it makes possible the use of a single type of building block for constructing an arbitrarily large distributed-memory parallel system based on the butterfly topology. The cost of systems so constructed would be lowered as the building block approach becomes increasingly popular.

An essential design consideration of a large network system is its reliability. When the system size grows, the probability of having all system components fault-free during a given operation period falls quickly and could soon reach an unacceptably low point. It is thus necessary to incorporate redundancy in the system design, especially for a large system, to ensure proper continuing operation even after some components fail, improving reliability. As soon as a failure arises and is detected, a fault-tolerant system reconfigures itself so as to isolate the failed components. If a system after reconfiguration fails to retain its rigid original structure, it may no longer support the service level the system is designed to aim at. This is often the case for a high-performance system, like the butterfly-based parallel system, because tasks to be executed on the system are partitioned in a way that best matches the node interconnection style of the target machine, and any change in its topology or size tends to impair execution efficiency considerably. Therefore, redundant nodes and links are needed in a system to guarantee its topology and size unchanged, even in the presence of operational faults, supporting the desired performance level.

While there are a few research reports on fault-tolerant butterfly networks used in the switching context [1]–[3] where such a network is for interconnecting nodes on its side(s) (and essentially belongs to the class of multistage interconnection networks [4]), insufficient attention has been given to the reliable design of butterfly parallel systems where each butterfly node is not simply a switch only; it includes computation and communication logics as well as storage. In the former situation, the fault-tolerant issue lies in assuring the existence of a path between every network input and output; whereas in the latter, it requires that all the butterfly nodes be connected in the rigid form. This paper focuses on the latter fault-tolerance situation.

We propose a butterfly-based parallel system with significantly enhanced reliability. The reconfiguration process in response to an arising failure in this design involves only several nodes, and can be carried out in a distributed manner. A reconfigured system is assured to deliver the same high performance since it maintains the full rigid butterfly topology.

This paper is organized as follows. Section II briefly reviews the butterfly architecture. The proposed reliable butterfly design is introduced in Section III, followed by a description of its reconfiguration procedure in Section IV. Section V evaluates the reliability of the proposed design. In Section VI, this design approach is extended to provide further reliability improvement.

II. Overview of the Butterfly Topology

A **b**utterfly-based **d**istributed-memory **m**ultiprocessor (called the BDM for short) of size $N = L \times (\log_2 L + 1)$ is arranged as L levels of $(\log_2 L + 1)$ stages, with nodes

Manuscript received March 11, 1992; revised April 16, 1993. This work was supported in part by the NSF under Grants MIP-8807761 and MIP-9201308 and by the State of Louisiana under Contract LEQSF(1992-94)-RD-A-32. A preliminary version of this work was presented at the 20th International Conference on Parallel Processing, August 1991.

The author is with the Center for Advanced Computer Studies, University of Southwestern Louisiana, Lafayette, LA 70504 USA; e-mail: tzeng@cacs.usl.edu.

IEEE Log Number 9401632.

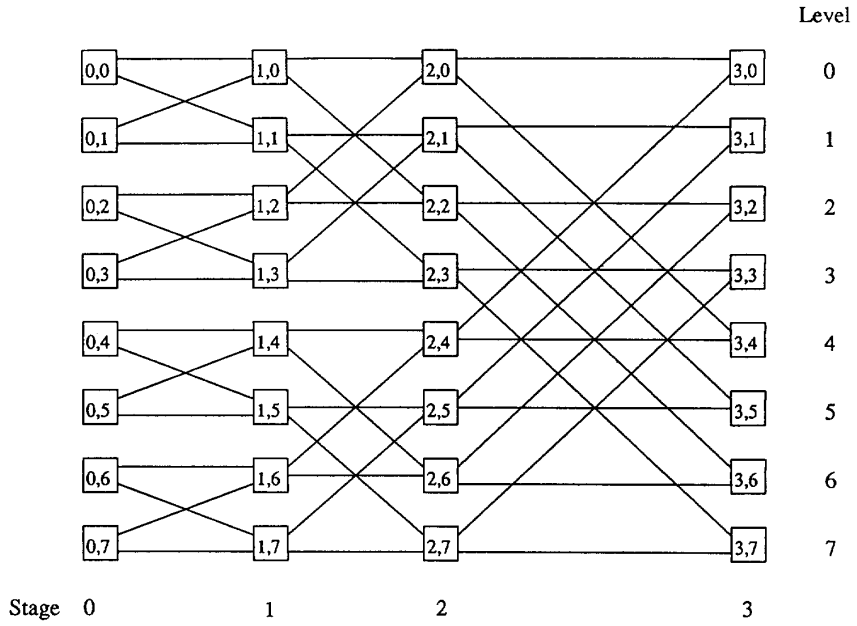

Fig. 1. A butterfly-based distributed-memory multiprocessor with 32 nodes.

in consecutive stages interconnected by the butterfly pattern. Figure 1 illustrates the BDM with 32 nodes (i.e., $L = 8$). Every link between two nodes can be used for bidirectional data transmission. Stages in the BDM are numbered in a sequence from 0 to $\log_2 L$, with 0 for the leftmost stage. Similarly, the nodes in every stage are labeled from 0 to $L-1$ (referred to as their *level* numbers), with 0 for the top node. A node at the l^{th} level of the g^{th} stage in the BDM can be uniquely denoted by (g, l).

Every node has degree 4, with two connected to nodes in the preceding stage and the other two connected to nodes in the next stage. A node in the two end stages utilizes its two links for connecting other nodes, and the remaining links may be used for I/O purposes. Let the links of a node used for connecting other nodes in the same level be called the *straight* (or S for short) links, and the rest be the *cross* (or C for short) links, then the BDM interconnection topology can be formally described below:

1) Nodes (g, l) and $(g+1, l)$ are connected through an S link, and
2) Nodes (g, l) and $(g+1, l+(-1)^{b_g} \times 2^g)$ are connected through a C link,

where $0 \le g < \log_2 L, 0 \le l < L$, and b_g is the bit with weight 2^g in the binary representation of l $(= b_{\log_2 L-1} \cdots b_1 b_0)$.

Note that a variant butterfly architecture considered earlier, called the reconfigurable chain-structured butterfly architecture (RECBAR) [7], has a wrap-around connection pattern present between the rightmost stage and the leftmost stage, and also every link is unidirectional (rather than bidirectional as in the BDM). A recent design called the Lambda network [5] allows for two-directional information flow over every link similar to the BDM, but it has a wrap-around connection pattern between the two end stages. A bidirectional network has lower diameter than its unidirectional counterpart. Without a wrap-around connection, the BDM can be laid out more efficiently since the wrap-around connection tends to occupy large silicon area.

III. Proposed Reliable BDM Design

A. The Design

A task to be executed efficiently on the BDM is often partitioned into subtasks which run concurrently at different nodes, with communication taking place in a way closely matching the interconnection pattern. A BDM with $L(= 2^n)$ levels has its every level connected to n other levels, each through a pair of C links of two successive nodes, as shown in Fig. 1. Level 1 in the figure, for example, is connected to level 0 through C links of nodes (0, 1) and (1, 1), to level 3 through C links of nodes (1, 1) and (2, 1), and to level 5 through C links of nodes (2, 1) and (3, 1). If (1, 1) becomes faulty, one direct connection between levels 1 and 0 is destroyed, as is one connection between levels 1 and 3. As a result, a task may no longer be executed efficiently on this machine after the fault arises due to the absence of node (1, 1) and the two direct connections to levels 0 and 3.

In order to ensure that the machine maintains its full strict structure despite the fault, we may 1) augment the connections between levels in a way that if a direct link exists between nodes (g_1, l_1) and (g_2, l_2), then an extra link is placed between nodes (g_1, l_1) and (g_2+1, l_2), and 2) add an extra node to the right end of every level, as depicted in Fig. 2, where the added node in level l is numbered (s, l). For example, a direct link is added between (0, 0) and (2, 1), since there is a direct connection existing between (0, 0) and (1, 1) in the original BDM given in Fig. 1. Similarly, an extra link is added between

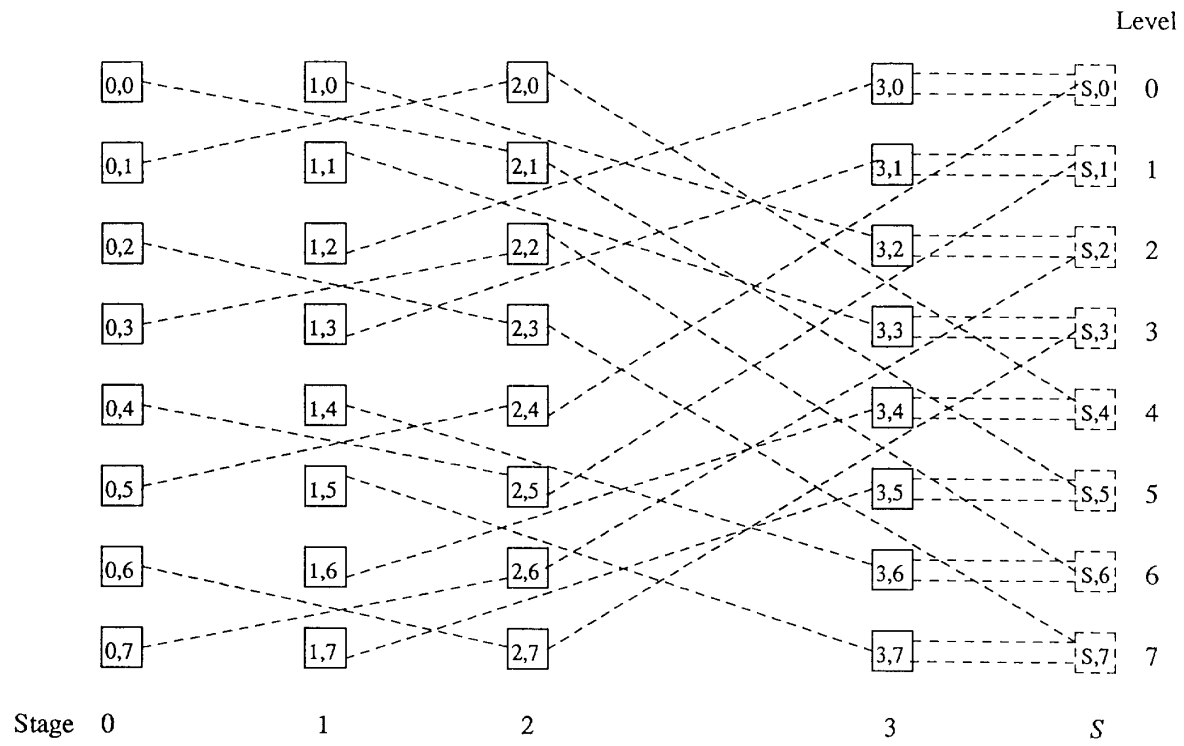

Fig. 2. The fault-tolerant BDM obtained from Fig. 1 by adding extra links and spare nodes. (Extra links and spares are denoted by dashed lines. Regular links are omitted for clarity.)

node (2, 4) and extra node (s, 0), as a cross connection exists between (2, 4) and (3, 0).

When node (1, 1) fails and is bypassed, a pair of direct connections still exists between levels 0 and 1, with one between (1, 0) and (0, 1) and the other between (0, 0) and (2, 1), effectively making node (2, 1) replace the role of the failed node (see Fig. 3(a) for the system after reconfiguration). With the extra links added, a pair of direct connections is also present between levels 1 and 3 after the said fault occurs, with one between (3, 1) and (1, 3) and the other between (2, 1) and (2, 3), as illustrated in Fig. 3(a). The link between (2, 1) and (2, 3) is realized by adding a control switch to the intersection of cross links between (1, 1) & (2, 3) and (2, 1) & (1, 3), and setting the switch to the "V" state (see Fig. 4(a)). Node (3, 1) therefore replaces the role of (2, 1) (which replaced the failed node) successfully. The spare node (s, 1) then replaces the role of (3, 1), and a pair of direct connections exists between levels 1 and 5.

The interconnection of each pair of cross links in the BDM is equipped with a control switch, which can be set at either the "V" state or the "X" state, as shown in Fig. 4(a). A pair of additional links added to the BDM is referred to as an *added cross connection pair,* and one control switch is placed at the intersection of each added cross connection pair, as depicted in Fig. 4(b). A BDM with added cross connection pairs and spare nodes is referred to as the FTBDM (which stands for the fault-tolerant BDM).

The control switch used is normally much simpler [8] than a node, and can be made more reliable as pointed out by earlier studies [9]. The use of control switches at regular cross connection pairs would somewhat reduce the reliability of individual cross connection pairs. However, if the switch is made very reliable (we can afford to do so, since the total number of such switches in a BDM with N nodes is less than $N/2$), relability degradation could be negligibly small. (A switch may be fabricated together with a node in the same module to get high reliability.) Furthermore, should such a switch or a cross connection fail, the failure can be treated as the node failure(s) (to be explained further in the next subsection) and is tolerable. This design thus results in enhanced reliability if a node is relatively more complex than a switch (a common case), as will be seen in Section V.

Four switches are added to each node so that a failed or unused node can be bypassed, as shown in Fig. 4(c). (Employing switches to bypass a failed/unused component can also be found in other designs, see, for example, [10], [11].) The spare node is connected to the external for I/O purposes, and two parallel links are present between a spare node and its immediate left neighbor (see Fig. 2) so that if the spare node is unused and bypassed (when no fault arises in that level), I/O data can directly reach the immediate left neighbor. A single failure in any one of the two links is tolerable by utilizing the spare node to replace the role of its (healthy) left neighboring node (which is bypassed), as shown in Fig. 3(b).

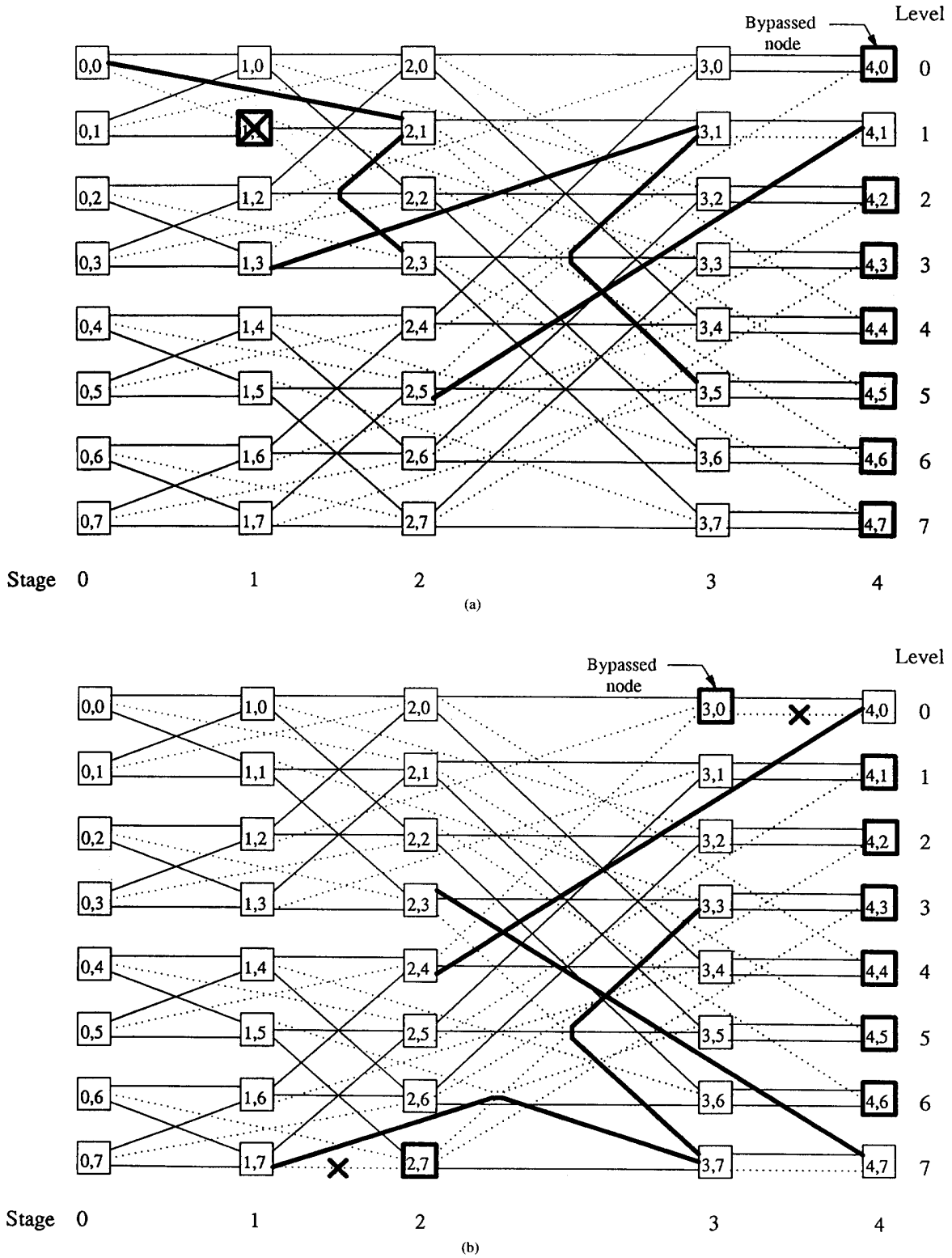

Fig. 3. A reconfigured system in the presence of (a) node (1, 1) failure, and (b) two link failures as marked. (Inactive links are denoted by dotted lines. Bold squares denote bypassed nodes.)

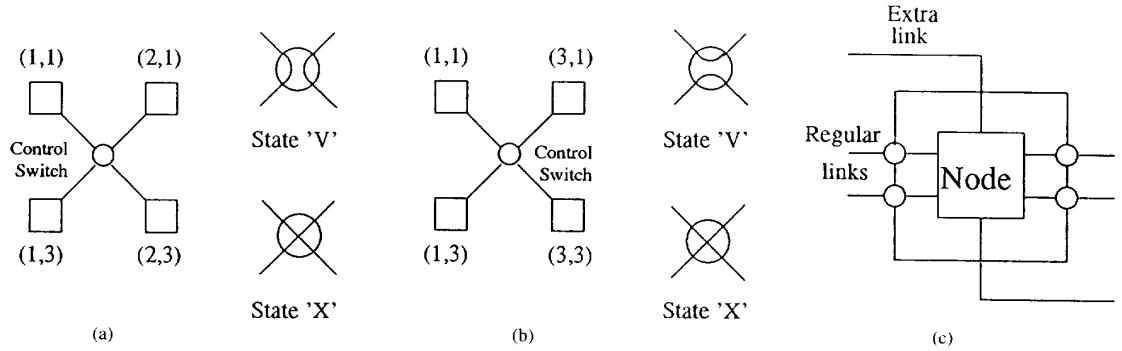

Fig. 4. (a) A pair of cross links with a control switch (which can be set to two possible states). (b) An added cross connection pair. (c) A node with four switches to allow "bypassing."

Formally, an FTBDM design with $L(=2^n)$ levels can be defined as a collection of $L \times (n+2)$ nodes, organized in $n+2$ stages (where the stage of spare nodes is numbered stage $n+1$). Every node now has two additional links for making extra connections (called E links for short), as depicted in Fig. 4(c). The interconnection style through C and S links is the same as that given in the previous section for a regular BDM, except that two parallel connections are added between (n,l) and $(n+1,l)$ through S links, for all $0 \le l < L$. The interconnection through E links is as follow:

Nodes (g,l) and $(g+2, l+(-1)^{b_g} \times 2^g)$ are connected through an E link, where $0 \le g < n, 0 \le l < L$, and b_g is the bit with weight 2^g in the binary representation of l. A complete FTBDM with 8 levels is illustrated in Fig. 2.

The proposed FTBDM has L spare nodes and $L \times (n+2)$ extra links (2 in the expression accounts for the two parallel connections present between a spare and its left neighbor), when compared with its regular counterpart. As a result, the node overhead ratio is $1/(n+1)$, and the link overhead ratio equals $(n+2)/2n$. This design requires more spare nodes than another known design, the RECBAR [7], where the number of spare nodes is n. However, our design is more reliable than the RECBAR, as will be illustrated in Section V. Furthermore, when a fault arises, reconfiguration in our design is simple and localized, with only a few nodes involved (as will be elaborated in the next section); whereas reconfiguration in the RECBAR needs the participation of all nodes.

B. Tolerance of Node and Link Failures

This design not only tolerates any single link or node failure, but also can withstand many multiple faults. The capability of tolerating node failures is addressed first, followed by a discussion of link fault tolerance. A node or link is considered faulty if it fails to provide correct operation. An FTBDM with $L\ (=2^n)$ levels is used for the following explanation.

Any single node failure is tolerable in the FTBDM. When a spare node fails, it is bypassed and no reconfiguration takes place since the spare in each level is bypassed initially. If a fault happens to an active node, the failed node is bypassed and its role is replaced by its physical successor (to its right in the same level), which in turn is replaced by its successor, etc., until the spare in the level is brought into the system. Each level can tolerate up to one node failure. After a node, say (g,l), failed and the system is reconfigured, a subsequent node failure may not be tolerable even when it is at a level other than l. In Fig. 3(a), for example, a subsequent failure at (0, 0) requires (1, 0) to substitute the role of the failed node during reconfiguration. In order for (1, 0) to substitute the role of (0, 0) appropriately, a direct connection between (1, 0) and (2, 1) is needed. However, such a connection cannot be established, and thereby the failure is intolerable. Similarly, a subsequent failure at (0, 3), (1, 3), or (2, 3) would make (3, 3) substitute its predecessor, (2, 3), implying that a direct connection should exist between (2, 1) and (3, 3). Again, such a connection is impossible to be realized, and the failure is intolerable. Likewise, the failure at (0, 5), (1, 5), (2, 5), or (3, 5) in Fig. 3(a) cannot be tolerated, either. As a result, after (1, 1) failed in level 1 of Fig. 2, eight $(=1+3+4)$ nodes are disallowed to fail subsequently in all other levels and are called *critical* nodes.

In general, the failure of (g,l) in the FTBDM introduces g critical nodes at level $l+\gamma_{g-1}, g+2$ critical nodes at level $l+\gamma_g, g+3$ critical nodes at level $l+\gamma_{g+1}$, etc., where $\gamma_t = (-1)^{b_t} \times 2^t$ with b_t being the bit with weight 2^t in the binary representation of l. Totally, there are $\delta = g+(g+2)+(g+3)+\cdots+n+(n+1)$ critical nodes created in all the levels other than level l. All these δ critical nodes plus all the $n+1$ active nodes in level l must be fault-free in order to guarantee the full rigid butterfly structure. Let Ψ be $\{x \mid x$ is a critical node or an active node in level $l\}$, then, a subsequent node failure is tolerable only if it is not in Ψ. After the second tolerable node failed, certain other nodes become critical, and those critical nodes can be identified in the same way. A new Ψ is determined in the presence of two node failures, and a third failure is tolerable, provided that it is not in the newly determined Ψ. Generally, any given multiple node failures decide a set Ψ, and a subsequent node failure is tolerable as long as it does not belong to the set Ψ. This criterion is useful for determining whether a given fault pattern is tolerable and will serve as the basis of our reliability analysis in Section V.

The FTBDM can tolerate any single link failure or cross connection pair failure. When a fault happens to a C link, it is treated as the failure of either one of the two nodes connected by the link, and thus causes reconfiguration in the same way as a node failure. If an S link fails, slightly different reconfiguration takes place. When the link between (1, 7) and (2, 7) in Fig. 3(b) becomes faulty, for example, node (2, 7), although healthy, is bypassed, and its role is replaced by (3, 7). The two cross connections between levels 7 and 5 exist between (1, 7) & (2, 5) and between (3, 7) & (1, 5) (through the bypassed node (2, 7)). The direct connection between two logically adjacent nodes in level 7, (1, 7) and (3, 7), is realized by the added cross connection pair between levels 7 and 5, when the associated control switch is set to the "V" state (see Fig. 4(b)). The spare node in level 7 is brought in to replace the role of node (3, 7), and a pair of cross connections between levels 7 and 3 are established between (3, 7) & (3, 3) and between (4, 7) & (2, 3), as illustrated in Fig. 3(b). This design assures its full strict topology after a link fails.

In general, a failed S link between stages g and $g+1$ in level l is replaced by an alternative connection created using the cross connection pair added between level l and level $l_1 = l+\gamma_g$, where γ_g is defined earlier. Then, node $(g+1, l)$ is bypassed, and its role is substituted by $(g+2, l)$, whose role in turn is replaced by $(g+3, l)$, etc., until the spare node in level l is brought in. It can be observed that after the PS link failed, all the nodes and links to the left of $(g+1, l_1)$ in level l_1, including $(g+1, l_1)$ itself, are disallowed to be faulty and become *critical*. This is because such a fault would require a direct connection established between $(g+2, l_1)$ and (g, l) so as to maintain the butterfly pattern, but that direct connection cannot be made (see Fig. 3(b) for the case of $g = 1, l = 7$ and $l_1 = 5$). Likewise, following the same argument as a node failure, we arrive at that there are $g+3$ critical nodes introduced in level $l+\gamma_{g+1}, g+4$ critical nodes in level $l+\gamma_{g+2}$, etc. It should be noted that all the links to the left of a critical node in those specific levels also become *critical* and cannot fail subsequently. Totally, $(g+2)+(g+3)+\cdots+n+(n+1)$ nodes in the FTBDM become critical as a result of the said S link failure. A *critical component* refers to a critical node or a critical link. The set Ψ due to the failed S link can be determined as $\{x \mid x$ is a critical component or an active component in level $l\}$. Along the same line as described above, we can decide whether any given fault pattern involving multiple node/link failures is tolerable using Ψ.

The failure of a cross connection pair, say between level l and level $l_2 = l+\gamma_g$, can be tolerated as follows: it is treated as the failures of both $(g+1, l)$ and $(g+1, l_2)$, and evokes reconfiguration accordingly. This is made possible because we simply substitute the failed pair using the added cross connection pair present between the two levels. The number of critical nodes (and links) created by a failed cross connection pair is twice the number due to the failure of $(g+1, l)$.

IV. Reconfiguration Procedure

At the beginning of operation when no fault exists, all the control switches on C links are set to the "X" state to implement the butterfly interconnection pattern, and all the E links are disabled. We assume that the PE is self-testable, capable of determining its own status. Every PE carries out self-test periodically, and a healthy PE is responsible for detecting possible faults at the links connected to it. After a fault arises and is detected by self-test, certain control switches on C links change their settings, and also some E links are activated with the control switches on them set appropriately. This is accomplished locally, with only several nodes involved.

To simplify subsequent explanation, the C link from (g, l) to $(g-1, l_1)$ is denoted as $C_{g-1}^{(g,l)}$, and the C link to $(g+1, l_2)$ is $C_{g+1}^{(g,l)}$. Similarly, the E links from (g, l) to $(g-2, l_3)$ and $(g+2, l_4)$ are referred to as $E_{g-2}^{(g,l)}$ and $E_{g+2}^{(g,l)}$, respectively. Consider an FTBDM with L $(= 2^n)$ levels. The procedure in response to a node failure is addressed first.

A. Reconfiguration for Node Failures

Assuming that a node fails at stage g in level l of the FTBDM. Reconfiguration due to this failure requires the participation of level l nodes which are at stages $k, g \le k \le n+1$. Specifically, the failed node is bypassed, and the nodes (k, l), for all $g+1 \le k \le n+1$, carry out the following steps in a distributed manner.

1) For $k \ge 2$:
   a) If node (k, l) has active $E_{k-2}^{(k,l)}$, reconfiguration fails; else the node activates $E_{k-2}^{(k,l)}$, with the switch on it set to the "X" state.
   b) If the node at the other end of this activated link, namely, node $(k-2, l+\gamma_{k-2})$, is faulty or bypassed, reconfiguration fails; else the node is informed of this change so that it would enable $E_k^{(k-2,l+\gamma_{k-2})}$ and disable $C_{k-1}^{(k-2,l+\gamma_{k-2})}$ (with the switch on it kept unchanged) for communicating with level l;
2) For $k \le n$:
   a) If the switch on $C_{k-1}^{(k,l)}$ of node (k, l) is at the "X" state, the switch is changed to the "V" state; else reconfiguration fails.
   b) If the node now connected directly to (k, l) is faulty, reconfiguration fails as well;
3) For $k < n$:
   Node (k, l) deactivates $C_{k+1}^{(k,l)}$ (with the switch on it kept unchanged).
4) One link between nodes (n, l) and $(n+1, l)$ is disabled.

When a fault occurs at stage 1 in level 1 of Fig. 2, for example, nodes (1, 1), (2, 1), (3, 1), and (4, 1) carry out this procedure, and the affected links upon reconfiguration are darkened in Fig. 3(a). After reconfiguration, every node in the intermediate stages has exactly four active links (despite being equipped with six links), and the full strict butterfly topology is preserved. In Step (1.a), reconfiguration fails when multiple failures arises in a single level. Consider Fig. 3(a), for example, a subsequent node failure in level 1, say the failure of (2, 1), would make reconfiguration unsuccessful because node (3, 1), during reconfiguration, finds its $E_1^{(3,1)}$ active. Step (1.b) or Step (2) fails when a subsequent failure happens to a

critical component. In Fig. 3(a), all nodes and S links which are in level 5 and which lie at or before stage 2 are critical components, as mentioned earlier. Any failure at these critical components, say the failure of (2, 5), causes (3, 5) to fail in reconfiguration following Step (2.a), since it finds that the switch on $C_2^{(3,5)}$ is not at the "X" state. The last two steps are for disabling unused links. For instance, (2, 1) deactivates $C_3^{(2,1)}$ in Fig. 3(a) because it no longer communicates with level 5.

It is clear from the above procedure that an arising fault at stage g causes no more than $3 \times (n + 1 - g)$ nodes to participate in the reconfiguration procedure. A failure at a later stage leads to fewer participators. When a failure is detected in the FTBDM, say at stage g in level l, a signal to initiate reconfiguration is sent to nodes (k, l), for all $g < k \leq n+1$, in the same level; every node after receiving such a signal starts reconfiguration individually, following the above four steps. This procedure is carried out in a distributed manner, and it provides an unsuccessful signal if a fault pattern is intolerable. Since every involved node performs the reconfiguration steps independently and simultaneously, it takes a constant time to complete overall reconfiguration.

Unlike our procedure, where no more than $3 \times (n + 1 - g)$ nodes are involved, the reconfiguration process of the RECBAR [7] requires all nodes, $(n + 1)(2^n + 1)$ in total, to participate. Once a fault is found in RECBAR, the faulty location has to be broadcast to all other nodes before reconfiguration starts. If broadcast is not supported, reconfiguration in RECBAR can be carried out only in a centralized fashion, which is undesirable for a large system.

B. Reconfiguration for Link Failures

The fault at a C link (or the switch on it) can be treated as a node failure, and evokes the same procedure as above. Reconfiguration in response to an S link failure is slightly different. Consider the failure of S link between (g, l) and $(g + 1, l)$. This fault results in 1) node (g, l) activating $E_{g+2}^{(g,l)}$ with the switch on it set to the "V" state, 2) node $(g + 1, l)$ connecting $C_g^{(g+1,l)}$ directly to its S link to the next stage (effectively bypassing this node), and 3) node $(g + 2, l)$ enabling $E_g^{(g+2,l)}$ and changing the switch on $C_{g+1}^{(g+2,l)}$ from the "X" state to the "V" state. Additionally, each level l node at stage $k, g + 2 < k \leq n + 1$, evokes the procedure given above for a node failure. The number of nodes involved in reconfiguration for the failed S link is no more than $3 \times (n-g)$, which is also stage-dependent. The affected links due to reconfiguration for link failures are darkened in Fig. 3(b).

Note that the fault at any link between a spare node and its left neighbor is treated as a fault at the left neighbor and is reconfigured accordingly, as shown in Fig. 3(b). In this case, three nodes participate in reconfiguration. In the presence of an intolerable fault pattern, reconfiguration fails when the procedure for the node failure is performed. It is clear that reconfiguration in response to a link failure again is carried out at every participating node independently, irrespective of which link fails.

V. Reliability Analysis

The FTBDM is considered failed when the number and the locations of the faults prevent it from reconfiguring back to its corresponding BDM. Many multiple link failures are tolerable, as was addressed earlier, and in practice a link tends to be far more reliable than a node, particularly for the link involving no control switch, such as the S link. For simplicity, our reliability analysis does not consider the failure of individual links, with the failure rate of a cross connection pair included in that of the nodes which are connected by the pair. On the other hand, the failures of added cross connection pairs are taken into account separately. We compare our analytic results with those of the BDM and the RECBAR, in which all links are considered totally fault-free.

The reliability for all nodes is assumed to be equal and exponentially distributed, with a constant failure rate λ (i.e., $R = e^{-\lambda t}$). Although this assumption is not necessarily accurate in all cases, it does provide an initial point for comparison and is a common assumption in reliability evaluation (see, for example, [7], [11], [12]). The reliability of a spare node is also assumed to be R. Further, an added cross connection pair plus its control switch is assumed to have reliability $R_{cs} = e^{-\lambda_{cs} t}$.

From the reconfiguration procedure given in the previous section, we observe that an arising node failure at level l in an FTBDM with $L = 2^n$ levels causes at most n added cross connections (plus their switches) to be activated during reconfiguration, and fewer such connections are activated when the fault arises at a later stage. In the presence of tolerable multiple faults, say f faults, no more than $f * n$ added cross connections are enabled and have to be healthy. Since the mean number of added cross connections activated upon reconfiguration in response to f faults is less than $f * n$, for all $f > 0$, the reliability of the FTBDM can be expressed as

$$\begin{aligned} R_{\mathrm{FTBDM}}(t) > {} & \beta_0 R^N + \beta_1 R^{N-1}(1-R) R_{cs}^n \\ & + \beta_2 R^{N-2}(1-R)^2 R_{cs}^{2*n} + \cdots \\ & + \beta_i R^{N-i}(1-R)^i R_{cs}^{i*n} + \cdots \\ & + \beta_N (1-R)^N R_{cs}^{N*n}, \end{aligned}$$

where $N = L \times (n+2)$ is the total number of nodes (including spares) in the system, and β_i is the number of ways in which i tolerable faults can occur in the system. β_i equals 0 for i exceeding the number of maximum tolerable faults, L (L total faults can be tolerated, if all of the faults are at stage n or $n + 1$). The ith term of the above summation reflects the situation that there are exactly i faults, which require less than $i * n$ healthy cross connections.

In order to evaluate FTBDM reliability, it is necessary to calculate the number of possible locations for the i^{th} tolerable faulty node to occur, given that the system has already reconfigured itself successfully in response to $i - 1$ faults, $i \geq 2$. This number depends on the positions of those existing $i - 1$ faults. The two initial β values are $\beta_0 = 1$ and $\beta_1 = N$ (as the first fault can be at any node locations), and a lower bound on the number of possible locations for the i^{th} fault is derived subsequently using the criterion for determining whether a given fault pattern is tolerable described in Section III.

Recall that, after a node failed at stage g in the FTBDM, there are $\delta = g + (g+2) + (g+3) + \cdots + n + (n+1)$ critical nodes introduced. The maximum value of δ is $\delta_{\max} = n(n+3)/2$ (obtained when g is equal to 0). A subsequent fault is tolerable, provided that it is not any one of the δ critical nodes, nor is in the same level that involves the failed node. As a result, the second fault can be at no fewer than $N - \Delta$ (where Δ is $\delta_{\max} + (n+1)$) possible locations for any given first failure, implying that $\beta_2 \geq N(N-\Delta)/2$. Any existing two tolerable faults introduce no more than 2δ critical nodes, and a subsequent fault can be at no less than $N - 2\delta - 2(n+1)$ locations (the last term accounts for locations of the two levels where the two faults reside). There are β_2 distinct fault patterns in the presence of two failures, and each pattern allows a third fault to be at no less than $N - 2\Delta$ locations, yielding $\beta_3 \geq N(N-\Delta)(N-2\Delta)/(2 \times 3)$.

Along the same line, it is easy to see that in general, for any given fault pattern with $i-1$ failures, the i^{th} tolerable fault can be at no fewer than $N - (i-1)\Delta$ possible locations, as $i-1$ failures introduce no more than $(i-1)\delta$ critical nodes. Since there are no less than $\beta_{i-1} = N(N-\Delta)(N-2\Delta)\cdots(N-(i-2)\Delta)/(i-1)!$ distinct fault patterns in the presence of $i-1$ failures, we have

$$\begin{aligned}\beta_i &\geq \beta_{i-1}(N-(i-1)\Delta)/i \\ &= N(N-\Delta)(N-2\Delta)\cdots(N-(i-1)\Delta)/i!,\end{aligned}$$

for all $i \geq 2$. Note that the expression on the right-hand side equals 0 if $N \leq (i-1)\Delta$. With β_i lower bounds available, we may calculate the FTBDM reliability lower bound using the above inequality for $R_{\text{FTBDM}}(t)$. Because the lower bound of system reliability gives a conservative probability that the system will still be operational at a specified point of time and the actual probability is at least that high, finding a lower bound is often sufficient and no further effort in obtaining the exact reliability expression is desired (particularly for the case where an exact expression is extremely difficult to get, like the FTBDM system). It is assured that the system will be operational over the time interval of interest with the likelihood no less than what is specified by the lower bound.

The reliability lower bounds on the FTBDM with $L = 2^4$ and 2^8 levels are depicted in Fig. 5, where the node failure rate λ is assumed to be 1.0 per unit time (e.g., 10^6 hours), and the failure rate of a cross connection plus its switch, λ_{cs}, is 0.1 per unit time. The exact reliabilities of the corresponding BDM and RECBAR [7] are also provided in this figure under the same node failure rate (i.e., 1.0 per unit time) but a zero link failure rate (i.e., totally fault-free links, an optimistic assumption). Notice that the assumption of fault-free links makes it simple to compare the RECBAR with the BDM (and also the FTBDM), since they have different types and amounts of links. The FTBDM has a significantly improved reliability over its BDM counterpart and is more reliable than the RECBAR. The reliability gap between the FTBDM and the BDM (or the RECBAR) increases as the system size grows, and in reality, the gap is expected to be larger than that shown in the figure.

A useful reliability improvement measure of a fault-tolerant design over its regular counterpart is the reliability improvement factor (RIF), defined as [6]

$$\text{RIF} = \frac{1 - R_{\text{reg}}}{1 - R_{\text{ft}}},$$

where R_{reg} and R_{ft} are the reliability of a regular design and the reliability of a fault-tolerant design, respectively. Since a fault-tolerant design needs more nodes due to spares, it appears meaningful and interesting to take into consideration the spare amount of a design in comparing different fault-tolerant designs. Let ϕ be the ratio of the number of nodes in a fault-tolerant design to the number of nodes in a regular design. We calculate the ratio of RIF to ϕ, called the *normalized RIF*. The fault-tolerant design with a larger normalized RIF is more effective and is desirable.

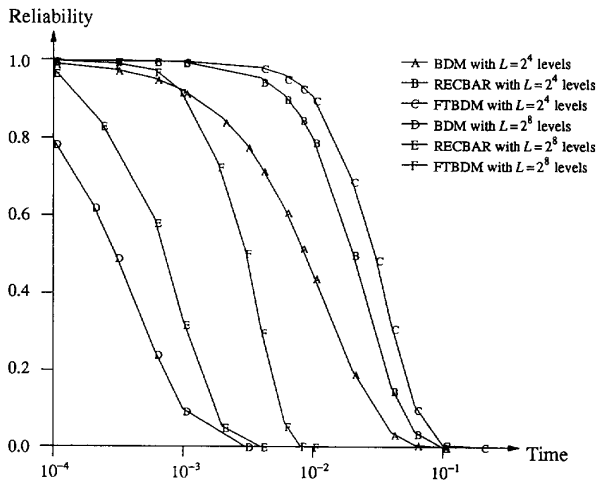

Fig. 5. Reliability comparison among BDM, RECBAR, and FTBDM under $\lambda = 1.0$ and $\lambda_{cs} = 0.1$ per unit time.

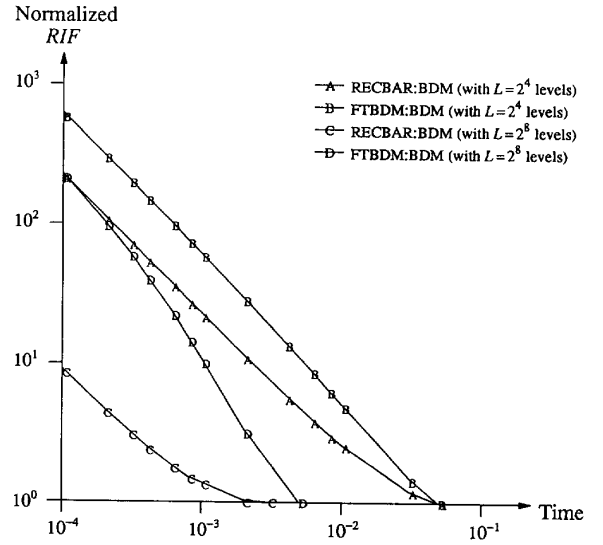

Fig. 6. Normalized reliability improvement factor (RIF) for RECBAR vs. BDM and for FTBDM vs. BDM under $\lambda = 1.0$ and $\lambda_{cs} = 0.1$ per unit time.

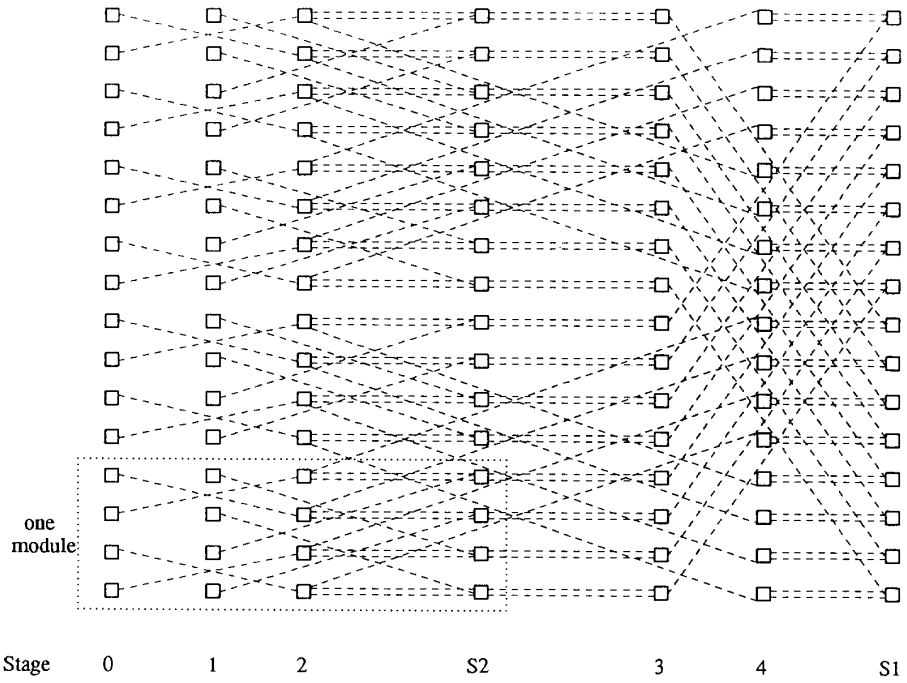

Fig. 7. A fault-tolerant BDM with two columns of spare nodes as indicated. (Extra links are denoted by dashed lines; regular links are omittedfor clarity.)

Figure 6 illustrates the normalized RIF's of the FTBDM and the BDM and of the RECBAR & the BDM for the same two system sizes as in Fig. 5. The FTBDM exhibits a much larger RIF than the RECBAR even when the incorporated spares are taken into account, and is therefore a more effective design. The normalized RIF for either fault-tolerant design tends to decrease as the mission time extends, because the reliability then lowers, leaving a larger $1 - R_{\text{ft}}$. At the time when the reliability of a fault-tolerant design drops to a certain value, say 0.5 (below which the system is of little use in practice), for instance, the RIF cannot go beyond 2 following its definition above, making normalized RIF less than 2. The FTBDM is found from Fig. 6 to have the normalized RIF of approximately 1.5 and 1.8 for $L = 2^4$ and 2^8, respectively, when its reliability equals 0.5 (at roughly time 0.03 and time 0.003, respectively, from Fig. 5).

VI. Design Extension

The reliable design addressed so far can tolerate at most one link or node failure per level. For a large system aimed at a long mission duration, it might be desired to consider an even more reliable architecture. In particular, we may wish to have a design capable of tolerating more than one failure per level. The proposed method can readily be extended to incorporate more than one column (i.e., stage) of spare nodes in the BDM for further reliability enhancement. While as many columns of spares as needed can be added to a BDM in general, we analyze only the case of incorporating two spare columns in the BDM here. A general case may be analyzed similarly.

Consider a BDM with $L = 2^n$ levels. One spare column, called $S1$, is placed at the rightmost side as before, and the second spare column, called $S2$, is added to the immediate right of stage $i, 0 \leq i \leq n$. The resulting structure, denoted by FTBDM_i^n, can be envisioned as composed of two parts, with the left part involving stages $0, 1, \cdots, i$ and $S2$, and the right part containing stages $i+1, i+2, \cdots, n$ and $S1$. An extra connection is established between nodes (g, l_1) and $(g+2, l_2)$ only if a C link exists between nodes (g, l_1) and $(g+1, l_2)$, for all $0 \leq g < n$. In addition, the regular S link between nodes (i, l) and $(i+1, l)$ is removed, and then two parallel connections are added between node (i, l) and spare node $(S2, l)$, as well as between spare node $(S2, l)$ and node $(i+1, l)$, for all $0 \leq l < L$. Figure 7 shows FTBDM_2^4. A structure so constructed is possible to tolerate up to two failures per level, provided that one failure is in the left part of the level and the other is in the right part. Spare column $S2$ is responsible for guarding against failures in the left part, whereas column $S1$ is for failures in the right part.

The overall reliability of FTBDM_i^n is dependent on the position of the second spare column i. The larger the i value, the more reliable the right part but the less reliable the left part, for a larger i means that $S2$ has to cover failures in a bigger part. It seems that the highest reliability is attained if spare columns are placed equidistantly. The exact reliability of FTBDM_i^n is very difficult to evaluate. However, we can estimate a lower bound on FTBDM_i^n reliability by making use of the lower bound expression derived earlier, based on the following observation. The left part of FTBDM_i^n consists of 2^{n-i} *independent* modules, each involving 2^i levels of $i+1$ nodes plus one spare. In Fig. 7, for example, the left part

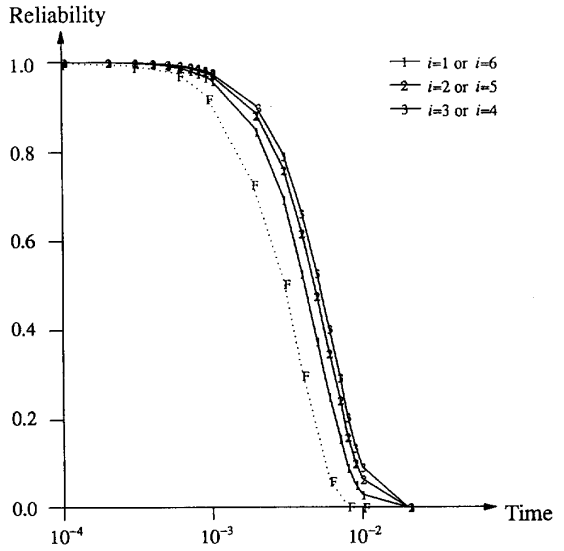

Fig. 8. Reliability comparison among FTBDM_i^8's for different i values under $\lambda = 1.0$ and $\lambda_{cs} = 0.1$ per unit time. (The dotted curve shows the FTBDM reliability given in Fig. 5.)

contains four modules, and each module can be viewed as one FTBDM with $L = 4$ levels. The left part has its reliability lower bounded by the product of its four constituent modules' reliability lower bounds. Similarly, the right part involves 2^{i+1} *independent* modules, and each module can be treated as one FTBDM with $L = 2^{n-i-1}$ levels. FTBDM_i^n exhibits reliability lower bounded by the product of the reliability lower bounds of its two parts.

Figure 8 illustrates the reliability lower bound of FTBDM_i^8 for various i under $\lambda = 1.0$ and $\lambda_{cs} = 0.1$ per unit time. As expected, the highest lower bound occurs at $i = 3$ or 4, namely, when $S2$ is placed in the middle so that spare columns are equidistant. If $S2$ moves toward either side, the reliability value drops consistently. The second spare column contributes to a noticeable reliability improvement.

VII. Conclusion

We have presented a reliable butterfly network suitable for the distributed-memory multiprocessor construction. Our design goal is to maintain the full rigid butterfly interconnection style even in the presence of faults (not just to assure that all workable nodes are connected), so that a reconfigured system can deliver the same high performance. The proposed design tolerates any single and many multiple node/link failures, significantly enhancing system reliability. Reconfiguration in response to an arising fault is simple and local, involves only a small fraction of system nodes, and can be carried out in a distributed fashion. It is shown that our design is more effective than a prior reconfigurable design. This design methodology can be extended to incorporate more than one column (i.e., stage) of spare nodes, arriving at different systems which meet different reliability requirements. With high reliability and fixed node degree, the proposed reliable butterfly networks appear fairly useful for constructing large sized systems.

References

[1] R. R. Koch, "Increasing the size of a network by a constant factor can increase performance by more than a constant factor," in *Proc. 29th Annu. Symp. Foundations of Comput. Sci.*, Oct. 1988, pp. 221–230.

[2] E. Upfal, "An $O(\log N)$ Deterministic packet routing scheme," in *Proc. 21st Annu. ACM Symp. Theory of Computing*, May 1989, pp. 241–250.

[3] T. Leighton and B. Maggs, "Expanders might be practical: Fast algorithms for routing around faults on multibutterflies," in *Proc. 30th Annu. Symp. Foundations of Comput. Sci.*, Oct. 1989, pp. 384–389.

[4] C.-L. Wu and T.-Y. Feng, "On a class of multistage interconnection networks," *IEEE Trans. Comput.*, vol. C-29, pp. 694–702, Aug. 1980.

[5] L. M. Napolitano, Jr., "The design of a high performance packet-switched network storage system," *J. Parallel and Distrib. Comput.*, vol. 10, pp. 103–114, Oct. 1990.

[6] B. Grey, A. Avizienis, and D. Rennels, "A fault-tolerant architecture for network storage systems," in *Proc. 14th Int. Symp. Fault-Tolerant Computing*, June 1984, pp. 232–239.

[7] V. Balasubramanian and P. Banerjee, "A fault-tolerant massively parallel processing architecture," *J. Parallel and Distributed Computing*, vol. 4, pp. 363–383, 1987.

[8] M. Blatt, "Effects of switch failure on soft-configurable WSI yield," in *Proc. 1990 Int. Conf. Wafer Scale Integration*, Jan. 1990, pp. 152–159.

[9] I. Koren and M. A. Breuer, "On area and yield considerations for fault-tolerant VLSI processor arrays," *IEEE Trans. Comput.*, vol. C-33, pp. 21–27, Jan. 1984.

[10] G. B. Adams, III and H. J. Siegel, "The extra stage cube: A fault-tolerant interconnection network for supersystems," *IEEE Trans. Comput.*, vol. C-31, pp. 443–454, May 1982.

[11] S.-Y. Kuo and W. K. Fuchs, "Reconfigurable cube-connected cycles architectures," *J. Parallel and Distrib. Computing*, vol. 9, pp. 1–10, May 1990.

[12] M. C. Howells and V. K. Agarwal, "A reconfiguration scheme for yield enhancement of large area binary tree architectures," *IEEE Trans. Comput.*, vol. 37, pp. 463–468, Apr. 1988.

Nian-Feng Tzeng (S'85–M'86–SM'92) received the B.S. degree in computer science from National Chiao Tung University, Taiwan, the M.S. degree in electrical engineering from National Taiwan University, Taiwan, and the Ph.D. degree in computer science from the University of Illinois at Urban-Champaign in 1978, 1980, and 1986, respectively.

He is currently an Associate Professor in Center for Advanced Computer Studies at the University of Southwestern Louisiana, Lafayette. From 1986 to 1987, he was a member of Technical Staff, AT&T Bell Laboratories, Columbus, Ohio. His current research interest is in the areas of parallel and distributed processing, fault-tolerant computing, high-speed networking, and high-performance computer systems.

Dr. Tzeng is on the editorial board of the IEEE Transactions on Computers, serves as Co-Guest Editor of a special issue of the *Journal of Parallela and Distributed Computing* on distributed shared memory systems, 1995, and is Co-Program Chair of the 8th International Conference on Parallel and Distributed Computing Systems, September 1995. He is a member of Tau Beta Pi, a member of the Association for Computing Machinery, and the recipiemt of the outstanding paper award of the 10th International Conference on Distributed Computing Systems, May 1990.