Partitionability of the Multistage Interconnection Networks

Yeimkuan Chang
Department of Information Management
Chun-hua Polytecnic Institute
Hsinchu, Taiwan R.O.C.
E-mail: ychang@cs.tamu.edu

Abstract – Partitionability allows the creation of many physically independent subsystems, each of which retains an identical functionality as its parent network and has no communication interference with other subsystems. We show that different permutation functions connecting the processors and the switches in the last stage of networks result in different partitionability. Based on a novel mapping scheme of MINs onto the hypercube structure, we show that the switches play a more important role on the subsystem availability than the processors. Subsystem fault tolerance of this class of MINs is also analyzed.

1 Introduction

Numerous MINs have been proposed for connecting multiple nodes in a multiprocessor. MIN's increasing popularity is related to their low cost compared with that of a full crossbar at a reasonable bandwidth. A class of MINs including baseline networks, regular SW banyan networks with spread and fanout of 2, indirect binary n-cube networks, and Omega networks have been shown to be topologically equivalent [1]. The IBM SP1/2 [2], CM-5[1] [3], and the BBN Butterfly [4] are good examples of multiprocessors that employ a MIN [5].

In a multiprocessor system, there is a possibility that a single task will degrade other tasks' executions by creating a hot spot in switching elements (SEs) and links. This situation can be further aggravated by a bad processor allocation scheme that does not utilize the interconnection network efficiently and uniformly. Therefore, it is advantageous to schedule the independent tasks to different portions of the system such that there is no communication interference among them.

The underlying theory of partitioning permutations is addressed in [6]. The partitionability of the baseline network, the generalized cube network, the indirect binary n-cube network, and the Omega network has been addressed in [7, 8, 9]. However, the impact on the network partitionability of different physical connections between processors and SEs in the last stage is generally ignored. The partitionability of two representative networks with 2^n processors, namely the indirect binary n-cube (called *n-icube*) and the Butterfly network (called *n-Butterfly*), is analyzed and compared in this paper.

The remainder of this paper is organized as follows. In Section 2, The partitionability of two groups of MINs based on the permutation functions between the processors and switches in the last stage of the networks is studied. A novel mapping scheme of these two groups of MINs onto the hypercube is developed. In Section 3, the SE fault tolerance analysis is given. Concluding remarks are given in the final section.

2 Partitionability of MIN

The permutation functions describing the connections between the processors and SEs in the last stage of an n-icube and an n-Butterfly are the "inverse shuffle" connection and "trivial" connection, respectively. Figs. 1 and 2 illustrate the interconnections of a 4-icube and a 4-Butterfly, respectively. In Fig. 1, processors 0 to 7 form a subnetwork which does not interfere with the other part of the network. The corresponding processors in the 4-Butterfly of Fig. 2 are processors 0 to 3 and 8 to 11, shown at the top half of the network. It can be seen that these eight processors in Fig. 2 also form a subnetwork without interference to other parts of the network. But, potential hot spots may take place on the links between stages 2 and 3. Another possible way to form a sub-Butterfly of eight processors is shown as solid lines in Fig. 2. Processors 0, 8, 1, 9, 6, 14, 7, and 15 can form a subnetwork without interference and hot spot links. The n-icube and the n-Butterfly differ in partitionabilities that result from different permutation functions between the last stage and processors. We will show systematically that different physical connections between the last stage and processors lead to different partitionability.

[1] The fat-tree architecture used in CM-5 is actually a MIN.

1063-7133/96 $5.00 © 1996 IEEE
Proceedings of IPPS '96

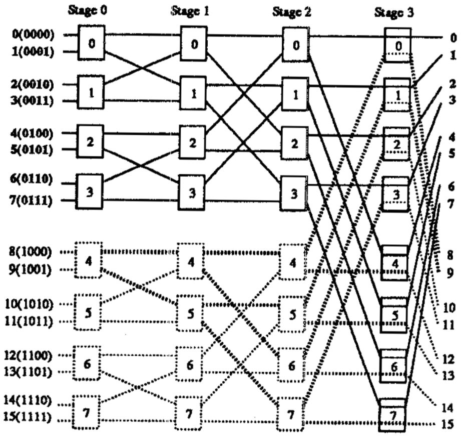

Figure 1: The 4-icube.

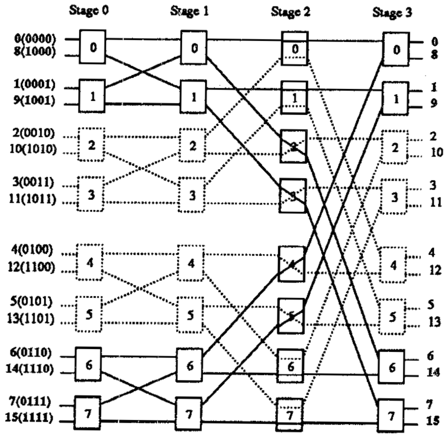

Figure 2: The 4-Butterfly.

In general, a set of interconnection functions are used to describe the connectivity of the interconnection paths between the SEs. A processor in a network of 2^n processors is represented as $P(p_{n-1}...p_1p_0)$. The label of an SE in stage i is $(p_{n-1}...p_1)_i$ or $(p_{n-2}...p_0)_i$. The interconnection function of a MIN, $I_i^\alpha[(p_{n-1}...p_1)]$ $= (t_{n-1}...t_1)_{i+1}$ denotes the link from output port α (0 or 1) of $SE(p_{n-1}...p_1)_i$ to $SE(t_{n-1}...t_1)_{i+1}$. For consistency, $P(p_{n-1}...p_10)$ and $P(p_{n-1}...p_11)$ are denoted as $I_{-1}^0[(p_{n-1}...p_1)_{-1}]$ and $I_{-1}^1[(p_{n-1}...p_1)_{-1}]$, respectively. The formal definition of the n-icube is first given as follows.

Definition 1: The n-icube is defined as follows: For the links preceding SEs in stage i $(0 \leq i \leq n-1)$, $N_{i-1}^0[(p_{n-1}...p_1)_{i-1}] = (p_{n-1}...0_i...p_1)_i$ and $N_{i-1}^1[(p_{n-1}...p_1)_{i-1}] = (p_{n-1}...1_i...p_1)_i$, from output ports 0 and 1 of $SE(p_{n-1}...p_1)_{i-1}$ to $SE(p_{n-1}...0_i...p_1)_i$ and $SE(p_{n-1}...1_i...p_1)_i$, respectively.

For the links between SEs of the last stage and the processors, $N_{n-1}^0[(p_{n-1}...p_1)_{n-1}] = P(0p_{n-1}...p_1)$ and $N_{n-1}^1[(p_{n-1}...p_1)_{n-1}] = P(1p_{n-1}...p_1)$, from output ports 0 and 1 of $SE(p_{n-1}...p_1)_{n-1}$ to $P(0p_{n-1}...p_1)$ and $P(1p_{n-1}...p_1)$, respectively. ■

Because Butterfly networks have different connections between the last stage and the processors, the addresses of processors need to be renumbered in order to implement correctly the n-icube self-routing scheme. The SEs are numbered in the same fashion as in the indirect n-cube, i.e., as 0 to $2^{n-1}-1$, from top to bottom. The processors are numbered 0, 2^{n-1}, $1, 1+2^{n-1}$, ..., i, $i+2^{n-1}$, ..., $2^{n-1}-1$, 2^n-1; i.e., processors $0p_{n-2}...p_0$ and $1p_{n-2}...p_0$ are attached to $SE(p_{n-2}...p_0)_0$. Thus, $(p_{n-2}...p_0)_i$ is assigned to an SE in stage i as its address. The formal definition of the n-Butterfly is given as follows.

Definition 2: The n-Butterfly is defined as follows: For the links preceding the SEs in stage i $(0 \leq i \leq n-1)$, $B_{i-1}^0[(p_{n-2}...p_0)_{i-1}] = (p_{n-2}...0_{i-1}...p_0)_i$ and $B_{i-1}^1[(p_{n-2}...p_0)_{i-1}] = (p_{n-2}...1_{i-1}...p_0)_i$ from output port 0 and 1 of $SE(p_{n-2}...p_0)_{i-1}$ to $SE(p_{n-2}...0_{i-1}...p_0)_i$ and $SE(p_{n-2}...1_{i-1}...p_0)_i$, respectively.

For the links between SEs of the last stage and the processors, $B_{n-1}^0[(p_{n-2}...p_0)_{n-1}] = P(0p_{n-2}...p_0)$ and $B_{n-1}^1[(p_{n-2}...p_0)_{n-1}] = P(1p_{n-2}...p_0)$ from output ports 0 and 1 of $SE(p_{n-2}...p_0)_{n-1}$ to $P(0p_{n-2}...p_0)$ and $P(1p_{n-2}...p_0)$, respectively. ■

The physical connections between stages in these two networks are exactly the same. The only difference occurs in the permutation functions between the SEs of the last stage and the processors. The Don't-Care symbol $(*)$ is used to denote 0 or 1; i.e., $(0p_{n-2}...p_0)$ and $(1p_{n-2}...p_0)$ can be denoted as $(*p_{n-2}...p_0)$. The SEs can be set to either *straight* or *cross* state.

Theorem 1: In an n-icube with the SEs in stage i set to the straight state, $P(*_{n-1}...\alpha_i \; ...*_0)$, $SE(*_{n-1}...\alpha_i... *_{j+1} *_{j-1}...*_0)_j$ for $j = 0...i-1$, and $SE(*_{n-1}...*_{j+1}*_{j-1}...\alpha_i...*_0)_j$ for $j = i+1...n-1$ form an $(n-1)$-cube where $\alpha = 0$ or 1 and $0 \leq i \leq n-1$.

Proof: see [10]. ■

Fig. 1 shows an example of a 4-icube with $n = 4$ and $i = 3$. The two 3-icubes shown in Fig. 1 are marked by solid and dotted lines. These two 3-icubes are obtained by setting the SEs of the last stage to the straight state. There exist n pairs of disjoint indirect binary $(n-1)$-icubes in an n-icube. Theorem 1 can be applied recursively to obtain subcubes of smaller sizes. The SEs utilized by a d-icube can be derived as

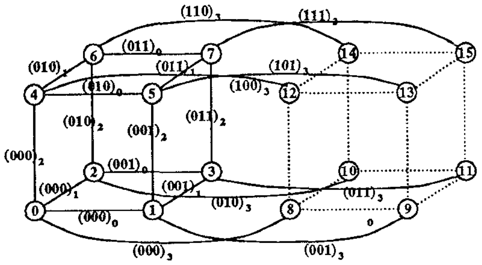

Figure 3: The mapping between SEs of an 4-icube and links of a direct 4-cube.

follows.

Theorem 2: A d-icube $(q_{n-1}...q_0)^2$ for $d \geq 1$ consists of $SE(q_{n-1} \ ... \ q_{i+1}q_{i-1} \ ... \ q_0)_i$ for all $i = 0$ to $n-1$.

Proof: see [10]. ■

The foregoing theorem simply states that the addresses of SEs in stage i employed in a d-icube are obtained by removing the i^{th} bit of the ternary address of the d-icube. As an example, the 2-icube $(1{*}0{*})$ in a 4-icube consists of (1) $SE(1{*}0)_0$; i.e., SEs 4 and 6 of stage 0, (2) $SE(1{*}{*})_1$; i.e., SEs 4, 5, 6, and 7 of stage 1, (3) $SE(10{*})_2$; i.e., SEs 4 and 5 of stage 2, and (4) $SE({*}0{*})_3$; i.e., SEs 0, 1, 4 and 5 of stage 3. In general, a d-icube utilizes $d{\times}2^{d-1}{+}(n-d){\times}2^d$ SEs, 2^{d-1} SEs from each of the d stages and 2^d SEs from each of the other $n-d$ stages.

Now we illustrate the relationship between a direct binary n-cube and an indirect binary n-cube. In an n-cube, a link is connected between two nodes if and only if their addresses differ in exactly one bit. We label the link connected to a processor $(p_{n-1}...p_0)$ along dimension i in an n-cube as $(p_{n-1}...p_{i+1}p_{i-1}...p_0)_i$. According to Theorem 2, the SEs employed by $P(p_{n-1}...p_0)$ are $SE(p_{n-1}...p_{i+1}p_{i-1}...p_0)_i$. Thus, there is an one-to-one mapping between the SEs of an n-icube and links of an n-cube. The SEs in stage i of the n-icube corresponds to the links along dimension i of the direct n-cube. Based on this mapping, the SEs in a d-icube of an n-icube are essentially the ones corresponding to the links which are incident to the processors of the d-icube.

Example 1: Fig. 3 shows the mapping between SEs of a 4-icube and the links of a 4-cube, where the SEs are labeled besides their corresponding links. The SEs (links) of the 3-icube (0***) are marked as solid lines.

Theorem 3: With the $SE(p_{n-2}...p_0)_i$ set to the straight state if $p_i \oplus p_{i-1} = 0$ and to the cross state if $p_i \oplus p_{i-1} = 1$ in an n-Butterfly ($\oplus$ is exclusive-or), $P(*_{n-1}...\alpha_i\beta_{i-1}...*_0)$, $P(*_{n-1}...\overline{\alpha_i\beta_{i-1}}...*_0)$, $SE(*_{n-2}...\alpha_i\beta_{i-1}...*_0)_j$, and $SE(*_{n-2}...\overline{\alpha_i\beta_{i-1}}...*_0)_j$, for $j = 1...n-1$ and $j \neq i$, form an $(n-1)$-Butterfly, where $\alpha_i\beta_{i-1} = 01$ or 00, and $1 \leq i \leq n-2$.

Proof: see [10]. ■

From the above theorem, we shall see that the n-Butterfly has less partitionability compared to the n-icube. The two $(n-1)$-Butterflies obtained by Theorem 3 are completely disjoint. There exist $n-2$ pairs of disjoint $(n-1)$-Butterflies in an n-Butterfly as opposed to n pairs in an n-icube[3].

Fig. 2 illustrates an example with $n = 4$ and $i = 2$. One 3-Butterfly consisting of processors $*00*$ and $*11*$ are shown in solid lines and the other 3-Butterfly consisting of processors $*01*$ and $*10*$ are shown in dotted lines. These two 3-Butterflies are obtained by setting SEs 0, 1, 6, and 7 to the straight state and SEs 2, 3, 4, and 5 to the cross state in the 2^{nd} stage.

Theorem 3 can be applied recursively to obtain subnetworks of smaller sizes. However, because of the characteristics of the Butterfly networks, how to obtain the subnetworks of smaller sizes in the n-Butterfly is not as simple as the n-icube. In order to formally represent the subnetworks of an n-Butterfly, we need the following definitions.

Definition 3: The symbol $*(i)$ is defined as a set of two i-bit binary strings which are complementary to each other, namely $*(i) = \{p_{i-1}...p_0, \overline{p_{i-1}}...\overline{p_0}\}$. We refer to $*(i)$ as an i-bit Don't-Care string. If the value of $p_{i-1}...p_0$ is known, $*(i)$ may be explicitly represented as $*(p_{i-1}...p_0)$. ■

Example 2: The Don't-Care string $*(3)$ may represent the set {000, 111}, {001, 110}, {010, 101}, or {011, 100}. Here, $*(000)$ and $*(111)$ represent the same set, {000, 111}. We can see that $*(0)$ and $*(1)$ represent {0, 1} which has the same meaning as the conventional Don't-Care symbol ($*$). If no confusion occurs, $*$ is used to denote $*(0)$ and $*(1)$. We refer to $*^i$ as i consecutive $*$'s.

Based on the above definition, the processors of the two $(n-1)$-Butterfly obtained by Theorem 3 can be represented as $*^{n-i-1}*(00)*^{i-1}$ and $*^{n-i-1}*(01)*^{i-1}$ $(1 \leq i \leq n-2)$, which consists of $n-1$ Don't-Care strings. As a result, a d-Butterfly can be obtained by applying Theorem 3 recursively as $*(i_{d-1})*(i_{d-2})*(i_{d-3})...*(i_1)*(i_0)$, where $i_{d-1} = 1$ and

[2] Throughout the paper, p_i denotes a binary bit; i.e., $p_i \in \{0,1\}$ and q_i denotes a ternary bit; i.e., $q_i \in \{0,1,*\}$.

[3] If we consider bi-directinal links used in IBM SP1/2, one more pair of subnetworks of size $n-1$ can be formed by using turnaround routing.

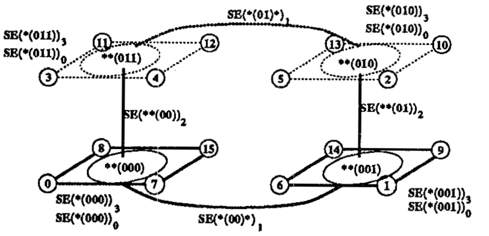

Figure 4: The mapping between SEs of a 4-Butterfly and links of a direct 4-cube.

$\sum_{j=0}^{j=d-1} i_j = n$. It essentially states that a d-Butterfly is obtained by dividing the n-bit binary string into d substrings. Each substring represents a Don't-Care string.

$P(p_{n-1}...p_0)$, $P(\overline{p_{n-1}}p_{n-2}...p_0)$, $P(\overline{p_{n-1}}...\overline{p_0})$, and $P(p_{n-1}\overline{p_{n-2}}...\overline{p_0})$, i.e. $P(**(p_{n-2}...p_0))$, forming a *cluster* of 4 processors, must be in the same sub-Butterfly. Therefore, the smallest sub-Butterfly is size 4. This is the immediate result from the fact that the n-Butterfly can only be divided into $n-2$ pairs of disjoint $(n-1)$-Butterflies stated earlier. To explicitly identify the SEs in a d-Butterfly, we need the following additional definition:

Definition 4: A k-complement of an i-bit Don't-Care string $*(p_{i-1}...p_0)$, $C_k[*(p_{i-1}\ ...\ p_0)]$, is defined as $C_k[*(p_{i-1}...p_0)] = *(p_{i-1}...p_k\overline{p_{k-1}}...\overline{p_0})$ for $0 \leq k \leq i$. ■

Note that $*(p_{i-1}...p_0) = C_0[*(p_{i-1}...p_0)] = *(\overline{p_{i-1}}\ ...\ \overline{p_0}) = C_i[*(p_{i-1}...p_0)]$. The SEs utilized by a d-Butterfly are thus derived as follows.

Theorem 4: The d-Butterfly $*(1)*(i_{d-2})...*(i_0)$, where $d \geq 2$, consists of $SE(*(i_{d-2})...*(i_1)*(i_0))_t$ for $0 \leq t \leq n-1$, and $SE(*(i_{d-2})...C_k[*(i_j)]...*(i_0))_t$ for all $j = 0$ to $d-2$ and $k = 1$ to $i_j - 1$, where $t = k + \sum_{l=0}^{l=j-1} i_l$.

Proof: see [10]. ■

Example 3: The 3-Butterfly $**(00)*$ in a 4-Butterfly consists of $SE(*(00)*)_t$ for $t = 0...3$, and $SE(*(01)*)_t$ for t = 2, which are shown as solid squares in Fig. 2.

Although the relationship between the n-Butterfly and a hypercube is not as evident as the n-icube. the n-Butterfly can still be represented by the links in the corresponding direct hypercube. We map an n-Butterfly onto an $(n-2)$-cube in which each link represents 4 SEs in the n-icube. According to Theorem 4, there exists a link representing $SE(**(p_{n-2}...p_0))$ and $SE(*C_k[*(p_{n-2}...p_0)])$ connecting the cluster $P(**(p_{n-2}...p_0))$ and the cluster $P(*C_k[*(p_{n-2}...p_0)])$ along dimension k where $1 \leq k \leq n-2$. The mapping is used for the purpose of easily characterizing the grouping property of processors in the n-Butterfly. As in the n-icube, it is easy to verify that the SEs of a d-Butterfly essentially correspond to the links incident to the processors of the d-cube in the corresponding direct n-cube. Fig. 4 shows the mapping of a 4-Butterfly onto a direct 4-cube. The SEs of the 3-Butterfly $**(00)*$ are shown as solid lines in Fig. 4.

3 Fault Tolerance Analysis

This section examines the effects of faulty SEs to the subsystem availability. The effects of a single faulty SE are analyzed first. The single fault analysis is then extended to the multiple fault analysis.

From Theorem 2, the d-icube $P(q_{n-1}...q_i...q_0)$ contains $SE(p_{n-2}...p_0)_i$, if $q_j = p_j$ or $*$ for $0 \leq j < i$ and $q_j = p_{j-1}$ or $*$ for $i < j \leq n-1$. In other words, any sub-icube containing $(p_{n-2}...p_i0p_{i-1}...p_0)$, or $(p_{n-2}...p_i1p_{i-1}...p_0)$, or both employs $SE(p_{n-2}...p_0)_i$. The processors in $(p_{n-2}...p_i * p_{i-1}...p_0)$ are the ones connected by the link associated to $SE(p_{n-2}...p_0)_i$ in the n-cube mapped from an n-icube. The impact of a faulty $SE(p_{n-2}...p_0)_i$ to the sub-icube availability is identical to that of two faulty processors, $(p_{n-2}...p_i0p_{i-1}...p_0)$ and $(p_{n-2}...p_i1p_{i-1}...p_0)$ for $0 \leq i \leq n-1$. We know that a processor belongs to C_d^n d-icubes since these C_d^n d-icubes containing a processor, $p_{n-1}...p_0$, can be established by replacing d of the n bits with $*$'s and there are C_d^n different ways to do it. Therefore, a faulty processor destroys C_d^n d-icubes. Similarly, based on the mapping of an n-icube onto an n-cube, the number of d-icubes destroyed by a faulty SE can be obtained as $2C_d^n - C_{d-1}^{n-1}$. Therefore, a faulty SE has a greater impact on the sub-icube availability than a faulty processor.

For the Butterfly networks, we have the following results.

Theorem 5: Given a faulty SE$(p_{n-2}...p_0)_t$ in an n-Butterfly, the d-Butterflies, $*(1)*(i_{d-2})...*(i_j)...*(i_0)$, are destroyed if (1) $p_{i_j-1+t_j}\ \cdots\ p_{t_j} \in *(i_j)$ for all $j = 0$ to $d-2$ and (2) there exists a j such that $t_j \leq t < t_{j+1} - 1$ and $p_{i_k-1+t_k}\ \cdots\ p_{t_k} \in *(i_k)$ for $0 \leq k < j$ and $j < k \leq d-2$, and $p_{i_j-1+t_j}\ \cdots\ p_{t_j} \in C_t[*(i_j)]$, where $n-1 = \sum_{l=0}^{l=d-2} i_l$ and $t_k = \sum_{l=0}^{l=k-1} i_l$.

Proof: see [10]. ■

It may be seen from Theorem 5 that $SE(p_{n-2}...p_0)_t$ in an n-Butterfly is contained in the 2-Butterfly $*(1)*(p_{n-2}\ ...\ p_0)$ or $*(1)*(p_{n-2}\ ...\ p_t\overline{p_{t-1}}\ ...\ \overline{p_0}) = *(1)C_t[*(p_{n-2}\ ...\ p_0)]$. It also indicates that the impact of a faulty $SE(p_{n-2}...p_0)_t$ to the sub-Butterfly availability is identical to that of processors $*(1)*(p_{n-2}...p_0)$ or $*(1)C_t[*(p_{n-2}...p_0)]$ if the faulty SE is not in the first or the last stage. In other words, $SE(p_{n-2}...p_0)_t$ destroys both of the 2-Butterflies, $*(1)*(p_{n-2}...p_0)$ and $*(1)C_t[*(p_{n-2}...p_0)]$. Similarly, the faulty SE in the first or the last stage has the same fault effect on the sub-Butterfly availability as a cluster of 4 processors associated with it. Based on the mapping of an n-Butterfly onto an $(n-1)$-cube, the number of d-Butterflies destroyed by a SE is C_{d-2}^{n-2} if the faulty SE is in stage 0 or $n-1$ or $2C_{d-2}^{n-2} - C_{d-3}^{n-3}$ otherwise.

The mapping of an n-icube and n-Butterfly onto a hypercube provides an easy way to characterize the network. As in the direct binary n-cube, two processors are said to be an antipodal pair if they cover all the $(n-1)$-icube in an n-icube. The addresses of the antipodal pair in an n-cube or n-icube differ in all the bits. There is only one antipode for a specific processor in an n-cube or n-icube. However, there are four antipodes for a specific processor, referred to the antipodal set, in the n-Butterfly. For example, the antipodal set of processor (0000) in Fig. 4 are processors in $**(000)$. To formally represent the antipodal set of a processor in an n-Butterfly, we need the following definition.

Definition 5: ALT is defined as a function which alternately complement the bits of $p_{k-1}...p_0$ from the right to the left. Formally, $ALT[p_{k-1}...p_0] = q_{k-1}...q_0$, where q_i and p_i are the i^{th} element from the right and $q_i = p_i$ if i is odd otherwise $q_i = \overline{p_i}$, for $0 \leq i \leq k-1$. For example, $ALT[p_3p_2p_1p_0] = p_3\overline{p_2}p_1\overline{p_0}$. ■

In general, if processors $p_{n-1}...p_0$ and $p_{n-1}ALT[p_{n-2}...p_0]$ in an n-Butterfly are faulty, all the possible $(n-1)$-Butterflies will be destroyed since $p_{n-1}...p_0$ and $p_{n-1}ALT[p_{n-2}...p_0]$ are always in different $(n-1)$-Butterflies. Since the four processors in cluster $P(p_{n-1}...p_0)$ are in the same sub-Butterfly, two faulty processors from cluster $P(p_{n-1}...p_0)$ and cluster $P(p_{n-1}ALT[p_{n-2}...p_0])$ will destroy all the $(n-1)$-Butterflies.

Let us define the antipodal pair of SEs in the n-icube (n-Butterfly) as the two SEs which if faulty will destroy all the possible $(n-1)$-cubes ($(n-1)$-Butterflies). Each SE in an n-icube or n-Butterfly may have more than one corresponding antipodal SEs. In general, the problem of finding antipodal SEs of a particular SE becomes a little complicated [4] since the faulty SEs can be located in any level or stage. Since a faulty SE has the same impact to the subsystem availability as two processors in an n-icube, the antipodal SEs of a particular SE can be obtained as follows.

Theorem 6: The antipodal set of an $SE(p_{n-2} ... p_0)_i$ in an n-icube consists of $SE(\overline{p_{n-2}} ... \overline{p_i}*\overline{p_{i-1}} ... \overline{p_{j+1}}\,\overline{p_{j-1}} ... \overline{p_0})_j$ for $j = 0..i-1$, $SE(\overline{p_{n-2}} ... \overline{p_i}\,\overline{p_{i-1}} ... \overline{p_0})_i$, and $SE(\overline{p_{n-2}} ... \overline{p_{j+1}}\,\overline{p_{j-1}} ... \overline{p_i} * \overline{p_{i-1}} ... \overline{p_0})_j$ for $j = i+1..n-1$.

Proof: see [10]. ■

The antipodal set of an $SE(p_{n-2}...p_0)_i$ is essentially the set of SEs covered by the 1-cube, $(\overline{p_{n-2}}...\overline{p_i} * \overline{p_{i-1}}...\overline{p_0})$. Note that $(\overline{p_{n-2}}...\overline{p_i} * \overline{p_{i-1}}...\overline{p_0})$ is the antipodal processors of processors $(p_{n-2}...p_i * p_{i-1}...p_0)$ connected by the link corresponding to $SE(p_{n-2}...p_0)_i$. According to the mapping between an n-icube and a direct n-cube, the antipodal set of an $SE(p_{n-2}...p_0)_i$ includes the SEs corresponding to the links incident to $(\overline{p_{n-2}}...\overline{p_i} * \overline{p_{i-1}}...\overline{p_0})$. Obviously, in the worst case, the number of faulty SEs which can be tolerated while maintaining a fault-free $(n-1)$-cube in an n-cube is one.

Theorem 7: The antipodal set of an $SE(p_{n-2} ... p_0)_t$ in an n-Butterfly, where $1 \leq t \leq n-2$, consists of $SE(ALT[*(p_{n-2}...p_t)]ALT[*(p_{t-1}...p_0)]])_i$ for $i = 0..n-1$, $SE(ALT[*(p_{n-2}...p_t)]C_i[ALT[*(p_{t-1}...p_0)]]])_i$ for $i = 1 .. t-1$, and $SE(C_i[ALT[*(p_{n-2} ... p_t)]]\, ALT[*(p_{t-1} ... p_0)]])_{i+t}$ for $i = 1 .. n-2-t$.

Proof: see [10]. ■

Note that the antipodal set of $SE(p_{n-2}...p_0)_t$ can also be obtained by deriving the intersection of SEs in all the $(n-1)$-Butterflies that do not contain $SE(p_{n-2}...p_0)_t$, i.e. the intersection of $SE[*_{n-2}...*_{i+1} *(p_i\overline{p_{i-1}}) *_{i-2} ...*_0]_j$ for all $j = 0$ to $n-1$ and $SE[*_{n-2}... *_{i+1} *(p_ip_{i-1}) *_{i-2} ...*_0]_i$ for $1 \leq i \leq t-1$ and $t+1 \leq i \leq n-2$. Obviously, the task of deriving the antipodal set of an SE is tedious. However, mapping an n-Butterfly onto a direct n-cube facilitates this task. For example, if $SE(011)_1$ in Fig. 4 is faulty, all the 3-Butterflies containing any processor of $**(011)$ and $**(010)$ are destroyed. Thus, the antipodal set of $SE(011)_1$ are marked as solid lines in Fig. 4.

Now we discuss the antipodal set of $SE(p_{n-2}...p_0)_k$ while $k = 0$ or $n-1$. If $SE(p_{n-2}...p_0)_k$ is faulty, all the $(n-1)$-Butterflies containing $**(p_{n-2}...p_0)$ are destroyed. The antipodal set of $**(p_{n-2}...p_0)$ are $*ALT[*(p_{n-2}...p_0)]$. Thus, the antipodal set of $SE(p_{n-2}...p_0)_k$ are all the SEs corresponding to the links incident to $*ALT[*(p_{n-2}...p_0)]$ in the mapped n-cube. For example, the antipodal set of $SE(000)_0$ in Fig. 4 are $SE[*(010)]_0$, $SE[*(010)]_3$, $SE[*(01)*]_1$, and $SE[**(01)]_2$.

As it has been shown that the antipodal set of a $SE(p_{n-2}...p_0)_i$ is essentially the set of SEs covered by the 1-cube, $(\overline{p_{n-2}}...\overline{p_i}*\overline{p_{i-1}}...\overline{p_0})$. The maximum number of faulty SEs, $\mathcal{K}(n, n-1)$, that can be tolerated while maintaining a fault-free $(n-1)$-cube in an n-

[4] The problem of finding a set of links in an n-cube that destroy all the $(n-1)$-cube is different. From [11], it is known that three links are sufficient.

cube is one. By using the divide-and-conquer technique, we can show that $2^m - 1$ faulty SEs can be tolerated while maintaining a fault-free $(n - m)$-cube in an n-cube. However, if n is much larger than m, we derive a better result as shown below.

Theorem 8: The maximum number of faulty processors that can be tolerated in an n-cube while maintaining a fault-free $(n - 2)$-cube is the minimum positive integer $r - 1$ such that

$$\binom{r-1}{\lfloor r/2 \rfloor - 1} \geq n. \qquad (1)$$

Proof: See [11]. ■

Remember that a faulty SE has the same effect on the sub-icube availability as two processors connected by the associated link in the mapping. We can immediately obtain the $\mathcal{K}(n, n-2)$ as follows.

Theorem 9: The $\mathcal{K}(n, n-2)$ of the n-icube is the minimum positive integer $r - 1$ such that

$$\binom{r-1}{\lfloor r/2 \rfloor - 1} + r - 1 \geq n \text{ and } n - r \geq 2. \qquad (2)$$

Proof: see [10]. ■

For example, an 8-icube can tolerate four SE faults while still maintaining a fault-free 6-icube. A 15-cube can tolerate five SE faults while still maintaining a fault-free 13-icube. Now we show the similar results for the Butterfly networks as follows.

Theorem 10: The antipodal set of an $SE(p_{n-2} \dots p_0)_t$ in an n-Butterfly consists of $SE(ALT[*(p_{n-2} \dots p_t)]ALT[*(p_{t-1} \dots p_0)])_i$ for $i = 0..n-1$, $SE(ALT[*(p_{n-2} \dots p_t)]C_i[ALT[*(p_{t-1} \dots p_0)]])_i$ for $i = 1..t-1$, and $SE(C_i[ALT[*(p_{n-2} \dots p_t)]]\, ALT[*(p_{t-1} \dots p_0)])_{i+t}$ for $i = 1..n-2-t$.

Proof: see [10]. ■

Note that the antipodal set of $SE(p_{n-2} \dots p_0)_i$ can also be obtained by deriving the intersection of SEs in all the $(n - 1)$-Butterflies that do not contain $SE(p_{n-2} \dots p_0)_i$, i.e., the intersection of $SE(*_{n-2} \dots *_{i+1} *(p_i \overline{p_{i-1}})\, *_{i-2} \dots *_0)_j$ for all $j = 0$ to $n - 1$ and $SE(*_{n-2} \dots *_{i+1} *(p_i p_{i-1})\, *_{i-2} \dots *_0)_i$ for $1 \leq i \leq t - 1$ and $t + 1 \leq i \leq n - 2$.

Theorem 11: The $\mathcal{K}(n, n-2)$ of the n-Butterfly is the minimum positive integer $r - 1$ such that

$$\binom{r-1}{\lfloor r/2 \rfloor - 1} + r - 1 \geq n \text{ and } n - r \geq 4. \qquad (3)$$

Proof: see [10]. ■

Corollary 1:
$\mathcal{K}(n, n-m) \geq 2^{m-2} \times \mathcal{K}(n-m+2, n-m)$.

Proof: see [10]. ■

4 Conclusions

We have shown that the indirect binary n-cube has more partitionability than the Butterfly network. Via the proposed mapping scheme, we derived the number of faulty SEs in an indirect binary n-cube or a Butterfly network of 2^n processors that can be tolerated while still maintaining a subsystem of a given size. We have shown that a switching element has the same fault tolerance effect on the subsystem availability as 2 (4 or 8) processors in the indirect binary n-cube (Butterfly network).

References

[1] C. L. Wu and T. Y. Feng, "On a Class of Multistage Interconnection Networks," *IEEE Trans. on Computers*, vol. C-29, no. 5, pp. 694–702, Aug. 1980.

[2] W. Gropp and E. Lusk, "Users Guide for the ANL IBM SPx DRAFT," Technical Report ANL/MCS-TM-00, Argonne National Laboratory, Argonne, IL, October 1994.

[3] C.E. Leiserson et. al., "The Network Arcitecture of the Connection Machine CM-5," In *Proc. ACM Symposium on Parallel Algorithms and Architectures*, pp. 272–285, 1992.

[4] BBN Advanced Computer Inc., *Inside the TC1000 Computer*, BBN Advanced Computer Inc., Cambridge, MA, February 1990.

[5] L. N. Bhuyan, Q. Yang, and D. P. Agrawal, "Performance of Multiprocessor Interconnection Networks," *IEEE Computer*, vol. 22, no. 2, pp. 25–37, February 1989.

[6] H. J. Siegel, "The Theory Underlying the Partitioning of Permutation Networks," *IEEE Transactions on Computers*, vol. C-29, no. 5, pp. 791–800, September 1980.

[7] H. J. Siegel, *Interconnection Networks for Large-Scale Parallel Processing: Theory and Case Studies*, New York, NY: McGraw-Hill Publishing Company, 1990.

[8] W. Lin and C. L. Wu, "A Distributed Resource Management Mechanism for a Partitionable Multiprocessor System," *IEEE Transactions on Computers*, vol. 37, no. 2, pp. 201–210, February 1988.

[9] L. M. Li, Y. Gui, and S. Moore, "Performance Evaluation of Switch-Based Wormhole Networks," In *Proc. International Conference on Parallel Processing*, pp. I-32–40, 1995.

[10] Y. Chang, "Processor Allocation and Fault Tolerance in Multiprocessors," *Ph.D. Dissertation, Texas A&M University*, 1995.

[11] M. Livingston, Q. Stout, N. Graham, and F. Harary, "Subcube Fault Tolerance in Hypercube," Technical Report CRL-TR-12-87, University of Michigan, Ann Arbor, MI, 1987.