In *Proc. 6th Distributed Memory Computing Conference* (1991), IEEE, pp. 34–41.

Fault Tolerance of the Cyclic Buddy Subcube Location Scheme in Hypercubes

(Preliminary Version)

Marilynn L. Livingston

Department of Computer Science
Southern Illinois University
Edwardsville, IL 62026-1653

Quentin F. Stout*

Elec. Eng. and Computer Science
University of Michigan
Ann Arbor, MI 48109-2122

Abstract

This paper examines the problem of locating large fault-free subcubes in multiuser hypercube systems. We analyze a new location strategy, the *cyclic buddy system*, and compare its performance to the buddy system, the gray-coded buddy system, and several variants of them. We show that the cyclic buddy system gives a striking improvement in expected fault tolerance over the above schemes and, since it can easily be implemented in parallel with little overhead, it provides an attractive alternative to these schemes. We also investigate the behavior of these location systems in the folded, or projective, hypercube, and find that the cyclic buddy system, which adapts naturally to this enhancement, significantly outperforms the other schemes. A combination of analytic techniques and simulation is used to examine both worst case and expected case performance.

Keywords fault tolerance, subcube location, subcube allocation, hypercube computer, buddy system, gray-coded system, folded hypercube.

1 Introduction

Parallel computers incorporating thousands of processors must be able to tolerate faulty processors and communication links if they are to achieve a usable mean-time-to-failure. This means, in particular, that such systems should be able to locate large fault-free subsystems efficiently. Systems with multiuser or multitasking capabilities should be able to perform this location efficiently too, for the problem of assigning a subsystem to a given user or task can be thought of as allocating a subsystem in the presence of faults, where the busy processors and dedicated communication links can be considered "faulty". For many of the well known interconnection topologies, including the hypercube, the problem of locating all fault-free subsystems is computationally intensive. Consequently, a reduction in the number of allocable subsystems is made to reduce the system overhead needed for their location and allocation. Reducing the number of allocable subsystems, unfortunately, can result in a substantial decrease in the fault-tolerance of the system.

This is particularly unfortunate with the hypercube, for one of its attractive features is that it offers a high degree of fault tolerance. Large fault-free subcubes may exist in the network despite the presence of many edge or node faults. Moreover, since most hypercube algorithms specify the dimension of the hypercube as one of the parameters, these algorithms can still be used in the presence of faults, although with some degradation. When the set of allocable subcubes is substantially reduced, however, a few faults can make all allocable subcubes faulty and cause the system to fail to allocate a subcube even when a fault-free one exists.

Both NCUBE and Intel offer hypercube multicomputers with multiuser and multitasking capabilities. Subcube allocation in these systems, as in most commercial hypercube systems, is accomplished using the *buddy* strategy. This strategy is not very fault-tolerant, for in a hypercube of dimension d, the buddy system recognizes only 2^{d-q} q-dimensional subcubes out of the possible $\binom{d}{q}2^{d-q}$ q-dimensional subcubes. In a hypercube of dimension 10, for example, the buddy system recognizes less than one percent of the existing 7-dimensional subcubes.

Several authors have studied strategies for the location of subcubes in hypercube multicomputers. Some approaches involve enlarging the set of subcubes recognized over that of the buddy system, other approaches involve computation- and/or memory-intensive algorithms for complete subcube recognition. Chen and Shin [4] introduced a gray-coded buddy strategy for subcube location that improves upon the buddy strategy, and proposed

*Partially supported by NSF/DARPA grant CCR-9004727

a multiple gray-coded scheme using $\binom{d}{\lfloor d/2 \rfloor}$ gray codes to effect complete subcube recognition. Al-Dhelaan and Bose [2] introduced a modified buddy strategy which improves upon the single gray-coded buddy scheme. In [16], we examined several subcube location schemes including multiple buddy and gray-coded buddy schemes, and introduced a new family of location schemes which generalized and improved upon the earlier schemes.

We make the distinction between the problems of subcube location and allocation, viewing subcube allocation as subcube location plus a policy to choose one of the located subcubes. To evaluate subcube allocation in an environment with dynamic allocation and de-allocation one must consider the properties of the subcube request sequence, the properties of subcube utilization time, and the penalties associated with not locating a subcube, as well as considering the location capabilities. In this paper, we restrict ourselves to the study of subcube location strategies, and leave the study of allocation strategies under various policies and request distributions for another time. Without restricting ourselves to a specific allocation policy, we can make useful comparisons among various location strategies by studying the number of faults required to make every locatable subcube faulty. For each subcube location scheme under consideration, we determine the minimum number of faults required to make each locatable subcube of a given dimension faulty (a worst case scenario), and determine the average number of faults required to make each locatable subcube faulty (expected case scenario), where the faults are assumed to be distributed uniformly randomly and independently. We contend that the *cyclic buddy* scheme, which we introduced in [15], should be the method of choice for subcube location in hypercubes. This scheme can be implemented in parallel with almost no system overhead [15], and, as our results here will show, it yields a substantial improvement in fault-tolerance over the buddy and gray-coded buddy schemes.

In Section 2, we describe the subcube location schemes we will study. The schemes under consideration are the buddy, double buddy, gray-coded buddy, double gray-coded buddy, and cyclic buddy. Section 3 shows some of our analytic results and computer simulations for these schemes in hypercubes of dimensions ranging from 6 to 20.

Recently, there has been considerable interest in an enhancement of the hypercube network in which an extra edge is included between each pair of diametrically opposed nodes [7, 8, 9, 11, 17]. This new structure, which has appeared under several different names (folded, twisted, enhanced, projective) will be referred to as a *projective* hypercube here. A d-dimensional projective hypercube has roughly one-half the diameter of the standard hypercube of dimension d, has $\binom{d+1}{q}2^{d-q}$ q-dimensional subcubes compared to the $\binom{d}{q}2^{d-q}$ q-dimensional subcubes in the standard hypercube of the same dimension, and retains the attractive features of the standard hypercube network. Unfortunately, subcube location schemes such as the buddy and gray-coded buddy cannot take advantage of the new edges of the projective hypercube. However, the cyclic buddy scheme adapts naturally to this network. In Section 4, we show analytic results and computer simulations that compare the performance, in the presence of faults, of the buddy and the cyclic buddy schemes in the projective hypercube for dimensions ranging from 6 to 20. Here, too, we find the cyclic buddy scheme is significantly more fault-tolerant than the other schemes.

Multiple cyclic buddy systems for the standard hypercube are discussed in Section 5. The problem of how to devise optimal cyclic buddy schemes which locate large non-overlapping sets of subcubes is introduced. The use of multiple cyclic buddy systems allows a gradual tradeoff to be made between the size of the set of locatable subcubes and the time allowed for the location strategy.

Proofs of the theorems are omitted due to space restrictions, but will appear in the final version of the paper.

2 Subcube Location Schemes

To facilitate our description of the subcube location schemes, we first introduce some notation. In a hypercube of dimension d, which we will denote by Q_d, each processor is assigned a unique identification number (id) that is a d-bit binary string. Two processors are directly connected provided their id's differ in only one component. Extending our notation of d-bit binary strings for processor id's, we will denote the subcubes of Q_d by strings from $\{0, 1, *\}^d$, where the number of $*$'s in the string is the dimension of the subcube.

2.1 Buddy Systems

The standard *buddy* system recognizes only the q-dimensional subcubes in which the low-order $d - q$ bits have the symbol $*$. For example, in a 5-dimensional hypercube, the buddy system would identify only the four 3-dimensional subcubes of the form $ab***$, where $a, b \in \{0, 1\}$. More generally, each permutation π of the integers $1, 2, \ldots, d$ gives rise to a single buddy system B_π which recognizes only the q-dimensional subcubes of the form $a_1 a_2 \ldots a_d$, where $a_{\pi(i)} = *$ for $d - q + 1 \leq i \leq d$. Notice that the standard buddy system corresponds to the case of the identity permutation. For each permutation π, B_π recognizes only 2^{d-q} of the possible $\binom{d}{q}2^{d-q}$ subcubes of dimension q in Q_d.

The buddy system is attractive since it is easy to implement. Moreover, it is an optimal scheme in the following sense. Suppose that a series of requests for subcube sizes is given, assume that no de-allocation is considered and that the presence of faults is ignored. Then the buddy system is an optimal allocation strategy in the sense that it fails to grant a request only if there is an insufficient number of available nodes to satisfy the request. In dynamic allocation and de-allocation, however, it can perform quite poorly. Studies of the behavior of subcube allocation in a dynamic situation have been made in [4, 5, 10] for the purpose of evaluating various policies governing the selection of which subcube to allocate when there are several available subcubes. Variations of the single buddy system have been suggested which allocate the union of several single buddy systems. For example, an orthogonal *double buddy system* consists of two single buddy systems B_π and B_τ, where π is the identity permutation and τ is the permutation that reverses the order of the bits. Thus, this scheme allocates q-subcubes in Q_d of the form $a_1 a_2 \ldots a_{d-q} * \ldots *$ together with the form $* \ldots * b_{q+1} b_{q+2} \ldots b_d$. For $q < d$, the orthogonal double buddy system allocates twice the number of subcubes as does the single buddy system. For example, if $d = 7$ and $q = 5$ then the standard single buddy system recognizes the 5-dimensional subcubes given by $ab * * * * *$, where $a, b \in \{0, 1\}$, and the orthogonal double buddy system recognizes, in addition, the 5-dimensional subcubes given by $* * * * * cd$, where $c, d \in \{0, 1\}$.

2.2 Gray-Coded Buddy Systems

Let g_d denote the binary reflected Gray code map from $\{0, \ldots, 2^{d-1}\}$ to d-bit strings. The standard single *gray-coded buddy system* locates q-subcubes that arise as pairs of $q-1$-subcubes of the form $\{a_1 \ldots a_{d-q+1} * \ldots *, b_1 \ldots b_{d-q+1} * \ldots *\}$, where $g^{-1}_{d-q+1}(a_1 \ldots a_{d-q+1})$ and $g^{-1}_{d-q+1}(b_1 \ldots b_{d-q+1})$ are consecutive mod 2^{d-q+1}. As in the case of the single buddy system, each permutation π of $1, 2, \ldots, d$ gives rise to a gray-coded buddy system G_π. Each G_π locates the same number of subcubes as does the orthogonal double buddy system.

To illustrate the standard single gray-coded buddy system, let $d = 7$, $q = 5$, and let g_3 denote the map { (0,000), (1,001), (2,011), (3,010), (4,110), (5,111), (6,101), (7,100) }. The 5-dimensional subcubes of Q_7 recognized by this scheme consist of the pairs of 4-dimensional subcubes of the form $a_1 a_2 a_3 * * * *, b_1 b_2 b_3 * * * *$, where $g_3^{-1}(a_1 a_2 a_2)$ and $g_3^{-1}(b_1 b_2 b_3)$ are consecutive mod 8.

The *double gray-coded buddy system*, first suggested in [4], locates q-subcubes by a pair of gray-coded buddy systems, one which corresponds to the identity permutation, and the other which corresponds to the permutation which reverses the order of the bits.

2.3 The Cyclic Buddy System

In contrast to the buddy and gray-coded buddy schemes, the location strategy which we now consider does not seem to arise naturally as a serial location system. We introduced this scheme, the *cyclic buddy system*, (*CBS*), in [15]. It locates q-subcubes of the form $a_1 \ldots a_d$, where a_i through a_{i+q-1} are $*$ for some i, with the subscripts calculated modulo d. For example, if $d = 7$ and $q = 5$, the *CBS* recognizes all 5-dimensional subcubes of the forms $ab * * * * *$, $*ab * * * *$, $* * ab * * *$, $* * * ab * *$, $* * * * ab *$, $* * * * * ab$, and $a * * * * * b$, where $a, b \in \{0, 1\}$. In general, a permutation, π, of the integers $1, 2, \ldots, d$, gives rise to a cyclic buddy system CBS_π which recognizes the q-dimensional subcubes of the form $a_1 \ldots a_d$, where, for some j, $a_{\pi(i)} = *$ for $i = j+1, j+2, \ldots, j+q$, where i is calculated modulo d.

Each cyclic buddy system locates $d2^{d-q}$ subcubes of dimension q, giving complete location for $q = d-1$ and $q = 1$. Thus, a *CBS* locates at least as many q-subcubes as the double buddy and the double gray-coded buddy systems, and locates strictly more than these systems when $d > 2$. We showed in [15] that each of the buddy, gray-coded buddy, and the cyclic buddy schemes can be implemented in $\Theta(d)$ time in parallel. Furthermore, the implied constants are such that the *CBS* implementation time is less than that required by either the double buddy or the double gray-coded buddy schemes.

3 Fault Tolerance of Location Schemes

The use of location procedures that recognize only a small subset of the available subcubes adversely affects the ability of the system to function in the presence of faults. Here we will consider these effects, in both worst case and expected case, for the location schemes described in Section 2. To describe these behaviors, however, we will need some additional notation.

Suppose A denotes a subcube location scheme, and $\mathcal{S}(A, d, q)$ denotes the set of subcubes of dimension q in Q_d locatable by A. Let $A_w(d, q)$ denote the minimum number of node faults that make faulty all the subcubes of $\mathcal{S}(A, d, q)$, and let $A_e(d, q)$ denote the corresponding expected number of node faults, where the faults are assumed to appear uniformly and independently among the 2^d nodes. We will use K, B, DB, G, DG, and C to denote the case of complete subcube recognition, the single buddy system, the double buddy system, the single gray-

coded buddy system, the double gray-coded buddy system, and the cyclic buddy system, respectively.

The relationships between the subcubes located by the various schemes are summarized by the following inclusions.

$$\begin{array}{ccc} \mathcal{S}(B,d,q) \subset & \mathcal{S}(DB,d,q) & \subset \mathcal{S}(K,d,q) \\ \mathcal{S}(B,d,q) \subset & \mathcal{S}(G,d,q) \subset \mathcal{S}(DG,d,q) & \subset \mathcal{S}(K,d,q) \\ \mathcal{S}(B,d,q) \subset & \mathcal{S}(C,d,q) & \subset \mathcal{S}(K,d,q) \end{array}$$

In general the inclusions are strict, except that all of the schemes recognize all d- and 0-dimensional subcubes, and the cyclic buddy scheme recognizes all $(d-1)$-dimensional subcubes of Q_d. Note that if $\mathcal{S}(A^1,d,q)$ is a proper subset of $\mathcal{S}(A^2,d,q)$ then $A_w^1(d,q) \leq A_w^2(d,q)$ and $A_e^1(d,q) < A_e^2(d,q)$.

In the worst case, as our next theorem indicates, the cyclic buddy system does no better than the single or double buddy or gray-coded buddy system. The values of $K_w(d,q)$ are not known for all d and q, but we include a few of the known values appropriate for comparison.

Theorem 3.1 *For $d \geq q$,*

(*i*) $C_w(d,q) = 2^{d-q}$,

(*ii*) [16], $B_w(d,q) = G_w(d,q) = DB_w(d,q) = DG_w(d,q) = 2^{d-q}$,

(*iii*) [6], $K_w(d,d-2) = \lg d + \Theta(\lg\lg d)$,

(*iv*) [6], $K_w(d,q) \geq 2^{d-q-1}\lg(q+3) - (d-q)\cdot \lg(d-q) - 2\lg\lg d$.

□

Exact analytic expressions for expected fault tolerance are extremely difficult to determine, particularly for $d-q > 1$. We see this reflected in the next theorem, in which which we only give bounds for the schemes under consideration.

Let

$$B^*(k) = x\sum_{i=1}^{x}\frac{1}{i},$$

where $x = 2^k$, and let

$$G^*(k) = \sum_{i=0}^{z-2} p(z,i)\frac{z}{z-i},$$

where $z = 2^{k+1}$ and $p(z,i)\binom{z}{i} = \binom{z}{i} - \binom{i}{z-i} - \binom{i-1}{z-i-1}$.

Theorem 3.2 *For $d \geq q$,*

(*i*) *For the buddy, double buddy, gray-coded buddy, double gray-coded buddy, cyclic, and complete location schemes, $A_e(d,q)$ is strictly monotonically increasing in d and strictly monotonically decreasing in q. Further, if $d-q$ is held fixed, then it is strictly monotonically increasing in d.*

(*ii*) [16], *For fixed $d-q=k$, $B_e(d,q)$ converges monotonically increasingly to $B^*(k)$ as $d \to \infty$.*

(*iii*) [16], *For fixed $d-q=k$, $G_e(d,q)$ converges monotonically increasingly to $G^*(k)$ as $d \to \infty$.*

(*iv*) *For each k there is a constant $DB^*(k)$ such that, for $d-q=k$, $DB_e(d,q)$ converges monotonically increasingly to $DB^*(k)$ as $d \to \infty$.*

(*v*) *For each k there is a constant $DG^*(k)$ such that, for $d-q=k$, $DG_e(d,q)$ converges monotonically increasingly to $DG^*(k)$ as $d \to \infty$.*

(*vi*) $C_e(d,d-1) = K_e(d,d-1)$,

(*vii*) $C_e(d,q) = \Theta(\lg d)$, *for fixed $d-q$,*

(*viii*) [3], $K_e(d,q) = \Theta(\lg d)$, *for fixed $d-q$.*

□

When $d-q$ is fixed, we see from part (ii) of the theorem that $B_e(d,q)$ is bounded by a constant approximately equal to $(d-q)2^{d-q}\ln 2$. Parts (iii), (iv), and (v) of the theorem show that when $d-q$ is fixed, each of $G_e(d,q)$, $DB(d,q)$, and $DG(d,q)$ is bounded by a constant, also. On the other hand, parts (vi), (vii), and $(viii)$ show that for $d-q$ fixed, $C_e(d,q)$ has the same order of growth as $K_e(d,q)$, which is logarithmic in d.

In [16], a study was begun to compare the fault-tolerance of various location schemes for $d = 20$ and $q = d-2$. Here we have extended this simulation to include smaller dimensions and also to include the cyclic buddy scheme. In Table 1, we show representative results from these studies for the buddy, double buddy, gray-coded buddy, double gray-coded buddy, and cyclic buddy schemes for dimensions $d = 6, 8, 10, \ldots, 20$ and $q = d-2$. We also include a column labeled "∞", which represents the limit as $d \to \infty$. For the buddy, double buddy, and gray-coded buddy schemes all values shown are exact, utilizing a dynamic programming approach based on observations in [16]. For the double gray-coded buddy scheme, an approach was used which mixed Monte Carlo techniques with the dynamic programming approach in order to reduce the variance below that of a simple Monte Carlo approach.

For the cyclic buddy and complete location schemes the values shown are based on simulation. In these simulations, one *trial* consisted of successively generating random faults until all q-subcubes became faulty, and recording the smallest fault number to fall in any q-subcube. Then, for any subcube location strategy, A, we would find and record the maximum fault number over all the q-subcubes locatable by A. For each *round*, which consisted

	d								
Scheme	6	8	10	12	14	16	18	20	∞
buddy	7.8	8.2	8.3	8.3	8.3	8.3	8.3	8.3	8.3
dbuddy	9.5	10.1	10.2	10.3	10.3	10.3	10.3	10.3	10.3
gray	9.3	9.9	10.0	10.1	10.1	10.1	10.1	10.1	10.1
dgray	11.1	12.0	12.2	12.2	12.2	12.2	12.2	12.2	12.2
cyclic	12.1	13.9	15.0	15.7	16.2	16.8	17.2	17.4	∞
complete	14.2	17.4	19.4	20.9	22.0	23.0	23.8	24.6	∞

Table 1: Expected Fault Tolerance of Some Location Schemes for $(d-2)$-dimensional Subcubes

of 1000 trials, the average maximum fault number was then recorded. The mean number of faults observed from thirty rounds are reported in the table. Since the standard deviations were all less than 0.2, they have not been explicitly shown.

The first four lines of Table 1 reflect the fact, obtained from Theorem 3.1, that for the simple location schemes, $A_e(d,q)$ is bounded by a constant when $d-q$ is held fixed. On the other hand, the results for the cyclic buddy system and the complete location scheme reflect the fact that $A_e(d,q)$ is unbounded when $d-q$ is fixed and d grows.

In order to make a more extensive comparison between the values of $B_e(d,q)$ and $C_e(d,q)$ for several values of q and d, we set up another simulation to only obtain estimates for the cyclic buddy scheme, comparing it to exact values for the buddy scheme. Again, working on the assumption that most of the interest in locating subcubes will be for large subcube dimension, we collected data for $q = d-3, d-2, d-1$ and $6 \leq d \leq 20$. Thirty rounds of 1000 trials each, were used to gather statistics on the average number of faults needed to make faulty all the q-dimenional subcubes recognized by the cyclic buddy scheme. Table 2 shows the mean number of faults observed from thirty rounds for the cyclic scheme, with the mean for the buddy scheme shown in parentheses. Standard deviations for the thirty rounds were all less than 0.4 and are not shown.

In Table 2, we continue to see the significantly superior behavior of the cyclic buddy scheme over that of the single buddy scheme. Within each column of the table, which represents a case of $d-q$ fixed, the values of $B_e(d,q)$ are nearly constant while the estimates for $C_e(d,q)$ continue to increase. Parts (i), (ii), (vi), and (vii) of Theorem 3.2 support these observations.

		q		
		$d-1$	$d-2$	$d-3$
	6	4.9 (2.9)	12.1 (7.8)	25.8 (18.2)
	8	5.4 (3.0)	13.9 (8.2)	32.8 (20.8)
	10	5.7 (3.0)	15.0 (8.3)	36.4 (21.5)
	12	6.0 (3.0)	15.7 (8.3)	38.1 (21.7)
d	14	6.2 (3.0)	16.2 (8.3)	39.4 (21.7)
	16	6.4 (3.0)	16.8 (8.3)	40.5 (21.7)
	18	6.6 (3.0)	17.2 (8.3)	41.7 (21.7)
	20	6.6 (3.0)	17.4 (8.3)	42.4 (21.7)
	∞	∞ (3.0)	∞ (8.3)	∞ (21.7)

Table 2: Values of $C_e(d,q)$ ($B_e(d,q)$)

4 The Projective Hypercube and the Cyclic Buddy Scheme

The *projective hypercube* of dimension d, denoted by PQ_d, is constructed from the standard hypercube Q_d by including an extra edge between each pair of diametrically opposed nodes. This enhancement to Q_d, which has been discussed informally since at least 1986, arose because many of the hypercubes in use at that time had at least one extra port on each node. Formal description of the projective hypercube has appeared in the literature in [7, 8, 9, 11, 17], although under different names such as "twistd", "enhanced", or "folded" hypercubes. These names, as with the name "projective", are suggested by the following construction. Let A and B be two node-disjoint d-dimensional subcubes of Q_{d+1}. Retain the edges that connect corresponding nodes in A and B, but transform, or twist, B so that each of its nodes is moved to the node that was diametrically opposed to it in B. Now collapse (project) B onto A, identifying pairs of corresponding nodes of A and B.

The projective hypercube contains many of the desirable properties of the standard hypercube network, such as scalability, symmetry, high degree of fault tolerance, low diameter, and good embedding properties. Although we restrict our study of PQ_d here to fault-tolerance behavior of the buddy and cyclic buddy location schemes, in future work we will investigate its embedding properties and additional subcube fault tolerance properties.

The q-dimensional subcubes in PQ_d can be viewed as arising either as q-dimensional subcubes of Q_d, or as a pair

of $(q-1)$-dimensional subcubes of Q_d. These pairs must be of the form $x_1 \cdots x_d$ and $y_1 \cdots y_d$, where each x_i is either 0, 1, or $*$, exactly $q-1$ of the entries are $*$, and y_i satisfies

$$y_i = \begin{cases} * & \text{if } x_i = * \\ 0 & \text{if } x_i = 1 \\ 1 & \text{if } x_i = 0 \end{cases}$$

This results in a total of $\binom{d+1}{q} 2^{d-q}$ q-dimensional subcubes.

Since PQ_d has many q-dimensional subcubes, one would expect to see an increase in subcube fault tolerance in PQ_d, and a consequent improvement among those location schemes that can use the extra edges. Unfortunately, the buddy and gray-coded buddy systems (and their double variants), cannot directly make use of these extra edges, so their worst case and expected case behaviors in PQ_d are the same as for Q_d. However, there is an indirect way to allow many hypercube location systems (including all of the ones studied here) to utilize the extra edges. To find q-dimensional subcubes, first find q-dimensional subcubes which ignore the extra dimension. Then find $(q-1)$-dimensional subcubes which ignore the extra dimension, and see if two of them are connected via the extra dimension. Although this offers an interesting alternative, we will consider here only the direct subcube location schemes in PQ_d.

The cyclic buddy system directly adapts to the extra dimension of the projective hypercube by just adding it to the list of dimensions to be cycled through. It locates $(d+1)2^{d-q}$ subcubes of dimension q in PQ_d, where the q dimensions used are consecutive in cyclic order. For example, it finds all 5-dimensional subcubes in PQ_7 of the form $a*****b$, $ab*****$, $*****ab$, $****abc \cup ****\bar{a}\bar{b}\bar{c}$, $***abc* \cup ****\bar{a}\bar{b}\bar{c}*$, $**abc** \cup **\bar{a}\bar{b}\bar{c}**$, $*abc*** \cup *\bar{a}\bar{b}\bar{c}***$, and $abc**** \cup \bar{a}\bar{b}\bar{c}****$, where $a, b, c \in \{0, 1\}$, and $\bar{x}$ indicates $1 - x$.

Let PA, PA_w, PA_e denote the quantities for the projective hypercube that A, A_w, A_e represent for the standard hypercube. From the definitions it is clear that $PC_w \geq C_w$, $PC_e > C_e$, $PK_w \geq K_w$, and $PK_e > K_e$. One can show that $PC_w(d, d-1)$ is 2 when d is odd and 3 when d is even. For $PC_w(d, d-2)$, it is 4 when d is congruent to 0 or 2 mod 3, and it is either 5 or 6 when d is congruent to 1.

Table 3 displays the results of some of our computer simulations of PC_e, comparing it to the exact values for the buddy scheme. The simulations were set up in the same manner as for the standard hypercube, using 30 rounds of 1000 trials in each round. In each trial, randomly generated faults were added to PQ_d until all the q-dimensional subcubes located by the cyclic buddy scheme became faulty. The average number of faults required in the 30 experiments is reported in Table 3. The standard deviations in these cases were all less than 0.4 and have not been displayed.

		q		
		$d-1$	$d-2$	$d-3$
	6	5.1 (2.9)	12.5 (7.8)	26.5 (18.2)
	8	5.5 (3.0)	14.4 (8.2)	33.6 (20.8)
	10	5.8 (3.0)	15.3 (8.3)	36.9 (21.5)
	12	6.1 (3.0)	16.0 (8.3)	38.7 (21.7)
d	14	6.3 (3.0)	16.5 (8.3)	40.0 (21.7)
	16	6.5 (3.0)	17.0 (8.3)	41.0 (21.7)
	18	6.6 (3.0)	17.4 (8.3)	42.0 (21.7)
	20	6.7 (3.0)	17.7 (8.3)	42.5 (21.7)
	∞	∞ (3.0)	∞ (8.3)	∞ (21.7)

Table 3: Values of $PC_e(d, q)$ ($PB_e(d, q)$) for the Projective Hypercube

As the results in Table 3 indicate, the difference in performance between the cyclic buddy scheme and the buddy scheme is even greater than that observed in the standard hypercube. Moreover, parallel implementation of a cyclic buddy scheme on a projective hypercube would be easy and would incur no larger overhead than the buddy strategy.

5 Multiple Cyclic Buddy Systems

As we have seen with both the buddy and gray-coded buddy schemes, the use of an appropriately selected second scheme of the same type can significantly improve the fault tolerance. For example, in the orthogonal double buddy scheme, or the double gray-coded buddy scheme, a second subset of recognizable subcubes is selected which has minimal overlap with the set of subcubes arising from the standard scheme of that type.

Given the standard cyclic buddy scheme, *CBS*, it is not clear how to select the permutation, π, that yields the "best" additional cyclic buddy scheme, CBS_π. That is, we want the recognizable subcubes of *CBS* and CBS_π to have minimal overlap to yield a "best" fault-tolerant set. To illustrate these issues, take $d = 10$, let π_1 denote the permutation of $1, 2, \ldots, 10$ given by $1, 3, 5, 7, 9, 2, 10, 8, 6, 4$ and let π_2 be the permutation given by $1, 3, 5, 7, 9, 4, 2, 10, 8, 6$. Both permutations have the property that, for any k, $2 \leq k \leq 8$, k consecutive numbers in either ordering do not occur as k consecutive numbers in the standard ordering. That is, the subcubes of dimension $10 - k$, where $2 \leq k \leq 8$, recognized by *CBS* are distinct from those recognized by CBS_{π_1} or by CBS_{π_2}. However, the permutations are not equivalent, for π_2 has the additional property

that any two of the 5-dimensional subcubes recognized by CBS_{π_2} differ by at least two dimensions from those recognized by *CBS*, and are thus "far" apart, which increases fault tolerance. It would be of considerable interest to formulate a useful definition of "best" and to devise a constructive way to determine a permutation that corresponds to the "best" CBS_π. We leave this as an open problem and turn now to the problem of using a set of cyclic buddy systems that will collectively recognize all subcubes of dimension at least q in a d-dimensional hypercube.

Let $C(d,q)$ denote the minimum number of cyclic buddy systems needed to recognize all subcubes of Q_d of dimension q or more. Then $C(d,q)$ is the minimum number of cyclic permutations of $\{1,2,..,d\}$ such that every sequence of distinct integers from $\{1,2,...,d\}$ of length $d-q$ appears in at least one of the cyclic permutations. For, if $\pi_1,\pi_2,\ldots,\pi_k$ denotes such a minimal set of permutations, then the multiple cyclic buddy scheme consisting of the k systems CBS_{π_i}, $1 \leq i \leq k$ recognizes all subcubes of dimension q or greater. As an illustration, it is straightforward to check that $C(6,4) = 3$ and that the set $\{123456, 163425, 146253\}$ is a minimum set of cyclic permutations with the above property. Theorem 5.1 shows that $C(d,d-2) = \lfloor d/2 \rfloor$ holds in general. (Due to space limitations the proof is omitted in this preliminary version.)

Theorem 5.1 *For $d \geq q \geq 1$,*

(i) $C(d,q) \leq C(d+1,q)$

(ii) $C(d,d-2) = \lfloor d/2 \rfloor$

□

Our proof of part (ii) of the Theorem gives a method for constructing a minimum set of cyclic permutations. From this, a set of $\lfloor d/2 \rfloor$ cyclic buddy schemes can be constructed whose union recognizes all subcubes of dimension $d-2$ or greater in a hypercube of dimension d. Furthermore, as we show in [15], such a location strategy can be implemented in parallel in $\Theta(d^2)$ time. In [15] there is a much more complicated algorithm that accomplishes complete location in $\Theta(d)$ time, but the simplicity of the scheme using the cyclic buddy system might be more attractive to use, particularly for feasible sizes of d.

6 Conclusion

Through our analytic results and computer simulations we have seen how the choice of a subcube location scheme can affect the fault tolerance of multiuser hypercube systems. As our results showed, the *cyclic buddy* system offers significant improvement over the most commonly used scheme: the buddy system. The cyclic buddy system proved to be significantly better, too, than suggested schemes such as the double buddy, gray-coded buddy, and double gray-coded buddy. We have shown in [15] that the cyclic buddy scheme can be easily implemented in parallel, with no more overhead than the buddy strategy, and can locate a largest available subcube in $\Theta(d)$ time. These results give strong support to our contention that the cyclic buddy strategy should be the subcube location strategy of choice.

Our simulation studies of subcube location schemes in the projective hypercube showed that the cyclic buddy system significantly outperforms the other schemes, for it adapts naturally to the additional edges of this enhanced hypercube.

Since the cyclic buddy system locates only $d2^{d-q}$ of the possible $\binom{d}{q}2^{d-q}$ subcubes of dimension q in Q_d, there is still considerable motivation to improve upon this scheme by considering the use of multiple cyclic buddy systems. The use of multiple schemes allows a gradual trade-off between improvement of fault tolerance of the location scheme over the resources allowed to implement the scheme. We have left open the problem of constructing optimal multiple cylic buddy systems.

References

[1] S. Al-Bassam, H. El-Rewini, B. Bose, and T. Lewis, "Efficient serial and parallel subcube recognition in hypercubes", *Proc. 5th Distributed Memory Computing Conf.* (DMCC5), (1990) pp. 324-332.

[2] A. Al-Dhelaan and B. Bose, "A new strategy for processor allocation in an n-cube multiprocessor", *Proc. Int. Phoenix Conf. on Comp. and Comm.* (Mar. 1989).

[3] B. Becker and H. Simon, "How robust is the n-cube?", *Info. and Comp.* **77** (1988), pp. 162-178.

[4] M.S. Chen and K.G. Shin, "Processor allocation in an n-cube multiprocessor using Gray codes", *IEEE Trans. on Computers* **36** (1987), pp. 1396-1407.

[5] S. Dutt and J.P. Hayes, "Subcube allocation in hypercube computers", *IEEE Trans. on Computers* **40** (1991), pp. 341-352.

[6] N. Graham, F. Harary, M.L. Livingston and Q.F. Stout,"Subcube fault tolerance in hyper-

cubes", *Jour. of Information and Computation* (to appear).

[7] A. Ghafoor, T.R. Bashkow, and I. Ghafoor, "Bisectional fault-tolerant communication architecture for supercomputer systems", *IEEE Trans. on Computers* **38** (1989), pp. 1425-1446.

[8] C.-T. Ho, "Full bandwidth communications on folded hypercubes", *Proc. Int. Conf. on Parallel Processing, Vol. I* (1990), pp. 276-280.

[9] C.-T. Ho, "Observation on the Bisectional Interconnection Network", *Technical Report*, IBM Almaden Research Center (1989).

[10] J. Kim, C.R. Das, and W. Lin, "A processor allocation scheme for hypercube computers", *Proc. Int. Conf. on Parallel Processing, Vol. II* (1989), pp. 231-238.

[11] S. Latifi, "The efficiency of the folded hypercube in subcube allocation", *Proc. Int. Conf. on Parallel Processing, Vol. I* (1990), pp. 218-221.

[12] S. Latifi and A. El-Amawy, "On folded hypercubes", *Proc. Int. Conf. on Parallel Processing, Vol. I* (1989), pp. 180-187.

[13] M.L. Livingston and Q.F. Stout, "Distributing resources in hypercube computers", *Proc. 3rd Conf. on Hypercube Concurrent Computers and Appl.* (1988), pp. 222-231.

[14] M.L. Livingston and Q.F. Stout, "On problems of distributing resources in parallel computers", in preparation.

[15] M.L. Livingston and Q.F. Stout, "Parallel allocation algorithms for hypercubes and meshes" in *Proc. 4th Conf. on Hypercube Concurrent Computers and Appl.* (1989), pp. 59-66.

[16] M.L. Livingston and Q.F. Stout, "Fault tolerance of allocation schemes in massively parallel computers", *Proc. 2nd Symp. Frontiers of Massively Parallel Computation* (1988), pp. 491-494.

[17] N.-F. Tzeng and S. Wei, "Enhanced hypercubes", *IEEE Trans. on Computers* **40** (1991). pp. 284-294.