Mathematical & Computational Applications, Vol. 7, No. 3, pp. 189-203, 2002

A METHOD TO INCREASE THE SIZE OF HEALTHY SUBCUBES SET IN A HYPERCUBE SYSTEM WITH NODE OR LINK FAILURES

Novruz Allahverdi
Department of Electronics and Computer Education, Technical Education Faculty, Selçuk University, Campus, 42031, Konya,Turkey. noval@selcuk.edu.tr

Abstract-In this study we considered and analyzed the different cases of node- and link-faults in a hypercube multiprocessor. We revealed that, the direct use of sharp product operation is not sufficient to discard only computational part (processor and memory), when only this part of node is faulty. We also showed that in case when some links in communication part (router) incident to a healthy node are faulty, the sharp product operation does not allow to leave the healthy links and the node incident to these links in the set of healthy subcubes. In order to remove this lack we propose in this study the formal procedures with aid of which we can subtract first only the faulty node, excluding the healthy links incident to this node and second only the faulty links, excluding also the healthy nodes incident to these links of such systems.

Keywords-Hypercube, fault tolerance, healthy subcubes, cube algebra, computational part, communication part.

1. INTRODUCTION

A hypercube is a distributed parallel system consisting of 2^n identical processors, each provided with its own sizable memory and interconnected with n neighbors. It has a homogeneous symmetric structure and has necessarily rich connectivity [4, 5]. It also has useful topology in which many other topologies, such as meshes, rings, trees, etc. can be embedded. In the 1980's such systems have been designed and started to be used. The Ncube's 3200 and 6400, Intel's iPSC series (iPSC/1, iPSC/2, iPSC/860), the Amatek's series (S-14), etc. are among them, that can have from 128 up to 4096 processors [14, 21, 30].

As the size of a hypercube grows, the probability of some processors or links in the system may have fault increases. One of the important issues in these systems is how to communicate messages in the presence of component failures. Reliable data communication is essential, especially when hypercubes are used for safety-critical and time-critical applications requiring high reliability [7]. It is very important to detect and isolate faulty processors to ensure correct completion of computation in such applications [7, 11, 35, 36].

In this paper, we attempt to provide procedures to overcome the effects of faulty components and to exploit a fault-free architecture in the presence of node and link faults. We consider node design, where each node contains two units, namely, a computational part (CPP) and a communication part (CMP) that can operate independently [21, 30, 20, 24]. The CMP often is called a *router*. The routes have the capability of forwarding the messages that are received from other routes toward the destination node. We assume that all the faults are static and detected before the reconfiguration data procedure starts.

In some studies [7] only node faults are considered and, as a result, the links incident on these faulty nodes are discarded from the further manipulation, whereas these links are non-faulty and can participate in data communication process. Like this, if links are faulty, then fault-free nodes incident on these links are also discarded and due to this, these nodes can not participate in computational and communicational processes, though they are non-faulty.

Formally, the operation of node or link discarded from any cube can be executed by the help an operation named coordinate subtraction (sharp product). But this operation does not directly allow the subtraction of only fault node or only fault link from the cube (or subcube). Dealing with this, we attempt to exploit a procedure where only faulty nodes and only faulty links can be discarded from the cube (subcube) with application of operations of cube algebra. With this we extend a set of non-faulty components of cube. Two procedures are developed which we call *node only coordinate subtraction* and *link only coordinate subtraction*. These procedures help formally to determine a set of fault-free subcubes upon which subcube allocation, data communication processes such as a routing, broadcasting, etc may be performed.

On the other hand, the number and the scattering of faulty components in hypercube are very important. In many cases, to achieve % 100 fault tolerances, the number of faulty components is limited by n or near n [6-11, 16, 17, 24-26, 31-36]. Latterly this limit was risen to $2n-k$, where k =1,2,3. If the number of faulty elements becomes greater than n, the developed algorithms are insufficient for routing, broadcasting, subcube and task allocation and deallocation, etc. processes in hypercube. We deal with the question: has any path from a source node to a target node in presence of arbitrary number of faulty elements, when these faulty elements are placed randomly in hypercube? It is obvious, that there is not a general solution of this problem. Only it may be possible to calculate a minimum number of faulty nodes (or links) that cause the destruction of the paths in n-dimensional cube. Grahan at all [18] calculated these values of the number of such nodes and links in worst cases. They determined, for example, for n=4 the communication between all nodes in hypercube is remained noninjured, while in the first case 5 nodes and in the second case 8 links are faulty. Clearly, the communication topology depends on the place of faulty components in hypercube and it may be happen randomly. On the other hand the following question remains open: what is the minimum number of faulty nodes and links mixed in general, that cause to injure the paths in n-dimensional cube?

Our developed formal procedures, early and short version which has described in [2], allow calculating a set of fault-free paths in faulty hypercube, without answering this question. After executing these procedures one receives a configuration that has only non-faulty components and due to it can realize a data communication between fault-free nodes. It is clear, the source node and the target node must not be placed in isolated subcubes. These procedures do not depend on the number of faulty components and it is indifferent from scattering of these components in hypercube.

To develop these procedures we use the operations of cube algebra, considered briefly in section 2. In section 3 we consider the structure of a hypercube node. The procedures that we developed are described in section 4. The paper concludes in section 5.

2. ON OPERATIONS OF CUBE ALGEBRA

The cube algebra was developed to execute some operations on the binary cubes (hypercubes). It includes almost all operations of the set theory and three own operations, that are called *coordinate product (star product, consensus or ☆-operation), coordinate subtraction (sharp product* or *#-operation)* and *coordinate intersection (∩- operation).*

Earlier, the cube algebra was used the Boolean functions in the form of cubes. Later, the operations of cube algebra were used for minimization of switching functions [12, 29, 22]. Now these operations are also use for subcube allocation problem [15] and for routing of the information over fault-free nodes and links in hypercube [3, 4] and for determination of neighborhood of subcubes in a faulty hypercube [1]. In these studies, it is shown that these operations are a good tool for definition of complete non-faulty subcubes of hypercube with faulty, prohibition or bused components, also for definition of the paths, including the shortest path from any vertex to other vertex in hypercube.

The operations of cube algebra were explained in detail in [13, 28]. We also modified the interpretation of these operations, dividing of its execution to two parts, where first we define vector of respective operation, second we define the ending result based on the vector. Thus we can apply these intermediate results (vectors) to determine on some other parameters in hypercube, for example; Hamming distance [28, 23]. Since, we also use these operations, let us consider them briefly.

Following [28, 13, 12, 23, 3], the *coordinate subtraction* of two cubes C_1 and C_2, denoted by $C_1 \# C_2$, is the set of subcubes including the nodes from C_1, but not from C_2. For example, $**0\#*11=**0$ or $\{**0,1**\}\#110=\{0*0,*00,10*,1*1\}$. The *coordinate product* of two cubes C_1 and C_2, denoted by $C_1 \star C_2$, is the subcube founded in C_1 and C_2 simultaneously or the subcube one part of which is in C_1 and the other is in C_2. For example, $*00 \star 1*1=10*$. The *coordinate intersection* of two cubes C_1 and C_2, denoted by $C_1 \cap C_2$, is the subcube presented in C_1 and in C_2 at the same time. For example, $0*0 \cap 10* = \varnothing$ or $10* \cap 1*1=101$.

After executing these operations, one can obtain repeated cubes and cubes with smaller dimensions which can be included in the cubes with bigger dimensions. Everywhere in this study, we will assume that the repeated cubes and the cubes with smaller dimensions included in the cubes with bigger dimensions are avoided.

The coordinate subtraction operation was applied to find the local prime implicants of the switching functions. It was later known that this operation is a good tool for definition of complete non-faulty subcubes of hypercube with faulty or prohibition vertices [15], also for the definition of the paths, including the shortest path from any vertex to other vertex in hypercube [3]. To discard the faulty components from the hypercube the subtraction operation may be used. By means of the coordinate product, the following operations may be realized: 1) Determination of the Hamming distance between the cubes, which is the number of links between them along the shortest path; 2) Intersection of two or more cubes with the aim of definition of their common parts (subcubes); 3) Unification of two m-cubes A and B in the $(m+1)$-cube C. The intersection operation is very useful for the determination of the common parts of

cubes. At first, it was applied to definition of the parts of some prime implicant for switching functions which are common with other prime implicants. This allows us to answer the question: is the given prime implicant extreme? From the point of routing in hypercube, the coordinate intersection operation can be used for revelation of the common parts of cubes with the aim of defining the parts of passage from one subcube to another. But cube algebra's operations, especially, subtract operation can not directly take into consideration the fact that the hypercube node consists from two parts: computational part (a processor and local memory) and communication part (router).

3. NODE STRUCTURE

Each hypercube node consists of *a computational part* (CPP) and *a communication part* (CMP) (Fig. 1) [21, 30, 24, 20]. The CPP consists of a node processor and some local memory. The CMP or router has a crossbar switch with $(n+1)$-inputs and $(n+1)$-outputs. The neighboring nodes are connected through n-input and n-output links. The processor, attached to the router, uses the other two lines. The router can connect multiple inputs to multiple outputs simultaneously as long as there is no destination conflict. If multiple messages are to be delivered to the processor, it is assumed that the router can accept all the messages. The router is responsible for all communications. It sends messages generated by the local processor over the system to the destination node. Each router compares the destination address of a message with its own address. If they match, the message is delivered to the local processor. Otherwise, the router chooses one of its neighboring nodes to transfer the message.

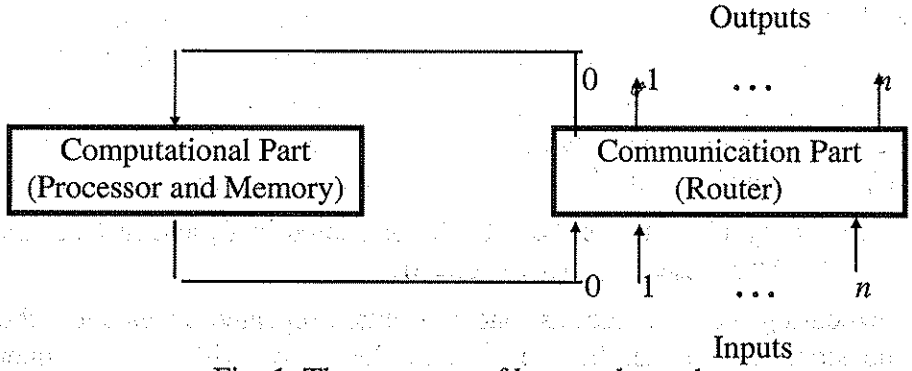

Fig. 1. The structure of hypercube node

In a faulty node where it is necessary to perform reliable data communication, subcube allocation and similar processes, a faulty node is generally not considered as a node consisting of two parts. But in a faulty node, different type of faults may take place, which must be taken into consideration. Possible cases of faults in a faulty node showed in Table 1. In this Table we show fault and faultless by 0 and 1, respectively

When the CPP and CMP are faulty (Case No 1 in Table), that means the node is faulty. Then this node must not participate in computational and communication processes in hypercube. Therefore, it must be taken out from the other cubes. This may be directly performed by the aid of the coordinate subtraction operation.

In the case when the CPP of a node is faulty (Case No 2 in Table), that means the node may participate only in communication process, so the respective node itself can not receive and send any information, but its CMP may organize message

communication between adjacent nodes to itself. This case will be considered in detail further.

Table 1. Fault and faultless of CPP and CMP

Case No	Computation Part	Communication Part	Situation of Nodes and Links
1	0	0	Node is faulty
2	0	1	CPP is faulty, CMP is not faulty
3	1	0	CPP is not faulty, CMP is faulty
4	1	1	Node is not faulty
5	1	F-H*	CPP is not faulty, some links in CMP are faulty
6	0	F-H*	CPP is faulty, some links in CMP are faulty

* F-H means that some links in the router are faulty and the rest is healthy, which may act normally.

If the whole CMP is faulty (Case No 3 in Table 1), then the respective node will not participate in communication process, i.e. it can not receive and send any information. Therefore, due to this fact, the whole node is isolated from the other nodes in hypercube. But since the CPP is not faulty, it may execute any calculation. From the parallel processing in hypercube point of view, this case does not play a great role and therefore, in order to achieve a set of healthy subcubes, this node may be subtracted from a set of the other subcubes.

When the node is not faulty (Case No 4 in Table 1) it is clear, that the node will be included in a set of healthy subcubes.

If a link or more links (but not all links) in a CMP are faulty (Case No 5), then the respective node can communicate only with definite neighbors. In this case, in order to reach a set of healthy subcubes we must subtract only a link (or links) without the nodes incident to this link (links) from the set of the subcubes. The coordinate subtraction operation does not directly allow this procedure. One way to obtain this procedure is the splitting of m-cube ($m \geq 2$) to m pieces of 1-cubes and execute the subtract operation of only the faulty links from the set of subcubes. But in many cases the splitting process in hypercube is not desirable (for example, in subcube and task allocation problems). On the contrary, in these cases it is required to have the cubes (subcubes) with the dimensions as greater as possible.

If a CPP and some links (but not all links) of CMP of corresponding CPP are faulty (Case 6), then the respective node can execute only the communication functions along the healthy links. The CPP of this node can not send and receive any information. In this case one must subtract both the CPP and the faulty links. Apparently, the Case 6 in Table 1 can be brought to the Case 5.

Thus, from the analysis of situation of the node we see that in two cases, namely in Case 2 and in Case 5, we need to exploit the procedures to determine a set of healthy subcubes, which will be the set for further manipulation in hypercube.

4. THE ONLY NODES AND ONLY LINKS SUBTRACTION PROCEDURES

Every node V in hypercube has label or address $v_n\, v_{n-1}...v_1$ with $v_i \in \{0,1\}$, where v_i is called the ith *bit* or ith *dimension* of the address. Sometimes, the i is called ith *coordinate*. In this case v_i is called the *value* of the coordinate i. The important property of the hypercube is that it can be constructed recursively from lower dimensional cubes (subcubes). Each subcube S_m (or m-subcube) has a unique address as $s_n\, s_{n-1}....s_1$ with $s_i \in \{0,1,*\}$, where exactly m $(0 \le m \le n)$ bits take the value * (* is a don't care symbol). A complete hypercube itself can be considered as a special subcube where all the bits of its address take the * value. Each node is also a special subcube in which no bit of its address takes the * value, and called 0-cube. Each link has one bit that takes the value * and called 1-cube, a quadrangle is called 2-cube and has two bits that take the value *. A cube is 3-cube, if its address has three * , etc.

There may be faulty nodes and/or faulty links in a hypercube. In this case, the hypercube is called *a faulty hypercube*. In case when the non-faulty nodes and links exist in n-cube, it is called *a complete cube*. A cube is called *a subcube of complete n-cube,* when at least one dependent (0 or 1) coordinate exists in the cube with dimension n. A cube S is the *subcube* of cube C if $C \star S = S$. A subcube S is called *a maximal subcube,* if subcube S is not subcube of any other subcube. Otherwise the subcube S is called *non-maximal*. As we note in Section 2, in this study we assume that the repeated and non-maximal cubes, if they exist, will be avoided after executing the operations of cube algebra. A subcube F is called a *complete faulty subcube (cube)* if F is a subcube (cube) including only faulty nodes and/or links. If a subcube does not have any faulty element, then it is called *a healthy subcube*. Though the finding of maximal subcubes is NP-hard [15], we attempt here to use the cube algebra's operations for our approach. In this study we mean that *the size of subcube set* is the number of elements in this set.
As applied to hypercube vertices of which there are the processors, computers or other similar active objects, the prime implicant term can be interpreted as maximal subcube including only non-faulty or forbidden vertices. Theorem 1 allows determining the method of finding the set of the non-faulty and maximal subcubes (NMS) in faulty hypercube [3].

Theorem 1. If F is a set of complete faulty cubes, then the set of NMS of n-cube including F is determined by the # -subtracting the set F from the complete n-cube and avoiding non-maximal cubes from the result.

Since in this study we will manipulate only cubes with nodes and links, let $V=v_1v_2...v_n$ is a vertex (node) and $L=l_1l_2...l_n$ is a link (edge), where $v_i \in \{0,1\}$; $\exists\, l_j=*$, $1 \le j \le n$ and for the remained values $i \ne j$ $l_j \in \{0,1\}$.

When we will subtract the CPP of a node, we will call this operation *subtract only node* (or *subtract CPP*), since in this case the links incident to this node will not be subtracted. Similarly, we call the subtraction only link from the cube as *subtract only link*, since the nodes incident to this link (or links) will not subtracted.

A. Subtract Only Nodes

The connection between the nodes of a hypercube is called *communication topology* of an n-cube (or subcube)

Theorem 2. Let only the CPP of node V is faulty and let A is a set of subcubes including the node V. Then this fault will not destroy the communication topology of subcubes in the set A.

Proof: When CPP of node V is faulty and CMP is not faulty, that means this node may participate in the communication processes only. That is, the CPP can not receive and send any information to other nodes. We may say that the CPP is isolated. Therefore the communication topology will remain as before, when CPP of this node was not faulty.

From the Theorem 2 we can conclude the following corollaries:

Corollary 1. When CPP of a node V, which is included in a set A, is faulty, the communication topology of subcubes of the set A will not change.

Corollary 2. When CPP of a node V, which is included in a set A, is faulty, then this CPP is isolated from the set A.

Abovementioned judgements mean that, when CPP of a node is faulty, a cube that includes this node will have the possibility to communicate messages between adjacent nodes excluding it. It can be seen from here that the fault of a CPP does not influence to communication processes between the nodes of hypercube. But we must show this node in the description of the cube, in order to exclude it from further computational processes. Therefore, we will show it as $\{A/V\}$, where V is a node from the subcubes in a set A and CPP of this V is faulty.

Example 1. Let in 3-dimensional cube the CPP of the node 001 is faulty. The little circles show CPP of the nodes and the big circles show CMP (Fig. 2, a). The shaded circle is faulty CPP of the node 001. If we use the coordinate subtract operation to define healthy subcubes and communication topology of this 3-cube, then we will obtain: ***#001={1**,*1*,**0} (Fig. 2,b). As seen from the set {1**,*1*,**0}, here we do not have the links 00*,*01 and 0*1, though they are not faulty. Therefore, although the set of healthy subcubes is the same, we must take into account fault of CPP of the node 001. Thus we will have {***/001} (Fig. 2, c).

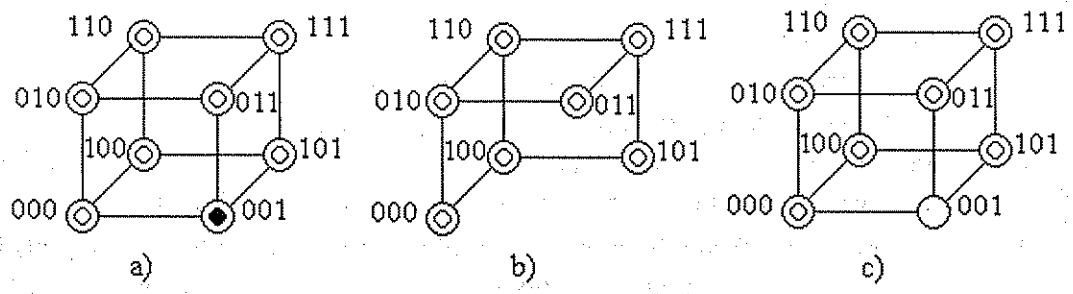

Fig.2. Example for subtracting only the CMP of node

B. Subtract Only Links

When a CPP is not faulty and one or more links (but not all links) of this node's CMP are faulty (Case No 5 in Table), in order to reach a set of healthy subcubes, we must subtract only a link (or links) from the other set of subcubes. It is clear, that a #-operation will also discard the nodes incident to these faulty links. But, as a matter of fact, these nodes are not faulty and they must remain in the set of healthy subcubes, say A. Consequently, the healthy links incident to these nodes will also discarded from the set A. If we will add these healthy links to the set A we will obtain the set called *partly*

extended set, say P_E. But in the set P_E it may be subcubes, which can unite and organize the subcubes with more dimensions. In the case where we obtain the subcubes with more dimensions in the set P_E, we call this set the *extended set*, say E. The following method allows achieving this procedure.

Lemma 1. Let $V=v_1v_2...v_i...v_n$ and $U=u_1u_2...u_i...u_n$ are nodes in an n-cube, neighboring along the coordinate i ($1\le i\le n$). Let $L=l_1l_2...l_i...l_n$ is the link between V and U. Let A is a set of subcubes including the link L ($L\in A$). Then, the nodes V and U do not exist in the set $S=A\#L$.

Proof: From the fact of neighborhood of nodes V and U along the coordinate i, it is clear, that the link between them will have a form $L=v_1v_2...*...v_n$ or $L=u_1u_2...*...u_n$, where $v_1=u_1;\ v_2=u_2;...;v_i=u_i;...;v_n=u_n$ ($1\le i\le n$). According to the definition of the # - subtraction, if we execute the subtracting from the $a_i\in A$ a don't care symbol, we obtain the same a_i, i.e. $a_i\#*=a_i$. Since the bits in the link L, different from the ith bit, will be subtracted from the set A ($S=A\#L$) and in result we will have the set S, where the values of the link L are absent, hence the nodes V and U will also not exist.

Lemma 2. Let the CMP of the node $V=v_1v_2...v_n$ is not faulty and one or more (but not all) links $L_V=\{\, l_1^k l_2^k \ldots l_j^k \ldots l_n^k \,\}$ (k is the number of faulty links $1\le k<n$) incident to the node V are faulty. Let A be the set of other cubes including L_V ($L_V\in A$). Then by operation $A\#L_V$ one obtains a set of subcubes, where the entire links incident to node V are absent.

Proof is obvious from the Lemma 1.

Lemma 3. (Generalized) Let $L^V=\{*l_2^V l_3^V ...l_n^V,\ l_1^V *l_3^V ...l_n^V,\ ...,\ l_1^V l_2^V ...l_{n-1}^V *\}$ is a set of links incident to the node V. Assume that $1=k<n$ links from this set are faulty. Let L_f^V is a faulty set of links incident to the node V ($L_f^V\subset L^V$). Then after the operation $S=A\#L_f^V$ in the set S it will absent:

(1) All the links in the set L^V;

(2) The nodes incident to the faulty links from the set L_f^V;

(3) All the links incident to the nodes incident to the faulty links.

From the Lemma 3 it is shown that, after executing the subtract operation $S=A\#L_f^V$ in the set S, we will not have some nodes and links although they are non-faulty. Thus, it becomes clear that formally, in order to include these non-faulty components in the set S, one have to add these components to this set. We have two types of these components: nodes and links. It is clear if we will add the link $L^{U,V}$ between the nodes U and V to the set S, then in this set, the nodes U and V will also exist. Thus we have to add to the set S the non-faulty links as components in order to obtain the extension of the set of healthy subcubes.

Theorem 3. Let $L^{U,V}$ is the faulty link between the nodes U and V. Let A is the set of subcubes including $L^{U,V}$. Then in order to obtain the partly extended healthy set P_E we must add to the set $S=A\ \#\ L^{U,V}$ the links incident to the nodes U and V, with the exception the faulty link $L^{U,V}$.

Proof: According to the Lemma 1 (or the Lemma 3), when we will execute the subtract operation $S=A\ \#\ L^{U,V}$ in order to obtain the fault-free set, in this set the nodes U and V will be absent, though they are not faulty. In this manner, according to the

Lemma 2 (or the Lemma 3) the links incident to the nodes U and V will absent in the set S, though they are also not faulty, except the link $L^{U,V}$. On the other hand, if we will add these fault-free links incident to the nodes U and V (except the link $L^{U,V}$) we will obtain all the fault-free links and nodes in the set S. Thus we can execute

$P_E = S \cup (L^U, L^V / L^{U,V})$,

where L^U, L^V are the links incident to the nodes U and V, respectively, $L^{U,V}$ is the link between the nodes U and V. The symbol / shows exception (subtraction) the link $L^{U,V}$ from the set of links incident to the nodes U and V.

In this way, we can obtain a partly extended set of healthy (fault-free) subcubes P_E. But in this set we will have subcubes, which can unite and thus, organize the subcubes with more dimensions. In other words, we have to find the maximal cubes in the set P_E. This problem arises in the processes of subcubes and tasks deallocation. But here we have one difference from these problems. In the subcubes and tasks deallocation problems the subcubes (or tasks) release accidentally and may be not achieve a subcube with more dimensions. In our partly extension set P_E, since we first subtract the faulty links and then add the healthy links having incidentity with other healthy subcubes and with each other, we always may be confident in that we can find the subcubes with more dimensions, then a link. From here, Theorem 4 follows.

Theorem 4. If we have only link faults and if the number of faulty links is less then n for $n \geq 3$, in the set P_E there always exists the subcube (or subcubes) with more dimensions than a link.

Now we have the problem to find the subcubes with more dimensions in the set E. By means of the coordinate (star) product operation, unification of two m-cubes A and B in the $(m+1)$-cube C may be defined. For this, the cubes A and B must have adjacent faces. The star product is the coface of these adjacent faces; it is the largest cube between two cubes [13]. The following examples illustrate it:

$01* \star 00* = 0**$; $0** \star 1** = ***$.

From Theorem 4, we can see that in the set E, we must have the cubes having adjacent faces. Therefore, using the star product operation we always can obtain a largest cube, covering both parent cubes.

Thus, from point of view of reliable data communication and subcube and task allocation processes in a faulty hypercube, the set E will be much more efficient than the set S, since the set E have more possibilities to manipulate the subcubes with more dimensions and with more quantity. The processes routing, broadcasting are in need of a set of adjacent fault-free subcubes, in order to make lightly passage from one subcube to others. Note that in subcube allocation problem, the set of disjoint fault-free subcubes is required, since there must be no interference of the messages on the communicating links. That is, the tasks running on the subcubes do not interfere with each other. The problem of partitioning a cube to the disjoint subcubes was examined in [15, 6].

In order to achieve the removal of the set of non-faulty links, we must take the links incident to faulty links. For this, in binary representation of a link (1-cube) we change the first bit (from right to left) by the symbol *, if it is not the symbol *. Since any link has one symbol *, instead of this symbol we first put 0 and then 1. The rest of bits do not change. Thus we obtain the first incident links. We continue this procedure (Fig. 3) with the second bit in 1-cube. Repeating this process with all the bits of 1-cube

we obtain the set of links, incident to the faulty link. But among these links there may be a link (links), which is (are) faulty. Therefore, we must check this set with the set of faulty links. If there are coincidences, these links will be removed out from the obtained set and thus we will achieve the set of healthy (fault-free) links, say L_{ff}. The procedure "adding of the non-faulty links" to the set S and thus we will obtain the intermediate set E.

Procedure *Finding_non-faulty_links* ($L_f^{U,V}$, L_f^U, L_f^V)

/* $L_f^{U,V}$ is current faulty link between the nodes U and V, L_f^U and L_f^V are faulty links incident to the nodes U and V, respectively */

```
begin
    for k=1 to n do
        input L_f^U and L_f^V
        L_nf1^k := L_f^{U,V} , L_nf0^k := L_f^{U,V}
    endfor
    for k=1 to n do
        for i=1 to n do
            if L_f^{U,V}[i] = * then
            L_nf1^k [i]:=1, L_nf0^k [i]:=0
            m:=i
        endfor
    endfor
    for i=1 to n do
        if i≠m then
        L_nf1^i [i]:=* , L_nf0^i [i:=*
        else  L_nf1^i :=Ø, L_nf0^i :=Ø
    endfor
    for i=1 to n do
        if  L_nf1^i =L_f^U or L_nf1^i =L_f^V
        then L_nf1^i := Ø
        if L_nf0^i =L_f^U or L_nf0^i =L_f^V
        then L_nf0^i := Ø
    endfor
end
```

Fig. 3. Procedure Adding_non-faulty_links

The full subcube recognition ability is one of the main issues of the hypercube processes. The procedure to increase the sizes of the available subcubes in the set E is presented in Fig. 4.

Procedure *Increase_size_subcubes* (S, L_{nf})

/*S is the set of healthy subcubes after subtracting the faulty links from the given cube, L_{nf} is the set of non-faulty links, defining after the procedure Finding_non-faulty_links*/

```
begin
    for i=1 to n-1 do
      E^i = S ☆ L_nf
      Define in E^i
      (1) 2- and more subcubes (let E^M)
      (2) These of 1- and 0-cubes, which are not included in the subcubes in
          E^M (let E^L)
      E^N:=E^M∪E^L
      L_nf:=E^N
    endfor
end
```

Fig. 4. Procedure for increasing size of subcubes

Example: Let in **** cube (n=4) the links F={000*, 0*01,111*,1*10} are faulty (Fig. 5, a).

(1) To reach the set of healthy subcubes we subtract from **** the set of these faulty links F.

S=****#F=****#{000*,0*01,11*1,1*10}={1*0*,0*1*,10*1,*011,01*0,*100} (see Fig. 5, b).

(2) We define the set of healthy links, incident to faulty links, which were subtracted from the cube **** together with the faulty links. These links are shown in Fig. 5,c by cut lines.

L_{nf}={00*0,0*00,*000,00*1,*001,010*,01*1,*101,1*11,11*1,*111,11*0,*110,10*0,*010,101*.

(3) Furthermore, to define the subcubes with more dimensions, first we execute the star product operation between S and L_{nf}:

E^1=S ☆L_{nf}={1*0*,0*1*,10*1,*011,01*0,*100} ☆{00*0,0*00,*000,00*1,*001,010*,01*1,*101,1*11,11*1,*111,11*0,*110,10*0,*010,101*}⇒{**00,*10*,1**1,10**,0**0,01**,**11,*01*,*0*1,*1*0}. Here we remove the 0- and 1-cubes from this set.

E^2=S ☆ E^1⇒{0**0,01**,**11,*01*,**00,*10*,1**1,10**,*0*1,*1*0}. The 0- and 1-cubes were also discarded from this set. We can see, that the subcubes in E^2 are the same in E^1. Therefore, we need not calculate E^3=S ☆ E^2. From the set S we take the subcubes with more dimension (S^M ={1*0* ,0*1*}) and add these subcubes to the set E^2. Thus, extended set of non-faulty subcubes will be E= S^M∪ E^2 (See Fig.5, d).

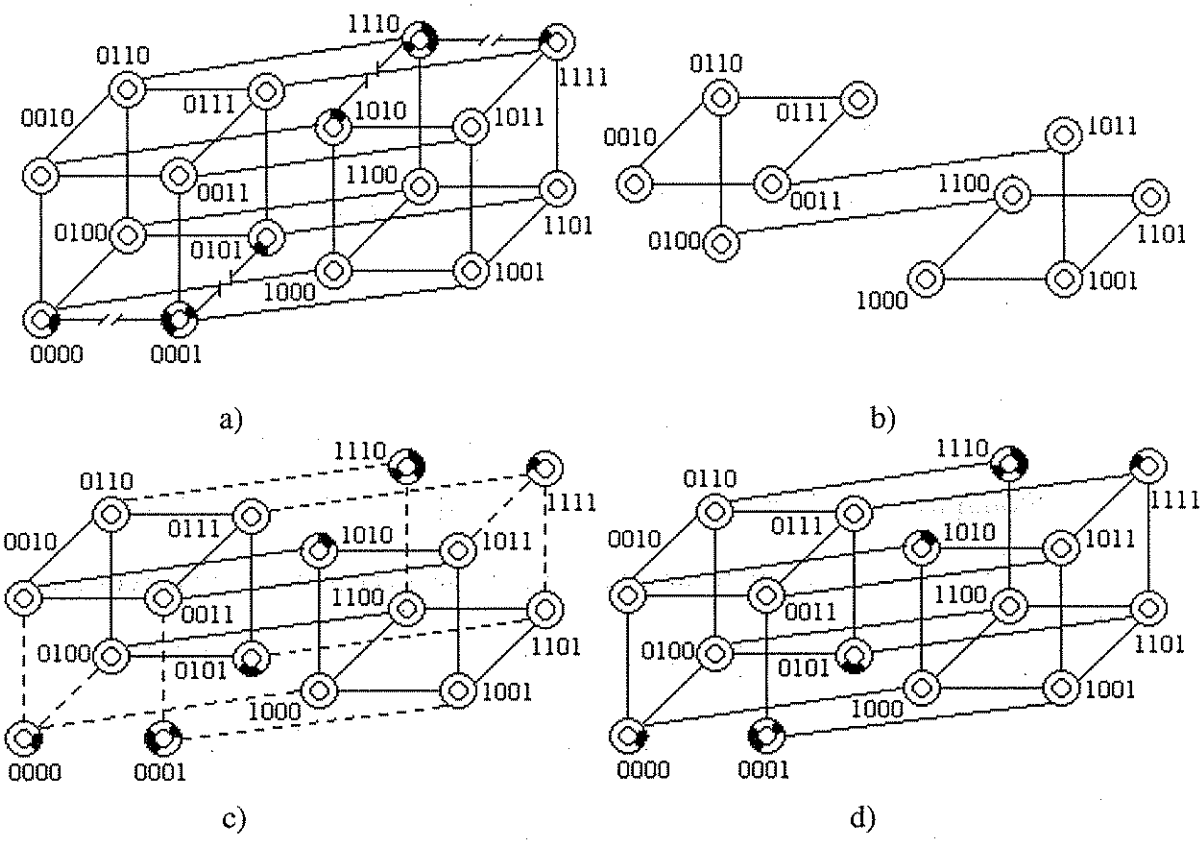

Fig. 5. Example for subtracting only links

In practice, a hypercube can have node and link faults simultaneously. In this case, it is necessary to pick out the node faults and the link faults and then to process them separately.

5. CONCLUSION

The reliability of a hypercube multiprocessor becomes a critical issue when the size of the system grows. In many methods, to obtain a healthy set of subcubes in a faulty hypercube, the faulty node and the incident links (or the faulty links and the incident nodes) are discarded logically from the other set of subcubes, including these nodes (or links) although these links (or nodes) may not be faulty. With this, the set of healthy subcubes is narrowed which prevents the manipulation with completely healthy elements existing in the hypercube. Since links incident to faulty nodes (or nodes incident faulty links) may be healthy, they must be included in the healthy set of components of the hypercube.

In this paper, we have proposed a method to extend a list of healthy subcubes in a faulty hypercube. Consequently, one can have more possibilities to manipulate in the hypercube processes such as data communication (definition of short path between source and target nodes in routing, broadcasting, multicasting, etc.), subcube allocation (constructing the subcubes with more dimension), etc. It allows, apparently, increasing the reliability of system and possibility to execute tasks with more dimension in hypercube.

Furthermore, large number of healthy subcubes can allow us to find a convenient path between a source node and destination node(s) using neighborhood of subcubes, since after an extension, we have more subcubes than original healthy subcubes set, we have highest possibility of having a neighborhood among these subcubes. Here we can use a coordinate product operation for the definition of a neighborhood of fault free subcubes in a faulty hypercube with a view to define a path between two subcubes. Such a procedure is developed in [1].

The developed procedures were implemented for the analysis of different routing algorithms in a faulty hypercube. The Hypercube Routing Simulator described in [19] and other simulations [37] were used for this purpose. Results of the simulations showed that except the case where source and target nodes are placed in isolated subcubes, these procedures found the message passing paths in all cases whereas without using these procedures, these paths may be non-minimal or could not be obtained. For example, let the nodes 0010 and 1000 are the source and target nodes respectively (Fig. 5,a). Then after obtaining a set of healthy subcubes by using coordinate subtraction operation we will have a non-minimal path between the nodes 0010 and 1000, which will be 0010→0110→0100→1100→1000 (see Fig. 5,b) and will have four steps. But if we use the extended set, then we will have minimal path 0010→1010→1000 (See Fig. 5, d), which will have only two steps. Another example, when after coordinate subtraction operation a path can not be found is as follows. Let the nodes 0000 and 1111 are the source and target nodes respectively. Then according to the non-extended set here there is not any path between these nodes (See also Fig. 5, b), but if we apply the enlarged set of healthy subcubes then the path between these nodes could be found as 0000→1000→1001→1101→1111 (see also Fig. 5, d). Thus the simulation showed that when we use the extended set for routing algorithms; we can find in some cases the shortest path between the nodes although we can not find these paths with the non-extended set of the healthy subcubes. In some cases, no paths can be found with the non-extended set whereas the extended set yields paths.

In many fault tolerance methods in hypercube, the proposed approaches depend on the number of faulty components, which is limited by the dimension of hypercube n or near n. Besides, these approaches also depend on the locations of faulty elements in hypercube. In our approach this lack is removed and the procedure does not depend on scattering of the faulty elements in hypercube. Thus, by aid of the proposed algebraic transformations one can receive a set of extended fault-free subcubes, that is, a beginning set for further manipulation in the hypercube. The existence of such fault-free subcubes can be also used to obtain efficient implementations for a wide range of hypercube algorithms.

REFERENCES

[1] N.M.Allahverdi, Procedure For Definition of Neighborhood in Faulty Hypercube Multiprocessor, *Mathematical and Computational Applications* **4**, 17-23, 1999.

[2] N.M Allahverdi, K.Erciyeş, Extension of Non-faulty Subcubes Set in A Faulty Hypercube Multiprocessor, *Proc. of 4th Symp. Of High Performance Computing and Interconnection Networks under 4th Intern. Joint Conference on Computer Science and Informatics*, Oct. 23-28, 1998, Durham, N.C., USA, **4**, 115-118.

[3] Allahverdi, N.M., Kahramanli, S.S. and Erciyeş K. A Fault Tolerant Routing Algorithm in Hypercube with Application Cube Algebra, *Journal of Systems Architecture* **46**, 201-205, 2000.

[4] N.M.Allahverdiyev, S.S.Safaraliyeva, Choice of Multiprocessor System Configuration for Digital Signal Processing, *USSR Report; Cybernetics, Computers and Automation Technology, Foreign Broadcast Information Service*, JPRS, 83843, USA, July 7, 1983.

[5] B., Becker, H.-U. Simon, How Robist Is the *n*-Cube? *Information and Computation* **77**, 162-178, 1988.

[6] J.Bruck, R. Cypher, D. Soroker, Tolerating Faults in Hypercubes Using Subcube Partitioning, *IEEE Trans.Comput.*.**41**, 599-605, 1992.

[7] Y. Chang, L. Bhuyan, Subcube Fault Tolerance in Hypercube Multiprocessors, *IEEE Trans.Comput.* **44**, 1108-1120, 1995.

[8] M.-S. Chen, K.G. Shin, Subcube Allocation and Task Migration, *IEEE Trans.Comput.* **39**, 1146-1155, 1990.

[9] M.-S. Chen, K.G. Shin, Depth-first search approaches for fault-tolerant routing in hypercube multicomputers. *IEEE Trans.on Paral.Distrib.Syst.* **1**, 152-159, 1990.

[10] G.M. Chiu, K.S.Chen, Use of Routing Capability for Fault-Tolerant Routing in Hypercube Multicomputers, *IEE Trans.on Comp.* **46**, 953-958, 1997.

[11] G.M. Chiu, S.P. Wu, A Fault-Tolerant Routing Strategy in Hypercube Multicomputers, *IEE Trans.on Comp.* **45**, 143-155, 1996.

[12] M.R. Dagenais, V.K. Agarwal, N.C. Rumin, McBOOLE: A new procedure for exact logic minimization. *IEEE Trans.Comput.-Aided Design*, **CAD-5**, 229-237, 1986.

[13] D.L. Dietmeyer, *Logic Design of Digital Systems*, Boston, MA: Allyn and Bacon, 1979.

[14] T.H. Dunigan, Performance of the Intel iPSC/860 and Ncube 6400 Hypercubes, *Parallel Computing*, **17**, 1285-1302, 1991.

[15] S. Dutt, J.P. Hayes, Subcube Allocation in Hypercube Computers, *IEEE Trans.Comput.***40**, 341-352, 1991.

[16] A.-H. Esfahanian, Generalized Measures of Fault-Tolerance with Application to N-Cube Networks. *IEEE Trans.Comput.* **38**, 1586-1591, 1989.

[17] P.Fraignaud, Asymptotically Optimal Broadcasting and Gossiping in Faulty Hypercube Multicomputers, *IEEE Trans. on Comput.* **41**, 1410-1419, 1992.

[18] N. Graham, F. Harary, M.Livingston, Q.F. Stout, Subcube Fault-Tolerance in Hypercubes, *Inform. and Comput.* **102**, 280-314, 1993.

[19] S.Güneş, *Analysis and Simulation of Fault Tolerance in Hypercubes Parallel Processing System,* Ph.D. Thesis, Selçuk University, Konya, Turkey, 2000.

[20] F.T. Hady, B.L. Menezes, The Performance of Crossbar-Based Binary Hypercubes, *IEEE Trans. on Comput.* **44**, 1208-1215, 1995.

[21] Intel, *iPSC User's Guide*, Intel 17455-03, Portland, OR, Oct.,1985.

[22] S. S. Kahramanli, N. M. Allahverdi, Compact method of minimization of Boolean functions with multiple variables. *Proc. Intern. Symp. of Application of Computers*, June 1993, Konya, Turkey, 433-440.

[23] S.S. Kahramanli, N.M. Allahverdi, Algebraic Approach to Transformations on Hypercube System, *Mathematical and Computational Applications* **1**, 50-59, 1996.
[24] J. Kim, R.D. Chita, Hypercube Communication Delay with Wormhole Routing, *IEEE Trans. on Comput.* **43**, 806-814, 1994.
[25] Y. Lan, An Adaptive Fault-Tolerant Routing Algorithm for Hypercube Multicomputers, *IEEE Trans.on Paral.and Distrib.Systems*, **6**, 1147-1152, 1995.
[26] T. C. Lee, J. P. Hayes, A fault-tolerant communication scheme for hypercube computers. *IEEE Trans. on Comput.* **41**, 1242-1256, 1992.
[27] Y.R.Leu, S.Y.Kuo, Fault-Tolerant Tree Communication Scheme for Hypercube Systems, *IEE Trans.on Comp.* **45**, 641-650, 1996.
[28] R.E. Miller, *Switching Theory, Vol.1, Combination Circuits,* New York; John Wiley and Sons, 1965.
[29] E. M. Nadjafov, S. S. Kahramanov, On the synthesis of multiple output switching scheme. *Scientific Notes of Azerbaijan Institute of Petroleum and Chemistry,* Vol. **IX**, 65-69, 1973.
[30] Ncube, *Ncube 6400 Processor Manual*, Ncube V1.0, Beaverton, OR, 1990.
[31] G.D. Pifarré, L. Gravano, G. Denicolay, J.L.C. Sanz, Adaptive Deadlock- and Livelock-Free Routing in the Hypercube Network, *IEEE Trans.on Paral.and Distrib.Systems*, **5**, 1121-1138, 1994.
[32] C.S., Raghavendra, P.-J. Yang, S.-B. Tien, Free Dimensions - An Effective Approach to Achieving Fault tolerance in Hypercube, *IEEE Trans.on Comput.*, **44**, 1152-1156, 1995.
[33] S. Rai, J.L. Trahan, T. Smailis, Processor Allocation in Hypercube Multiprocessors, *IEEE Trans. on Paral. Distrib. Syst.* **6**, 606-616, 1995.
[34] A. Sengupta, S. Bandyopadhyay, Deadlock-Routing in *K*-are Hypercube Network in Presence of Processor Failures, *Information Processing Letters* **34**, 323-328, 1990.
[35] J. Wu, An Optimal Fault-tolerant Nonredundant Broadcasting Scheme in Injured Hypercubes. *J.Paral. and Distrib.Computers*, **22**, 1995.
[36] J.Wu, Adaptive Fault-Tolerant Routing in Cube-Based Multicomputers Using Safety Vectors, *IEEE Trans: Paral.and Distrib.Systems* **9**, 321-334, 1998.
[37] A.B. Şahin, Fault-Tolernat Routing and Routing Algorithms, M.S. Thesis, International Computer Institute, Ege University, Izmir, Turkey, 2001.