Distributed Identification of All Maximal Incomplete Subcubes in a Faulty Hypercube

Hsing-Lung Chen

Department of Electronic Engineering
National Taiwan Institute of Technology
Taipei, Taiwan, R. O. C.

Nian-Feng Tzeng

Center for Advanced Computer Studies
University of Southwestern Louisiana
Lafayette, LA 70504

Abstract — Reconfiguring a faulty hypercube into a maximal incomplete cube tends to lower potential performance degradation, because a hypercube so reconfigured often results in a much larger system than what is attained by any conventional reconfiguration scheme which identifies only complete subcubes. This paper proposes an efficient strategy for identifying all the maximal incomplete subcubes present in a faulty hypercube. The proposed strategy is distributed in that every healthy node executes the same identification algorithm independently at the same time.

1. Introduction

All prior schemes proposed for reconfiguring faulty hypercubes assume that only complete subcubes are permitted and attempt to find the complete healthy subcubes with the maximal dimension in a faulty hypercube [1-3]. They often fail to maintain as many workable nodes as desired after reconfiguration due to the strong restriction on allowable system sizes. For example, a reconfigured hypercube in the presence of just one fault contains merely a half of its original nodes, discarding many healthy nodes simply to meet the system size constraint and possibly resulting in a severe performance loss.

This paper deals with a strategy for identifying maximal incomplete subcubes in a faulty hypercube, retaining as many healthy nodes as possible in order to keep performance degradation minimal. Unlike a complete one, an incomplete cube can be of any arbitrary size [4]. Our proposed identification strategy uncovers all maximal incomplete subcubes efficiently. A maximal incomplete subcube usually involves one maximal complete subcube plus certain smaller sized subcubes, making it possible to finish a given batch of jobs faster than its complete counterpart alone by supporting simultaneous execution of multiple jobs of different sizes. Moreover, an incomplete subcube may carry out a given job faster by assigning more nodes to execute the job cooperatively.

2. Preliminaries

Let $H_n$ denote an $n$-dimensional hypercube. Each node in $H_n$ is labeled by an $n$-bit string. A $k$-dimensional subcube in $H_n$ can be addressed uniquely by a string of $n$ symbols over set $(0, 1, *)$, where $*$ is a don't care symbol, such that there are exactly $k$ *'s in the string. An $n$-dimensional incomplete hypercube with $M$ nodes, $2^{n-1} \leq M < 2^n$, represented by $I^M_n$, is defined recursively as follows: $I^M_n$ comprises two components, $H_{n-1}$ and $J^M_{n-1} (k \leq n-1)$, with a link existing between a node $A$ in $H_{n-1}$ and another node $B$ in $I^M_{n-1}$ if and only if the addresses of nodes $A$ and $B$ differ in exactly one bit. $I^M_n$ comprises a set of complete cubes of dimensions $n-1$ and below, and no two constituent cubes are of the same size. Let the binary representation of $M$ be $<1, x_{n-2} x_{n-3} \cdots x_i \cdots x_0>$, then it is clear that $I^M_n$ contains, in addition to $H_{n-1}$, $H_i$'s for all $x_i = 1, 0 \leq i \leq n-2$. $H_i$ is a constituent cube of $I^M_n$ if and only if bit $x_i$ in the binary representation of $M$ equals 1.

In a faulty complete hypercube, one or multiple fault-free incomplete subcubes exist. Each such fault-free incomplete subcube is contained in exactly one smallest complete subcube, called the minimum cover subcube. For fault-free incomplete subcube $I^T_n$, $T > 2^{n-1}$, its minimum cover subcube is of dimension $s$, i.e., $H_s$, and nodes which are in the minimum cover subcube but are outside $I^T_n$ form a subcube (either a complete or an incomplete one), called the conjugate subcube. $I^T_n$ and its conjugate subcube together form a complete subcube of dimension $s$, and the conjugate subcube involves at least one fault (as otherwise, a fault-free $H_s$ exists). Every fault-free incomplete subcube $I^T_n$, $T > 2^{n-1}$, has one and only one conjugate subcube, denoted by $CS^{s-T}$.

The operations on subcubes in a hypercube can be performed elegantly, following a way similar to boolean algebra, if a subcube is represented as a min-term, i.e., product of boolean variables, obtained from the address of the subcube by replacing bit position $i$ with $b_i$ (or $\bar{b}_i$), if position $i$ is 1 (or 0), and then dropping all *'s. For example, subcube $00**1$ is represented by $\bar{b}_1 b_0$. The union of the three subcubes: $00**1$, $01**0$, and $00**0$, in $H_5$ is given by $\bar{b}_1 b_0 + \bar{b}_2 b_0 + \bar{b}_3 b_0 = b_3 b_0 + \bar{b}_2 b_0 = I_3$, which is $00^5$. As it becomes clear later, the use of a

H.-L. Chen was supported by the National Science Council of the Republic of China under Contract NSC81-0408-E011-511. N.-F. Tzeng was supported in part by the NSF under Grant MIP-9201308 and CCR-9300075 and by the State of Louisiana under Contract LEQSF(1992-94)-RD-A-32.
boolean expression to specify the union of subcubes greatly facilitates our identification procedure. Note that a
null expression denotes the whole hypercube.

From the expression for a union of subcubes in a
hypercube, one can get the addresses of all cube nodes
outside the union of subcubes directly, with the aid of
DeMorgan's theorem. Consider the union of three
subcubes in \( H_3: b_4 \oplus b_5 \oplus b_7 b_1 = W \), which in fact repre-
sents \( I^3_{2^3} \). The collection of all the nodes outside \( W \)
is expressed by \( \bar{W} = (b_4)(b_5)(b_7 + b_2 + b_1) \), which is simpli-
ified to \( b_2 b_3 (b_7 + b_1) \), designating an incomplete subcube
of size 6 (i.e., the conjugate subcube of \( I^3_{2^3} \), as expected).

For a faulty hypercube \( H_n \) and a given node \( A \) in
\( H_n \), it is possible to identify systematically every fault-
free subcube which involves the given node \( A \). In other
words, we can arrive at the expression characterizing the
set of \( P = \{ S_j \mid S_j \) is a fault-free subcube in \( H_n \) and \( S_j 
\) involves node \( A \} \) quickly and systematically, by determining "regions" which never contribute to any fault-free sub-
cube containing the given node \( A \) [3]. Each fault results
in one such a region, called a reject region, which is the
smallest subcube involving both the fault and the antipo-
dal node of \( A \), and its address is given by performing
operation \( \oplus \) on the labels of the faulty node and the
antipodal node of \( A \), and its address is given by performing
operation \( \oplus \) on the labels of the faulty node and the
antipodal node of \( A \). Assume that \( R \) denotes the collection of all reject regions in
\( H_n \), one corresponding to a fault. It has been shown
[3], by taking advantage of interesting properties of faulty
hypercubes, that all the cubes nodes in \( H_n \) can be partiti-
tioned into two disjoint sets with respect to node \( A: R \) and
\( P \). As a result, the fact that knowing \( R \) can directly get \( P 
\) holds valid for any faulty hypercube, and it is fundamen-
tally important to our identification process.

From the labels of all faulty nodes and a given node,
we may quickly obtain the addresses of all reject regions
(simply by performing the \( \oplus \) operations). A reject region
which lies entirely inside another reject region is then
removed to avoid redundancy. Each reject region left after
removing redundancy is represented by a minterm.
Expression \( R \) is the summation of all minterms, or equiva-
ently, is given in the form of sum-of-product. From \( R \),
we immediately arrive at \( P = \bar{R} \), which is in the form of
product-of-sum (from DeMorgan's theorem). Since \( P 
\) involves all the fault-free subcubes containing the given
node, we can identify the largest incomplete subcube from
\( P \), based on certain pertinent features described below.

3. Pertinent Features of Incomplete Subcubes

Consider an \( n \)-dimensional incomplete subcube with
size \( M \) in \( H_n \), \( I^M_{n} \). Let \( \Omega(M, j) \) be the value obtained by
resetting all the bits in the binary representation of \( M 
\) except for the \( j \)-th run of "1", with the first run of "1"
starting from the most significant bit. It is clear that \( I^M_{n} \)
consists of incomplete subcubes \( I^{\Omega(M, 1)}_{n}, I^{\Omega(M, 2)}_{n}, \ldots, I^{\Omega(M, m)}_{n} \),
and so on. An interesting property of \( I^{\Omega(M, 1)}_{n} \) is revealed
by the next theorem, whose proof can be found in [5].

**Theorem 1:** Incomplete subcube \( I^{\Omega(M, 1)}_{n} \) of dimension \( s 
\) in \( H_n \) involves exactly \( \eta \) \((s - 1)\)-dimensional complete
subcubes, which together constitute \( I^{\Omega(M, 1)}_{n} \), where \( \eta 
\) is the number of nonzero bits in the binary representation of
\( \Omega(M, 1) \).

Theorem 1 indicates that \( I^{\Omega(M, 1)}_{n} \) can be uniquely
expressed by the collection of the \( \eta \) \((s - 1)\)-dimensional subcubes \( y_1, y_2, \ldots, y_\eta \in e_r \), for all \( 1 \leq r \leq \eta \), i.e.,
y_1, y_2, \ldots, y_\eta (z_1 + z_2 + \ldots + z_s + \ldots + z_r) \). Note that these \( \eta 
\) complete subcubes are overlapped. Let \( \Pi \) and \( \Sigma \) denote
the product and the summation of boolean variables, then
the preceding expression becomes \( \Pi \eta \text{ } y_j (\Sigma \eta \text{ } z_j) \).

Let the minimum cover subcube of \( I^{\Omega(M, 1)}_{n} \) and the
conjugate subcube of \( I^{\Omega(M, 1)}_{n} \) be denoted by \( MC \) and
\( CS(1) \), respectively. Based on its recursive construction
nature, \( I^{MC}_{n} \) is composed of \( I^{\Omega(M, 1)}_{n} \) and \( I^{\Omega(M, 2)}_{n} \),
which denotes the incomplete subcube comprising \( I^{\Omega(M, \delta)}_{n} \) \( \delta = 2, 3, \ldots \). The minterm representation of any node in
incomplete subcube \( I^{\Omega(M, 2)}_{n} \) must involve boolean vari-
ables \( y_1, y_2, \ldots, y_\eta \), and \( z_1, z_2, \ldots, z_s \) (for otherwise, it
would be in \( I^{\Omega(M, 1)}_{n} \), whose minterm involves the factor of
\( \Sigma \eta \text{ } z_j \)). In other words, the minterm representation
of \( I^{\Omega(M, 2)}_{n} \) contains variables \( y_1, y_2, \ldots, y_\eta, z_1, z_2, \ldots, z_s \),
and \( z_r \) (perhaps plus some other boolean variables). Since the
minterm representation of \( CS(1) \) is \( y_1, y_2, \ldots, y_\eta, z_1, z_2, \ldots, z_s \),
\( I^{\Omega(M, 2)}_{n} \) must fall completely in \( CS(1) \).

In general, we may obtain the result with respect to
incomplete subcube \( I^{\Omega(M, \ell)}_{n} \) for any \( \ell \geq 2 \), following a
similar argument as above: \( I^{\Omega(M, \ell)}_{n} \) is contained entirely
in \( CS(\ell - 1) \), where \( CS(\ell - 1) \) denotes the conjugate sub-
cube of incomplete subcube \( z_\ell = (I^{MC}_{\ell - 1}) \cup I^{\Omega(M, 2)}_{\ell - 1} \cup \ldots \cup I^{\Omega(M, \ell - 1)}_{\ell - 1} \). This general result ensures that a specific
incomplete subcube in \( CS(\ell - 1) \) together with \( z_\ell \) forms
\( I^{MC}_{\ell - 1} \), and it provides the basis of our expression for
\( I^{MC}_{\ell - 1} \). Let \( I^{\Omega(M, 1)}_{n} \) and \( I^{\Omega(M, 2)}_{n} \) be expressed respectively
by \( y_1, y_2, \ldots, y_\eta, z_1, z_2, \ldots, z_s \), and \( y_1, y_2, \ldots, y_\eta, z_1, z_2, \ldots, z_s \), then the expression
for \( I^{\Omega(M, 2)}_{n} \) is given by \( y_1, y_2, \ldots, y_\eta, z_1, z_2, \ldots, z_s \).

From the
rule of \( E + EG = E + G \), the preceding expression equals
\( y_1, y_2, \ldots, y_\eta, z_1, z_2, \ldots, z_s \), and \( \Pi \eta \text{ } z_j \) are
complement with each other. Incomplete subcube \( I^{\Omega(M, 2)}_{n} \) in
CS(1) can be identified in a similar way as finding \( f^{(M, 1)} \) in \( MC \), and the remaining part \( f^{(M, z, 3)} \) is contained totally in CS(2), yielding \( \Phi_1 = \prod_{i=1}^{m} y_i = \prod_{j=2}^{n} z_{j,2} + \Phi_2 \), where \( \alpha_2 \) is the number of zero bits between the first run of "1" and the second run of "1" in the binary representation of \( M \), \( \eta_j \) is the number of nonzero bits in \( \Omega(M, 2) \), \( y_{j,2} \) are boolean variables \( b_i \)'s or \( \bar{b}_i \)'s, and \( \Phi_2 \) is determined by \( f^{(M, z, 3)} \) inside CS(2).

This process repeats until \( \Omega(M, i+1) = 0 \), for a certain \( i \), and at that time all the runs of "1" in \( M \) have been exhausted, resulting in the target incomplete subcube \( I^{(M)}_P \).

The general expression for \( I^{(M)}_P \) is thus

\[
I^{(M)}_P = \left( \prod_{i=1}^{n} y_{i,1} \right) \left( \sum_{j=2}^{n} y_{j,2} \right) \left( \sum_{j=3}^{n} y_{j,2} \right) \cdots \left( \sum_{j=m}^{n} y_{j,2} \right).
\]  

(1)

4. Incomplete Subcubes Involving a Given Node

A proper incomplete subcube in a faulty hypercube refers to a fault-free incomplete subcube which is not contained entirely in any other fault-free subcube. This section deals with identifying all the proper incomplete subcubes from \( P \), which equals \( R \), because non-proper incomplete subcubes cannot be the largest ones. While a complete subcube can be represented by a minterm, an incomplete subcube in general can be expressed by the form given in Eq. (1), called a comterm (standing for a compound term, denoted by \( C \)). It is easy to see from Eq. (1) that a comterm is reduced to a minterm if \( \alpha_i = 0 \), for all \( i \geq 2 \), and \( \eta_j = 0 \), for all \( j \geq 1 \). A comterm is thus an extended expression for subcubes, whether complete or incomplete ones.

In order to identify all fault-free subcubes, we convert \( P \) (in the product-of-sum form) to its sum-of-minterms equivalent, which signifies the collection of all the proper incomplete subcubes. During the conversion process, the distributive law is applied to minterms (sumterms) repeatedly until all expressions obtained are in the form of comterms given by Eq. (1). Applying the distributive law to the first minterm of \( x(y+z)(a+b) \) results in \( xy(a+b) + xz(a+b) \), where terms \( xy(a+b) \) and \( xz(a+b) \) are called the generic terms (i.e., generalized terms).

Conversion Process

The conversion process involves two steps: (1) determining an appropriate minterm from \( P \) and applying the distributive law to the determined minterm after it is added to \( P \), and (2) extracting the largest common component present in all the minterms. For a given \( P \), the appropriate minterm to be added contains all the variables which appear in one or multiple minterms of \( P \). The reason for step (1) to be involved is because comterms, unlike minterms or maxterms, do not possess the commutative property, and all the distinct comterms specified by \( P \) have to be derived so that all the proper incomplete subcubes referred to by \( P \) are explored. The purpose of step (2) is to guarantee producing comterms which denote the largest incomplete subcubes possible. For any given \( P \), steps (1) and (2) apply repeatedly till every expression obtained is a comterm, and each expression then refers to one incomplete subcube.

Consider a given node 000110 in \( H_6 \) with four faults: 010011, 101111, 100101, and 110110. The antipodal node of 000110 is 111001, and the four reject regions are 010011 \( \oplus 111001 \), 101111 \( \oplus 111001 \), 100101 \( \oplus 111001 \), and 110110 \( \oplus 111001 \) giving rise to \( R = b_4b_2b_5 + b_4b_2b_6 + b_3b_5b_7 + b_2b_7b_4 \). Expression \( P \) for the given node is

\[
P = (b_4 + b_2 + b_5)(b_3 + b_2 + b_6)(b_4 + b_2 + b_1 + b_0)(b_6 + b_5).
\]  

(2)

This expression is converted into its sum-of-comterm equivalence by applying the above two steps repeatedly, as depicted in Fig. 1, where the result is the collection of the incomplete subcube specified by comterms given at the leaves of the tree.

Simplification Criteria

In the course of conversion shown in Fig. 1, many unnecessary comterms are produced, warranting simplification to save conversion effort. Our goal is to avoid unnecessary comterms from being created in the conversion process so that the comterms produced at the end all correspond to proper incomplete subcubes, involving much less effort. The subsequent two lemmas provide the basis of our simplification criteria to be included in the conversion process, and their proofs are given in [5].

Lemma 1: Let expressions \( E_i \) and \( E_j \) be generated from \( E_f \) due to variables \( b_i \) and \( b_j \), respectively. If all minterms in \( E_i \) involve a common component and so do all minterms in \( E_j \), then incomplete subcubes specified by \( E_i \) and \( E_j \), denoted respectively as \( IS_i \) and \( IS_j \), satisfy \( IS_i \cap IS_j = IS_i \).

This lemma indicates that all expressions derived from \( E_f \) have to be treated further, if they may follow conversion step (2) to extract common components; none of them can be discarded immediately.

Lemma 2: Let expressions \( E_i \) and \( E_j \) be generated from \( E_f \) due to variables \( b_i \) and \( b_j \), respectively. If all minterms in \( E_j \) share no common component, then the incomplete subcube specified by the expression produced subsequently from \( E_j \) due to \( b_j \) is contained entirely in the incomplete subcube specified by \( E_i \).

This lemma reveals an interesting attribute that tells which expressions to be generated are contained in prior expressions and thereby can be dropped, often resulting in significantly less conversion effort. To this end, a set of
"discarded" variables is kept for each expression generated by distribution, where the set involves all the boolean variables that have been distributed (over the same expression) so far.

From Lemmas 1 and 2, we arrive at the simplification criteria to be incorporated in our conversion process: (i) the distribution sequence for an expression, say \( E \), follows that the variable which yields an expression with a common variable present in all its maxterms is treated first (recall that if a variable appears in all maxterms, the variable is extracted); in case there are multiple choices, the one that appears in more maxterms of \( E \) is selected earlier; and (ii) the set of discarded variables is produced for each expression generated at level \( i \) and the set is employed by the expression generated to avoid unnecessary distribution at level \( i + 1 \), provided that all the maxterms in the expression share no common component.

Making use of these simplification criteria, we obtain the conversion result of expression \( P \) for the faulty \( H_k \) given by Eq. (2), as illustrated in Fig. 2, where the set of discarded variables for the \( j \)th expression generated in level \( i \) is denoted by \( \Delta_{i,j} \) (with expressions in a level numbered rightwards). If the set of discarded variables for an expression involves all maxterm variables, the expression requires no distribution, like the last expression in level 1 of Fig. 2. The conversion process with the aid of the simplification criteria produces only necessary comterms (i.e., proper incomplete subcubes, see Fig. 2) as desired, saving lots of effort, and the collection of comterms produced is identical to that depicted in Fig. 1 after those unnecessary comterms are removed.

An algorithm for generating the sum-of-comterm (SOC) equivalence of a given product-of-sum expression \( P \) is provided below, where a stack is employed to keep expressions produced during conversion and their corresponding sets of discarded variables. The stack holds only expression \( P \) initially, and both the set of discarded variables for \( P \) and SOC are initialized with \( \emptyset \). After the algorithm ends, all comterms produced are stored in SOC, which is then scanned to find the largest size.

**Algorithm A:** (generate a sum-of-comterm equivalence):

While (the stack is not empty) {
    Pop an expression \( E_i \) together with its corresponding set of discarded variables from the stack;
    Let \( G_i \) be the genterm involved in \( E_i \);
    While (all maxterms in \( G_i \) share a common variable) {
        Extract the common variable according to conversion step (2);
        Flush the set of discarded variables;
    }
    If (\( G_i \) is in the comterm form, i.e., involves no more than one maxterm)
        Append the comterm derived from \( E_i \) to SOC;

else {
    Apply the distributive law to each variable in
    maxterms of \( G_i \), excluding discarded variables;
    Determine the sets of discarded variables for
    expressions newly generated;
    Push all generated expressions and their corresponding
    sets of discarded variables into the stack. }
}

Since our conversion process explores all incomplete subcubes which are specified by expression \( P \) and which contain a given node, Algorithm A is insured to identify all proper incomplete subcubes with respect to the given node, as the simplification criteria remove only unnecessary expressions. The size of an incomplete subcube specified by a comterm can be directly obtained from Eq. (1), as follows: Let the size be represented as an \( n \)-bit string, then the bit string (starting from the leftmost bit) consists of \( \alpha_1 \) 0's, followed by \( \eta_1 \) 1's, followed by \( \alpha_2 \) 0's, followed by \( \eta_2 \) 1's, and so on, until the comterm is exhausted; if the string formed is of length less than \( n \), say \( l \), then \((n-l)\) 0's are appended to the string.

5. Distributed Identification

Algorithm A lends itself perfectly to distributed identification by being executed at every healthy node independently, with the executing node itself treated as the given node. It is assumed that the set of faulty nodes is uncovered in a distributed manner by fault-free nodes, following the diagnostic algorithm introduced in [6]. After fault diagnosis is done, the address of a faulty node is broadcast to all nodes by a healthy neighbor, and every healthy node keeps this broadcast information.

On receiving the addresses of all faulty nodes, each healthy node identifies all the largest incomplete subcubes involving the node itself using Algorithm A. For a faulty \( H_k \), there are in general \( O(2^n) \) nodes participating in the identification process, and information calculated at each participant about the largest incomplete subcubes has to be taken into account in reaching the decision on maximal incomplete subcubes globally, i.e., the size of the maximal incomplete subcubes in the whole system is determined based on the largest incomplete subcube size found at each participating node.

We assume that the hypercube system has one host, which is in charge of reconfiguration after receiving the comterms denoting maximal incomplete subcubes (called the \textit{maximal comterms} for short), and which has a direct connection to each cube node. A distributed approach to determining the size of maximal incomplete subcubes is described next, followed by an algorithm which ensures sending all maximal comterms to the host non-redundantly. Cube nodes are partitioned into \( n+1 \) levels, according to the number of nonzero bits in their labels.
Size Determination

Consider a fault-free node at level $i$, $0 < i \leq n$, in $H_n$. The node, after carrying out Algorithm A, waits to receive the size of the largest incomplete subcube(s) determined at a healthy neighbor in level $i - 1$. A node is assumed to know the status of all its neighbors and ignores messages from its faulty neighbors. Once receiving messages from all its healthy level $i - 1$ neighbors, the node chooses the largest one among all received sizes and the size it found using Algorithm A, and sends the chosen largest size to all its neighbors in level $i + 1$ (where the level $i + 1$ neighbor of node $1^n$ is the host). Node $0^n$ receives no messages and sends the largest size it found to all its neighbors immediately after finishing Algorithm A.

In case a node in level $i$ has no fault-free neighbor in level $i - 1$, it forwards the largest size it found immediately to all its neighbors in level $i + 1$. On the other hand, if a level $i$ node has no healthy neighbor in level $i + 1$, it simply sends the largest size chosen to the host. In the last step of this determination, if the host receives multiple sizes, it selects the largest one as the size of the maximal conterm(s). The maximal size is then broadcast to all healthy nodes by the host, so that each node knows whether or not it contains the maximal conterm(s). For an $n$-dimensional hypercube with $m$ faults, the total number of messages sent to the host depends on the distribution of these faults and is always less than $n \times m$, since the failure of node $1^n$ causes $n$ messages to be sent to the host and any other fault results in fewer than $n$ messages directed to the host.

Gathering Maximal Comterm(s)

All maximal comterms are forwarded to the host to facilitate reconfiguration. It is desirable that comterms are sent to the host non-redundantly, in an attempt to avoid unnecessary traffic and computation. The subsequent theorem provides the basis for non-redundant delivery of comterms to the host, and its proof is included in [5].

**Theorem 2:** All the healthy nodes which come out with an identical maximal comterm form a fault-free complete subcube.

According to Theorem 2, we let one and only one node in $S_p$ responsible for sending the maximal comterm(s) to the host, preventing any redundancy. Among all the $S_p$ nodes, there is exactly one node at the highest level and that node is designated as the responsible node. The following algorithm elects the responsible node(s) in a distributed way, on the basis of the fact stated in Theorem 2: Each node in level $i - 1$, $0 < i \leq n$, sends the largest comterm(s) it found, if any, to all its neighbors in level $i$; if no such comterm is involved, it sends out a specific string, say the empty string. A level $i$ node responds to messages from level $i - 1$ nodes individually; if a received message carries the same comterm(s) as what the node found, the node responds by sending the comterm(s) back; else, it responds with the specific (empty) string. If the receiving node in level $i$ involves no maximal comterms, it responds to every message from level $i - 1$ with the empty string. A node in level $i - 1$ waits until all responses from level $i$ arrive, and if any of the received response involves the same comterm(s) as what the node itself found, the node is not a responsible node for directing the comterm(s) to the host; otherwise, it is a responsible node. From the preceding algorithm, the single node at the highest level within subcube $S_p$ is identified as the responsible node, and every such subcube has one responsible node determined. All the maximal comterm(s) are thus sent to the host non-redundantly.

6. Conclusions

A distributed strategy for identifying all maximal incomplete subcubes present in a faulty hypercube has been introduced. Every fault-free node is required to participate in the identification process by executing the same algorithm independently at the same time. The nodes responsible for sending the addresses of maximal incomplete subcubes to the host, after their size is determined in a distributed manner, are then elected through a distributed procedure. The host is ensured to receive the addresses of all maximal incomplete subcubes non-redundantly. This distributed identification strategy appears beneficial and practical for large hypercubes operating in a gracefully degraded mode.

References


Fig. 1. Converting a product-of-sum expression into its sum-of-conterm equivalence.

Fig. 2. Conversion with the simplification criteria incorporated.