Realization of Nonminimal Routing Algorithm in Faulty Hypercube Parallel Process System With Using Cube Algebra

Salih GÜNEŞ, Student Member, IEEE [1], Nihat YILMAZ[1], Süleyman CANAN[2]
Nevruz ALLAHVERDİ[3]

[1] Department of Electrical-Electronic Engineering, Selcuk University, 42031-Konya/TURKEY
[2] TÜBİTAK-MAM, [3] Computer and Electronic Programming Section,T.E.F., Selcuk University.
E-mail:sgunes@karatay1.cc.selcuk.edu.tr, nhtylmz@hotmail.com, scanan@mam.gov.tr

Abstract: *In this study, an algorithm was developed to guide messages from source node to destination nodes in faulty hypercube parallel process system for nonminimal routing. In each step the nearest node to destination node is algebrically determined and this node is chosen as a new source node, therefore the propagation of the message toward the destination was increased.This algorithm was simulated on a hypercube simulation software and it was seen that good results was obtained. The message guiding was based on a formal logic. Any disadvantage wasn't seen on the performans of this algorithm, even the number of faulty nodes was too much. However this algorithm has great performans of finding route over other route fing algorithms.*

1. Introduction

Recently, hypercube multicomputers have been drawing considerable attention from many researchers. A large amount of research effort has been directed towards hypercube systems. Due to their regular structure and low diameter, hypercube multicomputers are well suited for parallel processing. Several research and commercial hypercube machines have been built in recent years[1]. A variety of routing algorithms have been proposed for hypercube systems in the past. However, most of these algorithms are not suitable for routing messages when nodes fail in a hypercube. There also has been a number of fault-tolerant routing strategies proposed in the previous research[3,10,11,12,13].

Efficient algorithms for routing a message to its destination are critical to the performance of parallel computers. A routing mechanism determines the path a message takes through the network to get from the source to destination node. It takes as input a message's source and destination nodes. It may also use about the state of the network. It returns one or more paths through the network from the source to destination node[9].

Routing mechanisms can be classified minimal or nonminimal. A minimal routing mechanism always select one shortest path between source and destination node. In minimal routing scheme each link brings a message closer to its destination, but the scheme can lead to congestion in parts of the network. A nonminimal routing scheme, in contrast, may route the message along a longer path to avoid network congestion. Nonminimal routing algorithms are usually used for faulty-tolerant because they are able to find alternative paths when all the minimal paths are faulty[9].

In part II the properties of cube algebra and cube transformation are given. In part III an example is given about the proposed method. In part IV with the nonminimal algorithm, a simulated example is given. And finally the results of the simulations is discussed.

2. Notations and Properties of Cube Algebra

An n-dimensional hypercube can be modeled as a graph Q_n (V,E),with the node set $V(Q_n)$ and edge set $E(Q_n)$, where $|V|=2^n$, $|E|=n2^{n-1}$. Each node represents a processor and its memory(Figure 1.). Each edge represents a communication link between a pair of processors. The 2^n nodes are distinctly adressed by n-

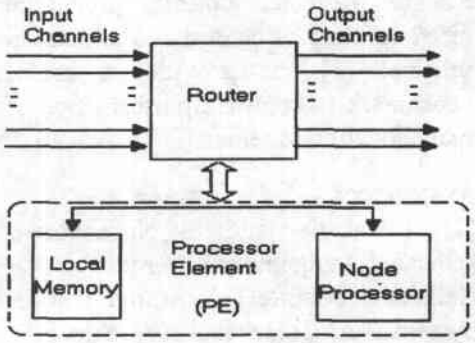

Figure 1. A generic node architecture

bit binary numbers with values from 0 to 2^{n-1}. Each node has links at n- dimensions, ranging from 0 (lowest dimension) to $n-1$(highest dimension), connecting to n neighbors. An edge connecting nodes u and v is said to be at dimension j or to be the jth dimensional edge if their binary adresses u and v differ at bit position j only. An edge in Q_n can also be represented by an n-character string with one hyphen $(*)$ and $n-1$ binary symbols $\{0,1\}$. If edge $e=(u,v) \in E(Q_n)$, u and v are jth dimensional neighbors to each other, then the edge can be represented by the binary address of node u or v with bit position j being replaced by a hyphen $(*)$. For example, $00*1$ represents edge $e=(0001,0011)$ A complete hypercube

itself can be considered special subcube, where all the bits of its address take the * value[1]. Each node is also a special subcube in which no bit of its address takes the * value, and is called 0-cube. Each link has one bit that takes the value * and is caled 1-cube, a quadrangle is called 2-cube and has two bits that take the value * . A cubes 3-cube, if its address has three *, etc[1,2]. In a hypercube there may be faulty nodes and/or faulty links. In this case the hypercube is called faulty hypercube. In case of nonfaulty nodes and links exist in n-cube it is called complete cube.

The cube algebra has a set of elements as $C = \{1, 0, *\}$. Earlier, the cube algebra was applied in connection with the presentation of the boolean functions in form of cubes and minimization[4]. These and other studies were considered with more detail and generalized [5,7]. This algebra also used for compact minimization of multiple and single output combination circuits[6] and for some algebraic transformations on hypercube[8]. Later the operations of cube algebra also used for subcube allocation in hypercube[14].

Now we examine the operation # -subtraction over subcubes and cubes. This operation is sometimes called # -operation or sharp product. *Let* $A = a_1a_2 \ldots a_i \ldots a_n$ and $B = b_1b_2 \ldots b_i \ldots b_n$, then the result R of the operation $A \# B$ will be determined by two transforms, as follows:

Step 1. $(A,B) \xrightarrow{\#} V$, where $V = v_1v_2 \ldots v_i \ldots v_n$;

Step 2. $A \xrightarrow{V} R$.

In the first step the element v_i will be determined as follows:

1) If $a_i = b_i$, then $v_i = z$;

2) If $a_i = *$ and $b_i \in \{0,1\}$, then $v_i = \overline{b}_i$;

3) If $b_i = *$, then $v_i = z$;

4) If $a_i, b_i \in \{0,1\}$ and $a_i = \overline{b}_i$, then $v_i = y$.

According to formed components of vector V the result $R = A \# B$ in the step 2 may be determined as follows:

1) If such $i \in \{1,2,\ldots,n\}$ exists for which $v_i = y$, then $A \xrightarrow{V} A$, that means $A \# B = A$;

2) If $v_i = z$ for all $i = \{1,2,\ldots,n\}$, then $A \xrightarrow{V} \emptyset$, that means $A \# B = \emptyset$;

3) If $v_j, v_k \ldots, v_m \in \{0,1\}$ for the coordinates $j, k, \ldots, m$, then $A \xrightarrow{V} \{a_1a_2 \ldots a_{j-1}v_ja_{j+1} \ldots a_n, a_1a_2 \ldots a_{k-1}v_ka_{k+1} \ldots a_n, a_1a_2 \ldots a_{m-1}v_ma_{m+1} \ldots a_n\}$.

3. Method

The method proposed in this work could be investigated in three stages. In the first stage the faulty nodes are subjected to coordinate substraction from nonfaulty cube. By applying this operation, F the faulty nodes set are minimazed to F_{min} set.

Here, neighbour nodes related to each other are represented together. In the second stage is the transformation applied which is the coordinate substraction mentioned before.

The new source nodes set are found by subjecting these two operation with the source node together. (In this set the initial source is present). Afterwards, the initial source is substracted from this set and pure source node sets are found. The initial source is included into F_{min} set. In the third stage, the recently obtained sources are compared algebrically with destination nodes. This operation is repeated up to reach destination node.

Now we search the method for determination of nonfaulty and maximal subcubes, each of which includes a current source in every stage.

Assuming $A = a_1a_2 \ldots a_i \ldots a_n$ and $f_j = \varphi_{j1}\varphi_{j2} \ldots \varphi_{ji} \ldots \varphi_{jn}$ are the codes of the source node and any faulty node, respectively. Then the cube f_j may be transformed to cube, $Q_j = q_{j1}q_{j2} \ldots q_{ji} \ldots q_{jn}$. according Nadjafov and Kahramanov[6]

1. if $\varphi_{ji} = a_i$ then $Q_j = \varphi_{ji}$,

   if $\varphi_{ji} = a_i$ or $\varphi_{ji} = *$ then $Q_j = *$

   $(j=1,\ldots,l;\ i=1,\ldots,n.)$

$$f_j \xrightarrow{A} Q_j = M_l \in Q$$

2. The set of Q_m is formed by taking out the nonmaximal cubes from the set of $Q = \{Q_j\}_{j=1}^{l}$.

3. The set of Qm is #-subtracted from complete n-cube. Nonmaximal and repeated cubes are taken out from this result and thus, the set of D_A including the source node A is formed.

4. The cubes in the set Q_{m1} are logical intersected, in the result of which the cube D_1 is formed.

5. All the cubes with dimensions $r \in \{1,2,\ldots,n-2\}$ in the set Q_{m2} are transformed to n-r pieces $(n-1)$-cubes, i.e. $r \times 1 \rightarrow (n-1) \times (n-r)$. The repeated cubes in the result of transformations are taken out and the set D_2 is formed.

6. The sets D_1 and D_2 are logical intersected in the result of which the set of subcubes D_A including the source node A is formed.

Example: For $A = 0010$, $F_{min} = \{0*01, 01*1, *100, 101*, 0000\}$ we find all *SNMS* which includes the node A. On the first and second methods we mark the transform $f_j \xrightarrow{A} Q_j \in Q$ as M_1 and M_2 respectively. The steps of the definition of *SNMS* are

the same for the corresponding above mentioned methods. According to :

1. $Q = M_1$: (F_{min} , A) = M_1: ({0*01, 01*1, *100, 101*, 0000},0010) = {**01, *1*1, *10*, 1***, **0*}.

2. Q_m = {*1*1, 1***, **0*}.

3. D_A = **** # Q_m = **** # {*1*1, 1***, **0*}={001*, 0*10}.

4. Nonminimal Routing Algorithm

The three stage explained in this method's part are:
I. Subcube Building Algorithm
II. Transformation Algorithm
III. Turn to Destination Algorithm

// Subcube Building Algorithm //

```
Begin
SonucMax:=0;
Repeat
for i:=1 to Group1number-1 do
Kup1:=Group1Data[i];
Kup2:=Group2Data[1];
Begin
lenght(Result):=lenght(cube1);
For i=1 to Length(cube1) Do
begin
if (Cube2 [i]='*') Or (Cube1[i]= Cube2[i]) Then
Result[i]:='Z';
if (Cube1[i]='*') And (Cube2[i]≠'*') Then
Result[i]:=İnvers(Cube2[i]);
if (Cube1[i] ⊃ ['0','1']) And (Cube2[i] ⊃ ['0','1']) And
(Cube1[i]≠ Cube2[i]) Then Result[i]:='y';
End;
if 'y'∈Result then  begin Result:= Cube1;
İntervalResultAdd(result); Exit; End;
OldResult:=Result;
ZNumbert:=0;
for i:=1 to Length(Result) do if Result[i]='Z' Then
Begin inc(ZNumber); ZPosition[ZNumber]:=i; End;
StarNumber:=0;
for i:=1 to Length(Cube1) do  if Cube1[i]='*' then
begin  inc(StarNumber ); StarPosition [StarNumber ]:=i;
End;
if ZNumber=Length(Result) Then
Begin İntervalResultAdd('F'); exit; End;
for i:=1 to Length(Result) do
if Result [i]='Z' then Result [i]:=Cube1[i];
Position:=0;
for i:=1 to starNumbert do
begin
if OldResult[starPosition[i]]='Z' then Continue;
inc(Position);
EndString[Position]:=Copy(Cube1,1,StarPosition[i]-1);
EndString[Position]:=EndString[Position]+
Result[StarPosition [i]];
EndString [Position ]:= EndString[Position]+Copy(Cube1,
StarPosition [i]+1 ,Length(Cube1)- StarPosition [i]);
for k:=1 to ZNumber do
EndString[Position][Zposition[k]]:=Cube1[ZPosition[k]];
End;
for i:=1 to StarNumber do if EndString[i]≠ '' Then
İntervalResultAdd (Endstring[i]); End;
Extract elemens from Group2
Group1Data:=Results;
Group1Number:=ResultMax+1;
if Group2Number>1 Then V:=0;
Until Group2Number-1=0;
End;
```

// Transformation Algorithm //

```
Procedure Transformasyon;
Var
k,t:integer;
Begin
for k:=1 to 255 do AraDizi[k]:='';
For k:=1 to HataliAdet do begin
AraDizi[k]:=YildizUret; { Dizi uygun bir biçimde
boyutlandırılıyor }
For t:=1 to Length(HataliNodlar[1]) do
if HataliNodlar[k][t]<>Source[t] Then
AraDizi[k][t]:=HataliNodlar[k][t] Else AraDizi[k][t]:='*';
End;
SonucMax:=0;
For k:=1 to HataliAdet do AraSonucEkle(AraDizi[k]);
Sadelestirme;
For k:=1 to SonucMax do AraDizi[k]:=Sonuclar[k];
AraAdet:=SonucMax;
End;
```

// Turn to Destination Algorithm //

```
Procedure Hedefeyonel
Begin
   For i:=1 to SonucMax Do Begin
S1:=Sonuclar[i];
S2:= hedefin adı;
FarkAdet:=0;
 p:=1;
 For k:=Length(S1) Downto 1 do Begin
       if S1[k]=S2[k] Then FarkAdet:=FarkAdet+J;
       p:=p*2;    End;
          Dizi[i]:=FarkAdet;
       DiziX[i]:=i; End;
   For i:=1 to SonucMax-1 do
     For j:=i+1 to SonucMax do
       if Dizi[i]<Dizi[j] Then Begin
                   Gecici:=Dizi[i];
                   Dizi[i]:=Dizi[j];
                   Dizi[j]:=Gecici;
                   Gecici:=DiziX[i];
                   DiziX[i]:=DiziX[j];
                   DiziX[j]:=Gecici
End;
For i:=1 to SonucMax do DiziData[i]:=Sonuclar[DiziX[i]];
For i:=1 to SonucMax do Sonuclar[i]:=DiziData[i];
End;
```

Example: Source node: 00000, destination node: 11111 and Faulty nodes ={ 00110, 00101, 01111, 10000, 10001, 10011, 10101, 10111, 11001}
The message orientation from source to destination node is as shown below:

00000→01000→01010→01110→01101→11100→ 11101→11111

The visualization obtained from the results is shown in Fig.2

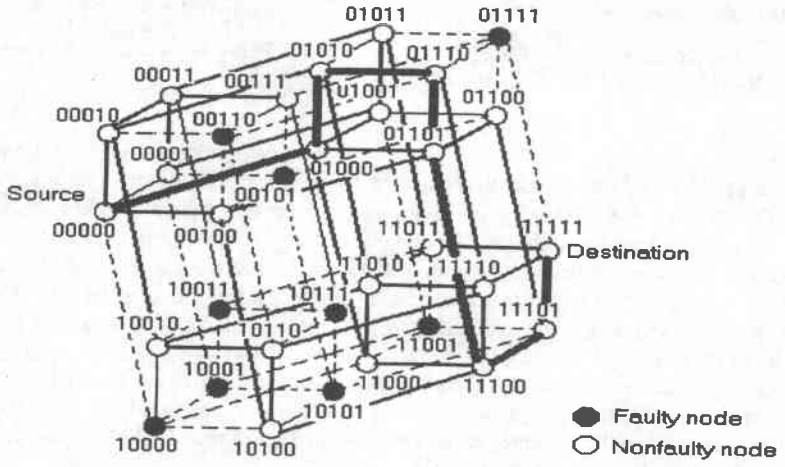

Figure 2. Nonminimal routing algorithm example for 5-dimensional faulty hypercube

4. Conclusions

In this study, an application of nonminimal routing message orientation algorithm of cube algebra has been implemented. The method which was used in the shortest path routing algorithm is simulated in hypercube simulator as nonminimal routing algorithm. When there is a congestion in message forwarding (if all the neighbours of the current node are faulty or the current node is busy), the message is going on his own way using the nearest node before that point.In the cases where practically finding the shortest path is not important, it is appearing a fast message forwarding algorithm. In the same algorithm, the application of the destination node incrementation is appearing as an advantage.

References

[1] Y. Saad and M.H.Schultz, "Topological Properties of Hypercubes", *IEEE Trans. on Comput. volC-* 37, no. 7, 867-872, July 1988.

[2] M.Chen and K.G.Shin, "Processor Allocation in An n-Cube Multiprocessor Using Gray Codes," *IEEE Transaction on Computers*, vol.C-36. no.12, pp. 1396-1407, 1987.

[3] Youran Lan, "An Adaptive Fault-Tolerant Routing Algorithm for Hypercube Multicomputers," *IEEE Transactions on Parallel and Distributed Systems*. vol. 6, no. 11, pp. 1147-1152, Nov. 1995

[4] S.S. Kahramanli and N.M. Allahverdi, "Compact Method of Minimization of Boolean Functions with Multiple Variables," *Proc. Intern. Symp. of Application of Computers*, pp. 433-440, 1993.

[5] R. Miller, "Switching Theory" vol. 1, *Combinational Circuits*, Moscow, 1970.

[6] E.M. Nadjafov and S.S. Kahramanov, "On the Synthesis of Multiple Output Switching Scheme," *Scientific Notes of Azerbaijan Institute of Petroleum and Chemistry*, vol. 9, no. 3, pp. 65-69, 1973.

[7] J.P. Roth, "Algebraic Topological Methods for The Synthesis of Switching Systems in n-variables," *The Institute for Advanced Study*, Princeton, , ESP56-02, New Jersey, 1956.

[8] S.S. Kahramanlı and N.M. Allahverdi, "Algebraic Approach to Transformation on Hypercube System," *Mathematical and Computational Applications*, vol.1, no.1, pp. 50-59, 1996.

[9] Jose Duato, Sudhakar Yalamanchili and Lionel Ni, "Interconnection Networks," *IEEE Computer Society Press*, 1997.

[10] Ge-Ming Chiu and Shui-Pao Wu, "A Fault-Tolerant Routing Strategy in Hypercube Multicomputers," *IEEE Transaction on Computers*, vol. 45, no. 2, pp. 143-155, Feb. ,1996.

[11] T.C. Lee and J.P.Hayes, "A Fault-Tolerant Communication Scheme for Hypercube Computers," *IEEE Computer*, vol. 26, no.2, pp. 62-76, Feb. 1993.

[12] J. Duato, "On the Design of Deadlock-Free Adaptive Routing Algorithms for Hypercubes: Theoretical Aspects," *Proc. Second European Distributed Memory Conf.*, pp. 234-245, 1991

[13] P.T. Gaughan and S. Yalamanchili, "Adaptive Routing Protocols for Hypercube Interconnection Networks," *Computer*, vol. 26, no.5, pp. 12-23, May 1993.

[14] S. Dutt and J.P. Hayes, "Subcube Allocation in Hypercube Computers," *IEEE Trans.Comput*, vol.40, no.3, pp. 341-352, 1991.