American Journal of Networks and Communications
2017; 6(2): 35-46
http://www.sciencepublishinggroup.com/j/ajnc
doi: 10.11648/j.ajnc.20170602.11
ISSN: 2326-893X (Print); ISSN: 2326-8964 (Online)

An Optimal Routing Algorithm for Horizontal Moving Signals in OCN for Massively Parallel Systems with Faulty Node/Link

Sanjukta Mohanty[1], Prafulla Kumar Behera[2]

[1]Department of Computer Application, North Orissa University, Sriram Chandra Vihar, Baripada, Odisha, India
[2]Department of Computer Science and Applications, Utkal University, Vani Vihar, Bhubaneswar, Odisha, India

Email address:
Sanjuktamohanty2005@gmail.com (S. Mohanty), p_behera@hotmail.com (P. K. Behera)

To cite this article:
Sanjukta Mohanty, Prafulla Kumar Behera. An Optimal Routing Algorithm for Horizontal Moving Signals in OCN for Massively Parallel Systems with Faulty Node/Link. *American Journal of Networks and Communications.* Vol. 6, No. 2, 2017, pp. 35-46.
doi: 10.11648/j.ajnc.20170602.11

Received: March 9, 2017; **Accepted**: March 21, 2017; **Published**: April 7, 2017

Abstract: An Octagon-Cell Interconnected Network (OCN) has many attractive features. To represent OCN an undirected graph G = (V, E) is used, in which V is the set of nodes in the graph and E is the set of edges in the graph. Already the optimal routing algorithm had been presented with its features in our past research work. This research paper presents the optimal routing algorithm for horizontal moving signals in OCN with a faulty node/link along the optimal path. OCN is expandable. Also the algorithm tells that, even the OCN is expanded; there is no effect to find the optimal path in presence of faulty nodes. OCN can be utilized in massively parallel computing. In a massively parallel system a large number of processors are used to perform a set of coordinated computation simultaneously. So OCN is assumed a type of integrated circuit with an array of hundreds or thousands of central processing units (CPUs) and random-access memory banks.

Keywords: Interconnected Networks, Octagon-Cell, Routing Algorithm, Parallel System

1. Introduction

In this research paper an optimal routing algorithm is introduced for interconnected processors, which communicates messages in a faulty octagon-cell. The optimal routing algorithm for octagon-cell interconnected network along with its attractive features had been described in [2]. Also the fault tolerant routing algorithm for horizontal moving messages for this network had been described in [1], which was not optimal, but approximated towards optimal value. In this paper an optimal fault tolerant routing algorithm is derived for horizontal moving messages in the network and calculated the relative error for each sample test. An octagon-cell interconnected network has many attractive features such as constant node degree, desirable diameter, bisection width [2]. The fundamental formula of desirable features of this cell has been presented in [28].

In an interconnection network, a fault tolerance scheme means the ability to continue operating in presence of faulty nodes / link failures [4]-[15]. If the number of interconnected processors rises, the probability of having faulty nodes increases and for successful transmission it is very much essential to find another fault free path [16], [23]-[25].

Selecting optimal paths for efficient inter process communication is essential in parallel processing systems. In this system, if each and every processor has the status of all processors then an optimal routing can be possible. In a system it may be possible for each component suffers from hardware or software problem. If the system can't handle the faulty problem, that is unreliable, inefficient [16]-[22].

2. Related Work

The fault tolerant routing schemes have been developed for various interconnected networks by many researchers. A fault tolerant scheme has been proposed for hexagonal arrays in [26]. It has been described that the routing scheme makes the reconstructed array transparent to the various algorithms utilizing the hexagonal array.

A mesh embedded interconnected hypercube network has

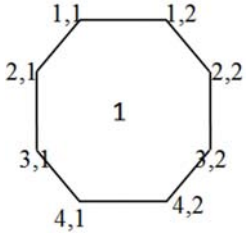
been analyzed to arrive at fault tolerant communication. An efficient routing algorithm has been proposed that can route a message from a source node to the destination in presence of fault free or single/multiple faulty nodes in mesh embedded hypercube interconnection networks in [4].

A fault tolerant routing algorithm has been described for star interconnection network in the presence of faults in [27]. A new fault tolerance algorithm has been described in [16] for hex-cell interconnection network and the algorithm guarantees the delivery of messages even with the presence of component failure.

An efficient distributed fault-tolerant routing algorithm for the hypercube has been described in [17] on the existence of a complete set of node-disjoint paths between any two nodes. It was presented that when a message is blocked by a node failure, the source node is warned and requested to switch to a different node-disjoint path.

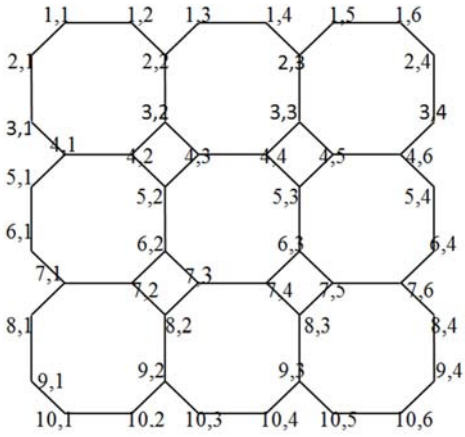
3. Octagon-Cell Network Topology

The optimal routing algorithm has been described for OCN in [2]. An octagon-cell has eight nodes. It has d levels numbered from 1 to d with depth d. Level 1 represents one octagon-cell. Level 2 represents eight octagon-cells surrounding the octagon-cell at level 1. Level 3 represents sixteen octagon-cells surrounding the eight octagon-cells at level 2 and so on [2].



(X, Y represents line no-X with node no-Y)

Figure 1. *Addressing nodes in Octagon-Cell with level-1.*



(X, Y represents line no-X with node no-Y)

Figure 2. *Addressing nodes in Octagon-Cell with level- 2.*

Due to the recursive structure of OCN routing can be done easily. The level numbering scheme is used in this algorithm. Each node in octagon-cell is identified by a pair (X, Y), Where X denotes the line number in which the node exists and Y denotes serial number of the node in that line. A node with the address (1,1) is the first node in first line. A node with the address (1,2) is the second node in first line and so on. Here again the optimal routing algorithm is presented for horizontal and vertical moves [2].

Case-1(a) Optimal Routing Algorithm for Horizontal Move for lines m where m mod 3 = 1 && $X_s < [(d*5) + (d-2)]$ [If ($X_s = X_d$ && $Y_s < Y_d$) and If ($X_s = X_d$ && $Y_s > Y_d$)]

Move (X_s,Y_s,X_d,Y_d)
If ($y_s < y_d$)
If ($x_s = x_d$ && y_s is odd)
Move (x_s, y_s+1, x_d, y_d)
Else If ($x_s = x_d$ && y_s is even)
Move (x_s+1, y_s/2+1, x_d, y_d)
Else If ($x_s \neq x_d$)
Move (x_s-1, 2y_s-1, x_d, y_d)
Else
Destination reached
Else
If ($y_s > y_d$)
If ($x_s = x_d$ && y_s is odd)
Move (x_s+1, y_s/2+1, x_d, y_d)
Else If ($x_s = x_d$ && y_s is even)
Move (x_s, y_s -1, x_d, y_d)
Else If ($x_s \neq x_d$)
Move (x_s-1, 2y_s-2, x_d, y_d)
Else
If ($y_s = y_d$)
Go to Vertical Move
Else
Destination reached

Case-1(b) Optimal Routing Algorithm for Horizontal Move for lines m where m mod 3 = 1 && $X_s = [(d*5) + (d-2)]$ [If ($X_s = X_d$ && $Y_s < Y_d$) and If ($X_s = X_d$ && $Y_s > Y_d$)]

Move (X_s,Y_s, X_d,Y_d)
If ($y_s < y_d$)
If ($x_s = x_d$ && y_s is odd)
Move (x_s, y_s+1, x_d, y_d)
Else If ($x_s = x_d$ && y_s is even)
Move (x_s-1, y_s/2+1, x_d, y_d)
Else If ($x_s \neq x_d$)
Move (x_s+1, 2y_s-1, x_d, y_d)
Else
Destination reached
Else
If ($y_s > y_d$)
If ($x_s = x_d$ && y_s is odd)
Move (x_s-1, y_s/2+1, x_d, y_d)
Else If ($x_s = x_d$ && y_s is even)
Move (x_s, y_s-1, x_d, y_d)
Else If ($x_s \neq x_d$)
Move (x_s+1, 2y_s-2, x_d, y_d)
Else
If ($y_s = y_d$)

Go to Vertical Move
Else
Destination reached

Case-2 Optimal Routing Algorithm for Horizontal Move for lines m where m mod 3 ≠ 1 [If ($X_s = X_d$ && $Y_s<Y_d$) and If ($X_s = X_d$ && $Y_s> Y_d$)]

Move (X_s,Y_s, X_d,Y_d)
If ($x_s = x_d$ && $y_s < y_d$ && x_s mod 3=2)
If ($x_s = x_d$ && x_s mod 3=2)
Move (x_s-1, 2y_s-1, x_d, y_d)
Else If ($x_s \neq x_d$ && y_s is odd)
Move (x_s, y_s+1, x_d, y_d)
Else If ($x_s \neq x_d$ && y_s is even)
Move (x_s+1, y_s/2+1, x_d, y_d)
Else
Destination reached
Else
If ($x_s = x_d$ && $y_s < y_d$ && x_s = 3n)
If ($x_s = x_d$ && x_s = 3n)
Move (x_s+1, 2y_s-1, x_d, y_d)
Else If ($x_s \neq x_d$ && y_s is odd)
Move (x_s, y_s+1, x_d, y_d)
Else If ($x_s \neq x_d$ && y_s is even)
Move (x_s-1, y_s/2+1, x_d, y_d)
Else
Destination reached
Else
If ($x_s = x_d$ && $y_s > y_d$ && x_s mod 3 = 2)
If ($x_s = x_d$ && x_s mod 3 = 2)
Move (x_s-1, 2y_s-2, x_d, y_d)
Else If ($x_s \neq x_d$ && y_s is even)
Move (x_s, y_s-1, x_d, y_d)
Else If ($x_s \neq x_d$ && y_s is odd)
Move (x_s+1, y_s/2+1, x_d, y_d)
Else
Destination reached
Else
If ($x_s = x_d$ && $y_s > y_d$ && x_s = 3n)
If ($x_s = x_d$ && x_s = 3n)
Move (x_s+1, 2y_s-2, x_d, y_d)
Else If ($x_s \neq x_d$ && y_s is even)
Move (x_s, y_s-1, x_d, y_d)
Else If ($x_s \neq x_d$ && y_s is odd)
Move (x_s-1, y_s/2+1, x_d, y_d)
Else
If ($y_s = y_d$)
Go to Vertical Move
Else
Destination reached

Case-3 Optimal Routing Algorithm for Vertical Move for lines m = X_s where m mod 3 = 1 [If ($X_s < X_d$ && $Y_s = Y_d$)]

Move (X_s,Y_s, X_d,Y_d)
If (y_d = 1 || 2) // Logical OR
For (i = x_s to x_d-1 in increasing order)
Move (x_s+1, y_s, x_d, y_d)
Else If (x_s mod 3 = 1 && x_d mod 3 ≠1 && y_s is odd)
Move (x_s, y_s+1, x_d, y_d)
Else If ((x_s mod 3 = 1 && x_d mod 3 ≠ 1 && y_s is even && $x_s < x_d$) || (x_s mod 3 = 1 && x_d mod 3 = 1))
Move (x_s+1, y_s/2+1, x_d, y_d)
Else If (x_s mod 3 = 1 && x_d mod 3 ≠ 1 && y_s is even && $x_s > x_d$)
Move (x_s-1, y_s/2+1, x_d, y_d)
Else If ((x_s mod 3 = 2 && x_d mod 3 ≠ 1 && $x_s \neq x_d$) || (x_s mod 3 = 2 && x_d mod 3 = 1))
Move (x_s+1, y_s, x_d, y_d)
Else If ((x_s = 3n && x_d mod 3 ≠ 1) && ($y_s \neq y_d$) || (x_s = 3n && y_d is odd && x_d mod 3 = 1))
Move (x_s+1, 2y_s-1, x_d, y_d)
Else If (x_s = 3n && x_d mod 3 = 1 && y_d is even) || (x_s = 3n && $y_s = y_d$ && x_d mod 3 ≠ 1)
Move (x_s+1, 2y_s-2, x_d, y_d)
Else If ($x_s = x_d$)
Go to Horizontal Move
Else
Destination reached

Case-4 Optimal Routing Algorithm for Vertical Move for lines m = X_s where X_s mod 3 ≠1 [If ($X_s < X_d$ && $Y_s = Y_d$)]

Move (X_s, Y_s, X_d, Y_d)
If (y_d = 1 || 2) // Logical OR
For (i = x_s to x_d-1 in increasing order)
Move (x_s+1, y_s, x_d, y_d)
Else If (x_s mod 3 = 2)
Move (x_s+1, y_s, x_d, y_d)
Else If (x_s = 3n)
Move (x_s+1, 2y_s-2, x_d, y_d)
Else If (x_s mod 3 = 1 && x_d mod 3 = 1 && y_s is even && $y_s \neq y_d$ && $x_s \neq x_d$)
Move (x_s, y_s-1, x_d, y_d)
Else If ((x_s mod 3 = 1 && x_d mod 3 = 1 && y_s is odd && $y_s \neq y_d$ && $x_s \neq x_d$) || (x_s mod 3 = 1 && x_d mod 3 ≠ 1))
Move (x_s+1, y_s/2+1, x_d, y_d)
Else If (x_s mod 3 = 1 && x_d mod 3 = 1 && $x_s = x_d$)
Go to Horizontal Move
Else If (x_s mod 3 = 1 && x_d mod 3 = 1 && $y_s = y_d$)
Go to Vertical Move Top to Bottom for X_s mod 3 = 1
Else
Destination reached

Note1 In similar way the vertical moves for $X_s > X_d$ can be followed. So the algorithm for this case is not presented. This can be followed from [2].

4. Optimal Fault Tolerant Routing Algorithm

There are six cases for optimal routing for horizontal moving messages. In this research paper an optimal fault tolerant routing algorithm is presented for the following cases.

Case: 1

Fault tolerant routing algorithm for horizontal move for lines m mod 3 = 1 [Move from left to right, If ($X_s = X_d$ && $Y_s < Y_d$)]

Case: 2

Fault tolerant routing algorithm for horizontal move for lines m mod 3 = 1 [Move from right to left, If ($X_s = X_d$ && $Y_s > Y_d$)]

Case: 3

Fault tolerant routing algorithm for horizontal move for lines m mod 3 =2 [Move from left to right, If ($X_s = X_d$ && $Y_s < Y_d$)]

Case: 4

Fault tolerant routing algorithm for horizontal move for lines m mod 3 =2 [Move from right to left, If ($X_s = X_d$ && $Y_s > Y_d$)]

Case: 5

Fault tolerant routing algorithm for horizontal move for lines m = 3n (n is any natural number) [Move from left to right, If ($X_s = X_d$ && $Y_s < Y_d$)]

Case: 6

Fault tolerant routing algorithm for horizontal move for lines m = 3n (n is any natural number) [Move from right to left, If ($X_s = X_d$ && $Y_s > Y_d$]

The fault tolerant algorithm is based on the optimal routing scheme in octagon-cell network [2]. Here six main cases are presented for fault tolerant scheme. The following notations have been used in our algorithm.

OSN-Original Source Node
FN-Faulty Node
FL-Faulty Link
DN-Destination Node
NBD- Neighborhood
Horizontal Right-HOZR
Horizontal Left-HOZL
Vertical Top-VERT
Vertical Bottom-VERB

4.1. Description of Model

The fault tolerance routing algorithm for octagon-cell interconnected network has been presented for horizontal moving messages in [1]. In this research paper the same protocol is considered as in [1]. Here the optimized algorithm is explained. It is assumed that each node has information about its three consecutive nodes on the original optimal path in which it could have gone if there won't be any faulty nodes, link failures and dead end state. Each node say 'A' checks its three consecutive nodes and links simultaneously along its original path. If any error occurs in one of three consecutive links or nodes, then the algorithm will work with respect to the address of source node at 'A'.

When a signal passes from a source node to a destination, it is very much essential to find a path of non-faulty nodes. For this purpose, each node can store the information about its three consecutive nodes and links along the original path. There are three possible cases for a node in an octagon-cell network. That is:

a) There are fault free nodes / links along the original path. This is called normal state.

b) If any faulty node or link occurs in the original path, this situation is called faulty state. This situation can be handled by the nodes along the path, because each node in the path has the status of its three consecutive nodes and links. So the original path will get changed by using the algorithm.

c) If destination node is faulty, then this situation is called dead end state.

Note2

a. Since it is derived the optimal fault tolerant scheme for the presence of faulty node or faulty link along the optimal path, so in all cases the link failure conditions are not mentioned, because the algorithm for link failure case is equivalent to the node failure along the optimal path. That is if the link $(x_s, y_s) \rightarrow (x_s^*, y_s^*)$ is failed along the optimal path, then this situation is equivalent to the left node failure of this link. If the right node is failed, then this is equivalent to the link failed connecting to (x_s^*, y_s^*) and (x_s^{**}, y_s^{**}).

b. The symbol "d" represents depth of the network and the word is used in our algorithm "w.r.t" represents "with respect to".

4.2. Description of the Fault Tolerant Scheme

This scheme has already been presented in [1]. When a message is to be sent from a source node to a destination node, the algorithm first finds the optimal path [2]. The message moves on the optimal path. In that path each and every node has the status about three consecutive nodes / links. If 'B' be the faulty node and 'A' be another fault free node on that path, then before reaching at 'B', the message first reaches at 'A'. 'A' has the status about next three consecutive nodes / links. 'B' is the node amongst the three consecutive nodes. So at that situation, the message suddenly goes away from that original optimum path and finds another fault free optimal path with respect to node 'A'. Else dead end may occur for which the message fails to reach at the destination. Let the optimal path be A→B→C→D→E→F, where the source node is 'A' and the destination node is 'F'. If 'F' is faulty then dead end occurs. If 'B' or 'C' or 'D' is faulty, then the algorithm will work with respect to the node 'A'. If 'E' is faulty, then the algorithm will work with respect to the node 'B'. If the optimal path be A→B→C→D, where the source and destination nodes are 'A' and 'D' respectively and if faulty node is 'B' or 'C', then in this case the algorithm will work w.r.t the source node 'A'.

Note3

To reduce the complexity of the algorithm, the following fundamental groups of pseudocodes are defined.

G1: Move $(x_s\text{-}1, y_s/2+1, x_d, y_d)$
Move $(x_s\text{-}1, y_s, x_d, y_d)$
Move $(x_s\text{-}1, 2y_s\text{-}1, x_d, y_d)$

G2: Move $(x_s\text{-}1, y_s/2+1, x_d, y_d)$
Move $(x_s\text{-}1, y_s, x_d, y_d)$
Move $(x_s\text{-}1, 2y_s\text{-}1, x_d, y_d)$
Move (x_s, y_s+1, x_d, y_d)

G3: Move $(x_s+1, y_s/2+1, x_d, y_d)$

Move (x_s+1, y_s, x_d, y_d)
Move (x_s+1, 2y_s-1, x_d, y_d)
G4: Move (x_s+1, y_s/2+1, x_d, y_d)
Move (x_s+1, y_s, x_d, y_d)
Move (x_s+1, 2y_s-1, x_d, y_d)
Move (x_s, y_s+1, x_d, y_d)
G5: Move (x_s-1, y_s/2+1, x_d, y_d)
Move (x_s-1, y_s, x_d, y_d)
Move (x_s-1, 2y_s-2, x_d, y_d)
G6: Move (x_s-1, y_s/2+1, x_d, y_d)
Move (x_s-1, y_s, x_d, y_d)
Move (x_s-1, 2y_s-2, x_d, y_d)
Move (x_s, y_s-1, x_d, y_d)
G7: Move (x_s+1, y_s/2+1, x_d, y_d)
Move (x_s+1, y_s, x_d, y_d)
Move (x_s+1, 2y_s-2, x_d, y_d)
Move (x_s, y_s-1, x_d, y_d)
G8: Move (x_s+1, y_s/2+1, x_d, y_d)
Move (x_s+1, y_s, x_d, y_d)
Move (x_s+1, 2y_s-2, x_d, y_d)

Case-I Fault tolerant routing algorithm for horizontal move for lines m, Where m mod 3 = 1 [Move from left to right, If ($X_s = X_d$ && $Y_s < Y_d$)]

Find the optimal path from OSN to DN
If (path is fault free)
Return true
Else If (FN is DN)
Return dead end
Else Go to the Sub-Cases

Sub-Case: A If (OSN x_s = 1 && OSN y_s is odd)

Case-1: If (FN is 1st NBD of OSN)
Step-1 Move VERB till $x_s = x_s$+3
Step-2 If (y_d is odd)
Move HOZR till $y_s = y_d$-1
Go to G1
Else If (y_d is even)
Move HOZR till $y_s = y_d$-2
Go to G2
Case-2: Else If (FN is 2nd NBD of OSN)
Step-1 Move VERB till $x_s = x_s$+3
Step-2 If (y_d is odd && NBD of FN)
Move HOZR till $y_s = y_d$+1
Move VERT till $x_s = x_d$
Move (x_s, y_s-1, x_d, y_d)
Else If (y_d is odd && FN is not NBD of (x_d, y_d))
Move HOZR till $y_s = y_d$-1
Go to G1
Else If (y_d is even && FN is NBD of NBD of (x_d, y_d))
Move HOZR till $y_s = y_d$
Move VERT till $x_s = y_d$)
Else If (y_d is even&& FN is not NBD of NBD of (x_d, y_d))
Move HOZR till $y_s = y_d$-2
Go to G2
Case-3: Else If (FN is 3rd NBD of OSN)
Step-1 Move (x_s, y_s+1, x_d, y_d)
Go to G3
Step-2 If (FN is NBD of (x_d, y_d) && y_d is even)
Move (x_s, y_s+1, x_d, y_d)
Move VERT till $x_s = x_s$-3
Else If (FN is not NBD of (x_d, y_d))
If (y_d is odd)
Move HOZR till $y_s = y_d$-1
Go to G1
Else If (y_d is even)
Move HOZR till $y_s = y_d$-2
Go to G2
Case-4: If (FN is at the line x_s+1, where x_s is OSN && FN is Not w.r.t OSN)
Step-1 Move (x_s+1, y_s, x_d, y_d)
Move (x_s+1, 2y_s-1, x_d, y_d)
Go to Step-2 of Case-2 of Sub-Case: A
Case-5: Else If (FN is on the line of OSN && Not w.r.t OSN)
If (FN y_s is odd)
Go to Step-1 and Step-2 of Case-3 of Sub-Case: A
Else If (FN is y_s even)
Go to Case-3 of Sub-Case: B

Sub-Case: B If (OSN x_s = 1 && OSN y_s is even)

Case-1: If (FN 1st NBD of OSN i. e on the line x_s+1, where x_s is OSN)
Step-1 Move (x_s, y_s-1, x_d, y_d)
Go to Case-2 of Sub-Case: A
Case-2: Else If (FN is 2nd NBD of OSN)
Step-1 Go to Step-1(excluding 1st line) and Step-2 of Case-3 Sub-Case: A
Case-3: Else If (FN is 3rd NBD of OSN)
Step-1Go to G4
Step-2 If ($y_s = y_d$-1)
Go to G1
Else If ($y_s \neq y_d$-1 && y_d is odd)
Move HOZR till $y_s = y_d$-1
Go to G1
Else If (y_d is even && $y_s = y_d$-2)
Go to G2
Else If (y_d is even && $y_s \neq y_d$-2)
Move HOZR till $y_s = y_d$-2
Go to G2
Case-4: Else If (FN is at the line x_s+1, where x_s is OSN && FN is Not w.r.t OSN)
Step-1 Move (x_s+1, y_s, x_d, y_d)
Move (x_s+1, 2y_s-1, x_d, y_d)
Go to Step-2 of Case-2 of Sub-Case: A
Case-5: Else If (FN is on the line of OSN && Not w.r.t OSN)

Step-1 If (FN y_s is odd)
Go to Step-1 and Step-2 of Case-3 of Sub-Case: A
Else If (FN is y_s even)
Go to Case -3 of Sub-Case: B

Sub-Case: C If (OSN ($x_s > 1$ && $x_s < [(d*5) + (d-2)]$ && y_s is odd))

Case-1: If (FN is 1st NBD of OSN)
Step-1 Go to Case-1 of Sub-Case: A
Case-2: Else If (FN is 2nd NBD of OSN)
Step-1 Move (x_s, y_s+1, x_d, y_d)
Move (x_s-1, y_s/2+1, x_d, y_d)
Move (x_s+1, 2y_s-1, x_d, y_d)
Step-2 If ($y_s = y_d$)
Destination reached
Else If ($y_s \neq y_d$)
Move HOZR till $y_s = y_d$
Case-3: If (FN is 3rd NBD of OSN)
Step-1Go to Case-3 of Sub-Case: A
Case-4: Else If (FN is at the line x_s+1, where x_s is OSN && Not w.r.t OSN)
Step-1 Move (x_s-1, 2y_s-1, x_d, y_d)
Move (x_s, y_s+1, x_d, y_d)
Move (x_s-1, y_s/2+1, x_d, y_d)
Move (x_s+1, 2y_s-1, x_d, y_d)
Step-2 If ($y_s \neq y_d$)
Move HOZR till $y_s = y_d$
Else If ($y_s = y_d$)
Destination reached
Case-5: Else If (FN is on the OSN line && FN is Not w.r.t OSN)
Step-1 If (FN y_s is even)
Go to Case-3 of Sub-Case-B
Else If (FN y_s is odd)
Go to Case-3 of Sub-Case-A

Sub-Case: D If (OSN (x_s > 1 && x_s < [(d*5 + (d-2)] && y_s is even))

Case-1: If (FN is 1st NBD of OSN)
Step-1Go to Step-1(excluding 1st line) and Step-2 of case-2 of Sub-Case: C
Case-2: Else If (FN is 2nd NBD of OSN)
Step-1 Go to Step-1(excluding 1st line) and Step-2 of Case-3 of Sub-Case: A
Case-3: Else If (FN is 3rd NBD of OSN)
Step-1 Go to Step-1 and Step-2 of Case-3 of Sub-Case: B
Case-4: Else If (FN is at the line x_s+1, where x_s is OSN && Not w.r.t OSN)
Step-1Move (x_s-1, 2y_s-1, x_d, y_d)
Move (x_s, y_s+1, x_s, y_d)
Move (x_s-1, y_s/2+1, x_d, y_d)
Move (x_s+1, 2y_s-1, x_d, y_d)
Step-2 If ($y_s \neq y_d$)
Move HOZR till $y_s = y_d$
Else If ($y_s = y_d$)
Destination reached
Case-5: Else If (FN is on the OSN line && FN is Not w.r.t OSN)

Step-1 If (FN y_s is even)
Go to Case-3 of Sub-Case-B
Else If (FN y_s is odd)
Go to Case-3 of Sub-Case-A

Sub-Case: E If (OSN (x_s = [(d*5) + (d-2)] &&y_s is odd))

Case-1: If (FN is 1st NBD of OSN)
Step-1 Move VERT till $x_s = x_s$-3
Step-2 If (y_d is odd)
Move HOZR till $y_s = y_d$-1
Go to G3
Else If (y_d is even)
Move HOZR till $y_s = y_d$-2
Go to G3
Move (x_s, y_s+1, x_d, y_d)
Case-2: Else If (FN is 2nd NBD of OSN)
Step-1 Move VERT till $x_s = x_s$-3
Step-2 If (FN is NBD of (x_d, y_d))
Move HOZR till $y_s = y_d$+1
Move VERB till $x_s = x_s$+3
Move (x_s, y_s-1, x_d, y_d)
Else If (FN is NBD of NBD of (x_d, y_d))
Move HOZR till $y_s = y_d$
Move VERB till $x_s = x_s$+3
Else If (FN is neither NBD nor NBD of NBD of (x_d, y_d))
If (y_d is odd)
Move HOZR till $y_s = y_d$-1
Go to G3
Else If (y_d is even)
Move HOZR till $y_s = y_d$-2
Go to G4
Case-3: Else If (FN is 3rd NBD of OSN)
Step-1 Move (x_s, y_s+1, x_d, y_d)
Go to G1
Step-2 If (FN is NBD of (x_d, y_d) && y_d is even)
Move (x_s, y_s+1, x_d, y_d)
Move VERB till $x_s = x_s$+3
Else If (y_d is odd)
Move HOZR till $y_s = y_d$-1
Go to G3
Else If (y_d is even)
Move HOZR till $y_s = y_d$-2
Go to G4
Case-4: If (FN is on the line of x_s-1, where x_s is OSN && FN is Not w.r.t OSN)
Step-1 Move (x_s-1, y_s, x_d, y_d)
Move (x_s-1, 2y_s-1, x_d, y_d)
Step-2 Go to Step-2 of Case-2 of Sub-Case: E
Case-5: If (FN is not w.r.t OSN && FN is on the line of OSN)
Step-1If (FN y_s is odd)
Go to Case-3 of Sub-Case: E
Else If (FN y_s is even)
Go to Case-3 of Sub-Case: F

Sub-Case: F If (OSN (x_s = [(d*5) + (d-2)] && y_s is even))

Case-1: If (FN is 1stNBD of OSN i. e on x_s-1, where x_s is OSN)

Step-1 Move (x_s, y_s-1, x_d, y_d)
Go to Case-2 of Sub-Case: E
Case-2: Else If (FN is 2nd NBD of OSN i. e on OSN line)
Step-1Go to Step-1 (excluding 1st line) and Step-2 of Case-3

of Sub-Case: E
Case-3: Else If (FN is 3rd NBD of OSN)
Step-1Go to G1
Step-2 If (y_d is odd)
Move HOZR till $y_s = y_d$-1
Go to G3
Else If (y_d is even)
Move HOZR till $y_s = y_d$-2
Go to G4
Case-4: If (FN is on the line x_s-1, where x_s is OSN && FN is Not w.r.t OSN)
Step-1 Move (x_s-1, y_s, x_d, y_d)
Move (x_s-1, 2y_s-1, x_d, y_d)
Step-2 Go to Step-2 of Case-2 of Sub-Case: E
Case-5: If (FN is Not w.r.t OSN && FN is on the line of OSN)
Step-1 If (FN y_s is odd)
Go to Case-3 of Sub-Case: E
Else If (FN y_s is even)
Go to Case-3 of Sub-Case: F
Test Case 1: Let (X_s, Y_s) = (4,2), (X_d, Y_d) = (4,6) and FN = (5,2). Using the algorithm of Horizontal moves, we have the optimal path to reach the destination is:
(4,2) →(5,2) →(4,3) →(4,4) →(5,3) →(4,5) →(4,6). The shortest path length is 6. Now using the above algorithm we have the following fault free path:
(4,2) →(3,2) →(4,3) →(4,4) →(5,3) →(4,5) →(4,6). The shortest path length is 6.
The relative error is 0

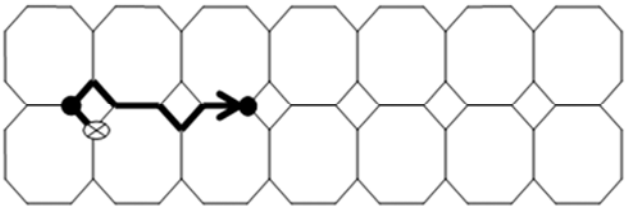


Figure 3. *[1st 7 lines of the Octagon-Cell network of depth 4 are drawn].*

Case-II Fault tolerant routing algorithm for horizontal move for lines m, Where m mod 3 = 1 [Move from right to left, If ($X_s = X_d$ && $Y_s > Y_d$)]

This case is similar to Case-I. So the algorithm is not presented.
Test Case 2: Let (X_s, Y_s) = (4,8), (X_d, Y_d) = (4,3) and FN = (4,4). Using the algorithm of Horizontal moves, we have the optimal path to reach the destination is:
(4,8) →(4,7) →(5,4) →(4,6) →(4,5) →(5, 3) →(4,4)→(4,3). The shortest path length is 7. Now using the above algorithm, the fault free path is:
(4,8) →(4,7) →(5,4) →(4,6) →(4,5) →(5, 3) →(6,3) →(7,4) →(7,3) →(6,2) →(5,2) →(4,3). The shortest path length is 11.So relative error is 0.57

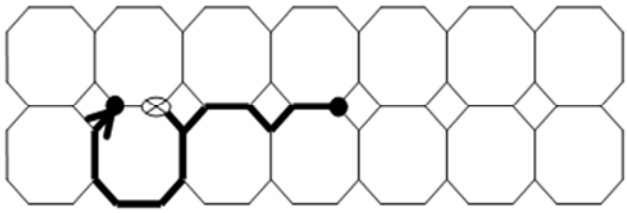


Figure 4. *[1st 7 lines of the Octagon-Cell network of depth 4 are drawn].*

Case-III Fault tolerant routing algorithm for horizontal move for lines m, Where m mod 3 = 2 [Move from left to right, If ($X_s = X_d$ && $Y_s < Y_d$)]

Find the optimal path from OSN to DN
If (path is fault free)
Return true
Else If (FN is DN)
Return dead end
Else Go to the Sub-Cases

Sub-Case: A If (OSN (x_s = 2))

Case-1: If (FN is 1st NBD of OSN, i.e FN is on the line x_s-1, where x_s is OSN || 2nd || 3rd NBD of OSN)
Step-1 Move (x_s+1, y_s, x_d, y_d)
Move HOZR till $y_s = y_d$
Move (x_s-1, y_s, x_d, y_d)
Case-2: Else If (FN is on the line x_s-1, where x_s is OSN && FN is Not w.r.t OSN)
Step-1 If (FN y_s is odd)
Move (x_s, y_s+1, x_d, y_d)
Move (x_s+1, y_s/2+1, x_d, y_d)
Move (x_s+1, y_s, x_d, y_d)
Move HOZR till $y_s = y_d$
Move (x_s-1, y_s, x_d, y_d)
Else If (FN y_s is even)
Go to above If Case (excluding 1st line)
Case-3: Else If (FN is on the line of OSN &&is Not w.r.t OSN)
Step-1 Move (x_s+1, y_s, x_d, y_d)
Move HOZR till $y_s = y_d$
Move (x_s-1, y_s, x_d, y_d)

Sub-Case: B If (OSN (x_s> 2))

Case-1: If (FN is 1st || 2nd NBD of OSN along the line x_s-1, where x_s is OSN)
Step-1 Go to Case-1 of Sub-Case: A
Case-2: Else If (FN is 3rd NBD of OSN i.e on the line of OSN)
Step-1 Move (x_s-1, 2y_s-1, x_d, y_d)
Move (x_s, y_s+1, x_d, y_d)
Move (x_s-1, y_s/2+1, x_d, y_d)
Move (x_s+1, 2y_s-1, x_d, y_d)
Move (x_s, y_s+1, x_d, y_d)
Move (x_s+1, y_s/2+1, x_d, y_d)
Step-2 If ($y_s = y_d$)
Destination Reached
Else Move HOZR till $y_s = y_d$
Case-3: If (FN is on the line x_s-1, where x_s is OSN && FN is Not w.r.t OSN)

Step-1 If (FN y_s is odd)
Go to Case-2 of Sub-Case: A
Else If (FN y_s is even)
Go to Step-1(excluding 1st line) && Step-2 of Case-2 of Sub-Case: A
Case-4: If ((FN is on the line of OSN && FN is Not w.r.t OSN) || FN is 3rd NBD of OSN)
Step-1 Go to Case-2 of Sub-Case: B
Test Case 3: Let (X_s, Y_s) = (2,2), (X_d, Y_d) = (2,6) and FN = (1,4). Using the algorithm of Horizontal moves, the optimal path to reach the destination is:
(2,2) →(1,3) →(1,4) →(2,3) →(1,5) →(1, 6) →(2,4) →(1,7) →(1,8) →(2,5) →(1,9) →(1,10) →(2,6). The shortest path length is 12. Now using the above algorithm, the fault free path is:
(2,2) →(3,2) →(4,3) →(4,4) →(3,3) → (4,5) →(4,6) →(3,4) →(4,7) →(4,8) →(3,5) →(4,9) →(4,10) →(3,6) →(2,6). The shortest path length is 14.
So relative error is 0.17

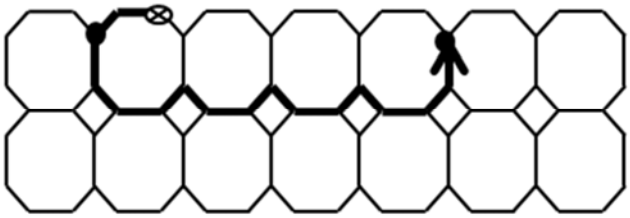


Figure 5. *[1st 7 lines of the Octagon-Cell network of depth 4 are drawn].*

Case-IV Fault tolerant routing algorithm for horizontal move for lines m, Where m mod 3 = 2 [Move from right to left, If ($X_s = X_d$ && $Y_s > Y_d$)]

This case is similar to Case-III. So the algorithm is not presented.
Test Case 4: Let (X_s, Y_s) = (5,6), (X_d, Y_d) = (5,1) and FN = (4,10). Using the algorithm of Horizontal moves, the optimal path to reach the destination is:
(5,6) →(4,10) →(4,9) →(5,5) →(4,8) → (4,7) →(5,4) →(4,6) →(4,5) →(5,3) →(4, 4) →(4,3) →(5,2) →(4,2) →(4,1) →(5,1). The shortest path length is 15. Now using the above algorithm, the fault free path is:
(5,6) →(6,6) →(7,10) →(7,9) →(6,5) → (7,8) →(7,7) →(6,4) →(7,6) →(7,5) →(6, 3) →(7,4) →(7,3) →(6,2) →(7,2) →(7,1) →(6,1) →(5,1). The shortest path length is 17. The relative error is 0.13

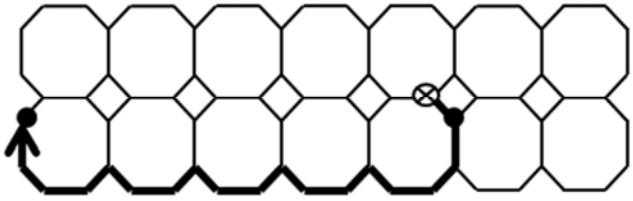


Figure 6. *[1st 7 lines of the Octagon-Cell network of depth 4 are drawn].*

Case-V Fault tolerant routing algorithm for horizontal move for lines m, Where m = 3n [Move from left to right, If ($X_s = X_d$ && $Y_s < Y_d$)]

Find the optimal path from OSN to DN
If (path is fault free)
Return true
Else If (FN is DN)
Return dead end
Else Go to the Sub-Cases

Sub-Case: A If (OSN (x_s < [[(d*5) + (d-2)] -1])
Case-1: If (FN is 1st || 2nd NBD of OSN)
Step-1 Move (x_s-1, y_s, x_d, y_d)
Move HOZR till $y_s = y_d$
Move (x_s+1, y_s, x_d, y_d)
Case-2: Else If (FN is 3rd NBD of OSN || FN is on the line of OSN)
Step-1 Move (x_s+1, 2y_s-1, x_d, y_d)
Move (x_s, y_s+1, x_d, y_d)
Move (x_s+1, y_s/2+1, x_d, y_d)
Move (x_s-1, 2y_s-1, x_d, y_d)
Move (x_s, y_s+1, x_d, y_d)
Move (x_s-1, y_s/2+1, x_d, y_d)
Step-2 If ($y_s = y_d$)
Destination Reached
Else Move HOZR till $y_s = y_d$
Case-3: If (FN is on the line x_s+1, where x_s is OSN && Not w.r.t OSN)
Step-1 If (FN y_s is even)
Move (x_s-1, y_s/2+1, x_d, y_d)
Move (x_s-1, y_s, x_d, y_d)
Move HOZR till $y_s = y_d$
Move (x_s+1, y_s, x_d, y_d)
Else If (FN y_s is odd)
Move (x_s, y_s+1, x_d, y_d)
Move (x_s-1, y_s/2+1, x_d, y_d)
Move (x_s-1, y_s, x_d, y_d)
Move HOZR till $y_s = y_d$
Move (x_s+1, y_s, x_d, y_d)

Sub-Case: B If (OSN (x_s = [[(d*5) + (d-2)] -1])
Case-1: If (FN is 1st || 2nd || 3rd NBD of OSN)
Step-1 Go to Case-1 of Sub-Case: A
Case-2: Else If FN is on the line x_s+1, where x_s is OSN && FN is Not w.r.t OSN)
Step-1 If (FN y_s is odd)
Move (x_s, y_s+1, x_d, y_d)
Move (x_s-1, y_s/2+1, x_d, y_d)
Move (x_s-1, y_s, x_d, y_d)
Move HOZR till $y_s = y_d$
Move (x_s+1, y_s, x_d, y_d)
Else If (FN y_s is even)
Move (x_s-1, y_s/2+1, x_d, y_d)
Move (x_s-1, y_s, x_d, y_d)
Move HOZR till $y_s = y_d$
Move (x_s+1, y_s, x_d, y_d)
Test Case 5: Let (X_s, Y_s) = (3,1), (X_d, Y_d) = (3,3) and FN = (4,3). Using the algorithm of Horizontal moves, the optimal path to reach the destination is:
(3,1) →(4,1) →(4,2) →(3,2) →(4,3) →(4, 4) →(3,3). The shortest path length is 6. Now using the above algorithm the

fault free path is:
(3,1) →(4,1) →(4,2) →(3,2) →(2,2) → (1, 3) →(1,4) →(2,3) →(3,3). The shortest path length is 8. The relative error is 0.33

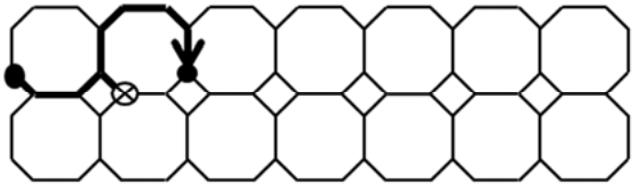


Figure 7. *[1st 7 lines of the Octagon-Cell network of depth 4 are drawn].*

Case-VI Fault tolerant routing algorithm for horizontal move for lines m, Where m = 3n [Move from right to left, If $(X_s = X_d \;\&\&\; Y_s > Y_d)$***]***

This case is similar to Case-V. So the algorithm is not presented.

Test Case 6: Let $(X_s, Y_s) = (6,5)$, $(X_d, Y_d) = (6,1)$ and FN = (7,1). Using the algorithm of Horizontal moves, the optimal path to reach the destination is:
(6,5) →(7,8) →(7,7) →(6,4) →(7,6) →(7, 5) →(6,3) →(7,4) →(7,3) →(6,2) →(7,2) →(7,1) →(6,1). The shortest path length is 12. Now using the above algorithm the fault free path is:
(6,5) →(7,8) →(7,7) →(6,4) →(7,6) →(7, 5) →(6,3) →(7,4) →(7,3) →(6,2) →(5,2) →(4,2)→(4,1) →(5,1) →(6,1). The shortest path length is 14.The relative error is 0.17.

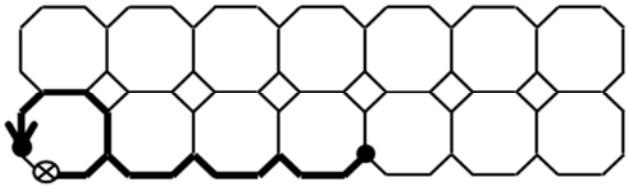


Figure 8. *[1st 7 lines of the Octagon-Cell network of depth 4 has been drawn].*

5. Result and Discussion

The algorithm developed in this paper has been tested for octagon-cell of depth 3 with 80 test cases and also this works on depth greater than 3. There are 40 test cases for the node addresses at the border of octagon-cell and other 40 for inside the network. It is seen that average delay for the nodes lying inside is 1.9 and for the nodes lying at the border is 3.2. The algorithm has already been described in [2] to find optimal path in any direction from source node to destination node in octagon-cell network. If any faulty node or faulty link appears in the optimal path then this algorithm will find the alternative best possible (optimal) path in presence of faulty node. If the destination node is faulty then dead end occurs and the algorithm stops working. From the table it can easily be verified that the optimal path length having faulty node present inside the network is less than the optimal path length of nodes present at the border of the network.

Table 1. *(80 cases with faulty nodes).*

Test Cases with Faulty Node Along the Optimal Path							
Cases	**Source Node**	**Destination Node**	**Optimal Length**	**Faulty Node Along the Optimal Path**	**Optimal Path Length with Faulty Node**	**Maximum Delay**	**Relative Error**
1	(4,2)	(4,6)	6	(5,2)	6	0	0
2	(4,2)	(4,6)	6	(4,3)	8	2	0.33
3	(4,2)	(4,6)	6	(4,4)	8	2	0.33
4	(4,2)	(4,6)	6	(5,3)	6	0	0
5	(4,2)	(4,6)	6	(4,5)	10	4	0.67
6	(13,7)	(13,10)	4	(13,8)	8	4	1
7	(13,7)	(13,10)	4	(14,5)	4	0	0
8	(13,7)	(13,10)	4	(13,9)	8	4	1
9	(2,2)	(2,6)	12	(1,3)	14	2	0.17
10	(2,2)	(2,6)	12	(1,4)	14	2	0.17
11	(2,2)	(2,6)	12	(2,3)	14	2	0.17
12	(2,2)	(2,6)	12	(1,5)	14	2	0.17
13	(2,2)	(2,6)	12	(1,6)	14	2	0.17
14	(2,2)	(2,6)	12	(2,4)	14	2	0.17
15	(2,2)	(2,6)	12	(1,7)	14	2	0.17
16	(2,2)	(2,6)	12	(1,8)	14	2	0.17
17	(2,2)	(2,6)	12	(2,5)	14	2	0.17
18	(2,2)	(2,6)	12	(1,9)	14	2	0.17
19	(2,2)	(2,6)	12	(1,10)	14	2	0.17
20	(2,2)	(2,6)	12	(2,6)	--------	Dead End	
21	(5,6)	(5,1)	15	(4,10)	17	2	0.13
22	(5,6)	(5,1)	15	(4,9)	17	2	0.13
23	(5,6)	(5,1)	15	(5,5)	17	2	0.13
24	(5,6)	(5,1)	15	(4,8)	17	2	0.13
25	(5,6)	(5,1)	15	(4,7)	17	2	0.13
26	(5,6)	(5,1)	15	(5,4)	17	2	0.13

Test Cases with Faulty Node Along the Optimal Path							
Cases	**Source Node**	**Destination Node**	**Optimal Length**	**Faulty Node Along the Optimal Path**	**Optimal Path Length with Faulty Node**	**Maximum Delay**	**Relative Error**
27	(5,6)	(5,1)	15	(4,6)	17	2	0.13
28	(5,6)	(5,1)	15	(4,5)	17	2	0.13
29	(5,6)	(5,1)	15	(5,3)	17	2	0.13
30	(5,6)	(5,1)	15	(4,4)	17	2	0.13
31	(5,6)	(5,1)	15	(4,3)	17	2	0.13
32	(5,6)	(5,1)	15	(5,2)	17	2	0.13
33	(5,6)	(5,1)	15	(4,2)	17	2	0.13
34	(5,6)	(5,1)	15	(4,1)	17	2	0.13
35	(5,6)	(5,1)	15	(5,1)	---------	Dead End	
36	(3,1)	(3,3)	6	(4,1)	8	2	0.33
37	(3,1)	(3,3)	6	(4,2)	8	2	0.33
38	(3,1)	(3,3)	6	(3,2)	8	2	0.33
39	(3,1)	(3,3)	6	(4,3)	8	2	0.33
40	(3,1)	(3,3)	6	(4,4)	8	2	0.33
	Average					1.9	0.22
	Optimal Routing of Nodes at the Border of the Octagon-Cell						
1	(1,3)	(1,8)	7	(1,4)	11	4	0.57
2	(1,3)	(1,8)	7	(2,3)	11	4	0.57
3	(1,3)	(1,8)	7	(1,7)	11	4	0.57
4	(1,3)	(1,8)	7	(1,5)	9	2	0.29
5	(1,3)	(1,8)	7	(2,4)	11	4	0.57
6	(16,1)	(16,5)	6	(16,2)	10	4	0.67
7	(16,1)	(16,5)	6	(15,2)	10	4	0.67
8	(16,1)	(16,5)	6	(16,3)	8	2	0.33
9	(16,1)	(16,5)	6	(16,4)	8	2	0.33
10	(16,1)	(16,5)	6	(15,3)	12	6	1
11	(16,10)	(16,1)	13	(16,9)	17	4	0.31
12	(16,10)	(16,1)	13	(15,5)	17	4	0.31
13	(16,10)	(16,1)	13	(16,8)	15	2	0.15
14	(16,10)	(16,1)	13	(16,7)	15	2	0.15
15	(16,10)	(16,1)	13	(15,4)	15	2	0.15
16	(16,10)	(16,1)	13	(16,6)	15	2	0.15
17	(16,9)	(16,4)	8	(15,5)	14	6	0.75
18	(16,9)	(16,4)	8	(16,8)	10	2	0.25
19	(16,9)	(16,4)	8	(16,7)	10	2	0.25
20	(16,9)	(16,4)	8	(15,4)	10	2	0.25
21	(16,9)	(16,4)	8	(16,6)	10	2	0.25
22	(16,9)	(16,4)	8	(15,3)	14	6	0.75
23	(16,7)	(16,3)	6	(15,4)	12	6	1
24	(16,7)	(16,3)	6	(16,6)	8	2	0.33
25	(16,7)	(16,3)	6	(16,5)	8	2	0.33
26	(16,7)	(16,3)	6	(15,3)	10	4	0.67
27	(16,7)	(16,3)	6	(16,4)	10	4	0.67
28	(16,6)	(16,2)	6	(16,5)	10	4	0.67
29	(16,6)	(16,2)	6	(15,3)	10	4	0.67
30	(16,6)	(16,2)	6	(16,4)	8	2	0.33
31	(16,6)	(16,2)	6	(16,3)	8	2	0.33
32	(16,6)	(16,2)	6	(15,2)	12	6	1
33	(1,9)	(1,1)	12	(2,5)	18	6	0.50
34	(1,9)	(1,1)	12	(1,8)	14	2	0.17
35	(1,9)	(1,1)	12	(1,7)	14	2	0.17
36	(1,9)	(1,1)	12	(2,4)	14	2	0.17
37	(1,9)	(1,1)	12	(1,6)	14	2	0.17
38	(1,9)	(1,1)	12	(1,5)	14	2	0.17
39	(1,9)	(1,1)	12	(2,3)	14	2	0.17
40	(1,9)	(1,1)	12	(1,4)	14	2	0.17
	Average= 3.2						0.42
	Average Delay for 80 Samples of Data = 2.55						

In the above table all possible faulty nodes have been taken along the optimal path between the source and destination nodes. For each case the relative error has been calculated. The average delays for the nodes inside the network and at the border are 1.9 and 3.2 respectively. Relative Error = (Fault tolerant optimal path length – Optimal path length) / Optimal path length. The average relative errors for the nodes inside the network and at the border are 0.22 and 0.42 respectively.

6. Conclusion

This research paper describes a simpler and an efficient fault tolerant optimal routing algorithm for horizontal moving signals in octagon-cell interconnected networks. Here the cases of depth 3 have been taken. The delay is unchanged if the depth is increased.

OCN is assumed a type of integrated circuit with an array of hundreds or thousands of central processing units (CPUs) and random-access memory banks. So this fault tolerant scheme can be useful in massively parallel system. In future we will develop the optimal fault tolerant scheme for the signals moving in all directions in the network.

References

[1] S. Mohanty and P. K. Behera. “Fault Tolerant Routing Algorithm in Octagon-Cell Interconnected Network for Horizontal Moving Messages”. International Journal of Computer Networks & Communications. Vol. 9, No. 1, January 2017, pp. 23-38.

[2] S. Mohanty and P. K. Behera. “Optimal Routing Algorithm in a Octagon-Cell Network”. International Journal of Advanced Research in Computer Science. Vol. 2, No. 5, September-October 2011, pp. 625-637.

[3] A. Sharieh, M. Qatawneh, W. Almobaideen and A. Sleit. “Hex-Cell: Modeling, Topological Properties and Routing Algorithm”. European Journal of Scientific Research, Vol. 22, No. 2, April 2008, pp. 457-468.

[4] N. G. Kini, M. S. Kumar and Mruthyunjaya H.S. “An Optimal Data Routing Scheme for Mesh Embedded Hypercube Interconnection network with Multiple Faulty Nodes”. International Journal of Computer Science and Engineering. 3:1, January 2009, pp. 26-30.

[5] A. Louri and H. Sung, “An Optical Multi-Mesh Hypercube: A Scalable Optical interconnection Network for Massively Parallel Computing”. Journal of Lightwave Technology. Vol. 12, No. 4, April 1994, pp. 704-716.

[6] T. C. Lee and J. P.Hayes. “A Fault-Tolerant Communication Scheme for Hypercube Computers”. IEEE Transactions on Computers. Vol. 41, No. 10, October 1992, pp. 1242-1256.

[7] S. Sharma and P. K. Bansal. “A New Fault Tolerant Multistage Interconnection Networks”. IEEE TENCON’ 02, Vol. 1, October 2002, pp. 347-350.

[8] S.-C. Wang and S.-Y. Kuo. “Fault Tolerance in Hyperbus and Hypercube Multiprocessors Using Partitioning Scheme”. IEEE International Conference on Parallel and Distributed Systems. December 1994, pp. 340-347.

[9] D. Nassimi and S. Sahni. “An Optimal Routing Algorithm for Mesh Connected Parallel Computers”. Journal of the ACM, Vol.27, January 1980, pp. 6-29.

[10] J. P. Shen. “Fault Tolerance Analysis of Several Interconnection Networks”. In Proceedings of International Conference on Parallel Processing. 1982, pp. 102-112.

[11] J. Bruck, R. Cypher and D. Soroker. “Tolerating Faults in Hypercubes Using Subcube Partitioning”. IEEE Trasactions on Computers. Vol. 41, No. 5, May 1992, pp. 599-605.

[12] K. V. Arya and R. K. Ghosh. “Designing a New Class of Fault Tolerant Multistage Interconneconnection Networks”. Journal of Interconneconnection Network. Vol. 6, No. 4, December 2005, pp. 361-382.

[13] K.-H. Chen and G.-M. Chiu. “Fault-Tolerant Routing Algorithm for Meshes without Using Virtual Channels”. Journal of Information Science and Engineering. 1998, Vol. 14, pp. 765-783.

[14] J. Chen, I. A. Kanj and G. Wang. “Hypercube Network Fault Tolerance: A Probabilistic Approach”. In Proceedings of the IEEE International Conference on Parallel Processing. August 2002.

[15] M. J. Serrano and B. Parhami. “Optimal Architectures and Algorithms for Mesh-Connected Parallel Computers with Separable Row/Column Buses”. IEEE Trasactions on Parallel and Distributed Systems. Vol. 4, No. 10, October 1993, pp. 1073-1080.

[16] M. Qatawneh, B. Hamed, W. AlMobaideen, A. Sleit, A. Qudat, W. Qutechat and R. Al-Soub. “FTRH: Fault Tolerance Routing Algorithm for Hex-Cell Networks”. International Journal of Computer Science and Network Security. Vol. 9, No. 12, December 2009, pp. 268-274.

[17] K. Day, S. Harous and A. Al-Ayyoub. “A Fault Tolerant Routing Scheme for Hypercubes”. Telecommunication Systems. Vol. 13, No. 1, May 2000, pp. 29-44.

[18] C. Decayeux and D. Seme. “3D Hexagonal Network: Modeling, Topological Properties, Addressing Scheme and Optimal Routing Algorithm”. IEEE Trasactions on parallel and Distributed Systems. Vol. 16, No. 9, August 2005, pp. 875-884.

[19] Q. Mohammad. “Adaptive Fault Tolerant Routing Algorithm for Tree-Hypercube Multicomputer”. Journal of Computer Science. Vol. 2, No. 2, February 2006, pp. 124-126.

[20] S. Chalasani and R. V. Boppana. “Fault-Tolerant Wormhole Routing in Tori”. International Conference on Supercomputing. July 1994, pp. 146-155.

[21] M. A. Omari and M. Mahafzah. “Fault-Tolerant Routing in hypercubes using masked interval Routing Scheme”. In Proceedings of the (1999) ACM symposium on Applied Computing. February-March 1999, pp. 481-485.

[22] J. Zhou and F. C. M. Lau. “Multi-Phase Minimal Fault-Tolerant Wormwhole Routing in Meshes”. Parallel Computing. Vol. 30, No. 3, March 2004, pp. 423-442.

[23] J.-H. Youn, B. Bose and S. Park. "Fault-Tolerant Routing Algorithm in Meshes with solid faults". Journal of Supercomputing. 37, 2006, pp. 161-177.

[24] F. Cristian, B. Dancey and J. Dehn. "Fault-Tolerance in Air Traffic Control Systems". Trasaction on Computer Systems (TOCS), Vol. 14, No. 3, August 1996, pp. 256-286.

[25] D. Wang. "A Low Cost Fault-Tolerant Structure for Hypercube". Journal of Supercomputing. Vol. 20, No. 3, November 2001, pp. 203-216.

[26] D. Gordon, I. Koren and G. M. Silberman. "Reconstructing Hexagonal Arrays of Processors in the Presence of Faults". Journal of VLSI and Computer Systems. Vol. 2, 1987, pp. 23-35.

[27] M. Rezazad and H. S. Azad. "A Routing Algorithm for Star Interconnection Network in the Presence of Faults". The CSI Journal on Computer Science and Engineering. Vol. 1, No. 4(b), Winter 2003, pp. 11-18.

[28] S. Mohanty and P. K. Behera. "Comparison of Octagon-Cell Network with other Interconnected Network Topologies and its Applications". International Journal of Computer Engineering and Applications. Vol. 7, No. 2, August 2014, pp. 201-211.