© September 2021| IJIRT | Volume 8 Issue 4 | ISSN: 2349-6002

Study of Patterns of Connectivity in Hypercube Interconnection Network Through Number Theoretic Notation

Rakesh Kumar Katare[1], Ritu Mishra[2], Neha Singh[3], Sunil Tiwari [4]
Department of Computer Science, A.P.S. University , Rewa (M.P.)

***Abstract* - This paper aims at revealing connectivity properties of the n- bit hypercube interconnection network by using the binary gray coding indexing scheme to label the processing nodes of the hypercube and then using the complement gray code strategy to disclose some useful connectivity patterns in the hypercube. Here we use the beauty of binary number theoretic notation and un-weighted gray code to study the patterns.**

***Index Terms* - Hypercube, Gray code, 2-bit complement matrix, Processor adjacency matrix.**

INTRODUCTION

This paper presents the study of connectivity patterns in the hypercube interconnection topology and aims at illustrating the topological properties of the hypercube through our studies. We will use here the binary gray coding as an indexing scheme for addressing the processing nodes of the hypercube because of its similarity with the hamming distance property of hypercube. In the hypercube two adjacent nodes (nodes connected directly through the link) has difference of just one bit in their binary representation and in the gray (non- weighted) code also the two adjacent gray codes just differ by one bit. Our study aims at revealing the connectivity alliances for the one dimensional, two dimensional, three dimensional and four-dimensional hypercube and thus establishing the validity of found relationships for the n-bit hypercube. In 2009 Katare, R.K and Chaudhari, N.S.[1] has studied the topological properties of hypercube by mapping it into the sparse matrix model. In the next year 2010 Katare et.al.[2] developed P-RAM algorithms and data structures to implement sparse matrix in parallel computing for shared memory model. Later in 2012 Katare, R.K et .al.[3] studied the link utilization of hypercube interconnection network by using relation matrix. In 2015 Tiwari, Sunil, Katare, R.K.[4] studied the fabric of architecture by using structural pattern and relation so as to reduce the complexity of the interconnection network by the use of ploylog and matrix notation. In successive year 2016 Tiwari, Sunil et.al.[5] studied the geometrical structure of parallel interconnection network through mapping of processors method and logical operations. While n the year 2018 Bharadwaj, Manish and Katare R.K.[6] used logical operations to study the connectivity properties in the Parallel and distributed systems. In the same year Kumari, Mamta et.al.[7] used connectivity matrix of interconnection network to study the structural relationship complexity in the interconnection network.

BACKGROUND

A hypercube can be defined by increasing the numbers of dimensions of a shape [8].
Zero dimension hypercube: A point in a plane is hypercube of dimension zero.
One dimension hypercube: If one moves this point one unit length, it will sweep out a line segment which is a unit hypercube of dimension one.
Two dimension hypercube: If one moves this line segment its length in a perpendicular direction from itself; it sweeps out a 2-d square hypercube.
Three dimension hypercube: If one moves the square one unit length in the direction perpendicular to the plane it lies on, it will generate a 3- dimensional cube.
Four dimension hypercube: If one moves the cube one unit length into the fourth dimension, it generates a 4-dimensional unit hypercube (a unit 4-bit).

This can be generalized to any number of dimensions, by using sweeping method of Minkowsi sum[1]**.**

We limit our study from one dimensional hypercube to the 4-dimensional hypercube and thus establishing our results for the n-dimensional hypercube.

According to the property of hypercube we know that the positional hamming distance between each adjacent node pair is one or we can say that each processing node in the interconnection network will be adjacent to each other if and only if the binary representation of the nodes differs by just one bit. Thus, we use the binary coding to address the nodes of the hypercube.

For the one-dimensional hypercube we have two processing nodes and thus one-bit binary representation is sufficient to address the nodes. When we use one bit representation method then maximum possible representation will be $2^1 = 2$ i.e. 0 and 1.

For the two-dimensional hypercube we have four processing nodes and thus two-bit binary representation is sufficient to address these nodes. When we use two-bit representation method then maximum possible representation will be $2^2 = 4$ i.e. 00,01,10 and 11.So according to the positional hamming distance , square hypercube will be as shown in the figure 1 below.

For the three-dimensional hypercube we have eight processing nodes and thus three-bit binary representation is sufficient to address these nodes because three-bit representation method gives maximum possible representation will be $2^3 = 8$ i.e. 000,001,010, 011,100,101,110 and 111.So according to the positional hamming distance , three dimensional hypercube will be as shown in the figure 1 below.

Similarly, for the four-dimensional hypercube we have sixteen processing nodes and thus four-bit binary representation is sufficient to address these nodes because combination of four bits gives maximum possible representation will be $2^4 = 16$ i.e. 0000,0000, 0010, 0011,0100, 0101,0110,0111,1000,1001,1010, 1011,1100,1101,1110 and 1111. So according to the positional hamming distance, four bit 4-bit hypercube will be as shown in the figure 1 below.

We use the gray code of 1 bit,2-bit,3-bit and 4 bit to index the processors in unit hypercube, square hypercube,3-D hypercube and 4-bit hypercube respectively because gray code follows the distance of 1- bit between two successive representations, thus preserving the property of hypercube.

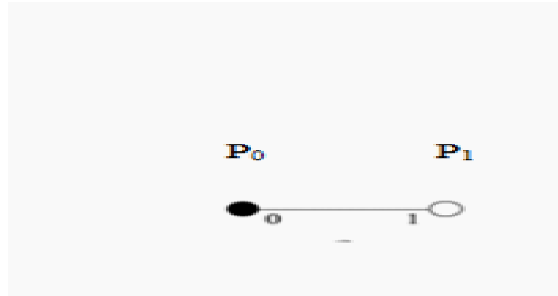

Figure 1(a) 1- Bit Hypercube nodes indexed with gray code

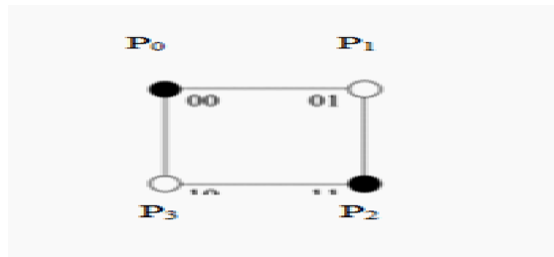

Figure 1(b) 2- bit square Hypercube nodes indexed with gray code

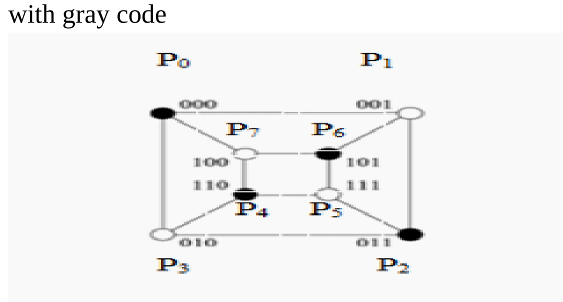

Figure 1(c) 3-bit 3-D Hypercube Hypercube nodes indexed with gray code

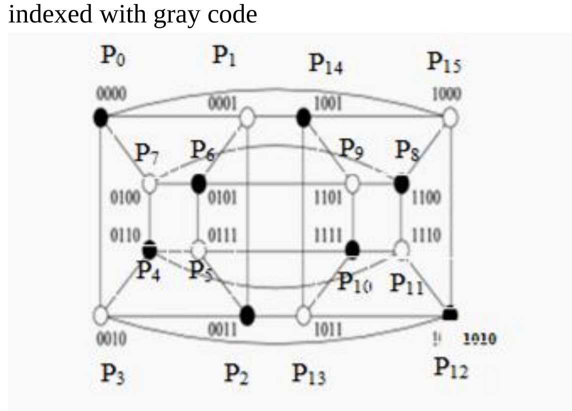

Figure 1(d) 4- bit Hypercube nodes indexed with gray code

Unit hypercube consisting of just two processing nodes needs just two binary indices. Say we have processors P_0 and P_1 we can index them as 0 and 1 through 1- bit gray code, in the same way we map the indices onto processing nodes in each type of hypercube which is summarized below in table no. 1,2,3 and 4.

S.No.	Processing node	Gray coding for the processing node
1	P_0	0
2	P_1	1

Table No.1 Unit Hypercube (2 nodes)) indexed through 1- bit gray coding

S.No.	Processing node	Gray coding for the processing node
1	P_0	00
2	P_1	01
3	P_2	11
4	P_3	10

Table No.2 Square Hypercube (4 nodes) indexed through 2- bit gray coding

S.No.	Processing node	Gray coding for the processing node
1	P_0	000
2	P_1	001
3	P_2	011
4	P_3	010
5	P_4	110
6	P_5	111
7	P_6	101
8	P_7	100

Table No.3 3-D Hypercube (8 nodes) indexed through 3- bit gray coding

S.no	Processing node	Gray coding for the processing node
1	P_0	0000
2	P_1	0001
3	P_2	0011
4	P_3	0010
5	P_4	0110
6	P_5	0111
7	P_6	0101
8	P_7	0100
9	P_8	1100
10	P_9	1101
11	P_{10}	1111
12	P_{11}	1110
13	P_{12}	1010
14	P_{13}	1011
15	P_{14}	1001
16	P_{15}	1000

Table No.4 4-bit hypercube (16 nodes) indexed through 4- bit gray coding

Lemma 1: Two nodes whose binary (gray) representation is complement to each other are never adjacent in hypercube.

Proof: To prove this lemma we need to identify the nodes whose binary representation are complement to each other

For this purpose we can modify the table no.2, 3 and 4 to recognize the processors those which have complementary binary indexing (binary gray code representations). After that we can use figure 2(a) and 2(b) whether they are diagonal to each other or not either in the same face of the cube or in the adjacent face of the hypercube

S. No	Processing node	Gray coding for the processing node	Complementary gray coding	Processing node with complement gray code
1	P_0	00	11	P_2
2	P_1	01	10	P_3
3	P_2	11	00	P_0
4	P_3	10	01	P_1

Table No.5 Square Hypercube (4 nodes) indexed through 2- bit gray coding

S. No.	Processing node	Gray coding for the processing node	Complementary gray coding	Processing node with complement gray code
1	P_0	000	111	P_5
2	P_1	001	110	P_4
3	P_2	011	100	P_7
4	P_3	010	101	P_6
5	P_4	110	001	P_1
6	P_5	111	000	P_0
7	P_6	101	010	P_3
8	P_7	100	011	P_2

Table No.6 3-D Hypercube (8 nodes) indexed through 3- bit gray coding

S. No.	Processing node	Gray coding for the processing node	Complementary gray coding	Processing node with complement gray code
1	P_0	0000	1111	P_{10}
2	P_1	0001	1110	P_{11}
3	P_2	0011	1100	P_8
4	P_3	0010	1101	P_9
5	P_4	0110	1001	P_{14}

6	P_5	0111	1000	P_{15}
7	P_6	0101	1010	P_{12}
8	P_7	0100	1011	P_{13}
9	P_8	1100	0011	P_2
10	P_9	1101	0010	P_3
11	P_{10}	1111	0000	P_0
12	P_{11}	1110	0001	P_1
13	P_{12}	1010	0101	P_6
14	P_{13}	1011	0100	P_7
15	P_{14}	1001	0110	P_4
16	P_{14}	1000	0111	P_5

Table No.7 4- bit hypercube (16 nodes) indexed through 4- bit gray coding

To study the connectivity patterns in the hypercube we know from its definition that only those processing nodes are adjacent to each other whose binary representations differ by just one bit ,or we can say only those two nodes will be directly connected through an edge who are hamming distance of 1 from each other. By this definition we can create an adjacency matrix of the processors to reveal some hidden properties of hypercube.

With the help of figure 1 let us create the adjacency matrix of square hypercube, 3-D hypercube and 4-bit hypercube.

	P_0	P_1	P_2	P_3
P_0		1	0	1
P_1	1		1	0
P_2	0	1		1
P_3	1	0	1	

Table No. 8: Square hypercube Processor adjacency matrix

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7
P_0		1		1		0		1
P_1	1		1		0		1	
P_2		1		1		1		0
P_3	1		1		1		0	
P_4		0		1		1		1
P_5	0		1		1		1	
P_6		1		0		1		1
P_7	1		0		1		1	

Table No. 9: 3-D hypercube Processor adjacency matrix

	P_0	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}	P_{15}
P_0		1		1				1			0					1
P_1	1		1				1					0			1	
P_2		1		1		1			0					1		
P_3	1		1		1					0			1			
P_4				1		1		1				1			0	
P_5			1		1		1				1					0
P_6		1				1		1		1			0			
P_7	1				1		1		1					0		
P_8			0					1		1		1				1
P_9				0			1		1		1				1	
P_{10}	0					1				1		1		1		
P_{11}		0			1				1		1		1			
P_{12}				1			0					1		1		1
P_{13}			1					0			1		1		1	
P_{14}		1			0					1				1		1
P_{15}	1					0			1				1		1	

Table No. 10: 4-bit Processor adjacency matrix

In the above matrices "1" shows adjacency relationship between the nodes and "0" shows that nodes are not adjacent.

In the square hypercube from table no. 5 as we know that,

P_0 and P_2, P_1 and P_3 are complementary to each other with respect to the gray code representation. From the adjacency matrix of the square hypercube table no. 8 we can clearly observe that neither P_0 and P_2 nor P_1 and P_3 are adjacent to each other because in the adjacency matrix 0^{th} row, 2^{nd} column as well as 2^{nd} row, 0^{th} column contains zero showing no adjacency relation between P_0 and P_2. Similar fact we can conclude about the node P_1 and P3 because entry in the 1^{st} row, 3^{rd} column and 3^{rd} row 1^{st} column corresponds to zero.

In the same way for the 3-D hypercube (table no. 6) we have derived that processing node P_0 and P_5, P_1 and

P_4, P_2 and P_7 and P_3 and P_6 are complementary to each other with respect to the gray code.

From the adjacency matrix of the 3-D hypercube (table no. 9) we see that the cell corresponding to the 0^{th} row, 5^{th} column contains 0 which shows that P0 and P5 are not adjacent to each other.

Cell corresponding the1^{st} row, 4^{th} column and 4^{th} row, 1^{st} column has 0 showing no adjacency relation.

Cells that correspond to row 2nd row and 7th column as well as 7^{th} row, 2^{nd} column consist of 0 that shows "not adjacent" relationship among P_2 and P_7 node.

Same relationship can be implied for the node P_3 and P_6 because 3^{rd} row ,6^{th} column and 6^{th} row, 3^{rd} column shows 0 that means P_3 and P_6 are not adjacent to each other.

So we can say that in the 3-D hypercube, nodes whose binary representation is complement to each other are never adjacent in the architecture which means that there is no direct link between them.

We can prove this for the 4-bit hypercube from table no. 7 and table no.10.

Since this statement holds true for the 2-bit 3-bit and 4- bit hypercube therefore we can say that this will be true for the n- bit hypercube also where n is any arbitrary positive integer.

Lemma 2: Diagonals are two bit complements in a hypercube.

Proof: We will try to prove this statement for square hypercube and 3-D hypercube and extend its validity for the n-bit.

For proving this we have to identify the nodes which are diagonally located in the plane to each other.

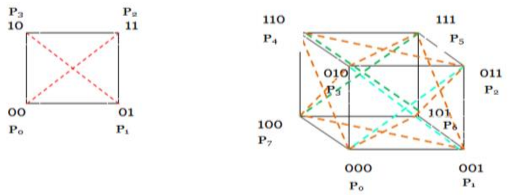

Figure 2(a) diagonally located nodes in square hypercube 2(b) diagonally located nodes in 3-D hypercube

In figure2 (a) diagonally located nodes are connected through red dotted lines in a square hypercube.

In figure 2(b) diagonally located nodes are classified or depicted using three colors. Cyan color dotted lines show diagonally located nodes in one dimension and green color dotted lines shows diagonal nodes in the other dimension. While the orange dotted lines shows the inter-dimension diagonally located nodes, in 3-D hypercube.

From figure 2(a), 2(b) and table no.2 and 3 we can list out the diagonally connected nodes of each processing node.

Processing node	Gray coding of the node	Diagonally located node	Gray coding of the Diagonally located node
P_0	00	P_2	11
P_1	01	P_3	10
P_2	11	P_0	00
P_3	10	P_1	01

Table No. 11 Diagonally located nodes in square hypercube

Processing node	Gray coding for the processing node	Diagonally located node in same dimension	Gray coding of the Diagonally located node	Diagonally located node in other Face	Gray coding of diagonally located node in other Face
P_0	000	P_2	011	P_4	110
				P_6	101
P_1	001	P_3	010	P_5	111
				P_7	100
P_2	011	P_0	000	P_5	111
				P_7	100
P_3	010	P_1	001	P_4	110
				P_6	101
P_4	110	P_6	101	P_0	000
				P_2	011
P_5	111	P_7	100	P_1	001
				P_3	010
P_6	101	P_4	110	P_0	000
				P_2	011
P_7	100	P_5	111	P_1	001
				P_3	010

Table No. 12 Diagonally located nodes in 3-D hypercube

Now let us construct a two bit complement matrix for square hypercube and 3-D hypercube

	00	01	10	11
00				1
01			1	
10		1		
11	1			

	000	001	010	011	100	101	110	111
000				1		1	1	
001			1		1			1
010		1			1			1
011	1					1	1	
100		1	1					1
101	1			1			1	
110	1			1		1		
111		1	1		1			

Table No. 13 Two bit Complement matrix of Square hypercube and 3-D hypercube

By using the 2- bit complement matrix of square hypercube and table no. 11 we can clearly observe that Processing node P_0 with gray code label 00 has two bit complement as 11 which is also gray code index label of processing node P_3, the diagonal node of P_0. In the Same way Processing node P_1 with gray code label 01 has two bit complement as 10 which is also gray code index label of processing node P_4, the diagonal node of P_1.Reverse can also be implied.

Through the 2- bit complement matrix of 3-D hypercube and table no. 12 we can derive the following conclusions:

Processing node P_0 with gray code label 000 has two bit complement as 011 which is also gray code index label of processing node P_3, the diagonal node of P_0 in same face. Also 101 and 110 are two bit compliments of 000, which are also gray code index label of processing node P_6 and P_4 respectively. P_6 and P_4 are diagonal nodes of the P_0 processing node in other adjacent faces of the cube.

For processing node P_1 (001) two bit complement gray code is 010,100 and 111 where 010 labeled node $P_{3\ is}$ diagonal nodes of P_1 in the same face whereas 100 labeled P_7 and 111 labeled P_5 are diagonal nodes of P_1 in the adjacent faces.

The node P_2(011) has 2- bit complement gray code as 000,101 and 110.The node P_0 labeled 000 is diagonal node of P_2 in the same face of the cube and P_4 labeled 101, 110 labeled P_6 are diagonal nodes of P_2 in the other adjacent faces of the cube.

Node P_3 labeled 010 is having two bit complements labeled node 001(P_1) is diagonal node in the same face, 100(P_7), 111(P_5) are diagonal nodes in the adjacent faces.

In the same way we can find this fact true for the other nodes P_4- P_7 also.

Therefore, we can say that this lemma holds true for hypercube labeled with 2-bit and 3- bit gray code. Hence, we can say that it can be also proven true for hypercube labeled with n-bit gray code too.

CONCLUSION

In our paper to study the connectivity patterns of hypercube interconnection topology we have allotted the gray codes to number the different processing nodes of the interconnection network. We have tried to disclose some of the important connectivity patterns for the one dimensional, two dimensional, three dimensional and four-dimensional hypercube by using 2-bit, 3-bit and 4- bit binary gray code indexing and thus establishing the validity of our study for the N-dimensional hypercube that will use n-bit gray code to index the processors through lemmas. We have derived two important connectivity properties here:

1. Two nodes whose binary representation is n- bit complement to each other are never adjacent in hypercube i.e. in n-bit hypercube, the n-bit gray code indexed processing node and its respective n- bit complement gray code indexed processing nodes will never appear adjacent to each other in the hypercube which means they will not have an (connecting link) edge that will directly connect them.
2. Diagonally located nodes in the hypercube are 2-bit complements to each other, either they are in same face of the hypercube or adjacent face of the hypercube, that means in the n-bit gray indexed code indexed hypercube for any arbitrary processing node we can observe that its diagonally located node either in the same face of the hypercube or in the other adjacent face, both have the gray coding which is two bit complement of each other.

REFERENCES

[1] Katare, R.K and Chaudhari, N.S. "STUDY OF TOPOLOGICAL PROPERTY OF INTERCONNECTION NETWORKS AND ITS MAPPING TO SPARSE MATRIX MODEL." International Journal of Computer Science and

Applications, Ó2009 Technomathematics Research Foundation Vol. 6, No. 1, pp. 26 – 39

[2] Katare. R.K, Rai,P.K., Chaudhari N.S.” P-RAM ALGORITHMS AND DATA STRUCTURES FOR SPARSE LINEAR SYSTEMS”. Journal of Engineering, Science and Management Education/Vol. 2, 2010/67-72

[3] Katare R.K., Chaudhari, N.S.” Study of Link Utilization of Perfect Difference Network and Hypercube”.

[4] Tiwari,Sunil, Katare, R.K, A Study of Fabric of Architecture Using Structural Pattern and Relation, IJLTEMAS, Volume IV, Issue IX, 2015.

[5] Tiwari,Sunil et.al. “Study of Geometrical structure of Perfect Difference Network (PDN) “International Journal of Advanced Research in Computer and Communication Engineering Vol. 5, Issue 3, March 2016

[6] Bharadwaj, Manish and Katare R.K.” Study of connectivity using AND and XOR logical operation on Perfect Difference Network for Parallel and Distributed Systems” International Journal of Electronics, Electrical and Computational System IJEECS ISSN 2348-117X Volume 7, Issue 3 March 2018

[7] Kumari,Mamta et.al. “Study of Complexity of Structural Relation for Interconnection Network” International Journal of Computer Applications (0975 – 8887) Volume 180 – No.22, February 2018

[8] https://en.m.wikipedia.org/wiki/Minkowsi_addition.html