iJRASET
International Journal For Research in Applied Science and Engineering Technology

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

**Volume:** 9 **Issue:** IX **Month of publication:** September 2021

**DOI:** https://doi.org/10.22214/ijraset.2021.38150

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429*
*Volume 9 Issue IX Sep 2021- Available at www.ijraset.com*

# Number Theoretic Model of Concurrent Access in Hypercube Interconnection Network

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

***Abstract:*** ***This paper proposes a mathematical logical model to use shared communication links in the hypercube interconnection network. To achieve our objective we have used the orthogonal property of binary coding used to label the processing nodes of the hypercube and applied binary coding based multiplexing technique and the XOR logic operation to separate the individual processing nodes data from the multiplexed signal.***
***Keywords: Gray codes, Hypercube, 1's complement logic, GIG codes, bipolar codes, orthogonal codes, Normalized inner product, Parallel computing System.***

## I. INTRODUCTION

In this paper we will try to develop a model of data communication between the processing nodes of hypercube interconnection network through a shared communication path. Our strategy is to convert the n- bit cyclic gray code (label of processing nodes) into the 2n bit orthogonal code. For this purpose we have used algorithm given by K. Usha and K. Jaya Sankar[1], and then we will use this generated 2n bit orthogonal binary code is used as data coding of each node. To make our idea work we assumed that whenever any processing nodes want to share data corresponding to bit "1" then it has to send its GIG code label or instead it has to send the 1's complement of its GIG code label to share the data corresponding to bit "0". Furthermore we will be using the bipolar representation to convert the m=2n bit GIG code into bipolar GIG code where "0" is coded as "-1" and "1" is coded as "+1". Then to separate the data bit sent from particular processing node from the multiplexed data signals available in the shared communication link from different processing nodes, which shares their data concurrently through the normalized inner product of multiplexed bipolar sequence and the bipolar GIG code sequence of the respective node and further divide the result with the total length of the bipolar GIG code sequence which is m=2n (where n is binary digits used to label the processing nodes of the hypercube).If result comes positive it shows that the processing node has sent its original bipolar GIG code sequence in a wish to send the data bit "1" while if result comes negative it shows that the processing node has sent its complement bipolar GIG code sequence in a wish to send the data bit "0".

## II. BACKGROUND

According to Saad and Schultz (1988) [2]. A hypercube is a multidimensional mesh of nodes with exactly two nodes in each dimension. A n d-dimensional hypercube consists of K nodes, where $K=2^n$ having following topological properties

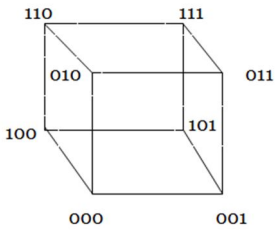
*1)* A hypercube has n special dimensions; where n can be any positive integer (including zero).
*2)* A hypercube has 2n vertices.
*3)* There are n connections (lines) that meet at each vertex of a hypercube.
*4)* All connections at a hypercube vertex meet at right angles with respect to each other [3], [4].
*5)* The Hypercube can be constructed recursively from lower dimensional cubes
*6)* An architecture where the degree and diameter of the graph is the same then they will achieve a good balance between, the communication speed and the complexity of the topology network.
*7)* An n-cube graph is an undirected graph of $k=2^n$ vertices labeled from 0 to $2^n$-1 and such that there is an edge between any two vertices if and only if the binary representations of their labels differ by one and only one bit [6].
*8)* Any two adjacent nodes A and B of an N cube are such that the nodes adjacent to A and those adjacent to B are connected in one-to-one fashion.
*9)* There are *n* different ways of tearing an n-cube, i.e., of splitting it into two (n - 1) sub cubes so that their respective vertices are connected in a one-to-one way [7].
*10)* There are n! $2^n$ different ways in which the 211nodes of an n-cube can be numbered.
*11)* A graph G=(V,E) is an n-cube if and only if

*a)* V has 211vertices;
*b)* Every vertex has degree n;
*c)* G is connected;
*d)* Any two adjacent nodes A and B are such that the nodes adjacent to A and those adjacent to B are linked one-to-one fashion.

From the study of hypercube properties we also observed that study of topological properties of hypercube graph is that the sequence numbering used to number the processing nodes corresponds to the gray coding scheme. Gray Code is a variable weighted code and is cyclic. This means that it is arranged so that every transition from one value to the next value involves only one bit change. The gray code is sometimes referred to as reflected binary, because the first half of the values compare with those of the last half of the values are same, but in reverse order.

A Gray code preserves the orthogonal property [3]. Hence we can generate equivalent orthogonal codes for gray code use it to number the processing nodes of hypercube [1]. An orthogonal code is a set of sequences in which all pairwise cross correlations are zero.

To generate n-bit cyclic gray code for n dimensional hypercube we can use the algorithm of n- bit conversion given by K. Usha and K. Jaya Sankar[1].As we know that n can be either even or odd corresponding to the $2^n$ number of processing nodes in a hypercube therefore we can use the algorithm accordingly. We are trying to develop our model for the 3-dimensional hypercube in which 3-bit gray code to label its processing node. As shown in the figure (1) below.




## III. 3-DIMENSIONAL HYPERCUBE LABELED WITH 3 BIT GRAY CODE

According to algorithm of Usha and K. Jaya Sankar[1] generation of n bit cyclic gray code involves three parts:

*A.* First part corresponds to generate binary cyclic gray code
*B.* Second part corresponds to the generation of inverse gray code for n bits.
*C.* Third part is the generation of orthogonal codes using n-bit gray code and inverse gray codes.

While the first part of the algorithm is again broken into three steps as described below:

*1) Step1.1:* Let an n-bit Cyclic Gray code be needed. Let (P1, P2, P3,…. Pn) be a permutation of (1,2,3,….n). The M = $2^n$ integers (0, 1, 2, …, ($2^n$-1))can be arranged in the following indexed indicial sets.

$Q_0 = 2^0$ {1, 3, 5…….}
$Q_1 = 2^1$ {1, 3, 5…….}
:
$Q_{n-1} = 2^{n-1}$

*2) Step 1.2:* Then, for any integer value of 'n', starting with the row of all zeros as a zeroeth row, the $i^{th}$ row is obtained from the $(i-1)^{th}$ row by replacing the $pj^{th}$ bit by its successor, if it is in Qj-1.

Let us consider the construction of a 3-bit binary Gray code. All the integers, i.e.,{0, 1, 2, 3,…..($2^3$-1)} are arranged in the form of indicial sets as shown below:

$Q_0 = 2^0$ {1, 3, 5,7} = 1,3,5,7
$Q_1 = 2^1$ {1, 3} = 2,6
$Q_2 = 2^2$ {1} = 4

As stated earlier, let (P1, P2, P3 … Pj ..Pn) be a permutation of (1,2,3,….j,….n). Since we are considering a 3-bit case, consider the permutation {2,3,1}Hence, $P_1 = 2$; $P_2 = 3$; $P_3 = 1$. The first code word is (0 0 0) which is the zeroth row of the code.

*3) Step-1.3:* To obtain 1st row, we have to change Pj$^{th}$ bit if '1' is in Qj-1
Here, 1 is in $Q_0$. Therefore, P1 bit is to be changed and $P_1$=2, hence the code is (0 1 0). Similarly, since '2' is in Q1, $P_2$ bit (i.e. 3rd bit) is to be changed, hence the code is (1 1 0). The resulting code obtained by continuing this procedure is tabulated in Table I

| i$^{th}$ Row | $P_j$ | Bit to be changed | 3-bit binary gray code | | |
|---|---|---|---|---|---|
| | | | 3 | 2 | 1 |
| 0 | - | - | 0 | 0 | 0 |
| 1 | $P_1$ | 2 | 0 | 1 | 0 |
| 2 | $P_2$ | 3 | 1 | 1 | 0 |
| 3 | $P_1$ | 2 | 1 | 0 | 0 |
| 4 | $P_3$ | 1 | 1 | 0 | 1 |
| 5 | $P_1$ | 2 | 1 | 1 | 1 |
| 6 | $P_2$ | 3 | 0 | 1 | 1 |
| 7 | $P_1$ | 2 | 0 | 0 | 1 |

Table I: 3-bit Cyclic Gray Code with permutation {2, 3, 1}

A total of n! Gray codes can be generated using the above technique for any integer value of 'n' and all these Gray codes are cyclic. Second part of the algorithm too is subdivided into three parts as mentioned below:

*a) Step 2.1:* Let us consider the construction of a 3-bit binary Inverse Gray code. All the integers,
i.e., {0, 1, 2,3,…..($2^3$ –1)} are arranged in the form of indicial sets as given below:
Q0 = $2^0$ {1, 3, 5,7} = 1,3,5,7
Q1 = $2^1$ {1, 3} = 2,6
Q2 = $2^2$ {1} = 4

*b) Step 2.2:* Since we are considering a 3-bit case, consider the permutation {2,3,1}. Hence, $P_1$ = 2; $P_2$ = 3; $P_3$ = 1. The first code word is (0 0 0) which is the zeroeth row of the code.
*c) Step 2.3:* To obtain 1st row, we have to change Pj$^{th}$ bit if '1' is in Qj-1. Here, 1 is in Q0.
Therefore, all the other bits except P1 are to be changed, hence the code is (1 0 1). Similarly, since '2' is in Q1, all the other bits except P2 bit are to be changed, hence the code is (1 1 0). This procedure is to be repeated for all the rows except M/2 th row (in this case it is 4$^{th}$ row).To obtain 4$^{th}$ row all the 3-bits are to be changed. The resulting code obtained by continuing this procedure is given in Table II.

| i$^{th}$ Row | $P_j$ | Bit to be changed | 3-bit binary gray code | | |
|---|---|---|---|---|---|
| | | | 3 | 2 | 1 |
| 0 | - | - | 0 | 0 | 0 |
| 1 | $P_1$ | 1,3 | 1 | 0 | 1 |
| 2 | $P_2$ | 1,2 | 1 | 1 | 0 |
| 3 | $P_1$ | 1,3 | 0 | 0 | 1 |
| 4 | all | 1,2,3 | 1 | 0 | 0 |
| 5 | $P_1$ | 1,3 | 0 | 0 | 1 |
| 6 | $P_2$ | 1,2 | 0 | 1 | 0 |
| 7 | $P_1$ | 1,3 | 1 | 1 | 1 |

Table II: 3-bit Inverse Gray Code with Permutation {2, 3, 1}

Similarly n! Inverse Gray Codes can be generated for any integer 'n' using n! Permutations. the third part of the algorithm we have four steps as discussed below: Two sequences are said to be Orthogonal when the Cross-correlation (inner product) between them is zero. With this construction procedure $2^n$ orthogonal code words of length 2n can be obtained. This algorithm can be applied to n=3 or 4 only.

- *Step 3.1:* Generate n-bit Gray code using the algorithm discussed in PART-I with any permutation.
- *Step 3.2:* Using the same permutation generate n-bit Inverse Gray Code using the algorithm in PART-II.
- *Step 3.3:* Append Inverse Gray Code to the Gray Code to result in 2n-length GIG Code (Gray Inverse Gray).
- *Step 3.4:* Each row of this GIG Code is a Binary Orthogonal Code word of length 2n.

For example a 6-bit (3+3) GIG Code of 8($2^3$) code words is constructed by appending 3-bit Gray Code with 3-bit Inverse Gray Code generated using the algorithms of PART-I & PART-II for a given permutation. Each row of this GIG Code is a 6-length Binary Orthogonal Code word.

| Processor Node | 3-bit gray code | 3-bit Inverse gray code | 6-bit GIG code | 6-length Orthogonal code (in decimal notation) |
|---|---|---|---|---|
| $P_0$ | 000 | 000 | 000000 | 0 |
| $P_1$ | 010 | 101 | 010101 | 21 |
| $P_2$ | 110 | 110 | 110110 | 54 |
| $P_3$ | 100 | 011 | 100011 | 35 |
| $P_4$ | 101 | 100 | 101100 | 44 |
| $P_5$ | 111 | 001 | 111001 | 57 |
| $P_6$ | 011 | 010 | 011010 | 26 |
| $P_7$ | 001 | 111 | 001111 | 15 |

Table III: Give the 6-length Orthogonal Code sets generated using the permutation {2,3,1} for processors numbered P0-P7

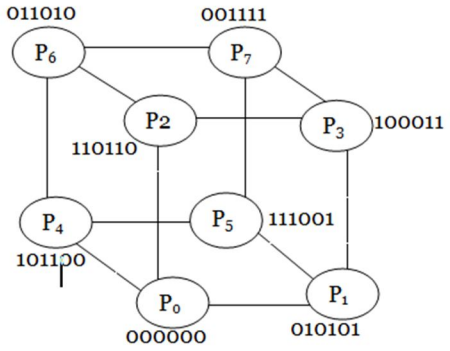



Figure 2: 6 bit GIG code labeled 3-D Hypercube

After development of orthogonal labels for processing nodes of hypercube let us now assume that if any arbitrary node wants to send data bit "1" then it will send its 6 bit GIG code onto the shared communication channel while if it wishes to transmit "0" it has to send the 6 bit complement GIG code. Also we will be using "+1" to code "1" and "-1" to code "0" Therefore the corresponding bipolar data bit sequence of each node is summarized below in the table IV.

| Processing node | 6bit GIG bipolar code for sending "1" | 6bit complement GIG bipolar code for sending "0" |
|---|---|---|
| $P_0$ | (-1-1 -1 -1 -1 -1) | (+1+1+1+1+1+1) |
| $P_1$ | (-1 +1 -1+1 -1 +1) | (+1-1+1-1+1-1) |
| $P_2$ | (+1 +1 -1 +1 +1 -1) | (-1-1+1-1-1+1) |

| | | |
|---|---|---|
| $P_3$ | (+1 -1 -1 -1 +1 +1) | (-1+1+1+1-1-1) |
| $P_4$ | (+1 -1 +1 +1 -1 -1) | (-1+1-1-1+1+1) |
| $P_5$ | (+1 +1 +1 -1 -1 +1) | (-1-1-1+1+1-1) |
| $P_6$ | (-1 +1 +1 -1 +1 -1) | (+1-1-1+1-1+1) |
| $P_7$ | (-1 -1 +1 +1 +1 +1) | (+1+1-1-1-1-1) |

Table IV: Orthogonal bipolar Code used as data bit coding

All data bit bipolar GIG code sequences are pairwise **orthogonal**, by which we mean that the normalized inner product of any two data bit GIG code sequences, **S** and **T** (written as **S .T**), is 0.

$$\mathbf{S \bullet T} \equiv \frac{1}{m}\sum_{i=1}^{m} S_i T_i = 0$$ **(Equation (i) )**

if **S .T** = 0, then **S .T'** is also 0. The normalized inner product of any data bit GIG code sequence with itself is 1:

$$\mathbf{S \bullet S} = \frac{1}{m}\sum_{i=1}^{m} S_i S_i = \frac{1}{m}\sum_{i=1}^{m} S_i^2 = \frac{1}{m}\sum_{i=1}^{m} (\pm 1)^2 = 1$$

**(Equation (ii) )**

Here m= length of GIG code sequence

This follows because each of the *m* terms in the inner product is 1, so the sum is *m*. Also note that

**S .S'** = 0 Equation (iii)

During each bit time, any processing node can transmit a 1 (by sending its GIG code sequence), it can transmit a 0 (by sending its complement GIG code sequence), or it can be silent and transmit nothing over the shared communication links.

When two or more nodes transmit simultaneously, their bipolar GIG data bit codes are added linearly. For example, if in one time slot three processors output 1 and one processor outputs 0, 1 will be received. We see example of one or more processors transmitting 1 bit at the same time. In an example, both *P1* and P2 transmit 1 bit so we get the sum of their bipolar GIG code sequence, namely:

$P_1$= (-1 +1 -1+1 -1 +1) and $P_2$=(+1 +1 -1 +1 +1 -1)

so intermixed signal S will be equal to

(0 +2 -2 +2 0 0)

$S.P_1$=(0 +2 -2 +2 0 0).(-1 +1 -1+1 -1 +1)/6

=(0+2+2+2 0 0)/6

=1

$S.P_2$=(0+2-2+2 0 0).(+1+1-1+1+1-1)/6

=(0+2+2+2 0 0)/6

=1

That proves true because $P_1$ and $P_2$ both transmitted their corresponding bipolar GIG code sequence in a wish to transmit "1" bit.

Similarly if $P_0$,$P_1$,$P_2$ and $P_3$ all wish to transmit simultaneously and the resulting multiplexed data bit bipolar sequence is

S = P0'+P1+P2' +P3= (0 0 0 0 0 4)

Then

S. P0= (0 0 0 0 0 4). (-1-1 -1 -1 -1 -1)/6

= (0 0 0 0 0 0 40 0 0 0 -4)/6

= -0.66

The result is negative therefore we can say that P0 node transmitted its complement bipolar GIG code sequence in a wish to transmit data bit "0"

S.P1= (0 0 0 0 0 4). (-1 +1 -1+1 -1 +1)/6
= (0,0,0,0,0,4)/6
=0.66

The result is positive therefore we can say that P0 node transmitted its bipolar GIG code sequence in a wish to transmit data bit "1"

S.P2=(0 0 0 0 0 4). (+1+1-1+1+1-1)/6
= (0 0 0 0 0 0 40 0 0 0 -4)/6
= -0.66

Like P0 result is negative therefore we can say that P2 node transmitted its complement bipolar GIG code sequence in a wish to transmit data bit "0"

And S.P3=(0 0 0 0 0 4). (+1+1-1+1 +1 -1)/6
=(0 0 0 0 0 4)/6
=0.66

Same as P0 result is positive P3 node transmitted its bipolar GIG code sequence in a wish to transmit data bit "1"

The number of nodes that send concurrently can be made arbitrarily large by using longer bipolar processing node sequences. For $2^n$ nodes, $2^n$ orthogonal bipolar GIG code sequences of length $m=2n$. However, one significant limitation is that we have assumed that all the processing nodes are synchronized in time at the receiver.

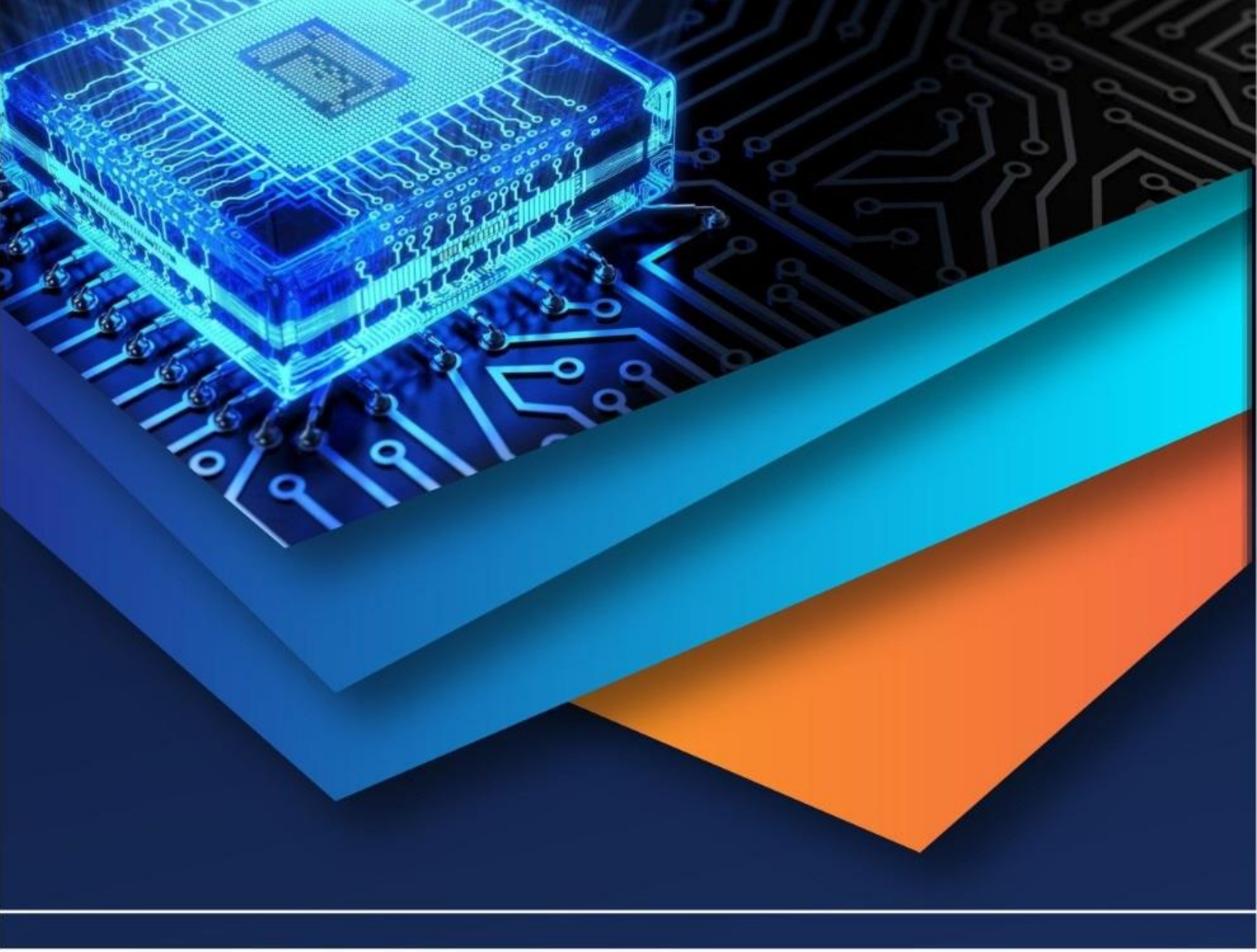
## IV. CONCLUSION

The following paper presents 1's complement logic and gray code based concurrent access mathematical model that can be used in parallel computing hypercube interconnection network, by each of the processing no de for sharing the communication links to send their data simultaneously and recover the individual data of each node which may be intermingled during the transition with data bits send by the other nodes. The bipolar GIG code sequences of each node are developed by using algorithm of K. Usha and K. Jaya Sankar [1] are utilized for concurrent access method over the shared communication link in hypercube interconnection topology .The original data bit sent by the processing node can be obtained from the intermingled signals through the normalized inner product of multiplexed signals and the bipolar GIG code sequence of the respective node. and further divide the result with the total length of the bipolar GIG code sequence which is m=2n (where n is binary digits used to label the processing nodes of the hypercube).If result comes positive it shows that the processing node has sent its original bipolar GIG code sequence in a wish to send the data bit "1" while if result comes negative it shows that the processing node has sent its complement bipolar GIG code sequence in a wish to send the data bit "0".

The number of nodes that send concurrently can be made arbitrarily large by using longer bipolar processing node sequences. For $2^n$ nodes, $2^n$ orthogonal bipolar GIG code sequences of length m=2n. However, one significant limitation is that we have assumed that all the processing nodes are synchronized in time at the receiver.

## REFERENCES

[1] K. Usha and K. Jaya Sankar, "Binary Orthogonal Code Generation for Multi-User Communication using n-bit Gray and Inverse Gray Codes" International Journal of Electronics and Communication Engineering. ISSN 0974-2166 Volume 5, Number 2 (2012), pp. 165-174
[2] Y. Saad, M.H. Schultz , "Topological properties of hypercubes ", IEEE Transactions on Computers Volume: 37 , Issue: 7 , Jul 1988
[3] Katare, R.K., Chaudhari, N.S., "Study Of Topological Property Of Interconnection Networks And Its Mapping To Sparse Matrix Model" International Journal of Computer Science and Applications, Oct. 2009 Technomathematics Research Foundation Vol. 6, No. 1, pp. 26 – 39, October 2009.
[4] Ammon, J., "Hypercube Connectivity within cc NUMA architecture" Silicon Graphics, 201LN. Shoreline Blvd. Ms 565, Mountain View, CA 94043.
[5] Tamiko, Teil, "The design of the connection machine" Design Issues, Volume 10, Number 1, spring 1994.
[6] Yves Robert, "The impact of Vector and elimination Algorithm", halsted press, a division of John Wiley & sons, New York 1990.
[7] P. Guerrier and A. Grenier, A Generic Architecture for on Chip Packetswitched Interconnections, In Proc. Design Automation and Test in Europe Conf. (DATE), pp. 250-256,2000.
[8] Katare, R.K ., Chaudhari, N.S., A Comparative Study of Hypercube and Perfect Difference Network for Parallel and Distributed System and its Application to Sparse Linear System, Vol. 2 Sandipani Academic,Ujjain [M.P.] pp. 13-30,2007.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

# INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY