Mapping Hex-Cells into Two-Dimension Mesh Network

Abdulelah Saif[*]
Department of Computer Science, King Khalid University, Dhahran Al-Janoob, K.S.A.

Keywords: Parallel Computing, Interconnection Networks, 2D Mesh, Hex-Cells, Tree-Hypercube, Mapping.

Abstract: Graph mapping is an important aspect for interconnection networks used for communication between processors in parallel systems. Some parallel algorithms use communication structures which can be represented by Hex-Cells HC. In order to run these algorithms on Two-Dimension 2D Mesh multiprocessor system, without changing the current topology and the running application, their communication graphs need to be embedded into 2D mesh. In this paper, we have developed an algorithm for embedding Hex-Cells HC(i) into 2D mesh M(2i,4i-1), where i = 1,2,3,…..i.e. To measure the efficiency of the algorithm, a comparison is done between 2D Mesh and Tree-hypercube in terms of dilation, congestion and expansion. As a result, the embedding of Hex-cells into 2D Mesh has dilation 1, congestion 1, expansion (4i-1)/3i, where i is the level of HC which is better than Tree-hypercube. Moreover, 2D Mesh embeds hex-cells for any level whereas Tree-hypercube embeds hex-cells for two levels.

1 INTRODUCTION

Chang and Chen (1997) state that one of the most important characteristics of any interconnection network is that it should be able to simulate any other interconnection networks without any costs. This feature puts such interconnection network a good candidate for working in general purpose parallel machine. Chang and Chen(1997) and Xinbo at al (2018),and Peng at al (2018) state that graph embedding is an important feature of parallel computing in order to convert from one network to another so that every element and edge in the guest graph is transformed into a node and edge in the host graph. This embedding has two benefits, the first reduces the time that processes spend by exchanging information among them, and the second reduces the total time because some processes are busy while others are not. This enhances the communication between processes as well as overcoming the congestion problem while the processes are communicating with each other. Additionally, there is also an important advantage of embedding, which is: If the guest graph is transferred to the host graph, the host graph can simulate the guest graph work (execute tasks in parallel) without any losses. If the guest graph is included in the host graph and there is less delay, less congestion, and less expansion, then the inclusion is optima. Thus, the goal of the embedding is to minimize the dilation, the congestion and the expansion cost. Reducing delay leads to reduced processes time and reduced congestion leads to reduced pressure in the host graph and reduced expansion reduces the hardware required in the host graph.

2D mesh is the popular general purpose networks with fixed node degree Emad (2008)) symmetry Grama at al (2003), and has the ability to embed other regularly networks. These characteristics of the 2D Mesh network make it able to map other networks efficiency. The problem of mapping Hex-Cells HC(d) network into 2D-Mesh has not attracted the attention of researchers towards it.

In this paper, We have developed an algorithm for embedding hex-cells HC(i) into 2D mesh M (2i, 4i-1), where i = 1,2,3,…..i.e. To measure the efficiency of the algorithm, a comparison is done between 2D Mesh and Tree-hypercube in terms of dilation, congestion and expansion. As a result, the embedding of Hex-cells into 2D Mesh has dilation 1, and congestion 1, expansion (4i-1)/3i, where i is the level of HC which is better than Tree-hypercube.

[*] https://mysite.kku.edu.sa/site/absaif/home

Moreover, 2D Mesh embeds hex-cells for any level whereas Tree-hypercube embeds hex-cells only for two levels.

Section 2 presents related works, section 3 presents meshes, section 4 presents hex-cell network, section 5 presents embedding hex-cells into 2d mesh, section 6 presents embedding hex-cells into 2d mesh with wraparound link(torus), and section 7 summarizes and concludes the paper.

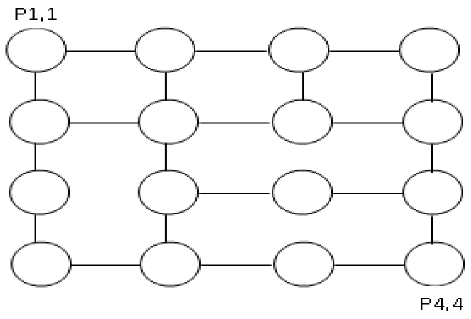
2 RELATED WORKS

Mesh is one of the most commonly used interconnection networks and, therefore, embedding between different meshes becomes a basic embedding problem. Not only does an efficient embedding between meshes allow one mesh-connected computing system to efficiently simulate another, but it also provides a useful tool for solving other embedding problems. This work shows an embedding of an s1* t1 mesh into an s2 * t2 mesh, where si <= ti (i = 1, 2), s1t1 = s2t2, such that the minimum dilation and congestion can be achieved and presents a lower bound on the dilations and congestions of such embeddings for different cases. Also, the work presents an embedding with dilation └s1/s2┘ + 2 and congestion └s1/s2┘ + 4 for the case s1 =>s2, both of which almost match the lower bound ┌s1/s2┐. Finally, for the case s1 < s2, the work presents an embedding which has a dilation less than or equal to 2* sqrt(s1), Shen (1997). Sang and Hyeong (1996) considered the problem of embedding complete binary trees into meshes using the row-column routing and obtained the following results: a complete binary tree with 2p-1 nodes can be embedded (1) with link congestion one into a 9/8(√2p)×9/ 8(√2p) mesh when p is even and a √(9/8*2p)×√(9/8*2p) mesh when p is odd, and (2) with link congestion two into a √(2p)×√(2p) mesh when p is even, and a √(2p-1)×√(2p-1) mesh when p is odd . Yang at al (2008), state that embedding torus in hexagonal honeycomb torus, states that a number of parallel algorithms admit a static torus-structured task graph. Hexagonal honeycomb torus (HHT) networks are considered as good candidates for interconnection networks. To execute a torus-structured parallel algorithm efficiently on an HHT, it is necessary to include the tasks to processors such that the communication overhead is minimum. This paper showed that a (3n, 2n) torus can be included into an nth-order HHT with congestion 4, dilation 3, expansion 1 and load factor 1. Consequently, a (3n, 2n) torus task graph can be executed on an nth-order HHT efficiently using a parallel algorithm. In Michael (2008), states that an undirected source graph G was included in a host graph EM. This paper presented an algorithm which was showed how to map G into EM with time and space O(|V |2) using the new ideas of islands and bridges. An island is a subgraph in the host graph which was mapped from one node in the guest graph while a bridge is an edge connecting two islands which was mapped from one edge in the guest graph. This work was motivated in real applications related to quantum computing and there was a need to map source graphs efficiently in the extended grid. CAHIT (1998) state that, a cubic tree is a tree in which all its internal vertices are of degree three except pendent vertices. This paper explores embedding cubic trees into rectangular grid of minimum size such that the edges are either horizontal or vertical segments. The method is based on the minimum area embedding of the three complete binary trees. The author gives necessary and sufficient conditions for cubic trees embeddable into a rectangular grid.

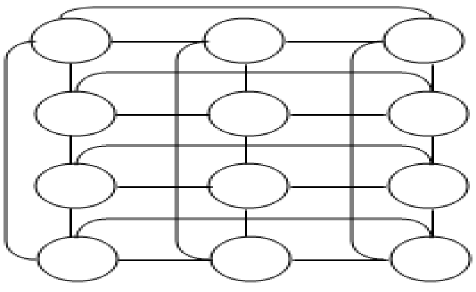
3 MESHES

In a Mesh network, the nodes are arranged in a k dimensional lattice of width w, giving a total of wk nodes or w*w in the case of 2D Mesh. Usually k=1 (linear array) or k=2 (2D array or 2D Mesh). Communication is allowed only between neighboring nodes. All interior nodes are connected to 2k other nodes, Mehdipourm (2016). A two-dimensional Mesh illustrated in Figure 1(a) is an extension of the linear array to two-dimensions. Each dimension has p nodes with a node identified by a two-tuple (i,j). Every node (except those on the periphery) is connected to four other nodes whose indices differ in any dimension by one. A variety of regularly structured computations map very naturally to a 2D Mesh. For this reason, 2D Meshes were often used as in parallel machines Grama at al (2003). Some data transfers in 2D Mesh may require 2((w*w)½-1) links to be traversed. This can be reduced by using wraparound connections between nodes on same row or column as in Figure 1(b) or when k=3 (three dimensions) as in Figure 1(c) Mehdipourm (2016).

a)



b)



c)

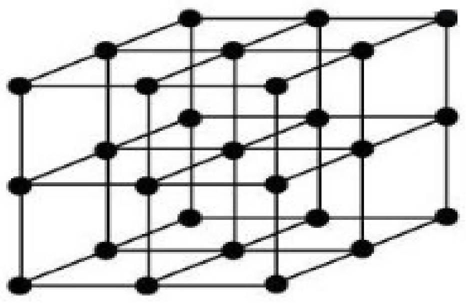


Figure 1: Meshes :(a) k=2 , w=4 without wraparound, (b) k=2 , w=3 with wraparound, and (c) k=3 ,w=3 with wraparound.

4 HEX-CELL NETWORK

A hex-cell network which has a depth d is defined by HC (d) and is constructed recursively using hexagonal cells, each hexagon has six nodes. HC (d) has d levels numbered from 1 to d, where, level 1 is the level with one hexagon cell. Level 2 is the six hexagon cells surrounding the hexagon at level 1. Level 3 is the 12 hexagon cells surrounding the six hexagons at level 2, as shown in Figure 2 the HC(d) network levels are labeled from 1 to d. Each level i has Ni nodes, which are the processing elements interconnected in a ring structure. Addressing nodes in HC is shown in Figure 3, Sharieh, et al, (2008).

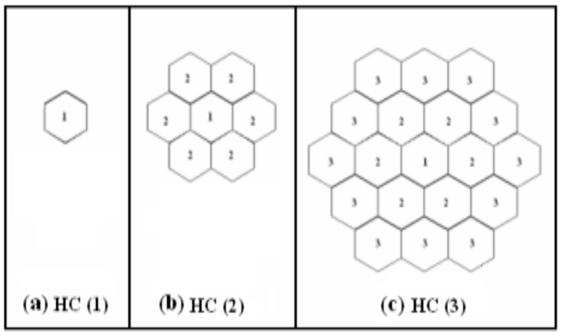


Figure 2: (a) HC (one level) (b) HC (two levels) (c) HC (three levels).

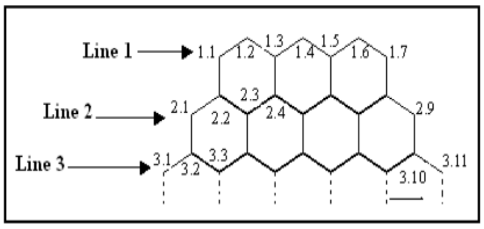


Figure 3: Addressing nodes in HC.

In HC(d), the number of nodes at level i is : Ni = 6(2i -1). The total number of nodes in HC(d) is : N = 6d2. The number of links in HC(d) is L = 9d2 -3d. The diameter of HC(d) is 4d-1.

5 EMBEDDING HEX-CELLS INTO 2D MESH WITHOUT WRAPAROUND LINK

s * t 2D Mesh, M, is a network in which the nodes are ordered in a Mesh of s rows numbered from 0 to s - 1 from top to bottom, and t columns numbered from 0 to t – 1 from left to right. The node at row i and column j is defined as M(i,j). The 2D Mesh is one of the most important networks and has received extensive studies due to numbers of reasons, among them many data structures especially arrays and matrices, fit into a Mesh-connected system. Moreover, some real multiprocessor computer systems have been produced based on Meshes. The study of mapping between meshes is used in many applications. One application is to allow a mesh to simulate other meshes of various ratios, which mean matrices of various shapes can be efficiently mapped to a mesh-connected system Shen (1997). Hex-cell can be included into 2D Mesh by including hex-cell nodes into 2D mesh nodes, and hex-cell edges into 2D Mesh edges; without adding edges (i.e. dilation 1),

```
Map_Hex-Cell_Into_2 D_Mesh( int d)  // d is  depth of Hex_cell network
{
int node(2*d,4*d-1)
max_columns_in_row =( 4 * d – 1) //determine the maximum columns in each row
First_row_element=d-1  //determine the first element in each row
For j=1 to d-1
max_columns_in_row = max_columns_in_row -2   // end of  loop
For i=1 to d-1
{
For j=1 to max_columns_in_row
{
node (i,j+ First_row_element)  in 2D Mesh = node (i,j) in HC
if(i>1 and (link is found between node (i,j) and node (i-1,j-1) in HC) and d>1)
Connect Node (i, j) in 2D Mesh with node (i-1, j) in 2D Mesh through a link
// end of if statement
}// end of internal loop
First_row_element = First_row_element-1
max_columns_in_row = max_columns_in_row +2
}// end of external loop
For i=d to d+1
{
For j=1 to max_columns_in_row
{
node (i,j)  in 2D Mesh = node (i,j) in HC
if(i>1 and (link is found between node (i,j) and node (i-1,j-1) in HC) and d>1)
Connect Node (i, j) in 2D Mesh with node (i-1, j) in 2D Mesh through a link
// end of if statement
}// end of internal loop
}// end of external loop
First_row_element = 1
For i=d+2 to 2*d
{
max_columns_in_row = max_columns_in_row -2
For j=1 to max_columns_in_row
```

Figure 4: The mapping algorithm of Hex-Cells HC(d) into 2D Mesh M(2d,4d-1).

```
{
node (i,j+ First_row_element)  in 2D Mesh = node (i,j) in HC
if(i>1 and (link is found between node (i,j) and node (i-1,j-1) in HC) and d>1)
Connect Node (i, j) in 2D Mesh with node (i-1, j) in 2D Mesh through a link
// end of if statement
}// end of internal loop
First_row_element = First_row_element +1
}// end of external loop
}// end of  Mapping function
```

Figure 4: The mapping algorithm of Hex-Cells HC(d) into 2D Mesh M(2d,4d-1) (cont.).

congestion 1 and with lower expansion. Smaller dilation leads to shorter communication delay where the host graph (2D Mesh) emulate the guest graph (hex-cell), and smaller expansion leads to more efficient utilization of the processor where the host graph (2D Mesh) emulates the guest graph (Hex-Cell). In the next section, the algorithm for mapping hex-cell into 2D mesh, M (i,j) is presented.

5.1 The Proposed Algorithm for Embedding Hex-Cells into 2D Mesh

We have designed an algorithm for embedding hex-cell HC(d) into 2D Mesh M(i,j) , where i=2d, j=4d-1 where d=1,2,3,…; without having additional number of edges when mapping edges of hex-cell (i.e. dilation one and congestion one), with expansion 1 when d =1, expansion 1.16 when d =2, expansion 1.22 when d=3 and expansion 1.3 for any value of d where d>10. Mapping nodes of hex-cell into nodes of 2D Mesh is done by mapping the addresses and edges of the hex-cell into the addresses and edges of 2D Mesh. Algorithm for embedding hex-cells into 2D Mesh networks is in Figure 4.

5.2 Examples on the Embedding Algorithm, and Discussion

Example 5.2.1; mapping hex-cell HC(1) (Figure 5) into 2D Mesh M(2,3). Figure 6 illustrates this example. In figures of 2D Mesh, the addresses within the nodes are for 2D Mesh and the addresses above the nodes are for HC.

Example 5.2.2; mapping hex-cell HC(2) (Figure 7) into 2D Mesh M(4,7), Figure 8 illustrates this example.

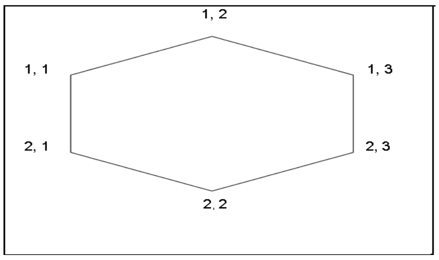


Figure 5: HC(1).

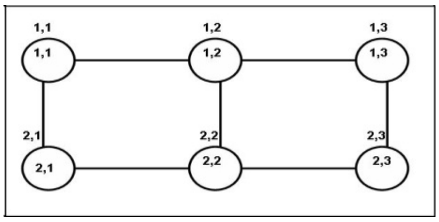


Figure 6: Mapping of HC(1) into M(2,3).

Example 5.2.3; mapping hex-cell HC(3) (Figure 9) into 2D Mesh M(6,11), Figure 10 illustrates this example.

This process continues in the same manner for mapping any level of hex-cells into 2D Mesh.

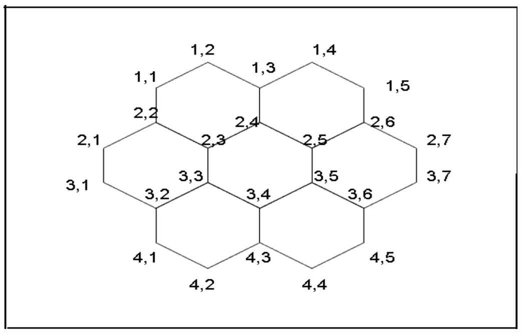


Figure 7: HC(2).

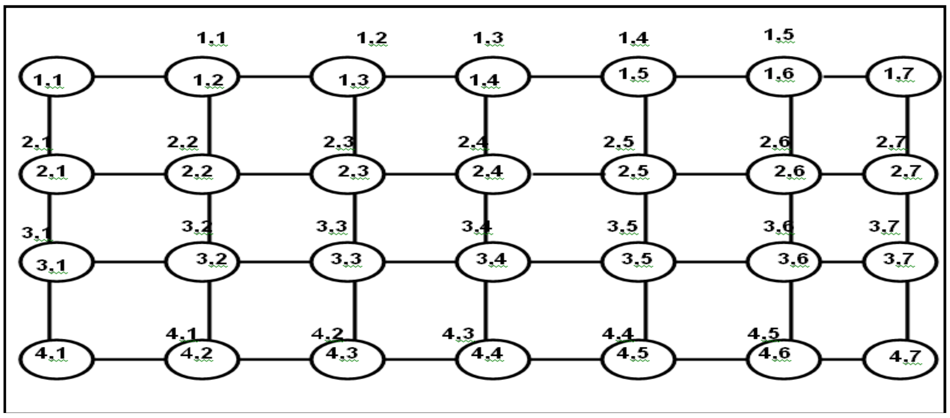


Figure 8: Mapping of HC(2) into M(4,7).

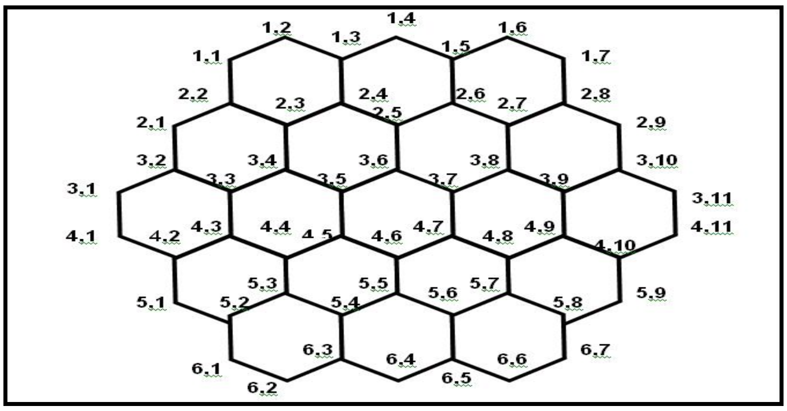


Figure 9: HC(3).

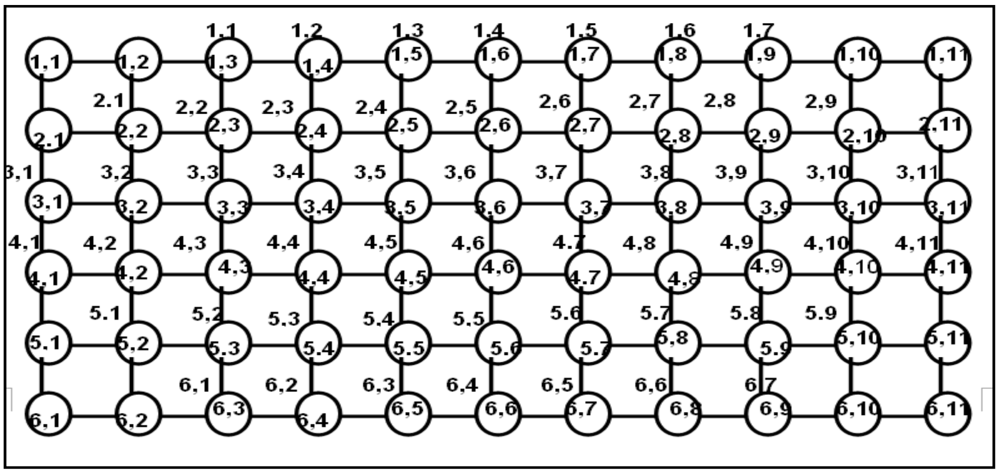


Figure 10: Mapping of HC(3) into M(6, 11).

5.3 A Note about Mapping Hex-Cells into 2D Mesh

In the following table, we list the total number of nodes in HC(d), M(2d,4d-1), and the number of extra nodes in the embedding of hex-cell into 2D Mesh.

Table 1: Number of extra nodes in embedding hex-cells into 2D Mesh.

D	Total number of nodes in HC(d)	Total number of nodes in M(2d,4d-1)	Number of extra nodes in embedding
1	6	6	0
2	24	28	4
3	54	66	12
4	96	120	24
5	150	190	40

Table 2: Comparison among TH and 2D Mesh after embedding HC into them.

d	Total number of nodes in HC(d)	Total number of nodes in TH(2,2d)	Total number of nodes in M(2d,4d-1)	Number of extra nodes in embedding HC(d) into TH (2,2d)	Number of extra nodes in embedding HC(d) into M(2d,4d-1)
1	6	7	6	1	0
2	24	31	28	7	4
3	54	127	66	73	12
4	96	511	120	415	24
5	150	2047	190	1897	40

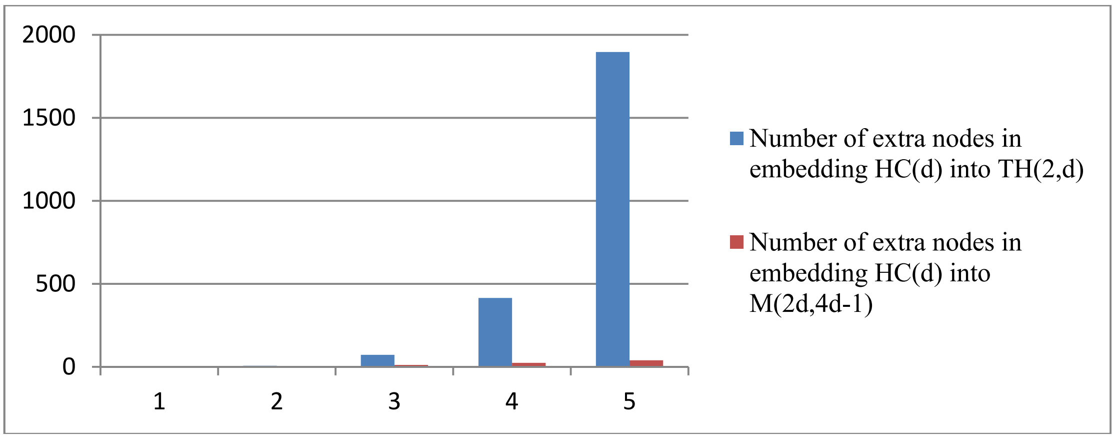


Figure 11: Comparison among TH and 2D Mesh after embedding HC into them.

5.4 Comparison between Tree-Hypercube TH Qatawneh (2008) and 2D Mesh

From the above table, and Figure 1, we notice that when mapping hex-cells into 2D Mesh, the number of extra nodes is less than the number of extra nodes when mapping hex-cells into tree-hypercube and therefore the expansion of embedding HC(d) into M(2d,4d-1) is less than expansion of embedding HC(d) into TH (2,2d).

6 EMBEDDING HEX-CELLS INTO 2D MESH WITH WRAPAROUND LINK

The most common topology attainable with nodes of four links is the mesh or square grid. By connecting the ends of the mesh around, a Torus is produced, Wilson.

Hex-Cells can be embedded into 2D mesh with wraparound link in the same way of embedding hex-cells into 2D mesh without wraparound link by the same algorithm mentioned above in section 5. But the benefit of wraparound link in the 2D mesh lies in the routing where the short path from the source to the destination is less than that of 2D mesh without wraparound link because the diameter of torus is less than that of 2D mesh without wraparound link.

Example 6.1.1; mapping hex-cell HC(1) (Figure 12) into torus, M(2,3). Figure 13 illustrates this example. In figures of torus, the addresses within the nodes are for torus and the addresses above the nodes are for HC.

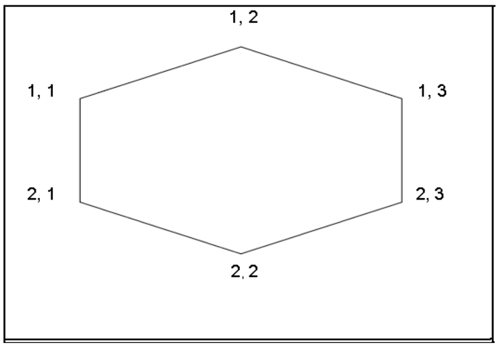


Figure 12: HC(1).

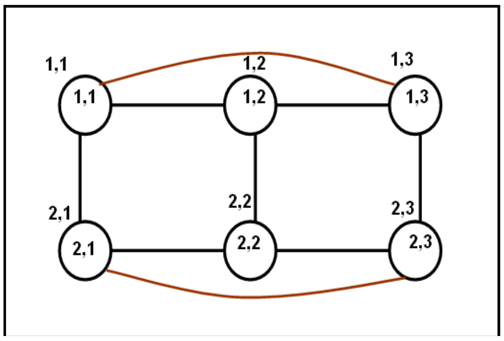


Figure 13: Mapping of HC(1) into torus(2,3).

Example 6.1.2; mapping hex-cell HC(2) (Figure 14) into 2D mesh M(4,7), Figure 15 illustrates this example.

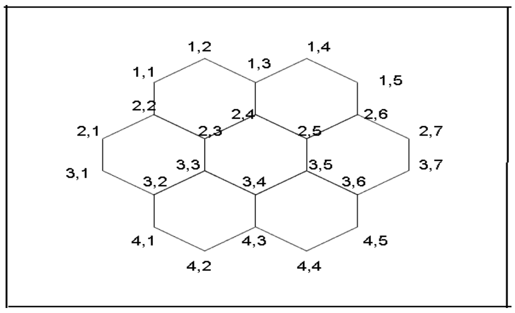


Figure 14: HC(2).

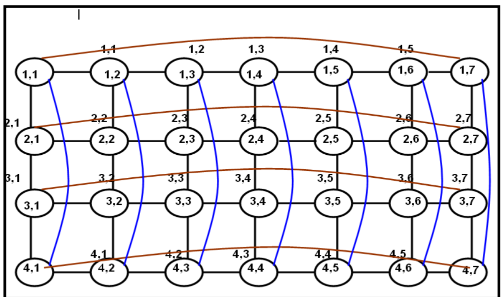


Figure 15: Mapping of HC(2) into torus(4,7).

As before, this process continues in the same manner for mapping any level of hex-cells into 2D mesh with wraparound link (torus).

7 CONCLUSIONS AND FUTURE WORK

One of the most important characteristics of any interconnection network is that it should be able to simulate any other interconnection networks without any costs. This feature puts such interconnection network a good candidate for working in general purpose parallel machine. Graph embedding is an important feature of parallel computing in order to convert from one network to another so that every element and edge in the guest graph is transformed into a node and edge in the host graph.

In this paper, We have developed an algorithm for including hex-cells HC(i) into 2D mesh M (2i, 4i-1), where i = 1,2,3,…..i.e. To measure the efficiency of the algorithm, a comparison is done between 2D Mesh and Tree-hypercube in terms of dilation, congestion and expansion. As a result, including Hex-cells into 2D Mesh has dilation 1, congestion 1, and expansion (4i-1)/3i, where i is the level of HC which is better than Tree-hypercube. Moreover, 2D Mesh embeds hex-cells for any level whereas Tree-hypercube embeds hex-cells only for two levels.

For future work, we suggest mapping hex-cells into other topologies that have diameter less than that of hex-cells, such as X-Torus and An nth-order HHT, and with lower dilation, congestion and expansion as possible.

REFERENCES

Chang, H.Y., & Chen, R.J. (1997). Embedding Cycles in IEH Graphs. *Information Processing Letters*, 64(4), 23-27.

Grama, A., Gupta, A., Karypis G., & Kumar, V. (2003). *Introduction to Parallel Computing*, Addison Wesley.

Emad, A. (2008). A Comparative Study on the Topological Properties of the Hyber-Mesh Interconnection Network. *Proceedings of The World Congress on Engineering*, 1.

Shen, X. (1997). On Embedding Between 2D Meshes of the Same Size. *IEEE Transactions on Computers*, 46(8).

Sang, L., Hyeong, C. (1996). Embedding of Complete Binary Trees into Meshes with Row-Column Routing. *IEEE Transactions on Parallel And Distributed Systems*, 7(5).

Yang, X., Tang, Y., & Cao, J. (2008). Embedding Torus in Hexagonal Honeycomb Torus. *Computers and Digital Techniques Journal*, 2(2), 86-93.

Michael, C. (2008). Embedding Graphs into the Extended Grid.

CAHIT. (1998). Embedding Cubic Trees into the Rectangular Grids.

Mehdipourm F. (2016). Interconnection Networks. Retrieved from https://www.sciencedirect.com/topics/computer-science/interconnection-networks.

Sharieh, A., Qatawneh, M., Almobaideen, W. & Sleit, A. (2008). Hex- Cell: Modeling, Topological Properties and Routing Algorithm. *European Journal of Scientific Research*, 22(3), 457-468.

Qatawneh, A. (2008). Embedding Hex-Cells into Tree-Hypercube Networks.

Wilson, P.A. Homogeneous Parallel Network Topologies for Factory Environments. *IEEE Transactions*.

Xinbo, L., Buhong, W., Zhixian, Y. (2018). Virtual Network Embedding based on Topology Potential. *Entropy*.

Peng, C., Xiao, W., Jian, P., & Wenwu, Z. (2018). A Survey on Network Embedding. *Association for the Advancement of Artificial Intelligence*.