Area Efficient Binary Tree Layout

Sourav Bhattacharya, Wei-Tek Tsai

Computer Science Department, University of Minnesota, MN 55455

Abstract

H-Tree layout for binary trees can utilize only 50% of the available nodes. Improved binary tree layout techniques [1,6] have been developed only after relaxing the rectangular grid model assumptions. This paper proposes an area-efficient VLSI layout strategy for full binary trees *without* relaxing the rectangular grid model assumptions. For a height-5 full binary tree we developed a (5×8) layout pattern on an ad hoc basis. This tile is area efficient than an equivalent H-Tree layout of a height-5 full binary tree. Using this tile we build higher level trees in a way identical to H-Tree. The area efficiency remains for any level of tree construction. The proposed layout has an improved aspect ratio than H-Tree and features a reduced length of the longest link.

Key words: Binary tree, H-tree, VLSI, Rectangular Grid Model, Border I/O.

1. Introduction

Among various interconnection styles binary tree-based parallel processors is popular because many algorithms map naturally onto the binary tree architecture efficiently. Also, a binary tree requires only three ports per node. Tree structures have the desirable property of logarithmic path from the root to any leaf element. It has simple and regular interconnection among nodes making it suitable for VLSI/WSI implementation. Recent advances in VLSI/WSI technology make it possible to construct a cost-effective binary tree interconnection network comprising of many Processing Elements (PE). Each PE is similar to a stand alone computer and may have its own local memory. PEs are either independently controlled or globally controlled.

Let FBT_k denote a full binary tree of height-k. Previous embeddings of binary trees into rectangular arrays have been of two types: hand embeddings [5] and recursive embeddings [3]. Hand embeddings have optimum processor utilization but they are not feasible for large constructions. Recursive embeddings follow a pattern for binding two FBT_k layouts to get a FBT_{k+1} layout. Early proposals for recursive embeddings were based on the classical H-tree approach [3]. More recent designs [1,6] employ tile based schemes to improve the area efficiency. H-tree approach works on rectangular grid model assumptions and can utilize only 50% of the layout area. With relaxations on grid model several layout schemes have been developed with area efficiencies higher than 50%. [1] asymptotically approaches 100% processor utilization for large trees *only* with the assumption that a PE can be used as a tree node and a connecting element between distant nodes. [6] achieves similar results using tiles. All these approaches however assume a hexagonal grid instead of a rectangular grid model. Therefore, the results are not directly comparable to the H-tree layout.

We propose a new scheme for laying out a full binary tree on a *rectangular* grid. A PE can only be used either as a tree node or as a connecting element between two nodes, but *not both*. In this regard our results are directly comparable to the results of H-tree layout. We have achieved area efficiency *without* relaxing any assumption in H-tree layout. Our contribution is in joining both hand embedding and recursive embedding styles. We use hand embedding to layout small-size binary trees. This embedding requires less area than equivalent H-trees. Then we follow the recursive pattern of H-tree to build higher level trees using this hand embedded patterns as building blocks. Our layout scheme uses a FBT_5 embedded in an ad hoc fashion on a rectangular grid. The dimensions of this tile is (5×8). This tile serves as a building block for higher level trees. Recursive technique to layout a FBT_{k+1} from two FBT_k is used in a way identical to the H-tree style. Therefore, this method can be automated in a way similar to H-tree. However, following H-Tree pattern a FBT_5 needs (7×7=49) layout area while our basic building block needs (5×8=40) lead-

ing to nearly 18% area compression. Since both H-tree and our method follow identical recurrence equations this area advantage will be retained for higher level trees also. We have developed the equations for length and width of this layout. Table 1 shows comparative measurements of the proposed layout with respect to H-Tree.

Ideally the aspect ratio of a layout is expected to be *unity* so that the chip can be fabricated in a square area. The aspect ratio of H-tree layout is *one* or *two* depending on whether the height of the tree is *odd* or *even* respectively. Our layout has better aspect ratio. Aspect ratio of the basic tile (embedding FBT_5) is 0.63 ($=\frac{5}{8}$). For FBT_6 layout area is (11×8) while aspect ratio = 1.38 ($\frac{11}{8}$). Asymptotically, aspect ratio in our layout becomes either $\frac{2}{3}$ or $\frac{4}{3}$ depending on whether the height of the binary tree is *odd* or *even* respectively. Hence, aspect ratio deviates less from unity in our layout scheme than in H-Tree. Similarly, we have developed equations for the longest link. The longest link in our layout is reduced compared to an equivalent sized H-tree. Since, length of the longest link controls the upper limit of the clock speed our layout can yield faster execution of binary tree algorithms than H-tree.

In section 2 we briefly cover the H-tree layout construction since our layout style also follow identical recursive construction. However, our base case is a hand-embedded tile. We illustrate this tile pattern and show an example overall construction of a FBT_7. Section 3 develops layout dimension equations and compares them with corresponding features of H-tree layout. Section 4 concludes the paper.

2. H-tree Layout and Area-Improved Tile

A full binary tree of height m, as shown in Fig. 1a (m=7), can be laid out in the most naive way as in Fig. 1b. This layout style is called *Standard Tree Layout*. The layout pattern follows directly from the tree interconnection. The layout is simple to construct and can provide direct I/O to all the leaf PEs since every leaf PE has a boundary access. However, it is area expensive. For a FBT_m the layout area = $m \times 2^{m-1}$. In other words, for laying out $O(N)$ PEs a $O(N \times \log_2 N)$ area is required. It also demonstrates several other disadvantages such as poor *aspect ratio*, exponentially increasing *longest link length*. A more sophisticated layout technique have been introduced in [3] which is described in the following.

The binary tree structure employs the *divide and conquer* paradigm [4]. The same principle is also used in [3] to layout a full binary tree. The layout algorithm is recursive in construction, i.e., it combines two FBT_m layouts and a root node to arrive at a FBT_{m+1} layout. Fig 2a shows this recursive construction style, where a dashed box indicates a FBT_m. Using this recursive construction style the layout pattern for a full binary tree of arbitrary height can be designed. Fig. 2b shows a FBT_7 layout. It may be noted that the pattern in Fig. 2a resembles the English letter H and hence this layout style is called *H-tree* layout. The area equations for H-tree layout can be arrived at using the inductive principles:

Base Case: FBT_1 is a single PE requiring 1×1 area.

Inductive Hypothesis: Let the layout dimensions of a FBT_m be $A \times B$, where $A \geq B$.

Induction Step: The layout area of FBT_{m+1} is $A \times (2B+1)$. For FBT_{m+2} the layout area is $(2A+1) \times (2B+1)$.

Let A_m^H denote the H-tree layout area for FBT_m. Then A_m^H can be derived from the above equations as:

$$A_m^H = (2^{\frac{m+1}{2}} - 1)^2 = 2^{m+1} - 2^{\frac{m+3}{2}} + 1$$

for odd values of m, and

$$A_m^H = (2^{\frac{m+2}{2}} - 1) \times (2^{\frac{m}{2}} - 1) = 2^{m+1} - 3 \times 2^{\frac{m}{2}} + 1$$

for even values of m. Hence, the area is of the order $O(2^{m+1})$. The length of the longest link, L_m^H, in this layout is $2^{\lfloor \frac{m}{2} - 1 \rfloor}$ (where the longest link implies the longest *internal* link of the layout *excluding* the link between the root and the external world.

Tile Layout for H-tree

Recent designs [1,6] have employed tile based layout schemes to improve the area efficiency. However, all these approaches relax the grid model assumptions. In other words, binary tree layout schemes which yield area efficiencies higher than 50% are not comparable to the original H-tree layout style. We follow grid model assumptions and thus our results are directly comparable to the H-layout features. In the following assumptions of embedding a graph over a 2-dimensional (or in general multi-dimensional) grid are stated [2].

1) Each node occupies unit area. Distinct nodes of the graph are embedded at distinct grid intersection points.

2) All edges have unit width. Edges run along grid lines and no two edges overlap except when crossing perpendicular.

3) Any intersection point on the grid can either be mapped to a node of the graph or the intermediate point of an edge, but *not both.*

4) Edges can run *only* along vertical or horizontal directions.

Proposed Layout: We have developed a hand embedded layout pattern for height-5 complete binary tree. The basic idea of the pattern is shown in Fig. 3a. *Four* FBT_3 (shown in Fig. 3b) are laid out in a 5×7 area. It may be noted that each FBT_3 utilizes 7 nodes and Fig. 3a has compacted placing *four* such layouts in an area of 5×7 (utilizing 4×7=**28** nodes out of the total 35 intersection points in the grid). This embedding is area efficient. It may be noted that from the above (5×7) grid area there can be two possible ways of area reduction: (4×7) and (5×6). The former requires 100% utilization of nodes leaving no room for edge layout. The second alternative provides only two (=30-**28**) spare nodes for edge layout.

The layout of Fig. 3a leaves no room for interconnecting four FBT_3 layouts and forming a FBT_5. An additional track is needed for laying out the level-4 intermediate nodes and also the root of FBT_5. This extra node can be added either along the length or along the width. Former approach would require 5×(7+1) (=40) layout area while the latter would need (5+1)×7 (=42) layout area. For area efficiency the first style is chosen. Fig. 4 shows layout pattern of a FBT_5. This serves as the basic tile which can be used for higher level tree layout patterns. Therefore, the layout construction methodology becomes:

Base Case: FBT_5 is following Fig. 4 requiring 5×8 area.

Inductive Hypothesis: Let the layout dimensions of a FBT_m be $A \times B$, where $A \geq B$ and $m \geq 5$.

Induction Step: The layout area of FBT_{m+1} is $A \times (2B+1)$. For FBT_{m+2} the layout area is $(2A+1) \times (2B+1)$.

Fig. 5 shows a FBT_6 layout. Higher levels of construction is exactly analogous to the H-tree layout style. Fig. 6a & Fig. 6b show layouts for FBT_7 and FBT_8 respectively.

3. Layout Dimensions

In this section we develop equations for the dimensions of our proposed layout. Since, length and width are two primary metrics of the layout, we first develop analytical expressions for them. Then using these expressions other metrics, e.g., area, length of the longest link, aspect ratio, can be evaluated. We compare the proposed layout with H-tree layout. Number of border leaf PEs is computed for a given tree height. Finally we compare the proposed layout with H-tree layout in terms of area, boundary I/O and aspect ratio.

3.1. Length, Width and I/O Equations

Let A_m^H denote the area taken by a FBT_m using H-tree style and let the length & width of the layout be L_m & W_m respectively. Expressions for L_m and W_m can be obtained from the area equations (relating inductive construction of H-tree layout) in section 2.

$$L_m = W_m = 2^{\frac{m+1}{2}} - 1, \text{ for odd values of } m$$

$$L_m = 2^{\frac{m+2}{2}} - 1, W_m = 2^{\frac{m}{2}} - 1, \text{ for even values of } m.$$

The proposed layout is recursively constructed in a way exactly similar to the H-tree style. However, unlike the latter case, the layout construction begins from a (5×8) tile pattern embedding FBT_5. This is reflected in the modified *base case* of layout construction in section 2. Let L_m^{new}, W_m^{new} denote the length and width of the proposed layout for FBT_m. Length and width equations of proposed layout for FBT_m is found as (detailed calculation in [7]):

$$L_m^{new} = \frac{3}{4} \times 2^{\frac{m+1}{2}} - 1, W_m^{new} = \frac{9}{8} \times 2^{\frac{m+1}{2}} - 1, \textit{for odd } m.$$

$$L_m^{new} = \frac{3}{2} \times 2^{\frac{m}{2}} - 1, W_m^{new} = \frac{9}{8} \times 2^{\frac{m}{2}} - 1, \quad \textit{for even } m.$$

Let B_m^H (B_m^{new}) denote the number of boundary leaf PEs in H-tree layout (proposed tree layout) of FBT_m. Expression for B_m^H is well known and can be found in [7]. In evaluating the expression for B_m^{new} we first consider the boundary I/O of the base-tile pattern (Fig. 4). (5×8) layout of FBT_5 has 4 leaf PEs along each horizontal boundary and 2 leaf PEs along each vertical boundary. Without counting the four corner PEs more than once this leaves (2×4+ 2×2-4)=8 boundary leaf PEs. For FBT_m (with *odd* values of m) there are $2^{\frac{m-5}{2}}$ tiles along both horizontal and vertical direction. At the next higher level FBT_{m+1} is constructed placing two such FBT_m layouts one above another. [7] develops equations for B_m^{new} as:

$$B_m^{new} = 2^{\frac{m-5}{2}} \times 12 - 4, \text{ for } \textit{odd} \text{ values of } m$$

$$B_m^{new} = 2^{\frac{m}{2}} \times 2 - 4, \text{ for } \textit{even} \text{ values of } m$$

3.2. Comparisons with H-tree Layout

Using the expressions developed in section 3.1 we compare different parameters of our proposed layout scheme with the H-tree layout and standard tree layout. First expressions for *area, aspect ratio* are shown and area ratio over H-tree layout is developed. We use the following notations in our analysis: A, AS, B and L represent *area, aspect ratio, number of border leaf PEs* and *length of the longest link* respectively. These parameters are qualified by using superscript notations: H and *new* indicating H-tree and our proposed layout styles. Thus, A_m^H indicates the area taken by H-tree layout for a FBT_m. L_m^{new} indicates length of the longest link in our proposed layout for a FBT_m. Details of these calculation are shown in [7].

$$A_m^{new} = [\frac{3}{4} \times 2^{\frac{m+1}{2}} - 1] \times [\frac{9}{8} \times 2^{\frac{m+1}{2}} - 1], \quad \textit{for odd } m.$$

$$A_m^{new} = [\frac{3}{2} \times 2^{\frac{m}{2}} - 1] \times [\frac{9}{8} \times 2^{\frac{m}{2}} - 1], \quad \textit{for even } m.$$

It may be noted that for either case (m=odd or even) the area ratio over H-tree layout

$$\frac{A_m^H}{A_m^{new}} = \frac{32}{27}, \text{ for large } m$$

Similarly, *aspect ratio*=$(\frac{length}{width})$ can be found as

$$AS_m^{new} = \frac{(3/4)}{(9/8)} = \frac{2}{3}, \text{ for } odd \text{ values of } m.$$

$$AS_m^{new} = \frac{(3/2)}{(9/8)} = \frac{4}{3}, \text{ for } even \text{ values of } m.$$

$$\frac{B_m^H}{B_m^{new}} = \frac{2^{\frac{m+3}{2}} - 4}{12 \times 2^{\frac{m-5}{2}} - 4} = \frac{4}{3}, \text{ for large } odd\ m.$$

$$\frac{B_m^H}{B_m^{new}} = \frac{3 \times 2^{\frac{m}{2}} - 4}{2 \times 2^{\frac{m}{2}} - 4} = \frac{3}{2}, \text{ for large } even\ m.$$

Fig. 7 plots area comparisons of our proposed layout with H-tree layout. Fig. 8 shows comparison of longest link length over H-tree layout. Finally, Fig. 9 shows comparison of boundary leaf PEs over H-tree layout.

4. Conclusion

This paper proposes a tile based scheme for binary tree layout. The layout follows strict grid model assumptions and in this regard its performances are directly comparable to H-tree layout [3]. It may be noted that none of the recent binary tree layout schemes [1,6], which report near 100% utilization of nodes, follow grid model assumptions strictly. Our idea is to generate a tile layout for height-5 full binary tree which is area efficient than an equivalent sized H-tree layout. Based on this tile higher level tree layouts are constructed using similar recurrence as of H-tree style. Since the base case in our approach is more area compact than H-tree the overall layout for higher levels of tree continue to retain the area efficiency. We have developed equations for layout area, boundary I/O and longest link length of the proposed layout and compared the advantages over H-tree layout. It remains as a future exercise to develop increasingly area efficient handcoded layout patterns for trees of height more than five. Such patterns can be used as base cases of higher level recursive construction to yield area optimized binary tree layout.

Appendix

Length and Width Equations

Let D_m denote any dimension (length or width) of the proposed layout.

Recurrence Equation: $D_m = 2 \times D_{m-1} + 1 = 2 \times (2 \times D_{m-2} + 1) + 1$

which leads to the *expression*: $D_m = 2^i \times D_{m-2i} + \sum(2^{i-1} + 2^{i-2} + \cdots + 2^0) = 2^i \times D_{m-2i} + 2^i - 1$

Evaluation of the avove expression starts when the layout size is base-tile, $m-2i=5$, i.e., $i=\frac{m-5}{2}$

Since, $L_5 = 5$ and $W_5 = 8$ (note that the base tile is 5×8) we have

Length (for *odd* values of m): $L_m = 2^{\frac{m-5}{2}} \times 5 + 2^{\frac{m-5}{2}} - 1$

$$= \frac{3}{4} \times 2^{\frac{m+1}{2}} - 1,$$

Width (for *odd* values of m): $W_m = 2^{\frac{m-5}{2}} \times 8 + 2^{\frac{m-5}{2}} - 1$

$$= \frac{9}{8} \times 2^{\frac{m+1}{2}} - 1$$

For *even* values of m similarly we can find out

$$L_m = \frac{3}{2} \times 2^{\frac{m}{2}} - 1, W_m = \frac{9}{8} \times 2^{\frac{m}{2}} - 1$$

Boundary I/O

Base tile (laying out a FBT_5) requires 5×8 layout area and has 4 leaf PEs along the longer side while 2 leaf PEs

along the other. Let m be an odd number. The layout of FBT_m will place $k \times k$ tiles in a rectangular fashion, where $k = 2^{\frac{m-5}{2}}$. Thus, the boundary I/O of FBT_m will be

B_m^{new} = two horizontal strips of k tile borders + two vertical strips of k tile borders - 4 corner PEs

$= 2 \times k \times 4 + 2 \times k \times 2 - 4 = 12 \times k - 4$

$= 12 \times 2^{\frac{m-5}{2}} - 4$, *for odd m*

For *even* values of m width of the layout gets doubled from its immediate previous *odd* case. Hence,

B_m^{new} = two horizontal strips of k tile borders + four vertical strips of k tile borders - 4 corner PEs

$= 2 \times k \times 4 + 4 \times k \times 2 - 4 = 16 \times k - 4$

$= 16 \times 2^{\frac{m-5}{2}} - 4$, *for even m*

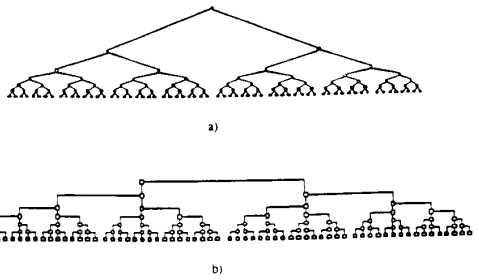

Fig. 1. Standard tree layout: a) height-7 full binary tree, b) standard tree layout.

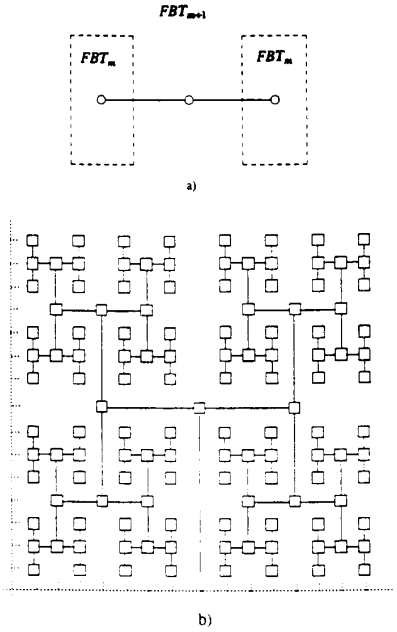

Fig. 2. H-tree layout: a) recursive construction, b) height-7 layout.

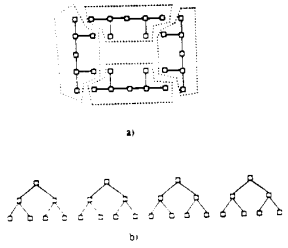

Fig. 3. Base tile pattern: a) layout of *four* FBT_3, b) four FBT_3.

Reference

[1] Gordon, D., "Efficient Embeddings of Binary Trees in VLSI Arrays", *IEEE Trans. on Computers*, Vol. C-36, Sept 1987, pp. 1009-1018.

[2] Gupta, A., K., Hambrusch, S., E., "Optimal Three-Dimensional Layouts of Complete Binary Trees", *Information Processing Letters*, 26, Oct 19, 1987, pp. 99-104.

[3] Horowitz, E. and Zorat, A., "The Binary Tree as an Interconnection Network: Applications to Multiprocessor Systems and VLSI", *IEEE Trans. on Computers*, Vol. C-30, April 1981, pp. 247-253.

[4] Sahni, S., Horowitz, E., "Fundamentals of Computer Algorithms", *Computer Science Press*, Potomac, Md., 1978.

[5] Snyder L., "Introduction to the Configurable, Highly Parallel Computer", *Computer*, Vol. 15, No. 1, 1982, pp 47-57.

[6] Youn H. Y., Singh D. Adit, "Near Optimal Embedding of Binary Tree Architecture in VLSI", *8th Int'l Conference on Distributed Computing Systems*, 1988, pp 86-93.

[7] Bhattacharya, S., Tsai, W. T., "Area Efficient Binary Tree Layout", *Technical Report, University of Minnesota*, Mar 1991.

along the other. Let m be an odd number. The layout of FBT_m will place $k \times k$ tiles in a rectangular fashion, where $k = 2^{\frac{m-5}{2}}$. Thus, the boundary I/O of FBT_m will be

B_m^{new} = two horizontal strips of k tile borders + two vertical strips of k tile borders - 4 corner PEs

$= 2 \times k \times 4 + 2 \times k \times 2 - 4 = 12 \times k - 4$

$= 12 \times 2^{\frac{m-5}{2}} - 4$, *for odd* m

For *even* values of m width of the layout gets doubled from its immediate previous *odd* case. Hence,

B_m^{new} = two horizontal strips of k tile borders + four vertical strips of k tile borders - 4 corner PEs

$= 2 \times k \times 4 + 4 \times k \times 2 - 4 = 16 \times k - 4$

$= 16 \times 2^{\frac{m-5}{2}} - 4$, *for even* m

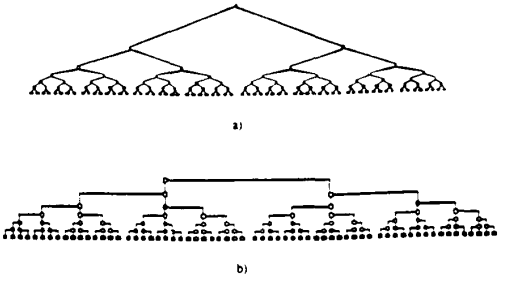

Fig. 1. Standard tree layout: a) height-7 full binary tree, b) standard tree layout.

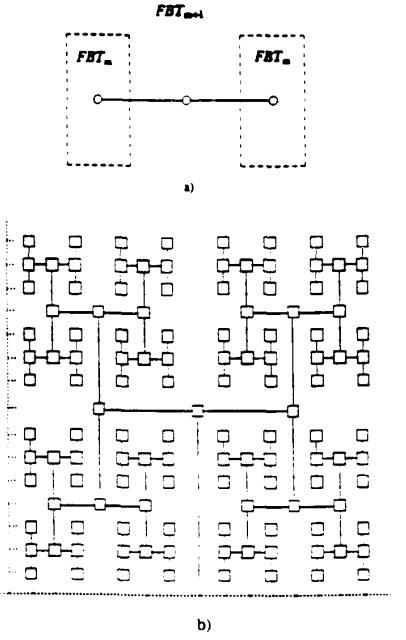

Fig. 2. H-tree layout: a) recursive construction, b) height-7 layout.

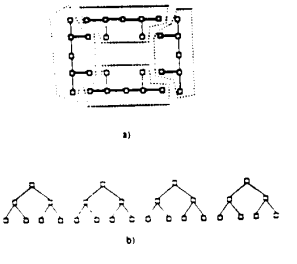

Fig. 3. Base tile pattern: a) layout of *four* FBT_3, b) four FBT_3.

Reference

[1] Gordon, D., "Efficient Embeddings of Binary Trees in VLSI Arrays", *IEEE Trans. on Computers*, Vol. C-36, Sept 1987, pp. 1009-1018.

[2] Gupta, A., K., Hambrusch, S., E., "Optimal Three-Dimensional Layouts of Complete Binary Trees", *Information Processing Letters*, 26, Oct 19, 1987, pp. 99-104.

[3] Horowitz, E. and Zorat, A., "The Binary Tree as an Interconnection Network: Applications to Multiprocessor Systems and VLSI", *IEEE Trans. on Computers*, Vol. C-30, April 1981, pp. 247-253.

[4] Sahni, S., Horowitz, E., "Fundamentals of Computer Algorithms", *Computer Science Press*, Potomac, Md., 1978.

[5] Snyder L., "Introduction to the Configurable, Highly Parallel Computer", *Computer*, Vol. 15, No. 1, 1982, pp 47-57.

[6] Youn H. Y., Singh D. Adit, "Near Optimal Embedding of Binary Tree Architecture in VLSI", *8th Int'l Conference on Distributed Computing Systems*, 1988, pp 86-93.

[7] Bhattacharya, S., Tsai, W. T., "Area Efficient Binary Tree Layout", *Technical Report, University of Minnesota*, Mar 1991.

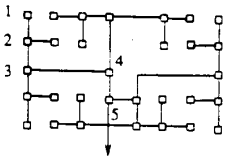

Fig. 4. Height-5 tree in (5x8) area.

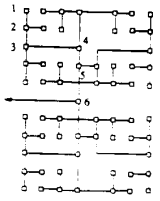

Fig. 5. Height-6 binary tree in (11x8) area.

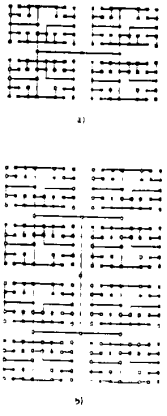

Fig. 6. Proposed layout: a) height-6 tree, b) height-7 tree.

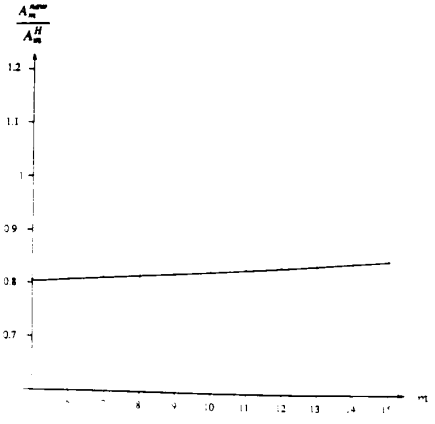

Fig. 7. Area comparison with H-tree layout.

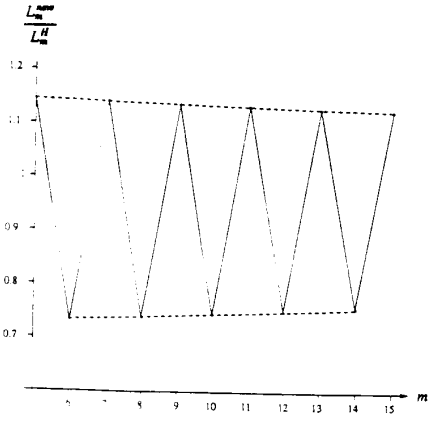

Fig. 8. Longest link length comparison with H-tree layout.

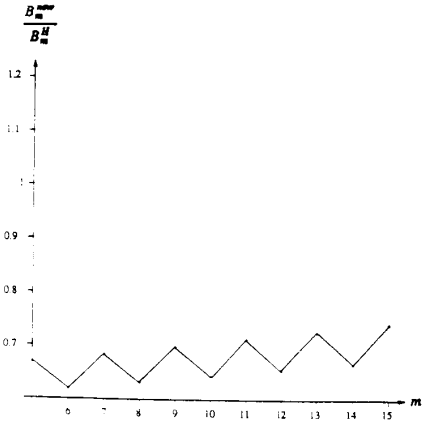

Fig. 9. Boundary I/O Comparison with H-tree layout.

Tree Height	H-Tree Layout Dimensions (length, width, area)	Our Layout Dimensions (length, width, area)
height 5	7, 7, 49	5, 8, 40
height 6	15, 7, 105	11, 8, 88
height 7	15, 15, 225	11, 17, 187
height 8	31, 15, 465	23, 17, 391
height 9	31, 31, 961	23, 35, 805
height 10	63, 31, 1953	47, 35, 1645

Table 1. Comparison of the proposed layout with H-Tree.