Quadtree Layout

Sourav Bhattacharya* Shekar Kirani
Wei-Tek Tsai
Computer Science Department, University of Minnesota, Minneapolis, MN 55455

Abstract

Quadtree data structure has been used in a number of applications. However, VLSI embedding of quadtree based parallel architecture using *grid model* has not been studied. This paper studies VLSI embedding of quadtree using grid model. H-tree layout for binary tree is extended for trivial quadtree layout, followed by developing two layout strategies for *rectangular* grids. Then, two generic layout styles (*standard layout* and *X-layout*) are proposed for higher order grids (e.g., hexagonal and octagonal grids). Base tile layout patterns are proposed for area compaction with recursive X-layout. In each case, layout dimensions and I/O bandwidth are computed. We demonstrate how the two generic layouts can be mixed to obtain higher I/O bandwidth and estimate the area sacrifice. An improved *recursive* layout mixing strategy is proposed.

1 Introduction

The quadtree data structure is extensively used in representing two dimensional data in applications, e.g., image processing, VLSI embedding, graphics. Since its original inception by Samet [15] quadtree data structure has been used for polygonal decompositions [16], storage of masks [8] for VLSI layouts, supporting region queries [11] etc. [10] addressed the problem of finding planar tessellations which yield the quadtree data structure. [13] compares between quadtree and 4-d trees in terms of supporting region queries. Due to its frequent applicability in 2-dimensional applications *quadtree* structure based interconnection networks appear promising. Similar trend can be observed with binary tree, where its popularity led to binary tree architecture development and VLSI embedding. Though a *ternary tree* is less popular, its VLSI layout has been investigated [12]. However, to our knowledge no study has been made on quadtree embedding on VLSI using *Grid Model.*

Geometrical aspects of tree (with branching factors = k^2 and k^2-1) layout have been considered in [1]. With k=2, this result can be used for generating quadtree layout. However, geometric layouts concern with the orientations of connecting wires only, unlike *grid model* which has further restrictions [7]. Similarly, geometric layouts does not consider issues such as 1) sharing a channel (edge) by multiple tree edges and consequent data pipeline overhead, 2) boundary I/O of the layout. These make VLSI layout using grid model different from the methods proposed in [1].

This paper studies VLSI layout for quadtree based interconnection networks using *Grid Model.* We consider *full quadtrees* only (analogous to *full* binary tree). Let a full quadtree (binary tree) of height-k be denoted as FQT_k (FBT_k). Quadtree is a degree=5 graph (*degree* of a graph equals the highest node *degree* in the graph) and hence its layout on strict rectangular grid (degree=4) is not straight forward. We use the concept of link repetition and/or node bypass [6] to obtain quadtree layout on strict rectangular grid. First, we demonstrate simple extension of H-tree layout for binary trees to obtain quadtree layout. Then we develop two other efficient layout schemes for *rectangular* grid model.

For higher order grids (e.g., hexagonal, octagonal) we propose two generic quadtree layout styles: *standard layout* and *X-layout.* The former minimizes area and I/O bandwidth, while the latter maximizes both these layout attributes. Both allow recursive layout construction, starting with a base pattern. We propose base X-layout patterns (i.e., tiles [18,2]) for hexagonal and octagonal grid models. Layout area, border I/O, length of longest link, aspect ratio and rate of data pipelining (due to link repetition) are used to characterize our proposed layout techniques.

One important issue in VLSI layout for trees is its *boundary I/O* which indicates the number of leaf processing elements (PEs or nodes) that have boundary access in the layout construction. Recent works on binary tree VLSI layout [3,14,17] have focussed on this issue. We develop quadtree layouts with emphasis on *area-I/O tradeoff.* We demonstrate how mixing the two generic layout styles (i.e., standard-layout and X-layout) strategies would allow higher I/O-bandwidth at

*email: sourav@cs.umn.edu, phone: (612) 626-7508

0-8186-2610-0/92 $3.00 © 1992 IEEE

Figure 1: Quadtree and its trivial layout from Binary Tree Layout.

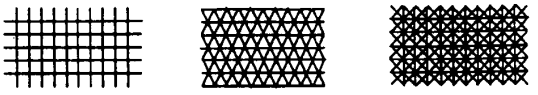

Figure 2: Underlying grid patterns used in our layout.

the expense of moderate area sacrifice. We propose a recursive layout mixing strategy which is more efficient than existing *single stretch* layout mixing approaches.

Section 2 introduces quadtree structure and shows a trivial approach of using H-tree layout (for binary trees) for quadtree layout. Section 3 proposes two layout styles using strict rectangular (degree=4) grid model. Section 4 develops two generic layout styles. Section 5 uses these two layout styles with hexagonal and octagonal grids. Section 6 introduces area-I/O tradeoff for these layouts and develops layout mixing strategies. Section 7 concludes the paper.

2 Preliminaries

Quadtree is a tree structure with each intermediate node having four children nodes. A full quadtree of height k (FQT_k) has all its leaf nodes (k-1) distant from the root. Fig.1a shows a FQT_3. Note that the number of nodes in a FQT_k is $\frac{4^k-1}{3}$. Related properties of quadtree data structure can be found in [15].

Extending Binary Tree Layout for Quadtrees: Following the generic H-tree pattern (for binary tree layout), a FBT_{2k-1} with (2^{2k-1}-1) nodes can be laid out in nearly 2^{2k} area [9]. If every alternate level (starting with lowest second level) internal nodes in this tree are considered as a switch, i.e., a forwarding connection, then a FQT_k is obtained. Fig.1b shows a FQT_2 constructed from FBT_3. Fig.1c shows the layout using H-tree style.

We refer to this layout as $layout_1$. FQT_k includes $\frac{4^k-1}{3}$ PEs. Thus, $\frac{area}{nodes-mapped}$ in this trivial layout scheme is $= \frac{2^{2k}}{(4^k-1)/3}$ which nearly equals 3. Let L_k, W_k, Len_k, A_k, AS_k and *rate* denote the length, width, length of the longest link (indicates the maximum wire length between any two nodes), area, aspect ratio ($\frac{length}{width}$) and data pipeline overhead of a FQT_k layout. Then the following relationships hold for $layout_1$.

- $L_k = W_k = (2^k\text{-}1)$
- $A_k = L_k \times W_k = (2^k - 1)^2$
- $Len_k = 2^{k-1}$, $AS_k = 1$, $rate = 2$

3 Rectangular Grid Layout

A popular model for laying out VLSI circuits is to view the circuit as a bounded degree graph G, where the nodes correspond to PEs and edges to wires. Let G_d denote the maximum node degree in G. VLSI layout using grid model selects a regular and sufficiently large grid with degree $\geq G_d$. The task of laying out G becomes embedding G onto the grid. A rectangular grid model is an infinite grid with degree=4, a hexagonal grid with degree=6 and an octagonal grid with degree=8. Fig.2 shows these grids. We propose layouts using rectangular grid in this section. Following sections develop layouts using the other two grid patterns. Note that the maximum node degree in FQT_k (=5) is higher than that in a strict rectangular grid model (=4).

Alleviating node degree relaxation: Given this degree restriction of rectangular grids, the quadtree layout problem can be approached in two ways: using *embedding with repetition* or with an assumption that a grid intersection point can simultaneously be a PE *and* part of an edge [6]. In the former case, two (or more) links of the tree are mapped to a single track-path in the grid. The idea is to share one physical track by multiple (p) logical links (of the tree) in a time-division multiplexed manner. This introduces a delay factor, when data pipeline is executed over the tree structure.

The second relaxation approach has been used in binary tree layout [6]. The idea is to let PEs (mapped to grid intersection points) perform simultaneously as a tree-node *and* a forward-node (as part of a sequence of edges). We refer to this relaxation as 'bypass-PEs'. [6] did not introduce any delay factor on account of this simultaneity requirement of a PE. Along similar lines our analysis also excludes additional delay factors due to this task overlap.

3.1 Layout using link repetition

In this approach one physical track in the rectangular grid model maps at most two edges of the quadtree. Some edges of the quadtree uniquely map to one physical track, the remaining edges share common track in a pair-wise manner. Fig.3a shows an example, where PEs marked 'A' and 'B' share one common track in connect-

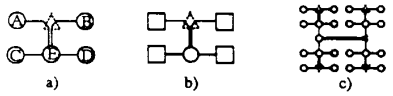

Figure 3: Layout scheme in strict rectangular grid model: a) base pattern, b) inductive construction, c) height-3 quadtree layout.

ing to the PE marked 'E'. In actual implementation the common track is time multiplexed between these two logical links. Dotted triangle in Fig.3a indicates a switch which performs necessary time multiplexing, i.e., can change the effective connection between two alternates. Note that two other nodes, marked 'C' and 'D', have direct and non-overlapped connection to root node 'E'.

Fig.3a serves as a base pattern for laying out FQT_2. Fig.3b shows the recursive tree construction pattern. Every rectangular block denote a FQT_k and circular block denote a single PE as the root node of FQT_{k+1}. Fig.3c shows sample layout of FQT_3. We refer to this layout as $layout_2$. Note that double line along an edge indicate two logical links being mapped. From layout geometry we obtain the following recurrence equations (with base case $L_1 = W_1 = 1$) as:

- For *even* k, $L_k = 2L_{k-1}$, $W_k = 2W_{k-1} + 1$.
- For *odd* k, $L_k = 2L_{k-1} + 1$, $W_k = 2W_{k-1}$.

Solving these recurrence relations we get parameters for $layout_2$ as:

- For *even* k, $L_k = \frac{2^{k+1}-2}{3}$, $W_k = \frac{5\times 2^{k-1}}{3} - \frac{1}{3}$
- For *odd* k, $L_k = \frac{2^{k+1}-1}{3}$, $W_k = \frac{5\times 2^{k-1}}{3} - \frac{2}{3}$
- $len_k \approx \frac{5\times 2^{k-2}}{3} = 0.41 \times 2^k$
- $AS_k \approx (4/5)$, $rate = 2$

This leads to $\frac{area}{nodes-mapped}$ as $= \frac{(5\times 4^k/9)}{(4^k-1)/3} = 1.66$. Fig.17 summarizes the characteristic of $layout_2$.

3.2 Layout using Bypass PEs

In this model a grid intersection point can serve both as a node of the tree and a link-bypass (similar to the assumption in [6]). Fig.4a shows the base layout pattern for a height-2 quadtree. Fig.4b shows the inductive construction pattern, where a rectangular block indicates FQT_k and circle indicates the root node for FQT_{k+1}. Bypass connections are shown using links passing around the node. Fig.4c shows an example FQT_3 layout. We refer to this layout as $layout_3$. From the layout geometry (with $L_1 = W_1 = 1$) we obtain

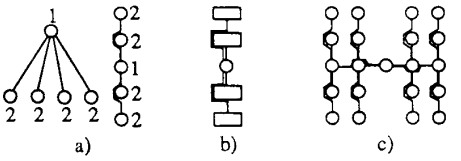

Figure 4: Grid model quadtree layout with bypass PEs: a) base pattern, b) inductive construction, c) illustrative FQT_3 layout.

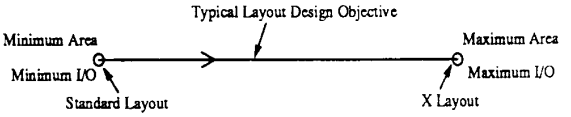

Figure 5: Area-I/O Tradeoff in Quadtree Layout Design.

- For *even* k, $L_k = 4 \times L_{k-1} + 1$, $W_k = W_{k-1}$
- For *odd* k, $L_k = L_{k-1}$, $W_k = 4 \times W_{k-1} + 1$

Solving these recurrence relations we get parameters for $layout_3$ as:

- For *even* k, $L_k = \frac{4\times(2^k-1)}{3} + 1$, $W_k = \frac{2^k-1}{3}$
- For *odd* k, $L_k = W_k = \frac{2^{k+1}-1}{3}$
- len_k (worst case) $\approx \frac{2^{k+1}}{3}$
- $AS_k \approx 1$ (4) for k = odd (even), $rate = 2$

This leads to $\frac{area}{nodes-mapped}$ as $= \frac{(4\times 4^k/9)}{(4^k-1)/3} = 1.33$. Fig.17 summarizes the characteristics of $layout_3$.

4 Generic Layouts

In this section we propose two generic layout schemes for higher degree grids. Rationale behind these two layout styles are as follows. Intuitively, *layout area* and *boundary I/O* metrics are orthogonal: minimizing area reduces boundary I/O, maximizing boundary I/O increases the area. Layout design objective is to arrive at a tradeoff, with adequate I/O support and moderate area overhead. Fig.5 shows this, where the leftmost end of the scale indicates maximum area compact (with least I/O) layout while the rightmost end indicates large layout area (with full I/O). A typical design objective is to arrive at an intermediate choice between the two extremes ends.

Note that the leftmost end of the scale in Fig.5 compare to the objective behind *H-tree* layout [9] of binary trees, while the rightmost extreme compare to the *standard tree* layout [5]. Along similar lines, we develop two

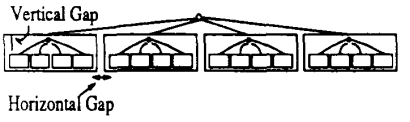

Figure 6: Generic Standard Quadtree Layout.

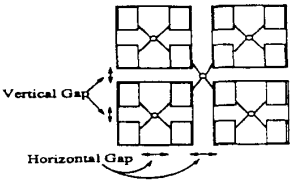

Figure 7: Generic X-style Quadtree Layout.

generic layout strategies for the two extreme ends of Fig.5. These layouts are for degree=6 or higher grids. We develop area, I/O equations for each. Then we demonstrate how these two layouts can be mixed to flexibly arrive at intermediate design choices.

4.1 Standard layout (4^k-k)

This layout is analogous to the *standard tree* layout for binary tree. Idea behind this layout is to layout all the leaf nodes of FQT_k horizontally one after another. 2^{k-1} leaf nodes in FQT_k require a horizontal spacing of the order $O(2^{k-1})$. Internal nodes of FQT_k are laid out using vertical slots: involving a total of k slots, one for each level.

Fig.6 shows *standard quadtree* layout construction. Smallest rectangular boxes indicate FQT_j layout using which FQT_{j+1} and FQT_{j+2} are constructed. Layout construction strategy is recursive: four independent layouts are placed horizontally one after another and connected to a parent node. Connections from child tree to the parent node are shown using straight lines, which in a specific grid model would follow the grid orientations. Depending on the underlying grid pattern (i.e., the geometric flexibility of track orientations) certain horizontal (vertical) gap = g_h ($=g_v$) is required. The layout dimensions follow:

$W_k = 4 \times W_{k-1} + 3 \times g_h$, $L_k = L_{k-1} + g_v + 1$, with L_3=2+g_v, W_3=16+3×g_h.

Objective behind this layout style is to maximize I/O bandwidth at the expense of area overhead. This layout approach has *all* leaf PEs along one horizontal boundary, i.e., 100% I/O bandwidth. However, it involves $O(k \times 4^k)$ area.

4.2 X-layout

This layout is analogous to the *H-tree* layout for binary tree [9]. The idea is to fold the layout in each level and obtain area compaction. Fig.7 shows the recursive layout construction method. Each smallest rectangle denote a FQT_j layout. Using four such layouts (placed similar to the letter 'X') we obtain a FQT_{j+1} layout and so on. Tree edges are shown using solid straight lines, which in specific grid model would follow the grid orientations. Let g_h (g_v) denote the horizontal (vertical) gap required with respect to a specific underlying grid. The layout dimensions follow:

$W_k = 2 \times W_{k-1} + g_h$, $L_k = 2 \times L_{k-1} + g_v$, with L_2=2+g_v, W_2=2+g_v.

The objective behind *X-tree* layout is to obtain maximum area compaction, at the expense of I/O bandwidth. This layout requires $O(4^k)$ area, however I/O bandwidth is sacrificed at every level of recursive construction.

5 Generic Layouts with Higher Order Grids

We apply the two generic layout styles to hexagonal (degree=6) and octagonal (degree=8) grids. In either case, we develop expressions for L_k, W_k, len_k, AS_k. In addition, for *X-layout* we estimate the I/O bandwidth. Note that with *standard* layout I/O bandwidth is maximum (100%).

X-layout follows a recursive construction, starting with a default FQT_1 base case. However, for smaller heights (e.g., 3 or 4) it is area-economical to develop compact, hand-generated layout patterns. The area required by these tile patterns are less than the corresponding (FQT_3 or FQT_4) X-layout generated starting with L_1=W_1=1. This results in a constant factor area improvement. Higher level FQT_k ($k \geq 3$) construction follows identical recurrence relations for *X-layout*, however, we start with base case as one of these tile patterns [2]. Consequently, the constant factor improvement is continued at all level of layout construction.

5.1 Hexagonal Grid

We have chosen a (degree=6) grid constructed using three set of infinitely extending parallel lines: S_1, S_2 and S_3. Lines in S_1 are parallel to the x-axis. Lines belonging to S_2 (S_3) have a forward angle = $tan^{-1}(2)$ [(180° - $tan^{-1}(2)$)] with those in S_1. Note that our hexagonal grid is different from the degree=6 grid used in [1]. We have chosen a (degree=6) grid with infinitely extendible characteristics without conforming to the geometric definition of hexagonal grids.

5.1.1 Standard Layout

Fig.8 shows the *standard layout* for quadtree on a hexagonal grid. Note that, g_h=0 and g_v=1. Thus we

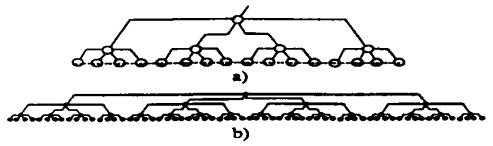

Figure 8: Standard Hexagonal Layout for Quadtree: a) FQT_3, b) FQT_4.

get: [$W_k = 4 \times W_{k-1}$] and [$L_k = L_{k-1} + 2$]. However, at the lowest level we notice that the gap g_v is unnecessary. From these equations we obtain FQT_k (with L_3=4, W_3=16) layout dimensions as:

- L_k = 2k-2, $W_k = 4^{k-1}$, $A_k = (2k\text{-}2)4^{k-1}$
- $len_k = 2^{2k-3}$, $AS_k = \frac{4^{k-1}}{2k-2} \approx \frac{2^{2k-3}}{k}$
- I/O bandwidth = 100%, $rate = 1$

5.1.2 X Layout

Fig.9 shows *X-layout* for quadtree on hexagonal grid. Fig.9b shows the basic mechanism of inductive layout construction. Layout construction starts with a base case: L_2=3, W_2=2 (Fig.9a). Fig.9c shows a larger construction, e.g., FQT_4. Following the layout geometry we notice $g_v = 1$ and $g_h = 2$. These make the layout equations as: [$W_k = 2 \times W_{k-1}+2$] and [$L_k = 2 \times L_{k-1}+1$].

Let io_k = number of leaf PEs (in FQT_k) with boundary access, and let $border_k$ denote the number of leaf PEs in FQT_k along one (of the four) layout boundaries. Then, $io_k = 4 \times border_k$ - 4, since the corner nodes are counted twice. From Fig.9a and Fig.9b we have $border_2 = 2$, $border_i = 2 \times border_{i-1}$. The recurrence solves as $border_k = 2 \times 2^{k-2} = 2^{k-1}$. Therefore, $io_k = 2^{k+1}$-4.

- $L_k = 2^k$-1, $W_k = 2^k$-2, $A_k \approx 4^k$
- $len_k = 2^{k-1}$, $AS_k = \frac{2^k-1}{2^k-2} \approx 1$, $rate$=1
- I/O bandwidth = $\frac{2^{k+1}-4}{4^{k-1}} = \frac{2^{k+1}-4}{2^{2k-2}} \approx \frac{1}{2^{k-3}}$

5.1.3 Tile Layout Patterns

X-layout starts with a base case as shown in Fig.9a. Layout dimension for FQT_3 is (6 × 7), i.e., an area = 42 units. However, by hand-coding it is possible to generate more area efficient tile layouts for FQT_3. Similar strategies have been noted with binary tree layout [2,18], where compact patterns for small height binary trees are developed and used as base cases with recursive H-tree construction style. These tiles are used

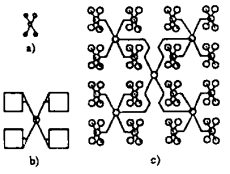

Figure 9: X-style Hexagonal Layout for Quadtree: a) Base case FQT_2, b) Inductive construction, c) FQT_4.

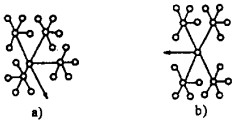

Figure 10: Area Efficient Base Tile Pattern for X-Layout: a) Area = 5.5 x 6, b) Area = 4.5 x 7.

for higher level X-layout constructions. Fig.10a and Fig.10b show two tile patterns requiring layout dimensions as (5.5×6) and (4.5×7) respectively. These result in area reduction by ≈ (42/33). Area equations shown above for X-layout would apply except for this constant multiplier factor improvement. I/O bandwidth is also similarly affected.

5.2 Octagonal Grid

Underlying octagonal grid is an infinitely extendible graph with degree=8. Four sets of lines S_1, S_2, S_3 and S_4 constitute the grid. Lines in set S_1 are horizontal (i.e., along x-axis), lines in S_2 (S_3 and S_4) have a forward angle = 45° (90° and 135°) with those in S_1.

Standard Layout and X-layout: Fig.11 shows *standard layout* for quadtree using octagonal grid model. Layout attributes for Fig.11 are identical to those in Fig.8. Fig.12 shows *X-layout* for quadtree using octagonal grid model. Layout attributes for Fig.12 are identical to those in Fig.9.

5.2.1 Tile Layout Patterns

Following the *X-layout* pattern for octagonal grids, FQT_3 (FQT_4) requires an area = (6 × 7) = 42 units [(14 × 15) = 210 units]. In this subsection we develop

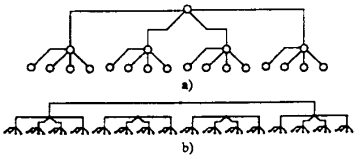

Figure 11: Standard Octagonal Layout for Quadtree: a) FQT_3, b) FQT_4.

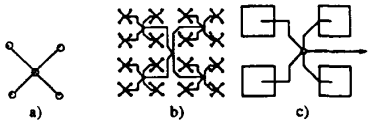

Figure 12: X-style Octagonal Layout for Quadtree: a) Base case FQT_2, b) FQT_4, c) Inductive construction.

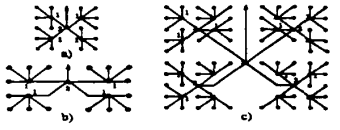

Figure 13: Compact Tile Layout for Quadtree: a) (5x5) layout for FQT_3, b) (4x7) layout for FQT_3, c) (10x11) layout for FQT_4.

handcoded compact tile patterns for FQT_3 and FQT_4. These provide a constant factor area-improvement over the generic X-layout. These tiles can be used with higher level construction identical to X-layout construction.

Fig.13a shows FQT_3 layout using $(4 \times 7) = 28$ unit area (i.e., an area reduction factor = $\frac{42}{28} = 1.5$). Fig.13b shows another FQT_3 layout using $(5 \times 5) = 25$ unit area (with reduction factor = $\frac{42}{25}$=1.68). Note that Fig.13b is area minimum for FQT_3 layout, since the later has 21 nodes and any further grid size reduction from (5×5) would imply a smaller grid size than the number of nodes in FQT_3.

Using this (5×5) tile as FQT_3 and X-style layout construction, we can generate a FQT_4 layout requiring $(12 \times 11 = 132)$ layout area. Fig.13c shows a hand-generated compact tile layout for FQT_4 requiring an area = $(10 \times 11 = 110)$. This yields an *additional* area advantage by a factor = $\frac{132}{110} = 1.2$. Note that with respect to the generic X-layout (as shown in Fig.12) this yields an area reduction factor = $\frac{210}{110} = 1.9$.

6 Area-I/O Tradeoff

Fig.5 presented area-I/O characteristic of *standard layout* and *X layout*. The former excels in area compaction but sacrifices in I/O, while the latter maximizes I/O bandwidth at the expense of area overhead. A typical layout design objective is to arrive at an intermediate stage of these two extreme ends. To achieve this, we mix the two layout strategies in a proportion as required by the I/O bandwidth requirement and tolerable area overhead.

Similar trend can be noticed in binary tree layout also. The two generic layout schemes for binary tree

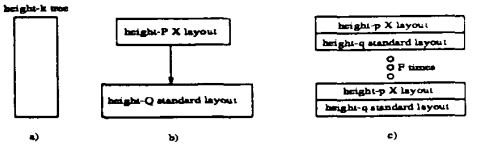

Figure 14: Quadtree Layout Mixing Strategies: a) Basic Tree, b) Single-stretch mixing, c) Recursive mixing.

(standard layout and H-layout) are mixed to arrive at intermediate design solutions. [3,7,14,17] proposed single-stretch layout mixing style, i.e., having a continuous range of *standard* and then building *H-layout* on top (or vice versa). Recursive (and fine grained) layout mixing strategies have been proposed recently with an order of magnitude improved performance [4].

This section develops dimensions for FQT_k layout, where Q levels of the tree are constructed using *standard layout* style and P $(= k-Q)$ levels are constructed using *X-layout* style. First we use single-stretch layout mixing, e.g., constructing FQT_Q *standard* layout as a basic block and using those as building elements in FQT_P *X-layout* construction. We estimate the I/O bandwidth improvement and area overhead. Then, we use recursive mixing strategy and obtain similar parameters towards layout dimensions. The idea is to form a composite pattern of FQT_{p+q} similar to the single-stretch mixing strategy. Then we repeat this pattern F times, where $P = F \times p$ and $Q = F \times q$. Fig.14 shows the idea behind these two strategies.

6.1 Layout Mixing - Single Stretch

Basic block: Basic building block in this scheme is a *standard* layout for FQT_Q. Following the equations for layout dimensions in Section 5.1.1 (and also 5.2) we have: [$L_Q = 2Q\text{-}3$, $W_Q = 4^{Q-1}$]. Note that all 4^{Q-1} leaf nodes are available along one boundary of the layout. We treat this FQT_Q layout as a tile for higher level X-layout construction.

Higher level X-layout construction: We construct a FQT_P using the basic block FQT_Q layout (described above) as leaf nodes. The recurrence relations over layout length (width) remains unchanged, however, in this case the base case becomes FQT_Q layout with [$L_Q = 2Q\text{-}2$, $W_Q = 4^{Q-1}$]. Similarly, the basic block FQT_Q layout has 4^{Q-1} leaf nodes along one particular border. Every higher level quadtree construction using *X-layout* style doubles the number of leaf blocks along two horizontal layout boundaries. Solving these we obtain:

- $W_{Q+P} = 2^P \times (2^{2Q-2}+1) - 2$, which is $O(2^{P+2Q})$.

Q	Sub-block Dimension	P	Total Layout Dimension	Boundary I/O
1	1 x 1	9	511 x 510	2020
2	2 x 4	8	383 x 766	2532
3	4 x 16	7	319 x 1150	4348
4	8 x 64	6	223 x 2110	8316
5	8 x 28	5	127 x 4126	16444
6	10 x 1024	4	87 x 8206	32796
7	12 x 4096	3	51 x 16390	65548
8	14 x 16384	2	29 x 32770	131076
9	16 x 65536	1	16 x 131072	262144

Figure 15: Area-I/O Tradeoff using Single-stretch Layout Mixing Strategy.

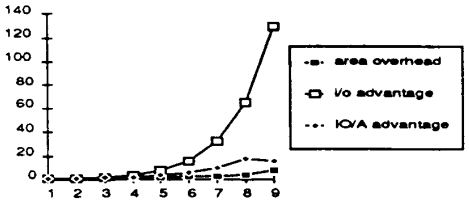

Figure 16: Overall Performance using Single-stretch Layout Mixing Strategy.

- $L_{Q+P} = (Q-1) \times 2^P$ - 1, which is $O(Q \times 2^P)$.
- $border_{Q+P} = 2^{2Q+P-1} + 2^{P+1}$ - 4.

Note that the choice of Q (and hence P) from a given tree height k is guided by the I/O bandwidth requirement. Higher I/O requirement leads to larger value of Q. Fig.15 shows different choices of Q for FQT_{10} and the corresponding layout dimensions along with the available I/O bandwidth.

We mark the pure *X-layout* as basis for area-overhead and I/O-improvement comparison. Mixing *Standard* layout increases the area overhead and provides added I/O bandwidth. For each design choice, we compute the area-overhead as the ratio of layout area to the basis layout area (pure X-layout). Similarly, we estimate the I/O-improvement factor as the ratio of current layout boundary I/O to the layout boundary I/O available at the basis X-layout. Then, we develop a composite metric: 'IO/A' = $\frac{I/O-improvement}{area\ overhead}$. Fig. 16 shows these metrics.

6.2 Layout Mixing - Recursive

The above layout mixing strategy uses one continuos stretch of ***standard*** layout style followed by another continuous stretch of *X-layout* style. This resembles a coarse grain layout mixing. The idea behind recursive layout mixing is to mix these two layout styles at a finer grain and then repeat the composite layout pattern until the desired tree-height is obtained. This idea was introduced [4] in the context of binary tree layout resulting in an order of improvement in layout area. We apply similar concept with *X-layout* for quadtree and demonstrate the advantages.

Let $X_m(a \times b)$ $[ST_m(a \times b)]$ denote a function indicating height-m quadtree layout construction using *X-layout* [*Standard layout*] style using $(a \times b)$ layout as basic block (i.e., as a leaf). Then the above *single-stretch* layout mixing strategy can be described as $Layout_{single\ stretch\ mixed} = X_P(\ ST_Q(1 \times 1)\ )$. Let F be an integer such that $P = F \times p$ and $Q = F \times q$. Then the idea behind FQT_{P+Q} construction using recursive layout mixing can be described as:

$Layout_{recursively\ mixed} = X_p(\ ST_q(\ X_p(\ ST_q(\ \ldots\ (\ X_p(\ ST_q(1 \times 1)\ )\ )\ )\ )\ )\ )$, where the recurrence is repeated F times.

It can be shown that the boundary I/O obtained using *single-stretch* layout mixing is the same in $Layout_{recursively\ mixed}$, provided the values of P and Q remain the same in either cases. However, the area needed by $Layout_{recursively\ mixed}$ is reduced by a factor F compared to the $Layout_{single\ stretch\ mixed}$. This implies that (1) the same I/O bandwidth enhancement can be done using the recursive style with an order of magnitude (to F) reduced area-overhead, (2) the area-overhead (i.e., price paid) for a certain additional I/O bandwidth (i.e., benefit) is F times less if the recursive layout mixing style is used.

Above claims regarding the benefits of recursive layout construction have been proved in [4] *for binary tree layout*. The proofs can be augmented for *quadtree layout* also in a similar manner establishing $O(F)$ improvement. Our focus in this paper is to derive the layout dimensions for recursively mixed style. Then we compare its performance with the single-stretch layout dimensions. Note that the I/O bandwidth of the recursively mixed layout would be identical to the rightmost column in Fig.15 (proof in [4]). Thus, we do not compare the boundary I/O of both the approaches. We demonstrate the area advantages only. Following X-layout and standard layout construction styles, we have

- $X_p(a \times b) = [2^p \times a + 2^p - 1] \times [2^p \times b + 2^{p+1} - 2]$
- $ST_p(a \times b) = [a + 2 \times p] \times [4^p \times b] = [a + 2 \times p] \times [2^{2p} \times b]$

A generic recurrence equation [$V_i = m \times V_{i-1} + n$] with V_0=1 solves as $V_F = m^F + n \times \frac{m^F - 1}{m-1}$. We use this result in proving the area advantage of the recursive layout scheme. From the above two relations, we have

$X_p(\ ST_q(\ a \times b\ )\ ) = [2^p(a + 2q + 1)\text{-}1] \times [2^p((2^{2q} \times b) + 2) - 2]$

Let l_i (w_i) denote the layout length (width) after i-th level of recurrence using the $X_i(\ ST_j(\ a \times b\ )\ )$ composite pattern. Then the following recurrences hold for l_i (w_i)

	Layout 1	Layout 2	Layout 3	Standard Layout	X-layout
Area	3	1.66	1.33	1.5k - 2.25	3
Pipeline Delay	2	2	2	1	1
Base Grid	Rectangular	Rectangular	Rectangular	Hexagonal/ Octagonal	Hexagonal/ Octagonal

Figure 17: Overall Performance Comparison: FQT_k Layout.

- $l_i = 2^p \times l_{i-1} + q \times 2^{p+1} + 2^p$ - 1
- $w_i = 2^{p+2q} \times w_{i-1} + 2^{p+1}$ - 2

In the single stretch strategy, we have $p = P$, $q = Q$ and $F = 1$. This yields $L_{P+Q} = 2^P(1 + 2Q + 1)$ - 1, which nearly equals $2Q \times 2^P$. Similarly, $W_{P+Q} = 2^P \times (2^{2Q} + 2)$. For recursively mixed layout, solving the above recurrences we get

- $l_F = (2^p)^F + (q \times 2^{p+1} + 2^p - 1) \times \frac{(2^p)^F - 1}{2^p - 1} \approx 2^P + 2^{p(F-1)}(q \times 2^{p+1} + 2^p - 1) \approx 2^P + 2^P \times (q \times 2 + 1) = 2^P \times (1 + q \times 2 + 1) = 2^P \times (2q + 2)$.
- $w_F = (2^{p+2q})^F + 2^{p+1} \times \frac{(2^{p+2q})^F - 1}{2^{p+2q} - 1} \approx 2^{P+2Q} + 2^{p+1} \times (2^{p+2q})^{F-1} = 2^{P+2Q} + 2^{P+2Q} \times \frac{2}{2^q} = 2^{P+2Q} \times (1 + \frac{1}{2^{q-1}}) \leq 2 \times 2^{P+2Q}$.

Therefore, ratio of layout lengths of single-stretch mixed and recursively mixed styles can be computed as $\frac{L_{P+Q}}{l_F} \approx \frac{2^P \times (2Q+1)}{2^P \times (2q+2)} \approx \frac{Q}{q} = F$.

Similarly, ratio of layout widths of single-stretch mixed and recursively mixed styles can be computed as $\frac{W_{P+Q}}{w_F}$. Now, from the above expressions for W_{P+Q} and w_F, we observe that both are greater than 2^{P+2Q} but less than $2 \times 2^{P+2Q}$. Hence, their ratio must be $\leq$ 2.

In other words, the recursively mixed layout has length $O(F)$ times shorter and width within a factor of 2 in comparison to the single-stretched mixed layout. Thus, layout area of the former is F times smaller than the latter. This proves our claim regarding area advantage of the recursively mixed layout compared to the single-stretched mixed layout. Since, both the layouts have identical I/O bandwidth, the former is an order (to the degree of recurrence) of magnitude improved version of the latter.

7 Conclusion

Quadtree data structure is popular in many applications and interconnection network using 4-ary tree structure is promising. Geometric layout strategies for general tree structures (with k^2 or k^2-1 branching factor) have been proposed [1], which can be used for quadtree layout (k=2). But to our knowledge VLSI embedding and related issues for quadtree has not been considered. This paper proposes several layout schemes for VLSI embedding of quadtree using grid model. Three different grids (rectangular, hexagonal and octagonal) are considered. In each case, we derive layout attributes (e.g., dimensions, pipeline delay factor). Finally, this paper focuses on I/O bandwidth of quadtree layout and proposes two different layout mixing strategies for enhanced I/O bandwidth.

Reference

[1] Ahuja, N., "Efficient Planar Embedding of Trees for VLSI Layouts", *Computer Vision, Graphics and Image Processing*, Vol. 34, 1986, pp 189-203.

[2] Bhattacharya, S., Tsai, W.T., "Area-Efficient Binary Tree Layout", *Proc. First Great Lake Symposium on VLSI*, Mar 1991, pp 18-24.

[3] Bhattacharya, S., Choi, Y.H., Tsai, W.T., "I/O Bound Binary Tree Layout", *Proc. First Great Lake Symposium on VLSI*, Mar 1991, pp 31-36.

[4] Bhattacharya, S., Tsai, W.T., "Recursive Binary Tree Layout Mixing", *Tech-Report 91-12, Computer Science Dept., University of Minnesota*,

[5] Brent, R., P. and Kung, H., T., "On the Area of Binary Tree Layouts", *Information Processing Letters*, vol 11, Aug 29, 1980, pp 46-48.

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

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

[8] Hsiao, P. Y., Feng. W. S., "New Algorithms Based on a Multiple Storage Quadtree for Hierarchical Compaction of VLSI Mask Layout", *Computer Aided Design*, Vol. 22, No. 2, Mar 1990, pp 74-80.

[9] 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.

[10] Mazumder, Pinaki, "Planar Decomposition for Quadtree Data Structure", *Computer Vision, Graphics, and Image Processing*, Vol. 38, 1987, pp. 258-274.

[11] Pauw, De Wim, Weyten, Ludo, "Multiple Storage Adaptive Multi-Trees", *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 9, No. 3, Mar 1990, pp. 248-252.

[12] Pinter, S. S., Wolfstahl, Y., "Embedding Ternary Trees in VLSI Arrays", *Information Processing Letters*, Vol. 26, Dec 1987, pp. 187-191.

[13] Pitaksanonkul, A., Thanawastien, S., Lursinsap, C., "Comparisons of Quad Trees and 4-D Trees: New Results", *IEEE Trans. on Computer-Aided Design*, Vol. 8, Nov 1989, pp. 1157-1164.

[14] P. Czerwinski, V. Ramachandran, "Optimal VLSI Graph Embeddings in Variable Aspect Ratio Rectangles", *Algorithmica*, 1988, Vol. 3, pp 487-510.

[15] Samet, H., "The Quadtree and Related Hierarchical Data Structures", *ACM Computing Survey*, Vol. 16, 1984, pp. 187-260.

[16] Tanimoto, S. L., Pavlidis, T., "A Hierarchical Data Structure For Picture Processing", *Computer Graphics, Image Processing*, Vol. 4, 1975, pp. 104-119.

[17] Tzeng, N.F., Bhattacharya, S., "Binary Tree Layouts with Adequate I/O Support", *Proc. IEEE Circuits and Systems*, 1990.

[18] Youn, H. Y., Singh, A. D., "On Area Efficient and Fault Tolerant Tree Embedding in VLSI", *Int'l Conf. on Parallel Processing*, 1987, pp 170-177.