A Unified Approach to Topology Generation and Optimal Sizing of Floorplans

Partha S. Dasgupta, Susmita Sur-Kolay, and Bhargab B. Bhattacharya

***Abstract*— Existing algorithms for floorplan topology generation by rectangular dualization usually do not consider sizing issues. In this paper, given a rectangularly dualizable adjacency graph and a set of aspect ratios of the modules, a topology which is likely to yield an optimally sized floorplan, is produced first in a top-down fashion by an AI-based search technique with novel heuristic estimates based on size parameters. It is shown that for any rectangular graph, there exists a feasible topology using only either straight or Z-cutlines recursively within a bounding rectangle. The significance of this result is four-fold: 1) considerable acceleration of the heuristic search, 2) topology generation with minimal number of nonslice cores, 3) guaranteed safe routing order without addition of pseudo modules, and 4) design of an efficient bottom-up heuristic for optimal sizing. Experimental results show that this integrated method elegantly solves floorplan optimization problem for general including inherently nonslicible adjacency graphs.**

***Index Terms*—AND-OR graph search, AO^* algorithm, heuristic search, nonslicible, VLSI floorplanning.**

I. Introduction

FLOORPLANNING is important not only for layout synthesis but even more for deciding on design alternatives that are likely to yield optimal quality during the early design stages. *Floorplan optimization problem* comprises two subtasks, namely, determining a suitable topology, and sizing, i.e., minimizing the area of the bounding rectangle by selecting the size (aspect ratio) of each module.

Constructive floorplanning algorithms based on graph dualization [1], [7], [21] generate *only a feasible topology* by dualization of a given adjacency graph in polynomial time; the number of topologies for a given interconnection pattern may be exponential. The sizing problem is polynomial-time solvable for slicing topologies, but NP-hard for general [16] and even hierarchical topologies of order 5 [10]. While the general case has been tackled with the branch-and-bound method [17] and simulated annealing [18], the latter may change the topology. Stockmeyer's optimal orientation algorithm [16] has been generalized for area optimization of hierarchical nonslicible topologies of order 5 [10], [19], and general topologies in [9].

Very few algorithms exist which integrate the two subtasks of floorplanning. The dynamic programming based technique in [20] provides an integrated approach for slicing floorplans. However, for inherently nonslicible [12] as well as some slicible rectangular graphs, this method introduces pseudo-modules.

The unified approach in our paper consists of two phases: 1) given any rectangular graph, a floorplan topology that is likely to have both a minimum area implementation and a minimal number of nonslice cores is generated; 2) the actual (near)-minimum area implementation for the generated topology is obtained. The solution space of possible topologies being very large, a well-known AND-OR graph search algorithm AO^* [8] is adapted with novel heuristic estimates to reduce the search effort. The decomposition is based on our key result that for any rectangularly dualizable adjacency graph, there exists a floorplan obtained by considering only slices and Z-cuts recursively; such a floorplan has a minimal number of nonslice cores yielding a channel definition with safe (free of cyclic dependence) routing ordering of the channels. All the channels are monotone [5], [13], i.e., either straight or Z-shaped and the number of Z-channels is minimal. Extensive testing on benchmarks confirm that this approach works well for any rectangular graph, producing notably small area.

The rest of this paper is organized as follows. In Section II, some terminologies related to the rectangular dualization method of topology generation are given. The notion of Z-cuts is introduced which is then used in topology generation with size parameters based on AND-OR graph search. Section III presents the bottom-up sizing method. Experimental results are presented in Section IV with concluding remarks in Section V.

II. Topology Generation Using Slices and Z-Cuts

A. Rectangular Duals

A *floorplan* is a rectangular dissection of a given rectangular section by isothetic line segments called cuts. The indivisible nonoverlapping rectangles correspond to the functional modules. A floorplan is said to be *slicible* if it is obtained recursively by using through cuts or *slices* only, otherwise it is *nonslicible*. It is assumed that the modules are rectangular, and the junctions among them are T-junctions only. A *wheel* (5-wheel) is a nonslicible floorplan with five modules [19]. A floorplan is said to be *hierarchical of order 5* if it can be obtained by recursively partitioning a rectangle into two parts by either a vertical or a horizontal line or five parts by a wheel [19].

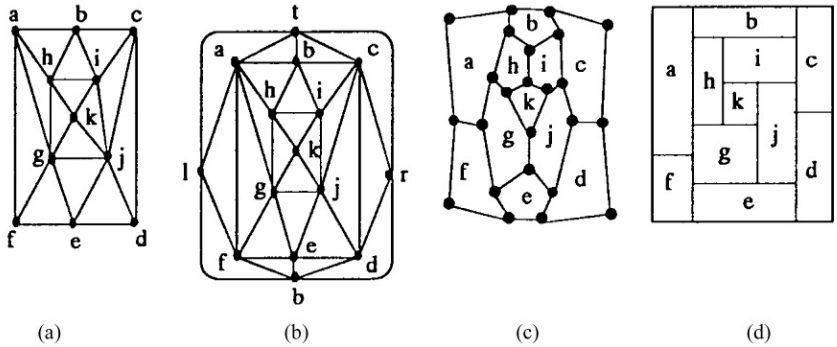

Fig. 1. Rectangular dualization: (a) rectangular graph, (b) extended rectangular graph, (c) geometric dual, and (d) a floorplan.

In the graph-theoretic *rectangular dualization* method for obtaining a floorplan topology, the interconnection requirements among modules are represented by a plane graph called the *adjacency graph* $G = (V, E)$ where for each module there is a vertex $v \in V$ and an edge $(u, v) \in E$ implies overlap in the boundaries of the modules u and v. The plane adjacency graphs which admit a rectangular floorplan are called *rectangularly dualizable* or rectangular graphs and were characterized in [1], [7]; these graphs have unique plane embeddings. One of the necessary conditions for rectangular dualizability is that G must not contain any complex three-cycle, where a *complex p-cycle* of a plane graph is a cycle of p edges having at least one vertex in the finite region bounded by it. A complex cycle will be denoted by $C_{p,m}$ where p is the length of the cycle and m is the number of vertices interior to the cycle. A rectangular graph may have more than one floorplan realization. The class of rectangular graphs which do not have any slicible realization is known as *inherently nonslicible* (*INS*) [12], [14].

A floorplan topology for a given rectangular graph G is obtained as follows. First at most, four *corner vertices* on the outermost cycle of the unique embedding of G [7] are chosen thus partitioning the cycle into at most four segments forming the sides top, right, left, and bottom. Then four external vertices t, r, l, and b, one for each side, are added. All vertices on a side are connected to the corresponding external vertex, and the edges $(t, r), (r, b), (b, l)$, and (l, t) are added to obtain an extended graph $E(G)$. Finally, G_d, the geometric dual of $E(G)$ is embedded rectilinearly to obtain a valid rectangular dual (floorplan) F; this embedding may not be unique. The outermost cycle of G_d is embedded as the bounding rectangle of F, the four sides of which are denoted by north, east, south, and west. Fig. 1(b)–(d), respectively, show an $E(G)$, its G_d, and a floorplan for the graph G of Fig. 1(a).

A *cut* in a floorplan is a simple path in G_d between two exterior vertices lying on opposite sides of the embedding of G_d. In the dual space of $E(G)$, the corresponding cut-set of edges decomposes $E(G)$ into exactly two nonempty subgraphs G_l and G_r. This cut-set is often referred to as *a cut in G*. The sequence $P_l(P_r)$ of vertices in $G_l(G_r)$ which are adjacent to these cut-edges forms the new boundary of any subfloorplan for $G_l(G_r)$. Any path in G_d is however not necessarily embeddable as a straight slice in F. A cut in F is *admissible* if it excludes at least two T-junctions on each module [12].

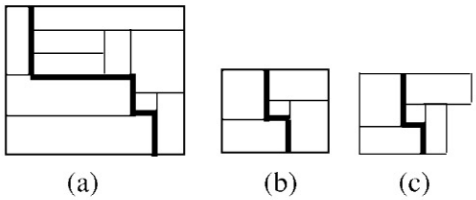

Fig. 2. (a) A generalized k-cut, (b) Z-cut in a rectangle, and (c) Z-cut in a rectilinear floorplan.

A *chord free path* (CFP) [21] in G is a path $P = (v_1, v_2, \ldots, v_n)$ where for all $i \neq j$: (a) $v_i \neq v_j$, (b) if $(v_i, v_j) \in G$, then $|i - j| = 1$. If P is not a *CFP*, then it has two vertices v_i and v_j such that $|i-j| > 1$ and the edge (v_i, v_j) is in G; this edge is called a *chord* of P. A chord $(v_i, v_j), i < j$ of P is said to be *maximal* if there is no other chord (v_r, v_s) where $r \leq i < j \leq s$. A cut is *chord-free* if both of its boundary paths are chord free. Hence, a *slice* in a floorplan corresponds to a cut in G which is chord free (embeddable as a straight line), and both G_l and G_r are rectangular. A slice is *feasible* if it is admissible, chord free, and none of the extended subgraphs on either side of it has a complex 3-cycle.

Two floorplans corresponding to the same rectangular graph are adjacency *equivalent*. A *canonical embedding* [14] of a floorplan is a adjacency equivalent floorplan with the minimum number of nonslice cores (5-wheels). Such an embedding may not be unique.

B. Z Cuts and Their Properties

Although complete characterization of slicibility of a rectangular graph is unknown, many sufficient conditions for slicibility and properties of slicible floorplans appear in [12], [21]. Some new properties related to nonslicible floorplans are presented next. Let F be a floorplan and G the corresponding rectangular graph.

Definition 1: A generalized k-cut in F (equivalently, a k-cut in G) corresponds to a cut in G for which the number of maximal chords on either of its boundary paths is at most k [Fig. 2(a)].

Definition 2: A Z-cut in F is a generalized k-cut with $k = 1$ [Fig. 2(b)].

The converse that any cut in F with two bends corresponds to a maximal chord in its G may not be true. A slice is a generalized k-cut with $k = 0$. Any generalized k-cut in F has

a rectilinear embedding with $2k$ bends or intermediate corners, thereby decomposing F into two subfloorplans each of which has a rectilinear polygonal boundary with $2k+4$ corners. Any generalized k-cut can always be rectilinearly embedded with $2k'$ bends for any $k' > k$. So the subfloorplans for a slice and a Z-cut have, respectively, rectangular and L-shaped boundaries. Floorplans with more than four corners will be referred to as *rectilinear floorplans* henceforth [Fig. 2(c)].

1) Sufficiency of Slices and Z-Cuts: For the sake of efficiency in optimal sizing, it is desirable to have a minimum number of nonslice cores in nonslicible floorplans, namely a canonical embedding. A canonical embedding of a hierarchical floorplan has either 0 or 1 nonslice core at each level of the hierarchy [14]. A single nonslice core at any level implies the existence of a Z-cut. However, the nonuniqueness of canonical embeddings leads to the observation that for a Z-cut in F, the cut-set in G may be a CFP. Determination of a nonslicible canonical embedding from a given G in a top-down fashion seems nontrivial. There exist floorplans which do not have a recursive bipartitioning using slices and Z-cuts only. Therefore, the following pertinent question is raised: *for any G, does there exist a rectangular dual which can be obtained top-down by using slices and Z-cuts only?* The affirmative answer is proven below.

Lemma 1: For any rectilinear floorplan with at least five exterior modules, there exists an equivalent rectilinear floorplan F_{rect} having a slice.

Proof: It was established in [21] that for any plane triangulated graph G, if G does not have any complex four-cycle $(C_{4,m}, m \geq 1)$ as a subgraph, then G has a slice. This implies that if the outermost cycle of G has a length greater than four, then it has a slice and hence the lemma follows. □

Observation 1: A rectilinear floorplan with four exterior modules has a Z-cut through its concave corner which splits it into two smaller slicible rectilinear floorplans [Fig. 2(c)]. For all other cases with two, three, and by Lemma 1 five or more exterior modules, a slice always exists, thereby rendering the search for cuts in top-down topology generation more efficient.

Lemma 2: Let G be a rectangular graph, C_v a k-cut in G which decomposes it into G_l and G_r, and $P_l(C_v)$ and $P_r(C_v)$ the corresponding boundary paths in G on left and right of C_v. If exactly one of $P_l(C_v), P_r(C_v)$ has a chord, then both G_l and G_r admit rectilinear floorplans.

Proof: Without loss of generality, let $P_l(C_v)$ have a chord. So the cut C_v is not a feasible slice and the subfloorplan for G_l must be rectilinear, namely L-shaped. The subfloorplan for G_r is rectangular since $P_r(C_v)$ has no chord; nevertheless, this can always be extended to make it L-shaped maintaining the adjacencies across the cut. □

Lemma 3: In a rectangular graph, the rectangular dualization problem for the subgraph interior to a $C_{4,m}$ in G is independent of that of the rest of G.

Proof: For the four vertices on the $C_{4,m}$, there are four rectangles in the floorplan enclosing a rectangular section. Thus, the subfloorplan corresponding to the interior of the $C_{4,m}$ must have a rectangular boundary, and the subgraph of G interior to the $C_{4,m}$ is also rectangular which may be treated as an independent problem. □

Lemma 4: In a complex four-cycle $C_{4,m}$ with $m \geq 1$ interior vertices, if all four of its exterior vertices are chosen as corners, a Z-cut can always be obtained.

Proof: Consider the following two cases.

Case 1) $m = 1$. The proof follows from Observation 1.

Case 2) $m > 1$. By Lemma 3 above, since the interior subgraph is a separate problem, a super-vertex corresponding to the composite rectangular subfloorplan for it is considered. Thus $C_{4,m}$ reduces to a $C_{4,1}$ and by case 1, a Z-cut between opposite sides can always be obtained in it. □

Thus, Lemmas 1–4 lead to the following important theorem:

Theorem 1: A rectangular floorplan can be obtained from a rectangular graph using recursive bipartitioning with generalized k-cuts, where $k \leq 1$, (i.e., slices and Z-cuts only).

The Z-cuts, as evident from Lemma 4, are used only for complex four-cycles with four distinct corners, the entire subgraph interior to the complex four-cycle being an independent subproblem. Using such Z-cuts has the advantage of restricting the search space to a moderate size even for the general floorplans. For rectangular graphs without complex four-cycles, our method always produces slicible floorplans. In other cases, need for Z-cuts depends on the specified sizing constraints.

Observation 2: The number of nonslice cores, i.e., the number of Z-cuts in the topology generated is minimal.

The rectangular dual produced by this scheme can be represented by a *binary floorplan tree* T_F where the root represents the bounding rectangle, an interior node represents either a slice or a Z-cut, and the leaves are the indivisible modules.

2) Safe Routing Order of Slices and Z-Cuts: In nonslicible topologies with straight channels, the channel dependency digraph CG_F has directed cycles which call for iterations in detailed channel routing [5], [13]. CG_F is a directed graph where for each channel there is a node, and an arc from i to j exists if one end of channel i terminates on channel j.

Rectilinear monotone channels in any floorplan yield acyclic channel dependency and thus a safe routing order (*SRO*), but routing of monotone channels with many bends is quite complex. Z-channels are relatively simpler. In the proposed method of topology generation a minimal number of Z-cuts are used.

Theorem 2: For a floorplan topology F obtained by top-down recursive bipartitioning using slices and Z-cuts only, its channel digraph CG_F has a safe routing order.

Proof: Clear. See [3] for details. □

C. Formulation as AND-OR Graph Search

1) AND-OR Graph Search: An optimization problem may be solved by recursively decomposing it into several subproblems until no further division is possible. At any stage there can be several possible decompositions. The root problem can be reconstructed in several ways from the possible combinations of the subproblems. The *search graph* is generally represented as an AND-OR graph, where each node is either an AND node representing actual decomposition, or an OR node denoting a

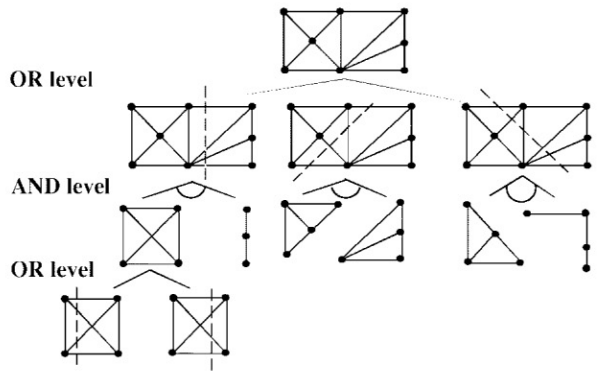

Fig. 3. AND-OR splitting of a rectangular graph.

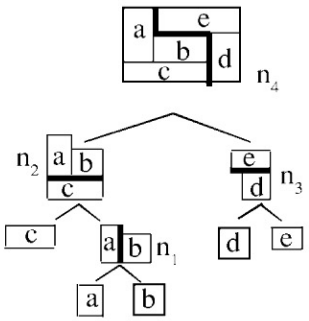

Fig. 4. A floorplan tree with slices and Z-cuts.

set of possible decomposition strategies. A solution, called the *solution graph* is a subgraph of the AND-OR graph. A cost function associated with each arc (node) of the search graph contributes to the cost of the solution(s). The objective is to find the solution graph with optimum cost for the root node s. A heuristic cost estimate $\hat{f}$ on the nodes is said to be *admissible* [8] if for each node n in the search graph $\hat{f}(n) \leq f_{\text{opt}}(n)$ where $f_{\text{opt}}(n)$ is the optimum cost of the solution graph rooted at n. To tackle an NP-complete problem in this framework, heuristic estimates are employed to direct the search, thus reducing the execution time.

A floorplan topology is determined by recursively bipartitioning the adjacency graph until all the subgraphs are single vertices. Each subgraph is realized as either a rectangle or an L-shaped composite module (L-module). An AO^*-based top-down search algorithm is used here for determining a potential topology. In the search graph, the root node s represents the given adjacency graph with a chosen set of corners, each AND node a bipartition (cut) of a composite module, and each OR node a set of possible cuts. The AND and OR nodes occur at alternate levels and have costs based on sizing constraints. Fig. 3 shows a portion of the AND-OR partitioning of an adjacency graph.

A solution graph is a binary tree with each AND (OR) node having two (one) successors. The output T_F shown in Fig. 4 is obtained from the solution tree by coalescing OR nodes with their (single) successors.

2) Determination of Cuts: Based on the results in Section II-B, the following strategy is used to find a cut in G_d or its subgraph: 1) for a rectangular boundary both slices and Z-cuts are to be found; 2) for an L-shaped boundary only slices need to be searched. The cut-finding procedure tests incrementally whether there is at most one (zero) maximal chord in case 1) [case 2)]; otherwise it backtracks.

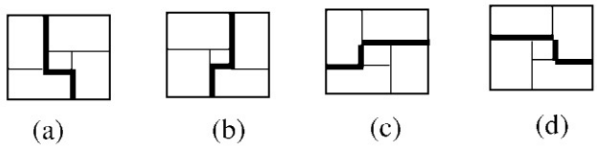

Fig. 5. Four types of Z-cuts (a) right-turning *rt*, (b) left-turning *lt*, (c) top-turning *tt*, and (d) bottom-turning *bt*.

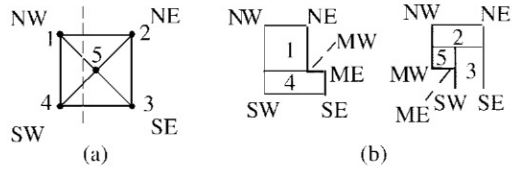

Fig. 6. (a) A complex four-cycle is split with a Z-cut with edge (2, 3) as chord and (b) corner assignment in the two resulting subfloorplans.

3) Corner Assignment: A preprocessing phase finds a set of corner assignments of the floorplan, of which one is chosen. The four convex corners of a rectangle are designated as NW, NE, SE, and SW in a clockwise manner starting from the top-left corner. For an L-module, the additional two corners are labeled as MW and ME (Fig. 6).

Bisection of a rectangle may yield either two smaller rectangles or two smaller L-modules, whereas bisection of a L-module may yield two smaller rectangles (slice through a concave corner) or a rectangle and an L-module.

Let $C = \langle v_1^d, v_2^d, \ldots, v_n^d \rangle$ denote a cut (path) in $G_d, P_l(C) = \langle v_1^1, v_1^2, \ldots, v_1^{n-1} \rangle$ and $P_r(C) = \langle v_2^1, v_2^2, \ldots, v_2^{n-1} \rangle$ be the two boundary paths.

The salient cases of corner assignment are now discussed.

Case a) Slice in a rectangle: For a vertical slice, let $G_l(G_r)$ be the left (right) subgraphs. Then, $\text{NW}(G_l) = \text{NW}(G)$, $\text{NE}(G_l) = v_1^1$, $\text{SE}(G_l) = v_1^{n-1}$, $\text{SW}(G_l) = \text{SW}(G)$; $\text{NW}(G_r) = v_2^1$, $\text{NE}(G_r) = \text{NE}(G)$, $\text{SE}(G_r) = \text{SE}(G)$, $\text{SW}(G_r) = v_2^{n-1}$.

Case b) Z-cut in a rectangle: There are four types of Z-cuts, namely, right-turning *(rt)*, left-turning *(lt)*, rising *(tt)*, and falling *(bt)* as shown in Fig. 5.

Fig. 6 shows a vertical Z-cut through a complex four-cycle (1, 2, 3, 4) The chord (2, 3) for this Z-cut is on the right [Fig. 6(a)]. The NW, NE, SE and SW corners of the new subfloorplans are obtained as in the previous case. In addition, there is a convex and a concave corner for both G_l and G_r [Fig. 6(b)]. A concave corner is represented by the two modules forming it. If the cut is of type *rt*, then $\text{MW}(G_l) = (1, 4)$, $\text{ME}(G_l) = 4$, $\text{MW}(G_r) = 5$, and $\text{ME}(G_r) = (5, 3)$.

Case c) Slice through a concave corner in an L-module: As in Fig. 7(b), $\text{NW}(G_l) = \text{MW}(G)$, $\text{NE}(G_l) = v_l^1$,

$\mathrm{SE}(G_l) = v_1^{n-1}$, $\mathrm{SW}(G_l) = \mathrm{SW}(G)$; $\mathrm{NW}(G_r) = \mathrm{NW}(G)$, $\mathrm{NE}(G_r) = \mathrm{NE}(G)$, $\mathrm{SE}(G_r) = \mathrm{SE}(G)$, $\mathrm{SW}(G_r) = v_2^{n-1}$.

Case d) Slice not through a concave corner in an L-module: As in Fig. 7(c), $\mathrm{MW}(G_l) = \mathrm{MW}(G)$, $\mathrm{ME}(G_l) = \mathrm{ME}(G)$, $\mathrm{NW}(G_l) = \mathrm{NW}(G)$, $\mathrm{NE}(G_l) = v_1^1$, $\mathrm{SE}(G_l) = v_1^{n-1}$ and $\mathrm{SW}(G_l) = \mathrm{SW}(G)$; $\mathrm{NW}(G_r) = v_2^1$, $\mathrm{NE}(G_l) = \mathrm{NE}(G)$, $\mathrm{SE}(G_l) = \mathrm{SE}(G)$, and $\mathrm{SW}(G_l) = v_2^{n-1}$.

For the slice in Fig. 7(d), new corners are assigned similarly.

4) Cost Computation: Each cut has a cost which is an ordered list of the following.

a) Estimate for the minimum wasted area of the floorplan corresponding to a node, where the *wasted area* is the area of the floorplan minus the sum of the area of its modules. This is computed from the estimated minimum dimensions of the corresponding floorplan.
b) Estimated ranges of the dimensions along the cut for the two parts on two sides of the cut. Greater is the overlap, better is the fitting of the two halves of the floorplan on two sides of the cut and hence the quality of the final floorplan. For a slice, this measure is computed for the entire range of the cut. A Z-cut is considered as a collection of three segments, and a composite measure is obtained from the overlap measures for each segment.
c) The ratio of the areas on the left (bottom) and on the right (top) of the cut. This is for a balanced cut.
d) The ratio of the proportional areas on the left (bottom) and on the right (top) of the cut, where *proportional area* is defined as the actual area divided by the number of modules in the corresponding part. This helps in achieving a uniform distribution of the modules over the floorplan and in reducing the height of the search graph.

For slices through the concave corner of an L-module, parameters (b) and (c) are ignored. Each node n in the search graph has a nonnegative cost estimate $\hat{f}(n)$ and an actual cost $f(n)$. The cost estimate $\hat{f}$ in the explored search graph $G_{\exp}$ is derived from the costs (a)–(d) defined above. Thus, $\hat{f}(n)$ is a 4-tuple $[\hat{f}^1(n), \hat{f}^2(n), \hat{f}^3(n), \hat{f}^4(n)]$.

The actual cost $f(n)$ of a node n in the underlying search graph G_{und} is a similar ordered list. Let $f_1^1(n), f_2^1(n), f_3^1(n)$, and $f_4^1(n)$ be the lengths of the longer and shorter horizontal edges, and the longer and shorter vertical edges respectively, for a module corresponding to node n. Then $f^1(n) = f_1^1(n) * f_3^1(n) - [f_1^1(n) - f_2^1(n)] * [f_3^1(n) - f_4^1(n)]$. For rectangular module, $f_1^1(n) = f_2^1(n)$ and $f_3^1(n) = f_4^1(n)$.

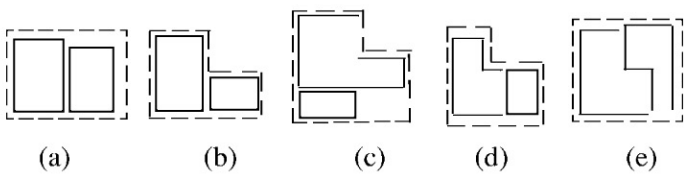

Fig. 7. Five types of nodes in bottom-upsizing algorithm.

$f(n)$ is defined as follows.

a) n is a terminal leaf. $f(n) = \hat{f}(n)$ where the values of $\hat{f}^m(n)$ for $m > 2$ are don't cares.
b) n is an OR node with immediate successors $n_1, n_2, \cdots, n_k$ in G_{und}. Here, $f(n)$ is given by the 4-tuple corresponding to a *nondominated* successor node n_i, where a node n_i is said to be nondominated among a set of nodes if there is no other node in the set that is better than n_i in the given order of the elements of f.
c) n is an AND node with immediate successors n_1 and n_2. $f(n)$ is a function of $f(n_1)$ and $f(n_2)$ defined in the manner shown at the bottom of the page where $v(n_i)$ is the number of vertices in the subgraph for node n_i. The cost estimate $\hat{f}(n)$ is defined similarly.

At an unexplored OR node n, $\hat{f}(n)$ is estimated by considering the given implementations of the exterior vertices for node n. For the module n, $\hat{f}^1(n)$ is computed by considering the minimum and maximum dimensions of the noncorner exterior modules and all possible dimensions of the corner modules.

An unexplored AND node directly inherits the cost estimate from its predecessor. For all other nodes, the costs are estimated from the aspect ratios and adjacency subgraphs in a way similar to that for $f(n)$. As the explicit graph grows, $\hat{f}(n)$ becomes an increasingly finer estimate of $f(n)$. For the root node $\hat{f}(s)$ is ideally equal to $f(s)$. Since $\hat{f}$ involves only the minimum estimates, it is admissible.

For a SOLVED node n [8], $\hat{f}(n)$ may not be the actual minimum cost of the corresponding partial floorplan. To improve the solution quality, whenever a node is SOLVED, the best implementation for the corresponding topology is computed in a bottom-up manner, and the estimated dimensions are set accordingly. We will refer to this as *partial cost computation* (*PCC*). PCC can be done selectively at different levels of the search graph to have a tradeoff between the processing time and the solution quality. The proposed method fails to consider any interaction between the subgoals (AND nodes). Thus, the best solution of a subgoal (obtained by PCC) may

---

$f^1(n)$ is computed from $f^1(n_1)$ and $f^1(n_2)$ discussedas in [3].

$f^2(n) =$ overlap along common boundary of n_1 and n_2.

$$f^3(n) = \frac{f_1^1(n_1) * f_3^1(n_1) - \left[f_1^1(n_1) - f_2^1(n_1)\right] * \left[f_3^1(n_1) - f_4^1(n_1)\right]}{f_1^1(n_2) * f_3^1(n_2) - \left[f_1^1(n_2) - f_2^1(n_2)\right] * \left[f_3^1(n_2) - f_4^1(n_2)\right]}.$$

$$f^4(n) = \frac{\left[f_1^1(n_1) * f_3^1(n_1) - \left[f_1^1(n_1) - f_2^1(n_1)\right] * \left[f_3^1(n_1) - f_4^1(n_1)\right]\right]/v(n_1)}{\left[f_1^1(n_2) * f_3^1(n_2) - \left[f_1^1(n_2) - f_2^1(n_2)\right] * \left[f_3^1(n_2) - f_4^1(n_2)\right]\right]/v(n_2)}.$$

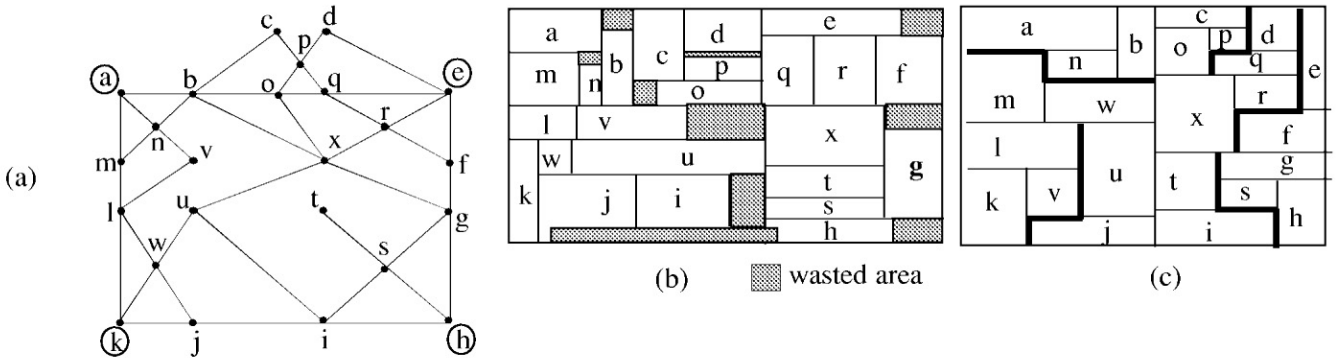

Fig. 8. Example 2: (a) input graph, (b) solution with slices only, and (c) solution with slices and Z-cuts.

not correspond to the best solution of its predecessor subgoal, and hence an optimum solution is not provably guaranteed [6].

5) Search Algorithm for Top-Down Topology Generation: Formal description of the proposed algorithm AO_FP^* for Phase I, namely the generation of a potential topology is presented below. Let G' be the actual search graph generated by the algorithm located at the bottom of the page. Restricting the cuts to slices only naturally reduces the space and time complexities of AO_FP^* considerably.

III. Optimal Sizing

The second phase of our method achieves optimal sizing in a bottom-up fashion described in this section. The output of Phase I is a binary floorplan tree T_F. Using the given set of possible implementations of the leaf modules, the implementations of the root node of T_F can be obtained by a post-order traversal of T_F and combining the implementation lists of children of an internal node to form that of their parent. An optimal implementation of the floorplan can then be obtained by selecting one with minimum area from the irredundant set of implementations for the root. The minimum area of a floorplan clearly implies its minimum wasted area.

Implementation of a rectangular module is given by a 2-tuple (w, h) where w is its width and h is its height. For an L-module, it is a 4-tuple (w^1, w^2, h^1, h^2) where $w^1(w^2)$ represent the lengths of the longer (shorter) horizontal edges

---

Algorithm AO_FP^* (**Phase I**)
Input: **A** rectangular graph G.
Output: **A** floorplan topology in the form of a binary slicing tree, T_F.
begin
input G which is the start node s of G' and the initial
nonleaf unexplored node of the potential solution graph (*psg*);
repeat (* until s is labeled SOLVED *)
choose a nonleaf unexplored node n of the marked *psg*; expand n;
for each successor n_i of n not already in G' **do begin**
compute $\hat{f}(n_i)$; **if** n_i is a terminal leaf **then** label n_i SOLVED;
create S := $\{n\}$;
end;
repeat (* until S is empty *)
if (m is a node in S) **and** (no descendant of m in G' occurs in S)
then remove m from S;
if (m has no successor) **and** (m is not a leaf) **then** $e := \infty$;
($^*e$: a temporary cost variable *)
else if m is an OR node **then begin**
choose the successor m_i of m with minimum $\hat{f}(m_i)$;
let this node be m_{min}; $e := \hat{f}(m_{min})$; mark the arc (m, m_{min});
if m_{min} is SOLVED **then** label m SOLVED;
end;
else if m is an AND node **then**
if both of its successors m_1 **and** m_2 are SOLVED **then**
compute the *partial cost* e_p of m; $e := e_p$; (* PCC *);
else compute e using $\hat{f}(m_1)$ and $\hat{f}(m_2)$;
if $e <> \hat{f}(m)$ **then** $\hat{f}$(m) := e;
if (value of $f(m)$ changes in the previous step)
or (m is labeled SOLVED) **then**
add to S all immediate predecessors of m along marked arcs;
until S is empty;
until s is labeled SOLVED;
output marked subgraph of the AND-OR search graph as T_F;
end.

and $h^1(h^2)$ represent those of the longer (shorter) vertical edges. For an L-module, an implementation (w^1, w^2, h^1, h^2) dominates another $(w^{1'}, w^{2'}, h^{1'}, h^{2'})$ if $w^1 \leq w^{1'}, w^2 \leq w^{2'}, h^1 \leq h^{1'}$ and $h^2 \leq h^{2'}$. A set of implementations Σ is redundant if there exist distinct implementations α and β in it where α dominates β. Thus, for area minimization, β can be discarded, and only an irredundant set of implementations need to be maintained.

For the convenience of processing and to have a higher degree of pruning, irreducible *R-list*'s and *L-list*'s are used as in [19].

The nonslicing structures generated by the proposed method are hierarchical in nature [19]. Five major categories of nodes that may occur in the floorplan tree are as follows.

- **Type A)** A rectangle formed by combining two smaller rectangles as in Stockmeyer's algorithm [Fig. 7(a)] [16].
- **Type B)** A L-module formed by combining two rectangular modules: generalization of an α-node [19] [Fig. 7(b)].
- **Type C)** A composite L-module obtained by joining a rectangular module along either of the longer edges of an L-module: generalization of the β-node [19] [Fig. 7(c)].
- **Type D)** A composite L-module obtained by joining a rectangular module along either of the shorter edges of an L-module: generalization of the γ-node [19] [Fig. 7(d)].
- **Type E)** A rectangle obtained by attaching two L-modules with facing concave edges [Fig. 7(e)].

An irreducible R-list or L-list at a node of T_F is obtained by appropriately combining the irreducible R-lists and/or L-lists of the successors. There are five basic procedures, called *type i* procedures, $i \in \{A, B, C, D, E\}$ to construct a set of irreducible R-lists or L-lists for a type i node. In all the cases, m is a node of type $i, i \in \{A, \ldots, E\}$ with m_1 and m_2 as its successors.

Type A Procedure (proc_A): Nodes m, m_1, and m_2 correspond to rectangles. Let R_{m_1} and R_{m_2} be the R-lists for nodes m_1 and m_2, respectively. This procedure constructs at most $|R_{m_1}|+|R_{m_2}|-1$ implementations in an R-list R_m. The construction of an element of R_m is as in [16].

Type B Procedure (proc_B): Nodes m_1 and m_2 represent rectangles. We assume that m_1 is the rectangle on the left (vertical cut) or at the bottom (horizontal cut) and m_2 is the rectangle on the right (vertical cut) or on the top (horizontal cut).

Let R_{m_1} and R_{m_2} be the R-lists for nodes m_1 and m_2, respectively. The procedure constructs a set $\mathcal{L}_m$ of $|R_{m_1}|$ irreducible L-lists $\{L_m^1, L_m^2, \cdots, L_m^{|R_{m_1}|}\}$ with each list containing $|R_{m_2}|$ elements. Construction of each element of $\mathcal{L}_m$ is a generalization of α-procedure [19].

Type C Procedure (proc_C): Nodes m_1 and m_2 are, respectively, a rectangle and an L-module and are combined along a longer edge of m_2. The dimensions of m_1 are in an R-list R_{m_1} containing M elements, and those of m_2 are in a set of L-lists $\mathcal{L}_{m_2} = \{L_{m_2}^1, L_{m_2}^2, \cdots, L_{m_2}^N\}$. For a vertical cut, the procedure is preceded by a processing as described below. The widths and heights of the modules m_1 and m_2 are interchanged; and each of the L-lists and the R-list is rearranged to obtain new L-lists and R-list. The total number of elements in the L-lists is $T = \sum_{i=1}^{N} |L_{m_2}^i|$ and hence the average time complexity of the preprocessing will be $O(T \log T)$ which will not affect the overall time complexity of the sizing. For a horizontal cut, this preprocessing is not required. The procedure is identical to procedure-β [19]. For a vertical cut, the dimensions of the composite modules are rearranged to the original ordering in time $O(T \log T)$.

Type D Procedure (proc_D): Nodes m_1 and m_2 are a rectangle and an L-module, respectively, and are combined along a shorter edge of m_2. The dimensions of m_1 are in an R-list R_{m_1} containing M elements, and those of m_2 are in a set of L-lists $\mathcal{L}_{m_2} = \{L_{m_2}^1, L_{m_2}^2, \cdots, L_{m_2}^N\}$. For a horizontal cut, along the shorter horizontal edge of m_2, a preprocessing is done as in the previous case. The procedure is identical to procedure-γ [19]. For a horizontal cut, the dimensions of the composite modules are rearranged to the original ordering.

Type E Procedure (proc_E): Nodes m_1 and m_2 represent two L-modules. The dimensions of m_1 and m_2 are given in two sets of L-lists $\mathcal{L}_{m_1} = \{L_{m_1}^1, L_{m_1}^2, \cdots, L_{m_1}^p\}$ and $\mathcal{L}_{m_2} = \{L_{m_2}^1, L_{m_2}^2, \cdots, L_{m_2}^q\}$. The dimensions of m are formed by combining these L-lists sequentially from the beginning to the end and the new dimensions are stored in a single R-list, which is later pruned to remove redundant elements. If the xth element of $L_{m_1}^i$ and the yth element of $L_{m_2}^j$ are $(w_1^1, w_1^2, h_1^1, h_1^2)$ and $(w_2^1, w_2^2, h_2^1, h_2^2)$, respectively, the composite rectangle will have dimensions given by $[\max(w_1^1+w_2^2, w_1^2+w_2^1), \max(w_1^1+w_2^2, w_1^2+w_2^1), \max(h_1^1, h_2^1, h_1^2+h_2^2), \max(h_1^1, h_2^1, h_1^2+h_2^2)]$ for vertical cut and $[\max(w_1^1, w_2^1, w_1^2+w_2^2), \max(w_1^1, w_2^1, w_1^2+w_2^2), \max(h_1^1+h_2^2, h_2^1+h_1^2), \max(h_1^1+h_2^2, h_2^1+h_1^2)]$ for horizontal cut. Certain key observations which enable us to reduce the size of the R-list follow. Let $p \in L_{m_1}^i$ and $q \in L_{m_2}^j$ be implementations of m_1 and m_2, respectively, with $h_p^1 > h_q^1$. For any r which appears before q in the L-list $L_{m_2}^j, h_r^1 < h_q^1 < h_p^1$, so r need not be considered for a vertical cut. Similar reductions are applied to a horizontal cut.

A pruning procedure as in [19] is used on the L-lists and R-lists formed. The proposed sizing algorithm is described at the bottom of the next page. This phase becomes polynomial-time solvable if only slices are used. Only type A-procedure has to be used with R-lists.

1) Complexity of the Algorithm: Since the first phase of our method uses a best-first AND-OR graph search, it is difficult to provide an exact theoretical complexity. This can only be judged from the empirical observations.

The bottom-up sizing algorithm is complicated for floorplan topologies having wheels. Thus, in discussing the complexity of sizing, the sizing for a wheel is primarily dealt with. For slices only case, the complexity of this phase becomes $O(b \log b)$ where there are b modules with two possible aspect ratios each [16].

Consider the wheel in Fig. 4 along with the cuts through it. Let q_i be the number of implementations for module b_i. Then the total number of implementations of the L-module at n_1

is q_1q_2 (q_1 L-lists, each having q_2 implementations). Each of these q_1 L-lists is now combined with the q_3 implementations of module b_3 to form q_1 L-lists each having $(q_2 + q_3 - 1)$ implementations. At n_3, there are q_5 L-lists, each having q_4 implementations. Thus at n_4, q_1 L-lists, each having $(q_2 + q_3 - 1)$ are to be combined with q_5 L-lists, each having q_4 elements. These form several R-lists. Total number of R-lists is q_1q_5, each of $(q_2+q_3+q_4-2)$ elements. The time required in type-E procedure is dominated by merging these R-lists and the time required for it is $O(q_1q_5(q_2+q_3+q_4-2)\log(q_1q_5))$. The total time required is the sum of the times in the individual nodes and is clearly dominated by the time required in the type-E procedure at node n_4. Assuming $q_1 = q_2 = q_3 = q_4 = q_5 = q$, the time complexity of the sizing for the wheel is $O(q^3 \log q)$. Although the time complexity of the sizing method used by us is greater than that in [10], our method is simpler to implement.

IV. Experimental Results

The proposed method has been implemented in C on DEC Alphastation 250 running at 266 MHz clock speed. Experiments have been performed on some benchmark problems available from the literature and on some randomly generated rectangular graphs. In Tables I and II, these benchmarks are represented as examples 1–17; of these, examples 11–17 are INS. Examples 1–6 are taken from [20], [17], [21], [19], [15], and [2], respectively. Examples 7, 8, and 10 are randomly generated, example 9 is from [11], examples 11–15 are taken from [19], and examples 16 and 17 are from [15]. Details of aspect ratios appear in [3]. Some of the solutions obtained by us are shown in Figs. 8–10.

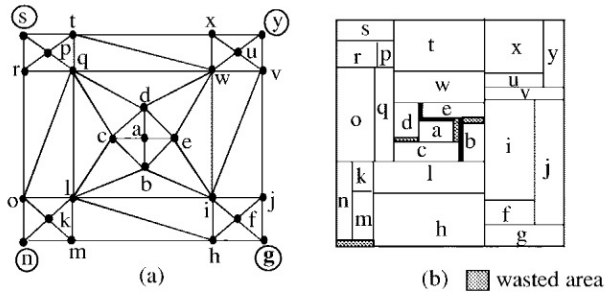

Fig. 9. Example 15: (a) an INS input graph and (b) solution obtained.

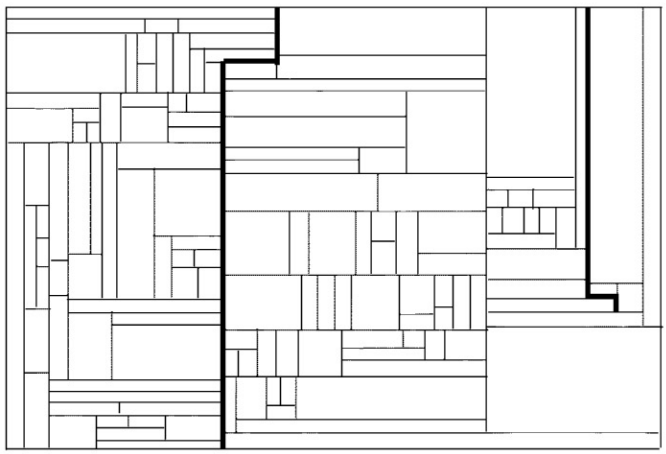

Fig. 10. Example 6: Topology generated with slices and Z-cuts (sizing not shown).

Two versions of the proposed method have been tested: AO_FP^* allowing slices and Z-cuts and $AO\_FP^*_{\text{slice}}$ allowing

---

Sizing Algorithm (Phase II)
Input: T_F, a potential topology from AO_FP^*.
Output: A minimum-area floorplan for T_F.
proc_botup()
input: sets of irreducible R-lists associated with the modules (from given sets of aspect ratios), and T_F
output: an irreducible R-list R_s for s, the rootnode of T_F.
begin for all nodes i in T_F, computed[i] := 0; search(s); **end**.
search(n) (* recursively scan the subtree below node n *)
begin
if n is a leaf node **then** computed[n] := 1;
else begin
left (n) := left successor of n; right(n) := right successor of n;
if (computed[*left*(n)] = 1) **and** (computed[*right*(n)] = 1) **then begin**
computed[n] := 1;
(* compute dimensions of n from those of its successors *)
case of type(n)
A : proc_A(n);
B : proc_B(n);
C : proc_C(n);
D : proc_D(n);
E : proc_E(n);
end;
end;
else begin search(*left*(n)); search(*right*(n)); **end**;
end;
end.

TABLE I
EXPERIMENTAL RESULTS WITH SLICES AND Z-CUTS

Example	# implementations	Area	Wasted area (%)	# Z-cuts	CPU time (secs.)		Distinct nodes	Total nodes
					Phase I	Phase II		
1	3^{10}	30	0.00	0	0.008	0.00001	80	155
2	2.03×10^{16}	1024	0.00	5	11.75	0.03	3601	7547
3	4^{22}	140	5.71	0	8.36	0.0001	6396	13268
4	4^{40}	242	0.83	0	1121.34	0.0009	88962	182513
5	4^{16}	72	0.00	2	0.39	0.0003	406	725
6	2^{137}	196.42	3.73	2	9018.7	1.98	273876	512204
7	2^{23}	88	0.00	0	0.42	0.0009	729	1340
8	2^{41}	247	0.00	0	1051	0.0009	70115	182923
9	4^{49}	360	18.33	0	383.03	0.001	53502	129944
10	2^{201}	310	3.9	0	3899.16	0.08	175212	321242
11*	3^{25}	117	14.53	1	7.69	0.01	5757	12289
12*	4^{25}	160	6.25	1	9.57	0.01	5727	11989
13*	5^{25}	468	14.53	1	16.72	0.01	5718	12190
14*	6^{25}	320	6.25	1	16.1	0.07	5693	11877
15*	8^{25}	638	5.96	1	39.99	0.01	3038	6136
16*	3^{18}	77	0.00	1	0.36	0.0003	369	606
17*	3^{15}	56	0.00	2	0.42	0.0001	626	1071

(* Inherently nonslicible input graphs).

TABLE II
EXPERIMENTAL RESULTS WITH SLICES ONLY

Example	Area obtained	Wasted area (%)	CPU time (secs.)		Distinct nodes	Total nodes
			Phase I	Phase II		
1.	30	0.00	0.008	0.00001	80	155
2.	1200	14.67	0.33	0.0001	418	448
3.	140	5.71	8.36	0.0001	6396	13268
4.	242	0.83	779.25	0.0009	72400	172493
5.	84	14.29	0.12	0.0012	118	178
6.	416.25	54.57	2043.04	0.01	50944	105996
7.	88	0.00	0.42	0.0009	729	1340
8.	345	0.00	751	0.0009	68201	152313
9.	360	18.33	383.03	0.001	53502	129944
10.	310	3.9	3899.16	0.08	175212	321242
11-15	INS		0.0001	N.A.	516	940
16.	INS		0.0001	N.A.	119	170
17.	INS		0.0001	N.A.	0	0

slices only. In both the cases, the CPU time required in each of the two phases, the area obtained, and the percentage of wasted area are noted. The space requirement is reduced by using a graph search instead of a tree and the reduction is noted in terms of the *total nodes generated* and the *number of distinct nodes* in the search graph as given in Tables I and II.

Version AO_FP^* produces a hierarchical floorplans of order 5. Most of the results obtained have nominal wasted area, and the time and space requirements are not too large, except for very large input graphs. Table I shows the performance of AO_FP^* for the above benchmarks; sizing complexity is demonstrated in terms of the total number of possible implementations.

The search space requirement and the execution times are observed to be quite small for $AO\_FP^*_{\text{slice}}$ compared to AO_FP^* but the wasted area is generally larger. It fails to produce any feasible implementation for input INS graphs, and reports failure within 0.0001 second. Table II summarizes the performance of the version $AO\_FP^*_{\text{slice}}$.

The complexity of the heuristic search being much larger than that of sizing, the former clearly dominates the overall complexity of our method.

In our experiments, for slicible graphs, the version AO_FP^* outputs either a slicible or a nonslicible floorplan depending on the sizing parameters, whereas for INS graphs, it definitely yields a nonslicible floorplan.

The number of implementations for examples 6 (with 137 modules) and 10 (with 201 modules) are too large, and hence the CPU time required is quite high. To restrict the search space to a reasonable limit, the maximum number of cuts in the input graph is kept small and varied with the depth of a node in the search graph as specified by the user for better solution quality.

To derive the set of aspect ratios for each module, initially a nonslicing floorplan topology is obtained without considering the aspect ratios. Next, based on this topology, a suitable pair of dimensions for each module is estimated. If (a, b) are the estimated dimensions for a module, the set of aspect ratios for the latter is chosen $\{(a, b), (b, a)\}$. For these aspect ratios, $AO\_FP^*_{\text{slice}}$ has more than 50% of wasted area, but AO_FP^* yields a nominal wasted area [3].

V. CONCLUSION

This paper demonstrates an application of AI-based graph search methods to an important problem in VLSI layout design. Apart from the novelty of the approach, the results so far in terms of area and time are very promising compared

to those of few alternative methods [20] reported earlier. Our top-down topology generation preserves all adjacencies of the input graph; however, the bottom-up sizing method may not guarantee 100% preservation of these adjacencies. The amount of violation observed for the above examples varies from 0 to 16% [3].

The utility of our approach is three-fold. First, it proves the sufficiency of using certain types of cuts for transforming a rectangular graph into a floorplan the channels in which can be safely routed later on. Second, optimal sizing for nonslicible topologies can be performed in polynomial time. Third, it illustrates the applicability of well-studied AI algorithms to VLSI design. Further improvement of the proposed algorithm has appeared in [4]. It can be easily extended to weighted adjacency graphs by modifying the path search routine.

The heuristic search method has scope of improvement. For rectangular graphs with a large number of possible slices at the root, a depth-first strategy is needed. Some dominance rules may be framed to reduce the search space. Moreover, the CPU time and node generations can be reduced by using a weighted inadmissible heuristic estimate at little cost of solution quality.

REFERENCES

[1] J. Bhasker and S. Sahni, "A linear algorithm to find a rectangular dual of a planar triangulated graph," in *Proc. 23rd ACM/IEEE Design Automation Conf.*, June 1986, pp. 108–114.

[2] Y. Cai and D. F. Wong, "A channel/switchbox definition algorithm for building-block layout," in *Proc. 27th ACM/IEEE Design Automation Conf.*, June 1990, pp. 638–641.

[3] P. S. Dasgupta, "Studies on the application of AI techniques to VLSI design," Ph.D. dissertation, Univ. Calcutta, India, Feb. 1996.

[4] P. S. Dasgupta and S. Sur-Kolay, "Slicibility of rectangular graphs and floorplan optimization," in *Proc. Int. Symp. Phys. Design*, Apr. 1997, pp. 150–155.

[5] M. Guruswamy and D. F. Wong, "Channel routing order for building-block layout with rectilinear modules," in *Proc. IEEE Int. Conf. Computer-Aided Design*, 1988, pp. 184–187.

[6] R. E. Korf, "Learning to solve problems by searching for macro-operators," *Res. Notes in Artificial Intelligence*, vol. 5, 1985.

[7] Y. T. Lai and S. M. Leinwand, "Algorithms for floorplan design via rectangular dualization," *IEEE Trans. Computer-Aided Design*, vol. 7, pp. 1278–1289, Dec. 1988.

[8] N. J. Nilsson, *Problem Solving Methods in Artificial Intelligence*. New York: McGraw-Hill, 1971.

[9] P. Pan and C. L. Liu, "Area minimization for floorplans," *IEEE Trans. Computer-Aided Design*, vol. 14, pp. 123–132, Jan. 1995.

[10] P. Pan, W. Shi, and C. L. Liu, "Area minimization for hierarchical floorplans," in *Proc. Int. Conf. Computer-Aided Design*, 1994, pp. 436–440.

[11] M. Rebaudengo and M. S. Reorda, "GALLO: A genetic algorithm for floorplan area optimization," *IEEE Trans. Computer-Aided Design*, vol. 15, pp. 943–951, Aug. 1996.

[12] S. Sur-Kolay and B. B. Bhattacharya, "On the family of inherently nonslicible floorplans in VLSI layout design," in *Proc. IEEE Int. Symp. Circuits Syst.*, Singapore, June 1991, pp. 2850–2853.

[13] S. Sur-Kolay and B. B. Bhattacharya, "The cycle structure of channel graphs in nonslicible floorplans and a unified algorithm for feasible routing order," in *Proc. IEEE Int. Conf. Computer Design*, Oct. 1991, pp. 524–527.

[14] S. Sur-Kolay and B. B. Bhattacharya, "Canonical embedding of rectangular duals with applications to VLSI floorplanning," in *Proc. 29th ACM/IEEE Design Automation Conf.*, June 1992, pp. 69–74.

[15] S. Sur-Kolay, "Studies on nonslicible floorplans in VLSI layout design," Ph.D. dissertation, Jadavpur Univ., India, Nov. 1991.

[16] L. Stockmeyer, "Optimal orientation of cells in slicing floorplan designs," *Inform. Control*, vol. 57, pp. 91–101, 1983.

[17] S. Wimer, I. Koren, and I. Cederbaum, "Optimal aspect ratios of building blocks in VLSI," *IEEE Trans. Computer-Aided Design*, vol. 8, pp. 139–145, Feb. 1989.

[18] D. F. Wong and C. L. Liu, "Floorplan design for rectangular and L-shaped modules," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, 1987, pp. 520–523.

[19] T. C. Wang and D. F. Wong, "Optimal floorplan area optimization," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 992–1002, Aug. 1992.

[20] K. H. Yeap and M. Sarrafzadeh, "A unified approach to floorplan sizing and enumeration," *IEEE Trans. Computer-Aided Design*, vol. 12, pp. 1858–1867, Dec. 1993.

[21] K. H. Yeap and M. Sarrafzadeh, "Sliceable floorplanning by graph dualization," *SIAM J. Discrete Math.*, vol. 8, pp. 258–280, May 1995.

Partha S. Dasgupta received the B.Tech. degree in radiophysics and electronics and the M.Tech. and Ph.D. degrees in computer science from the University of Calcutta, India, in 1983, 1985, and 1997, respectively.

At the Indian Institute of Management, Calcutta, he was on the Technical Staff of a UNDP sponsored project between 1987 and 1992, Systems Analyst at the Computer Center from 1992 to 1996, and has been a faculty member in the MIS group since 1996. His current research interests are in VLSI CAD, AI, and analysis of algorithms.

Susmita Sur-Kolay received the B.Tech. (Honors) degree in electronics and electrical communication engineering from Indian Institute of Technology, Kharagpur, in 1980, and the Ph.D. degree from Jadavpur University, India in 1991.

She was a Research Assistant at the Laboratory for Computer Science, Massachusetts Institute of Technology, Cambridge, from 1980 to 1985. She was a Research Fellow at the Indian Statistical Institute, Calcutta, from 1987 to 1991, and Postdoctoral Fellow at University of Nebraska, Lincoln, in 1992. Since 1993, she has been a Reader in the Department of Computer Science and Engineering, Jadavpur University. She served on the Program Committee of the 10th International Conference on VLSI Design in 1997. Her research interests include algorithms, VLSI CAD, and computational geometry.

Dr. Sur-Kolay was awarded the President of India Gold Medal in 1980.

Bhargab B. Bhattacharya received the B.Sc. (Honors) degree in physics, the B.Tech. and M.Tech. degrees in radiophysics and electronics, and the Ph.D. in computer science, all from the University of Calcutta, India.

Since 1982, he has been on the faculty of the Indian Statistical Institute, Calcutta, where currently he is a Professor of Computer Science. From 1985 to 1987 he was with the Department of Computer Science, University of Nebraska, Lincoln, as a Visiting Assistant Professor. He served in the conference committees of the International Test Conference from 1986 to 1987, the Asian Test Symposium in 1995, the International Conference on VLSI Design from 1996 to 1998, and the International Conference on High-Performance Computing in 1997. He also worked as the Tutorial Co-Chair in the 7th International Conference on VLSI Design in 1994 and as the Program Co-Chair at the 10th International Conference on VLSI Design in 1997. His research interests include fault-tolerant computing, logic synthesis and design for testability, VLSI physical design, and computational geometry.