A New Approach to the Function and Technique of Global Routing

E. Shragowitz, J. Lee, [*] *S. Sahni*

Computer Science Department

University of Minnesota

4-192 EE/CSci Building

200 Union St. S.E.

Minneapolis, MN 55455

[*] J. Lee is currently with AT&T, Murray Hill.

Abstract

In the first part of this paper we introduce a new algorithm for global routing based on minimization of maximal level of track usage. The properties of this algorithm are studied and important relations are obtained. In the second part we describe an implementation of this algorithm as an element of a combined placement-global routing procedure, where global routing is interlaced with the placement. Global routing is used for estimation of the availability of routing resources for potential placements. Experimental results using real designs are provided.

1 Introduction.

The layout problem is traditionally broken up into two successive subproblems, placement and routing, to reduce the inherent computational complexity. There are two factors to be considered for the placement problem: geometric fit and routability. To satisfy the geometric constraints, all modules must be placed within the chip without overlapping. Routability is guaranteed if the router can connect all nets for a placement which has geometric fit. The routing problem is also traditionally broken up into two subproblems: global routing and detailed routing. During the global routing step, approximate locations are assigned to nets. These locations span several routing tracks. During the detailed routing process, exact track positions are assigned to every connection path. For each chip design style: the gate array and its new modification sea-of-gates, standard cell and custom cell, there are certain differences in the formulation of placement and routing problems. Breaking up the layout problem into placement and routing, however, is applicable to all of them. In this paper we are going to describe a layout methodology where global routing is included in the placement procedure and is used to reserve area for routing during the placement process and to make placement dependent on the availability of resources for detailed routing. Global routing simultaneously performs its traditional function of assigning connection paths to routing channels. A new global router is proposed for this purpose. It can be used along with traditional design methodology as a fast global router, or it can be applied to a combined placement-routing procedure. This global router has some proven properties and was used as part of the layout system for a 20K sea-of-gates technology.

The idea of combining placement with global routing attracted the attention of researchers only recently [SZE86, LUK86, DAI87, SHR87-1]. In [DAI87] this problem is addressed in an hierarchical fashion for the floor-planning of a building-block layout. An hierarchical bottom-up clustering methodology is applied. Clusters are formed from blocks, first with larger connectivity, then average. Each cluster has less than five members on each level of hierarchy. For every net, global routing graphs are refined recursively. The global routing solution is used to construct the I/O pad goal for the floor-plan evaluation at the next level. In this way floor-planning and global routing restrict the aspect ratio of the blocks.

The research reported here was supported in part by the National Science Foundation under grants MIP 86-17374 and DCR-8420935, and by Control Data Corporation under grant 86M107.

Szepieniec [SZE86] described an hierarchical slicing methodology. A sliced layout is obtained in a process of repetitive dissection of the initial chip domain into smaller rectangular domains by sets of parallel lines alternatively horizontal and vertical. Then optimization is performed by moving sub-slices within the parent slice. The slice is subsequently routed. The routing process consists of a systematic determination of the wiring pattern. The routing is performed immediately after moving the sub-slices and the results are used to update the net spans. These net spans are then used in the cost function which controls the movement of the sub-slices. Thus placement and routing are intertwined in this methodology.

In [LUK87] an hierarchical global wiring procedure for custom-design chips is described in conjunction with a methodology which generates global wiring before designing of individual macros. Macros are interpreted as arbitrary-sized rectangulars without any internal blockages for over-the-cell routing. The chip area is assumed to be given along with height and width of individual macros and wiring capacities of macro boundaries. The class of floor-plans considered in this paper belongs to the class of floor-plans sliced according to a definition proposed by Otten [OTT82]. A slicing floor-plan is recursively performed by repeated use of partitioning into four sub-slices. The partition stops at the leaf macro. The adjacency graph is generated as a representation of the floor-plan, and it is partitioned recursively. The global wiring is formulated as an hierarchical integer-programming model similar to that in [BUR83].

2 The Function of the Global Router in the Layout Process.

Traditionally placement and global routing are considered to be separate steps of layout. It was acceptable to place first and route later because it was relatively easy to approximate the resources required for routing in the placement step of layout. But such an approximation becomes very difficult when routing is performed over-the-cells and cells are of different size. This made it necessary to include a global routing step in the placement algorithm.

This paper consists of two parts. The first part is the description of a mathematical model of global routing. It contains also general algotithm and analysis of its properties (paragraphs 3-5). The second part (paragraphs 6-10) describes an application of the global router as a part of combined placing-routing procedure for the sea-of-gates design style.

The effective placement of unequally sized cells is a difficult problem by itself, even if no

consideration is given to the problem of routability. Additional constraints on routability enforced by placement only complicate the matter. Our approach to this complex combinatorial problem is based on the method of ‘‘successive augmentation.’’ A solution is obtained by adding new elements to partial solutions. This method, which might be called ‘‘constructive,’’ does not allow iterative improvement.

The solution is produced as a successive sequence of slices with an accumulated partial solution satisfying constraints on geometric fit and routability. A general structure for the algorithm is presented in **Figure 1,** and an intermediate step layout is shown in **Figure 2.**

Like the other work mentioned earlier, our work is based on clustering. Unlike earlier work, however, this clustering is not hierarchical. The disadvantage of hierarchical clustering during placement is that those who use it try to obtain a minimal covering rectangular for clusters on each level of hierarchy. Even the best asymptotic, 2-dimensional packing algorithms, without any additional constraints, achieve an asymptotic bound 5/4 with respect to optimal packing [BAK81]. It is easy to see that, for n levels of hierarchy, a proven asymptotic worst case bound for such methodology will be $(\frac{5}{4}\text{opt}+\beta)^n$, where β is a constant. Thus, utilization of chip area may be < 0.5 opt for small n. Such additional steps as compaction may help here, but it is not obvious that they are always applicable and effective. Our approach has been to limit the solution strategy to clustering without following an hierarchy. The main purpose is to achieve higher packing densities than other methodologies could achieve. In this methodology, clusters are not generated by preprocessing but created dynamically based on the results of previous steps. So the whole approach can be characterized as a ‘‘greedy’’ approach to layout, where each new step in the process is a new slice of layout with guaranteed routability [SHR87-1].

3 A Mathematical Model of Global Routing.

The chip to be routed may be described as a directed graph $G = \{I, J\}$ where I is the vertex set and J is the set of directed edges. Let $m = |I|$ and $n = |J|$, and let W be the edge incidence matrix. Then, $w_{ij} = 1$ if edge j leaves vertex i; $w_{ij} = -1$ if this edge enters vertex i; $w_{ij} = 0$ of edge j is not incident on vertex i. Each vertex of G represents a cell on a single layer of the chip. The edges represent cell adjacency. An edge connects two vertices that represent cells on different layers iff a via is possible between these two cells. The capacity of such an edge equals

the number of allowable vias between the two cells. An edge connects two vertices on the same layer iff the cells represented by these two vertices are adjacent in the routing constraints for that layer. For example, only horizontal (vertical) adjacencies are permitted on a horizontal (vertical) routing layer. Edges are assigned directions arbitrarily. The capacity of an edge that connects two cells on the same layer is equal to the number of routing tracks between the two cells. Let c_j denote the capacity of edge j and let p_j denote the cost of using edge j in a connection. We may assume that all edge capacities and costs are positive.

We assume that the nets to be routed have been decomposed into two pin nets. Let t be the number of nets and let A be the net incidence matrix. We assume that the two end points of each net are arbitrarily designated *source* and *sink*, respectively. $a_{ik} = 1$ if vertex i of G is the source of net k; $a_{ik} = -1$ if vertex i is the sink of net k; and $a_{ik} = 0$ otherwise. Let $x_{jk} = 1$ if edge j is used, in the chosen orientation, to route net k. $x_{jk} = -1$ if edge j is used, in the reverse orientation, to route net k. $x_{jk} = 0$ if edge j is not used in the routing of net k. The global routing problem may be formulated as:

$$\text{Minimize } \sum_j \sum_k p_j |x_{jk}|$$

subject to the constraints,

$$\sum_j w_{ij} x_{jk} = a_{ik}, \ \forall \text{ vertex } i \text{ and } \forall \text{ net } k, \qquad \text{(a)}$$

$$\sum_k |x_{jk}| \le c_j, \ \forall \text{ edge } j, \qquad \text{(b)} \qquad (1)$$

$$x_{jk} \in \{0,1,-1\}, \ \forall \text{ edge } j \text{ and } \forall \text{ net } k. \qquad \text{(c)}$$

The minimization function measures the cost of the routes; the constraint (a) ensures that each net is routed; and the constraint (b) ensures that the routing satisfies the capacity constraints of all the edges of G. Constraint (a) requires that for each net k and each vertex i exactly one edge be incident to i if i is an end point of the net. When i is not an end point, 0 or two edges are allowed. 0 implies that the vertex is not on the connection path while 2 implies that it is. Note that in the absence of the minimization function, constraint (a) does not prohibit cycles on the connection path. However, under the assumption of positive edge costs, cycles are forbidden. This model interprets global routing as a multicommodity flow problem and was introduced before by one of the authors [SHR87-2].

It should be noted that minimization of the connection length (or cost) is a primary goal only for nets with potential late arrival problems. For all other nets, all reasonable routes that satisfy the capacity constraints are acceptable. Shortest connections can themselves be a source of early signal arrival problems and hence undesirable. Thus we may replace the minimum connection length requirement of the above formulation with one that requires the completion of the maximum number of nets, with the length of each connection being within some limits α_k and β_k. The new formulation is:

Maximize $\sum_k z_k$

subject to the constraints,

$$\sum_j w_{ij} x_{jk} = a_{ik}, \forall \text{ vertex } i \text{ and } \forall \text{ net } k \text{ with } z_k = 1, \quad \text{(a)}$$

$$\alpha_k \le \sum_j p_j |x_{jk}| \le \beta_k, \forall \text{ net } k \text{ with } z_k = 1, \quad \text{(b)} \qquad (2)$$

$$\sum_k |x_{jk}| \le c_j, \forall \text{ edge } j, \quad \text{(c)}$$

$x_{jk} \in \{0,1,-1\}, \forall$ edge j and $\forall$ net k,

$z_k \in \{0,1\} \forall$ net k, $k = 1,2,...,t$,

Here, $z_k =$ { 1 iff the connection for net k is completed.
0 otherwise.

Another formulation for global routing results if we let the edge capacities be variable and require that all connections be completed, i. e., $\sum_{k=1}^{t} z_k = t$.

Minimize $\max\{c'_j\}$

subject to the constraints,

$$\sum_j w_{ij} x_{jk} = a_{ik}, \forall \text{ vertex } i \text{ and } \forall \text{ net } k \quad \text{(a)},$$

$$\alpha_k \le \sum_j p_j |x_{jk}| \le \beta_k, \forall \text{ net } k \quad \text{(b)}, \qquad (3)$$

$\sum_k |x_{jk}| \le c'_j$, $\forall$ edge j (c),

$x_{jk} \in \{0,1,-1\}$, $\forall$ edge j and $\forall$ net k.

Global routing under each of the above three models is NP-hard. Let us consider models (2) and (3) for the same graph G. Then the following theorem can be established.

Theorem 1.

If $c'_j \le c_j$ $(j=1,2,...,n)$, then any feasible solution of model (3) where $\sum_k z_k = t$ is an optimal solution of model (2).

Proof. First we need to show that any feasible solution of model (3) is a feasible solution of model (2). This follows from the fact that conditions (a) and (b) for both models are identical. Condition (c) of model (2) is satisfied by condition of the theorem. A feasible solution of model (3) for which $\sum_k z_k = t$ is an optimal solution of model (2) because it is a feasible solution of model (2) and the optimization criterion $\sum z_k$ produces the highest value t. □

Let us consider the graph G with the capacities of all edges equal to $c_1 > 0$. Denote the maximal number of completed connections in model (2) as $\gamma(c_1)$, where $\gamma(c_1) = \max k(c_1)$. We assume here that $\gamma < t$, where t, as before, is the number of connections defined on the graph G. In the more general case, when capacities of edges are given by the vector $c = [c_j]$, we can say $\gamma([c_j]) = \max k([c_j])$. Let us establish some facts relating maximal number of completed connections with capacities of edges.

Theorem 2.

1. If $c_2 > c_1$, then $\gamma(c_2) > \gamma(c_1)$.
2. If $c_j^{(2)} > c_j^{(1)}$ for $j=1,2,...,n$, then $\gamma([c_j^{(2)}]) > \gamma([c_j^{(1)}])$.
3. If $c_j^{(2)} \ge c_j^{(1)}$ for $j=1,2,...,n$, then $\gamma([c_j^{(2)}]) \ge \gamma([c_j^{(1)}])$.

Proof.

1. Let us consider the third statement first. From $c_j^{(2)} \ge c_j^{(1)}$ it follows that $c_j^{(2)} = c_j^{(1)} + \delta_j$, where $\delta_j \ge 0$. Let us represent a graph G with the vector of capacities $[c_j^{(2)}]$ as the union of two sets

of edges $G=\{I,J=J_1\cup J_2\}$, where $J_1=\{j\in J \mid c_j=c_j^{(1)}\}$ and $J_2=\{j\in J \mid c_j=\delta_j\}$. Let us divide a set of completed connections K into three subsets:

$$K_1=\{k\in K, j\in J_1:\{\sum_{j\in J_1}\omega_{ij}x_{jk}=a_{ik}\}$$

$$K_2=\{k\in K, j\in J_2:\{\sum_{j\in J_2}\omega_{ij}x_{jk}=a_{ik}\}$$

$$K_3=\{k\in K\setminus(K_1\cup K_2)\mid j\in J_1\cup J_2:\left\{\sum_{j\in J_1\cup J_2}\omega_{ij}x_{jk}=a_{ik}\right\}.$$

Let us denote the number of elements in each subset as k_1, k_2, and k_3 respectively, where $k_1=\gamma([c_j^{(1)}])$ according to definition.

Then $\gamma([c_j^{(2)}])\geq k_1+k_2+k_3=\gamma([c_j^{(1)}])+k_2+k_3\geq\gamma([c_j^{(1)}])$

2. If $c_j^{(2)}>c_j^{(1)}$, then $\delta_j>0 \ \forall j$, and graph $G_1=(I,J_2)$ with capacities of edges δ is isomorphic to graph G. At least one additional connection can be completed on the graph G_1, i. e., $k_2>0$. Thus, $\gamma([c_j^{(1)}])+k_2+k_3>\gamma([c_j^{(1)}])$.

3. The first statement is just a partial case of the second. □

4 High Level Description of the Algorithm.

From the results of the previous section it follows that any feasible solution of model (3), which satisfies the additional constraint $c'_j\leq c_j, \ \forall j$, is an optimal solution of model (2). Although the existence of a feasible or optimal solution to model (3), which satisfies the additional constraints on capacities of edges cannot be guaranteed, an attempt to find such a solution can be made. While we cannot propose a polynomial algorithm to find the optimal solution of model (2) we can propose a greedy optimization procedure for model (3) that provides an optimal solution to model (2). when the objective function value is sufficiently small.

The algorithm for global routing described in this paper belongs to the group of ‘‘greedy’’ algorithms because it considers only a subset of all connections at a time. This could represent a flaw in other circumstances, but in our case it is not. One of the main functions of this global router is to provide feedback to the placement procedure during the process of successive augmentation of placement where the positions of small subsets of all cells are decided.

We will describe two applications for our global router. The first is the traditional one where global routing follows placement. The function of the global router in this case is to assign wires to routing channels. The second application is in an integrated routing-placement procedure in which the router dynamically provides constraints for the placement steps. It also performs the traditional function of assigning wires to channels. A simplified high-level description of this algorithm for global routing can be given in the following form.

Initial Step

(1) Assign the capacity 1 to all edges of the graph model.

(2) Assign the cost 1 to all edges of the graph model.

(3) Mark those vertices of the graph which represent cells adjacent to the left side of the chip.

General Step

(1) Mark vertices of the graph adjacent to those marked during the previous iteration of the General Step.

(2) Reconsider the cost of the edges by assigning a cost of ∞ to all edges with a remaining capacity of 0.

(3) For each nonrouted connection with both ends marked, find a path of finite cost and subtract one from the capacity of each edge used by a connection.

(4) If a path of finite cost does not exist, add capacity to some edges and go to step 2.

Figure 3. GR Algorithm.

5 Some Additional Facts About the GR Algorithm.

Step 4 of the GR algorithm added capacity 1 to some edges of the graph G. From Theorem 2 it follows that addition to the capacities of all the edges results in an increase in the number of completed connections. At the same time it also follows from Theorem 2 that the increase can be achieved by adding capacity to just some edges of the graph, not necessarily all. The next question to answer is how to find a minimal set of edges to which to add capacity for each iteration of

algorithm GR. It is clear that if a connection cannot be found for the existing level of capacities, then the set of arcs with capacity 0 forms a cutset. Two approaches to this problem present themselves. The first is to re-route previously routed connections and, by so doing, to decrease the number of elements in the cutset. [SHR87-2] describes a procedure of this type. The success of such a procedure in reducing a cutset to the extent that the searched-for connection can be found is not guaranteed, while computational efforts may be quite substantial. Therefore, algorithm GR does not use this approach. The second possible solution is to add capacity to edges with current capacity 0. How this approach was applied in some situations can be seen from the following sections.

The choice of edges selected for capacity increase is based on probabilistic considerations. Let us consider a planar lattice graph **(Figure 4).** Let us assume that vertices to be connected are separated by *m* grids in the horizontal direction and *n* grids in the vertical direction. The number of distinct shortest paths S_{AB} between arbitrary vertices *A* and *B* is given by the formula $S_{AB}=\frac{(m+n)!}{m!n!}$. The number of distinct paths for arbitrary vertex *V* located inside the rectangle formed with *A* and *B* at opposite corners can be computed as $S_{AV}=\frac{(x_1+y_1)!}{x_1!y_1!}$, where x_1 is the number of horizontal grids and y_1 the number of vertical grids separating *A* and *V*. Similarly, $S_{BV}=\frac{(x_2+y_2)!}{x_2!y_2!}$, where $x_1+x_2=m$ and $y_1+y_2=n$. Then the probability that the shortest path between *A* and *B* will pass through vertex *V* will be $p(V)=\frac{S_{AV}\cdot S_{VB}}{S_{AB}}$. Let us select a vertex *Z* adjacent to *V*. The number of shortest paths between *Z* and *B* can be represented as $S_{ZB}=\frac{(x_3+y_3)!}{x_3!y_3!}$, where x_3 and y_3 are horizontal and vertical components of the distance between *Z* and *B*. (Without loss of generality we can assume that $x_3\le x_2, y_3\le y_2$.) Then the conditional probability of using vertex *Z* given that vertex *V* is used is $p(Z\,|\,p(V)=1)=\frac{S_{ZB}}{S_{VB}}$. The composite probability of using vertices *V* and *Z*, i. e., the probability of using edge *VZ*, can be determined by the formula

$$p(Z,V)=p(Z\,|\,V)p(V)=\frac{S_{ZB}}{S_{VB}}\frac{S_{AV}\cdot S_{VB}}{S_{AB}}=\frac{S_{AV}\cdot S_{ZB}}{S_{AB}} \qquad (4)$$

Two special cases can be identified: 1. $V=A$, and 2. $Z=B$ If $V=A$, then $p(Z\,|\,p(V)=1)=p(Z,V)=p(Z)=\frac{S_{ZB}}{S_{AB}}$. Let us observe that $S_{AV}=1$, for $A=V$. Then $p(Z,V)=\frac{S_{AV}\cdot S_{ZB}}{S_{AB}}$ for $A=V$. Likewise, $S_{ZB}=1$ for $Z=B$ and $p(Z,V)=\frac{S_{AV}\cdot S_{ZB}}{S_{AB}}$. Now the following theorem can be formulated.

Theorem 3.

The probability of using an arbitrary edge of the planar lattice graph by the shortest path between two points is given by formula (4).

Corollary 1.

The probability of using edges adjacent to the end points of a connection is given by $p_1=\frac{m}{m+n}$ for a horizontal edge and $p_2=\frac{n}{m+n}$ for a vertical edge.

Proof. For an edge adjacent to endpoint A, Theorem 3 gives $p(Z,V)=\frac{S_{ZB}}{S_{AB}}$. There are two cases: $x_3=m$ and $y_3=n-1$, or $x_3=m-1$ and $y_3=n$. For the first case, $S_{ZB}=\frac{(m+n-1)!}{m!(n-1)!}$, and for the second case, $S_{ZB}=\frac{(m+n-1)!}{(m-1)!n!}$. Then for the first case $p_1=p(Z,V)=\frac{(m+n-1)!m!n!}{n!(m-1)!(m+n)!}=\frac{m}{m+n}$ and for the second case $p_2=\frac{n}{m+n}$. □

Corollary 2.

If $p(Z,V)>p(R,W)$, then $S^{(1)}_{AB}(G\setminus(ZV))<S^{(2)}_{AB}(G\setminus(RW))$.

Proof. Let us consider two distinct edges of lattice graph G: (ZV) and (RW). Let $p(ZV)$ and $p(RW)$ be probabilities of the usage of these edges as defined by Theorem 3. Let $p(ZV)>p(RW)$, then from formula (4) it follows that $S_{AV}\cdot S_{ZB}>S_{AR}\cdot S_{WB}$. Let us remove the edges (ZV) and (RW) one at a time and denote the number of remaining distinct shortest paths between A and B on the reduced graph as $S^{(1)}_{AB}$ and $S^{(2)}_{AB}$ respectively. Then $S^{(1)}_{AB}=S_{AB}-S_{AV}\cdot S_{ZB}$ and $S^{(2)}_{AB}=S_{AB}-S_{AR}\cdot S_{WB}$. From this it immediately follows that $S^{(1)}_{AB}<S^{(2)}_{AB}$. □

Corollary 3.

1) If the set of edges of the lattice graph G is a minimal cutset J_0, then $\sum_{VZ\in J_0} p(V,Z)=1$.

2) But the converse is not true. It does not follow from $\sum_{VZ\in J_0} p(V,Z)=1$. that J_0 is a minimal cutset.

Proof.

1) $$\sum_{VZ\in J_0} p(V,Z)=\sum_{VZ\in J_0} S_{AV}\cdot\frac{S_{ZB}}{S_{AB}}=\frac{\sum_{VZ\in J_0} S_{AV}\cdot S_{ZB}}{S_{AB}}=\frac{S_{AB}}{S_{AB}}=1.$$

Here $\sum_{\substack{V\in I_1\\ Z\in I_2}} S_{AV}\cdot S_{ZB}=S_{AB}$ follows from the definition of a minimal cutset.

2) See **Figure 5.** It is sufficient to provide a counter-example.

Theorem 4.

A theorem analogous to Theorem 3 can be obtained for the 3-D lattice graph shown in **Figure 3.**

Proof. The difference between 2-D and 3-D cases is contained in the estimation of the number of shortest paths between two points in a 3-D graph. However, no particular formula for the number of shortest paths was used in proving formula (4). Therefore, the results of Theorem 3 also hold for the 3-D lattice graph of **Figure 3.** The number of distinct shortest paths in the 3-D graph of **Figure 3** depends on the positions of the end points with respect to the layers.

1)	$S_{AB}=m+1$	A and B are both located on the same horizontal layer.
2)	$S_{AB}=n+1$	A and B are both located on the same vertical layer.
3)	$S_{AB}=1$	A and B are each located on different layers. □

Corollary 1.

Corollaries analogous to corollaries 2 and 3 of Theorem 3 can be formulated for the 3-D case. The proof is self-evident.

From theorems 3 and 4 and their corollaries it follows that different edges of latice graph have different probabilities of being used by routes. When routing is performed one net at a time,

capacities of some edges can be ehausted earlier than capacities of other edges and edges with the remaining capacity equal to 0 become blocked for any future use. A number of blocked edges will increase with the number of net routed. Let us denote a set of blocked edges after routing of m nets as S_m and $|S_*|$ to be a minimal cutset size (see **Figure 5**) . Let $p(S_m)$ to be a probability that the set S_m is a cutset.

Theorem 5.

$p(S_{m+1}) \geq p(S_m)$

Proof. The following cases are possible: 1. If $S_{m+1} = S_m$ then $p(S_{m+1}) = p(S_m)$;

2. If $S_{m+1} \neq S_m$ then $S_m \subset S_{m+1}$. Consider the following subcases:

a) $|S_{m+1}| < |S_*|$

In this case $p(S_{m+1}) = p(S_m) = 0$

b) $|S_{m+1}| = |S_*|$

Now, $p(S_{m+1}) > p(S_m) = 0$

c) $|S_{m+1}| > |S_*|$

Then $p(S_{m+1}) = p(S_m) + p'$, where $p' = \sum_{\substack{i=1 \\ i \neq m}}^{m+1} p_i(S_m^i)$. Here $p_i(S_m^i)$ is a probability that the i'th combination of m edges from the set of $m+1$ edges is a cutset. From $|S_m| \geq |S_*|$ it follows that $p' > 0$ and therefore $p(S_{m+1} \geq p(S_m)$. □

Let $\bar{p}_{ab}(S_m)$ denote the probability that any path between vertices a and b ($a, b \in I$), traverses one or more edges from the set S_m (edges with zero remaining capacity). Then $\bar{p}_{ab}(S_m) = p_{ab}(S_m).p(S_m)$, where $p_{ab}(S_m)$ is the probability that a and b belong to different components of $G' = (I, J \setminus S_m)$. Note the following corollary to Theorem 5 can be formulated.

Corollary 1.

If $S_m \subset S_{m+1}$ then $\bar{p}_{ab}(S_{m+1}) \geq \bar{p}_{ab}(S_m)$

Proof. $\bar{p}_{ab}(S_m) = p_{ab}(S_m).p(S_m)$ and $\bar{p}_{ab}(S_{m+1}) = p_{ab}(S_{m+1}).p(S_{m+1})$. $p_{ab}(S_{m+1})$ is defined on the graph $G'' = (I, J \setminus S_{m+1})$ which is a component of the graph $G' = (I, J \setminus S_m)$. Thus, $p_{ab}(S_{m+1}) \geq p_{ab}(S_m)$. From Theorem 5 it follows that $p(S_{m+1}) \geq p(S_m)$. □

From Theorems 3-5 it follows that the probability of failure for any subsequent route will not increase if the number of blocked edges of the graph is kept unchanged. This goal can be achieved by reserving certain part of the capacity of each edge and releasing a part of it when no path can be found. How much of capacity to reserve and how to release it was decided by the heuristic procedure described in the second part of this paper.

6 An Incremental Global Router as Part of an Integrated Placement-Routing Procedure.

This approach was applied to the physical layout of large (20K) sea-of-gates chips. Sea-of-gates design style attempts to retain the advantages of gate-arrays while alleviating drawbacks associated with their low utilization of chip area. The major idea of this new organization of the chip is to make it possible to use any area for the placement of functions or routing. The CMOS *sea-of-gates* chip contains a matrix of primitive cells without any special channels separating the rows and columns. Each cell consists of rows of *p*-type and *n*-type transistors and includes several horizontal and vertical tracks distributed over the body of the cell and located on two layers of metal. Such an organization of the chip allows one to route over the cell and ignore the existence of transistors located there or to use the cell as part of the function. Primitive cells are used to form a library of functional cells. These functional cells differ in size but are usually rectangular **(Figure 2).**

Such features of the design style as variable size of functional cells and over-the-cell routing make this technology difficult for fully automatic layout.

7 General Structure of the Solution.

Our solution method is based on the decomposition of the original large problem into subproblems of smaller size. The solution is produced constructively, one slice at a time, with each partial solution satisfying certain constraints on routability. The construction of the solution starts from the left-most slice and proceeds to the right-most slice. The general structure of the algorithm is given in **Figure 1.**

Our concept of slice is different from the traditional one [SZEP86, OTTE83, OTTE82, LAUT80]. Usually the chip is hierarchically bisected into a set of rectangles, which are called slices. This division is an attempt to find a rectangular dual to the connectivity graph, and it is

performed as a floor-planning step of the custom design. In our approach, slices are generated dynamically one after another in a constructive manner, and they may have different sizes in both vertical and horizontal dimensions. During slice formation, the chip is divided into two parts: the placed part, called the closed area; and the nonplaced part, called the open area **(Figure 2).** The closed area is filled with the already placed functional cells and their interconnections. So it cannot be used for the placement of any additional functional cells. Because of the left-to-right direction of the placement, the closed area is located on the left side of the chip. The closed area can be defined as a minimal polyhedron covering the occupied cells. The open area of the chip is the set of unused cells to the right of the closed area. The border area is a subset of the open area comprising the cells which have a common edge or vertex with the already occupied cells in the closed area.

The slice is defined as a connected subarea of the open area. The lower boundary and upper boundary of the slice are cell rows. The left boundary consists of basic cells adjacent to the closed area. The exact position of the right boundary of the slice is determined after completing the slice placement. The height of the slice is determined by the number of cell rows between the upper and lower boundaries.

8 Routability Analysis by the Incremental Global Router

Global routing as a part of the placement procedure is one of the most important and unique characteristics of this placement algorithm. The function of global routing in a conventional layout system is the reduction of complexity by elimination of the net ordering problem from the detailed routing. The global routing procedure in this system is used to reserve routing resources during the placement process and to make the placement dependent on availability of such resources for detailed routing. It also measures routability by counting the number of unroutable connections before the detailed routing and reports results of such measurements. Such an application of global routing allows us to estimate the necessary number of routing tracks in the border part of the open area and to perform placement of functional cells for the new slice accordingly. This means that some locations where requests for routing tracks are high will be reserved for routing and nothing will be placed there and functional cells can be placed only at those locations where they can be routed.

The central point of this global routing is prevention of the blockage of some long connections by shorter connections and concentration of such blockages in certain areas of the chip. This goal is achieved by the dynamic update of the track map after each routing step. The critical available tracks are reserved (temporarily blocked) for future usage. When the routing cannot continue without reserved tracks, reserved tracks are released step by step in a given order.

```
    procedure global-routing ;

    begin
(1)   Expand closed area and modify track map;
(2)   for each net with new pins do
      begin
(3)     Release tracks around pins of the net;
(4)     while there are new pins do
        begin
(5)       Mark partial-route as target and new pins as source;
(6)       Modified-Lee-Router ;
(7)       while failed and reserved tracks exist do
          begin
(8)         Release some tracks;
(9)         Mark source and target again;
(10)        Modified-Lee-Router ;
          end;
(11)      Move a new pin and its path to partial-route;
(12)      Reserve released tracks;
        end;
(13)    Reserve tracks around pins of the net;
      end;
    end;
```

Figure 6. Global Routing Procedure.

8.1 Global routing procedure

Figure 6 shows the implementation of the global routing procedure for a slice. The *route* for a net is defined as a set of pins and their interconnections (see **Figure 7**). The route has the structure of a Steiner tree (not necessarily of optimal connection length). All pins in the closed area and their interconnections compose a partial route for a net. All cells which contain the partial route become the target of the router. All cells which contain new pins of the same net in a new slice become sources. In **Figure 7**(a), pins A and B and their interconnection path are targets for the source pins C and D. The connection path from the source to the target is constructed by a modified Lee's router. A source pin and a new connection path are included into a partial route and the new partial route becomes the target (pins A, B and C and their interconnections in **Figure 7** (b)). The remaining pins become the next source (pin D in **Figure 7** (b)). This process is continued until every source pin becomes a target. There is no special ordering of source pins in this algorithm. The closest source pin to the target will be connected first and included into the target. The above process is repeated for every net with pins within the latest slice.

When the router fails to find a path or the path found is long, then reserved tracks are released step by step. The router retries to find a path which is sufficiently short. This procedure repeats until a path is found or no more reserved tracks exist. All released tracks, except tracks used by the current path, are reserved again for the next routing process.

8.2 Modified Lee's Router for Global Routing

A basic cell is represented by two cells for the purpose of routing: one cell for the vertical layer and another cell for the horizontal layer. Two cells are called *adjacent* if their corresponding tracks are adjacent on the chip. There are at most three adjacent cells for each cell. The modified Lee's router operates with three types of cell maps: *track map*, *length map* and *width map*. A cell of each map stores some information for the corresponding cell of the chip.

A cell of the track map *tmap* stores the number of tracks for each track type. The tracks are classified into blocked tracks, active available tracks and reserved available tracks. The active

tracks can be used immediately for interconnection paths, the reserved tracks are temporarily blocked for usage. The router uses only active tracks for the current stage of routing ($tmap_e$ is the number of active tracks of cell *e*). It does not consider the positions of tracks within a cell, so it is called a global router.

The source and target for this Lee-type router are sets of cells. In the length map *lmap*, each cell contains the length of the shortest path found so far from the source cells to this cell. In the beginning, every source cell is marked 0, every target cell is marked -1 and all other cells contain $+\infty$ in the length map. The connection path from the source to the target is constructed by propagation of the wavefront through the cell maps (statements 3-14 in **Figure 8**). Initially, the set of source cells becomes the wavefront *W*. The position of the next wavefront *NW* is defined as a set of unused cells which are adjacent to the current wavefront *W* and were not included in any previous wavefront. An unused cell *e* is defined as a cell which satisfies $tmap_e > 0$. Every cell of a new wavefront in the length map is given the value *clength* which is the shortest path length. If a wavefront stops without reaching the target, there is no path between source and target (see statement 15 in **Figure 8**). If a wavefront reaches the target, the shortest path has found.

The notion of path width is used to break ties between paths of equal length. The width of a cell *e* is defined as the number of active tracks, $tmap_e$, within the cell. The width of a path is a total width of all cells along the path (**Figure 7**). Each cell of the width map *wmap* stores the maximum width of all shortest paths found so far from the source cells to this cell. Even though the shortest path to the cell was found, another wider path with the same length can be found later (see statement 9 in **Figure 8**).

To trace back the path (see statements 16-22 in **Figure 8**), a touched cell is selected from the target as the first cell in a path. Ties are broken by selecting a cell with wider path width. The next cell in a path is found from adjacent cells to a current cell. The widest path is selected among all paths available because it allows more even distribution of routes. The track map is modified by marking one active track from each cell along the path as a blocked track.

```
procedure Modified-Lee-router ;

/* T: a set of target cells; S: a set of source cells */
/* W, NW: wavefronts; P: path */

begin
(1)   Initialize length map lmap, width map wmap and wavefront W;
(2)   clength ← 1;

      /* Propagation of wavefronts */
(3)   while W ≠ ∅ and W ∩ T = ∅ do
      begin
(4)     NW ← ∅;
(5)     for e ∈ W do
(6)       for adjacent cell e´ of cell e do
(7)         if lmap_e´ ≥ clength and tmap_e´ > 0 then
(8)           if lmap_e´ = clength then
(9)             wmap_e´ ← max{wmap_e´, wmap_e + tmap_e´}
(10)          else
              begin
(11)            lmap_e´ ← clength;
(12)            wmap_e´ ← wmap_e + tmap_e´;
(13)            NW ← NW ∪ {e´};
              end ;
(14)      W ← NW ; clength ← clength + 1;
      end;

      /* Failed to route */
(15)  if W = ∅ then "failed" ;

      /* Succeed, trace back the path */
(16)  Find e ∈ W ∩ T with maximum wmap_e;
(17)  P ← {e}; rwidth ← wmap_e; rlength ← clength;
(18)  while rlength ≥ 0 do
      begin
(19)    rlength ← rlength − 1;
(20)    Find an adjacent cell e´ of cell e such that
          lmap_e´ = rlength and rwidth = wmap_e´ + tmap_e;
(21)    e ← e´ ; rwidth ← wmap_e
(22)    P ← P ∪ {e};
      end;
    end;
```

Figure 8. Modified Lee's Router.

8.3 Track Reservation Strategy

There are three levels of critical tracks.

reserved. Otherwise, the global router will use tracks in this area preventing future placement of functional cells in the open area and making future routing difficult.

(2) Available tracks adjacent to pins of non-completed nets are considered next critical (**Figure 9**). Thus, one available track is reserved from every adjacent cell to the pins of non-completed nets. This is done to prevent the situation, when all adjacent cells around pins are blocked. In this case the pins are isolated from the outside world and the new pins within the slice cannot be connected to the old pins.

(3) Finally, some number of tracks is reserved from every basic cell in the routing area. This is done to maintain as many non-blocked basic cells as possible. By preventing tracks of basic cells with a small number of available tracks from being used and by encouraging the use of tracks of basic cells with large number of available tracks, the connection wires will be distributed more evenly and the number of blocked cells will be decreased. Then at later stages, the probability of finding paths is higher.

At first, the router does not use the reserved tracks for a path. When the router has failed to find a path, then reserved tracks are released in reverse order of criticality and the router retries. The order of release for the reserved tracks is (3) $\rightarrow$ (2) $\rightarrow$ (1). If the router fails after releasing all reserved tracks, there is no path without re-routing previously routed connections.

Our experiments show that the track reservation strategy works very well. Without track reservation, the connection length at first is close to the Manhattan distance. But later, because of congestions of blocked cells, the connection length becomes longer with respect to Manhattan distance and the total number of tracks used for interconnections also becomes larger. As a result, in the final stages of the algorithm without reservations, the number of failed connections and the number of unplaced functional cells increase substantially.

9 Experimental Results

This CMOS *sea-of-gates* chip design style is used by several major computer manufacturers (UNISYS, CDC, ETA and others). Placement for such technologies is performed semi-automatically with the help of the graphic work-stations. Experimental data were provided by Control Data Corporation.

The software system of which the global router is a part was written in Fortran 77 and runs on the Apollo families of workstations, DN300 series and DN3000 series. This program contains approximately 13000 lines of code and requires 1.5 MB of memory.

All test cases described in this paper represent real designs implemented on 20K *sea-of-gates* technology. Two methodologies were applied to get the physical layout. The traditional one, which included semi-automatic placement and automatic routing, and the new one described in this paper. The traditional methodology usually requires substantial time to achieve initial placement and then several iterations of the cycle, place and route, to improve its routability. Our methodology achieves placement in one pass automatically and then routes it. Few iterations were performed at most. As a result, weeks were cut from the design time for *sea-of-gates* chips.

The quality of the layout is measured by the number of unplaced functional cells and the success rate of the detailed routing. Both fully automatic and semi-automatic methodologies achieved 100% placement, but it takes 1-3 hours on Apollo DN330 for automatic methodologies and many days with the semi-automatic placer. In both cases, the results of placement were sent to the Cyber mainframe for detailed routing. The number of unconnects resulting from an automatic placement was equal or slightly larger then the number of unconnects produced by a semi-automatic placement after several iterations (see **Table**). For the automatic placement, the percentage of completed connections ranged from 99.95% to 100%.

The designs considered in this study differed in a variety of parameters. The most important for these were utilization of chip area by functional cells, and number of functional cells, number of connections. Additional experiments were conducted by industry experts on their own premises and we were informed that the results achieved by the automatic system were interpreted as very encouraging.

Figure 10 represents successive stages of layout for the last test case from **Table**. In this figure, the white rectangles are functional cells and the black areas are used for routing.

10 Conclusion

This work demonstrates that the traditional decomposition into placement and routing may not be an optimal strategy for some new technologies. In the case of *sea-of-gates*, it became clear that routability is so important issue that it cannot be addressed by simple predictive techniques and it requires more detailed treatment. In this work, this problem was solved by successive augmentation algorithm which includes geometric and routability components. The routability aspect is resolved by the new incremental global router. The correctness of such approach was confirmed by applications to design of large sea-of-gates chips.

REFERENCES

[BAK81] Brenda S. Baker, et al, "A 5/4 Algorithm for Two-Dimensional Packing," Journal of Algorithms 2, pp.348-368, 1981.

[BUR83] M. Burstein, "Heirarchical VLSI Layout: Simultaneous Placement and Wiring of Gate Arrays," VLSI ’83, International Conference on VLSI, Trondheim, pp.46-60, 1983.

[DAI87] W. Dai, M. Sato, E. Kuh, "A Dynamic and Efficient Representation of Building-Block Layout," 24th Design Automation Conference, pp.376-384, 1987.

[GARE79] M. R. Garey, D. S. Johnson, "Computers and Intractability," W. H. Freeman & Co., 1979.

[LAUT80] U. Lauther, "A Min-cut Placement Algorithm for General Cell Assemblies Based on a Graph Representation," Journal of Digital Systems, Vol-4, No-1, pp.21-34, 1980.

[LEE61] C.Y. Lee, "An Algorithm for Path Connections and its Application," IRE Trans. Electron. Comput., pp.346-365, Sept., 1961.

[LUK86] W. Luk, D. Tang, C. Wong, "Heirarchical Global Wiring for Custom Chip Design," 23rd Design Automation Conference, pp.481-489, 1986.

[OTTE82] R. H. J. M. Otten, "Automatic Floorplan Design," 19th Design Automation Conference, pp.261-267, 1982.

[OTTE83] R. H. J. M. Otten, "Efficient Floorplan Optimization," International Conference on Computer Design, pp.499-501, 1983.

[SHR87-1] E. Shragowitz, J. Lee, S. Sahni, "Placer-Router for ‘Sea-of-Gates’ Design Style," International Conference on Computer Design, pp.330-335, 1987.

[SHR87-2] E. Shragowitz, J. Keel, "A Global Router Based on Multicommodity Flow Model," Integration, the VLSI Journal 5, pp.3-16, 1987.

[SOUK78] J. Soukup, "Fast Maze Router," 15th Design Automation Conference, pp.100-102, 1978.

[SZEP86] A. A. Szepieniec, "Integrated Placement/Routing in Sliced Layouts," 23th Design Automation Conference, pp.300-307, 1986.

Chip type	20k						
Matrix of basic cells	127 x 94						
Utilization of chip area by functional cells (%)	44.9		66.9		69.2	75.7	
Number of functional cells	1774		2069		1694	3473	
Number of connections	8163		11579		11290	15816	
Algorithm	SP †	manual	SP †	manual	SP †	SP †	manual
Placement (%)	100	100	100	100	100	100	100
Global routing (%)	100		100		100	100	
Detailed routing (%)	100	100	100	100	99.95	99.99	99.99
† Static selection + Position priority							

Table. Experimental Results for 20K Chip

Figure 10A. Successive Stages of Layout

Figure 10B. Successive Stages of Layout