Layer Assignment for Rubber Band Routing

Tal Dayan[*]
Wayne Wei-Ming Dai[*]

UCSC-CRL-93-04
January 20 1993

Board of Studies in Computer Engeeniring
University of California at Santa Cruz
Santa Cruz, CA 95064

EMail: tal@cse.ucsc.edu

[*] This work was supported in part by the National Science Foundation under the Grant MIP-9058100, and in part by IBM Corporation.

Abstract

The flexibility of the rubber-band wire model is very promising for routing and optimizing today's complex VLSI and MCM. In this paper we described a practical layer assignment algorithm used with a rubber band router. The algorithm uses a steepest descent approach with an heuristic function which estimate the wiring length of the generated assignment. We formulate the cost function and present an efficient way to compute it. The use of the cost function enable to control the final layout and to achieve balance between wire length and number of vias. The algorithm can be use as well for cost-driven one an a half layer designs in which one wiring layer has only short "jumpers" and is embedded in the ground layer.

1. Introduction

The complexity and the of today's VLSI and multi-chip modules (MCM) and the desire for cost and performance driven routing, make the work of automatic routers more and more computation consuming.

Three dimensional maze routing [Hanafusa90], [Miracky91] for example, which is a widely used technique, suffer from sensitivity to net ordering. When routing net at a time, the router consider only narrow view of the problem and can make sub optimal decisions which affects the final result. Another approach to solve the multi layer problem is to divide it into pairs of x,y layers. This restriction of one layer one direction results in higher number of layers and is restricted to rectilinear routing. Slice [KEY-YONG92] uses an original method in which considers all the nets in a sliding window and peals a maximal planar subset of the interconnect for each layer. This approach has short run-time and interconnect length. Its disadvantage is that it grid based, does not balance between the amount of wiring on the layers which, can result in high crosstalk and low yield.

A common representation used by today's routers is a geometric representation that captures the exact location of the wires. This representation requires the router to determine the exact location of the wires in early stages and makes moving wires, and terminals around time consuming task. A more flexible representation is the rubber-band sketch. The rubber-band sketch represents wires as elastic paths that contract to the shortest possible length that maintains the same topology. Because rubber-bands specify topology and not the exact location of wires, they support efficient sketch modifications such as moving terminals, and rerouting wires.

In this paper we describe the rubber band router a MCM design system called Surf, with emphasis on the layer assignment phase. The Surf system is based heavily on the properties of rubber band sketch and generates an optimized multi-layer, rubber band-sketch for multi terminal nets. To the best of our knowledge, this is the reported router of its kind.

This paper is organized in the following manner. In section 2 an overview of Surf system, in which the proposed layer assignment algorithm has been implemented, is described. Section 3 includes a formulation of the layer assignment problem as an combinatorial optimization problem and the proposed heuristic. In section 4, an overview of the layer assignment algorithm is presented. In section 5 the cost function is defined. Section 6 describes the assignment states during the operation of the layer assignment algorithm. Section 7 describe the branch ordering done at the end of the assignment. Section 8 describes how the best assignment of a specific net is computed. In section 8 the detour estimation algorithm is described. Sections 10 describes the incremental implementation of the algorithm and section 11 describes how the algorithm can be used in a one and an half layer, cost driven design. Section 12 concludes the paper.

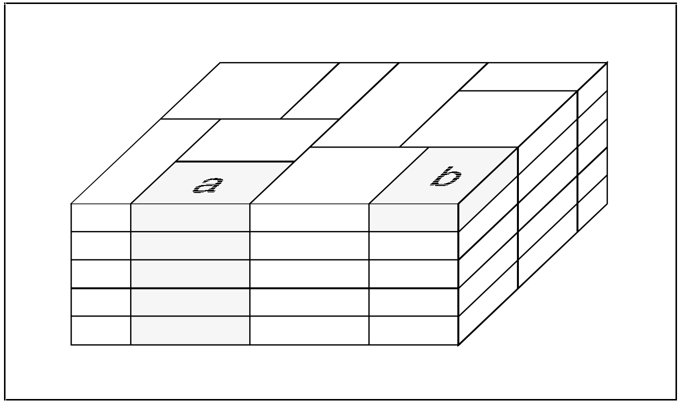

Figure 1 - To reduce the routing complexity and the sensitivity to net routing ordering, the global router splits the MCM three dimensional space, into three dimensional bins (a). The layer assignment, reduces the problem furthermore into single-layer, two-dimensional bins, on which the actual topological routing is performed (b).

2. Surf Router Overview

Surf is an MCM design system which supports automatic and interactive layout of MCM from the net list and design rules specification to the final layout. An important concept of Surf is the Least Commitment Principle (LCP). The LCP states that a decision should be deferred as much as possible. The rationale behind this principle is that over-commitment in early stages of the design, when less information is available, may adversely affect the results by over-constraining the later stages. As a consequence of this principle, Surf works in a series of successive refinements, starting from an abstract representation and ending with the detailed geometric artwork. The rubber-band model fits well into this philosophy because it captures the conductivity and topology of the wiring without committing to exact geometric locations. The exact geometric location are determined later, after the entire rubber band sketch is known and verified.

The main steps in Surf routing are

Global Routing (Figure 1) - In this step the three dimensional MCM routing problem is split into smaller three dimensional routing problems called *bins*. These sub-problems are later solved separately and then merged back into a single solution. This subdivision of the problem reduces the complexity of the overall solution. When processing a partition, this step analyzes all the nets "simultaneously" and as a result, the global routing does not suffer from the net ordering problem.

Layer Assignment (figure 1) - In this step, each bin is divided into a set of two-dimensional problems, each on a different layer. This is done by adding necessary vias and assigning nets, or portions of nets, to layers. The layer assignment phase is the main subject of this paper.

Rubber-band routing (Figure 2b)- This phase does the topological routing of the single layer sub-problems (bins). The rubber band router, represents the rubber bands as paths over a constrained Delanay triangulation [LU91] [Chew89] and routes the nets one at a time using shortest path algorithm. The resulting rubber band sketch is then optimized by relocating vias and Steiner points, and by improving the topology of the wiring. This phase is discussed in details in [Dayan91].

Conversion to geometric layout - This final phase converts the rubber band sketch into a detailed geometric layout in the rectilinear octilinear or any-angle routing modes. This is done by converting the rubber band sketch to an extended rubber band sketch (figure 2c) [DKS91] that satisfies the spacing constrains. Once, all spacing constrains are met, the ERBS is transformed to a final geometric layout (figure 2d) [STAPA92].

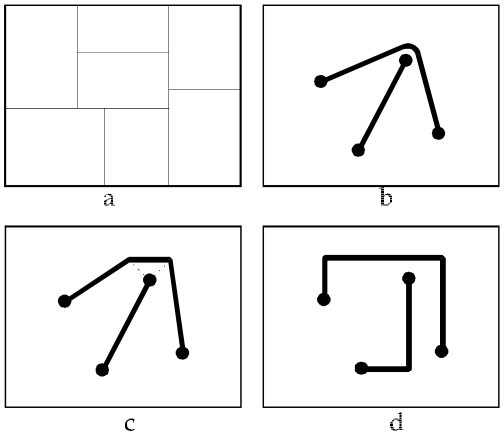

Figure 2 - **Routing stages in Surf**. The global routing and division to bins (a). The rubber band sketch of a bin (b). The extended rubber-band sketch which satisfies the spacing constrains (c) and the final rectilinear (or octilinear) artwork (d).

3. Layer Assignment Problem

The layer assignment algorithm converts a multi layer routing problem to a set of single layer sub-problems, one for every layer. The layer assignment algorithm accepts the net list, number of layers to be used, and the boundaries of the routing area.

The algorithm breaks the multi terminal nets to two terminal nets, introduces vias, and assigned subnets to layers.

The output of the algorithm is a set of single layer, two terminals nets, routing problems, one for each layer. Each such sub-problem includes the terminals and vias reside on this layer and an ordered list of point to point connections (called *branches*). This output is used by the rubber band router which routes the branches by the order specified by the layer assignment..

4. Layer Assignment Overview

The layer assignment phase of Surf accepts a three dimensional, multi-layer, routing problem and reduces it into a set of single-layer, two-terminal net problems, one for each layer. As a by product, it also determines the order in which the branches are routed.

The operation of the layer assignment is done in three steps

Breaking the multi-terminal nets into two terminal nets. This is done by converting the multi-terminal nets to Steiner trees using a geometrical and topological search algorithm [DAYAN91].

Initial partial assignment - The multi-terminal terminal nets whose all their points which are on the bin boundary are assigned. These kind of nets are quite common as they global assignment creates cross points on the bin boundary. The initial assignment is done by applying the CPSP algorithm and 'pealing' a maximal set of non conflicting nets for each layer.

Completion step (see algorithm 1) - the initial assignment is completed by assigning all the non assigned two terminal net. It uses the steepest descent approach and a cost function which prefers assignment with shorter wiring and less vias. On each iteration, a two terminal net whose best assignment will increase the cost the least is assigned to its best assignment. The difficulty in implementing this approach is how to find which net to assign and to what assignment. In following sections we describe an efficient algorithm that find the best assignment of the best two terminal net.

Optimization step - the completed assignment is improved by iterativly reassigning a two terminal net with the most cost improvement. It is similar to the completion step except that now all the nets are candidates for reassignment rather than only the free nets.

Ordering the branches. In this step, the point to point connections (*branches*) in the solution are ordered by their preferred routing order. This order, which is constrained to guaranty planarity, is used by the rubber band router when routing the branches one at a time.

The completion and improvement steps can also be performs with non greedy algorithm such as group migration or simulated annealing, which are less expected to be trapped in local minimums .

```
$\Psi \leftarrow$ set of non assigned two terminal nets.
while $\Psi \neq \emptyset$ do
    $n \leftarrow$ a net in $\Psi$ with minimal cost increase.
    $a \leftarrow$ best assignment of $n$.
    assigned net $n$ to $a$.
    $\Psi \leftarrow \Psi - \{n\}$.
end while
```

Algorithm 1 - Completing the assignment by assigning all the free two-terminal nets. On each step, the two terminal nets which increases the cost the least is assigned to its best assignment.

5. The Cost Function

The layer assignment algorithm uses a cost function defined over the set of feasible layer assignment solutions. This function is used to guide the algorithm to better solutions which achieve shorter wiring and low number of vias.

The cost function of a solution S is a sum of two terms, the cost of the layer crossing vias and the cost of the total wire length. The user controlled via cost parameter K_v is used to balance the number of vias and the total wiring length.

$$Cost(S) = K_v \bullet Via\_Cost(S) + Wiring\_Cost(S)$$

The via cost of a solution S is the sum of the via cost of all the points. The wiring component of the cost is the estimated total wire length. As a solution for the layer assignment problem does not include the detailed routing , the wiring length can not be directly calculated from it, and a more indirect approach is required. Several possibilities for the definition of the wire length function are:

Minimal wire length - This is the wire length of the routing with minimum wire length among all the routing solutions. This approach is probably the most desirable from theoretic view point since it is independent of the router in use and can possibly lead to a optimal routing. This function however is exponential in nature cannot be calculated efficiently.

Actual wire length - This is the wire length after routing with a specific router used in the system. This approach has the advantage that it considers the strengths and weakness of the actual router in use. This however requires to perform a detailed routing every time the cost function is evaluated during the layer assignment algorithm.

Estimated wire length - When used with a proper estimation function, this approach combines the benefits of the minimal and actual wire length approaches as it can be calculated efficiently. The rest of the paper is focused on this approach and includes an estimation function and an efficient method to compute it.

For a layer assignment solution S, the length of the wiring can be expressed as a sum of two terms, the total direct length of the branches in S and a value which represent the extra wiring due to detours. While the total length of the branches can be calculated efficiently and accurately, the detour component represents the estimated part of the wiring length. To estimation function used in is based on relationships between *components*.

(DEF) A *component* is a maximal connected set of branches which are assigned to the same layer.

Components are important for the detour estimation as each component acts as an obstacle for wires of other components, and thus the detours length depends on the components sizes, location, and interrelation.

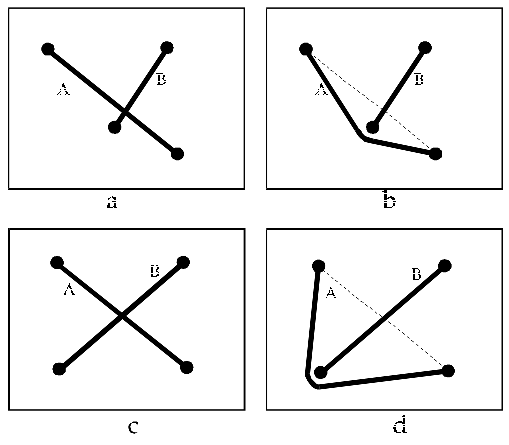

Figure 3 - Pair wise detour of component pairs. The estimated detour between the pair in (a) is low as it can be routed with short detour (b). The estimated detour of the pair in (c) is higher as any routing of the pair will have long detour (d).

(DEF) A pair wise detour of two components is the difference in total wire length between the shortest routing of the two components independently of other components, and the sum of their length.

The pair wise detour function is non negative and is greater than zero only if the two components are on the same layer and intersect which each other.

Figure 3 illustrates the pair wise detour in two simple cases of two terminal (i.e. single branch) components. The components in (a) do intersect but have a relatively low detour as the pair can be routed with a short detour (b). The component pair in (c) has an higher detour as every routing of the pair will have a long detour (d).

The pair wise detour function indicates the *amount* of conflict between components. This new concept of *continuous* conflict scale, as opposed to binary intersection check, enables the proposed layer assignment algorithm to consider not only if wire do intersect and conflict which each other, but also the *amount* of the conflict.

The pair wise detour function is used to estimate the total detour of a layer assignment solution S by summing the pair wise detour of all the components pairs.

This estimation, which is the core of the heuristics in the proposed algorithm is neither an upper bound nor a lower bound of the actual detour.

6. Assignment States

After converting the multi-terminal nets into Steiner trees, each tree edge is divided into *sections* (Figure 5). These section which represent the smallest unit of assignment, are fixed during the entire layer assignment. The layer assignment is modeled as a process in which each section is assigned to a layer. During the layer assignment, a section may be free which means that it has not yet been assigned to any specific layer.

The use of sections as the basic assignment units implies that vias can be introduced only on a section boundary, and thus the accuracy of the layer assignment result depends on the granularity of the sections. Our experience shows that having 5-10 evenly spaced section for each two terminal net is sufficient, and that increasing the number of sections above that value does not improve the result significaly. The main reason for that is that the layer assignment is done for each bin separately, and the bin already represent small parts of the routed plan. In addition, the inaccuracy in the via position can be fixed by a force-driven via relocation optimization on the rubber-band itself as described in [DAYAN91].

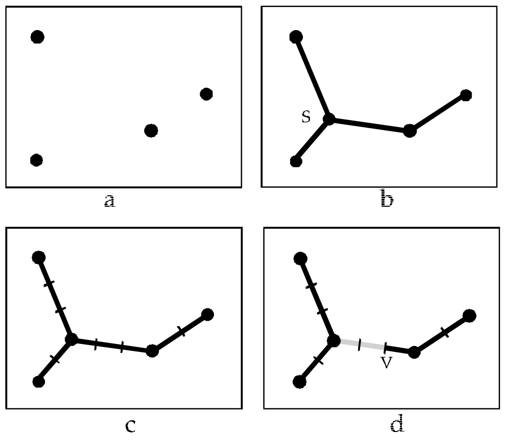

Figure 4 - Steps in assigning a multi-terminal net. The terminals in (a) are converted to a Steiner tree (b) with introduction of a Steiner point S. The tree edges are then divided into sections (c) and the sections are assigned individually (d). Two adjacent sections, assigned to different layers, define a via (V).

During the layer assignment process, the state of the assignment is represented by the function

$$color: \Pi \Rightarrow L \cup \{\perp\}$$

which maps the fixed set of sections Π to layers or to the free (unassigned) layer $\perp$.

The specific assignment algorithm described in this paper assigns on each step *all* the sections of a two terminal net. As a result, states in which a two terminal net has both assigned and unassigned sections are not reached.

The cost of a state is defined by the overall cost function described earlier. If a state represents a partial assignment, the cost is calculated only on its assigned sections. Branches are represented in a state by a maximal set of adjacent sections which belong to the same tree edge and are assigned to the same layer. Two or more adjacent sections assigned to different layers define a layer crossing at the junction. If the sections are on the same tree edge then this is a new via point, introduced by the layer assignment.

7. Branch Ordering

When the assignment is done, the branches defined by the assignment are ordered to achieve low wire length and to guaranty planarity during the rubber band routing. This order is use by the rubber band router which routes the branches one at a time. Figure 5a shows an example of how a wrong order of the branches can result in much longer wiring. Figure 5b shows how an improper ordering may result in non planarity of the wiring which does not allow the single layer rubber band router to complete the routing.

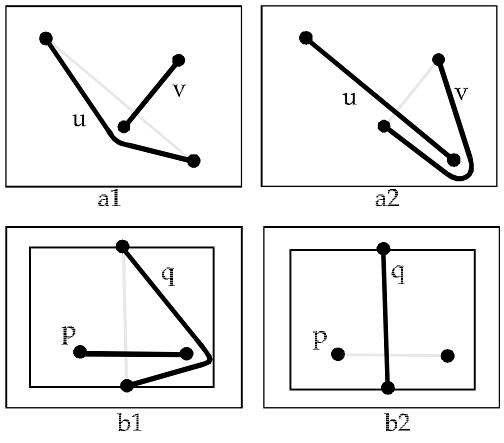

Figure 5 - The importance of branch ordering. In case a, when branch v is routed first (a1) the detour is shorter then when u is routed first (b2). In case b, when p is routed first (b1) the sketch is planar but when the order is reversed (b2), branch p cannot be routed as branch q splits the bin into two non connected areas.

The branch ordering is done in two steps. First the a basic order is generated to achieve low wiring length and then it is constrained to guaranty planarity.

The basic order is done in component units as there is no conflict between branches of the same component. After ordering the components, the branches of each component are selected in arbitrary order. The order is based on an order function which indicates how desirable it is to route a pair of component in a specific order. This function is directly derived from the overall cost function used by the layer assignment.

(DEF) Let *a, b* be a pair of component. The function *ordered_pairwise_detour(a,b)* is defined as the extra length of routing *b* after *a* compared to routing *a* and *b* independently.

This function which has non negative values, indicates the cost of routing the pair in a specific order.

(DEF) Let Z be a set of components and $c \in Z$ a member of that set. The function ordered_set_detour(c, Z) as

$$ordered\_set\_detour(c,Z) = \sum_{p \in Z} ordered\_pair\_detour(c,p)$$

This function has lower values for component which are more desirable to be routed before all the other components of the set Z.

The basic oriented ordering is one by iterativly choosing the component which is the most desirable to be routed first among all the remaining components. The branches of each component are then ordered arbitrarily as there relative order does not effect the wiring length. Since that there is no interaction between components assigned to different layers, the ordering can be done independently for each layer.

After ordering the branch for short wiring length, the order is constrained to guaranty planarity. This is done by identifying the branch which can split the bin into non connected areas and moving these branches to the end of the list.

(DEF) Let *c* be a component and let $b_1..b_n$ be the branches of *c* as ordered by the wire length oriented ordering. A branch b_i is called *hot branch* if the number of non connected areas of the bin after routing (independently of other components) $b_1..b_i$ is larger then the number of non connected are after routing $b_1..b_{i-1}$. A branch which is not hot is called *cold* branch.

This definition is for a specific order of the branches, so branch can be *hot* in one ordering of the branches and *cold* in another.

Let $b_1..b_m$ be the branches of the layer assignment as ordered by the wire length oriented ordering. This list is a merge of two sub lists of the cold branches and of the hot branches. The ordered is constrained to guaranty planarity by creating a new order which includes the cold branches sub list and then the hot branches sub list.

(LEMMA) The new constrained order insures planarity, or in other word, the local router will be able to complete the routing of the branches one at a time in that order.

(PROOF) By contradiction. let b_0 be the first branch which its two ends can not be connected. This implies that the two ends are in two non connection areas of the bin and the two areas are not connected due to previous routed branches. Branch b_0 can be either cold or hot:

Case 1: Branch b_0 is cold. By the construction of the constrained order, this implies that all the branches routed before it are cold as well. By the definition of cold branch, these previous branches do not split the bin and thus b_0 can be connected, contradiction.

Case 2: Branch b_0 is hot. By definition of hot branch, this implies that b_0 closes a path P_0 between two points on the bin boundary. Since b_0 can not be routed, its two end points are in two areas which are non connected because of another path P1, formed by previous branches. Those crossing paths p_0, and p_1 can not be of the same component as components which are planar trees. Let C_0, C_1 denote the components of P_0, P_1 respectively. The pair wise detour of C_0, C_1 is infinite as they are not planar and cannot be routed on the same layer. Such components would not be generated by the layer assignment algorithm. Q.E.D.

8. Finding Best Assignment for a Net

The assignment completion and the assignment optimization steps assign on each iteration a two terminal net to its best assignment. This problem of finding a net best assignment is formulated as follows:

(PROBLEM) Given S an assignment state, and n a two terminal net, find an assignment a of n which minimizes $cost(S^a)$, where S^a is state S with net n assigned to a.

Let e denote the existing assignment of n in S, and S^* denote state S with net n unassigned (i.e. free). The cost of S^a can be expressed as the change in the overall cost when n is first unassigned and then assigned to a :

$$cost(S^a) = cost(S) + \Delta(S, S^*) + \Delta(S^*, S^a)$$

where:

$$\Delta(S_1, S_2) = cost(S_2) - cost(S_1)$$

The terms *cost(S)*, and Δ *(S,S*)* are independent of assignment *a*, so *cost(S^a)* is minimized iff Δ *(S*,S^a)* is minimized. By definition of *asgContrib()*, Δ *(S*,S^a)* is exactly *asgContrib(a,n)* and thus the best assignment is the one with minimum wire cost contribution.

(LEMMA) Let B_1, B_2 be two branches in assignment *a* of *n*. Let C_1, C_2 be their component respectively. The components C_1, C_2 are assigned to different layers or they do not intersect, including end points of their edges.

(PROOF) By contradiction, we assume that C_1, C_2 are on the same layer and they do intersect. As each multi terminal net is a planar tree, C_1, C_2 cannot cross each other except for common end point. If C_1, C_2 do intersect, this end point is on the two-terminal net *n*, otherwise the multi terminal net *m* of *n* would has a cycle. Since a component can has only a single branch in any two terminal net, the intersection of C_1, C_2 is by the branches b_1, b_2. This is possible only if b1, b2 are adjacent but since they are on the same layer (as their components), they form together a *single* branch (by the definition of branch). Q.E.D.

The cost contribution of an assignment can be expressed as a sum of the contributions to the via cost of and to the wiring cost of the overall cost:

$$asgContrib(a,n) = wireContrib(a,n) + viaContrib(a,n)$$

As the components are planar, branch cannot intersect with branches of the same component. This implies that the wire contribution of a net is the some of the wire contribution of its branches independently:

$$wireContrib(a,n) = \sum_{b_i} Contrib(b_i,n)$$

And the total contribution of assignment a to the overall cost can be expressed as:

$$asgContrib(n,a) = viaContrib(n,a) + \sum_{b_i} Contrib(b_i,n)$$

In Figure 6 the assignment includes two branches and define a single layer crossing. The contribution of this assignment to the overall cost is a via cost of 1, and the sum of wire contributions of the two branches.

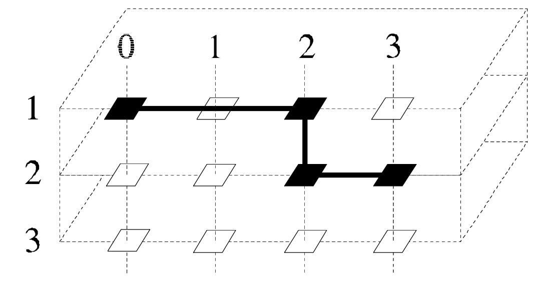

Figure 6 - An assignment of a two terminal net with three layers and 3 sections. Sections [0,1] and [1,2] are assigned to layer 1 and form a single branch. Section [2,3] is assigned to layer 2 and form a branch by itself. The adjacent edges of sections [1,2] and [2,3] form a via at point 2 which crosses a single layer.

This observation enables us to use a shortest path algorithm to find the best assignment of a two terminal net. The search is performed on a graph whose nodes are the grid points as in the example in Figure 6 with addition of a start node *S* and end node *E*. We will first describe a simplified graph which illustrates the concept, and then the complete graph which addresses the problems in the simplified graph. Figure 7 shows the simplified search graph for the example in figure 6.

Let *(i,j)* denote node *j* on layer *i*. Each node *(i,j)* has vertical edges to the same point one layer below it *(i+1, j)* and one layer above its *(i-1, j)*, and has forward horizontal edges to the nodes *(i,k), k>j.* The vertical edges represent layer crossing and have cost of a single layer crossing. The horizontal edges represent branches and their cost is the wire cost contribution of the branch. The start and end nodes *S* and *E* have zero cost edges to the nodes on the two ends of the net.

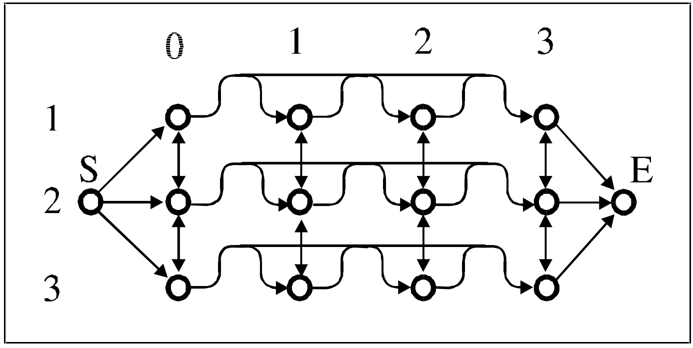

Figure 7 - A simplified search graph for the best assignment of the net in figure 6. Each node has vertical edges which represent layer crossing and forward horizontal edges which represent the possible branches. Nodes S and E are artificial nodes which represent the path begin and end. This graph is oversimplified as it can has two adjacent branches of the same layer. The complete graph is described in the text

The shortest path algorithm is applied on the graph to find the minimum cost, directed path, between *S* and *E*. The cost of a path from *S* to *E* is exactly the cost contribution of that assignment. As every possible assignment can be represented by a path, the path with lowest cost represents the best assignment of that net.

The present.ed simplified graph however, ignores the possible layer crossing at the end points of the two terminal nets. If for example, an already assigned net is already assigned to be connected to point 0 on layer 2, assigning the current two terminal net to be connected to that point on layer 1 adds a layer crossing which is not represented by the edges cost. cost. This is fixed by removing edges from S (E) to layers on which the start (end) point of current net is not connected yet by previously assigned nets. When the point is not connected at all on any layer, no edge is removed as connecting it to any layer will not add a via to any existing connections.

Another inaccuracy of the simplified graph is that it enables paths from S to E which does not represent valid assignment. By definition of branch, adjacent branches of the same two terminal net can not be assigned to the same layer or they will be considered as a single branch. This constrain is not enforced by the graph. For example, the path S, (1,0), (1,1), (2,1), (1,1), (1,3), E has two adjacent branches on layer 1 (from 0 to 1 and from 1 to 3) and the sum cost of these two branches can be lower then the cost of the combined branch from 0 to 3.

Source		Destination		Cost
Node	Qualifier	Node	Qualifier	
S		(l,0,U)	Start(l)	0
		(l,0,D)	Start(l)	0
		(l,0,F)	Start(l)	0
(l,p,U)	l∈[2..m], p∈[0..n]	(l-1,p,U)		via cost
	l∈[2..m], p∈[0..n]	(l-1,p,F)		via cost
	l∈[1..m], p=n, End(i)	E		0
(l,p,D)	l∈[1..m-1], p∈[0..n]	(l+1,p,D))		via cost
	l∈[1..m-1], p∈[0..n]	(l+1,p,F)		via cost
	l∈[1..m], p=n, End(i)	E		0
(l,p,F)	l∈[1..m], p∈[0..n-1]	(l,k,U)	k∈[p+1..n]	wireContrib(l,[p..k])
	l∈[1..m], p∈[0..n-1]	(l,k,D)	k∈[p+1..n]	wireContrib(l,[p..k])
	l∈[1..m], p=n, End(i)	E		0

Table 1 - The edges of the complete search graph for finding a two terminal net best assignment *(l,p,d)* denotes node on layer l, point *p*, and direction *d*. *wireContrib(l,[j..k])* is the wire cost contribution of the branch between points *j* and *k* on layer *l*. *Start(l)* is true iff point *0*, the start point of the net, is already connected on layer *l* by previous nets or if it is not connected at all on any layer.

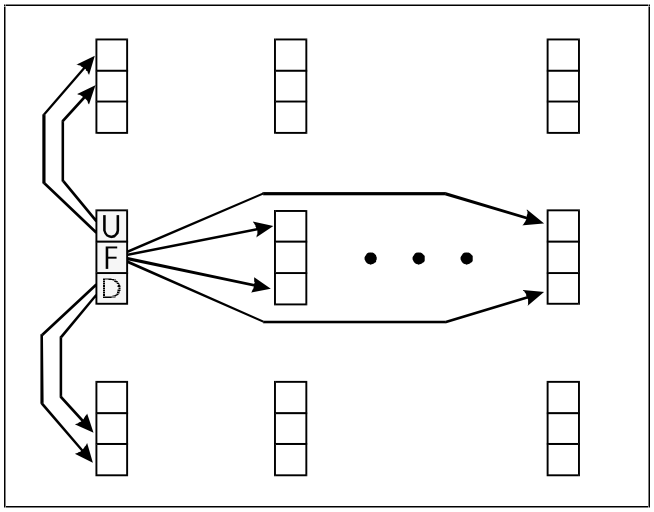

Figure 8 - A typical node (shadowed) in the complete search graph. The *up* and *down* subnodes have outgoing vertical edges which represent vias. The forward sub node has horizontal edges which represent branches. Splitting each node to three subnodes eliminate the invalid paths in which two adjacent branches are on the same layer. Table 7 has the formal description of the graph.

To avoid these illegal paths, the graph is modified by splitting each node *(l,j)* into three nodes *(l,j,d)* where *d* is one of the three directions *up*, *down*, and *forward*. The direction of each of the three sub nodes represents a valid direction in which a sub path which ends at this node can be continued. The complete graph and the cost of its edges are summarized in table 1.

9. Component Pair-Wise Detour

The wire cost term of the cost function is defined as a sum of estimated detour of pair of component. This detour of two components *a,b* is define as the minimum of the estimated detour when *a* is routed before *b* and when *b* is routed before *a*, independently of any other component. This introduces the ordered pair wise detour problem which is formulated as follows:

(PROBLEM) Let *a, b* denote two components. Find the extra length (detour) of *b* when it is routed after *a* compared to its length when it is routed separately.

The length of *b* when routed separately is merely the sum of the lengths of its edges, as it is planar trees. The length of *b* when routed after *a* is the sum of the length of its edges b_i when they routed severalty after a:

$$length(b|a) = \sum_{b_i} length(b_i|a)$$

This observation reduces the detour problem to the basic detour problem, formulated as follows:

(PROBLEM) For a given finite planar tree *T* on a plan and two points *u, v,* find the length of the shortest rubber band path between *u* and *v* which does not crosses *T*.

(DEF) Let *v* be a node of a planar, finite tree *T*. The tree edges adjacent to *v* splits the infinitely small neighborhood of *v* into non connected area called *regions*.

The number of region of a node v is *max(deg(v)*,1*)* where *deg(v)* is the number of edges adjacent to v. Figure 9 shows an instance of the basic detour problem. The regions of the tree are marked with squares.

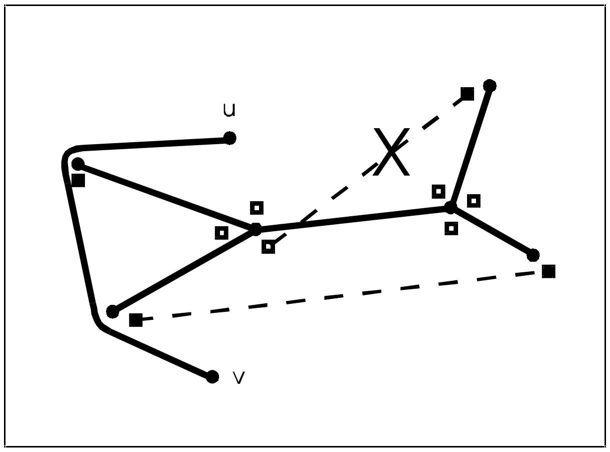

Figure 9 - The basic detour problem. The branch *u,v* is routed after the tree and thus may have a detour. The actual length of the branch can be find by formulated the problem as a shortest path problem in a graph of regions (marked as squares). Regions of angle smaller than π (hollow squares) can not be on the path so they can be ignored. The broken line shows a non visible region pair (marked with *X*) and a visible pair.

(DEF) Let, *a,b* be two points on the plane. We say that *a,b* are *visible* to each other if the straight line between them does not intersect with the tree *T*.

(DEF) Let *a (b)* denote a region of node *k (l)* of a tree *T*,. We say that *a* and *b* are *visible* if for every arbitrary small $\varepsilon > 0$, *a (b)* has point *x*, *(y)* such that $|x-k| < \varepsilon$, $(|y-l| < \varepsilon)$, and *x*, and *y* are visible to each other.

The rubber band path between the branch ends *u,v* is computed by finding the shortest path in the regions graph *G=(V,E)*. The vertices of the graph are $E=R\cup\{region(u),region(v)\}$ where R is the set of all regions of T. Two vertices share a common edge iff they are visible to each other. It can be shown that a shortest path from *u,v* defines a valid rubber band between *u,v* and that every rubber band path between *u,v* has an equivalent path in *G*. This implies that a shortest path of on the graph is also a shortest rubber band path.

The size of the graph can reduced by removing vertices and still maintaining the exact set of solutions. A region whose angle is smaller than π can not be on the shortest path as the path can be shorten by relaxing the rubber band at this point. Any region whose its angle is exactly π can be removed (figure 10-b) as its two adjacent collinear regions are already visible. These vertex elimination leaves up to a single vertex for each tree node.

The density of the graph can be reduced as well by eliminating visibility edges. In Figure 10-a the edge between the region of nodes *v,u* can be removed as *u* is in the shaded area of *v* and thus an edge between them can not be part of any valid rubber band path.

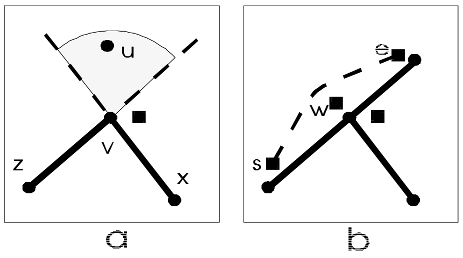

Figure 10 - Two examples of region graph reduction. As node u is in the shaded area (a) of the region of v, they can not be adjacent on a rubber band path even though they are visible. The collinear region in (b) can be removed as its two adjacent regions are already visible.

10. Using Incremental Calculation

The run time of the algorithm can be improved by using incremental calculation. The implementation of the layer assignment algorithm in Surf system keeps computation results from the previous assignment iteration and whenever a new net is assigned, it updates only the required values. This cached information includes, among other values, the intersection dependency between components, and the best assignment of each net.

11. Cost Driven Routing

The presented layer assignment algorithm uses a cost function which directs it to more desire solutions. This approach enables to control the layer assignment and the final routing by merely modifying the cost function. An example of such modified cost is the cost driven one and an half layer routing which has been incorporated into the Surf system.

The use of MCM in the mass production of consumer products can reduces the overall cost by eliminating time consuming production steps and reducing the number of discrete components. This however requires to keep the production cost of the MCM as low as possible. One of the major factors of an MCM cost is the number of layers, and thus it should be as low as possible. A practical approach of reducing the number of layers is to use a single interconnect layer and having short jumpers inside small islands in the ground layer. By having this islands small enough, the effectively of the ground layer is maintained and a second signal wiring layer is eliminated.

This routing style has been incorporated into Surf by extending the cost function. The modified cost function has an new term which increase the cost as function of the size and number of components on the restricted layer:

$$cost'(S) = cost(S) + \sum_{C} diameter(c)^{K}$$

Where *diameter(c)* is the maximal distance between two nodes of the component *C* and *k*≥ *1* is a user controlled constant.

12. Summary Conclusion

The flexibility of a rubber band representation of MCM interconnect enables the use of new algorithm which provide better control over the final layout. One of this algorithm is the layer assignment algorithm presented in this paper. The algorithm uses cost function and a combinatorial optimization algorithm to partition the multi-terminal net, multi-layer problem into a set of two terminal net, single layer sub-problems. The algorithm scales well with growing number of layers, and supports rectilinear, octilinear and any- angle routing modes. A unique property of the algorithm is that it does not merely check for net crossing, but consider the *amount* of conflict. By controlling the cost function parameters, a desire balance between wire length and number of vias is achieved. A simple modification to the cost function enable different layout styles such as one and an half layer routing which reduces the number of layers in cost sensitive MCM designs.

13. Bibliographic

[Chew89] Chew, L.P.; Constrained Delaunay Triangulation; Algorithmica, vol 4, pp 97-108, 1989

[STAPA92] Staepelaere J. D.; Geometric Transformation for a Rubber Band Sketch; Master Thesis, Univ. of California, Santa Cruz; 1992.

[Dai90] Dai, Wayne Wei-Ming and Kong, Raymond and Jue, Jeffrey and Sato, Masao; Rubber Band Routing and Dynamic Data Representation; Digest of Tech. Papers of IEEE International Conf. on Computer Aided Design, pp 52-55, 1990

[Dai91a] Dai, Wayne Wei-Ming; Performance Driven Layout of Thin-film Substrates for Multichip Modules; Extended abstract volume of 1991 Multichip Module Workshop, pp 114-121, 1991

[Dayan91] Dayan T. Dai, Wayne Wei-Ming; Force-Driven Constrained Wiring Optimization; Technical Report UCSC-CRL-91-40; CMPE, Univ. of California at Santa Cruz, 1991

[DDS91] Dai, Wayne Wei-Ming and Dayan, Tal and Staepelaere, David; Topological Routing in Surf: Generating a Rubber-Band Sketch; Proceeding of 28th Design Automation Conf; pp 39-44; 1991

[DKS91] Dai, Wayne Wei-Ming and Kong, Raymond and Sato, Masao; Routability of a Rubber-Band Sketch; Proceeding of 28th Design Automation Conf., pp 45-48, 1991

[Hanafusa90] Hanafusa, A., Y. Yamashita, and M. Yasuda, Three-Dimensional Routing for Multilayer Ceramic Printed Circuit Boards, Proc IEEE Int'l Conf. on Computer-Aided Design, pp. 386-389, Nov 1990.

[KG89] J. Mark Keil and Carl A. Gutwin; The Delaunay Triangulation Closly Approximates the Complete Euclidean Graph; Springer-Verlag Lecture Notes on Computer Science, vol 382, 1989, pp 47-56

[Lauther87] Lauther, Ulrich; Top Down Hierarchical Routing for Channelless Gate Arrays Based on Linear Assignment; VLSI 87: VLSI Design of Digital Systems, 1987

[Lu91] Lu, Yizhi and Dai, Wayne Wei-Ming; A Numerically Stable Algorithm for Constructing Constrained Delaunay Triangulation and Application to Multichip Module Layout; Proceeding of China 1991 International Conference on Circuits and Systems; pp 644-647; 1991

[Maley85] Leiserson, C.E. and Maley, F.M.; Algorithms for routing and testing routability of planar {VLSI} layouts}; Proceedings of the 17th Annual {ACM} Symposium on Theory of Computing, pp 69-78, 1985

[Marek86] Marek-Sadowska, Malgorzata; Route Planner for Custom Chip Design Digest of Tech. Papers of IEEE; International Conf. Computer Aided Design, pp 246-249, 1986

[Miracky91] Miracky, R. et al, Technology for Rapid Prototyping of Multi-Chip Modules, IEE Int'l COnf. on COmputer Design, pp. 588-591, 1991.