A MULTI-LAYER ROUTING METHODOLOGY USING A BOOLEAN SATISFIABILITY BASED ROUTER

by

Yunjian Jiang, Sunil P. Khatri, Robert K. Brayton
and Alberto Sangiovanni-Vincentelli

Memorandum No. UCB/ERL M99/16

19 March 1999

COVER

A MULTI-LAYER ROUTING METHODOLOGY USING A BOOLEAN SATISFIABILITY BASED ROUTER

by

Yunjian Jiang, Sunil P. Khatri, Alberto Sangiovanni-Vincentelli
And Robert K. Brayton

Memorandum No. UCB/ERL M99/16

19 March 1999

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

A Multi-layer Area Routing Methodology using a Boolean Satisfiability based Router

Yunjian Jiang Sunil P. Khatri Alberto Sangiovanni-Vincentelli
Robert K. Brayton

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley

March 19, 1999

Abstract

We propose an area routing technique which is suitable for deep sub-micron applications. In its most general form, the routing problem can be viewed as finding wiring connections between electrically equivalent pins, utilizing one or more layers of metal, while avoiding obstacles present possibly on every layer of metal. This general formulation (called *area routing*) has been traditionally used for routing printed circuit boards (PCBs) and multi-chip modules (MCMs). This formulation is general enough to encompass routing in integrated circuits as well.

However, more specialized forms of routers have traditionally been used for ICs. We argue why the traditional IC routing methodology of first performing a global routing step, then utilizing switch-box or channel routers, is inadequate in this new scenario. Instead of performing a global-then-detailed routing approach, we choose a routing technique that partitions all the nets into smaller clusters, and then routes all the nets in a cluster using a Boolean satisfiability (SAT) based search strategy.

In our scheme, we first assign each net to a routing layer pair. The set of nets on each layer pair is now partitioned into several clusters such that the overlap between nets in different clusters is minimized. All the nets in any cluster are routed simultaneously, and each net cluster is routed independently. The ordering of net clusters is done using a simple heuristic.

Within a cluster, all nets are routed simultaneously using SATRE, our SAT based routing engine. Routes for a net can be chosen from predetermined set of routing patterns, which utilize 2, 3 or 4 vias. We assign Boolean variables to represent all the possible routes that the net can use, and create SAT clauses to represent the routing constraints for each net. This step requires only $O(log_2(N))$ Boolean variables, where N is the size of the routing problem. The transformation of the area routing problem into an instance of Boolean Satisfiability (SAT) is thus performed very efficiently. Next, we perform a binary search to find a satisfying solution of the SAT instance, using *non-chronological backtrack* to give rise to an efficient search procedure.

The novel feature of this scheme is that it eliminates the global-then-detailed routing paradigm used in earlier generations of routers. Rather, it proposes a new routing methodology, where the set of all nets is partitioned into net clusters, followed by the *simultaneous* routing of all the nets in a single cluster. Further, the scheme is easily adapted to a situation where several metal layers are available for routing.

In our experiments, we assume that 4 metal layers are available for routing. We show that our router is able to obtain extremely dense routes with reasonable run-times. The high quality of our results is evidenced by the fact that over all the examples we tried, the total net length was on average 1.47% larger than the sum of Manhattan length for all the nets.

1 Introduction

The routing problem in VLSI physical design has been shown to be NP-complete. The traditional approach divides the problem into two phases, global and detailed routing [1]. Different strategies and algorithms are applied in each phase.

Until a few generations ago, ICs typically had two layers of metal along with polysilicon available to realize the routing. Since the functional blocks also used these metal layers to realize their internal wiring, *over-the-cell* routing was not feasible. Thus explicit routing regions needed to be defined. This, coupled with a row-based layout methodology gave rise to a natural partition of the routing space into channels and switch-boxes. Therefore, specialized detailed routers, such as *channel* and *switch-box routers* worked well in this scenario. With the availability of three or four layers of interconnect the situation changed only marginally.

However, with the advancement of processing technology, it is predicted that fabrication processes will have as many as nine layers of metal within the next seven years [2]. As a result, a routing methodology which can intelligently handle this extra flexibility will be required. Such a methodology should carefully budget the available routing resources on different metal layers.

Given the large number of metal layers available for routing, explicit routing regions are not required to be defined for routing. The prevalent industry practice for high end designs is to abut rows of cells and use an area router to perform the interconnections. According to [3], channels are no longer accurate models for routing resources in such technologies and area routing is a robust and general approach that can be used. In these respects, the IC routing problem bears a striking similarity to the MCM and PCB routing problems, domains where area routing techniques have been very successful.

The rapidly increasing size and complexity of designs in the DSM era requires major modifications to CAD algorithms and data structures, and in some cases, suggests altogether new CAD approaches and methodologies. In particular, this is true for the routing problem as well. [4] details how the changes in processing technology affect the routing problem, and surveys routing techniques that are likely to be useful in the DSM era.

In this work, we introduce a new formulation of the VLSI area routing problem, which transforms the routing problem into an instance of the SAT problem. In our formulation, global and detailed routing phases are combined in the same framework. The routing area, which can be of any size and shape, is divided into grids to represent the routing resources. The inputs to the area routing problem are the set of nets to be routed, the number of available metal layers and the description of blockages on various metal layers. Examples of blockages can be placed modules, pre-routed nets or any region that nets are not allowed to pass. The output of the area router is a legal route that respects the above routing constraints.

Based on some experiments we performed, we determined that the majority of nets have a Manhattan length that is about 1/6 to 1/4 times N, the size of the routing problem[1]. As a result, in our approach, we partition nets into clusters such that there is minimal interference between clusters. These clusters are be routed independently, without adversely affecting the overall routing quality. The route for each net is confined to remain within a *bounding rectangle* which is slightly larger than the bounding box of the net. As a result, nets have sufficient routing flexibility and at the same time, the problem size is kept small.

Initially, nets are assigned the metal layer pair on which they will be routed. Based on their Manhattan routing length, nets are partitioned into *local*, *semi-global* and *global* nets. Local nets are assigned to the lowest layer pair and global nets are assigned to the highest layer pair. Semi-global nets are partitioned into n parts, such that pairs of nets with a large degree of *overlap*[2] are placed in different parts. Each of the resulting parts are routed on different metal layer pairs. In our current implementation we chose $n = 2$, since we use 4 layers of metal for routing (i.e. we have 2 metal layer pairs for the routing).

[1] The size of the routing problem is defined as the number of grid points in the semi-perimeter of the routing area.

[2] The overlap of a pair of nets refers to the common area between the bounding boxes of the nets.

After this, the nets on each layer pair are grouped into k clusters, such that the overlap between nets in different clusters is minimized. Nets that have a large amount of overlap are assigned to the same cluster. Clusters are then ordered based on a heuristic that measures their relative routing difficulty.

Each cluster is routed using SATRE, our SAT based routing engine, which routes all the nets in the cluster simultaneously. The routes that are created for the i^{th} cluster are considered as blockages while routing subsequent clusters. The routing problem is transformed into an instance of the SAT problem by assigning Boolean variables to represent the possible valid routes, and creating SAT clauses to represent the various routing constraints. This transformation is extremely efficient, and requires $O(log_2(N))$ variables, where N is the size of the routing problem. The resulting SAT problem is now solved using an efficient SAT solver which utilizes the concept of non-chronological backtracks [5]. Each cluster is routed using either of 2, 3 or 4 via patterns. If the cluster cannot be routed using 2-via patterns, then 3-via patterns are used. If the cluster cannot be routed using 3-via patterns, then 4-via patterns are used.

In the subsequent sections, we elaborate on and substantiate our claims. Section 2 describes previous research in SAT-based routing methodologies. Section 3 describes our routing methodology in detail. Section 4 describes preliminary results that we have obtained using our routing methodology. Finally, in Section 5, we make concluding comments and discuss further work that needs to be done in this area.

2 Previous Work

Even though Boolean Satisfiability (SAT) is shown to be NP-complete, a host of efficient techniques exist for solving large problem instances [5] [6] [7]. Some attempts [8], [9] have been made to transform the routing problem into a SAT problem. Boolean variables are created to represent the routing resources and routing constraints are represented in the form of Boolean functions. Each assignment of the Boolean variables that satisfies the SAT problem corresponds to a complete routing solution. Thus the routing of all nets is performed simultaneously.

In [8], Devadas first showed the feasibility of using SAT to optimally solve physical layout problems. He formulated various NP-complete layout problems, namely two and multi-layer channel routing, two-way partitioning and one-dimensional and two-dimensional placement, as SAT problems. The size of problems that can be solved using this formulation is limited and the routing formulation is restricted to the channel routing structure.

Wood and Rutenbar [9] extended this idea to FPGA routing, where they assign variables to the routing fabrics within the routing channels in island-style FPGA's. A global router is invoked to decompose the FPGA routing problem into small problems for the SAT router. The focus of this work is restricted to routing in the FPGA context.

A recent approach to SAT-based routing, also in the context of FPGA routing, is described in [10]. In this approach, the formulation of the problem is similar to [9]. The main feature of this work is that an efficient SAT solver [5] is used as the solution engine.

An efficient *pattern based* routing strategy for MCMs was introduced in [11]. This approach performs routes using a series of patterns with up to 4 vias. Routing is performed two layers at a time. Each layer pair is processed column-by-column, and unroutable nets are routed on subsequent layer pairs. Even though this work utilizes a pattern based routing strategy with up to 4 vias, it is different from our approach in that it processes nets column-by-column, using a left-to-right sweep of the routing area. Our approach, on the other hand, casts the routing problem as a SAT problem, and routes all nets simultaneously using an efficient SAT solver.

A modification of [11] was introduced in [12]. In this greedy router, the scan of the routing area is performed in two directions simultaneously. The router of [12] is also designed for MCM routing.

3 Our Approach

The first step in our routing methodology is to assign a metal layer pair for the routing of each net. Initially, nets are partitioned into three *groups* based on their Manhattan length. These groups correspond to *global* nets, *semi-global*

nets and *local* nets. Global nets are assigned to the highest metal layer pair, while local nets are assigned to the lowest metal layer pair for routing. Semi-global nets are partitioned into n parts, and each of the resulting parts are routed on a separate layer pair. In our implementation we chose $n = 2$.

The nets corresponding to each metal layer pair are further partitioned into smaller net clusters. All nets in a net cluster are routed simultaneously. The clustering of each group is performed using *METIS* [13], a k-way partitioning tool.

Each net cluster is routed using one of several pre-defined patterns. Initially 2-via patterns are tried, and if this does not work, 3-via patterns are tried, followed by 4-via patterns. For each net, we first determine the bounding rectangle in which we constrain the routes for that net to reside. After this bounding rectangle is computed for each net, we generate the SAT routing clauses for the net. At this point, the SAT based routing algorithm is called to simultaneously route all the nets in a cluster. Each net is assigned routing resources, and when this assignment is performed for a given net, an efficient checking routine is called to determine if this assignment of routing resources conflicts with the resources used by a previously routed net (or blockage). If a conflict is found, we perform a back-track operation, otherwise we bind the routing resources to the current net and update the routing resource data structure.

Our routing scheme assumes that each net has exactly two terminals. General n-terminal nets can be decomposed into a series of two terminal nets using an algorithm described in [14].

The remainder of this section details each step of our routing methodology. Section 3.1 describes the algorithm for constructing the bounding rectangle for each net. Section 3.2 describes the patterns that our router can utilize for each net. The transformation of the routing problem into a Boolean satisfiability problem is described in Section 3.3. The SAT solution strategy is detailed in Section 3.4. Finally, the construction of net groups and clusters is described in Section 3.5.

3.1 Construction of Bounding Rectangles for Nets

Each net is assigned a *bounding rectangle* in which its routing must be completed. Consider a net i, with source vertex $s \equiv (x_s, y_s)$, and target vertex $t \equiv (x_t, y_t)$. Then, the width of the bounding rectangle is given by:

$$W = |x_t - x_s| + \alpha \cdot |y_t - y_s| + \beta \tag{1}$$

Similarly, the height of the bounding rectangle is given by:

$$H = |y_t - y_s| + \alpha \cdot |x_t - x_s| + \beta \tag{2}$$

After some experiments, we determined that $\alpha = 0.15$ and $\beta = 5$ were good choices of parameters for a wide variety of applications. For extremely dense routes, larger values of these parameters can be chosen.

The *bounding box* of a net is the smallest rectangle that encloses the source and target vertices of that net. Figure 3 shows the relationship between the bounding box and the bounding rectangle of a sample net.

3.2 Routing Patterns

Our router searches several patterns while performing the routing of any net. These patterns fall into three distinct classes, described below.

3.2.1 2-Via Patterns

The simplest routing patterns allowed are 2-via pattern. In general, for a given net, there can be 4 such patterns.

These patterns are described in Figure 1. These patterns are represented by a wire trace emanating from the source vertex (s), in either the north (Figure 1a), south (Figure 1b), east (Figure 1c) or west (Figure 1d) directions. After this

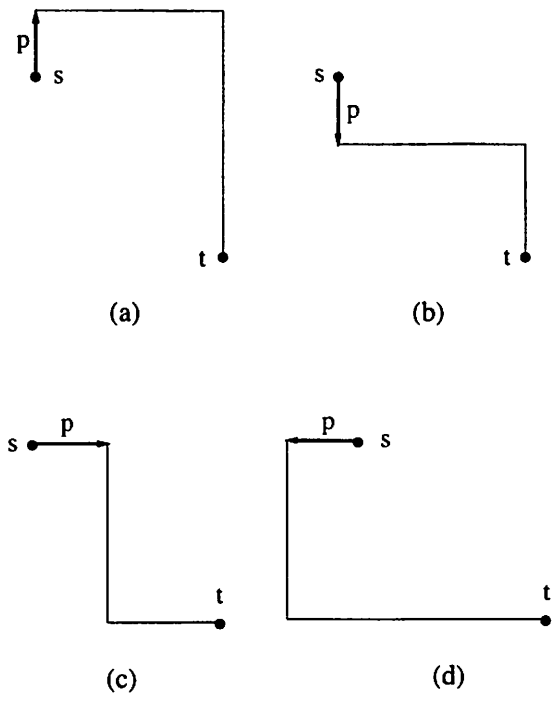


Figure 1: 2-Via Routing Patterns

wire trace has traversed a distance p in any direction, the route is completed using an L-shaped segment ending at the target vertex (t).

For ease of notation, we henceforth refer to these patterns respectively as *2N*, *2S*, *2E* and *2W* patterns. In this notation, the "2" represents the fact that the pattern is a 2-via pattern, while the letter following it represents the initial direction of the wire trace emanating from the source vertex.

For a given net, each pattern type can result in a large number of routing choices, since the value of the distance p is variable.

3.2.2 3-Via Patterns

3-via patterns can be represented by a wire trace emanating from the source vertex, in any of the cardinal directions, and traversing a distance p_1 in this direction. After this, the trace switches to any of the two orthogonal directions and traverses a distance p_2 in that direction. Finally, the route is completed using a L-shaped segment ending at the target vertex.

For a given net, there can be 8 such 3-via patterns. Using the notation introduced earlier, we label these patterns as 3NE, 3NW, 3SE, 3SW, 3EN, 3ES, 3WN and 3WS. These patterns are shown in Figure 2.

3.2.3 4-Via Patterns

4-via patterns are represented by a wire trace emanating from the source vertex in any of the cardinal directions, and traversing a distance p_1 from the source vertex. At this point, the trace switches to one of the two orthogonal directions, for a traversed distance of p_2. Next, the trace proceeds in an orthogonal direction for a distance of p_3, following which the route is completed using a L-shaped segment which ends at the target vertex.

For a given net, 16 such 4-via patterns are possible. These patterns can be represented as 4NES, 4NEN, 4NWS, 4NWN, 4SES, 4SEN, 4SWS, 4SWN, 4ENE, 4ENW, 4ESE, 4ESW, 4WNE, 4WNW, 4WSE and 4WSW.

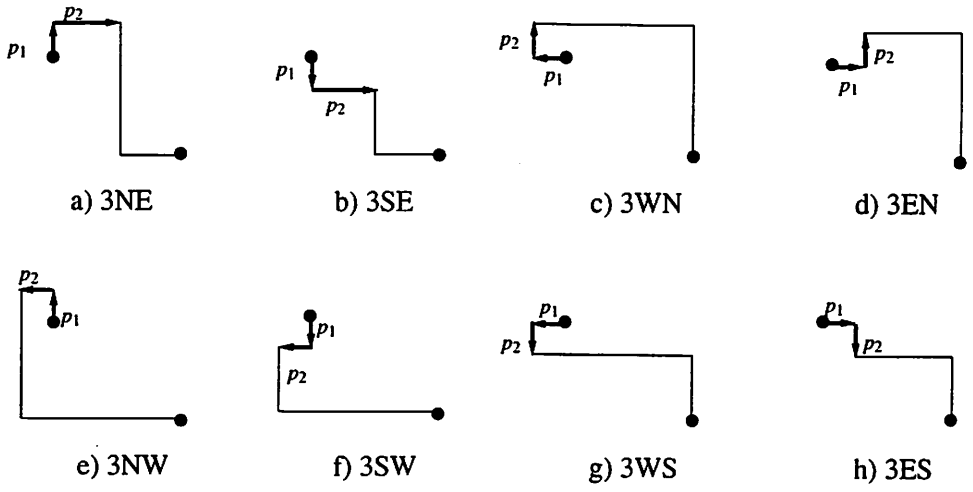


Figure 2: 3-Via Routing Patterns

3.3 SAT-based Formulation of Area Routing

In this section, we will describe the SAT based formulation of the area routing problem using the example of 2-via patterns. The formulation for 3-via and 4-via patterns are analogous, and are not separately described.

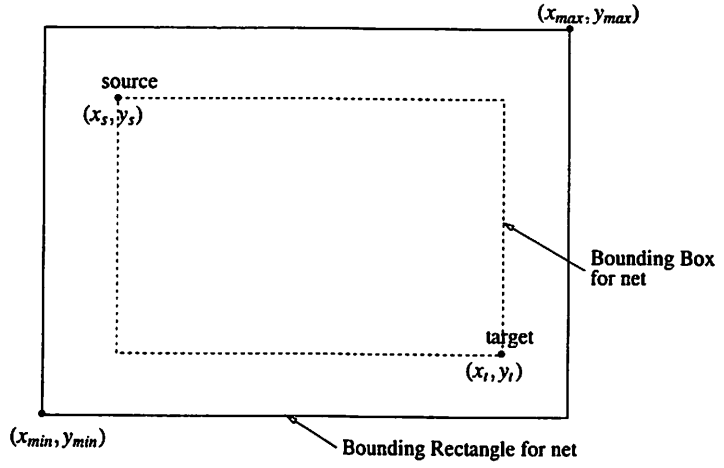


Figure 3: Bounding Clauses for a Sample Net

Consider a net, with a bounding rectangle as shown in Figure 3. For each such net, we assign $n+2$ Boolean variables, where $n = log_2(max(H,V))$. Here $H = x_{max} - x_{min}$, and $V = y_{max} - y_{min}$. The n Boolean variables are used to encode the parameter p referred to in Section 3.2.1. The additional 2 Boolean variables are used to encode the direction of the initial wire trace emanating from the source vertex. We refer to the direction variables as $d \equiv (d_0, d_1)$, and the parameter variables as $p \equiv (p_0, p_1, \cdots, p_{n-1})$. Since the total number of nets that our SAT based router handles simultaneously is bounded, the total number of Boolean variables in the problem is $O(log_2(N))$, where N is the size of the routing problem.

Now the routing problem is transformed into an instance of the Boolean satisfiability (SAT) problem. The SAT formula is created in *Conjunctive Normal Form* (CNF). The SAT formula consists of several *clauses*, each of which is a sum of several *literals*. A literal corresponds to a Boolean variable of a net (i.e. d_i or p_i), or its complement. Clauses are used to encode the routing constraints that the problem imposes. There are several types of clauses, which are as

follows:

3.3.1 Bounding Clauses

These clauses are used to ensure that a net is routed within its bounding rectangle. Consider the net shown in Figure 3. In this example, if the direction of the initial wire trace from the source is N, then the parameter p is constrained to be less than $y_{max} - s_y$. This constraint can be represented as the Boolean expression

$$d \equiv N \Rightarrow p < (y_{max} - y_s) \tag{3}$$

Similarly, other bounding clauses can be written as well, for the cases in which the initial wire trace emanates in other directions. These clauses are:

$$d \equiv S \Rightarrow p < (y_s - y_{min}) \tag{4}$$

$$d \equiv E \Rightarrow p < (x_{max} - x_s) \tag{5}$$

$$d \equiv W \Rightarrow p < (x_s - x_{min}) \tag{6}$$

3.3.2 Equivalence Clauses

In the routing scenario described above, whenever $p = 0$, then directions N and S are equivalent. Likewise, directions E and W are equivalent too. The clauses written to represent this condition are given below.

$$(p = 0) \Rightarrow (d = E) + (d = S) \tag{7}$$

3.3.3 Direction Clauses

These clauses are used exclusively for 3-via and 4-via nets. These clauses restrict the allowable directions that a route can choose after it has traversed a distance p_1 in the initial direction. Instead of allowing the route a choice of two directions, only one direction is allowed. This is the direction that allows the route to proceed towards the target vertex. The other direction (which moves the route away from the target) is not allowed. The justification of these clauses is that they lower the Manhattan length of the routes.

In the case of 4-via nets, this directional restriction is enforced after the route performs its second turn as well (after the route traverses a distance p_2).

In our implementation, we have a command-line option which can disable the use of these direction clauses. This is useful for particularly difficult examples.

3.4 SAT Search Strategy

Once the SAT problem is created, we then search it for a solution. Our SAT based router can route several nets (referred to as a *net cluster*) simultaneously.

Among the nets in a cluster, those nets with the smallest Manhattan route length are routed first. The rationale for this choice is that nets with smaller Manhattan lengths have lower routing flexibility, and hence it would be better to route these nets first. Nets with larger Manhattan route lengths have a larger routing flexibility on account of their larger bounding rectangles, hence they are routed later.

Initially, the routing of a net cluster is attempted using 2-via patterns. If the routing for any net cluster fails at this stage, 3-via patterns are tried, and if this fails as well, 4-via patterns are tried.

While routing a single net, we favor choices of d and p which allow the route to be completed with a route length equal to the Manhattan distance between the source and target vertices. If no such route is possible, then we search values of d and p which result in a route with length greater than the Manhattan route length.

While searching for a route for a given net, the variables (d_k and p_k) corresponding to net are successively bound. If no route is possible for the i^{th} net in a net cluster, then the last net j which conflicts with the routes for i is recorded. When a backtrack needs to be performed, we can directly jump to the j^{th} net, skipping all nets in between i and j. This idea is similar to the concept of *non-chronological backtrack* which is employed in the *GRASP* SAT solver.

The search strategy is shown graphically in Figure 4. In this figure, the SAT search trees for the routing of net_i and net_k are shown. Nodes in these search trees which are crossed out with a dotted line represent routes that would cause a conflict. While routing net_k, for example, both branches following the node labeled d_1 result in conflicts. For the left branch, assume that the earliest net that causes the conflict is net_2. For the right branch, assume that the earliest net causing the conflict is net_i. Since both branches following the node labeled d_1 cause a conflict, a backtrack is required. Assuming i ¿ 2, the we can directly backtrack to the net_i, rather than the net immediately preceding net_k.

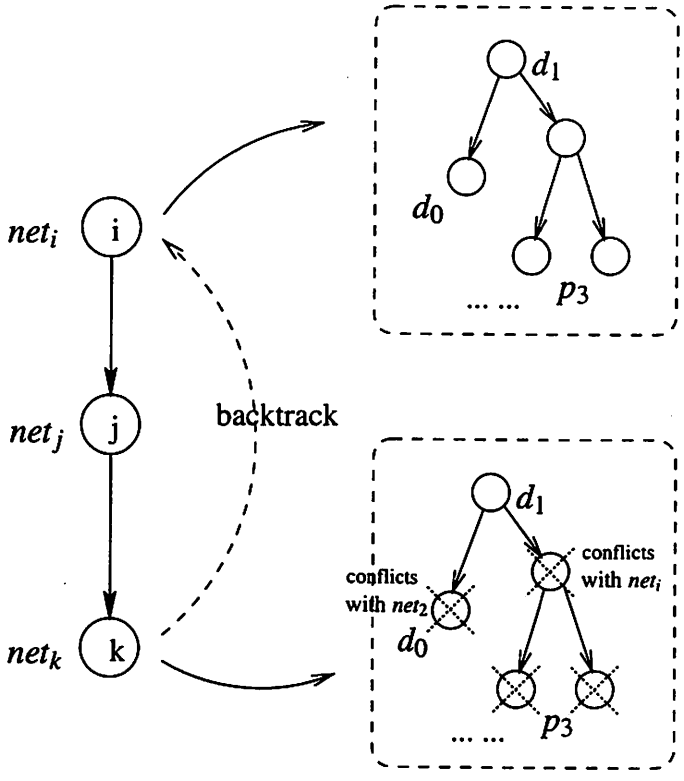


Figure 4: Example of SAT Search Strategy

When variables of a particular net are bound, a separate *conflict checking routine* determines if the routing resources corresponding to the current choice of variables conflicts with routing resources that have already been assigned to previously routed nets. The current route is decomposed into a series of net segments. For each such net segment, we check if the corresponding metal segment has been utilized by prior nets or blockages. This is done by an efficient sorted list-based algorithm.

As the solution proceeds, the route for net k is considered as a blockage for subsequent routing problems that are invoked. As a result, we do not completely eliminate the ordering dependency among nets. However, this ordering dependency is reduced, due to our ability to route several nets at once.

3.5 Partitioning and Clustering Strategy

3.5.1 Partitioning Nets into Metal Layer Pairs

Before the SAT router is invoked, the nets in the routing problem are grouped into three groups based on their Manhattan length. These groups correspond to *global* nets, *semi-global* nets, and *local* nets.

In our current implementation of the router, we assume 4 metal layers are available to perform the routes. As a result, each of the nets in the three groups needs to be routed using either the Metal1-Metal2 layer pair, or the Metal3-Metal4 layer pair. All local nets are assigned to the Metal1-Metal2 layer pair. Similarly, global nets are assigned to the Metal3-Metal4 layer pair.

Semi-global nets are partitioned into two groups, using METIS [13], a partitioning tool from the University of Minnesota. The first such group is routed using the Metal1-Metal2 layer pair, while the second group is routed using the Metal3-Metal4 layer pair. In order to partition semi-global nets, the routing problem is first transformed into a Net Non-overlap Graph, $G(V,E)$, such that vertex v_i of G corresponds to the net n_i, and the edges $e_{i,j}$ between vertices i and j represent the inverse relative overlap between the nets n_i and n_j. Assuming the bounding box area of net n_i is A_i, and that of net n_j is A_j, and the common area between their bounding boxes is A_{ij}, then

$$e_{i,j} = \frac{(A_i) \cdot (A_j)}{A_{i,j}} \tag{8}$$

If there is no overlap between the bounding boxes of nets n_i and n_j, then an edge weight of

$$e_{i,j} = (A_i) \cdot (A_j) \tag{9}$$

is inserted between the vertices i and j. As a result, the Net Non-overlap Graph is a complete graph.

This choice of edge weight ensures that the mincut-based partitioning tool will place interfering nets in different layer pairs, thus ensuring their routability.

3.5.2 Constructing Net Clusters

After each net has been assigned a metal layer pair, we then further partition the nets in each metal layer pair into several net clusters. Based on our experiments, if the net cluster size is less than 25, the SAT router is able to efficiently find the routing solution. As a result, we partition the nets of each metal layer into clusters whose size is approximately 25. This partitioning is done using METIS as well. The nets corresponding to each metal layer pair are transformed into a Net Overlap Graph $G^*(V,E)$, such that vertex v_i of G^* corresponds to the net n_i, and the edges $e_{i,j}$ between vertices i and j represent the relative overlap between the nets n_i and n_j. Assuming that the area of the bounding box of net n_i is A_i, and that of net n_j is A_j, and the common area between their bounding boxes is A_{ij}, then

$$e_{i,j} = \frac{A_{i,j}}{(A_i) \cdot (A_j)} \tag{10}$$

The construction of the Net Overlap Graph is graphically shown in Figure 5

Choosing edge weights in this fashion ensures that nets with large values of overlap are grouped together within a cluster. Also, nets in different clusters have minimal overlap. As a result, the routing order dependency between net clusters is minimized.

3.5.3 Ordering Net Clusters

Once all the net clusters have been determined, we perform the routing of each cluster using the SAT based router. For each cluster, we find the *cluster bounding rectangle*, which is defined as the smallest rectangle that covers the bounding rectangles of all the nets in that cluster. Clusters whose cluster bounding rectangles overlap most with those of other clusters are routed first.

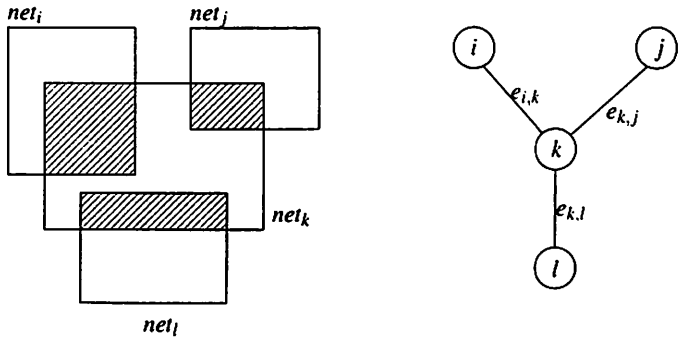


Figure 5: Construction of Net Overlap Graph

4 Layout Experimental Results

To validate our methodology, we perform the routing of several examples using 4 layers of metal. The first step is a partitioning step. Nets are ordered based on the minimum Manhattan length of their routes. Among these, the local nets (smallest 20% of the nets) are assigned to the first metal layer pair (metal layers 1 and 2). The global nets (largest 20% of the nets) are assigned to the second metal layer pair (metal layers 3 and 4). The remaining semi-global nets are partitioned using METIS [13], as described in Section 3.5.1. After this step, each net is assigned a metal layer pair on which it will be routed.

Next, METIS is used to cluster the set of nets on each metal layer pair such that nets in each cluster have a high degree of overlap. The overlap of nets in different clusters is minimized, as described in Section 3.5.2. The clustering is performed such that each cluster has approximately 25 nets each. This choice is motivated by the fact that the SAT routing engine can efficiently route about 25 nets at a time.

Finally, clusters are ordered according to the scheme described in Section 3.5.3. Clusters are now routed using SATRE, our SAT based routing engine. All nets in a given cluster are routed simultaneously.

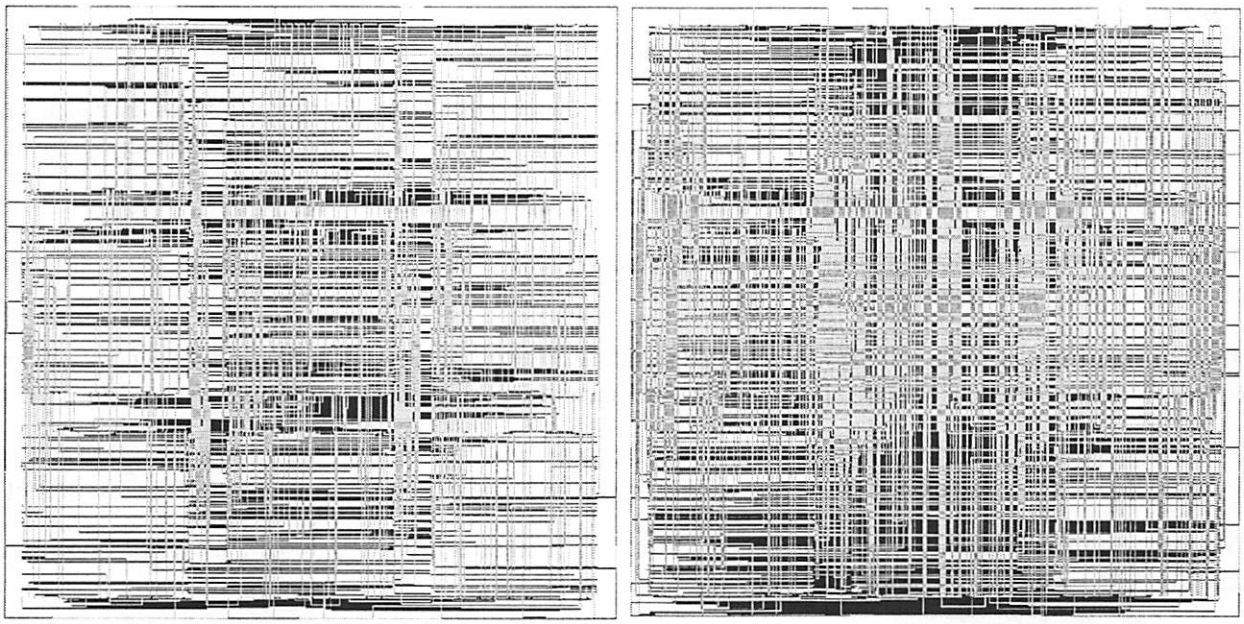
The SAT solver we used is based on the SAT solver distributed within the *atpg* package of SIS [15]. Our SAT solver incorporates significant modifications to the SIS SAT solver. One major difference is the implementation of *non-chronological* backtrack, as described in Section 3.4. The other major modification is the implementation of the conflict checking routine, also described in Section 3.4. As mentioned in Section 3.3.3, we have a command-line option which can disable the use of direction clauses. In the experiments we report in this section, we have disabled the use of direction clauses.

The examples we chose were modifications of MCM routing examples. Since there are no standard set of benchmarks for the VLSI area routing problem, we were forced to use the modified MCM examples.

Our code was implemented in SIS [15], and results were obtained on a 625MHz DEC Alpha 21164 based machine, with 2GB of memory.

Routing results are described in Table 1. In this table, column 2 lists the number of 2-terminal nets in the example. Column 3 lists the size of the routing problem. In all cases, as described in column 4, we performed the routing using 4 layers of metal. Column 5 reports the average number of wire segments utilized in each net. Notice that these numbers are close to 3, which suggests that 2-via patterns were usually sufficient to route most of the nets. Column 6 reports the total wire length, while column 7 reports the best case wire length possible. Column 7 is obtained by summing the Manhattan wire lengths of all the nets in an example. We note that, on average, the total wire length is a mere 1.47% larger than the total Manhattan wire length. In the worst case (example t300), the total wire length is a low 2.92% larger than the total Manhattan wire length. This suggests that SATRE is able to deliver extremely high quality routes. Column 8 provides run-times for the overall routing algorithm.

The router output for both layer pairs of the t500 example is shown in Figure 6. The routing solution for layers 1 and 2 appears less dense, since it has more blockages in the routing area. This is because the source and target vertices of routes on layers 3 and 4 create blockages on layers 1 and 2.



(a) Solution for layers 1 and 2

(b) Solution for layers 3 and 4

Figure 6: Solution for t500

Example	Nets	Problem Size	Layers	Segments per net	Total wire length	Manhattan wire length	runtime
t200	211	400	4	2.82	22651	22232	4.2
t300	437	600	4	3.06	89564	87022	33.4
t400	751	800	4	3.37	208335	206453	361.0
t500	1025	1000	4	3.31	342285	339919	401.0
t600	717	1200	4	2.86	236593	234447	284.1

Table 1: Area Routing Results

5 Conclusions and Future Work

In this paper, we showed that in the deep sub-micron technology era, the traditional notion of global-then-detailed routing is not viable. Instead, we propose a scheme which allocates the nets to be routed to different layer pairs. The nets for each layer pair are then partitioned into net clusters, each of which is routed independently, using SATRE, our SAT-based area routing engine.

This new routing methodology has several novel features.

- The global-then-detailed routing approach is replaced by an approach which assigns a metal layer pair to each net based on the capacities of each metal layer pair. This approach is more realistic for DSM technologies.
- We are able to handle routing problems with an arbitrary numbers of metal layers.
- The SAT based router represents the routing problem in an extremely efficient way. For a problem of size N, only $O(log_2(N))$ SAT variables are required.
- The SAT based router routes all the nets in a given net cluster simultaneously. Although all net clusters are not routed simultaneously, this feature significantly alleviates the ordering dependencies among nets to be routed.
- The SAT based router delivers extremely high-quality routes, as evidenced by the fact that for all the examples we tried, the total net length was on average 1.47% larger than the sum of Manhattan length for all the nets.
- Another feature contributing to the efficiency of the SAT based router is the *non-chronological* [5] back-track scheme that is employed in the SAT solver.

In the future, we plan to utilize better SAT solvers. Another promising approach to the routing problem which we plan to address is that of *constructive* routing, where nets are incrementally, and simultaneously routed. Also, the SAT formulation needs to be extended such that cross-talk between adjacent wires is minimized. Finally, better ways to handle multi-terminal nets need to be investigated.

References

[1] N. Sherwani, *Algorithms for VLSI physical design automation*. Kluwer Academic Publishers, 1998.

[2] "The National Tecnology Roadmap for Semiconductors." `http://notes.sematech.org/97melec.htm`, 1997.

[3] H. Zhou and D. F. Wong, "Crosstalk-constrained maze routing based on lagrangian relaxation," in *Proceedings International Conference on Computer Design*, pp. 628–33, 1997.

[4] S. Khatri, A. Mehrotra, M. Prasad, R. Brayton, and A. Sangiovanni-Vincentelli, "Routing techniques for deep sub-micron technologies," Tech. Rep. UCB/ERL M99/15, Electronics Research Laboratory, University of California, Berkeley, March 1999.

[5] M. Silva and J. Sakallah, "GRASP-a new search algorithm for satisfiability," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, pp. 220–7, November 1996.

[6] R. Zabih and D. A. McAllester, "A rearrangement search strategy for determinning propositional satisfiability," in *The National Conference on Artificial Intelligence*, pp. 155–60, 1988.

[7] T. Back, A. Eiben, and M. E. Vink, "A superior evolutionary algorithm for 3-SAT," in *Evolutionary Programming VII 7th International Conference, EP98*, pp. 125–36, March 1998.

[8] S. Devadas, "Optimal layout via Boolean satisfiability," in *Proceedings of the International Conference on Computer-Aided Design (ICCAD)*, pp. 294–97, November 1989.

[9] R. G. Wood and R. A. Rutenbar, "FPGA routing and routability estimation via Boolean satisfiability," *IEEE Transaction on VLSI systems*, vol. 6, pp. 222–97, June 1998.

[10] G. Nam, K. Sakallah, and R. Rutenbar, "Satisfiability-based layout revisited: Detailed routing of complex FPGAs via search-based Boolean SAT," in *International Symposium on Field Progrmmable Gate Arrays (FPGA'99)*, pp. 167–75, February 1999.

[11] K. Khoo and J. Cong, "An efficient multilayer MCM router based on four-via routing," *IEEE Transcations on Computer-Aided Design of Integrated Circuits and Systems*, vol. 14, pp. 1277–90, October 1995.

[12] Y. Cha, C. S. Rim, and K. Nakajima, "A simple and effecitive greedy multilayer router for MCMs," in *Proceedings of the International Symposium on Physical Design*, April 1997.

[13] G. Karypis and V. Kumar, *A Software package for Partitioning Unstructured Graphs, Partitioning Meshes and Computing Fill-Reducing Orderings of Sparse Matrices*. http://www-users.cs.umn.edu/ karypis/metis, September 1998.

[14] A. Margarino, A. Romano, A. D. Gloria, F. Curatelli, and P. Antognetti, "A tile-expansion router," *IEEE Transactions on Computer-Aided Design*, vol. 6, pp. 507–17, July 1987.

[15] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldanha, H. Savoj, P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli, "SIS: A System for Sequential Circuit Synthesis," Tech. Rep. UCB/ERL M92/41, Electronics Research Lab, Univ. of California, Berkeley, CA 94720, May 1992.