A Gridless Multi-Layer Router for Standard Cell Circuits using CTM Cells

Hsiao-Ping Tseng and Carl Sechen

Department of Electrical Engineering, Box 352500
University of Washington, Seattle, WA 98195

Abstract

We present a gridless multi-layer router suitable for standard cell circuits using central terminal model (CTM) cells. A CTM cell has pins in the middle which split the over-the-cell routing region into top and bottom parts. Our router routes nets in both the channel (if needed) and over-the-cell. The router uses a combined constraint graph and tile expansion algorithm. It achieves channelless solutions for the Primary1 circuit by routing over the cell in three layers. For classical channel routing examples, it achieves solutions at density for Deutsch's difficult example in two, three, four and five metal layers. It also generates equal or better results compared to the best of the previous channel routers for all the examples we have tried.

1 Introduction

The detail routing problem for standard cells is an important issue in VLSI physical design automation. As multi-layer metal technology evolves, a router has to handle the fact that the different metal layers have different design rules and routing over the cell becomes available. For a grid-based router, different grid sizes can cause a via alignment problem between different layers. Some critical nets (e.g. clock, power nets) need to be routed with a larger wire width than other signals. To handle these problems, our router takes two gridless approaches - *a graph based pre-router* and *a tile expansion maze router.* It routes nets in the over-the-cell region for the *Central Terminal Model* (*CTM*) cells.

Central Terminal Model (CTM) cells have pins in the middle and split the over-the-cell(OTC) routing region into a top part and a bottom part. In this model, a channel router is able to process the problem by moving the boundaries of the channel to exclude the intra-cell routing wires and evenly utilizing the bottom half of one OTC region and the top half of another. We use our "channel" router to do over-the-cell routing in the multi-layer standard cell layout system that we are developing. All of our CTM cells [22][23] are automatically generated such that their pins are near the centers of the cells (in the *y* direction, where the rows run horizontally). A "channel" in our system has its top pins near the center of one row and its bottom pins near the center of the neighboring row just below the previous one.

Our experimental results in Section 7 show that a pure maze router doesn't generate a good solution efficiently. Usually, an intensive rip-up and reroute process is necessarily involved in order to achieve the best result for a maze router. The rip-up and reroute process runs rigorously and slowly for congested channels or areas. However, an efficient rip-up and reroute process would require global information on net-ordering. We therefore developed a graph based algorithm to pre-process the net-ordering problem to assist the maze router, and thus quickly generate a good gridless solution. Our combined approach has the time efficiency of a graph-based router and the better results of a maze router.

Constraint graph based algorithms for two [7] and three [8][9] metal layers have been developed. The three layer routers [8][9] use two separate graphs to represent the pair of H-V (horizontal-vertical) layers. Instead, we developed a new *unified multi-layer constraint graph* (MCCG) to store constraints from all H and V metal layers. The algorithm for the layer assignment of pins and horizontal wires in the multi-layer problem also uses the MCCG to select a least-cost assignment.

However, the graph-based algorithm doesn't have information on how to generate useful doglegs and overshoots. For very congested channels, we observe that overshoot doglegs and an unrestricted layer wiring scheme are helpful to reach the density solution. There are many routing algorithms that can handle doglegs efficiently, such as the symbolic router YACR2[6], MIGHTY[5], Mulch[12], Chameleon[13], dogleg router[10], the three-layer router[14], the four-layer router[11] and greedy routers[15][16][17]. However, these routers either cannot handle an arbitrary number of layers or they cannot handle variable width wires. We take the tile expansion maze routing approach to generate necessary doglegs and thus optimize the preliminary solution of the constraint graph based pre-router.

The tile expansion routing model has been developed for area routers [2][3]. Both of them adopt the restricted layer wiring scheme and only allow H-V tile expansion on neighboring layers. Our tile expansion router allows expansions along the orthogonal direction or the straight direction on the same layer or neighboring layers. This

ED&TC '97 on CD-ROM

flexible routing style is achieved by a special technique called the *dual tile plane*. The dual tile plane, which is constructed by simply transforming the coordinates of corner-stitched horizontal strips, has the advantages of fast and symmetrical horizontal and vertical expansions. An efficient multi-level rip-up and reroute algorithm is implemented to iteratively move nets from an uncongested track to other tracks so that the uncongested track can be eliminated.

The remainder of the paper describes our multi-layer rip-up and reroute tile expansion channel router called Sockeye. Sockeye is the first report of the use of a tile expansion maze router for the *variable height* channel routing problem. By variable height channel routing we mean that a single execution of the routing algorithm will result in a completely routed channel. The only thing that is not determined *a priori* is the height or number of tracks needed. In contrast, all previous uses of the maze routing algorithm in routing have been in the domain of fixed area routing. In this case, either the router will succeed in routing the specified netlist in the provided area or it will fail. Should it fail, the user (or a controlling program) must completely restart the router from the beginning while providing increased routing area (usually tracks). Typically this process iterates several times before the routing of a channel or area is completed. Sockeye addresses the variable height channel routing problem as follows: First, it quickly generates an initial solution using a graph-based pre-router which is briefly described in Section 3. Second, employing a tile expansion maze router, it iteratively selects a track and then seeks to reroute the nets using this track to the other tracks. Should it be successful, the empty track is deleted. The iterations continue, starting with the least congested tracks, until the multi-layer channel is routed in density or a predetermined time limit has been exceeded. As our results will show, almost always the former occurs first. In any case, a completely routed channel is the result. We describe the CTM cell model in Section 2 and the graph-based pre-router in Section 3. The routing model of the corner-stitching data structure and the dual tile plane is described in Section 4. Section 5 presents the tile expansion models, maze routing algorithm and multi-level rip-up and reroute algorithm. The time and space complexities of the tile expansion algorithm are described in Section 6. Our experimental results are shown in Section 7.

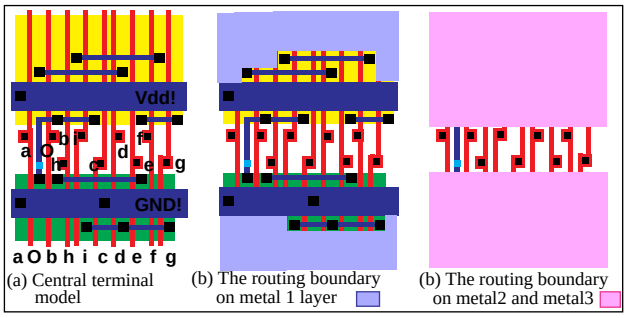
2 The Cell Model

The standard cell design style has been used for decades. Due to the processing technology improvement to multiple metal layers, cell design for over-the-cell routing has become an important key feature to increase the layout density in the last few years. Many physical cell design models have been proposed [22][23][24][25][26][27][28]. In order to utilize the over-the-cell routing area for inter-cell routing by a channel router, we adopt the *central terminal model* (*CTM*) in [22][23]. Our approach uses a generic channel router to evenly assign nets into the combined routing region between the line of top pins and the line of bottom pins on each layer.

In some other cell designs which have pins along the boundaries of the cells, the over-the-cell routing problem is solved in association with the channel routing problem [27]. The dependency between the over-the-cell routing process and the channel routing process prohibits the routing problem from being solved in parallel (i.e. the inter-cell routing between a pair of rows cannot be solved independently of other pairs of rows). Our approach determines the inter-cell routing between two rows without using routing regions in other rows and thus processes the routing problem in parallel.

We show an example of the CTM cell in Figure 1(a). The inter-cell over-the-cell routing regions are shown (shaded areas) in Figure 1(b) and (c).

Figure 1: Center terminal model [22][23] and the inter-cell routing boundary of the cell



3 The Constraint Graph Based Pre-Router

Our multi-layer graph-based pre-router based on ideas from Howard Chen's two-layer router in [7]. We have also proposed a number of extensions to this prior work

3.1 The Constraint Graph Based Routing Model

Horizontal Constraints and Vertical Constraints

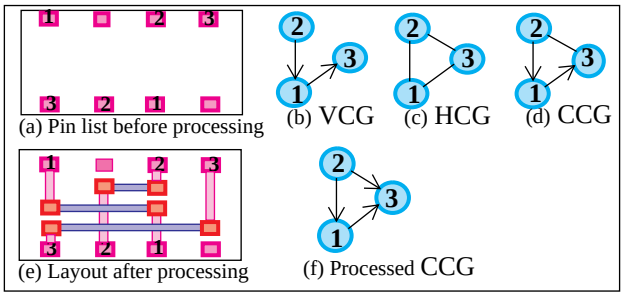
Each layer is chosen to route only either horizontal (H) wires or vertical (V) wires. A multi-terminal net is broken into two-pin nets. A vertical constraint (VC) is introduced when two vias have a vertical violation on the V layer. A horizontal constraint (HC) is introduced when two horizontal wires have a horizontal violation on the H layer.

Node and Graph

A node represents a two-pin net. An edge between two nodes could be either directed as a vertical constraint or undirected as a horizontal constraint. A vertical constraint graph (VCG, see Figure 2(b)) is composed of nodes with directed edges. A horizontal constraint graph (HCG, see Figure 2(c)) is composed of nodes with undirected edges. A combined constraint graph (CCG, see Figure 2(d)) is composed of nodes with undirected and directed edges. The VCGs from the vertical layers and the HCGs from the horizontal layers are merged to form the *multi-layer combined constraint graph* (MCCG). Unlike the other graph

based routers[8][9] using separate routing graphs for each pair of layers, we have a *unified* graph for the multi layers. Based on this constraint graph model, we developed a multi-layer pre-router.

Figure 2: Examples of combined constraint graphs



3.2 The Algorithm

The channel routing problem could be considered as a one-dimensional compaction problem. The goal is to reduce the channel height. We transform the geometrical dependency information of nets into a constraint graph. By assigning all the undirected edges to proper directions, we can determine the proper order of nets and obtain a *complete* MCCG. The longest path of the MCCG is equivalent to the channel height of the layout. Thus, a gridless channel routing problem is transformed into the problem of minimization on the critical path of the MCCG by assigning proper direction for undirected edges (see Figure 2(e)(f)).

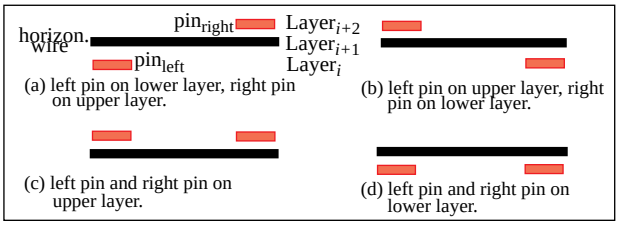
Problem Formulation - In the multi-layer problem, each vertical (V) layer has at least one adjacent horizontal (H) layer and each horizontal layer has at least one adjacent vertical layer. For example, a four layer partition model can be one of the HVHV, VHVH, HVVH and VHHV schemes. The pins can only reside on V layers. The horizontal metal segments can only reside on H layers. The vertical segments are constructed from pins to horizontal segments.

The algorithm is divided into two phases. First, pins and horizontal wires of nodes are assigned to proper layers with minimum weight such that the length of the critical path of the MCCG is minimized. Second, the edge assignment algorithm processes undirected edges in the MCCG to minimize the length of the critical path. The layout is obtained from the complete MCCG. These algorithms are described in subsections 3.2.1 and 3.2.2.

3.2.1 Layer Assignment of Pins and Nodes

In a hierarchical design style, the layer assignment of pins can be done in the global routing stage or in the detail routing stage. Our router allows the processing of the layer assignment of pins in the detail routing stage. If the pins have been already assigned in the global routing stage, only the layer assignment of horizontal wires (nodes) is invoked. The layer assignment for horizontal wires has four different schemes (see Figure 3). All the possible H and V layer combinations for pins and horizontal wires of the two-pin nets are tested iteratively. The combination with minimum weight for node *i* (i.e. the length of the longest path through node *i* in the MCCG) is chosen and then the pins and the horizontal wire of node *i* are assigned to the chosen layers. This layer assignment process efficiently minimizes the longest path in the MCCG.

Figure 3: Layer assignment of horizontal wire



The three-layer Trigger algorithm[8] takes a different approach. It assigns all horizontal wires to horizontal layers H1 and H2 initially and generates weighted constraint graphs WCG1 and WCG3 for H1V2 and V2H3 respectively. It takes an insert-all-and-remove approach which can not handle the multi-layer pin assignment problem because the pins can be on other than the V2 layer for a general multi-layer problem. Our approach is an insert-one-after-another style. According to the given prefixed layer routing scheme, our layer assignment process tries all the possible layer combinations and chooses the best. The preliminary MCCG constructed in this process is further processed by the edge assignment algorithm, which is described in the following subsection.

3.2.2 Edge Assignment

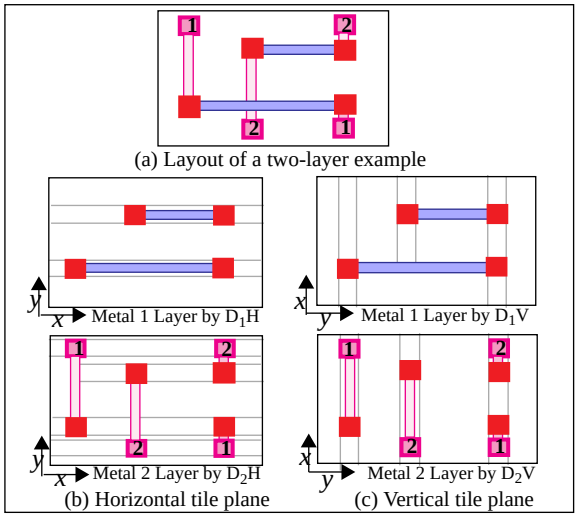
An extended multi-layer version of the two-layer Glitter algorithm[7] was developed to process the undirected edges in the MCCG. Instead of a weighted constraint graph for the two-layer channel routing problem, we use the MCCG to combine multiple VCGs and multiple HCGs for the multi-layer channel routing problem. The Glitter-like algorithm is applied to process the MCCG as follows: 1) The *edge selection* algorithm is first invoked to process the most demanding undirected edges, i.e. those undirected edges which would increase the length of the critical path if assigned to an improper direction, 2) the *node selection* algorithm is applied to choose the most demanding nodes, i.e. those nodes which are on the critical path and closest to the boundaries, and to place them close to either the upper boundary or the lower boundary, and 3) procedures 1 and 2 are iteratively executed until all the nodes and undirected edges are processed.

4 The Tile Expansion Routing Model

The initial solution from the graph based pre-router is first converted into the corner-stitching tile representation in the second stage. The corner-stitching data structure has advantages in retrieving the local geometric information and the space interval information. It was introduced by Ousterhout[1]. Tiles on the plane are stitched and combined either into strips of maximal horizontal extent called a horizontal (H) tile plane or strips of maximal vertical

extent called a vertical (V) tile plane.

Figure 4: Dual tile plane representation



(a) Layout of a two-layer example

(b) Horizontal tile plane (c) Vertical tile plane

A special technique to mirror the coordinates of an H tile plane at 45 degrees can produce a dual V tile plane. We define this type of coupled H and V tile planes as the *Dual Tile Plane* (see Figure 4(b), (c)). The coupled H and V tile planes have consistent geometric information on solid tiles except the coordinates are switched mutually on both planes. The maximal H (V) strip tiles on the H (V) tile plane allow fast tile expansion in the H (V) direction. It extends the horizontal routing efficiency of the conventional single tile plane to both directions of tile expansion.

The dual tile plane D_i is the coupled H (denoted D_iH) and V (denoted D_iV) representation for metal layer *i*.

5 The Tile Expansion Routing Algorithm

Our tile expansion router essentially takes the A* maze routing algorithm technique[4] and uses a *metal expansion model* and a *via expansion model* to find a feasible route with minimum cost. The *metal expansion* model is applied when the routing path is expanded on the current metal layer. The *via expansion* model is applied when the routing path is expanded to the adjacent metal layer. Our expansion models allow unrestricted layer routing - H to V, H to H, V to H and V to V. The rip-up and reroute algorithm allows *overlapped* expansion, in which it seeks a feasible route with a minimum of overlapping with existing solid tiles. We describe the expansion models in section 5.1 and explain the A* maze routing algorithm in section 5.2. The multi-level rip-up and reroute scheme is presented in section 5.3. The summary of the Sockeye algorithm is in section 5.4.

5.1 Expansion Models and Rules

A tile (space or solid) can be expanded either into a tile (space or solid) on the same dual plane or into a tile (space or solid) on the neighboring dual plane. A solid tile with a different signal from the current routing signal is called a *non-equivalent* tile. A non-equivalent tile is allowed to be used in the rip-up and reroute stage. A tile is *expanded* if the expansion path has gone through it. A tile is *generated* if an expanded node selects it for the next step of expansion. The current selected working tile on the end of the expansion path is called an *active tile* (see H0 in Figure 5(a)).

5.1.1 Metal Expansion

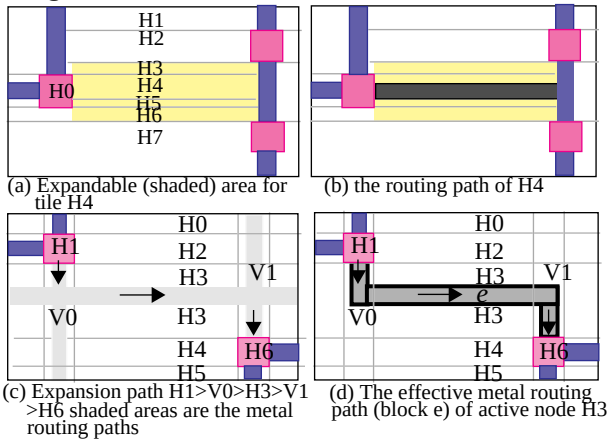
Tile expansion on the same metal layer is called *metal expansion*. It essentially generates the necessary metal wire on the expansion path. The expansion can be along both directions. The terminology for this expansion method is described as follows:

Expandable Area - The maximum rectangular space area covering the selected tile is the expandable area. The fragmented space tiles are accumulated into a large enough area which satisfies the design rules for routing the metal wire (see the expandable area of H4 in Figure 5(a)).

Routing Path for Metal Expansion - The routing path for metal expansion is the formation of metal wires. It is constructed from the left boundary to the right boundary inside the expandable area (see Figure 5(b)).

Effective Routing Path for Metal Expansion - The actual metal covered area on the routing path is called the effective metal routing path. See the block *e* of active node H3 in Figure 5(c), (d)).

Figure 5: Expandable area and metal routing path



(a) Expandable (shaded) area for tile H4

(b) the routing path of H4

(c) Expansion path H1>V0>H3>V1 >H6 shaded areas are the metal routing paths

(d) The effective metal routing path (block e) of active node H3

The active tile first selects the neighbor tiles or overlapped tiles on the same layer as candidates. Then strict design rules are applied to check the routing path of these selected tiles. Only qualified tiles are generated by the active tile.

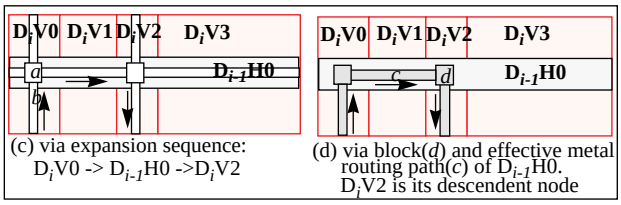
5.1.2 Via Expansion

The tile expansion between neighboring metal layers is called *via expansion*. It essentially generates a necessary via and metal wires on both adjacent layers. The expansion can be along both directions. The terminology for this expansion model is described as follows:

Routing Path for Via Expansion - The routing path for via expansion consists of two parts - the metal routing path and the via block. The metal routing path (block *b* in Figure 6(a)) is constructed like the routing path in metal expansion. The via block is the formation of a via

between the active tile and the selected tile (see block *a* in Figure 6(a)).

Figure 6: Via Expansion on Dual Tile Plane



(c) via expansion sequence: D_iV0 -> $D_{i-1}H0$ ->D_iV2

(d) via block(*d*) and effective metal routing path(*c*) of $D_{i-1}H0$. D_iV2 is its descendent node

Effective Routing Path for Via Expansion - The effective routing path for via expansion consists of two parts - the effective metal routing path and the via block (e.g. blocks *c* and *d* in Figure 6(b), $D_{i-1}H0$ is the active node).

The active tile first selects the overlapping tiles on the adjacent layers as candidates. Then, strict design rules are applied to check the routing path of these selected tiles. Only qualified tiles are generated by the active tile. Our via expansion algorithm allows expansion along the same direction on two adjacent layers to increase the optimization capabilities of the maze router.

5.2 Maze Routing Algorithm

We use the A* algorithm as the search algorithm in our tile expansion router. The A* algorithm was introduced to the routing problem by Clow [4]. It is an efficient maze routing algorithm with the time efficiency of a depth-first algorithm but still guarantees to find the optimal path according to the given weight function.

In every step of expansions, the metal expansions (H to V, H to H, V to H and V to V) and the via expansions (H to V, H to H, V to H and V to V) from the least-cost tile are tested to propagate the wave. By expanding tiles incrementally from the starting pin tile toward the goal pin tile according to the A* algorithm, an optimal route is found when the goal pin tile is expanded. We implemented a rip-up and reroute algorithm based on the maze routing algorithm in a strip-by-strip removal scheme to compact the channel height. It is described in the next section.

5.3 Multi-level Rip-up and Reroute

The rip-up and reroute (RR) algorithm allows the expansion path to overlap with a *non-equivalent* solid tile having a different signal than the current net being routed. By applying this *overlapping tile expansion* feature, a feasible optimal (with respect to the cost function used) route is always found with a minimum of overlapping with non-equivalent solid tiles by the maze router. The router first rips up those two-pin nets which own the overlapping non-equivalent solid tiles, draws the new route for the current net, then continues this rip-up and reroute process for those ripped up overlapped nets.

To prohibit nets from being cyclically thrashed out by each other in congested channels, a multi-level control mechanism was implemented. The *level* of rip-up and reroute is defined as the number of times that the rip-up and reroute procedure has been executed recursively. A feasible route found at a level *i* RR process is denoted as a *tentative route*. A tentative route in the level *0* RR process is not allowed to overlap with non-equivalent tiles. A least congested horizontal region (which would be a track in a grid-based problem) in the channel is selected, all the nets overlapped with that region are ripped up and rerouted elsewhere by a level *n* RR process. The level *n* RR process for node *p* is accomplished by following procedures - (1) search for a tentative route, (2) remove nets which own the overlapped tiles on the route, (3) draw the route, (4) run a level *n-1* RR process for the removed nets, (5) if any net in step 4 fails, collect the *failed* solid tiles owned by the *failed* nets, and (6) call procedure 1 with $n = n\text{-}1$ and with the failed solid tiles not being allowed to be expanded. If the level *n* RR process fails to reroute nets from the selected region to elsewhere, find the next least congested region and continue the process. In our implementation of the multi-level RR algorithm, a *multi-level backtracking* queue is developed to restore the tile planes if any level of the RR process fails.

5.4 Summary of the Sockeye algorithm

Previous uses of the maze routing algorithm in routing have been in the domain of fixed area routing. In this case, either the router will succeed in routing the specified netlist in the provided area or it will fail. Should it fail, the user (or a controlling program) must completely restart the router from the beginning while providing increased routing area (usually tracks). Typically this process iterates several times before the routing of a channel or area is completed. This can be an exasperating experience when confronted with today's very large ASICs containing 100 channels or more.

Sockeye addresses the variable height channel routing problem as follows: First, it quickly generates an initial solution using the graph-based pre-router. Second, employing the tile expansion maze router, it iteratively selects a track and then seeks to reroute the nets using this track to the other tracks. Should it be successful, the empty track is deleted. The iterations continue, starting with the least congested tracks, until the multi-layer channel is routed in density or a predetermined time limit has been exceeded. As our results will show, almost always the former occurs first. In any case, a completely routed channel is the result.

6 Complexity of the Algorithm

The execution time of the graph based routing algorithm is significantly less than the tile expansion maze router's execution time. Therefore we focus our complexity analysis on the tile expansion maze router.

The storage requirement of the tile expansion algorithm is composed of the tiles in the dual corner-stitching tile planes, the nodes in the tile expansion tree and the back-up tiles in the back-tracking queue. First, the number of solid tiles is proportional to the number of two-pin nets. The

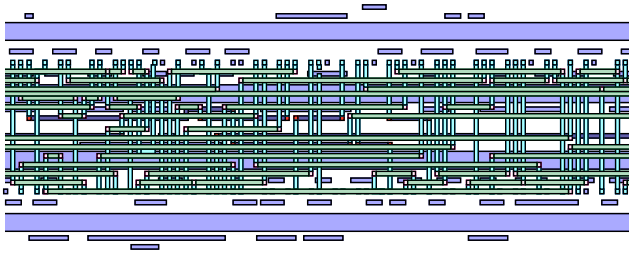
storage requirement for dual tile planes is O(*n)*, where *n* is the number of two-pin nets in the problem. Second, the number of nodes in the tile expansion tree is equal to the number of tiles on the dual tile planes, which in the worst case is when all solid tiles and space tiles are generated and we cannot find a feasible route. Thus the storage requirement for the tile expansion tree in the worst case is also O(*n*). Third, the storage requirement for the multi-level backtracking queue depends on the complexity of the multi-level RR process. The backtracking queue backs up the modified or deleted tiles from the executed tile operations which rip up nets and draw nets on the dual tile planes at each RR level. Let the number of backed up tiles for a net construction be a constant *CT* and that for a net destruction to be a constant *DT*. In the worst case that all nodes in a level *l* RR process have overlapping tentative routes and recursively invoke a level *l-1* RR process to reroute their descendent nodes, $(r(CT+DT))^l$ back-up tiles are stored for each successful route, where *r* is the average number of overlapped nets in each level. In the worst case that the new route overlaps with all other existing routes, *r* is equal to (*n-1*). However, the worst case is highly unlikely since in most cases a non-overlapping route for a net can be found with no need to invoke a level *l-1* RR process, in which the expected and observed space complexity is O(*n*). For the worst case, the space complexity of our algorithm is $O(n + (n-1)^l) = O(n^l)$.

The maze routing algorithm has two hierarchical parts - the route searching procedure and the multi-level rip-up and reroute procedure. The execution time for the route searching procedure is O(*n*) in the worst case in which the tile expansion tree generates all the tiles on the dual tile planes if no route can be found. But this is very unlikely. The execution time to draw a new route is O(*n*), because the tile creation is proportional to the number of space tiles [1]. The worst case of the searching procedure occurs when the new route always overlaps with the existing nets and thereby invokes the next level of the RR process. The level *l* RR procedure calls the route searching procedure $(R \bullet OL)^l \bullet Q$ times, where *R* is a user specified limit on the number of tentative routes for each level, *OL* is the average number of overlapping nets with a tentative route and *Q* is the number of nets on the given rip-up and reroute region (track). In the worst case that all the nets exist in the removed region ($Q = n$) and the new optimal route overlaps with all other existing nets ($OL = n$-1), the time complexity is $O((R \bullet n)^l n \bullet n) = O(n^{l+2})$. However, we found that the route searching procedure is usually able to find a non-overlapping tentative route and therefore the expected or observed time complexity is $O(n^2)$.

7 Experimental Results

Sockeye is implemented in GNU C++ and runs on Unix systems. The current use of Sockeye is in the realm of gridless over-the-cell routing problems for CTM cells which are transformed into variable height channel routing problems, complete with blockages and prerouting (due to intra-cell routing). Since the classic benchmark examples for channel routing are all gridded, we also demonstrate the effectiveness of Sockeye on these gridded examples.

Figure 7: A test channel from Primary1 (an MCNC benchmark circuit) (density=15) routed in one channel track using three metal layers (shaded area is the intra-cell wire)



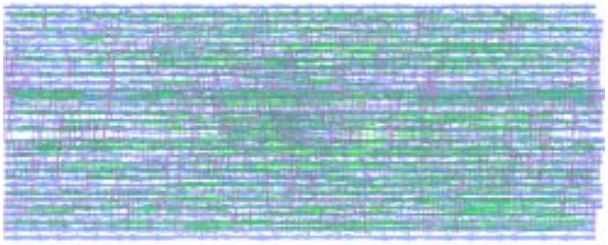
We tested the router on the Primary1 circuit using three layers and the cell library from [22][23]. The placement for Primary1 was obtained using TimberWolfSC[30][31], and the global routing was generated by TimberWolfGR[29].

Figure 7 shows the complete routing solution of a Primary1 test channel with density=15 routed using only one channel track. A complete routed solution for Primary1 is shown in Figure 8. The number of channel tracks is shown in Table 1, in which 2CR stands for the channel density routed by a two-layer channel router and 3CR stands for the channel density routed by a three-layer channel router. Our router generates a channelless solution for every channel and outperforms other OTC routers (see Table 2). Note that ICR-3[24] also routes CTM cells. Our router takes less than 5 seconds to complete every channel in Table 1 on an Intel PentiumPro 200MHz.

Our router also outperforms the other channel routers on the benchmark circuits. A comparison with other multi-layer routers for Deutsch's difficult example [19] is shown in Table 3. The two-layer and three-layer results for examples 3a, 3b, 3c in [18] and the randomly generated examples r1-r4 in [21] are compared with other routers in Table 5. Deutsch's difficult example is denoted as *diff*. The channel height numbers marked with an asterisk are results not achieved previously by any other router. The channel height of two-layer solutions routed by the graph-based pre-router and the number of RR levels applied to each example are also shown in Table 5. The execution times are for a Digital AlphaStation 250/266. Some examples are marked with two numbers for CPU time, the left number is the time needed to achieve a density plus one solution and the right number is the time for a density solution. Figure 10 shows how the time is distributed in eliminating uncongested tracks for the r4 example in two layers as Sockeye approaches a density solution (the 24 track initial solution is generated by the pre-router). We observed that density solutions can be achieved for examples 3a, 3b, 3c by a level 1 or 0 RR process. For the more difficult examples, a level 5 RR process is sufficient to achieve density solutions. The layout of a two-layer solution for Deutsch's difficult example is shown in Figure 11. It is noteworthy

that we did not permit stacked vias for the 3 or more layer cases. Nonetheless, our multi-layer gridless router still obtained equal or better results than the other routers in all cases. Summing the number of tracks for the best reported routing result for each of the eight examples yields the plot shown in Figure 9. Note that Sockeye reached the density solution for all eight examples.

Figure 8: A complete layout of Primaryl in three metal layers



Channel #	2CR (tracks)	3CR	Sockeye 3-layer
1	7	4	0
2	8	4	0
3	11	6	0
4	13	7	0
5	14	7	0
6	12	6	0
7	14	7	0
8	13	7	0
9	13	7	0
10	12	6	0
11	7	4	0
12	9	5	0
13	9	5	0
14	9	5	0
15	9	5	0
Total	160	85	0

Table 1: Channel height in Primary I routed in 3 metal layers

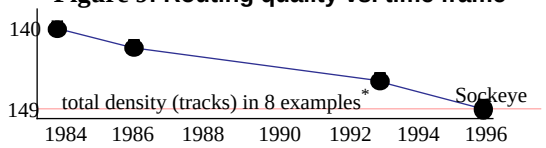
OTC Router	WILMA	ICR-3	Sockeye
2CR	298	298	160
3CR	172	172	85
OTC routing in 3-layer	83	104	0
Improv. %	52	42	100

Table 2: Comparison with OTC routers on Primary I

# of layers (wiring scheme)	2 (HV)	3 (HVH)	4 (HVHV)	5 (HVHVH)
Density	19	10	10	7
Our router	19	10	10	7
Chame-leon [13]	19	11	10	7
F-F-L [20]			10	7

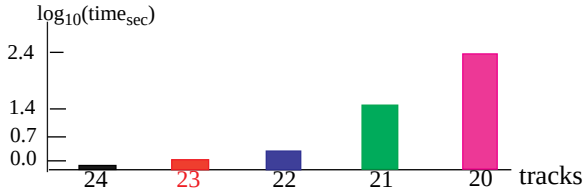
Table 3: Comparison of multi-layer routers for Deutsch's difficult example

Figure 9: Routing quality vs. time frame



*r1-r4, YK3a-3c, Deutsch's difficult examples in two layer model

Figure 10: Incremental execution time toward a density solution for the r1 example in two layers



The value of the graph based pre-router in solving the net ordering problem is readily seen from the data in Table 4. In producing the data for this table, we turned off the graph based pre-router. In other words, only the tile expansion A* maze router was used. Even though the maze router was given 2-4 extra tracks beyond density, it was still unable to find a valid routing (except for one example), even after extensive rip-up and reroute. This proves that a fast graph-based pre-router can greatly improve the results produced by a pure maze router, both in routing quality and run time. This is a result not previously reported in the literature and apparently not known in practice.

	Examples				
	diff.	**r1**	**r2**	**r3**	**r4**
Density for two layers (tracks)	19	20	20	16	15
Given routing area (tracks)	21	24	22	19	18
# of two-pin nets	224	136	122	146	173
# of failed nets w/o RR	16(7%)	7(5%)	4(3.3%)	9(6%)	11(6%)
# of failed nets w/ level 1 RR	7(3.1%)	3(2%)	0	2(1.4%)	3(1.7%)

Table 4: Sockeye results when the pre-router is not used

8 Conclusion

We have presented a gridless multi-layer router for central terminal cells by using a combined constraint graph and tile expansion algorithm. Our approach represents the first maze router to take advantage of a graph based algorithm to solve the net ordering problem. It achieved channelless solutions for the Primary1 circuit by routing over the cell in three layers. For classical channel routing examples, it achieved solutions at density for Deutsch's difficult example in two, three, four and five metal layers. It also generated equal or better results compared to the best of the previous channel routers for all the examples we have tried. In fact, Sockeye is the first router to achieve density solutions for the r1, r3 and r4 examples for two layers and for the r3 example for three layers.

9 Acknowledgments

We wish to acknowledge the financial support provided by the Semiconductor Research Corporation, the Center for the Design of Analog/Digital ICs (CDADIC), LSI Logic, Intel Corporation and Digital Equipment Corporation.

References

[1] John K. Ousterhout, "Corner Stitching: A Data-Structuring Technique for VLSI Layout Tools." *IEEE Transactions on Computer-Aided Design*, Vol. CAD-3, NO. 1, pp. 87-100, January 1984.

[2] Chia-Chun Tsai, Sao-Jie Chen and Wu-Shiung Feng, "An H-V Alternating Router." *IEEE Transactions on Computer-Aided Design*, Vol. 11, No. 8, pp. 976-991, August 1992.

[3] A. Margarino, A. Romano, A. De Gloria, F. Curatelli and P. Antognetti, "A Tile-Expansion Router." *IEEE Transactions on Computer-Aided Design*, Vol. CAD-6 No, 4, pp. 507-517, July 1987.

[4] Gary W. Clow, "A Global Routing Algorithm for General Cells." Proceedings of ACM/IEEE 21st Design Automation Conference, pp. 45-51.

[5] Hyunchul Shin and Alberto Sangiovanni-Vincentelli, "MIGHTY: A 'Rip-up and Reroute' Detailed Router." Proceedings of IEEE International Conference on Computer-Aided Design (ICCAD), 1986, pp. 2-5.

[6] A. Sangiovanni-Vincentelli *et al*., "A new gridless channel router: Yet another channel router the second(YACRII." Proceedings of IEEE International Conference on Computer-

Aided Design (ICCAD), 1984, pp. 72-75.

[7] Howard H. Chen and Ernest S. Kuh, "Glitter: A Gridless Variable Width Channel Router." *IEEE Transactions on Computer-Aided Design*, Vol. CAD-5, No. 4, pp.459-465, October 1986.

[8] Howard H. Chen, "Trigger: A Three-Layer Gridless Channel Router." Proceedings of IEEE International Conference on Computer-Aided Design(ICCAD), 1986, pp. 196-199.

[9] Roshan Gidwani and Naveed A. Sherwani, "MISER: An Integrated Three Layer Gridless Channel Router and Compactor." Proceedings of 27th ACM/IEEE Design Automation Conference, pp. 698-703, 1990.

[10] Bryan Preas, "Channel Routing With Non-Terminal Doglegs." Proceedings of European Design Automation Conference, pp. 451-458, 1990.

[11] Jingsheng Cong, D. F. Wong and C. L. Liu, "A New Approach to Three or Four Layer Channel Routing." *IEEE Transactions on Computer-Aided Design*, Vol. 7, No. 10, pp. 1094-1104, October 1988.

[12] Ronald I. Greenberg, Alexander T. Ishii, and Alberto L. Sangiovanni-Vincentelli, "Mulch: A Multi-Layer Channel Router Using One, Two, and Three Layer Partitions." Proceedings of IEEE International Conference on Computer-Aided Design (ICCAD), 1988, pp. 88-91.

[13] Douglas Braun *et al.*, "Chameleon: A New Multi-Layer Channel Router." Proceedings of 23rd Design ACM/IEEE Automation Conference, pp.495-502, 1986.

[14] Yzi Yoeli, "A Robust Channel Router." *IEEE Transactions on Computer-Aided Design*, Vol. 10, No. 2, pp. 212-219, February 1991.

[15] Tai-Tsung Ho *et al.*, "A General Greedy Channel Routing Algorithm." *IEEE Transactions on Computer-Aided Design*, Vol. 10, No. 2, pp. 204-211, February 1991.

[16] Tai-Tsung Ho, "New models for four and five-layer channel routing." Proceedings of 29th ACM/IEEE Design Automation Conference, pp. 589-593, 1992.

[17] Tai-Tsung Ho, "A Density-Based Greedy Router." *IEEE Transactions on Computer-Aided Design*, Vol. 12, No. 7, pp. 974-981, 1993.

[18] Takeshi Yoshimura and Ernest S. Kuh, "Efficient Algorithms for Channel Routing." *IEEE Transactions on Computer-Aided Design*, Vol. CAD-1, No. 1, pp. 25-35, January 1982.

[19] D. Deutsch, "A 'dogleg' channel router." Proceedings of 13th ACM/IEEE Design Automation Conference, pp. 425-433, 1976.

[20] Sung-Chuan Fang, Wu-Shiung Feng and Shian-Lang Lee, "A New Efficient Approach to Multilayer Channel Routing Problem." Proceedings of 29th ACM/IEEE Design Automation Conference, pp. 579-584, 1992.

[21] R. Rivest, "Benchmark Channel-Routing Problems." private communication from Dr. T. T. Ho, McNeese State University, 1982.

[22] Bingzhong Guan and Carl Sechen, "An Area Minimizing Layout Generator for Random Logic Blocks." Proceedings of IEEE Custom Integrated Circuits Conference, pp. 457-460, 1995.

[23] Bingzhong Guan and Carl Sechen, "Efficient Standard Cell Generation When Diffusion Strapping Is Required." Proceedings of IEEE Custom Integrated Circuits Conference, 1996.

[24] Bo Wu, et al, "Over-the-Cell Routers for New Cell Model." Proceedings of 29th ACM/IEEE Design Automation Conference, pp. 604-607, 1992.

[25] Sreekrishna Madhwapathy, et al, "An Efficient Four Layer Over-the-Cell Router." 1994 IEEE International Symposium on Circuits and Systems, pp. 187-190, vol. 4, June, 1994.

[26] S. Bhingarde, et al, "Middle Terminal Cell Models for Efficient Over-the-Cell Routing in High Performance Circuits," *IEEE Transactions on VLSI Systems*, pp. 462-472, December 1993.

[27] S. Natarajan, et al, "Over-the-Cell Channel Routing for High Performance Circuits." Proceedings of 29th ACM/IEEE Design Automation Conference, pp. 600-603, 1992.

[28] Jason Cong, Bryan Preas, and C. L. Liu, "Physical Models and Efficient Algorithms for Over-the-Cell Routing in Standard Cell Design." IEEE Transactions on Computer-Aided Design, pp. 723-734, Vol. 12, No. 5, May 1993.

[29] William Swartz and Carl Sechen, "A New Generalized Row-Based Global Router." Proceedings of IEEE International Conference on Computer-Aided Design (ICCAD), 1993, pp. 491-498.

[30] William Swartz and Carl Sechen, "Timing Driven Placement for Large Standard Cell Circuits." Proceedings of 32nd ACM/IEEE Design Automation Conference, pp. 211-215, 1995.

[31] Wern-Jieh Sun and Carl Sechen, "Efficient and Effective Placement for Very Large Circuits." *IEEE Transactions on Computer-Aided Design*, Vol. 14, no.3, pp. 349-359, March, 1995.

		Two layer model								Three layer model						
						Sockeye								Sockeye		
Examples	Density	YACR [6]	Cham [13]	MIGHTY [5]	Density-based[17]	height (tracks)	height by pre-router	CPU time	RR level	Trigger [8]	Cham	Robust [14]	Density-based	height (tracks)	CPU time	RR level
r1	20	22	22	22	21	20*	24	0.9m/4m	5		12		11	11	2.2m	5
r2	20	21	20	20	20	20	22	0.9m	4		11		10	10	1.8m	5
r3	16	18	18	17	17	16*	19	1.6m/6m	5		9		9	8*	1.1m/7m	4
r4	15	17	17	17	16	15*	18	1.8m/7m	5		9		8	8	2.9m	4
YK3a	15	15	15	15	15	15	16	5s	0	8	8	8	8	8	40s	0
YK3b	17	18	18	17	17	17	17	1.4s	0	9	10	9	9	9	28s	0
YK3c	18	19	19	18	18	18	19	6s	0	9	10	9	9	9	1.5m	1
diff.	19	19	19	19	19	19	22	2.1m	5	11	11	10	10	10	2.3m	5

Table 5: Comparisons for two-layer and three-layer cases, where m stands for minutes and s stands for seconds

Figure 11: Two metal layer result for Deutsch's difficult example