An Efficient Tile-Based ECO Router with Routing Graph Reduction and Enhanced Global Routing Flow*

Jin-Yih Li
Design Automation Department
Taiwan Semiconductor Manufacturing Company Ltd.
Hsin-Chu 300, Taiwan
jyliv@tsmc.com

Yih-Lang Li
Computer and Information Science Department
National Chaio-Tung University
Hsin-Chu 300, Taiwan
ylli@cis.nctu.edu.tw

ABSTRACT

Engineering Change Order (ECO) routing is frequently requested in the later design stage for the purpose of delay and noise optimization. ECO routing is complicated by huge existing obstacles and the requests for various design rules. Tile-based routers have work with fewer nodes of the routing graph than grid and connection-based routers; however, the number of nodes of the tile-based routing graph has grown to over a thousand millions for SOC designs. This work depicts a new ECO routing design flow with routing graph reduction and enhanced global routing flow. Routing graph reduction reduces the complexity of nodes by removing redundant tiles and aligning neighboring tiles to merge adjacent block tiles. Routing graph reduction reduces tile fragmentation such that the ECO router can run twice as fast without sacrificing routing quality. Enhanced global routing flow incorporates ECO global routing with extended routing and GCell restructuring to prevent routing failure in a routable routing. The ECO router with new design flow can perform up to 20 times faster than the original tile-based router, at the cost of only a very small decline in routing quality.

Categories and Subject Descriptors

B.7.2 [**Hardware**]: Integrated Circuits—*Design Aids*: Placement and Routing

General Terms: Algorithms, Design, Performance

Keywords: ECO routing, gridless router, tile-based router, connection-based router, global routing

* This work was partially supported by the National Science Council of Taiwan by Grant No. NSC 92-2220-E-009-031.

ISPD'05, April 3–6, 2005, San Francisco, California, USA.

1. INTRODUCTION

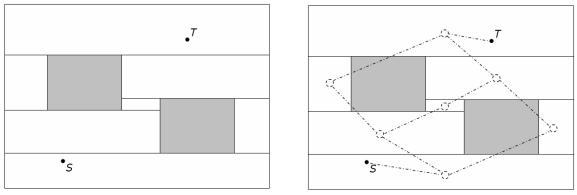
In the era of deep submicron (DSM) technology and SoC design, multi-million gate designs raise new challenges for layout optimization, where interconnect optimization becomes a dominant factor by the trend of ongoing shrinkage of the devices, wire width and wire space required by wires. Wire sizing and wire spacing have been surveyed [1], with reference to different design rules on critical or sensitive nets for eliminating excess delay and noise.

ECO routing, which is usually a point-to-point routing operation, is commonly requested toward the end of the design process to optimize delay and noise or to complete an imperfect layout. Many existing interconnections, inevitably flattened traverse operation, different design rules for delay and noise issues, and the possible need for re-extracting a routing environment make ECO routing highly complicated. A gridless router is more flexible than the grid router in accommodating different design rules. A straightforward realization of the gridless router uses fine uniform grids, or manufacturing grids. Although it can accommodate various routing rules, the induced huge routing graph for a large design makes this method inapplicable because it requires too much searching time and memory space. Different models have been investigated to reduce the routing graph, [2-15], of which the connection graph and the tile-based graph are the most popular types. In [5], a connection graph was constructed by extending lines through the boundaries of all obstacles until they intersect with other obstacles or boundaries of the routing region. Its drawback is its inconvenient pre-construction and representation. Cong [6, 7, 8] presented an implicit connection graph whose extended lines may pass through obstacles. The implicit connection graph has the advantages of fast graph creation and the guarantee ability to find an optimal path, although it has more nodes and edges than that in [5]. The implicit graph behavior necessitates the performance of query operation to determine the legality of next move. As a matter of fact, the connection-graph model generates too many graph nodes to allow a fast path search for a large design.

The tile-based model [9-15] is another gridless approach, in which the routing region are partitioned by obstacles into space tiles and block tiles, and represented using a corner-stitching data structure [16]. Figure 1(a) shows a horizontal tile plane with the

maximum horizontally stripped property. A space tile corresponds to a node of the tile-based routing graph and an edge is present between two nodes if the related space tiles are adjacent to each other. Figure 1(b) presents the related routing graph in Fig. 1.(a). The path is sought by tile propagation between two adjacent tiles of a layer or two overlapping tiles of neighboring layers. Both connection graph and tile-based approaches can be used to find an optimal path for ECO point-to-point routing, but a large design, such as a chip-set design, can yield several hundred million tiles for a single routing layer. Therefore, the routing graph must be simplified to accelerate ECO routing.

This work focuses on increasing the tile-based point-to-point routing speed. The rest of this paper is organized as follows. Section 2 briefly reviews the tile-based router and defines basic terminology. Section 3 presents the new design flow for increasing the ECO routing speed. Section 4 presents the routing graph reduction and Section 5 presents the enhanced global routing flow. Section 6 discusses the experimental results obtained using a real design. Finally, Section 7 draws conclusions.



(a) **(b)**

Figure 1. Tile-based graph.

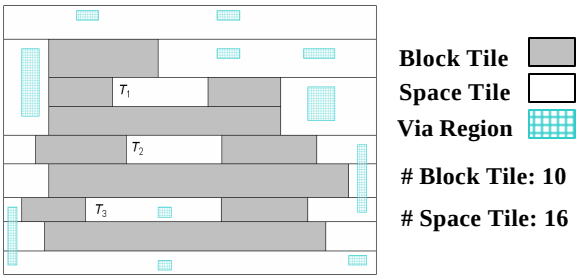


Figure 2. Redundant tiles.

2. PRELIMINARIES

2.1 Tile-Based Router

Tile-based routers have been developed and thoroughly investigated. Fast queries and powerful geometric Boolean operations on the corner-stitching tile plane make routing efficient. The concept behind the tile-based router is to route the centerline of a path along the space tiles on corner-stitching tile planes that are generated by adding the contours of width $w_s + w_w/2 - 1/2$ to all existing shapes, where w_s is the space rule of the net to be routed to all shapes of the same layer as the routed net and w_w is the wire width. The contours can guarantee that the newly created path will not induce any design rule violation. In multi-layer routing, the available via regions at which the center of a new via can be placed can also be calculated in a similar way.

Tile-based point-to-point routing consists of two stages - tile propagation and path construction [14]. Tile propagation is undertaken to reach the target in all possible ways over the space tiles of a single layer and across adjacent layers. Single-step tile propagation is from the current space tile to its neighboring space tile of the same layer or an adjacent layer; the neighboring space tile of an adjacent layer is the adjacent-layer space tile that can accommodate a via that connects the current space tile to itself. Tile propagation applies a pre-defined cost function to guide the search for the path; it then finds an optimal list of free tiles. In multi-layer routing, the tile list may include tiles in different layers. Finally, path construction generates a minimum-corner path that passes through the list of tiles.

2.2 Terminology

The following definitions are used in this work.

Definition. (*essential/redundant tile*) If a space tile can contribute to further propagation, it is called an *essential tile*; otherwise, it is called a *redundant tile*.

Definition. (*conjunct tile*) A tile, *A*, is referred to as a *conjunct tile* of tile *B* if a single-step tile propagation from *A* to *B* on the same layer or across adjacent layers is feasible.

Definition. (*one-conjunct*) A space tile is said to be *one-conjunct* if it has only a single conjunct tile.

Definition. (*0-conjunct*) A space tile is said to be *0-conjunct* if it has no conjunct tile.

Definition. (*global cell*) An entire layout is partitioned into tiles. Each tile is referred to as a *global cell* (*GCell*).

Definition. (*Active GCell*) Connected GCells, the result of ECO global routing, which are used to guide tile propagation are called *active GCells*.

Definition. (*Idle GCell*) Gcells that are not active GCells are called *idle GCells*.

Figure 2 illustrates these definitions. Figure 2 depicts three redundant tiles, T_1, T_2 and T_3. For instance, T_1 and T_3 can be reached by tile propagation from their top neighboring space tile and a neighboring space tile of an adjacent layer, respectively; however, tile propagation from T_1 or T_3 can not explore any new space tile. Accordingly, T_1 and T_3 are the one-conjunct space tiles and tile T_2 is 0-conjunct. Notably, T_3 is accessible only from a tile of another layer through the via region. If T_3 is accessible from more than two tiles of other layers through the via region, then it is an essential tile.

3. ECO ROUTING DESIGN FLOW

The tile-based router comprises three stages - corner-stitching tile plane construction, tile propagation and path construction. The complexity of the visited graph nodes is to be reduced during routing to increase the routing speed. A new ECO routing design flow, which contains *Routing Graph Reduction* (*RGR*) and *Enhanced Global Routing Flow* (*EGRF*) is proposed. RGR includes the removal of redundant tiles and the alignment of

neighboring tiles; the former removes space tiles that do not contribute to further tile propagation; the latter shrinks space tiles in an attempt to merge adjacent block tiles. During the alignment of neighboring tiles, extended blockage is prevented from reducing the number of conjunct tiles of the shrunk space tile. RGR can not only improve routing performance but also preserve routability. EGRF incorporates ECO global routing with extended routing and GCell restructuring to prevent failure in a routable routing. Traditionally, detailed routing follows global routing, and a new global routing may be requested if detailed routing fails to find a path. The flow is enhanced by introducing extended routing and GCell restructuring. Besides, each GCell has six internal edges used to configure the connectivity, which indicates a path that passes through itself and connects two neighboring GCells, among its four neighboring GCells. Extended routing is applied by expanding the range of the GCell through which the routing path cannot pass, and then rerouting the extended GCell. If extended routing also fails, then the internal edges of those active Gcells that cannot reach to their neighboring GCells by tile propagation are restructured. Two adjacent GCells can be connected in three ways in a single direction; for instance, a GCell can be connected to its right neighbor from top to right, from left to right, or from bottom to right. If no path that connects two adjacent Gcells exists, then GCell restructuring breaks that connectivity in a manner to prevent subsequent global routing from involving these two GCells in the same way. Section 5 presents more details concerning the GCell's internal edge. Visited GCells are used as start points, and global routing is performed again to yield new results following GCell restructuring. This process is repeated until a feasible path is identified or all GCells have been visited. Figure 3 depicts the proposed new ECO routing design flow.

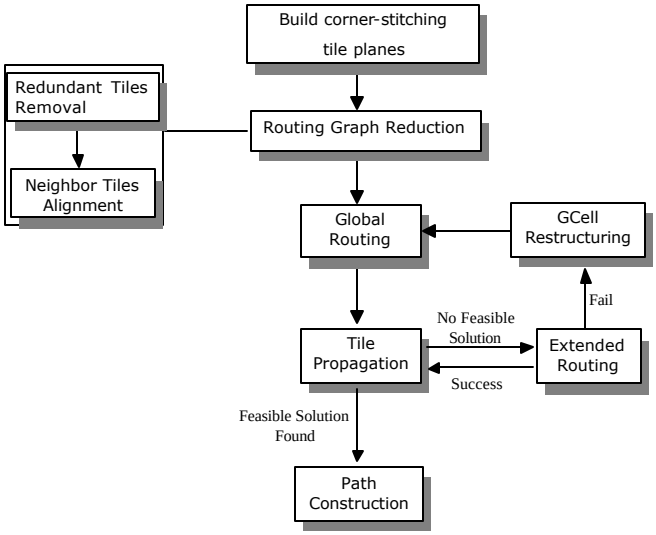


Figure 3. New ECO routing design flow.

4. ROUTING GRAPH REDUCTION

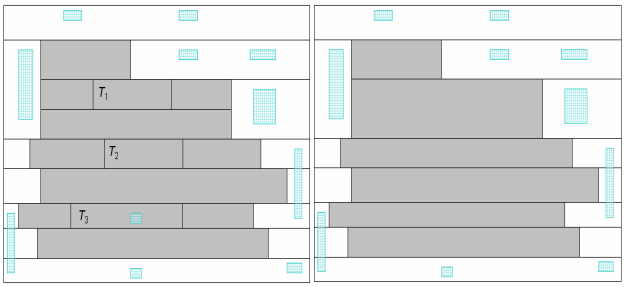
The corner-stitching tile planes of a design with numerous existing obstacles are generally fragmented. This phenomenon increases the computational complexity of tile propagation. This section presents two methods – the removal of redundant tiles and the alignment of neighboring tiles - to reduce tile fragmentation.

4.1 Removing Redundant Tiles

The process of tile propagation was observed. Many paths are terminated because they enter a space tile that has no exit that allows further tile propagation. Such space tiles do not contribute to routing, but they increase the problem of tile fragmentation. Clearly the redundant tiles are the one-conjunct space tiles and the 0-conjunct space tile.

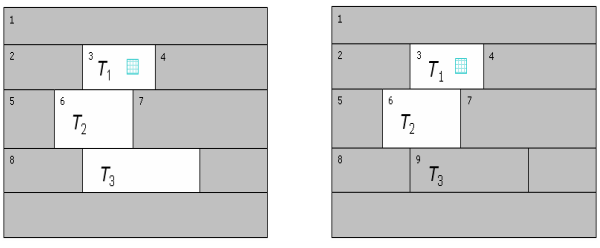
The corner-stitching data structure supports fast neighbor finding and area enumeration that visits each tile exactly once. Accordingly, the redundant tiles can be efficiently removed by examining one-conjunct and 0-conjunct space tiles within an enumeration operation over the entire tile plane. Notably, only the redundant space tiles are changed into block tiles without merging them with their neighboring block tiles; the intermediate tile plane is not maximally horizontally or vertically stripped because merging during enumeration may disrupt the order of enumeration. After the enumeration operation has been completed, the tile plane is reconstructed to preserve the maximally horizontal or vertical strip. Figure 4(a) shows an intermediate tile plane during the removal of redundant tiles, where T_1, T_2 and T_3 are initially space tiles. Figure 4(b) displays the final result of removing redundant tiles and the tile complexity is reduced by seven tiles.

Essential tiles may become redundant after their neighboring redundant tiles have been removed, as depicted in Fig. 5(a). Initially, tiles T_1 and T_2 are essential tiles. After the redundant tile T_3 is removed, T_2 becomes redundant, as shown in Fig. 5(b). Also T_1 becomes a redundant tile after T_2 is removed.



(a) **(b)**

Figure 4. Removal of redundant tiles.



(a) **(b)**

Figure 5. Essential tiles become redundant tiles.

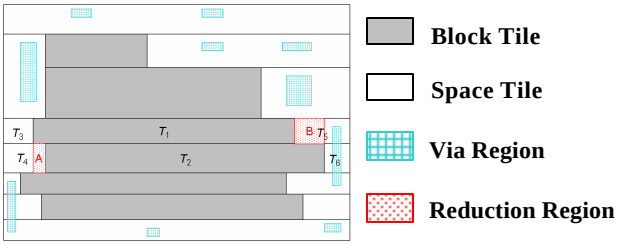
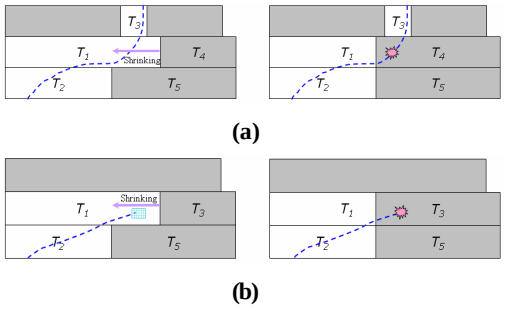


Figure 6. Neighboring Tiles Alignment.

4.2 Aligning Neighboring Tiles

This section illustrates the alignment of neighboring tiles using the tile plane of the maximally horizontal strip. On a maximally horizontally stripped tile plane, the adjacent block tiles typically form ragged left and right boundaries. Ragged boundaries cause tile fragmentation. The basic idea is to shrink space tiles to align them with the ragged border such that the neighboring block tiles can merge, as depicted in Fig. 6. In Fig. 6, tiles T_4 and T_5 are shrunk, so tiles T_1 and T_2 are enlarged such that they can be merged to become a tile, where A and B are the reduction regions; leftward shrinkage is applied to T_4, and rightward shrinkage is applied to T_5. Here, a rightward/leftward shrinkage on T_i is considered and a rightward/leftward shrinking is applied to T_i's left/right border.



(a) **(b)**

Figure 7. Two illegal shrinkings.

Notably, not all space tiles can be shrunk. If a space tile, say T_1, is to be shrunk to be merged with another tile, say T_2, then a feasible shrinking must conform to the following two rules.

Rule 1. A shrinking cannot obstruct an existing path from T_1 to its neighboring tile, say T_3, of the same layer. Figure 7(a) depicts an illegal shrinking that does not satisfy this property. Figure 7(a) shows an original routing path between tiles T_1 and T_3. However, this path will disappear if a leftward shrinking is performed on T_1 to align it with T_2.

Rule 2. A shrinking cannot obstruct an existing path from T_1 to its neighboring tile in adjacent layer. This rule requires that no via region can overlap the reduction region. In Fig. 7(b), a leftward shrinking of T_1 will wipe out a routing path that is connected to an adjacent-layer space tile.

Before the shrinking process can be introduced, the following notation is presented.

- T_a: currently processed space tile.
- $l(T_i)/r(T_i)$: *x-coordinate* of tile T_i's left/right edge.

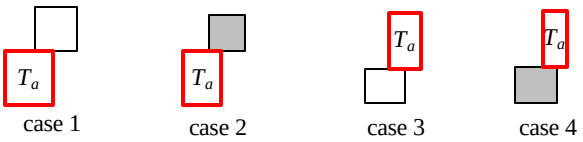
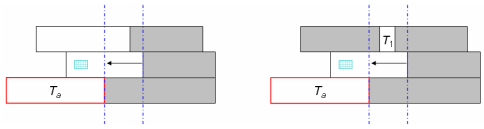


Figure 8. Four shrinking cases.

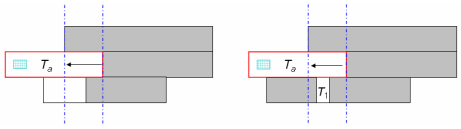
- $R_{LS}(T_i)/R_{RS}(T_i)$: leftward/rightward shrinking on T_i.
- $l(R_{LS}(T_i))/r(R_{LS}(T_i))$: stop/start position of a leftward shrinking on T_i.
- $l(R_{RS}(T_i))/r(R_{RS}(T_i))$: start/stop position of a rightward shrinking on T_i.
- $N_{rt}(T_i)$: rightmost top neighbor of tile T_i.
- $N_{tr}(T_i)$: topmost right neighbor of tile T_i.
- $N_{lb}(T_i)$: leftmost bottom neighbor of tile T_i.
- $N_{bl}(T_i)$: bottommost left neighbor of tile T_i.

Figure 8 lists four cases of shrinkage during enumeration to elucidate how to decide whether a shrinking on a space tile is feasible. Only space tiles are considered because shrinking a useless space region will neither reduce routability nor produce incorrect routing results. In contrast, shrinking block tiles may make a path across present blockages. T_a's top and bottom neighbors differentiate these four cases. Only *Rule 1* is discussed in the following since *Rule 2* is clear.



(a) **(b)**

Figure 9. Shrinking Case (1).



(a) **(b)**

Figure 10. Shrinking Case (2).

Case (1): $N_{rt}(T_a)$ is a space tile and the candidate to be shrunk. The goal is to perform an $R_{LS}(N_{rt}(T_a))$, where $l(R_{LS}(N_{rt}(T_a))) = r(T_a)$ and $r(R_{LS}(N_{rt}(T_a))) = r(N_{rt}(T_a))$. *Rule 1* requires that a top or bottom neighboring space tile of $N_{rt}(T_a)$, say T_1, cannot exist, such that $l(R_{LS}(N_{rt}(T_a))) \le l(T_1) < r(R_{LS}(N_{rt}(T_a)))$. Figure 9(a) depicts legal shrinking, while Fig. 9(b) shows illegal shrinking.

Case (2): $N_{rt}(T_a)$ is a block tile and T_a is the candidate to be shrunk. The goal is to perform an $R_{LS}(T_a)$, where $l(R_{LS}(T_a)) = l(N_{rt}(T_a))$ and $r(R_{LS}(T_a)) = r(T_a)$. *Rule 1* requires that a bottom neighbor space tile of T_a, , say T_1, cannot exists such that $l(R_{LS}(T_a)) \le l(T_1) < r(R_{LS}(T_a))$. Figure 10(a) depicts a legal shrinking, while Fig. 10(b) shows an illegal shrinking.

Case (3): $N_{lb}(T_a)$ is a space tile and the candidate to be shrunk. The goal is to perform an $R_{RS}(N_{lb}(T_a))$, where $l(R_{RS}(N_{lb}(T_a))) =$

$l(N_{lb}(T_a))$ and $r(R_{RS}(N_{lb}(T_a))) = l(T_a)$. *Rule 1* dictates that a top or bottom neighboring space tile of $N_{lb}(T_a)$, say T_1, cannot exist, such that $l(R_{RS}(N_{lb}(T_a))) < r(T_1) \le r(R_{RS}(N_{lb}(T_a)))$.

Case (4): $N_{lb}(T_a)$ is a block tile and T_a is the candidate to be shrunk. The goal is to perform an $R_{RS}(T_a)$, where $l(R_{RS}(T_a)) = l(T_a)$ and $r(R_{RS}(T_a)) = r(N_{lb}(T_a))$. *Rule 1* dictates that a top neighbor space tile of T_a, say T_1, cannot exist, such that $l(R_{RS}(T_a)) < r(T_1) \le r(R_{RS}(T_a))$.

Tiles are only shrunk during enumeration to maintain the correct enumeration order. After the enumeration was completed, the tile plane was reconstructed to preserve the maximum horizontal strip.

5. ENHANCED GLOBAL ROUTING FLOW

Global routing is used to guide the detailed routing to reduce the range of the path search. A new global routing may be requested if detailed routing fails to find a path. The flow is enhanced by introducing extended routing and GCell restructuring to guarantee the finding of a feasible solution to routable routing.

5.1 Global Routing Graph

The entire layout is partitioned into several *global cells* (*GCell*). The *GC(i,j)* is defined as the GCell at the *i-th* column and *j-th* row. The global routing graph *G*(*V*,*E*) is defined as follows; each GCell corresponds to a node of *G*(*V*,*E*) and an edge exists between two nodes if their related partitions are adjacent. The *V(i,j)* is defined as the vertex at the *i-th* column and *j-th* row. Each GCell and its related vertex have six common internal edges - *nw*, *ws*, *se*, *en*, *ns* and *ew*. The *nw* edge represents the connectivity between the partition's north and west boundaries and the *se* edge represents the connectivity between the partition's south and east boundaries. The connectivity of an internal edge is either *connected* or *disconnected*. A connected *nw* edge of a partition, say P_i, indicates the existence of a routing path across P_i , which connects P_i's north and west neighbors. For instance, if *V(i,j)'s* internal edge *nw* is disconnected, then no path connects nodes *V(i,j+1)* and *V(i-1,j)* across *V(i,j)*. All internal edges are set to be initially connected.

Most likely, an ECO routing region is very congested and numerous existing obstacles fragment available routing regions, so finding a straight path across a GCell is difficult and the path may be zigzag. Accordingly, the via region is more important than the wire region in helping a path to pass through a congested region. Estimating the routability of the resources based on the via region model for ECO global routing differs greatly from doing so for traditional global routing, which mainly calculates the utilization ratio for the regions. If a GCell has more available via resources, then a path that passes through it can be more easily found. The availability of via resources is estimated from the total area of the available via regions, which is calculated quickly by enumerating the space tiles on the via tile planes. The via capacity, *VC*, of a GCell is defined as,

$$VC = VA/A, \quad (1)$$

where *VA* is the total area of the available via region of the GCell, and *A* is the area of the GCell. Each vertex in G is assigned a cost, *c*, which is inversely proportional to *VC*. Cost *c* is defined by a piecewise linear function as,

$$c = \begin{cases} 1/VC & if\ VC > t \\ k/VC & if\ VC \le t \end{cases} \quad (2)$$

where *t* is a threshold value, and *k* is an amplification scalar. Experimental observations reveal the predicament of obtaining a path through the GCell when *VC* <0.01, so the cost is amplified by multiplying it by a scalar. Also, each edge of G is assigned a length cost, *lc*, to indicate the net timing factor. The length cost *lc* is set to be *2/t*, and the total cost is to add up every visited vertex cost *c* and their length cost *lc*. Based on the cost design, the global router can avoid passing through the vertex with little via resource at the expense of using more wire length. For those nets with timing constraints, the total length cost of a path can not exceed its timing budget; therefore, the global router may find a path containing some congested region in order to meet its timing budget.

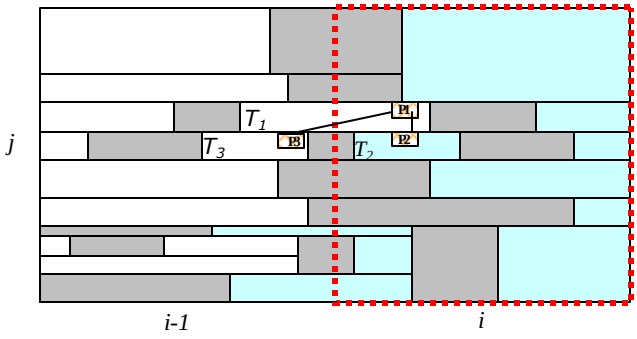


Figure 11. Tile propagation outside the active

Dijkstra's algorithm is applied in G to find the minimum-cost path that connects two terminals to be routed. The active GCells are the those on the minimum-cost path, and their related nodes associated with *G(V,E)* are referred to as *active nodes*. Further propagation entirely out of active GCells is prohibited and the path along such propagation is called an *idle path*. In Fig. 11, *GC(i,j)* is an active GCell, while *GC(i-1,j)* is not. The path node *p1* can propagate into tiles T_2 and T_3 at *p2* and *p3*, respectively. *p3* enters a space tile wholly within an idle GCell, so the path *p3*®*p1* is an idle path. To maintain idle paths, each idle GCell stores the idle paths inside itself in its own heap, called the *idle-path heap*. The idle paths of an idle GCell will be popped up and become active for further propagation when the idle GCell is selected and becomes active during subsequent global routing.

5.2 Extended Routing & GCell Restructuring

This section describes the solution to the problem when the router cannot find a feasible path along the active GCells. Figure 12(b) is an example of routing failure. Figure 12(a) shows the active GCells of a global routing from *s* to *t* and the propagation status: darker GCells stand for the *visited GCells*. A visited GCell

through which a path does not pass is called a *blocked GCell*. Accordingly, *GC(1,6)*, *GC(1,5)*, *GC(1,4)*, *GC(1,3)*, *GC(2,3)*, *GC(3,3)* and *GC(4,3)* are visited GCells, and *GC(4,3)* is a blocked GCell. A blocking wall can be imagined to block all of the paths in *GC(4,3)*, to propagate to the next GCell, *GC(5,3)*. Two methods are proposed to solve this problem. The first is to expand the search range around the blocked GCell, in a process called extended routing. The second is the GCells restructuring. Based on the previous routing results, the internal edges of the visited GCells are restructured to prevent the subsequent global routing from including as a routing path the two GCells that are connected by a disconnected internal edge, and then a new list of active GCells is rescheduled by performing a new global routing.

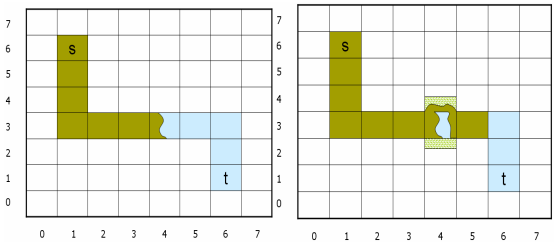


Figure 12. An example of extended routing

5.2.1 *Extended Routing*

Extended routing is conducted to expand the search scope by a half of the width of a GCell around the blocked GCell. The tile propagation starts from the path nodes in the idle-path heaps of those idle GCell adjacent to the blocked GCell. For example, *GC(4,3)* is the blocked GCell in Fig. 12(b), and the search scope is expanded around *GC(4,3)*. The path nodes in the idle-path heaps of *GC(4,4)* and *GC(4,2)* will be popped up for further propagation. Extended routing increases the capacity to connect two active Gcells. Figure 12(c) depicts a successful connection between *GC(3,3)* and *GC(5,3)*.

5.2.2 *GCell Restructuring*

If an extended routing still cannot pass through a blocked GCell, the active GCells are rescheduled to find another path. Before a new global routing is restarted, the internal edges' connectivity states of the active GCells must be modified to reflect the blocking information between the GCells. In Fig. 13(a), *GC(i,j)* is the blocked GCell. Clearly, no path passes through *GC(i,j)* and connects *GC(i-1,j)* to *GC(i+1,j)* and *GC(i,j+1)*. The internal edges' connectivity states of the active GCells can be very easily determined. The connectivity states of the internal edges of *GC(i,j)* are to be determined and the routing direction through *GC(i,j)* is from left to right; only the idle-path heaps of *GC(i,j+1)* and *GC (i,j-1)* need to be checked. If the idle-path heaps of *GC(i,j-1)* and *GC(i,j+1)* are empty, then the connectivity states of edges *nw* and *ws* should be disconnected. Meanwhile, the connectivity state of edge *ew* must be set to disconnected if *GC(i,j)* is a blocked GCell. In Fig. 13(a), the connectivity states of edges *nw*, *ws* and *ew* are set to disconnected.

After the internal edges have been restructured, a new global routing is performed using the visited GCells as the start vertices, as shown in Fig. 13(a). Based on the updated connectivity states of the internal edges, the results of new global routing will not select the blocked path. The active GCells used in the next tile propagation include the new active GCells and the visited GCells. The new active GCells were idle GCells before. Some of them must be adjacent to the visited GCells and at least one of them must have a nonempty idle-path heap. These nonempty idle-path heaps are popped up to initiate the new tile propagation. In Fig. 13(b), *GC(3,2)*, *GC(3,1)*, *GC(4,1)* and *GC(5,1)* are new active GCells, and *GC(3,2)*, which is adjacent to the old active GCell, *GC(3,3)*, must have a nonempty idle-path heap.

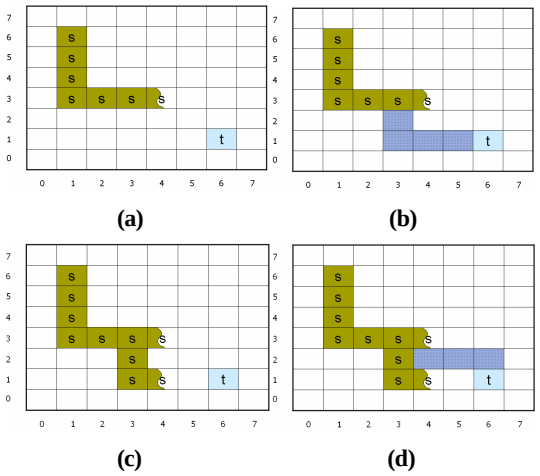


Figure 13. GCell restructuring and rescheduling.

GCell restructuring and rescheduling may be repeated several times during the tile propagation. In Fig. 13(c), tile propagation stops again at *GC(4,1)*. Therefore, active GCells are restructured and rescheduled, as shown in Fig. 13(d). The path nodes in the idle-path heaps of *GC(4,2)* may come from *GC(4,3,)*, *GC(4,1)* and *GC(3,2)*, and they will be popped up for further propagation. Notably, when the active GCells are rescheduled, the number of active GCells increases. The visited GCell will be revisited if newly entering paths are of lower cost than before. The worst case is to make all of the GCells active such that the tile propagation propagates over the whole tile plane. Accordingly, a feasible solution will always be found for a routable routing. Additionally, even when the worst case happens, the routing speed does not substantially drop off because most of the routing regions that have already been visited are not visited again as a result of good routing resource estimation.

6. EXPERIMENTAL RESULTS

A tile-based ECO router with a new routing design flow proposed in this work were implemented using *C++* language. The routing was undertaken using a *2.4GHz* Pentium4 PC with 1GB RAM using a real VLSI design that consists of 114,155 standard cells and contains 632,634 *metal-2* rectangles, 399,993 *metal-3* rectangles, 399,852 *via-12* vias and 618,218 *via-23* vias.

Table 1 presents the number of tiles of the metal2-layer and metal3-layer planes. The results without routing graph reduction are listed at the second column, and the data in the third column is obtained after the routing graph has been reduced. The number of tiles is reduced by 57%. Table 2 lists the pre-process time before routing. The second column presents the times taken to construct corner-stitching planes and the third column presents the time required for RGR, where the former is necessary for both traditional approach and the newly proposed design flow in this work, and the later is the additional computation time required by this work.

Seven p2p routings were performed using three methods to yield the routing statistics, including the routing time (tile propagation time), the length of the wire and the number of vias. The second column in Table 3 concerns pure tile propagation, and the third column concerns the application of routing graph reduction. The routing graph reduction reduces the routing time by approximately 50%, as shown in column T1. The speed of the ECO router can be approximately doubled, as shown in column T2. The wire length and the number of vias are almost the same.

Table 1.The number of tiles of the tile planes.

# of Tiles / Layer	Origin		After RGR	
	#Block Tiles	#Space Tiles	#Block Tiles	#Space Tiles
Met2	1067179	973528	470325	380798
Met3	591120	541706	289144	218444
Total	3173533 (C_1)		1358711 (C_2)	
Reduction rate	0.571			

※ Reduction rate: $(C_1-C_2)/C_1$

Table 2. The pre-process time.

Pre-process	CS Plane Construction (T_{cs})	RGR (T_{rgr})
Time (second)	21.656	5.889

Table 4 shows the results of the new ECO routing design flow, and the right two items in the second column reveal how many times the extended routing and GCell restructuring and rescheduling are performed. The routing time is around 85% lower than for the pure tile-based router, as shown in column T3. The increase in wire length ranges from 3% to 30%, as shown in column W. The wire is much longer for TEST5 because the start terminal is located in a very dense region, so the global router cannot easily find a good global path.

7. CONCLUSIONS

In this paper, a new ECO routing design flow with routing graph reduction and enhanced global routing flow is proposed. Routing graph reduction reduces the node complexity of the routing graph by removing redundant tiles and aligning neighboring tiles in an attempt to merge adjacent block tiles. Routing graph reduction reduces tile fragmentation such that the ECO router can perform two times faster than the traditional tile-based router without sacrificing routing quality. Additionally, a newly enhanced global routing flow that incorporates ECO global routing with extended routing and GCell restructuring can prevent routing failure in a routable design. Compared with the traditional tile-based router, the runtime can be reduced by around 85%.

8. REFERENCES

[1] J. Cong, L. He, C.-K. Koh, and P. Madden, "Performance optimization of VLSI interconnect layout," *Intergr. VLSI J.*, vol. 21, no. 1–2, pp. 1–94, Nov. 1996.

[2] T. Ohtsuki, "Gridless routers—New wire routing algorithms based on computational geometry, *in Proc. Int. Conf. Circuits and Systems*, pp. 802–809, May 1985.

[3] K. L. Clarkson, S. Kapoor, and P. M. Vaidya, "Rectilinear shortest paths through polygonal obstacles in O(n(log n)) time," in *Proc. 3rd Annual Symp. Computational Geometry*, 1987, pp. 251–257.

[4] Y. Wu, P. Widmayer, M. Schlag, and C. Wong, "Rectilinear shortest paths and minimum spanning trees in the presence of rectilinear obstacles,"*IEEE Trans. Computers*, vol. C-36, no. 1, pp. 321–331, 1987.

[5] S.Zheng, J.S. Lim, and S. Iyengar, "Finding obstacle-avoiding shortest paths using implicit connection graphs,"*IEEE Trans. Computer-Aided Design*, vol. 15, no. 1, pp. 103-110, Jan. 1996.

[6] J. Cong, J. Fang, and K. Khoo, "An implicit connection graph maze routing algorithm for ECO routing," *in Proc. Int. Conference Computer-Aided Design*, pp. 163–167, Nov. 1999.

[7] J. Cong, J. Fang, and K. Khoo,"DUNE: A multilayer gridless routing system with wire plan-ning,"*in Proc. Int. Symp. Physical Design*, Apr. 2000, pp. 12–18.

[8] J. Cong, J. Fang, and K. Khoo,"DUNE - A multilayer gridless routing system,"*IEEE Trans. Computer-Aided Design*, vol. 20, no. 5, pp. 633–647, May. 2001.

[9] M. Sato, J. Sakanaka, and T. Ohtsuki, "A fast line-search method based on a tile plane," *in IEEE Int. Symp. Circuits and Systems*, pp. 588–591, May 1987.

[10] A. Margarino, A. Romano, A. De Gloria, F. Curatelli, and P. Antognetti, "A tile-expansion router,"*IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 507–517, July 1987.

[11] R. Eric Lunow, "A Channelless, Multilayer Router,"*in 25th ACM/IEEE Design Automation Conference*, pp. 667 – 671, 1988.

[12] L. C. Liu, H.-P. Tseng, and C. Sechen, "Chip-level area routing," *in Proc. Int. Symp. Physical Design*, pp. 197–204, Apr. 1998.

[13] C. Tsai, S. Chen, and W. Feng, "An H-V Alternating Router," *IEEE Trans. Computer-Aided Design*, vol. 11, pp. 976–991, Aug. 1992.

[14] J. Dion and L. M. Monier,"Contour: A tile-based gridless router,"Western Research Laboratory, Palo Alto, CA, Research Report 95/3.

[15] Zhaoyun Xing and Russell Kaog, "Shortest Path Search Using Tiles and Piecewise Linear Cost Propagation,"*IEEE Trans. Computer-Aided Design*, vol. 21, no. 2, pp. 145–158, Feb. 2002.

[16] J. K. Ousterhout, "Corner Stitching: Adata-structuring technique for VLSI layout tools," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, pp.87–100, Jan. 1984.

Table 3. Apply Routing Graph Reduction.

Test name	Pure tile-based router			Apply RGR			*T1*(%)	*T2*(%)
	RT (T_a)	WL (W_a)	#Vias (V_a)	RT (T_b)	WL (W_b)	#Vias (V_b)		
TEST1	128.3	12035.07	480	65.8	12035.08	480	48.7	44.2
TEST2	82.6	11553.93	478	41.8	11553.96	478	49.5	42.3
TEST3	73.9	11399.45	394	38.3	11399.48	394	48.2	39.4
TEST4	65.3	9667.94	398	33.1	9667.97	398	49.3	40.3
TEST5	52.2	11194.86	508	26.5	11194.92	508	49.7	38.0
TEST6	121.5	12618.18	498	64.4	12618.21	498	47.0	42.2
TEST7	147.1	11732.99	502	75.6	11733.03	502	48.7	44.7

※*T1* : $(T_a-T_b)/T_a$, *T2* : $(T_a-(T_b+T_{rgr}))/T_a$

※*RT* : routing time(second) , *WL*: wire length(um)

Table 4. Apply ECO Global Routing with Routing Graph Reduction.

Test name	New ECO routing design flow					*T3*(%)	*W* (%)	*V*(%)
	RT (T_c)	WL (W_c)	#Vias (V_c)	#ER	#GCRS			
TEST1	8.56	12497.55	184	0	0	88.7	3.8	-61.6
TEST2	4.62	12607.58	530	1	0	87.2	9.1	10.8
TEST3	4.28	12173.36	536	0	0	86.2	6.7	36.0
TEST4	4.62	10687.03	416	4	4	83.9	10.5	4.5
TEST5	5.29	14437.81	588	4	2	78.5	28.9	15.7
TEST6	5.56	13533.45	516	0	0	90.5	7.2	3.6
TEST7	6.92	13097.15	262	1	0	91.2	11.6	-47.8

※ The global routing partitions the layout to 28x28 GCells and each GCell's width is about 122 pitches.

※ *ER* : Extended Routing, *GCRS*: GCell restructuring and rescheduling

※ *RT* : routing time(second), *WL*: wire length(um), *T3* : $(T_a-(T_c+T_{rgr}))/T_a$, *W* : $(W_c-W_a)/W_a$, *V*: $(V_c-V_a)/V_a$