The Star-Routing Algorithm Based on Manhattan-Diagonal Model for Three Layers Channel Routing

Jiann-Chyi Rau, Po-Han Wu, Chia-Jung Liu, and Yi-Chen Lin
Department of Electrical Engineering, Tamkang University
151 Ying-Chuan Road Tamsui, Taipei County Taiwan 25137, R. O. C.
{jcrau, phwu, cjliu, yclin}@ee.tku.edu.tw

Abstract: – In this paper, we employ gridded model for channel routing and place the terminals which are horizontally aligned. We have developed a two-layer channel router that can eliminate the constraints due to overlap. The proposed approach is suitable for cell/IP-based channel-less circuit with a few channels. Our developed tool can route the nets in nearly linear time achieving to the advantage of time to market, and lead to the area overhead of 6.34% increase in average. The area overhead results from the space insertion, and we also have shown that the proposed algorithm can achieve 100% routing on most ISCAS'85 benchmarks. In addition, the number of channel tracks can be minimized by our algorithm. We proposed the star-routing algorithm for three layers channel routing using Manhattan-Diagonal Model to solve the channel routing problem. We drew up the smaller grid-model, and in order to avoid violating the DRC, so the algorithm has a restriction for the third metal-layer in routing step. We don't need to increasing extra spaces and moving any pins in order to finish the routing completely. The foregoing is good for hard blocks to finish the routing easily, and it does not replace the location of hard blocks because of routing incompletely. Therefore, the height of the entire routing channel can reduce a lot.

Key-Words: Channel routing, Manhattan-Diagonal Model, Electronic Design Automation (EDA)

1. Introduction

Integrated circuit technology has developed in the 1960s from the integration of a few transistors currently in use and the integrated circuits contain more than 105 transistors. Nowadays, the number of transistors in circuits grows rapidly and the circuit design becomes a very complex and difficult task. Due to large number of components, the physical design must be helped with computers. The phases of physical design extensively use computer-aided design tools [1], and many phases have already been partially or fully automated. There are different targets that one would like to optimize. Our work focuses on how to carry out routing completely within reasonable time.

Routability is a key factor of digital integration circuits design and a positive aspect of the former standard cell design style [2] based on routing channels. The standard cell architecture (Fig. 1) consists of rectangular cells with the same height, which are placed in several rows, and the space between two rows being called channel.

In recent year, the advanced semiconductor manufacturing technology provides several layers of Metal to deal with the more and more complex ICs. In nanometer-scale processing, a VLSI chip may contain several million transistors. As a result, it is highly probable that tens of millions of connecting nets have to be routed completely and successfully in the layout step. The problem of the routing becomes more and more complicated. Our work focuses on how to carry out routing completely without adding extra space and moving any terminals.

The routing problem consists of interconnecting the cells that have been assigned positions as a solution of the placement problem. The specification of a routing problem will consist of the position of the terminals, the netlist that indicates which terminals should be interconnected and the area available for routing in each layer.

Routing is normally performed in two stages. The first stage is called global routing [3], determines each net will be connected through which wiring channels. The second stage is called detailed routing [4], finds the precise paths and the actual usage of metals. It is one of the most fundamental steps in physical design cycle. The channel routing is the important part of detailed routing, which is executed after placement.

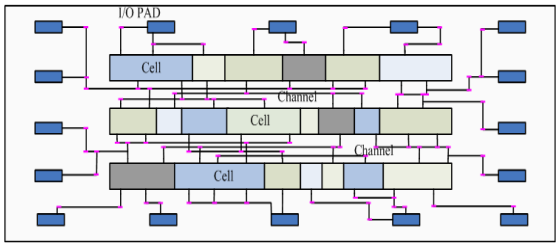


Fig. 1. Standard cell architecture.

2. Preliminary

2.1 Channel Routing Model

The channel routing model [5] (Fig. 2) consists of several parts of components. The nets of terminals are placed on upper and lower boundaries. The channel to be routed is defined by a rectangular region with two rows of terminals along its top and bottom sides. The horizontal metal layer is called trunk and the vertical metal layer is called branch. When horizontal metal layer and vertical metal layer are needed to connect and we must penetrate two layers with via. Channel routing model combine with grid-based model that make the physical design easier and let the design time shorter.

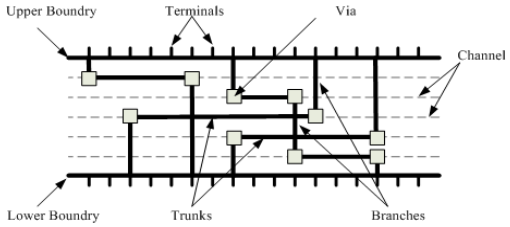


Fig. 2. Channel routing model.

2.2 Channel Routing Problem

The main target of the channel routing is to minimize the channel height [6][7]. In real designs, each channel is assigned fixed height by the floorplanner and the task of channel router is to complete the routing within the assigned height. If channel router can not achieve 100% routing in the assigned height, channel router must to expand the channel, which changes the floorplan. This requires routing the channels in a predefined order, so that such expansions can be accommodated, without impact on the floorplan.

2.3 Grid Model

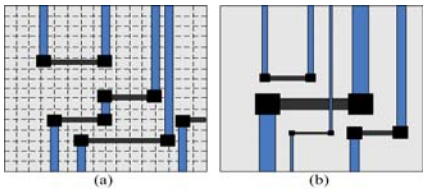


Fig. 3. (a) Grid based model. (b) Grid-less based model.

Routing model can divide into two models. The first model is called grid based model (Fig. 3(a)). There are designed rules between each grid when route in the grid based model just placed the cells and terminals in the grids. The second model is called grid-less based model [8] (Fig. 3(b)). Using grid-less based model to design circuits is very flexible. But the drawbacks of using grid-less based model which leads designed rules checking more complex and used more memory to record the data structure of routing.

Left-Edge Algorithm

```
00 BEGIN
01   Sort all nets on their leftmost positions;
02   Select the next net n with the lowest left-end position;
03   IF (n has no descendents in VCG)
04     Place n on the lowermost available track;
05     Delete n from the sorted list;
06     Delete n form VCG;
07   ELSE
08     Goto 02
09   END IF
10   Continue scanning the sorted list and form it select those nets
     which do not overlap with nets assigned to this track and have
     no descendents in VCG;
11   Remove all selected nets form the list;
12   IF list ≠ Ø then Goto 02;
13   EXIT
14 END
```

Fig. 4. Left-edge algorithm.

2.4 Left-Edge Algorithm

The left-edge algorithm [9] is applicable to channel routing problems which do not allow doglegs [10] and vertical constraints. In other words, it does not appear cycles in vertical constraint graph [5][9]. The left-edge algorithm sorts the intervals that are the horizontal line segments of the nets, in ascending order, relative to the X coordinate of the left end point of intervals. And then it allocates a track to each of the intervals. To allocate an interval to a track, left-edge will scan through the tracks from the top to the bottom and assigns the net to the first track that can adapt the net. The allocation process is restricted to one layer since the other layer is used for the vertical line segments of the nets. Fig. 4 shows the left-edge algorithm.

2.5 Manhattan-Diagonal Model

Fig. 5(a) shows the Manhattan model that it was used to solve the channel routing problem in the past thirty decades [11, 12]. It could make the placement and routing obviously simple, but it did not benefit for two points that the relationship between them is diagonal. In other words, it only could be routed the horizontal and vertical directions. Recently, the Manhattan-Diagonal (MD) model (Fig. 5(b)) was proposed in [13], and it includes not only the horizontal and vertical directions but also the positive and minus 45 degrees directions. In a few years, many researches of routing are used the Manhattan-Diagonal model. A new greedy non-Manhattan channel routing algorithm using the MD model was proposed in [14]. The non-overlap MD model and overlap MD model are proposed in [15]. Ho et al. [16] are proposed a multilevel full chip routing for the X-based

architecture. The X-architecture has been proposed for interconnect delay. It can not only reduce the wire length and via count but also improve performance and routability for chip. In 2000, Chen et al. [17] proposed an efficient bubble-sort technique to solve the channel routing problem. In this paper, we use the MD Model to solve the problem.

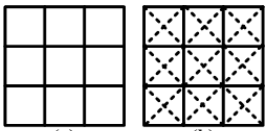


Fig. 5. (a) Manhattan model. (b) Manhattan-Diagonal model.

2.6 Grid Size

In the past, the grid size of using two layers to route the Manhattan model [11, 12] shows in Fig. 6(a), and the value is 0.66. In other researches of using MD model [14-17] for routing, we analyzed the grid size of these. We found out grid size for the most of these papers is bigger than the grid size of our proposed algorithm because of it must be conformed to the Design-Rule Check (DRC). Fig. 6(b) shows the grid size of Metal-2 or Metal-3 for MD model, and the value is 0.975. The diagonal line segment is Metal-2 based on the manufacture of 0.18 μm, and the upper left, upper right, lower left and lower right grid are the contacts between Metal-1 and Metal-2.

The grid size of our proposed algorithm shows in the Fig. 6(c), and the value is 0.66 because we drew up the special rule for Metal-3 to route the diagonal line segment. In regular situation, we use the Metal-1 to route the vertical line segment and Metal-2 to route the horizontal line segment. The Metal-3 is used to route the diagonal line segment. If the Metal-3 is used to route from the upper right grid to the lower left grid, we drew up that the upper left and lower right grid are not used the Metal-3 for other terminal and be restricted for this terminal. Thus our grid size could be smaller than others. Supposing the number of routing track is bigger than other's, but our routing height after conversion is still smaller than their because of we use the smaller grid size.

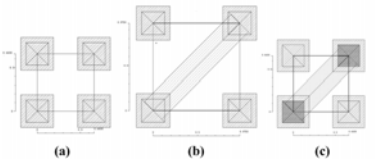


Fig. 6. Three kinds of grid model.

2.7 Designation

The name of our proposed algorithm is called star-routing. Fig. 7 shows the example of star-routing. If the terminals are situated at the upper left, lower left, upper right, lower right, above and below of the center, the routing result of our algorithm is like a star shape. The upper left, lower left, upper right and lower right terminals are used the Metal-3 to route the diagonal line, and the terminals are located at above and below of center and used the Metal-1 to route the vertical line.

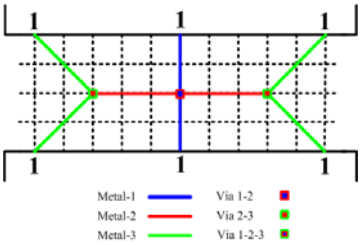


Fig. 7. The star shape of our star-routing algorithm.

3. The Extensible Channel Router

The terminals of cells are needed to place before routing. Therefore, beginning of routing we must know the cells that are placed in certain positions. In our work, we placed the terminals of cells with two rows and used the region between the two rows to route the nets of circuit.

We use a simple placement algorithm [18] to place the terminals of cells. First, we placed the input pins of circuit on the left upper row and placed the output pins of circuit on right lower row possibility.

3.1 Horizontal Constraint Graph

After placement the terminals of circuit, we can know the positions of each terminal. Therefore, we can compute every horizontal line segments. The first step, we will find the leading terminal of each net. Then, we will find the last terminal of each net. And we can record the two data of each net and store it in the data structure. We will take the data structure to operate the following steps.

According to the data structure, the horizontal constraint graph will be set up by the information of each net. Fig. 8 is an example of horizontal constraint graph [5][9]. We do not consider the order of the horizontal line segments. Therefore, we can set up the horizontal constraint graph using undirected graph.

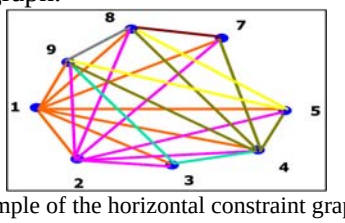


Fig. 8. Example of the horizontal constraint graph (HCG).

3.2 Vertical Constraint Graph

In order to avoid the vertical constraints and we will set up the vertical constraint graph before routing. Consider the terminals which include upper and lower rows, we can regard them as many pairs of columns. The zero numbers represent the space of the placement. We will find the upper and lower terminals of one pair that are not contained zero numbers. Then, we will consider these pairs of terminals to set up the vertical constraint graph.

After finding these terminals, we will set up the vertical constraint graph (Fig. 9). We need to use directed graph to represent the vertical constraint graph. Because of the upper terminals must be above the lower terminals. We use the directed graph to represent the vertical constraint graph different from horizontal constraint graph.

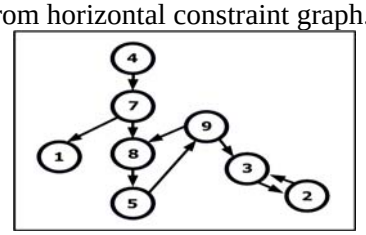


Fig. 9. Example of the vertical constraint graph (VCG).

3.3 The Extensible Router

After left-edge algorithm and have been set up the constraint graphs we will get the preliminary routing result which is including the violations of vertical constraints. We will look table up in vertical constraint graph and try to break the cycles in the vertical graph.

3.3.1 Insertion of Spaces

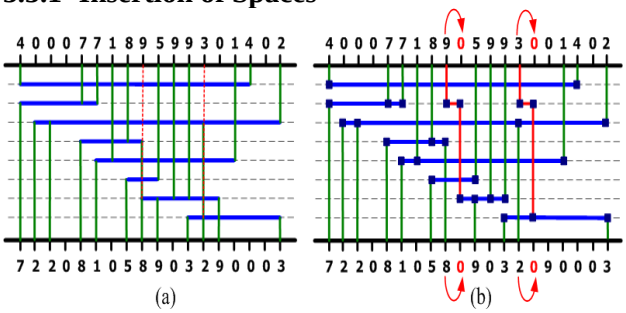


Fig. 10. (a) Preliminary routing with the vertical constraints. (b) Solution of vertical constraint violations.

Now, we face the problem that is violations of vertical constraints. We will insert pairs of spaces near by the terminals which are violations of vertical constraints. Then we will try to find the remnant tracks to route the un-routed terminals. If there are not the remnant tracks to route the terminals, we will add extra track in the most top of the channel to route the un-routed terminals. Fig. 10(a) is an example of preliminary routing result with vertical constraints. There are two violations of vertical constraints in the example. Now, we face the problem that is violations of vertical constraints. In Fig. 10(b), we will insert pairs of spaces near by the terminals which are violations of vertical constraints. Take Fig. 10(b) for example, we will find the second track to route the two terminals which are violations of vertical constraints.

3.3.2 Update the Data Structure

After insert the spaces, we need to update the data structures, which are the sites of terminals and horizontal line segments. The first, we must to recompute the sites of terminals and according to these sites to change the data structure of horizontal line segments. Now, we get a new solution of routing and it can achieve 100% routing [19]. Fig. 11 shows the insert space algorithm.

Insert Space Algorithm

```
00 BEGIN
01   Find all the vertical constraints from VCG;
02   Record the terminals which are violated vertical constraint;
03   FOR (i = 0 to i = count of terminals - 1) Do
04     Insert pair of spaces near by the terminal;
05     Update the site of terminal;
06     IF (Exist redundant tracks to route the un-routed terminals)
07       Re-route the terminal on the redundant;
08       Update the date structure of tracks;
09     ELSE
10       Insert a new track at top;
11       Re-route the un-routed terminal at top track;
12       Update the date structure of tracks;
13     END IF
14   END FOR
15 END
```

Fig. 11 Insert space algorithm.

3.4 Channel tracks optimization

In our experiments, we will divide the condition into two cases. One case is end of the upper line segment corresponding with head of the lower line segment and the other case which is head of the upper line segment corresponding with end of the lower line segment. Fig. 12(a) shows the two cases. Before optimizing the segments, we must to know data structure of each horizontal line segments that record coordinates of all segments. Then our algorithm will find the two cases to reduce the channel tracks. Fig. 12(b) shows the horizontal line segments after optimizing and then we reduce the channel tracks from two to one.

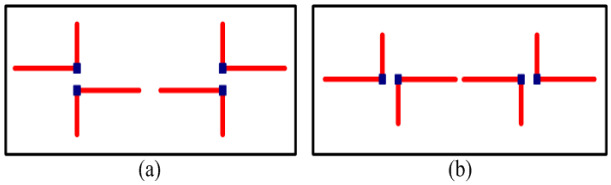
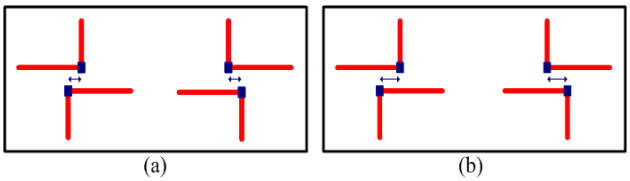


Fig. 12 (a) Two cases that we will optimize.
(b) After optimizing the horizontal line segments.



(a) (b)

Fig. 13. The extended cases (a) Overlap one contact. (b) Overlap two contacts.

Further, we will extend more cases to reduce the channel tracks. The two cases that are the upper horizontal line segment overlapped the lower horizontal line segment, if the two line segments in the same channel track. The Fig. 12 shows the extension of the idea. Fig. 13(a) is more one contact than original case and Fig. 13(b) is more two contacts than original case.

In the second and third cases, we will insert two and three pairs of zero terminals to shift the horizontal line segment respectively.

4. Star-Routing Algorithm

Fig. 14 shows the star-routing algorithm flowchart that we proposed, and it divides into eight steps.

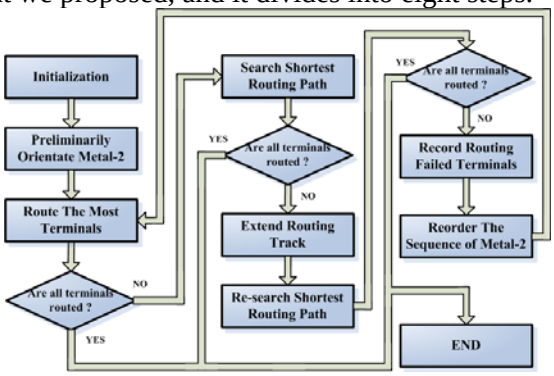


Fig. 14. Star-routing algorithm flowchart.

4.1 Initialization

At first we placed the terminals of cells or modules with two rows that meant up and down rows such as Fig. 15. It is also the example in [14] and [15]. We can know the sequence and correlation of that. We can compute the start location, second start location, the end location and second end location of every horizontal line segments and record it. The recording value is the vertical location referred to the up or down row. The second start location is the second vertical location referred to the up or down row from the start location. We place the horizontal line segments according to that.

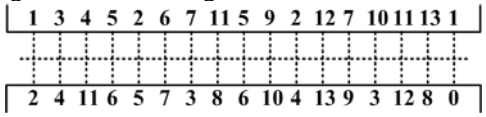


Fig. 15. An example of the channel routing.

4.2 Preliminary Orientate Metal-2

Algorithm of Orientating the Metal-2

```
/*Node_count is the count of the N
  track_count_tmp is the count of the track, the value is temporarily equal to the value of Node_count
  N = {N0, N1, N2, …, NNode_count} //N is the set of trunk nodes */
01  FOR( n = 0 to n = Node_count )
02    FOR( i = 0 to i = track_count_tmp)
03      IF ( B_end is occupied by another trunk at i ) CONTINUE;
04      ELSE
05        IF ( From B_start to B_end is not occupied by other trunk at i ) Place B at the track i;
06        ELSE
07          IF ( The overlap between A and B ≤ i & the length of B > i )
08            IF ( A_end is situated at down row & B_start is situated at down row ) CONTINUE;
09            ELSE IF ( A_end is situated at up row & B_start is situated at up row )
10              CASE ( Are A_second_end and B_second_start in the overlap between A and B ? )
11                A ( YES ) , B ( YES ) : CONTINUE;
12                A ( YES ) , B ( NO ) :
13                  IF ( The terminal at the right side of B_start belongs to B ) CONTINUE;
14                  ELSE A = Upwards Vertical Metal-1 at i
                        AND B = -45˚Metal-3 at i & update B_start ;
15                  END IF //L-13
16                A ( NO ) , B ( YES ) :
17                  IF ( The terminal at the left side of A_end belongs to A ) CONTINUE;
18                  ELSE A = 45˚Metal-3 at i & update A_end
                        AND B = Upwards Vertical Metal-1 at i ;
19                  END IF //L-17
20                A ( NO ) , B ( NO ) :
21                  CASE ( Does the terminal at the left side of A_end belong to A ?
                           Does the terminal at the right side of B_start belong to B ? )
22                    A ( YES ) , B ( YES ) : CONTINUE;
23                    A ( NO ) , B ( YES ) : A = 45˚Metal-3 at i & update A_end
                             AND B = Upwards Vertical Metal-1 at i ;
24                    A ( YES ) , B (NO ) : A = Upwards Vertical Metal-1 at i
                             AND B = -45˚Metal-3 at i & update B_start ;
25                    A ( NO ) , B ( NO ) : A = Upwards Vertical Metal-1 at i
                             AND B = -45˚Metal-3 at i & update B_start;
26                  END CASE //L-21
27                END CASE //L-10
28              ELSE IF ( A_end is situated at up row & B_start is situated at down row )
29                IF ( A_second_end is in the overlap between A and B ) CONTINUE;
30                ELSE
31                  IF ( The terminal at the left side of A_end belongs to A ) CONTINUE;
32                  ELSE A = 45˚Metal-3 at i & update A_end
                        AND B = Downwards Vertical Metal-1 at i ;
33                  END IF //L-31
34                END IF //L-29
35              ELSE IF ( A_end is situated at down row & B_start is situated at up row )
36                IF ( B_second_start is in the overlap between A and B ) CONTINUE;
37                ELSE
38                  IF ( The terminal at the right side of B_start belongs to B ) CONTINUE;
39                  ELSE A = Downwards Vertical Metal-1 at i
                        AND B = -45˚Metal-3 at i & update B_start ;
40                  END IF //L-38
41                END IF //L-36
42              END IF //L-08
43            END IF //L-07
44          END IF  //L-05
45        END IF //L-03
46      END FOR //L-02
47  END FOR //L-01
```

Fig. 16. Algorithm of orientating the Metal-2.

After computing the correlation of the nodes, we can get the all horizontal line segments (also called trunks) including the start, second start, end and second end location. Afterwards we will follow the sequence of the horizontal line segments. It interlaced by up and down side from the right to left side. We find the suitable routing track from up to down for the every horizontal line segments. We define the symbols in this step as follows, and Fig. 16 shows the algorithm of orientating the Metal-2.

i: the number of the routing track

i+1: the number of the next routing track

A: the horizontal line segment which had placed at the routing track (***i***)

B: the horizontal line segment that has to find the suitable routing track

A_start or ***A_end***: the start (or end) location of ***A***, and the information of that the terminal is situated at the up or down row

B_start or ***B_end***: similar to ***A_start***

At first, we check the Metal-2 of the ***B_end*** location at track ***i*** is occupied by another horizontal line segment or not. If ***B_end*** is occupied by another trunk at ***i***, we will check the next routing track (Fig. 17(a)). If the nodes from ***B_start*** to ***B_end*** are not occupied by other trunk at ***i***, we place ***B*** at the track i (Fig. 17(b)). In addition to the foregoing cases, the value subtracting the ***A_end*** location from the ***B_start*** location is smaller than or equal to the value i, and the horizontal length of ***B*** must be greater than the value ***i***. If the ***B*** is conformed to the foregoing rules, we will consider more situations below, or check the routing track ***i+1***. Supposing the terminals referred to the ***A_end*** location and ***B_start*** location both belong to the down row, we will check the routing track ***i+1*** because of our proposed algorithm orientates the Metal-2 from up to down side of the routing region, and therefore we cannot predict that how many routing tracks that used by other horizontal line segments. The B consequently cannot place at the track i (Fig. 17(c)).

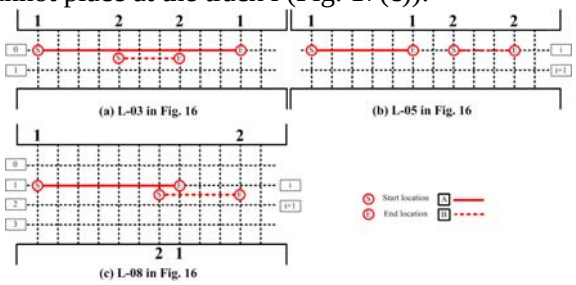


Fig. 17. Illustration I of Fig. 16.

If the terminals referred to the ***A_end*** location and ***B_start*** location both belong to the up row, we check the overlap between ***A*** and ***B*** that other terminals belonging to ***A*** or ***B*** situated at the down row and check the terminal at the right (left) side of the ***B_start*** (***A_end***) belongs to ***B*** (***A***) or not. Fig. 18 shows the ***B*** cannot place at track i. Fig. 18(a), (b) and (c) will connect the signal ***A*** with the signal ***B***, and the Fig. 18(d) will violate the special Metal-3 rule that we established. Fig. 19 shows the correct routing result using our algorithm. And the other situations are all similar to the foregoing.

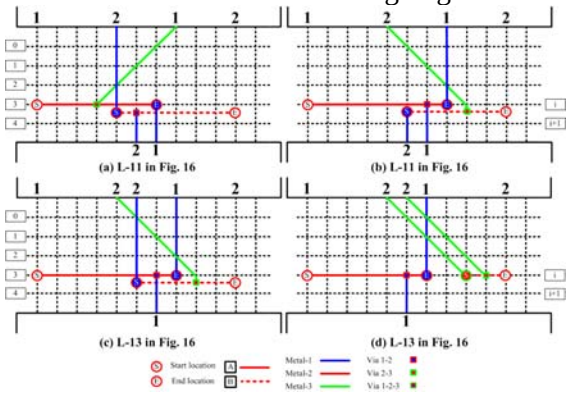


Fig. 18. Illustration II of Fig. 16.

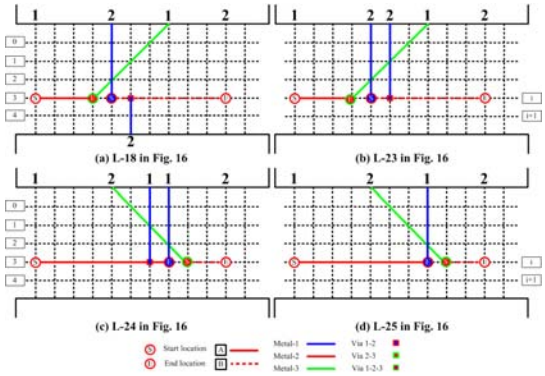


Fig. 19. Illustration III of Fig. 16.

If we use the Metal-3 to route the diagonal line, we will first check that the diagonal line form the terminal to the absolute position can be routed or not. Fig. 20(a) shows that it can be routed by the Metal-3 to route the diagonal line. If it cannot be routed, we will analyze the vertical line of the terminal and check that can be routed by the collocation of the Metal-1 to route the vertical line and the Metal-3 to route the diagonal line (Fig. 20(b)).

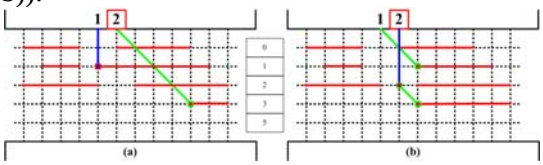


Fig. 20.The rule of Metal-3 to route the diagonal line.

4.3 Route The Most Terminals

We receive the placed track of every horizontal line segment after the algorithm of orientating the initial Metal-2. Afterwards we follow the sequence of the horizontal line segments. It interlaced by up and down rows from the right to left side. We enable the unconnected terminals to route suitably to which horizontal line segment. If terminal is referred to the start or end location in which horizontal line, we first consider using the Metal-3 to route the diagonal line. Supposing it cannot be routed, we check it can use Metal-1 to route the vertical line or not. Fig. 21(a) shows that the routing result is violating the special Metal-3 rule drawing up by us. Fig. 21(b) shows that the routing error for connecting the two signals. Fig. 21(c) shows that routing success of using Metal-1. Fig. 21(d) is the routing failure of using Metal-1 for connecting the two vertical Metal-1 lines. We classify the kind of terminals belonging to Fig. 21(a) or (b) or (d) into the preliminary un-routed set. Fig. 22 is the routing result of Fig. 15, and we can notice the four terminals without using Metal-3 to route the diagonal line or Metal-1 to route the vertical line. In next step, we solve the preliminary un-routed set by another routing method.

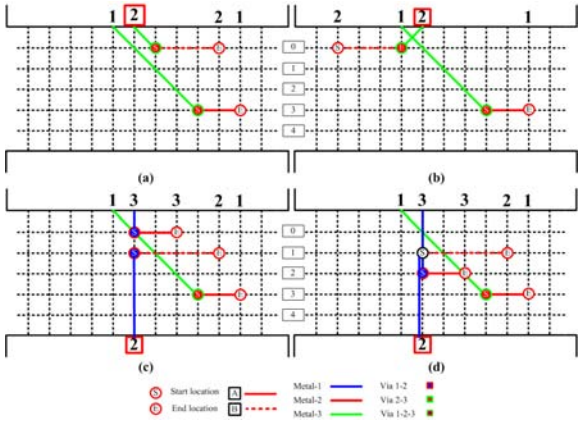


Fig. 21. Successful and failed cases of routing the most terminals.

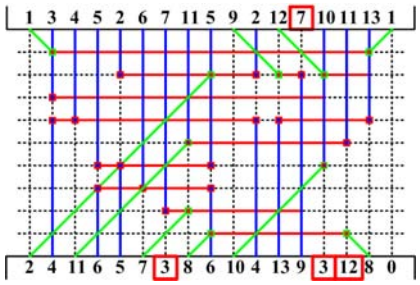


Fig. 22. The routing result of the most terminals for Fig. 15.

4.4 Search Shortest And Multilayer Routing Path

We receive the preliminary un-routed terminal set after above steps, so that we aim at the set to individually find the shortest and multilayer routing path. We use the single linked list to store the shortest and multilayer routing path. We define the symbols in this step as follows.

GA: the grid node has already confirmed to route for this terminal

GB: the grid node is checking at this moment for this terminal

GA(1) or ***GB***(1): the grid node purely used the Metal-1

GA(2) or ***GB***(2): the grid node purely used the Metal-2

GA(3) or ***GB***(3): the grid node purely used the Metal-3

GA(12) or ***GB***(12): the grid node used the Metal-1 And Metal-2, and also used via between Metal-1 and Metal-2. Besides, Metal-2 is the basis for searching the next grid node

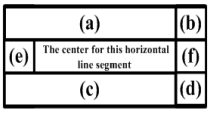
GA(21) or ***GB***(21): similar to ***GA***(12), but Metal-1 is the basis for searching the next grid node

GA(23) or ***GB***(23): the grid node used the Metal-2 and Metal-3, and also used via between Metal-2 and Metal-3. Besides, Metal-3 is the basis for searching the next grid node

GA(32) or ***GB***(32): similar to ***GA***(23), but Metal-2 is the basis for searching the next grid node

GA(13) or ***GB***(13): the grid node used the Metal-1, Metal-2 and Metal-3, and also used two kinds of via. One is between Metal-1 and Metal-2, and another is between Metal-2 and Metal-3. Besides, Metal-3 is the basis for searching the next grid node

GA(31) or ***GB***(31): similar to ***GA***(13), but Metal-1 is the basis for searching the next grid node



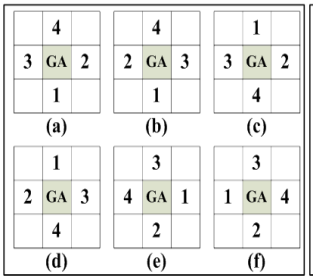


Fig. 23. Searching basis is Metal-1 or Metal-2.

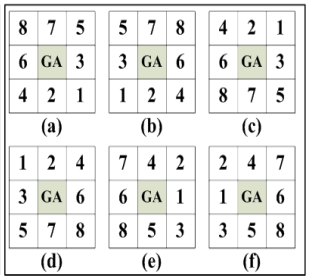


Fig. 24 Searching basis is Metal-3.

For every terminal of preliminary un-routed set, we search the routable path from the coordinate of this terminal, and therefore the coordinate (***GA***) is the first confirmed grid node for the shortest routing path. At first, we record the center of the horizontal line belonging to the preliminary un-routed terminal. Search the next routable and shortest grid node (***GB***) according to relative position for the recorded center and the preceding routing grid node (***GA***). The searching sequence of the routable grid node (***GB***) shows below.

Fig. 23 shows the searching basis of the confirmed grid node (***GA***) is Metal-1 or Metal-2, and Fig. 24 shows the searching basis is Metal-3. If grid node GA is above the track, and it is at the portside or over the center for horizontal line segment belonging to this terminal. The searching sequence: down → right → left → up (Fig. 23(a)). The searching sequences of the other cases show in Fig. 13 and Fig. 14.

If the grid node ***GB*** is checking at this moment for this terminal, and searching basis of grid node ***GA*** is Metal-1 (Fig. 25). At first, we check the Metal-1 of grid node ***GB*** that is occupied by another terminal or not. Supposing the Metal-1 of grid node ***GB*** is occupied by another terminal, we check the Metal-2 of grid node ***GB***, and then we check the Metal-3 of grid node ***GB*** that is occupied or restricted by another terminal. If the searching basis is Metal-2, the checking sequence of ***GB*** is : Metal-2 → Metal-

3 → Metal-1. Because the probability of Metal-1 and Metal-2 are occupied by other terminal is greater than Metal-3 for the most grid node. If the searching basis is Metal-3, the checking sequence of ***GB*** is : Metal-3 → Metal-2 → Metal-1.

We can finish routing the most terminals of preliminary un-routed set through the above searching rules. Fig. 26 shows the routing result of searching shortest and multilayer routing path of the Fig. 15.

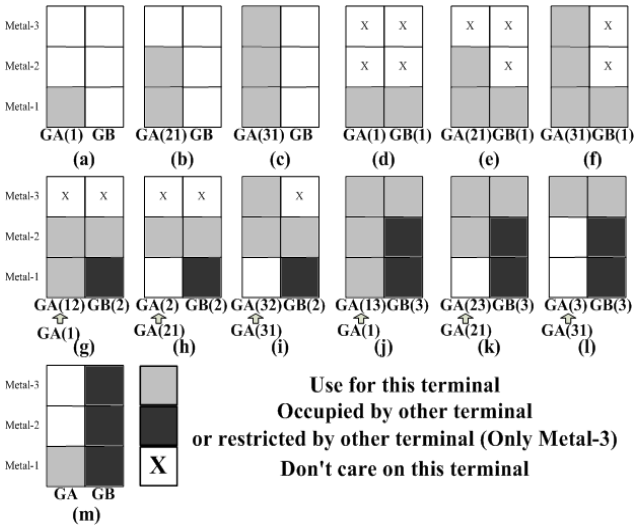


Fig. 25. Searching basis of Metal-1 for ***GA***.

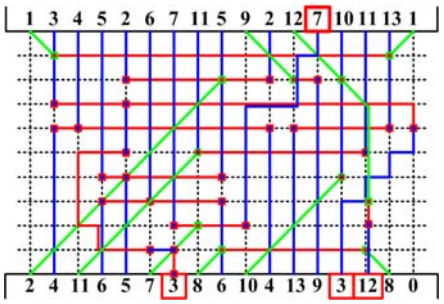


Fig. 26. The routing result of Fig. 15.

4.5 Extend Routing Track

The most terminals of preliminary un-routed set through the above searching shortest and multilayer routing path can find the routing path. Only a few terminals can not find the routing path after the above searching shortest and multilayer routing path. In this step, we analyze that the terminals belong to the up or down row or both up and down rows. If the terminals belong to the up row (Fig. 27(a)), we add a routing track at the up side of the routing region. Fig. 27(b) shows the extending way of Metal. After extending the routing tracks, we can research the shortest and multilayer routing path for this terminal. This extending way of Metal will not make the error of other already routed terminals and have more choices of routing path for the few terminals that can not find the routing path. It also can keep the diagonal routing path of Metal-3 for other terminals.

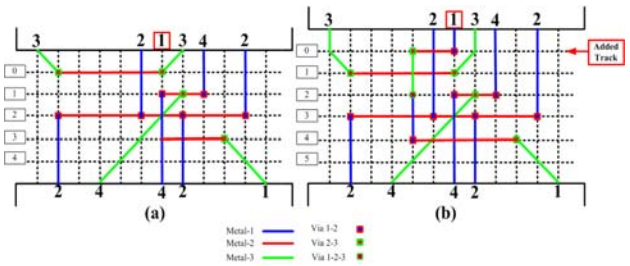


Fig. 27. The extending sample of Metal.

4.6 Reorder the Sequence of Metal-2

After extending the routing tracks, we research the shortest and multilayer routing path. Supposing the terminals still route abortively, we record it and go back to the first step that reorder the sequence of Metal-2. At first, we place the horizontal line segments of the recorded terminals at the suitable routing tracks. The step can reorder the sequence of all routing tracks and the recorded terminals can be routed completely at first, and therefore the whole terminal in the channel can be routed successfully.

5. Experimental Results

We implement our algorithm in C language and run it on Intel Pentium 4 2.4 GHz CPU with 1 GB RAM, Windows XP operation system.

The extensible router was compared to the ChAOS router [20]. We performed our experiments on some ISCAS 85 benchmarks, which are combinatorial logic circuits.

We can observe the sixth column of Table I, the extra terminals that we need to add to solve the vertical constraints. Extensible router is better than ChAOS router that we can reduce the area overhead and the two routers utilized the same channel tracks. The percentages of area overhead are compared with the original terminal numbers. We gather statistics the difference of area overhead between the two routers in the last column of Table I. Our approach can reduce 3.34% in average of the total area than ChAOS router.

Table II shows the extra terminals that we need to add after horizontal segment optimization. Besides, the Table II also shows the channel tracks that we need to route the circuits. In several cases we can reduce the channel tracks effectively and the results are showed in each case of tracks column.

Table III shows the comparison routing result with [12] and our star-routing algorithm. In this test case,

the grid size is the same for [12] and our star-routing algorithm. Therefore we compare the track number of two results. In this test set, the percentage of our and other's routing result is approximately 41% to 72%. It shows the very huge advancement using our star-routing algorithm.

Table II shows the routing result of [14], [15], the Example Case (Fig. 15) and our star-routing algorithm. We analyze the grid size of every routing example and compare with our results. In the most test cases, our routing track number is smaller than track number of the other's result. The track number of YK3c is specially grater than other's routing result. But, we compute the height of two results, and therefore we can see that the height of our routing result is still less than the other's height because our used grid size is smaller than other's grid size. In this test set, the percentage of our and other's routing result is approximately 21% to 38%.

TABLE I.
Comparison of area overhead.

Circuit		ChAOS Router [20]			Our Router			Difference
Name	Pairs of Terminals	Add Extra Terminals	Utilize Track	Area Overhead (%)	Add Extra Terminals	Utilize Track	Area Overhead (%)	in Overhead Between Two Router
c17	9	2	11	22.22%	1	11	11.11%	(-)11.11%
c432	253	28	196	11.06%	21	196	8.3%	(-)2.76%
c499	300	40	243	13.33%	29	243	9.66%	(-)3.67%
c880	514	60	441	11.67%	45	441	8.75%	(-)2.92%
c1355	802	106	587	13.20%	72	587	8.97%	(-)4.23%
c1908	1189	73	912	6.13%	51	912	4.28%	(-)1.85%
c3540	2140	88	1717	4.11%	60	1718	2.8%	(-)1.31%
c5315	2929	160	2453	5.40%	92	2453	3.14%	(-)2.26%
c6288	3608	3	2447	0.083%	2	2447	0.05%	(-)0.033%

TABLE II.
The Results of Horizontal segment optimization.

Circuit	Without Horizontal Segment Optimization		Horizontal Segment Optimization								
			Case 1			Case 2			Case 3		
Name	Terminals	Track	Add Extra Terminals	Track	Reduction of Track (%)	Add Extra Terminals	Track	Reduction of Track (%)	Add Extra Terminals	Track	Reduction of Track (%)
c17	10	11	1	11	0%	1	11	0%	14	7	(-)36.36%
c432	274	196	18	196	0%	34	196	0%	51	196	0%
c499	329	243	14	243	0%	66	243	0%	132	239	(-)1.64%
c880	559	441	40	440	(-)0.22%	80	440	(-)0.22%	189	422	(-)4.3%
c1355	874	587	14	586	(-)0.17%	133	586	(-)0.17%	295	577	(-)1.7%
c1908	1240	912	33	912	0%	121	912	0%	186	912	0%
c3540	2200	1718	38	1718	0%	138	1718	0%	399	1640	(-)4.54%
c5315	3021	2453	60	2452	(-)0.04%	201	2452	(-)0.04%	595	2349	(-)4.23%
c6288	3610	2447	1	2447	0%	5	2447	0%	5	2447	0%

TABLE III.
Comparison of [12] and Star-routing.

Example	Grid Size	[12]		Our		Difference
		Metal Layer	Routing Track	Metal Layer	Routing Track	Percentage (%)
c17	0.66	2	11	3	5	(-)54.55%
c432	0.66	2	196	3	64	(-)67.35%
c499	0.66	2	243	3	119	(-)51.03%
c880	0.66	2	441	3	259	(-)41.27%
c1355	0.66	2	587	3	198	(-)66.27%
c1908	0.66	2	912	3	251	(-)72.48%

TABLE IV.
Comparison of [14], [15] and Star-routing.

Example	[14,15]				Our				Difference
	Metal Layer	Grid Size	Routing Track	Height	Metal Layer	Grid Size	Routing Track	Height	Percentage (%)
YK3a	2	0.975	15	15.6	3	0.66	14	9.9	(-)36.54%
YK3b	2	0.975	17	17.55	3	0.66	17	11.88	(-)32.31%
YK3c	2	0.975	18	18.525	3	0.66	21	14.52	(-)21.62%
DDE	2	0.975	19	19.5	3	0.66	19	13.2	(-)32.31%
Fig. 15	2	0.975	10	10.725	3	0.66	9	6.6	(-)38.46%

6. Conclusions

In recent years, the advance of COMS technology has led to a great development, especially on the complexity of the digital circuits. In this work, we have investigated the grid-based model and then used this model to solve the channel routing problem. The results show that our router can achieve 100% routing and reduce a lot of routing area than before.

We used the grid-model to deal with the routing problem. The advantage of using grid-model needs not too complex data structure to solve it, and making the complexity of routing lower. Besides, we used the three metal-layers to be the major components of routing. We also used the positive and negative 45 degrees routing path, and therefore reduced the routing path and length. In our proposed algorithm, we drew up the smaller grid-model, and in order to avoid violating the Design Rule Check (DRC), so the algorithm has a restriction for the third metal-layer in routing step. Although we use three metal-layers in our algorithm, this way can not only reduce the signal length but also decrease the antenna effect between signals than the other multilayer routing algorithm. We don't need to increasing the extra spaces and moving any pins to finish the routing completely. The foregoing is good for hard blocks to finish the routing easily, and it does not replace the location of hard blocks because of routing incompletely. Therefore, the height of the entire routing channel can reduce a lot.

References:

[1] M. J. Bass and C. M. Christensen, "The Future of The Microprocessor Business," *IEEE Spectrum*, Vol. 39, No. 4, pp. 34-39, Apr. 2002.

[2] M. E. Daniel and C. W. Gwyn, "CAD Systems for IC Design," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 1, No. 1, pp. 2-12, Jan. 1982.

[3] J. Soukup, "Global Router," *in Proc. 16th Design Automation Conf.*, pp. 481-484, Jun. 1979.

[4] E.S. Kuh and T. Ohtsuki, "Recent Advances in VLSI Layout," *in Proc. IEEE*, Vol. 78, No. 2, pp. 237-263, Feb. 1990.

[5] H.W. Leong and C.L. Liu, "Discretionary Channel Routing," *IEE Proc. Circuits, Devices, and Systems*, Vol. 135, No. 2, pp. 45-57, Apr. 1988.
[6] Y. Cai and D.F. Wong, "Optimal Via-Shifting in Channel Compaction," *in Proc. European Design Automation Conference*, pp. 186-190, Mar. 1990.
[7] U. Choudhury and A. Sangiovanni-Vincentelli, "Constraint-Based Channel Routing for Analog and Mixed Analog/Digital Circuits," *in Proc. Int. Conf. Computer-Aided Design*, pp. 198-201, Nov. 1990.
[8] H.H. Chen and E.S. Kuh, "Glitter: A Gridless Variable-Width Channel Router," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 5, No. 4, pp. 459-465, Oct. 1986.
[9] T. Yoshimura, "An Efficient Channel Router," *in Proc. 21st Design Automation Conf.* , pp. 38-44, Jun. 1984.
[10] M. M. Wada, "A Dogleg Optimal Channel Router With Completion Enhancements," *in Proc. 18th Design Automation Conf.*, pp. 762-768, Jun. 1981.
[11] T. Yoshimura and E.S. Kuh, "Efficient Algorithms for Channel Routing," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 1, No. 1, pp. 25-35, Jan. 1982.
[12] C. J. Liu, Y. C. Lin and J. C. Rau, "The Grid-Based Two-Layer Routing Algorithm Suitable for Cell/IP-Based Circuit Design," *to appear in Proc. IEEE Int. Conf. Electronics, Circuits and Systems*, 2008.
[13] K. Chaudhary and P. Robinson, "Channel Routing by Sorting," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, Vol. 10, No. 6, pp. 754-760, Jun. 1991.
[14] D. Lou, Y. Li, X. Zhang and J. Yu, "A Routing Algorithm for Irregular Channel Based on MD Model," *in Proc. Int. Conf. Communications, Circuits and Systems*, Vol. 2, pp. 1263-1266, May 2005.
[15] S. Das and B. Bhattacharya, "Channel routing in Manhattan-diagonal model," *in Proc. 9th Int. Conf. VLSI Design*, pp. 43-48, Jan. 1996.
[16] T. Y. Ho, C. F. Chang, Y. W. Chang and S. J. Chen, "Multilevel Full-Chip Routing for the X-Based Architecture," *in Proc. 42nd Design Automation Conf.*, pp. 597-602, Jun. 2005.
[17] S. S. Chen, C.H. Yang and S. J. Chen, "Bubble-Sort Approach to Channel Routing," *IEE Proc. Computers and Digital Techniques*, Vol. 147, No. 6, pp. 415-422, Nov. 2000.
[18] C. P. Hsu, "General River Routing Algorithm," *in Proc. 20th Design Automation Conf.*, pp. 578-583, Jun. 1983.
[19] R. L. Rivest and C. M. Fiduccia, "A Greedy Channel Router," *in Proc. 19th Design Automation*, pp. 418-424, Jun. 1982.
[20] G. B. V. dos Santos, M. de Oliveira Johann, and R. A. da Luz Reis, "Channel based routing in channel-less circuits," *in Proc. Int. Symposium on Circuits and Systems*, pp. 4, May 2006.