5th International Conference on VLSI and CAD
October 13 - 15, 1997, Seoul, Korea

CD-P06

A DOUBLE-TRACK GREEDY CHANNEL ROUTING ALGORITHM

C.S.K.Yeung, C.K. Yuen, P.Y.S.Cheung, S.K. Tse and W.L. Ko
Department of Electrical and Electronic Engineering, The University of Hong Kong
6/F CYC Bldg., EEE, The University of Hong Kong, Pokfulam Road, Hong Kong
csky@eee.hku.hk, ckyuen@eee.hku.hk, cheung@eee.hku.hk
Fax no.: (852)25598783

Abstract

This paper presents a new VLSI channel routing algorithm which completes the topmost and the bottommost tracks at a time based on the two-layer restricted manhattan model. By doing so, we attempt to generate better solutions heuristically from a wider solution space than previous methods. In addition, we propose a new routing constraint parameter which complements the roles of channel density in optimizing the number of required tracks. We have implemented the algorithm and achieved the best possible result against other similar works.

Introduction

Channel routing for automated physical VLSI design still remains as one of the most focused optimization problems even though it has been extensively studied [1,3-5]. The problem has been proved to be NP-complete and most presently known algorithms were adopting heuristic approaches to find the probably optimal solutions. The quality of the solutions, however, depends not only on the heuristics themselves but also the scope of the solution space on which they are applied. Most algorithms route a channel track-by-track and therefore limit the solution space to all possible moves of nets on current track. In our proposed method, we extend the space by considering the assignment of nets for both an upper and a lower track. In some cases, a net is permitted to occupy a free column in the upper track only if it is followed by an assignment of the same net to a designated column in the lower track to maximize channel utilization. Comparing to all the examples from [1,3-5], we have obtained equal or better results without introducing much complexity.

Channel Description

The most common way to capture a channel's difficulty is by an associated vertical constraint graph (*VG*) and a horizontal constraint graph (*HG*). The *VG* imposes an order in which a track assignment of nets should follow and the *HG* indicates the congestion profile of a channel. Notice that the *VG* is by no means unique as any permutation on channel columns yields the same *VG*. Moreover, the availability and the locations of free columns cannot be identified in either graphs. In order to describe a channel effectively for our needs, we have borrowed the modified vertical constraint graph (*MVG*) concept introduced by *Shimamoto* and *Sakamoto* [2] and built our algorithm from such graph.

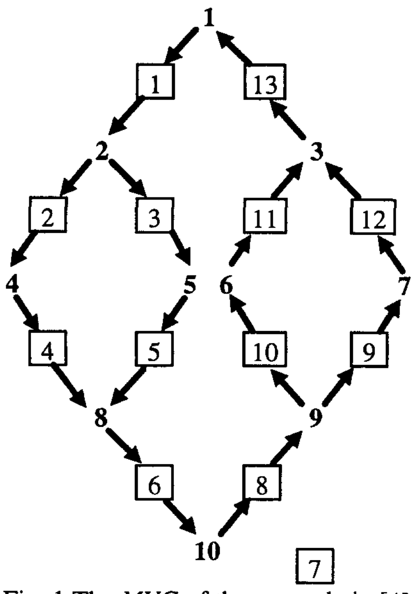

Fig. 1 The *MVG* of the example in [4]

The most significant aspect that differentiates our method from others lies in the wider scope of the solution space and the finer selection process for a final solution, which consists of two completed tracks, from the candidates. The solution space is now made of segments from both an upper and a lower track. After identifying a respective candidate set for either track, we alternatively select a partial solution from the first candidate set, immediately updating the routing constraints, and seek for a partial solution from the other set based on the new constraints. By repeating the process, two tracks would have been completed.

As an example, the *MVG* for the channel in [4] corresponds to Figure 1 above where **1** and $\boxed{1}$ represent a net node and a column node respectively.

In general, the paths in the *MVG* can be identified either *cyclic* or *non-cyclic*. In tracing along a path, when all the net nodes are arranged in a loop, it is named a *perfect cyclic path* shown as the outmost cycle in Figure 1. A *leading net node* is the net node of a path in which no other net node precedes it. For example in Figure 1, if the edge from net node **2** to column node $\boxed{3}$ is deleted, net node **5** will become a leading net node. On the contrary, an ***ending net node*** does not have any other net node following it in some path. Net node **7** in Figure 1 would be changed to an ending net node if the edge from column node $\boxed{12}$ to net node **3** is removed. Most often there are present of branches in a path. The net node at the junction is labeled as ***multi-terminal net node***, that is, net node **2** in Figure 1. Sometimes ***free column*** shows off as both the upper and lower terminals of a column are unoccupied. Column $\boxed{7}$ demonstrates such node in Figure 1. If there is only one of the terminals unoccupied, that column is named ***singly-free column***. Wire segments of any net can be coupled to these columns. With the associated methods all of the above wire segments are ready for route.

Heuristic

In our proposed heuristic, we consider two phases, the extracting phase and the selecting phase. In the extracting phase, all possible wire segments for both the topmost and the bottommost tracks are sorted out and proceeded to the next phase for selection. With the above descriptions of the channel, all feasible wire segments can be identified with the associated methods. ***Net Reduction R*** extracts wire segments of leading and ending nets and finishes the connection of these nets. ***Edge merging H*** looks for multi-terminals and connects them together. ***Column substitution S*** and ***Path transfer T*** link nets to the free and singly-free columns. Their respective sets of wire segments are shown below:

$R = \{\ w:$	The associated net of w is either leading net or ending net$\}$
$H = \{\ w:$	The associated net of w is a multi-terminal net$\}$
$S = \{\ w:$	The column occupied by one end of w is a free column$\}$
$T = \{\ w:$	The column occupied by one end of w is a singly-free column$\}$

After performing the extracting phase, we form two candidate sets at a time starting from the topmost and bottommost tracks for the channel. The selection criteria are based on the channel density d_m and the longest path length parameter v_m. In Figure 1, $d_m = 4$ and $v_m = 8$. It is noticed that in tradition the number of total tracks required is at least the maximum of both, i.e., $N \geq max\{d_m, v_m\}$. Therefore d_m and v_m should be considered as the two constraints in choosing a suitable candidate for routing, complementing each other. In manhattan model, d_m cannot be reduced. Wire segments which will not increase d_m are said safe and given a higher priority for the selection. However, we can minimize v_m by using the proposed ***tuple value***, which represents the relative distances of a net node from the starting and the ending of a path, (p_l, p_e). It can be assigned by using the Breath-First-Search (BFS) starting from all leading and ending net nodes. For the perfect cycle without any leading and ending net node, an arbitrary node is chosen to start the search. Tracing from net 1 in Figure 1, Table 1 gives the initial tuple value for each net.

Net	1	2	3	4	5	6	7	8	9	10
Tuple Value	(1,8)	(2,7)	(8,1)	(3,6)	(3,6)	(7,2)	(7,2)	(4,5)	(6,3)	(5,4)

Table 1 Initial tuple value for each net

As the tuple value gives the possible shortest path to a net, we reduce the path length by selecting the wire segment which involves the greater tuple value for routing. For example in Figure 2, after the wire segment from column $\boxed{2}$ to column $\boxed{3}$ of net **2** has been routed at the top track, the wire segment from column $\boxed{4}$ to column $\boxed{5}$ of net **8** is connected at the bottom track changing column $\boxed{4}$ to a single-free column with **0** as the terminal at the bottom. Besides, it also helps in breaking the cycle in the path such as performing the column substitution method at column $\boxed{7}$. The complete routing of the example from [4] is illustrated in Figure 2.

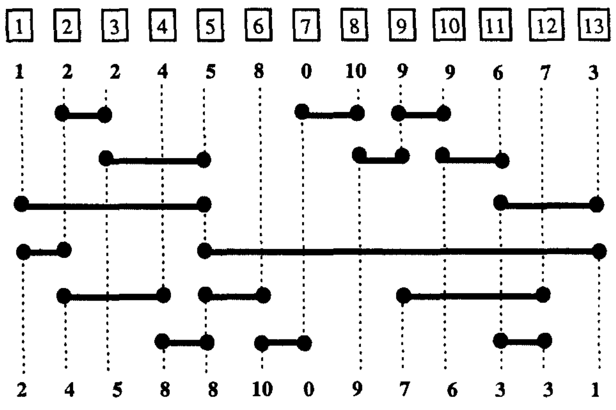

Fig. 2 Channel routing example

Form the about discussion, after optimizing the suitable wire segments by their tuple values, we have to choose the wire segments by considering their safety and effectiveness. We proposed a cost function below as the criteria of selecting the candidates:

$\delta = (Sf, V, C, N_d, p, S_p, M)$

It consists of 7 different parameters which are arranged in the order of decreasing significant. Sf sets from 0 to 1 if the wire segment is safe and therefore it is the most vital parameter. Then we have to consider whether the path length of the associated net is greater than the channel density making V to 1. Besides, if the net is completed after the connection, C becomes 1. After that, we reduce the channel density by choosing the one who cuts most columns with d_m, i.e. the larger N_d value. This parameter is only suitable for R and H methods. Then we choose the associated net in a longer path, p. Of course, shortening the span of a net, S_p, will surely reduce the complexity in the HG. Finally, the priority of different methods are set as $R > H > S > T$. It is reasonably assumed that S and T may increase the channel density. On the contrary, R can complete the connection of a net. We choose the wire segment with the larger cost by comparing the parameters in its significant order for connection.

Conclusion

This Double-Track Greedy Algorithm has been implemented in C language on a sparc20 station and the results are listed in Table 2. We have obtained equal or better results to the pervious works in terms of the number of tracks. It gives an effective break in long and cyclic path as assigning the candidates to both the top and the bottom tracks simultaneously. Moreover, this algorithm depends on the graph alone, showing no difference in the number of tracks even when the channel is mirrored.

Problem Data (No. of nets)	[1]	[3]	[4]	[5]	[#]
Ex. 1 (21)	12	12	-	10	10*
Ex. 3a (45)	15	15	15	15	14*
Ex. 3b (47)	17	-	17	12	11*
Ex. 3c (54)	18	18	18	16	14*
Ex. 4b (57)	17	18	-	23	17*
Ex. 5 (64)	20	20	-	19	19*
Diff. Ex. (72)	28	27	19	36	21

[#] Our proposed Double-Track Greedy Algorithm
(*) Optimal solutions for the examples without exits.

Table 2 Results for examples given in [1,3-5] (number of tracks)

References

[1] T. Yoshimura and E. J. Kuh, "Efficient Algorithms for Channel Routing," IEEE Trans. CAD Integr. Circuits Syst., pp. 25-35, Jan. 1982.
[2] T. Shimamoto and A. Sakamoto "Cycle Breaking and Longest Path in Channel Routing Problem," Electronics and Communications in Japan, Part 3, Vol. 72. No. 11,1989.
[3] J. S. Wang and R. C. T. Lee, "An Efficient Channel Routing Algorithm to yield an optimal solution," IEEE Trans. Comput., pages 957-962, July 1990.
[4] T. T. Ho. "A General Greedy Channel Routing Algorithm," IEEE Trans CAD, Vol. 10, No. 2, pp. 204-211, Feb. 1991.
[5] S. Soh, S. Rai and R. Mehrotra, "An Efficient Approach for Channel Routing in VLSI," Computers Elect. Engng, Vol. 17, No. 2, pp. 105-112, 1991.