Parallel Channel Routing

Mehdi R. Zargham
Computer Science Department
Southern Illinois University
Carbondale, Illinois-62901
(618)-536-2327

Abstract

A parallel algorithm is proposed for solving the problem of channel and switchbox routing in the design of VLSI chips. The algorithm is suitable for implementation on a shared-memory multiprocessor environment. Our approach does not impose restrictions on the channel type (such as fix or variable channel widths) and the number of available layers. The algorithm contains three major phases: I) dividing the channel into several regions by selecting some columns, 2) assigning tracks to nets of the selected columns, and 3) assigning tracks to nets of the columns in each region.

1. Introduction

One of the problems in the algorithmic design of VLSI chips is channel and switchbox routing. Given a collection of cells and ports (or terminals) on their boundaries, the problem is to connect the ports of these cells for a specific design requirement. A specific number is assigned to each port; ports with the same number are connected by a net.

Several algorithms for the above problem have been proposed. However, to our knowledge, none of these algorithms are suitable for implementation in a shared-memory multiprocessor environment. Since shared-memory multiprocessors have been widely available during the past three years, it was decided to develop a router suitable for these machines. One of the main issues that must be considered in developing such a router is that the algorithm should be independent of a number of available processors on the machine. That is, the algorithm should give the same result when it runs by one processor or more than one processor. Hopefully the addition of more processors will improve the performance. To achieve these goals the algorithm, for a given number of processors, will produce works (or sub-tasks) in order to keep each processor busy as much as possible.

Initially, all of the processors may be viewed as being in a "global pool." Then a processor will leave the pool, producing a "pool of works." These works are independent of each other. That is, once a processor is assigned to a work it does not need to communicate with other processors. When a pool of works is produced, the main steps of the router involve the operation of a processor finding a work and the possibility of adding new works to the pool.

The question that remains to be addressed is "what is a work?" To answer this question it is assumed there is a grid superimposed over the channel; the horizontal grid lines are called "tracks" and the vertical ones are called "columns." Then, in our router, a work is considered to be the assignment of tracks to the nets which are passing through a column. The assignment is done based on the global information about the channel (which is known in the beginning) and/or the tracks which already have been assigned to one of the neighboring columns.

In this paper, it is assumed that there are only two layers available for routing a channel or a switchbox. However, the router can be easily modified to route channels with more than two layers.

In the next section, some proposed ideas and algorithms that have some impact on the development of our algorithm are discussed. Section three contains a more detailed description of our router. Section four discusses the aspect in parallel implementation of the router. Finally, section five concludes the paper.

2. Related Work

The router should be able to generate a compact and relatively fast routing layout. The solution to this problem is basically a heuristic one; either an algorithm which may fail during the process of routing or an algorithm with restrictions imposed on the problem. Most of the solutions to this problem that have been proposed [2,3,4,5,6,13,14,15], impose some restrictions on the channel type (such as: switchbox, variable channel widths, and etc.) and/or number of layers (such as: one layer, two layer, three layer and etc.). To eliminate most of these restrictions, recently Enbody and Du have proposed a general purpose router [7]. The main outline of their algorithm is that: "A routed channel can be realized by solving individual columns and then combining them using a matching process." This is a nice concept when the ideal assignment of tracks for the nets of each column can be obtained easily. Unfortunately, in their paper, no details are given for finding these assignments. However, the idea of assigning tracks to a column has helped us in developing our algorithm.

Another concept that gave us some insight into our algorithm is due to the work of Kaplan [8]. His router uses a window that moves in steps from the bottom to the top of the channel. At every position, the window area is filled with conductor segments. A choice is made in order to assign some of these segments to a track. One of the criteria that has been used for making such a choice is based on the vertical distance between the window and the port to which the segment should be connected. The possibility of selecting the segment should increase as the vertical distance decreases. In our algorithm the idea of distance is used as a criteria for finding a path for a net from one column to the next.

3. The algorithm

The router has three main phases as follows:
Phase 1: Dividing the channel into several regions

CH2540-3/88/0000/0128$01.00 © 1988 IEEE

(or sections) by selecting some columns.
Phase 2: Assigning tracks to nets of the selected columns.
Phase 3: Assigning tracks to nets of the columns in each region.

3.1. Phase 1:

The router begins by reading in the data about nets, number of regions (r) and number of available processors (p). Given r (where $r<p$), the router will divide the channel into r regions. This is done by selecting $r-1$ columns among all the columns in the channel. Figure 1 shows a channel which is divided into k regions by selecting columns $c_1, c_2, \ldots, c_{k-1}$.

The selection of columns is made by optimizing the following two goals: 1) increasing number of nets in each column, and 2) dividing the channel into equal (or near equal) regions. That is, while we are dividing the channel into regions, we try to choose the columns with a density, as much as possible, near to the maximum density (the density of a column is defined to be the number of nets in that column, see [2,3]). The selection of a column with a large number of nets will limit the number of choices for assigning tracks to its nets. Thus, the problem of assigning ideal (or near ideal) tracks to their nets becomes easier to solve.

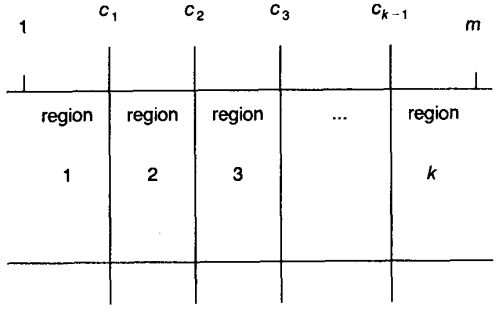


Figure 1. A channel, which has m columns, is divided into k regions.

3.2. Phase 2:

To find a suitable track assignment to the nets of a column, say column j, the following three steps are done.

i) Finding a preliminary partial ordering between the nets which pass through j.
ii) Finding a final ordering (with or without alternative) between the nets.
iii) Assigning tracks to the nets.

The preliminary ordering is made on the basis of a Vertical Constraint Graph [5,15], denoted by VCG. (A VCG is a dependency graph that specifies the order in which nets must be placed from top to bottom in a channel). For example, if there are three nets a, b, and c in column j, and there is a path from a to c, and a path from b to c, the nets will be ordered as:

level 1: [a, b], and
level 2: [c],

which means: nets a and b should be assigned to tracks above the track for c. If the length of a path between two nets is greater than a factor of maximum density, the path will be ignored at this step. This rejection eliminates the side effects of some long cycles that may exist in VCG. However, if there is a short cycle, some of the edges will be ignored in this step to create an acyclic graph.

The final ordering and track assignment is made by grading the nets. The grading is done with a grade function that assigns a value within the range [-1, 1] to each net. The value -1 means that the net should be assigned to one of the bottom tracks of the channel. As the value approaches to 1, the choice for the track will approach to higher tracks of the channel.

Given the net b in column j, the grade function f is modeled as:

$$f(b) = p_1(VCG-distance)+p_2(x-distance),$$

where:
p_1 and p_2 are tuning parameters,
$VCG-distance$ is the relative distance of b from other nets in VCG, and
$x-distance$ is the relative horizontal distance of b to the nearest terminal node on the right and left of column b.

The $VCG-distance$ gives a global view of the net's position respective to the other nets in the channel, and is computed as:

$VCG-distance$ =

(maximum path size from every node to b in VCG)
- (maximum path size from b to every node in VCG).

The $x-distance$ gives relative horizontal distance to the nearest terminal node on the right and left of column j, and is evaluated according to their position in VCG and distances from their terminals.

$$x\_distance = \frac{d}{1+|x_{b,r}-x_j|} + \frac{d}{1+|x_{b,l}-x_j|}$$

where:
$x_{b,r}$ is the x-coordinate of the nearest terminal of net b on the right side of column j (including column j),
$x_{b,l}$ is the x-coordinate of the nearest terminal of net b on the left side of column j (including column j),
d is an integer which present the position of terminal, and is defined as:
$d=1$, if the nearest terminal of b is connected to the top of channel.
$d=-1$, if the nearest terminal of b is connected to the bottom of channel.

Thus, if the x-distance approaches to -1, it is better to assign a track near to the bottom of channel. Otherwise, if it approaches to 1, assign a track near to the top of channel.

3.3. Phase 3:

In phase 2, tracks were assigned to nets of the leftmost and/or rightmost column of every region. For example, in Figure 2, we have already assigned tracks to the columns i and j of region i.

We begin to assign tracks to each column of region from left (right) to right (left). Given that tracks are already assigned to column $j+k$ (for $0<k<i-1$), the column $j+k+1$ can be routed by considering the recent assignment of $j+k$ and i, and the grade

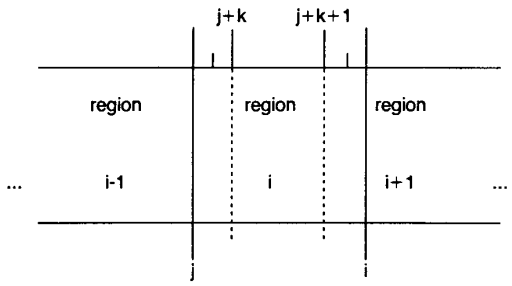


Figure 2.

function f, where:

$$f = p_1(VCG-distance)+p_2(x-distance)+p_3(y-distance)$$

(p_1, p_2, and p_3 are tuning factors).

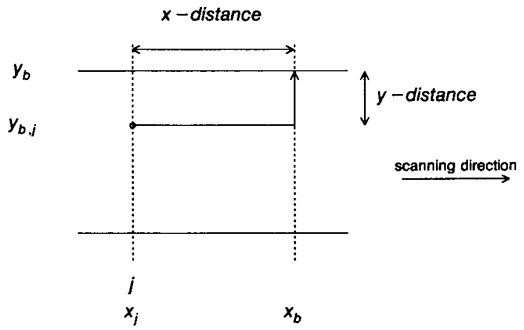
For a given net b, which is already assigned to a track t in column $j+k$, the $y-distance$ is the vertical distance between t and b's terminal. This distance, as $x-distance$, helps in deciding which direction (up, down, or straight) the net b should be routed. Based on the number of b's terminals, the $x-distance$ and $y-distance$ are calculated differently. Figure 3 shows the values for $x-distance$ and $y-distance$ in two cases. In the first case (shown in Figure 3.a), it is assumed that the net b has only one terminal node on the scanning direction. In the second case (shown in Figure 3.b), it is assumed that the net b has more than one terminal node on the scanning direction. In the later case, for each terminal of the net b, say b_i, the grade value for $x-distance$ and $y-distance$ is computed. Among all these values, the maximum absolute value is chosen for computing $f(b)$.

Once the tracks have been assigned to the nets of a column we can determine possible routings between that column and the adjacent column. As suggested by Embody and Du [7], we allow four types of moves that nets can make in order to route between two adjacent columns. The moves are straight, dogleg, h-jog, and exchange, see Figure 4.

The H-jog and exchange moves are possible. This is because, as pointed out by Deutsch [10], most routers use a design rule checker which determines the space between the wires to be the space between vias. Since vias are wider than wires, by judicious adjustment of these vias we can achieve these moves. To make the process of adjustment easier, we have tried to reduce the number of vias. This is done by assigning the same layer to the horizontal and vertical tracks in some special cases. If the router fails to route a column, it backtracks to previous column and tries another alternative.

To avoid an excessive amount of computation, a similar approach to the one proposed in [7] has been taken. The approach is based on limiting the number of alternatives at each column.

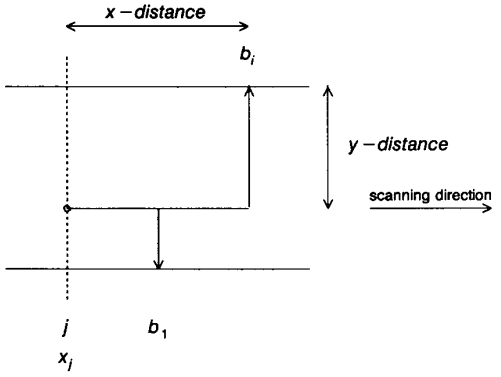
To clarify some of the routing techniques of the algorithm, two examples are given. Figures 5 and 6 present two well-known difficult routing problems which are routed by our algorithm. Figure 5 shows two different solutions for the Burstein's difficult channel [11]. In the first solution the channel is divided into two regions by the selection of column 4 (see Figure 5.a). In the other solution (Figure 5.b), the columns 4 and 9 are selected and the channel is divided into three regions. Figure 6 presents the Burstein's switchbox which is routed by two processors.



$$x-distance = \frac{d}{1+|x_b-x_j|}$$

$$y-distance = d(|y_b-y_{b,j}|)$$

(a) - Net b has only one terminal node on right side of j.



$$x-distance(b_i) = \frac{d}{1+|x_{b_i}-x_j|}$$

$$y-distance(b_i) = \frac{d}{1+|y_{b_i}-y_{b,j}|}$$

(b) - Net b has more than one terminal node on right side of j.

Figure 3. The values for $x-distance$ and $y-distance$.

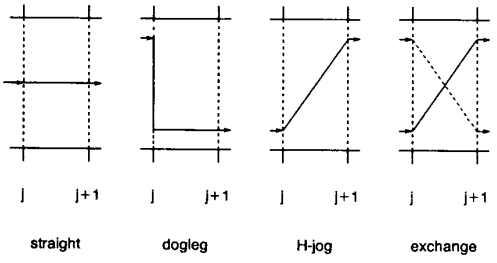
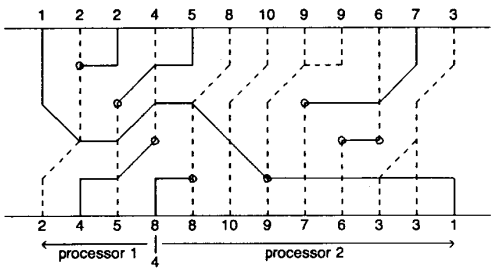
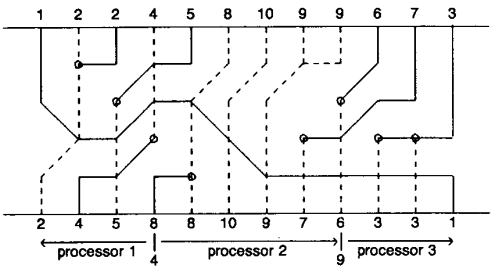


Figure 4 - Four types of moves for routing.



(a) - Two processors were used



(b) - Three processors were used

Figure 5. Burstein's difficult channel.

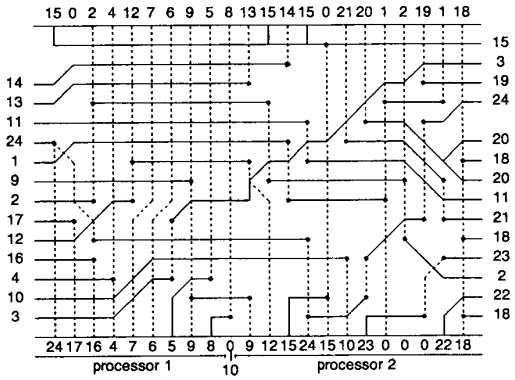


Figure 6. Burstein's difficult switchbox.

4. Parallel Processing

One of the main issues that must be solved is how to assign a new work to a processor. To discuss this problem, we present a search tree for one of the solutions of Burstein's channel. Figure 7 shows an AND/OR search tree for the solution which was presented in Figure 5. In this figure, except node 1, a column is assigned to each node. The nodes 1 and 2 are called AND nodes, and the other nodes are called OR nodes. In this tree, we say a leaf node (or terminal node) succeeds if there is a track assignment for the nets in the column which has been assigned to this node. An OR node, which presents several alternatives, succeeds if all the successor nodes on only one of its branch succeed (note: an OR node may have only one branch or more than one branch). The successor nodes for a node, say n, are the nodes of the subtree which is rooted by n. An AND node succeeds if there is at least one track assignment to every column which is assigned to its successors. Thus, the channel problem has a solution, if all the AND nodes succeed.

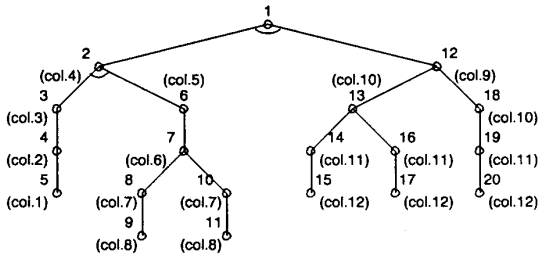


Figure 7. Search tree for one of the solutions of Burstein's channel

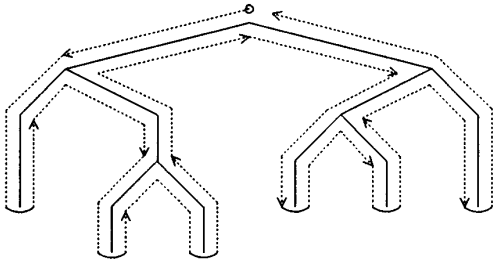
In the beginning, all of the processes are in a global pool. Initially, one processor, say processor 1, will leave the pool and begin dividing the channel into three regions. As the result, processor 1 will produce an open AND node 1, and two other nodes 2 and 12. By open node, we mean a node that has a child which is not yet claimed by a processor. At this time, processor 1 will move down to node 2 and another processor, say processor 2, will go to node 12. Processor 1 will assign a unique track assignment to column 4, and makes node 12 an open AND node by producing two nodes 3 and 6. Notice that node 3 is assigned to the column on the left side of column 4 (that is column 3) and node 6 is assigned to right side column.

In parallel with processor 1, processor 2 will assign two alternative track assignments to column 9, and makes node 12 an open OR node by producing two nodes 12 and 18. The best alternative will be given to node 12 (the leftmost branch, which is also called favorite branch), and the second alternative will be given to node 18. At this time, there are two open nodes 2 and 12. Each processor at that node will move to the node on the leftmost branch. That is: processor 1 will go to node 3 and processor 2 to node 13. However, there is still one more processor in the poll, say processor 3. Processor 3 can go either to node 6 or node 18. We move processor 3 to node 6. This is done, in case we are looking for only a single answer, the execution time will be reduced if we try to assign track to the columns which are not considered by any processor yet. When more than one answer is desired, the choice may not make so much difference in this step. Once processor 3 is assigned to node 6, node 2 becomes close, which means every child is processed.

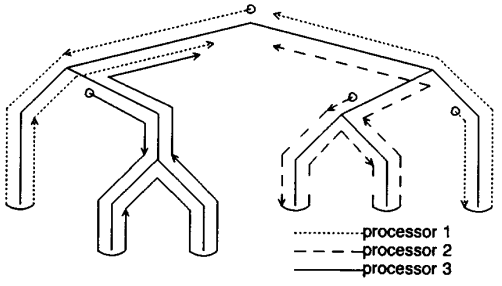
The mechanism for assigning a new work to a processor could be changed as a result of requesting one or all of the solutions to the problem. In the case when all answers are desired it is better to give a processor a great amount of work (a large subtree) to do, thus preventing its quick return to the global pool. This method prevents the overhead associated with switching between unrelated works (by unrelated works we mean the works which are on different subtrees). Overbeek and Lusk have proposed a similar approach for the next step in the Gigalip's Project [12]. The project is to develop a tool for or-parallel execution of Prolog.

Figure 8 demonstrates the above method for our search tree.

Table I shows different execution times for the search tree. The table was obtained by implementing the approach on Balance 8000 multiprocessor. (The Balance 8000 contains two to twelve tightly-coupled 32-bit microprocessors sharing a common memory pool.) The reason that the execution times are so high is mostly because of the low speed microprocessors which have been used in Balance 8000. However, for us the speed up which can be achieved by increasing the number of processors is important and the actual computation time is irrelevant.



(a) - one processor



(b) - three processors

Figure 8. Assigning processors to the search tree of Figure 7.

Table I.

Execution times for all solutions in search tree of Figure 4.	
number of processors	execution time/sec.
1	1.33
2	.70
3	.62
4	.50
5	.49
6	.49

In case only one answer is desired, it seems better to assign an idle processor to the most recent open OR-node. This will enable us to reach a possible answer quickly. At present, we are investigating this method.

5. Conclusion

The algorithm is perfect for implementation on any shared-memory multiprocessor environment (in our case, we have used Balance 8000), and does not impose restrictions on the number of processors. The increase of number of processors will reduce the computation time. However, there is always an upper bound on the number of processors which depends on the search tree for the routing problem. In addition to parallelism, the algorithm is general purpose in nature and could be used in a variety of routing problems.

References

[1] Mead, C. A. and L. A. Conway, "Introduction to VLSI Systems", Addison-Wesley, Reading, Mass., 1980.

[2] Printer, R. Y., "River Routing: Methodology and Analysis", Proc. Third Caltech Conference on VLSI, Bell Laboratories, Murray Hill, NJ., pp. 141-163, 1983.

[3] Chan, W. S., "A Channel Routing Algorithm", Proc. Third Caltech Conference on VLSI, Computer Science Press, Rockville, MD., pp. 117-140, 1983.

[4] Yoshimura, Y. and E. S. Kuh, "Efficient Algorithms for Channel Routing", IEEE Trans. on CAD, Vol. CAD-1, pp. 25-35, 1982.

[5] Chen, Y. K. and Lun Liu, "Three-Layer Channel Routing", IEEE Trans. on CAD, Vol. CAD-3, No.2, April, pp. 156-163, 1984.

[6] Malgorzata Marek-Sadowska, "Two-Dimensional Router for Double Layer Layout", Proc. of 22nd Design Automation Conference, pp. 117-122, 1985.

[7] Enbody, R. J. and H. C. Du, "General Purpose Router", Proc. of 24th Design Automation conference, pp. 637-640, 1987.

[8] Kaplan, David, "Routing with a Scanning Window - a Unified Approach", Proc. of 24th Design Automation Conference, pp. 629-632, 1987.

[9] Rivest, R. L. and C. M. Fiduccia, "A Greedy Channel Router", Proc. of 19th Design Automation Conference, pp. 418-424, 1982.

[10] Deutsch, D. N., "Compacted Channel Routing", Proc. ICCAD, pp. 223-225, 1985.

[11] Burstein, M. and Pelavin, R., "Hierarchical Wire Routing", IEEE Trans. on CAD, Vol. CAD-2, No. 4, pp. 223-234, 1983.

[12] Overbeek, R. and Rust Lusk, "Dispatching Logic for Tree-Walking", Technical paper, Argonne National Laboratory, 1987.

[13] Han, S. and S. Sahni, "Single-Row Routing in Narrow Streets", IEEE Trans. on CAD, Vol. CAD-3, No. 3, pp. 235-241, 1984.

[14] Han, S. and S. Sahni, "Layering Algorithms For Single-Row Routing", IEEE Trans. on CAD, Vol. CAD-6, No. 1, pp. 95-102, 1987.

[15] Pitchumani, V. and Qisui Zhang, "A Mixed HVH-VHV Algorithm for Three-Layer Channel Routing", IEEE Trans. on CAD, Vol. CAD-6, No. 4, pp. 497-502, 1987.