A Parallel Algorithm for Floorplanning with Integrated Global Routing

Shaodi Gao and K. Thulasiraman

Department of Electrical and Computer Engineering
Concordia University
Montreal, Quebec H3G 1M8, Canada

(Preliminary Version)

Abstract

In this paper we present a parallel algorithm for floorplanning with integrated global routing. The algorithm applies a hierarchical decomposition strategy that recursively divides the original problems into simple, independent subproblems for parallel processing. At each level of the hierarchy, the floorplanning, which is based on circuit partitioning, is followed by a global routing to estimate the routing area more precisely. The algorithm is implemented on a shared-memory machine and experiment results on different examples show relative speedup between 4 and 5 for 8 processors. The speedup is achieved without compromising the design quality.

1 Introduction

The recent advances in the VLSI technology allow the fabrication of highly complex circuits on single chips. Time-consuming software tools are needed to successfully design such chips. Parallel processing has become an essential approach to handle the rapid growing computational complexity. During the past few years, many parallel algorithms have been developed for various VLSI physical design problems. In this paper we present a parallel algorithm for floorplanning with integrated global routing.

Most physical design systems divide the whole process into three phases: Floorplanning (or placement), global routing and detailed routing. Floorplanning generates a relative position for all cells based on the connectivity and the dimensions of the cells. Global routing then determines the rough topology of the interconnections (nets) among the cells so as to achieve a uniform and low wiring density. Finally, detailed routing transforms the cells and rough wires into a geometric layout.

In traditional design systems, global routing was carried out after the completion of floorplanning. However, without the routing information, it is very difficult to predict the routing areas and therefore the dimensions of the cells during floorplanning. On the other hand, the floorplan hierarchy obtained by recursive partitioning lends itself easily to an incorporation of global routing into the floorplanning process. Several sequential algorithms have been proposed to combine floorplanning and global routing in the same process. Burstein and Hong [1] laid the ground work. Dai and Kuh [3] integrated global routing into the floorplanning process. They used a bottom-up approach to establish the floorplan hierarchy. Langauer and Müller [8] applied a different approach, in which partitioning is used for floorplanning.

In the recent past, a noticeable of work have been carried out to parallelize algorithms for floorplanning and global routing [2, 5, 11, 13, 14]. For example, a number of parallel algorithms based on techniques such as simulated annealing have been described in [13]. These techniques have inherent *data dependency* problems: the input of one task may depend on the output of another task. In order to relax this dependency, one has to rely on some inaccurate data, which can cause deterioration of the layout quality. On the other hand, the hierarchical approach decomposes the original problem recursively into a set of independent subproblems, which can be solved separately. Therefore, this approach offers a natural way for parallelization. We have retained the basic ideas from the algorithm of [8], but have made some adjustments to facilitate parallel processing. Our objective

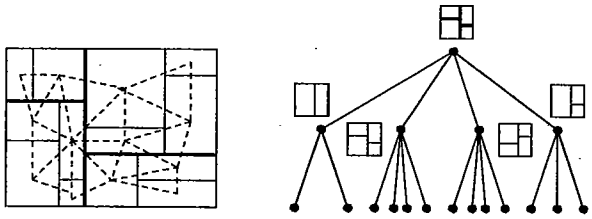

Figure 1: A floorplan, its routing graph and slicing tree.

in this work was to use parallel processing to speed up the floorplanning and global routing process without compromising the quality of the resulting layouts.

Section 2 of this paper describes our hierarchical floorplanning and global routing process. Section 3 discusses the issues of the design of the parallel algorithm in a shared-memory environment. Implementation details and experiment results are given in Section 4. Conclusions are presented in Section 5.

2 The Floorplanning and Global Routing Process

Our floorplanning algorithm adheres to the well-known scheme of slicing-based hierarchical floorplanning. The process can be divided into three phases.

Phase 1 (Circuit Decomposition): We apply the top-down mincut method to decompose the problem recursively into subproblems. The decomposition is represented by a slicing tree, which will be used to guide the whole floorplanning process (cf Figure 1).

Phase 2 (Floorplan Sizing): This phase traverses the slicing tree bottom up. At each node in the slicing tree a shape function of the corresponding subfloorplan is computed. This shape function takes all possible topologies for the subfloorplans into account. During sizing we estimate wiring by doing an version of global routing on the subfloorplans.

Phase 3 (Floorplan Computation and Global Routing): We first choose a floorplan alternative on the shape function at the root of the slicing tree. Then we realize the chosen floorplan by traversing the slicing tree top down breadth-first. At each tree node, a global routing is computed for the corresponding subfloorplan.

The following subsections describe the essential details of the process.

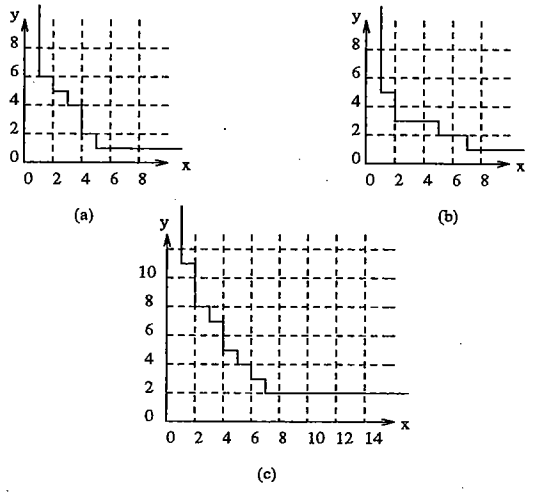

Figure 2: Computation of a shape function.

2.1 Circuit Decomposition

The mincut-based decomposition involves partitioning of a circuit into two halves of predetermined sizes such that the number of nets shared between the two halves is minimized. Since the problem is NP-complete, we apply the heuristic approach by Fiduccia and Mattheyses [4]. We first divide the circuit into two sets of cells of roughly equal sizes. Then we exchange cells between these two sets to reduce the number of nets which have pins in both sets.

At next level each of these halves will be partitioned. The procedure repeats itself until each subcircuit contains only one cell. This recursive procedure and the partitioning hierarchy is captured by the slicing tree with each tree node representing two consecutive levels of partitioning. Therefore, the slicing tree has degree 4 in general. This allows us to take more floorplan topologies into account than is possible with binary slicing trees.

2.2 Floorplan sizing

The shape function of a subfloorplan is defined as the lower area bound of all possible rectangles of the subfloorplan. It is expressed as the x and y dimensions of the rectangles. Figure 2(a) and (b) show the shape functions of two different subfloorplans. If we combine these two subfloorplans by a horizontal slicing cut, then Figure 2(c) shows the shape function of the resulting subfloorplan. That means for each x dimension, we add the y dimensions to get the new y

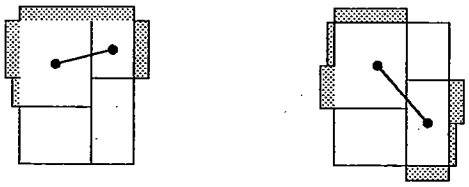

Figure 3: The model of adding routing area.

dimension. Similarly, we can compute the shape function if two subfloorplans are combined by a vertical cut. Since the slicing tree has degree 4, we have to process each topology alternative by executing twice the computation described above.

The routing area estimate for a topology alternative is determined by a thorough analysis of internal routing demands. This analysis amounts to solving an unconstrained global routing problem on this alternative. We consider the channelless routing model. Each route for a net gives rise to an estimate of routing area that is added for segments of the routing (cf. Figure 3). The amount of area added is governed by track demand factors, as in [15]. The unconstrained global routing problem computes a global routing of all nets that minimized the area of the rectangular box circumscribing the floorplan topology alternative together with all of the added routing area. This problem can be solved by means of integer program.

2.3 Floorplan computation and global routing

The sizing process carried out in Phase 2 and the choice of the layout alternative at the root of the slicing tree narrow the set of candidate floorplan topologies at each interior node of the cut tree down to a small number of topologies, all of which have the same area estimate. However, there are still several such topologies. Some of these topologies can be transformed into each other by rotation and reflection, for instance.

In Phase 3, we select one of these topologies on the basis of wiring cost. Intuitively, we select the topology alternative that minimizes the cost of extending the rough global routing that already exists on the top levels of the slicing tree. Then we extend the rough global routing to include the selected topology alternative.

Luk et al. [9] have shown how to do top-down refinement on the routing graph, which is a dual version of the flooplan, in order to refine the rough global routing. The refinement process iteratively expands nodes v of the routing graph that represent interior nodes x of the slicing tree into graph constructs G_x that represent the floorplan topology pertaining to v. Then, the edges incident to x are expanded to sets of edges that connect to G_x. After expanding the routing graph, the rough global routing has to be refined such that it uses G_x.

Here we use a different method than Luk et al. [9], because we want to consider all nets simultaneously, in order to be able to handle interdependencies between different nets. First, we distribute nets that are routed across an expanded edge e in the rough global routing onto the edges e was expanded into. Here we use a method based on min-cost flow that was put forth by Lauther [6] and Marek-Sadowska [10]. The min-cost flow networks that are generated have the size $O(1)$ and the corresponding min-cost flow problem can be solved in linear time in the number of nets running across the expanded edge in the rough global routing.

After the nets are distributed over the expanded edges in the routing graph, we compute the part of the global routing that pertains to G_x, by solving an integer program. For a detailed description of the global routing process see [5].

3 The Parallel Algorithm

The important aspects of our parallel algorithm has already been presented in the last section, such as hierarchical decomposition and slicing tree guided task generation. In this section we describe more about the organization of our algorithm and further exploit parallelism in individual routing tasks.

3.1 The Overall Structure

In Phase 1 each partitioning generates two separate subcircuits which can be further partitioned independently by different processors. There is no date-dependency problem. Therefore, the entire partitioning can be carried out in parallel. The bottom-up floorplan sizing process in Phase 2 and the top-down floorplan computation and global routing process in Phase 3 are guided by the slicing tree established during the circuit partitioning. For each node in the slicing tree the corresponding task will be generated. For Phase 2, each task is an independent computation of a size function followed by a routing area estimation. For Phase 3, each task includes performing global routing in a super-graph by means of integer programming and solving corresponding net assignment prob-

lem by finding min-cost flows. The parallel execution of the node tasks follows a breadth-first scheme: Tasks at the same level of the slicing tree will be carried out in parallel by the available processors. Because of our decomposition strategy, these tasks (at the same level) are completely independent from each other. All information exchanges follow through global (shared) data structures. Each processor gets the global data before starting a task and update the global data after completing the task. There is no need for date exchanges among processors during task execution periods. After all tasks at a slicing tree level are completed, a new set of independent tasks corresponding the next level nodes are generated and ready to be assigned to the available processors for parallel execution.

Algorithm Parallel Global Routing
Input: A slicing structure with I levels and a netlist
Output: A set of Steiner trees for the netlist.
$i = 0$; /* level 0 of the slicing tree */
while $i < I$
  dopar /* parallel processes for level i nodes */
    find the super-graph routing for the node
    dopar /* parallel processes
          for the super-edges of the node */
      find net assignment for the super-edge
    endpar
  endpar
  $i := i + 1$;
endwhile

3.2 Parallelization within a Task

Exploitation parallelism in individual routing tasks is important for further speed-up of the overall routing process. It is particularly effective at higher levels of the slicing tree, where there are not enough tasks for parallel processing. On the other hand, we have to take into consideration the overheads caused by the parallel task generation and management. If the task size is too small, the overheads may outweigh the benefits gained by parallelization. After experimenting with several options, we decided to choose the net assignment for parallel processing, because solving the min-cost flow problem is fairly time-consuming. An H_4 super-graph has five assignment tasks, each for one super-edge. These tasks can be executed in parallel. If the number of processors is not large, we apply this parallelism only at the first few levels of the slicing tree, because at lower levels there are enough nodes to be assigned to the available processors.

The basic structure of the parallel algorithm for the global routing in Phase 3 is given as follows.

4 Experimental Results

We have implemented the parallel floorplanning algorithm on a BBN GP1000 machine under Mach 1000 operating system. The program contains about 7000 lines of C code. Experiments were performed on two MCNC general cell benchmarks, ami33 and ami49, and two randomly generated larger examples. Testing was done on one, two, four and eight processors. Table 1 summarizes the relative speedups for different problem instances. The speedup by eight processors is round five for problems with large number of cells. The speedup for smaller problems is not so good, because are only few levels in the slicing tree and at most of these levels the power of parallel processing cannot be full utilized.

Chips	p	Time(sec)	Speedup
ami33	1	40.35	1.0
	2	23.74	1.7
	4	15.52	2.6
	8	11.53	3.5
ami49	1	121.43	1.0
	2	67.46	1.8
	4	43.37	2.8
	8	28.92	4.2
C1	1	174.43	1.0
	2	96.91	1.8
	4	52.86	3.3
	8	39.64	4.4
C2	1	403.54	1.0
	2	212.39	1.9
	4	122.28	3.3
	8	77.60	5.2

Table 2: Parallel speedup

The floorplanning results obtained by a single processor and by multiprocessors are the same. This is because the hierarchical decomposition creates floorplanning tasks which are totally independent from each other. Table 1 shows the example chips and the wirelengths of the global routing results . Our algorithm can always find a route for every net if there exits such a solution, because it always take the capacity constraints into consideration during minimization of the wirelength.

Chips	Cells	nets	Wirelength
ami33	33	123	183
ami49	49	408	1625
C1	63	500	2533
C2	92	1000	14524

Table 2: Routing Results

5 Conclusions

We have presented a parallel algorithm for floorplanning with Integrated global routing. The algorithm takes a hierarchical approach to decomposed the floorplanning and global routing problem into independent subproblems for parallel processing. It achieves very good speedup without compromising the quality of the solutions.

References

[1] M. Burstein and S. J. Hong, "Hierarchical VLSI layout: Simultaneous wiring and placement," *Proc. of VLSI'83*, pp. 45-60, Elsevier Publishers B.V., The Netherlands, 1983.

[2] R. J. Brouwer and P. Banerjee, "PHIGURE: A parallel hierarchical global router," *Proc. 27th Design Automation Conference*, pp. 650-653, 1990.

[3] W. W.-M. Dai and E. S. Kuh, "Simultaneous floorplanning and global routing for hierarchical building block layout," *IEEE Transactions on Computer-Aided Design*, CAD-6(5), pp. 828-837, 1987.

[4] C. M. Fiduccia and R. M. Mattheyses, "A linear-time heuristic for improving network partitions," *Proc. of 19th Design Automation Conference*, pp. 175-181, 1982.

[5] S. Khanna, S. Gao, and K. Thulasiraman, "Parallel hierarchical global routing for general cell layout," to appear in *Proc. of 5th IEEE Greate Lakes Symposium on VLSI*, 1995.

[6] U. Lauther, "Top-down hierarchical global routing for channelless gate arrays based on linear assignment," *Proc. VLSI 87*, pp. 141-151, Elsevier Science Publishers, 1987.

[7] T. Lengauer, *Combinatorial Algorithms for Integrated Circuits Layout*, John Wiley & Sons, 1990.

[8] T. Lengauer and R. Müller, "A robust framework for hierarchical floorplanning with integrated global route," *Proc. Int. Conf. Computer-Aided Design*, pp. 148-151, 1990.

[9] W. K. Luk, P. Sipila, M. Tamminen, D. Tang, L. S. Woo, and C. K. Wong, "A hierarchical global wire routing for custom chip design," *IEEE Transactions on Computer-Aided Design*, CAD-6(4), pp. 518-533, 1987.

[10] M. Marek-Sadowska, "Route planner for custom chip design," *Proc. Int. Conf. Computer-Aided Design*, pp. 246-249, 1986.

[11] R. Nair, S. J. Hong S. Liles, and R. Villani, "Global wiring on a wire routing machine," *Proc. 19th Design Automation Conference*, pp. 224-231, 1982.

[12] O. A. Olukotun and T. N. Mudge, "A preliminary investigation into parallel routing on a hypercube computer," *Proc. 24th Design Automation Conference*, pp. 814-820, 1987.

[13] J. Rose, "Parallel global routing for standard cells," *IEEE Transactions on Computer-Aided Design*, CAD-9(10), pp. 1085-1095, 1990.

[14] T. Watanabe, H. Kitazawa, and Y. Sugiyama, "A parallel adaptable routing algorithm and its implementation on a two-dimensional array processor," *IEEE Transactions on Computer-Aided Design*, CAD-6(2), pp. 241-250, 1987.

[15] G. Zimmermann, "A new area and shape function estimation for VLSI layouts," *Proc. 25th Design Automation Conference*, pp. 60-65, 1988.