PARAGRAPH: A Parallel Algorithm for Simultaneous Placement and Routing Using Hierarchy

Randall J. Brouwer
Engineering Department
Calvin College
Grand Rapids, MI 49546

Prithviraj Banerjee
Center for Reliable and High-Performance Computing
The University of Illinois at Urbana-Champaign
Urbana, IL 61801

ABSTRACT
In this paper, we propose a new parallel algorithm for combined standard cell placement and routing. Our focus in this research has been to develop a hierarchical decomposition scheme so that the subproblems are completely independent of each other and can be evaluated in parallel. We have developed a parallel algorithm such that the solution quality does not degrade with the addition of multiple processors, a common problem encountered by most previously reported parallel placement and routing algorithms.

The new parallel algorithm hierarchically integrates a quadtree cell placement algorithm, a bisection placement algorithm, a global routing algorithm, and a detailed routing algorithm. Unique partitioning techniques are used to decompose the various stages of the algorithm into independent tasks which are evaluated in parallel using dynamic task scheduling. Finally, we present results of an implementation of our parallel algorithm on a shared memory multiprocessor for several industrial benchmark circuits.

1. INTRODUCTION
In view of the increasing complexity of very large scale integrated circuits (VLSI), there is a growing need for sophisticated computer-aided design (CAD) tools to automate the synthesis, analysis, and verification steps in the design of VLSI systems. A recent approach to handling the problem's complexity and decreasing the running time of such tasks has been to apply parallel processing [1]. The advantages of parallel processing include: reduced runtimes, the ability to solve larger problems, and the availability of low-cost multiprocessors. Some of the tasks in the automatic design of integrated circuits which have been solved with parallel processing include the following [1]: cell placement, floor planning, global routing, design-rule checking, circuit extraction, logic simulation, circuit simulation, and test generation/fault simulation. The above results have demonstrated the wide variety of CAD applications that can be solved with parallel processing. However, it has also become very clear that parallel algorithm design is very difficult for certain types of applications.

A problem with many of the above parallel CAD algorithms is that as the number of processors increases, the quality of the solution degrades. One reason why this happens in parallel placement and routing algorithms, for example, is because processors must assume that the current state information contained within themselves is accurate while evaluating alternate moves of cells or nets. However, some processors may be changing important state information that may not be immediately reflected in other processors. It is very important to partition the tasks to be solved in parallel in such a way as to minimize the interaction among the tasks being solved simultaneously. Hierarchical methods can be used very effectively to partition a problem into independent subtasks, provided the partitioning is done carefully.

In this paper, we propose a parallel algorithm for simultaneous placement and global routing of cells in a row-based layout. Our focus in this research has been to develop a hierarchical decomposition scheme so that the subproblems are completely independent of each other and can be evaluated in parallel. We have developed a parallel algorithm such that the solution quality does not degrade with the addition of multiple processors.

2. BACKGROUND AND PREVIOUS WORK
The cell placement problem involves placing a set of cells or gates on a VLSI layout, given a netlist with the primary goal of determining the best location of each cell so as to minimize the total area of the layout and the length of the nets connecting the cells together. The task of global routing is to take a netlist, a list of pin positions, and a description of the available routing resources and determine the connections and macro paths for each net. In this paper, we will primarily focus our attention on row-based layouts (e.g. gate array, standard cell, and sea-of-gates). However, the algorithm presented is applicable for other design styles as well.

Most of the previous work in parallel algorithms for placement [23][5] have been based on simulated annealing, which is a form of iterative improvement based placement algorithm giving excellent quality of placement results. Recently, there have been some theoretical parallel algorithms for placement based on hierarchical decomposition techniques, but no results of the quality of the placements produced have been reported from this work [6].

There have been several parallel algorithms proposed for global routing. Two algorithms for maze routing have been developed, specifically for hypercube multiprocessors [78]. A different approach, developed for shared-memory multiprocessors [9], determines the best of all possible two-bend routes for each two-pin subnet of each net, by iteratively ripping up and rerouting nets.

Until now we have discussed algorithms for placement and routing by viewing these tasks separately. Noting the benefits of combining placement and routing, recently researchers have
begun to develop techniques which combine algorithms for the two problems, placement and routing. Srivastava et al. [10] proposed a novel hierarchy-based integrated placement and routing algorithm. The algorithm depends on having the underlying layout arranged as a slicing layout. Shragowitz et al. presented a placement and routing algorithm for use in the layout of sea-of-gates style chips. Suaris and Kedem [12] proposed an algorithm for combined placement and routing of standard cells based on quadrisection (an extension of bisection) [13]. Again in a hierarchical fashion, the cells are placed, based on the terminal propagation of each net, followed by global routing to generate a spanning tree for the nets.

The above reported algorithms for simultaneous placement and routing are very difficult to parallelize since they are inherently sequential. In the next section we will report on a new algorithm for simultaneous placement and routing which can be readily parallelized as well.

3. PARALLEL PLACEMENT AND ROUTING ALGORITHM

In this section, we will discuss a new parallel algorithm for placement and routing that will ensure that the solution quality does not degrade with the addition of processors. The specifics of the algorithm presented refer to standard cell layouts, but with slight alterations they can be applied to other row-based layouts as well.

3.1. Overview

The first step of the algorithm consists of a chip-level floorplanning which seeks to estimate pad, cell, and routing resources. The central step is the combined placement and routing of the cells and nets. The algorithm employed in this step will be explained in detail throughout this section. The final step is the detailed routing which takes the placement and global routing information and sets up the appropriate channels for channel routing. Since each channel is independent, these will be solved in parallel as well.

The placement and routing step consists of a number of operations which are executed in a certain sequence at each level of the hierarchical decomposition. The operations are (1) the placement of the current set of cells, based on the quadrisection algorithm of Suaris and Kedem [13], (2) the routing of nets for the quadrisection placement, similar to the algorithm described in PHIGURE by Brouwer and Banerjee [14], (3) the restricted global bisection of cells, and (4) the two-by-N routing of the nets as the bisection. Since we have tightly combined these placement and routing tasks, each operation intimately depends on the results of the other operations.

3.2. Quadrisection-based placement

In the quadrisection method of Suaris and Kedem [13], the layout is partitioned into four groups (a two-by-two array of bin), and cell movements occur among any of the four groups. An extension of the bisection heuristic for the selection of the cells is applied, which minimizes the net cuts over all four boundary segments of the two-by-two bin array through the movement of the selected cells. At each level of the quadrisection decomposition, a portion of the layout is selected and divided into four quadrants. Thus, at level k in the decomposition, the entire layout is divided up into a \(2^k \times 2^k\) array of quadrisection regions.

As part of the cost of a net, pseudo pins are determined for each net's connections to regions outside of the current layout portion. The cost function of [13] assumes each net can always be routed with the shortest path. We propose a better cost function which determines how each net is routed and calculates each net's cost based on the routing crossings of the four boundary segments [15]. For each pin configuration, the net cost function involves enumerating the various routing alternatives similar to the global routing model in [14]. Similar to the simple cost function, the boundary crossing information for each net, which is determined after a global routing is performed, can be stored as a vector of residency flags. As with the standard cost function, the cost function can be evaluated in \(O(1)\) time.

In addition to determining which cell movements provide the minimum cost gain, another important criterion in the selection of the cells is the determination of whether the movement of the cell would cause an imbalance in the total area of the cells occupied by each quadrant. We propose another important enhancement to the Suaris-Kedem Quadrisection algorithm [12] by including the ability to swap cells of equal size. A secondary restriction on the selection of the second cell for the swapping is that the cell must be in the quadrant \(r\), has a cost gain of 0, and have no nets in common with the first cell selected.

The sequence of selecting and moving cells is repeated until no cells can be selected for movement or when a sequence of \(k\) selections of cells with gains > 0 has taken place. The stored state information is then used to backtrack and undo cell movements which have only worsened the net states and the partition of the cells into quadrants. The steps of cell selection followed by backtracking are called a pass and are repeated \(p_{max}\) times, or until no gains are made on consecutive passes.

3.3. Routing of the quadrisection

At the end of the quadrisection operation, the cells of the portion of the layout have been placed in one of the four quadrants while minimizing the net crossings over the boundaries between the quadrants. A quadrisection routing operation is then used to verify and lock in the placement of the nets across the four boundary segments. A single iteration of the routing algorithm presented in [16] is used to determine the routing of the nets across the four quadrant boundaries.

If the route-based cost function is used in the quadrisection placement algorithm, it is necessary to know the routing of the nets in the quadrisection region before quadrisection can take place. The routing must be based on the current placement of the cells at the beginning of quadrisection. Thus, one iteration of the two-by-two routing algorithm will be performed before as well as after the quadrisection cell placement when the route-based cost function is used.

To determine the best routing of the nets, an accurate measure of the routing capacities across the four quadrisection boundaries must be made. Since the exact locations of the cells is not known until the placement algorithm completes, we measure the routing capacity along the horizontal boundaries as the average number of feedthroughs available divided by the number of rows over which the cells are to be placed. The simplex computations of the linear programming can then insert feedthrough cells in the rows or increase the channel height, if needed [14]. As cells are moved, the horizontal capacity measure can vary and must be recalculated before every routing. The vertical boundary capacities are an average of the number of tracks available in the channels intersecting the boundary.

3.4. Initial placement for quadrisection

It is essential for the quadrisection placement algorithm that the cells to be placed are initially divided into four groups.
In [13] a two-stage min-cut scheme is used to generate the initial partition, or seed, for the quadrisection. In this section we propose a new method called Restricted Global Bisection for providing the initial partition for the quadrisection-based placement.

3.4.1. The X-dimension restricted global bisection

The bisection is performed separately in the X-dimension and the Y-dimension. An X-dimension bisection consists of the cells located in a pair of full-height columns between $x_k$ and $x_{k+1}$. The columns widths narrow as the hierarchical progression continues. The columns are vertically partitioned into $2^k$ regions, where $k$ is the hierarchy level. A bisectioning of the cells is then performed subject to the restriction that cells may only move to the same partition in the adjacent column of the pair. Thus, every cell's y-coordinate remains untouched. To initiate the bisection, a clustering algorithm is applied to divide the cells.

In the same manner as [17], cells are assigned a cost function based on the net connections and are moved or swapped between the bisection halves to reduce the horizontal net length. To model the restricted movement of cells in the cost function, we have devised a cost function based on the number of crossings by a minimal length net over the entire bisection line between the columns.

3.4.2. The Y-dimension restricted global bisection

Alternately, the Y-dimension restricted global bisection algorithm partitions the layout into horizontal strips the full width of the layout area. By applying the cost function horizontally, we reduce the demand of the nets for a high number of feedthroughs and route each net in as few channels as possible. The cell size restrictions on movement are similar to quadrisection, but consideration is given to the area available on each half. In the same manner as quadrisection, the sequence of cell selections and movements until no more moves are possible is called a pass and is followed by a backtrack to the last best state. A sequence of passes is performed until either a limit is reached or until no further gains can be made.

3.4.3. Combining X- and Y-dimension bisectioning

Since the X- and Y-bisection algorithms alter the x- and y-coordinates of the cells, respectively, they are independent of each other, and both directions can be evaluated simultaneously to determine each cell's initial quadrant. However, since the x- and y-coordinates of each cell is set independently, the balance of cell areas may not be valid. Therefore, we move selected cells from the fullest quadrant to the least full one before proceeding.

3.5. Two-by-N global routing

Following the bisection operation, a two-by-N routing of each bisection region is performed, using the Burstein and Pelavin algorithm [16] with the parallel algorithm implementation ideas presented by Brouwer and Banerjee [14]. The routing goal is to determine the sets of nets crossing each half of the partition lines running perpendicular to the the x- and y-bisection lines.

Although the bisection routing was introduced as immediately following the bisection placement, it is necessary to perform a routing immediately after the quadrisection placement. The two-by-N bisection routing following the quadrisection placement is necessary not only because the balancing of cell areas may change the best routing between quadrisections evaluated in parallel, but also to optimize the routing connections following movements of cells among the quadrants. Thus, at each level of the hierarchical decomposition, the bisection routing algorithm is effectively applied twice.

3.6. Parallel Algorithm Summary

In Figure 1, a graphical description of the placement and routing algorithm is shown. In this figure, the operations performed at each level of the hierarchical decomposition are denoted by a set of circles between a pair of horizontal dashed lines intersecting the appropriate column. The circles represent instances of the operations to be performed on a portion of the layout. For example, in the quadrisection placement column, one circle at hierarchy Level 0 represents a quadrisection covering the entire layout. Four circles at Level 1 represent the four quadrisections, each covering one-fourth of the layout.

At each level of the decomposition, the cells are initially placed using the global X- and Y-bisection placement algorithm. This is immediately followed by a two-by-N routing of the same regions to determine the net crossings for the boundaries of each quadrisection placement region on that level. Next, a two-by-two routing of each quadrisection placement region is performed, taking into account the nets crossing through and ending in the region, to set up the current routing configuration for each net to be used in the quadrisection cost function. The quadrisection placement algorithm is then used to improve the current locations of the cells (provided by the previous bisection placements). Since the previous two-by-N routing may need to be modified due to movements of the cells inside the quadrisection region, the two-by-N routing algorithm is repeated. Finally in the hierarchy level, a two-by-two routing is applied to each quadrisection region to fix the net crossing locations on the four cut lines separating the four quadrants. Figure 2 shows the operations on a portion of the layout for hierarchy level number two.

<table>
<thead>
<tr>
<th>Level</th>
<th>Quadrisection</th>
<th>Bisect Place</th>
<th>Bisect Route</th>
</tr>
</thead>
<tbody>
<tr>
<td>k</td>
<td>Place</td>
<td>Route</td>
<td>Horiz</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>o</td>
<td>o</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>o oo oo</td>
<td>o</td>
</tr>
</tbody>
</table>

Figure 1. Placement and routing decomposition.
4. RESULTS

The parallel algorithm for placement and routing has been implemented in the PARAGRAPH (PARallel Algorithm for Routing and Placement Hierarchically) package in C on an Encore 510 Multimax shared memory multiprocessor.

Suaris and Kedem [1312] have already demonstrated that the quadrisection approach to cell placement and global routing is very competitive with respect to the quality of placement results produced by the best cell placement algorithms such as min-cut, force-directed, and simulated annealing, and with modifications to our implementation and the enhancements we have proposed, we expect to achieve similar or better results. In addition, our algorithm is designed to have more parallelism to exploit the use of parallel machines.

In the following we will first show the comparison with our implementation of the Suaris-Kedem algorithm (this approach was taken in order to run some controlled experiments of the advantages of addition one new feature at a time) and our new proposed algorithm. We have done extensive investigations of a large number of features which are reported in [15]. The investigations included the following: the effect of route-based quadrisection, the effect of cell swapping, and the effect of global restricted bisection for initial placement on the solution quality. In this section we will only report on the final results of the layout quality produced by PARAGRAPH and the Suaris-Kedem quadrisection algorithm.

The parallel placement and routing algorithm was evaluated on a number of placement problems. Two of the circuits are the Primary1 (P1) and Primary2 (P2) benchmarks from the Microelectronics Center of North Carolina (MCNC). The remaining are other standard cell circuits of varying sizes. Table 1 provides the number of cells, the number of pads, and the number of nets in each of the circuits. In the following tables, we evaluate the effect of the various algorithm options on the placement quality which will be measured by three quantities. The total length of the net segments in the channels is denoted as WL, the number of routing tracks as determined by summing up the maximum channel density for all channels is denoted as TC, and the layout area of the rows of cells and the channels is denoted as LA.

Table 2 shows the comparison of the layout quality produced by our PARAGRAPH algorithm and the Suaris-Kedem quadrisection algorithm. As can be seen, the layout qualities in all three measures are comparable in both algorithms. It should be noted that the quality of the layout produced by PARAGRAPH is independent of the number of processors used.

Table 3 provides information on the speedups attained for the example circuits using a variable number of processors. To improve the speedup values for large numbers of processors, it is important to eliminate load imbalances among processors and to partition the tasks in the top-level decomposition steps into sub-tasks that may be executed in parallel. The current focus of the research is to fully develop the partitioning of the tasks at the top.

Table 1. Benchmark statistics

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Cells</th>
<th>Pads</th>
<th>Nets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z1</td>
<td>469</td>
<td>37</td>
<td>494</td>
</tr>
<tr>
<td>Z2</td>
<td>1661</td>
<td>61</td>
<td>1979</td>
</tr>
<tr>
<td>Z3</td>
<td>2776</td>
<td>64</td>
<td>3258</td>
</tr>
<tr>
<td>P1</td>
<td>752</td>
<td>81</td>
<td>1185</td>
</tr>
<tr>
<td>P2</td>
<td>2907</td>
<td>107</td>
<td>3710</td>
</tr>
</tbody>
</table>
routing algorithms. We have found that at the higher levels of
resources may be
existing sequential placement and routing algorithms and have
demonstrated their effectiveness. We have
parallel processing. We have discussed enhancements over
have developed a parallel algorithm such that
6. REFERENCES
level node. Another area of further investigation is the effect of
eliminating the scheduling barriers so that processes need not
wait for other processes to finish before continuing execution.
5. CONCLUSIONS
In this paper we have presented a new parallel algorithm for
simultaneous placement and global routing that is well-suited for
parallel processing. We have discussed enhancements over
existing sequential placement and routing algorithms and have
demonstrated their effectiveness. We have also reported on an
implementation of the parallel algorithm on a shared-memory
multiprocessor.

Our focus in this research has been to develop a hierarchical
decomposition scheme so that the subproblems are completely
independent of each other and can be evaluated in parallel. We have
developed a parallel algorithm such that the solution quality does not degrade with the addition of multiple processors, a
common problem observed by previous parallel placement and
routing algorithms. We have found that at the higher levels of
any decomposition scheme, it may be necessary to develop ways
to partition the relatively few tasks so that all processing
resources may be fully utilized. To achieve this parallelism at the
top levels, the algorithm will be extended in the future to allow
for concurrent evaluation of interdependent tasks.
6. REFERENCES
computer-aided design applications,” ICCAD-88
Tutorial, also Tech. Rep. no. CSG-104, Coordinated
Science Laboratory, Univ. of Illinois, Urbana, IL, May
annealing for standard cell placement on a hypercube
34-37, Nov. 1986.
simulated annealing algorithms for standard cell
placement on hypercube multiprocessors,” IEEE Trans.
1990.
simulated annealing on a multiprocessor,” IEEE Trans.
Computer-Aided Design, vol. CAD-6, no 4, pp. 534-549,
June 1987.
“Parallel Standard Cell Placement Algorithms with
Quality Equivalent to Simulated Annealing,” IEEE
hypercube architecture,” Proc. Int. Conf. Parallel
investigation into parallel routing on a hypercube
multiprocessor computer,” Proc. Int. Conf. Parallel
1986.
combined place and route scheme for standard cells,”
234-244, Mar. 1989.
quadrisection and its application to standard cell
placement,” IEEE Trans. Circuits Syst., vol. 35, no. 3,
Hierarchical Global Router,” Proc. 27th Design
[15] R. J. Brouwer, Parallel algorithms for placement and
heuristic for improving network partitions,” Proc. 19th

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Number of Processors</th>
<th>Runtime (s)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z1</td>
<td>1</td>
<td>124</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>69</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>47</td>
<td>2.6</td>
</tr>
<tr>
<td>Z2</td>
<td>1</td>
<td>1648</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1016</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>721</td>
<td>2.3</td>
</tr>
<tr>
<td>Z3</td>
<td>1</td>
<td>3062</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1900</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1318</td>
<td>2.3</td>
</tr>
<tr>
<td>P1</td>
<td>1</td>
<td>270</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>160</td>
<td>1.7</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>109</td>
<td>2.5</td>
</tr>
<tr>
<td>P2</td>
<td>1</td>
<td>3988</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2655</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1857</td>
<td>2.1</td>
</tr>
</tbody>
</table>

Table 2. Layout comparisons

<table>
<thead>
<tr>
<th>Circuit</th>
<th>PARAGRAPh (Our) (WL(x1000))</th>
<th>QUADRlESECTION (Suarias-Kedem) (WL(x1000))</th>
<th>LC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z1</td>
<td>518</td>
<td>15787</td>
<td>567 224 17252</td>
</tr>
<tr>
<td>Z2</td>
<td>12722</td>
<td>232974</td>
<td>16873 1392 222567</td>
</tr>
<tr>
<td>Z3</td>
<td>7934</td>
<td>57320</td>
<td>8751 1534 69953</td>
</tr>
<tr>
<td>P1</td>
<td>1918</td>
<td>28531</td>
<td>1928 388 28341</td>
</tr>
<tr>
<td>P2</td>
<td>16080</td>
<td>116653</td>
<td>15705 1464 17018</td>
</tr>
</tbody>
</table>

Table 3. Speedup measurements

<table>
<thead>
<tr>
<th>Circuit</th>
<th>Number of Processors</th>
<th>RunTime (s)</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Z1</td>
<td>1</td>
<td>124</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>69</td>
<td>1.8</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>47</td>
<td>2.6</td>
</tr>
<tr>
<td>Z2</td>
<td>1</td>
<td>1648</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1016</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>721</td>
<td>2.3</td>
</tr>
<tr>
<td>Z3</td>
<td>1</td>
<td>3062</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1900</td>
<td>1.6</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1318</td>
<td>2.3</td>
</tr>
<tr>
<td>P1</td>
<td>1</td>
<td>270</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>160</td>
<td>1.7</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>109</td>
<td>2.5</td>
</tr>
<tr>
<td>P2</td>
<td>1</td>
<td>3988</td>
<td>1.0</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2655</td>
<td>1.5</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>1857</td>
<td>2.1</td>
</tr>
</tbody>
</table>