On the Design of a Parallel Algorithm for VLSI Layout Compaction

K. Thulasiraman, M.A. Comeau, R.P. Chalasani, A. Das

Dept. of Electrical and Computer Engineering

and

J.W. Atwood

Dept. of Computer Science

Concordia University
1455 de Maisonneuve W.
Montréal, Canada
H3G 1M8

ABSTRACT

In this paper we present the essential features of an approach for the design of a parallel algorithm for the layout compaction problem. We begin with a formulation of the problem presented by Yoshimura in [4]. This formulation is in terms of the dual transshipment problem. Our approach to the solution of the dual transshipment problem involves repeated applications of three basic steps, namely, testing feasibility, shortest-path computations and performing concurrent pivot operations. Our discussion is in terms of marked graph concepts and results presented in [5], [6]. Our approach can also be used in the study of the relative placement problem discussed in [7] by Mlynski and Weiss.

I. INTRODUCTION

VLSI designers have come to rely heavily on automatic layout tools. Therefore it has become necessary that these tools have the ability to do a good job minimizing the final area and other cost-critical factors. But better quality comes at the price of much longer computation time. One way to alleviate this problem is to take advantage of fast commercially available parallel computers. Thus for different steps in the layout phase of the VLSI design process, parallel algorithms which are suitable for implementation on a MIMD multiprocessor and which result in considerable savings in computing time are called for. Recent work on the design of multiprocessor based algorithms for certain phases of the layout problem may be found in [1]. [2], [3].

In this paper, we are concerned with the design of a parallel algorithm for the layout compaction problem. The paper is organized as follows. In Section II, we present a brief review of literature on the compaction problem. We also present the work of Yoshimura [4] wherein he formulates the problem as a dual transshipment problem. The remainder of the paper is concerned with our research towards design of distributed and parallel algorithms for the dual transshipment problem. Our approach is based on results and concepts from the theory of marked graphs presented in [5], [6].

II. LAYOUT COMPACTION

Compaction is the CAD tool used to pack rough sketches or symbolic diagrams to produce IC layouts. Manual compaction is tedious, time-consuming, and error-prone; automated compaction tools can greatly shorten the layout design cycle. The aim of layout compaction can be stated as follows: Starting from an initial layout and without changing its topology, a final mask layout has to be achieved with a minimum chip area and consistent with design rules. The restriction to invariance of topology is necessary in order not to render the previous steps of placement and routing obsolete. It is achieved by maintaining relative neighbors, i.e. adjacency of layout elements. Elements are either on the bottom-most mask level, e.g. diffusion windows, or on higher design levels, e.g. transistors and cells. These elements are not allowed to jump across each other during compaction.

Most of the available compactors are based on some one-dimensional approach which solves the two-dimensional compaction problem by two one-dimensional procedures, i.e. a horizontal compaction and a vertical compaction. These two procedures are applied successively. Thus, the layout elements are moved in one direction at a time changing either their x-coordinates only during horizontal compaction or their y-coordinates only during vertical compaction.

A simultaneous compaction in both the horizontal and the vertical directions is preferable, since it avoids the tradeoff of moving an element in either direction. Although a general two-dimensional compaction strategy is not yet available, all attempts to treat x- and y- positions simultaneously at least during some part of the compaction procedure shall be classified as two-dimensional approach.

In our work, we will be primarily concerned with the one-dimensional approach. Of the several strategies based on the one-dimensional method, the constraint graph approach seems to be most promising from the point of view of parallel implementation. This approach consists of two main steps: (1) build the constraint graph to indicate the relative positions and the minimum distance required among the elements, (2) solve the constraint graph to minimize the chip area using the longest path method. The basic constraint-graph approach separates the compaction problem into two independent compactions, one in the X-direction and the one in the Y-direction. During X-compaction, elements can only move strictly horizontally to the left, and during Y-direction elements can only move strictly vertically to the bot-

CH2868-8/90/0000-0352$1.00 © 1990 IEEE

tom. Simultaneous compaction in both directions have been considered in the literature. In addition to compressing spaces, the compaction process may expand the input to resolve design-rule violations. The user can do as many X- and Y-compactions as necessary. To satisfy all spacing constraints, each node element must be at least its longest path's length away from the boundary element, but it does not need to exceed that distance. Hence, a solution of the maximum-packing problem is that each node should be at that distance, assuming the boundary element is located at $x = 0$. Since all spacing constraints are satisfied by this solution, X-compaction also corrects design-rule violations in the X-direction that may have been in the violations. Overall, the constraint-graph model offers better flexibility than the other models and still allows for efficient implementation.

Our work on parallel algorithm design will start with a formulation of Yoshimura [4] described below.

Given : Initial placement of blocks, horizontal wire segments and vertical wire segments,

Subject to: Reservation of the relative positions between blocks and horizontal wire segments. No overlaps between blocks and wire segments.

Minimize: Chip height and total wire length in Y-direction.

First a directed graph G(V,E) is constructed, where V and E are the node set and the edge set, respectively. Each node corresponds to a block upper edge, a block lower edge or a horizontal wire segment. Using this graph, a minimum height layout can be obtained by calculating a "constrained longest distance tree".

The total wire length minimization problem is formulated as a linear programming problem. The constraints are described as follows: $u_i - u_j > d_{ij}$ where u_i and u_j are y-coordinates for nodes i and node j, respectively and d_{ij} is a constant. An optimum solution is constructed using a variation of the simplex method.

It can be seen that the above formulation of the compaction problem is in terms of the dual transshipment problem. We also note that Mlynski and Weiss [7] formulate the relative placement problem in terms of the dual transshipment problem. Thus the approach we shall be presenting in the following sections will also be applicable in the study of the relative placement problem. For a good treatment of standard algorithms for network optimization problems, [8] may be referred.

III. Dual Transshipment Problem: A New Approach

Given a marked graph G = (V,E) on n nodes and m edges, with the token vector M_0 associated with the edge set E, let A denote the incidence matrix of G. Then the dual transshipment problem is a linear program defined as follows:

$$\text{minimize } WY$$

$$subject\ to\ A^t Y \geq -M_0 \quad (1)$$

$$Y \geq 0 \quad (2)$$

where Y is a column vector of dimension n and W is a row vector also of dimension n. Vectors W and M_0 are specified. M_0 is called a marking of G and the elements of M_0 are called tokens. Note that unlike in traditional marked graphs, we do not restrict tokens to non-negative integers.

A vector $Y \geq 0$ which satisfies (1) is called a feasible solution of the dual transshipment problem. If such a vector exists, then the given dual transshipment problem is feasible. So, the first step in the solution of the dual transshipment problem is to test the feasibility of the problem. We now present an algorithm for this problem. In the following, the token of a directed path is given by the sum of the tokens of the edges on the path. Similarly a token of a directed circuit is defined.

Theorem 1:

The dual transshipment problem is feasible if and only if the marked graph G has no directed circuit of negative token. //

Let d_{ij} denote the token of a minimum-token directed path from node i to node j. Let

$$\gamma_i = \max\{0, -min_j\{d_{ij}\}\},\ i = 1,2,\ldots,n \quad (3)$$

Theorem 2:

The vector $Y = (\gamma_1, \gamma_2, \ldots, \gamma_n)^t$ is a feasible solution of the dual transshipment problem if the marked graph G has no directed circuit of negative token. //

Proof of the above theorems may be found in [9].

To present an algorithm which obtains the feasible solution given in the above theorem, let us define the node firing operation as follows. Firing x times a node v refers to the operation of adding x to the token of every outgoing edge at v and subtracting x from the token of every incoming edge at v. In the following, M(e) denotes the token of edge e. Firing number of a node refers to the number of times the node has been fired. To start with all the firing numbers are zero. Note that in our algorithm the value of y_i at any time will in fact be equal to the firing number of the node at that time.

Algorithm FEASIBLE:

1 Let M = M_0

2 While there exists an edge $e = (i,j)$ such that M(e) < 0, fire node i, -M(e) times, updating M.

Theorem 3:

If the marked graph has no negative token directed circuits under the token vector M_0, then algorithm FEASIBLE terminates in a finite number of steps after firing each node i exactly γ_i times.

Proof of correctness and termination of this algorithm may be found in [9].

We next present an implementation of the above algorithm which achieves a time complexity of O(mn). The main steps in this implementation are:

Step 1: For each i, set $\gamma_i = 0$. Let M_0 be the initial token vector associated with the edge set.

Step 2: For each i, compute $\sigma_i = \max\{0, -min_j \{M(i,j)\}\}$

Step 3: Fire each node i, σ_i times and update γ_j by adding σ_i to its current value (Note that firing results in updating M also).

Step 4: If σ_i for all i, STOP. ELSE return to Step 2.

A distributed/parallel implementation of the above algorithm which achieves a time complexity of O(n) and message complexity of O(mn) is given in [9], In this implementation each node is associated with a single processor and information is communicated from one processor to another through messages.

Suppose that the given dual transshipment problem is feasible. Then, after an application of Algorithm FEASIBLE, all the tokens associated with the edges will be non-negative. Let the corresponding graph be G'. From this point onwards, our approach is to decrease the value of the objective WY as much as possible until optimality is reached. This is achieved by firing the nodes in an appropriate manner without ever allowing the tokens to become negative. Thus we fire only negative weight nodes. This is repeated until no further firing of these nodes is possible. Let the graph at this point be denoted as G''. Note that in G'' at each negative-weight node, there will be at least one edge with zero token, incident into the node.

Also we show that each negative-weight node v_i would have been fired exactly f_i times where f_i is the token in G' of a shortest path to v_i from a positive weight node. So to avoid redundant firings we compute in G' the value of f_i's for all the negative weight nodes and then fire the nodes accordingly. This will transform the graph G' to G''. Note that this step involves only shortest path computations and can be done very efficiently using the distributed/parallel algorithm presented in [10]. We shall call this as Algorithm SHORT-PATH.

Consider now, the graph G''. A number of edge tokens in G'' will be zero. The subgraph of G'' induced by the zero-token edges may not be connected. In that case, let the connected components of this subgraph be G_1, G_2, ..., G_k. We then construct the graph G''' as follows. Node i in G''' represents G_i and the edge e_{ij} (directed from node i to node j) will be assigned the smallest of the tokens of all edges directed from G_i to G_j.

Next we compute the weight of each node (which is now a cluster of nodes) in G''' given by the sum of the weights of all the nodes in the corresponding cluster. Then we apply algorithm SHORT-PATH to compute for each negative-weight node its shortest path from a positive weight node and then fire these nodes by the appropriate amounts. Note that firing a cluster x times results in adding x to the current firing numbers of all the nodes in the cluster.

We repeat the above process until all nodes coalesce into a single cluster. At this point we will have obtained a basic feasible solution (represented by a spanning tree) of the dual simplex. We now test the optimality of the solution using the simplex optimality criterion. Suppose the solution is not optimal. Then we determine for each branch (i,j) of the spanning tree (representing the basic solution) the corresponding fundamental cutset. Let the corresponding vertex partition be $(V_i, \overline{V}_i)$. Assume without loss of generality that the node i is in V_i. Then V_i will be called the fundamental cluster corresponding to the branch (i,j). If the weight of the cluster V_i is negative then we could fire V_i to decrease the value of the objective. Firing a fundamental cluster V_i is in fact the same as a simplex pivot operation with respect to the branch (i,j). Firing all negative weight fundamental clusters may result in producing negative tokens. So, we have designed a strategy to concurrently fire certain negative weight fundamental clusters in an appropriate manner so that no token will become negative during this process.

At the end of above steps, the solution may not be basic. In such case, we repeat the above process until an optimum basic feasible solution is obtained.

Thus summarizing, our approach consists of the following main steps. We start with the given graph and the associated edge tokens prescribed by M_0.

Step 1: We apply Algorithm FEASIBLE to test feasibility of the problem. At the end of this step, all the edge tokens will be non-negative.

Step 2: We then fire negative weight nodes as much as possible with a view to decreasing the objective function. This process can be performed very efficiently using Algorithm SHORT-PATH. When no more firings of negative weight nodes is possible, the nodes will partition into clusters. At this point, we fire negative weight clusters as much as possible. Again, this can be done efficiently by constructing a smaller graph in which a node represents a cluster and then applying Algorithm SHORT-PATH on this new graph.

Step 3: We repeat step 2 until we obtain a basic solution (represented by a spanning tree) of the dual simplex. At this point, we test the optimality of the solution using the simplex optimality criterion. If the solution is not optimal we fire concurrently in an appropriate manner negative weight fundamental clusters and decrease the value of the objective as much as possible.

At the end of Step 3, the solution may not be basic. We now repeat steps 2 and 3 until optimality is reached.

The interesting features of the above approach are:

i) No auxiliary graph is constructed to test feasibility.

ii) Node and cluster firing operations of step 2 can be performed efficiently using Algorithm SHORT-PATH.

iii) Several simplex pivot operations are performed concurrently in step 3. The classical simplex approach does not permit concurrent pivots because these operations will destroy basicness of the solution.

iv) If the solution at the end of step 3 is not basic, step 2 will convert it to a basic solution with an objective value less than that of the previous basic solution. In other words, step 2 converts a non-basic solution to a basic one without ever increasing the value of the objective function.

This algorithm is now under implementation suitable for execution on the Hypercube computer. We are currently trying to port this -dual algorithm to the Cosmic Environment and Reactive Kernel systems (CE/RK) [11] which support a multi-process message passing environment. The CE/RK environment provides uniform communication between processes independent of their actual location. Since our algorithm consists of a collection of processes which communicate with each other by passing messages it will be fairly easy to port this algorithm to the CE/RK environment. We plan to study the run-time characteristics of the parallel algorithm under these conditions.

IV. SUMMARY

In this paper we have presented the essential features of a parallel algorithm for the layout compaction. Starting with a formulation (in terms of the dual transshipment problem) we have designed a parallel algorithm which in addition to shortest-path computations, involves concurrent pivot operations. This algorithm can also be used to study the relative placement problem studied in [7]. The parallel algorithm is now under implementation.

REFERENCES

[1] R.J. Brouwer and P. Banerjee, "*A Parallel Simulated Annealing Algorithm for Channel Routing on a Hypercube Multiprocessor*", **Proc. Intl. Conf. on Computer Design**, 1988, pp. 4-7.

[2] M.Y. Hseuh, "*Symbolic Layout and Compaction of Integrated Circuits*", U.C. Berkeley, UCB/ERL Report, M79/80, 1979.

[3] J.S. Rose, W.M. Snelgrove and Z.G. Vranesic, "*Parallel Standard Cell Placement Algorithms with Quality Equivalent to Simulated Annealing*", **IEEE Trans. CAD of ICAS**, Mar. 1988, pp. 387-96.

[4] T. Yoshimura, "*A Graph-Theoretic Compaction Algorithm*", **Proc. of ISCAS 85**, 1985, pp. 1455-1458.

[5] K. Thulasiraman and M.A. Comeau, "*Maximum Weight Markings in Marked Graphs: Algorithms and Interpretations Based on the Simplex Method*", **IEEE Trans. on Circuits and Systems**, Vol. CAS-34, Dec 1987, pp. 1535-1545.

[6] K. Thulasiraman and M.A. Comeau, "*Structure of the Submarking Reachability Problem and Network Programming*", **IEEE Trans. CAS**, Jan 1988, pp. 89-100.

[7] B.X. Weiss and D.A. Mlynski, "*A Graph-Theoretic Approach to the Relative Placement Problem*", **IEEE Trans. CAS**, Vol. 35, March 1988, pp. 286-293.

[8] V. Chvátal, "Linear Programming," W.H. Freeman and Company, New York, 1983.

[9] M.A. Comeau, K. Thulasiraman and K.B. Lakshmanan, "*An Efficient Asynchronous Distributed Protocol to test Feasibility of the Dual Transshipment Problem*", **Proc. 25th Allerton Conf. on Communication, Control and Computing**, Sep. 1987, pp. 634-640.

[10] K.B. Lakshmanan, K. Thulasiraman and M. Comeau, " *An Efficient Distributed Protocol for the Single Source Shortest Path Problem In Networks with Negative Weights*", **IEEE Trans. on Software Engineering**, May 1989, pp. 639-644.

[11] C.L. Seitz, J. Seizovic and W.K. Su, "*The C Programmer's Abbreviated Guide to Multicomputer Programming*", **Caltech Computer Science Technical Report**, Caltech-CS-TR-88-1, Jan. 1988.