Parallel Network Primal-Dual Method on a Shared Memory Multiprocessor and A Unified Approach to VLSI Layout Compaction and Wire Balancing

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

Concordia University, Montreal

Abstract

We present a unified approach to the layout compaction and wire balancing problem. We show that the layout compaction problem can be solved by an algorithm which also solves the primal-dual initialization problem. We formulate the wire balancing problem as a transshipment problem and show that the results of the compaction problem can also used to initialize the primal-dual method for solving this transshipment problem. In fact wire balancing reduces to applying the primal-dual method on a graph much smaller than the original constraint graph. Distributed/Parallel algorithms for these problems have been implemented on BBN Butterfly machine and are now being tested on benchmark problems.

I. Introduction

Graph and network optimization techniques play a major role in VLSI CAD. The transshipment problem generalizes several of the network optimization problems such as the shortest path, maximum flow and matching problems. For this reason there has been considerable effort in the literature towards designing efficient sequential and parallel algorithms for this problem [3]. All these efforts have resulted in variations of two very efficient sequential approaches for the solution of this problem. They are: **the network simplex** method and **the primal dual** method [1]. A recent paper discusses parallelization of the network simplex method [6]. However the network simplex method does not offer much scope for parallelization because the pivot operation is inherently sequential in nature.

The present paper has two objectives. First, we are concerned with the design of a parallel algorithm for the network primal-dual method. Second, we provide an application of this parallel algorithm by presenting a unified transshipment formulation of the VLSI layout compaction and wire balancing problem.

The paper is organized as follows. In Section II, we present the transshipment problem and the elements of the primal-dual method. In Section III, we present a unified formulation of the layout compaction and wire balancing problem. This formulation is in a form ideally suited for solution by the primal dual method.

II. The Transshipment Problem

Consider a network N with the underlying graph G = (V, E). Some of the vertices in N represent sources and are called **supply** vertices. Some of the others represent demand centers called **sinks**. There may be vertices which neither supply nor demand. These are called **neutral** vertices. The supply or demand at a vertex v_i is denoted by b_i. For a neutral vertex, b_i=0. Each edge (v_i , v_j) is associated with a weight w_{ij}, which represents the cost of transporting a commodity along the edge. Each edge (v_i , v_j) is also associated with a capacity *cap (i , j)* representing the maximum amount of the commodity that the edge can accomodate. Given the supplies available at the sources and the demands at the sinks, the **transshipment problem** is to arrive at a routing pattern for a given commodity so that the demands are satisfied at minimum cost. This problem [1] is a linear programming problem and can be formulated as follows :

minimize : W X

subject to :

$$A^* X = b \tag{1}$$

$$0 \le X \le C \tag{2}$$

where

W = Row vector of edge weights w_{ij}; X = Column vector of edge flows x_{ij}; $A^* = -A$, where A is the incidence matrix of the graph G underlying the network N; and C =

0-8186-3180-5/92 $3.00 © 1992 IEEE

Column vector of edge capacities *cap (i , j)*.

Associated with any linear programming problem there is a **dual problem**. The original problem is then called the **primal problem**. The dual of the transshipment problem has n dual variables $y_1, y_2, \ldots, y_n$. The optimum values for y_i's would maximize the sum $\sum_i b_i y_i$. An important result in linear programming theory is stated next.

If x_{ij}'s and y_i's represent optimum solutions for the primal and dual problems, respectively, then

$$x_{ij} = 0 \text{ , if } y_i - y_j + w_{ij} > 0$$
$$x_{ij} = cap(i\ ,j) \text{ , if } y_i - y_j + w_{ij} < 0. \qquad (3)$$

The above conditions are called the **complementary slackness** conditions.

There are two distinct approaches to the transshipment problem - the primal and primal-dual approaches [1].

The primal-dual approach starts with an X and Y satisfying (2) and (3). It then updates X and Y (without violating (2) and (3)) until X satisfies (1). This approach is quite amenable to distributed and parallel implementations, since it uses the maximum flow and shortest path algorithms as building blocks.

The primal-dual approach consists of three main steps : (i) Initialization, (ii) Updating Y, (iii) Updating X.

In the initialization step a pair of vectors X and Y satisfying (2) and (3) are selected. We can show that this can be achieved by using a modified version of algorithm FEASIBLE of [2]. Essentially this algorithm finds y_i's such that $y_i - y_j + w_{ij} \geq 0$ for all edges (i, j). The remaining two steps can be implementated using a shortest path algorithm and a maximum flow algorithm. Details of a distributed implementation of the primal-dual method and its simulation on a shared memory multiprocessor are given in [7]. This implementation uses the shortest path algorithm of [5] and the maximum flow algorithm of [4].

III. Layout Compaction and Wire Balancing: A Unified Approach

In **layout compaction** one starts with an initial layout and seeks to achieve a final mask layout (without changing the topology) which has minimum chip area and is consistent with the design rules. Invariance of network topology is required in order not to render the previous steps of placement and routing obsolete. At the end of layout compaction relative positions of all the circuit elements will be available. Changing the positions of these elements (to be precise, those elements which lie on a longest path between the chip boundaries) will result in increased chip width. But the positions of the others could be varied without causing design rule violations and yet maintaining compacted chip width. In **wire balancing**, one seeks to achieve minimum overall wire length by adjusting the positions of the elements which do not lie on the longest paths mentioned above.

The constraint graph approach to the above problem [8] proceeds as follows. From the initial layout a graph G = (V, E), called the **constraint graph,** is constructed. We assume that the layout is Manhattan, i.e., edges of each circuit element are either horizontal or vertical. Each vertex of G represents a circuit element or a group of circuit elements that are physically connected. Each vertex v_i is associated with a variable y_i representing the position of the corresponding circuit element. In the following the circuit element corresponding to vertex v_i will also be referred to as v_i. In G, there is an edge between two vertices, if there is a design rule constraint between the corresponding elements. There are three types of constraints: **minimum, maximum** and **equality** constraints.

Minimum constraint of the type $y_j - y_i \geq a$ states that v_i is to the left of v_j and there is a minimum spacing requirement of 'a' units between them. This constraint is represented in G by an edge (v_j, v_i) directed from v_j to v_i with an associated weight w_{ji} of value $-a$.

A maximum constraint of the form $y_j - y_i \leq b$ or equivalently $y_i - y_j \geq -b$ is represented by an edge (v_i, v_j) directed from v_i to v_j with weight w_{ij} of value "b".

An equality constraint can be regarded as a pair of minimum and maximum constraints. Thus an equality constraint will be represented by a pair of oppositely directed edges both with zero weight.

Two special vertices, called the source (v_s) and the sink (v_t), are used to represent the right most boundary and the left most boundary of the layout, respectively. Circuit elements which correspond to vertices with no outgoing edges could be placed at the left boundary, and so we add to G edges directed from each one of these vertices to the sink v_t. These edges have zero weight. For a similar reason, we add to G zero-weighted edges directed from the source v_s to the vertices with no incoming edges. The additional edges so added ensure that the circuit elements will not be moved beyond the left and right boundaries. They play a key role in the wire balancing phase.

Besides weights, edges are also associated with costs to indicate the relative costs of wires. We derive vertex costs from edge costs. Let c_{ij} denote the cost of the edge (v_i, v_j) directed from v_i to v_j. Then the cost γ_i of vertex v_i is given

$$\gamma_i = \sum_{j \in V} c_{ij} - \sum_{j \in V} c_{ji}$$

Here we assume that $c_{ij} = 0$ if there is no wire between v_i and v_j. Thus a negative vertex cost indicates that moving the vertex to the right will decrease the overall wire length and a positive vertex cost indicates that moving a vertex to the right will increase the overall wire length.

Let Y denote the vector of y_i's, W the vector of w_{ij}'s and γ, the vector of vertex costs. Also let $A = -A^*$. Then in layout compaction we seek to obtain a $Y \geq 0$ such that

$$A^t Y \geq -W$$

and that the difference between the largest and the smallest y_i's is as small as possible. Thus we can formulate this problem as an LP problem as follows.

Layout Compaction

$$Minimize \sum_i y_i$$

$$A^t Y \geq -W$$

$$Y \geq 0 \quad (4)$$

Our algorithm FEASIBLE of [2] in fact solves the above problem if there are no negative-weight directed circuits in G. If we assume that there are no such circuits, then the y_i-values obtained at the end of FEASIBLE represent the positions of the different circuit elements. Each vertex v_i with $y_i = 0$ will be at the left boundary and each v_i with the maximum y_i will be at the right boundary. It can be shown that the maximum y_i-value in fact is the length of the most negative length path in G from the source to the sink. In traditional approaches, such a path in fact corresponds to a longest path from the sink v_t to the source v_s.

Let $y_i = \lambda_i$ at the end of Algorithm FEASIBLE. Recall that this algorithm modifies the edge weights at each step. When all the edges are non-negative (that is $\lambda_i - \lambda_j + w_{ij} \geq 0$), the algorithm terminates. Interestingly, at termination the weights of the edges on the most negative directed path from v_s to v_t will all be zero. So by a simple traversal of G starting from v_s, we can identify all the vertices on these most negative paths. Let S_l denote the set of these vertices.

As we mentioned before, our aim in wire balancing is to keep each vertex $v_i \in S_l$ at $y_i = \lambda_i$ and adjust the y-values of the vertices in S_l and achieve minimum wire length. Thus we have the following formulation of the wire balancing problem.

Wire Balancing

$$Minimize \sum_i \gamma_i y_i$$

$$\text{subject to } A^t Y \geq -W \quad (5)$$

$$y_i = \lambda_i, \text{ for } v_i \in S_l, \quad (6)$$

$$y_i \geq 0 \text{ for all } v_i.$$

Constraint (6) can be replaced by:

$$y_i - y_j + w_{ij} = 0 \quad (7)$$

for every edge (v_i, v_j) on a most negative path from v_s to v_t.

Now it is a simple matter to represent each of the above equality constraints by adding to G oppositely oriented edges with weights w_{ij} and w_{ij} between v_i and v_j, and between v_j and v_i and then proceed with the solution of the optimization problem. However, such an approach will result in a significant increase in the size of the graph. In fact, we can considerably reduce the size of the graph as discussed below.

Constraint (7) suggests that in any new solution of the wire balancing problem,

$$y_i = \lambda_i + k, \text{ for } v_i \in S_l$$

where k is a fixed constant. We can take advantage of this property as follows.

We construct a new graph G' by replacing the vertices in S_l by a single new vertex, say, v_l, and then removing all the edges connecting the vertices in S_l. Consider now an edge (v_i, v_j) in G. If $v_i \notin S_l$ and $v_j \notin S_l$, G' will have an edge (v_i, v_j) with weight w_{ij}. If $v_i \in S_l$, then G' will have an edge (v_l, v_j) with weight $(\lambda_i + w_{ij})$. If $v_j \in S_l$, then G' will have an edge (v_i, v_l) with weight $(-\lambda_j + w_{ij})$. If after this contruction, there are parallel edges, in G', between two vertices then we can remove all of them except the one with the smallest weight. Let the new graph be denoted by G''. In G'' the weight of every vertex $v_i \notin S_l$ will be equal to γ_i. For the new vertex v_l representing S_l, the cost γ_l will be given by

$$\gamma_l = \sum_{v_i \in S_l} \gamma_i$$

We now have the following formulation of the wire balancing problem; where γ_i, y_i, w_{ij} refer to the quantities defined for G''.

Wire Balancing

$$Minimize \sum_i \gamma_i y_i$$

$$\text{subject to } A^t Y \geq -W \qquad (8)$$

$$Y \geq 0$$

where A is the incidence matrix of G''.

The dual of the above problem is:

$$Maximize \sum_{i,j} (-w_{ij}) x_{ij}$$

$$\text{subject to } AX \leq \Upsilon \qquad (9)$$

$$X \geq 0$$

where x_{ij} is the flow variable associated with edge (v_i, v_j) and γ is the vector of γ_i's. Equivalently, we have

$$Minimize \sum_{i,j} w_{ij} x_{ij}$$

$$\text{such that } A^* X \geq -\Upsilon \qquad (10)$$

$$X \geq 0$$

Recall that $A^* = -A$.

We can convert the above LP formulation into the form of the transshipment problem presented in Section II. For this purpose we first add to G a dummy vertex v_d. Then, for each vertex v_i in G'' add an edge (v_i, v_d) with zero cost.

We have thus cast the wire balancing problem into a standard transshipment problem. The primal-dual method discussed in Section II and its parallel implementation can now be used to solve the wire balancing problem. A most important observation at this point is that we can initiate the primal dual with

$y_i = \lambda_i$ for all $v_i \notin S_l$, and

$y_l = 0$ for the vertex v_l representing S_l.

Thus the y_i-values obtained at the end of our compaction phase can be used to initialize the primal dual method to be applied to solve the wire balancing problem!

The y_i-values at the end of the application of the primal dual method will provide the relative locations of the elements in the final layout.

IV. Summary

We have presented a unified approach to the layout compaction and wire balancing problem. We have shown that the layout compaction problem can be solved by an algorithm which also solves the primal-dual initialization problem. We have formulated the wire balancing problem as a transshipment problem and have shown that the results of the compaction problem can also used to initialize the primal-dual method for solving this transshipment problem. In fact wire balancing reduces to applying the primal-dual method on a graph much smaller than the original constraint graph. Distributed/Parallel algorithms for these problems have been implemented on BBN Butterfly machine and are now being tested on benchmark problems.

V. References

[1] Chvatal, V., *Linear Programming*, Freeman Company, Potomac, Maryland, 1983.

[2] Comeau, M.A., K. Thulasiraman and K.B. Lakshmanan, "An Efficient Asynchronous Distributed Protocol to Test Feasibility of the Dual Transshipment Problem," Proc. Allerton Conf. on Communication, Control and Computing, Sept. 1987.

[3] Goldberg, A.V., "Efficient Graph Algorithms for Sequential and Parallel Complexity," *Ph.D. Thesis*, Lab. for Computer Science, M.I.T., Cambridge, MA, 1987.

[4] Goldberg, A.V., and R.E. Tarjan, "A New Approach to the Maximum Flow Problem," *J. ACM*, Vol. 35, 921-940, 1988.

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

[6] Peters, J., "The Network Simplex Method on a Multiprocessor," *Networks*, Vol. 20, No. 7, 845-859, 1990.

[7] Thulasiraman, P., "A Distributed Protocol for the Network Primal-Dual Method and Simulation on a Shared Memory Multiprocessor," *M.A.Sc. Thesis*, Dept. of Elect. & Comp. Engg., Concordia University, Montreal, 1991.

[8] Yoshimura, T., "A Graph-Theoretic Compaction Algorithm," *Proc. Intl. Symp. Circuits and Systems*, 1445-1458, 1985