A Path-based Timing-driven Quadratic Placement Algorith*

Wenting Hou, Xianlong Hong, Weimin Wu and Yici Cai
Department Computer Science and Technology
Tsinghua Univ Beijing 100084 China
Tel:86-(10)62785564
Fax:86-(10)62781489
e-mail: houwt98@mails.tsinghua.edu.cn

Abstract – This paper presents a path-based timing-driven quadratic placement algorithm. The delay of the path acts as the timing constraints. In the global optimization step, it tries to satisfy the timing constraints. In the partition step, it tries to decrease the cut number of critical paths. It has some special skills, such as decrease the delay on the longest path, pad assign, to decrease the delay further. Results show this algorithm can make the timing behavior improve more than 20%.

I INTRODUCTION

With the development of VLSI technology, the feature size of semi-conductor process becomes smaller and smaller. The interconnect delay dominates the total delay. Decreasing the wire delay is significant in physical design. Placement is an important step in physical design, whose result affects the quality of routing deeply. So controlling the wire delay in placement is necessary and wise.

Timing-driven placement schemes have been classified as net-based and path-based [1, 2]. Net-based algorithms [3, 4, 5] assign delay budget on each net and seek to control the delay on the net. Net-based approaches usually over constrain the placement. Path-based algorithms [6, 7, 8] over the shortcoming of net-based algorithm, but they fail in controlling numerous paths.

There are many timing-driven placement algorithms based on random search. [4] follows the genetic paradigm. [7] is based on simulated annealing. Because it is impossible to enumerate all the paths, [4][7] can only control the first several longest paths. With the increasing of the number of the controlled paths, the running time of algorithms is enlarged. These algorithms aren't suit to large circuits.

[2] is a timing-driven placement based on partition and optimization. During the partition step, it first allocates the critical cells into the regions to minimize the cut number on the critical paths. In the optimization step, it tries to get a minimum total quadratic wire length and maintain the partition result of the partition step. As we know, partition can only minimize the delay in local view. It can't optimize the whole circuit in global view. And the optimization step can only maintain the partition result. It can't optimize the delay of the circuit further.

[5] adopts quadratic programming method to get the minimum of the total wire length. It is a net-based timing-driven placement. It adds different weight to each net. It can get a good result on timing behavior in global view. But it is careless the timing behavior in the partition step. If the critical path is partition into several different regions, the total delay of the path is increased heavily, though it is added a large weight to decrease the total wire length.

In this paper, we present a path-based timing-driven placement algorithm. It bases on quadratic programming. The delays of critical paths act as the timing constraints to the quadratic problem. It can control the largest path delay and can get the optimized result in global view. To control the paths' delay doesn't increase the timing complexity of the algorithm. In the partition step, it tries to minimize the number of cut on the critical path to avoid enlarging the path delay. And it has a skill to decrease the largest delay of the circuit further.

The remainder of the paper is organized as follows: In the second part, it describes the fundamental theories. In the third part, it presents the global optimization. In the fourth section, it describes the timing-driven partition. In the fifth section, there are some skills to decrease delay further. In the sixth and seventh section, the experimental result and conclusion.

II. FUNDAMENTAL THEORIES

The circuit can be presented by a hypergraph $G(V, E)$, $V=V_1 \bigcup P$. One part of the vertex set, $V_1=\{v_1,v_2,...,v_n\}$, represents all of the movable cells in the circuit. The other part of the vertex set, $P=\{p_1,p_2,...,p_q\}$, represents all of the pads in the circuit, such as the primary input and primary output pads. The hyperedge set $E=\{e_1,e_2,...,e_m\}$ represents the connection of the circuit. Every edge is assigned a weight $w(e)$.The coordinates of $cell_i$ are (x_i, y_i).

The objective function of the quadratic optimization is the weighted sum of the squared lengths of all nets:

$$\phi = \sum_{e\in E} L_n w_n = \sum_{e\in E} w_n \sum_{i,j\in n} (x_i - x_j)^2 + (y_i - y_j)^2 \quad (1)$$

The objective function can be written in matrix form as in

$$\mathbf{\Phi(x,y) = x^T C x + d_x^T x + y^T C y + d_y^T y} \quad (2)$$

The vector $\mathbf{d_x}$ and $\mathbf{d_y}$ is generated by the connection between the cells and pads.

The n x n connection matrix $C=(c_{ij})$ for G represents the relationship between edges and vertices. For each edge $e_k=(v_i,v_j) \in E$, $c_{ij}=-w(e_k)$ and $c_{ji}=-w(e_k)$; the diagonal entry c_{ii} is determined as follows:

$$c_{ij} = \begin{cases} -w(e_j) & e_j = (v_i, v_j) \in E \wedge i \neq j \wedge v_j \in V_1 \\ \sum_{j=1 \cap j\neq i}^{n} \left| w(e_j) \right| & e_j = (v_i, v_j) \in E \wedge i = j \\ 0 & otherwise \end{cases} \quad (3)$$

*Supported by the China National Science Foundation 60121120706, National Science Foundation of USA CCR-0096383, Hi-Tech Research and Development (863) Program of China 2002AA1Z1460 and 973 national key research project (No.G1998030403).

In order to distribute the cells evenly on the chip, distribution constraints are added. In every partition step, the father region is partitioned into two son regions. And the cells belonging to the father region are assigned to the son regions. The partition level is the level of the partition tree. The number of regions in l^{th} partition level is 2^l.

In each partition level, the relationships between cells and regions are fixed, so the distribution constraints are fixed. Assume the set of cells in region *r* is τ_r and (μ_r, ν_r) is the center of region *r*. The distribution constraints can be represented as

$$\frac{\sum_{i\in\tau_r} x_i}{|\tau_r|} = \mu_r \qquad \frac{\sum_{i\in\tau_r} y_i}{|\tau_r|} = \nu_r \tag{4}$$

Combining the objective function in Eq.2 and the constraints in Eq.4, a *linearly constrained quadratic programming problem (LQP)* is obtained:

$$LQP{:}\min\{\mathbf{\Phi}(\mathbf{x},\mathbf{y}) = 1/2\mathbf{x}^{\mathbf{T}}\mathbf{C}\mathbf{x} + \mathbf{d}_{\mathbf{x}}^{\mathbf{T}}\mathbf{x} + 1/2\mathbf{y}^{\mathbf{T}}\mathbf{C}\mathbf{y} + \mathbf{d}_{\mathbf{y}}^{\mathbf{T}}\mathbf{y} \mid \mathbf{A}^{(\mathbf{l})}\mathbf{x} = \boldsymbol{\mu}^{(\mathbf{l})}; \mathbf{A}^{(\mathbf{l})}\mathbf{y} = \boldsymbol{\nu}^{(\mathbf{l})}\} \tag{5}$$

In l^{th} partition level, the size of matrix ***A*** is a 2^l x *n*. In matrix ***A***, the a_{ij} equal to $1/\tau_j$, if $cell_i$ belongs to $region_j$, or a_{ij} is zero. The number of total non-zero entries is *n*.

A timing-analysis step is adopted to identify the critical paths. The timing constraints are added to *LQP* according to the critical paths.

In the timing-analysis step, the hypergraph *G* is searched from primary input vertexes and registers vertexes first to get the latest arriving time of each vertex. Then the hypergraph *G* is searched from the primary output vertexes and registers vertexes to get the earliest requiring time. The slack of each vertex is equal to the requiring time minus the arriving time. If the slack is negative, the vertex is critical. The critical path is the longest path from each pair of the primary input/register vertexes and the primary output/register vertex whose delay exceed the users' required cycle time. Critical paths are found. With the timing-analysis, the critical paths can be updated dynamically.

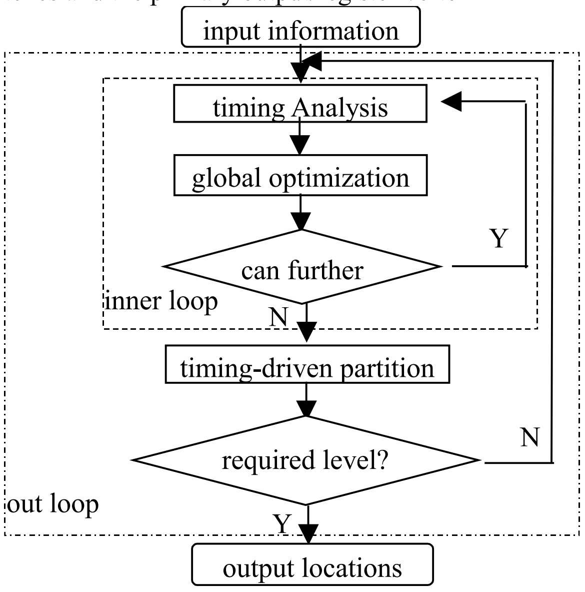

Figure 1 main flow of timing-driven placement

The processing procedure of our algorithm is illustrated in Figure1. The timing-analysis step and the global optimization step are iterated along the algorithm. The iteration is the inner loop in Figure1. The iteration will continue until the timing behavior can't be improved. Then it does the timing-driven partition. The partition will not stop until reach the required partition level.

III. Global optimization

The Elmore delay model and quadratic wire length model are used to describe the interconnect delay. Some mathematics approximate methods are used on the interconnect delay. The constraint of one path's total delay can be written as follows:

$$\sum_{n\in path} f(R_{unit}, C_{unit}, C_{pin}) \sum_{e_{ij}\in n} (x_i - x_j)^2 + (y_i - y_j)^2 \le T \tag{6}$$

where, R_{unit}, C_{unit} are the unit resistance and capacitance respectively; C_{pin} is the driven capacitance of the net; *T* is the cycle time. After the timing constraints are added to the *LQP*, the global optimization is as follows:

$$\begin{aligned} &\min \mathbf{\Phi}(\mathbf{x},\mathbf{y}) = 1/2\mathbf{x}^{\mathbf{T}}\mathbf{C}\mathbf{x} + \mathbf{d}_{\mathbf{x}}^{\mathbf{T}}\mathbf{x} + 1/2\mathbf{y}^{\mathbf{T}}\mathbf{C}\mathbf{y} + \mathbf{d}_{\mathbf{y}}^{\mathbf{T}}\mathbf{y} \\ &s.t\ \ \mathbf{A}^{(\mathbf{l})}\mathbf{x} = \boldsymbol{\mu}^{(\mathbf{l})} \quad ; \quad \mathbf{A}^{(\mathbf{l})}\mathbf{y} = \boldsymbol{\nu}^{(\mathbf{l})} \\ &\sum_{n\in path_p} f(R_{unit}, C_{unit}, C_{pin}) \sum_{e_{ij}\in n} (x_i - x_j)^2 + (y_i - y_j)^2 \le T \\ &\quad p = 1 \ldots q \end{aligned} \tag{7}$$

where q is the total number of critical paths.

Lagrange Successive Over-Relaxation is used to convert Eq.7 into unconstraint problem as follows (only in *x* direction):

$$\min \mathbf{\Psi}(\mathbf{x}) = \mathbf{\Phi} + \boldsymbol{\alpha}(\mathbf{A}^{(\mathbf{l})}\mathbf{x} - \boldsymbol{\mu}^{(\mathbf{l})}) + \boldsymbol{\beta}\Big(\sum_{n\in path_p} f(R_{unit}, C_{unit}, C_{pin}) \sum_{e_{ij}\in n} (x_i - x_j)^2 - T\Big) \tag{8}$$

Eq.8 is still a convex problem. After solving this problem, the global optimization point can be obtained. Letting the gradient equal to zero yields the equations:

$$\begin{aligned} &\nabla_x \mathbf{\Psi} = 0 \qquad \mathbf{C}\mathbf{x} + \mathbf{d}_{\mathbf{x}} + \mathbf{A}^{(\mathbf{l})\mathbf{T}}\boldsymbol{\alpha} + \boldsymbol{\beta}\mathbf{F}\mathbf{x} = 0 \\ &\nabla_\alpha \mathbf{\Psi} = 0 \qquad \mathbf{A}^{(\mathbf{l})}\mathbf{x} - \boldsymbol{\mu}^{(\mathbf{l})} = 0 \end{aligned} \tag{9}$$

which can be rearranged to give the linear system

$$\begin{bmatrix} \mathbf{C} + \boldsymbol{\beta}\mathbf{F} & -\mathbf{A}^{(\mathbf{l})\mathbf{T}} \\ -\mathbf{A}^{(\mathbf{l})} & \mathbf{0} \end{bmatrix} \begin{pmatrix} \mathbf{x} \\ \boldsymbol{\alpha} \end{pmatrix} = -\begin{pmatrix} \mathbf{d}_{\mathbf{x}} \\ \boldsymbol{\mu}^{(\mathbf{l})} \end{pmatrix} \tag{10}$$

where ***F*** is generated by the timing constraints. ***GMRES*** is used to solve Equ.10. Then the position of every cell can be gotten. The positions can satisfy the distribution constraints and can obtain the minimum total wire length, but can't guarantee to satisfy the timing constraints. So after solve the linear system, vector **β** will be modified according to the timing analysis result. The inner loop in Figure1 will iterate many times until the timing behavior at this partition level can't be improved.

βF will not destroy the sparsity of Matrix C. Because βF only adds some values to the non-zero entries of C and doesn't change zero entries. So after adding the timing constraints, the timing complexity of solving the linear system will not increase.

In fact, βF equal to add weight to each net of every critical path. If a net is passed by several critical paths, the net is very important. Decreasing the delay on the net will decrease several critical paths' delay. In our algorithm, the weight of the net will be added several times according to different critical paths. So in the global optimization, the net will have a shorter distance and a shorter delay.

In the linear system Equ.10, the gradient of α is also set to be zero. So after solving the linear system, the distribution constraints will be satisfied automatically. So whatever the timing constraints add to the linear system, the cells will be assigned to the chip evenly. This method will escape the problem that the cells converge into blocks since over timing constraints are added to the linear system.

IV. TIMING-DRIVEN PARTITION

In the partition step, the father region will be partition into two son regions along a partition line. And the cells belonging to the father region will be assigned into the son regions according to the positions of the cells. That will generate a partition tree, as shown in Figure 2.
The distribution constraints will be set according to the partition result. The distribution constraints resemble a pull, which will pull the cells to be separated evenly on the chip. And the timing constraints resemble another kind of pull, which will pull the cells on the critical path together to shorten the delay of the path.

If cells on one critical path are assigned into different partition regions in the former partition level, it is a disaster to the timing behavior of the circuit. In the former partition level, the timing-driven partition will try to put cells on one critical path into the same region. That is to say, the partition will decrease the cut number on critical paths.
When one cell on the critical path locates in the different region from its forward and backward cells, just as in Figure 3a, in the partition step, the cell will be adjusted to region A, though it locates in region B. Then the cell will be added a pull to the center of region A because of the distribution constraints. With the iteration process, the shape of the critical path will be looked like in Figure 3b.

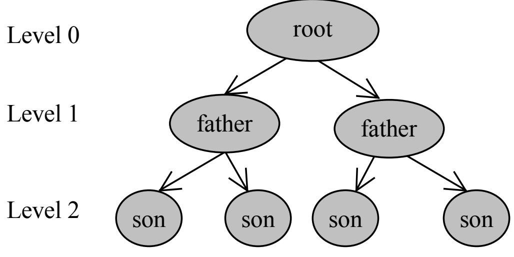

Figure 2 three level partition tree

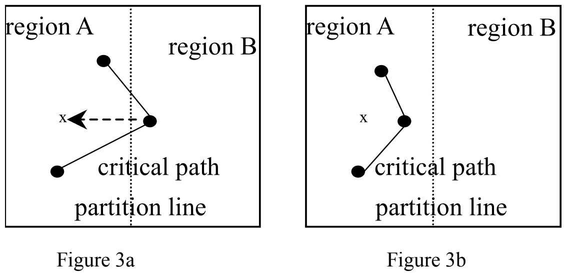

Figure 3 one critical in different regions

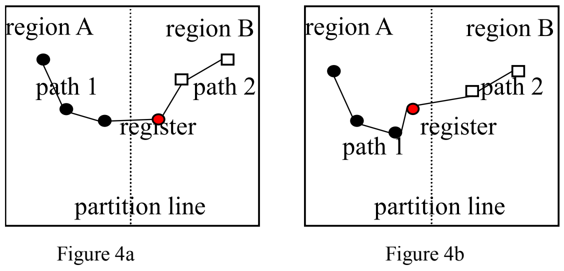

Figure 4 allocating the register

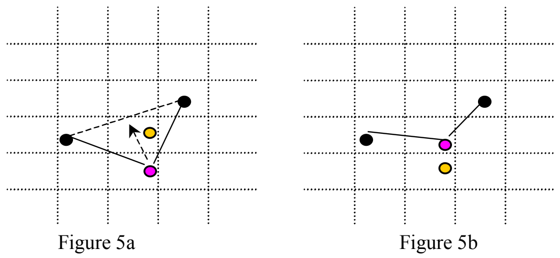

Figure 5 partition step in the later level

When a cell is the terminal of two critical paths and the two paths locate in the different regions, usually which is a register, as shown in Figure 4a, the partition step will compare the two paths. It will select a more critical path and put the register into the same region with the path. As the iteration process, the result of the register's position will be look like in Figure 4b.

In the later partition level, the regions become smaller and smaller and the critical path can't be put into one region. In the partition step, the cells on the critical path will exchange with other cells those are nearer to the line from the backward cell to the forward cell to make the cells on the critical path to be aligned. For example, in Figure 5a, the red cell finds a yellow cell, which is nearer to the line from the backward cell to the forward, and exchange with the yellow cell. Later the critical path will be shortened.

The timing-driven partition is a heuristic method. It can decrease the cut number of the partition line on the critical path and avoid making critical paths very long.

V. OTHER SKILLS

Some other skills can be used in the timing-driven placement algorithm to decrease the maximum delay of the chip further.

A. Process for the longest path

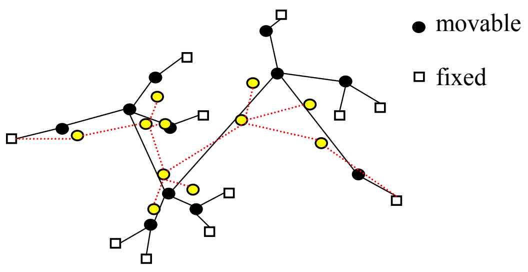

Figure 6 result of the procedure for the longest path

The process for decreasing the longest path's delay can be done after the inner loop. This procedure will help to get the good partition result.

In the procedure, the objective is to decrease the total wire length of the longest path. As shown in Figure 6, the cells those are the terminals of the net on critical path are deemed as movable cells, others are fixed cells.

The problem can be formalized as Eq.11:

$$\min \mathbf{P}(\mathbf{x},\mathbf{y}) = \mathbf{1/2x^TGx + b_x^Tx + 1/2y^TGy + b_y^Ty}$$

$$\nabla_x \mathbf{P}(\mathbf{x},\mathbf{y}) = 0 \qquad \mathbf{Gx = -b_x^T} \qquad (11)$$

$$\nabla_y \mathbf{P}(\mathbf{x},\mathbf{y}) = 0 \qquad \mathbf{Gy = -b_y^T}$$

The matrix $\boldsymbol{G}$ is generated by the connection of movable cells on the nets of the critical paths. And vector $\boldsymbol{b_x}$ and $\boldsymbol{b_y}$ are generated by the connection between the movable cells and the fixed cells.

Because there is no constraint in the process, the critical path will converge a little, just as the yellow cells in Figure 6. The total wire length of the nets on the critical path will decrease and the path delay will decrease.

The procedure can do by interleaving the timing-analysis for several times until the maximum path delay can't improve. This procedure can decrease the maximum path delay to a degree.

B. Pad assign

If the primary input and the primary output pads are movable, the procedure of pad assign can be done to decrease the path delay. The procedure can also be added after the inner loop. The positions of pads can be obtained as Equ.11:

$$x = \frac{\sum_{i=1}^{n} w_i x_i}{n} \qquad y = \frac{\sum_{i=1}^{n} w_i y_i}{n} \qquad (11)$$

Assuming a pad is connected with other n movable cells, which have the coordinates (x_i, y_i). w_i is set according to the slack of $cell_i$. After getting the coordinates, the positions of pads will be assigned to the nearest boundary.

VI. Experimental results

We have implemented the algorithm and tested them on Sun V880. We tested some MCNC benchmark. Table1 list the characteristics of these circuits.

The results of wire length driven placement and timing-driven placement are shown in table 2. From the results of table 2, we can see the timing-driven procedure can decrease the maximum delay more than 20%.

TABLE1
THE CHARACTERISTIC OF CIRCUITS

benchmark	#cell	#net
fract	125	147
struct	1888	1920
biomed	6417	5742

TABLE2
RESULTS OF WITH AND WITHOUT TIMING PROCEDURE

benchmark	total wire length		maximum delay	
	wire length driven	timing-driven	wire length driven	timing-driven
fract	49812	61098	31.2	21.4
struct	1240723	1257891	66.4	50.6
biomed	4935677	5111624	40.7	33.1

VII. Conclusion

In this paper, we present a path-based timing-driven quadratic placement algorithm. It adds the timing constraints according to the critical paths to the global optimization problem. This kind of constraints can control the maximum path delay of the circuit. And in the partition step, it tries to assign the cells on one critical path into near regions to avoid enlarging the path delay. The procedure for the longest path and the procedure of pad assign will decrease the maximum delay to some extension.

References

[1] Majid Sarrafzadeh, David A.Knol and Gustavo E.Tellez "A Delay Budgeting Algorithm Ensuring Maximum Flexibiligy in Placement", IEEE transactions on Computer-Aided Design of Intergrated Circuits and Systems. Vol 16 No11 Nov 1997 pages1332-1341

[2] Shih-Lian Ou and Massoud Pedram "Timing-driven Placement Based on Partitioning with Dynamic Cut-net Control", In *ACM/IEEE Design Automation Conference, 2000, pp 472-476.*

[3] R.S.Tsay and J.Koehl, "An Analytic Net Weighting Approach for Performance Optimization in Circuit Placement", In *Proc 28th ACM/IEEE Design Automation Conference, 1991 pp 636-639.*

[4] H.Eisenmann and F.M.Johannes, "Generic Global Placement and Floorplanning," In *35th Proceedings of the ACM/IEEE Design Automation Conference,, pp 269-274,*1998

[5] B.M.Riess and G.G.Ettelt, "SPEED: Fast and Efficient Timing Driven Placement," In *Proc of the IEEE International Symposium on Circuits and System, pp 377-380,*1995

[6] A.Srinivasan, K,Chaudhary, and E.S. Kuh, "RITUAL: A Performance Driven Placement Algorithm," *IEEE Trans. On Circuits and Systems-II: Analog and Digital Signal Processing, Vol 39,No.11, pp 825-840,* November 1992

[7] W.Swartz and C.Sechen, "Timing-Driven Placement for Large Standard Cell Circuits," In *32nd Proc of the ACM/IEEE Design Automation Conference, pp 211-215,* 1995

[8] M.Sarrafzadeh and M.Wang,"NRG: Global and detailed placement," In *Proc.1997 ICCAD, pp532-537.*