第21卷第11期 半 导 体 学 报 Vol. 21, No. 11
2000年11月 CHINESE JOURNAL OF SEMICONDUCTORS Nov., 2000

A New Timing-Driven Placement Algorithm Based on Table-Lookup Delay Model*

YU Hong(于 泓), HONG Xian-long(洪先龙),
YAO Bo(姚 波) and CAI Yi-ci(蔡懿慈)

(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084, *China*)

Abstract: An algorithm is presented for obtaining placements of cell-based very large scale integrated circuits, subject to timing constraints based on table-lookup model. A new timing delay model based on some delay tables of fabricators is first simplified and deduced; then it is formulated as a constrained programming problem using the new timing delay model. The approach combines the well-known quadratic placement with bottom-up clustering, as well as the slicing partitioning strategy, which has been tested on a set of sample circuits from industry and the results obtained show that it is very promising.

Key words: table-lookup; timing; clustering; quadratic placement
EEACC: 2570
CLC number: TN402 **Document code**: A **Article ID**: 0253-4177(2000)11-1129-10

1 Introduction

There have been extensive studies on timing-driven placement in recent years. The approaches toward this problem fall into two main categories: net-based and path-based. In a typical net-based one, potential critical paths and acceptable delays at each cell are calculated, from which slack of a net on each path is derived. The slack of a net gives the upper and lower bounds of the net's size, which serve as the constraints during the subsequent placement phase. The basic idea of a net-based approach can be combined with ideas such as iterative improvement[1], constructive placement[2,3], incremental timing analysis[4], as gives rise to several variations.

Net weighting[2,5] is a technique often used in this approach; which puts a weight on the nets with smaller slack, or gives priority to critical nets. In this way a constrained optimization problem is turned into an unconstrained one. It has been reported that these techniques can achieve favorable results. However, it should also be noted that the net weights are under invetsigation, which makes it difficult to apply proper mathematical

* Project Supported by National Natural Science Foundation of China (No. 69776027) and by 973 National Key Project (No. G1998030413).
Received 28 January 2000, revised manuscript received 1 May 2000

analysis to the problem.

In the path-based approach, all or a subset of the paths are taken into account in the formulation of the problem, often with a set of linear constraints. It is expected that the problem can be handled more mathematically in this approach, since the timing in VLSI is inherently path-oriented.

Jackson and Kuh[6] proposed an approach based on linear programming. Gao *et al.*[7] proposed a new method of iterative modification of net bounds for taking advantage of the computational efficiency of the net-based approach. Srinivasan *et al.*[8] proposed an approach based on Lagrangian Relaxation. It is observed that only a small subset of timing requirements are active as constraints at one time, thus the problem caused by a large number of paths can be effectively avoided. Timing requirements by a set of linear inequalities are concluded to be on the non-differentiable Lagrangian.

However, there is a difference of opinion about how to calculate the timing delay of a path between the academician and fabricators. The academicians often adopt Elmore delay model or Sakruai delay model to analyze and compute the timing delay of each path, while the fabricators prefer to trust their own delay tables which are obtained by experiments and statistics.

In this paper, a new timing-driven placement algorithm based on table-lookup delay model is proposed in order to eliminate this divergence. As far as we know, there has been no such kind of timing-driven placement algorithm yet. We first simplify and deduce a new timing delay model based on the delay tables of some fabricators; then we write a non linear program to solve the placement problem using the new timing delay model. This algorithm incorporates the well-known quadratic placement with bottom-up clustering and the slicing partitioning strategy, which has been proved to be efficient, effective and fast-to-run.

Quadratic wirelength is choosen as our objective function because it can result in fewer very-long nets than linear objective function, i. e., the quadratic function can place components more sparsely and result in the better timing property and fewer components overlapping each other.

2 Table-Lookup Timing Model

2.1 Delay Table

The nonlinear delay model stores vendor-specific delay information in the technology library in the form of Table-Lookup and supports a high correlation between nonlinear vendor delay models and the timing analyzer calculations. Delay analysis involves calculating the total delay, which comprises cell delay and connect delay:

$$D_{\text{total}} = D_{\text{cell}} + D_{\text{c}} \tag{1}$$

where D_{cell} is the delay caused by the gate itself, typically measured from the 50 percent input pin voltage to the 50 percent output pin voltage, D_{c} is the connect delay. The delay D_{c}

of net k is:

$$D_{ck} = R_k\left[\sum_k C_{pin} + (1/2)C_k\right] \tag{2}$$

where the coefficient 1/2 is due to the distribution characters of R_k. The connect delay of D_c can be computed from (2) easily. Conceptually, D_{cell} can be determined from a delay table, which depends on two parameters: input transition time and output capacitance. The former is the time that is required for the input pin of node to change state. Naturally another parameter, output transition time $\Delta T'_k$ is needed to denote the time for the output pin of node k to change state, which is a portion of the input transition time of the node $(k+1)$. The total input transition time of node $(k+1)$ is: $\Delta T_k = \Delta T'_k + D_{ck}$. Table 1 is a typical delay table.

Table 1 A Typical Delay Table

T \ C	0.022	0.052	0.12	0.278	0.646	1.5
0.112	0.1325	0.1731	0.2624	0.4841	0.9806	2.1814
0.14	0.1325	0.1731	0.2625	0.4836	0.9801	2.1816
0.255	0.1325	0.1757	0.2662	0.4815	0.9827	2.1811
0.543	0.1332	0.174	0.2672	0.4822	0.9825	2.1813
1.301	0.1369	0.1837	0.273	0.4814	0.9756	2.1741

From the delay table, we can see that D_{cell} is roughly a nonlinear increasing function of the parameters input transition time and output capacitance. In geometry, the track of the function is a curved surface, as shown in Fig. 1.

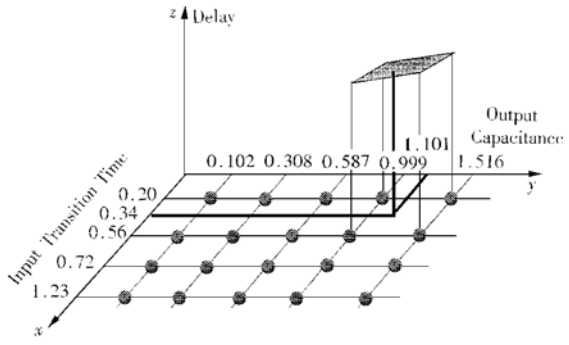


FIG. 1 Curved Surface

2.2 Curved Surface Fitting

Let D_k denote the delay of a certain cell k, and ΔT_k the time for the output pin K to change state; then from section A, we may obtain such supposition:

$$D_k = f(\Delta T_{k-1}, C_{k-\text{total}}) + b = K_{k1}\Delta T_{k-1} + K_{k2}C_{k\text{total}} + b \tag{3}$$

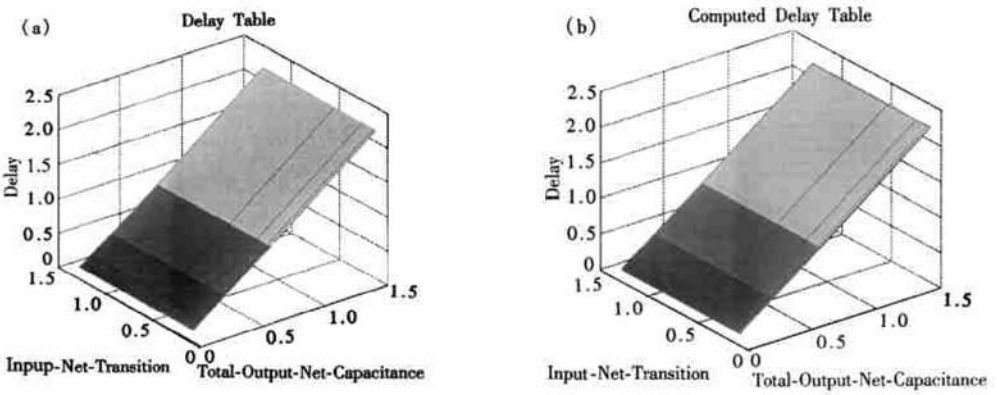
$$\Delta T'_k = f(\Delta T_{k-1}, C_{k-\text{total}}) + lb = L_{k1}\Delta T_{k-1} + L_{k2}C_{k\text{total}} + lb \tag{4}$$

Obviously, we can determine the coefficients according to the delay-tables, which are provided by the fabricators easily. The least-squares fit method is adopted in practice. Take Table 1 for example: for Table 1, the corresponding result of fitting is:

$$K_{k1} = 1.383235,\ K_{k2} = 0.001411,\ b = 0.098862$$

When we use formula (3) instead of Table 1, the minimum deviation is 1.3580% while the maximum is 6.0284%. Figure 2 is the comparison between the fitting plane and the corre-

sponding primary curved surface. According to the computing results, it can be seen that our formula fits the delay table perfectly. We may also determine the coefficients of formula (4) from another delay table in the same format as Table 1.



Primary Curved Surface Fitting Plane

FIG. 2 Comparison of Fitting Plane and Primary Curved Surface

2.3 Computing Path Delay

Formula (3) represents the delay of a single cell. As for the delay of a whole path from node S to node T, such as a path shown in Fig. 3, there are n parts and $n+1$ nodes in it.

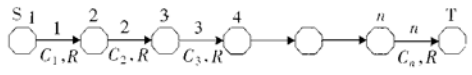


FIG. 3 Computing Path Delay

D_k denotes the timing delay from the input pin of node k to its output pin, ΔT_{k-1} denotes the input transition time of node K, and $C_{k\text{total}}$ the output capacitance of node K. For the sake of simplication, we give the initial boundary conditions as $D_0 = 0$ and $\Delta T_0 = 0$. For node 1, according to (3), we obtain:

$$\begin{aligned} D_1 &= K_{1t}\Delta T_0 + K_{1c}C_{1\text{total}} + b_1 \\ &= K_{1c}C_{1\text{total}} + b_1 \\ &= K_{1c}\left(\sum_1 C_{\text{pin}} + C_1\right) + b_1 \\ &= m_{c11}C_1 + n_1 \\ m_{c11} &= K_{1c},\ n_1 = K_{1c}\sum_1 C_{\text{pin}} + b_1 \end{aligned} \tag{5}$$

where $\sum_1 C_{\text{pin}}$ stands for the sum of capacitance of all sank pins of net 1 and C_1 is itswire capacitance. In the similar way, we have obtained:

$$\begin{aligned} \Delta T_1' &= L_{1t}\Delta T_0 + L_{1c}C_{1-\text{total}} + lb_1 \\ &= L_{1c}C_{1-\text{total}} + lb_1 \end{aligned}$$

$$= L_{1c}\left(\sum_{1} C_{pin} + C_1\right) + lb_1$$

$$= p_{c11}C_1 + q_1$$

$$p_{c11} = L_{1c},\ q_1 = L_{1c}\sum_{1} C_{pin} + lb_1$$

For node 2, the input transition time includes the delay of net 1, namely, D_{1c}. So we have got:

$$\Delta T_1 = \Delta T_1' + D_{c1} = P_{c11}C_1 + q_1 + R_1\left(\sum_{1} C_{pin} + (1/2)C_1\right)$$

$$= P_{c11}C_1 + P_{R11}R_1 + P_{RC11}R_1C_1 + q_1$$

$$P_{c11} = L_{1c},\ P_{R11} = \sum_{1} C_{pin},\ P_{RC11} = 1/2,\ q_1 = L_{1c}\sum_{1} C_{pin} + lb_1$$

analogically, for node n, we can obtain:

$$\begin{aligned}
D_n &= m_{Cn1}C_1 + m_{Cn2}C_2 + \cdots + m_{Cn,n-1}C_{n-1} + m_{Cn,n}C_n + m_{Rn1}R_1 + m_{Rn2}R_2 + \cdots \\
&\quad + m_{Rn,n-1}R_{n-1} + m_{RCn_1}R_1C_1 + m_{RCn2}R_2C_2 + \cdots + m_{RCn,n-1}R_{n-1}C_{n-1} + n_n \\
m_{Cn1} &= K_{nt}P_{Cn-1,1},\ m_{Cn2} = K_{nt}P_{Cn-1,2},\ \cdots,\ m_{Cn,n-1} = K_{nt}P_{Cn-1,n-1},\ m_{cnn} = K_{nc} \\
m_{Rn1} &= K_{nt}P_{Rn-1,1},\ m_{Rn2} = K_{nt}P_{Rn-1,2},\ \cdots,\ m_{Rn,n-1} = K_{nt}P_{Rn-1,n-1} \\
m_{RCn1} &= K_{nt}P_{RCn-1,1},\ m_{RCn2} = K_{nt}P_{RCn-1,2},\ \cdots,\ m_{RCn,n-1} = K_{nt}P_{RCn-1,n-1} \\
n_n &= K_{nt}q_{n-1} + K_{nc}\sum_{n} C_{pin} + b_n \\
\Delta T_n &= P_{Cn1}C_1 + P_{Cn2}C_2 + \cdots + P_{Cn,n-1}C_{n-1} + P_{Cn,n}C_n \\
&\quad + P_{Rn1}R_1 + P_{Rn2}R_2 + \cdots + P_{Rn,n-1}R_{n-1} + P_{Rnn}R_n \\
&\quad + P_{RCn1}R_1C_1 + P_{RCn2}R_2C_2 + \cdots + P_{RCn,n-1}R_{n-1}C_{n-1} + P_{RCn,n}R_nC_n + q_n \qquad (6) \\
P_{Cn1} &= L_{nt}P_{Cn-1,1},\ P_{Cn2} = L_{nt}P_{Cn-1,2},\ \cdots,\ P_{Cn,n-1} = L_{nt}P_{Cn-1,n-1},\ P_{cnn} = L_{nc} \\
P_{Rn1} &= L_{nt}P_{Rn-1,1},\ P_{Rn2} = L_{nt}P_{Rn-1,2},\ \cdots,\ P_{Rn,n-1} = L_{nt}P_{Rn-1,n-1},\ P_{Rnn} = \sum_{n} C_{pin} \\
P_{RCn1} &= L_{nt}P_{RCn-1,1},\ P_{RCn2} = L_{nt}P_{RCn-1,2},\ \cdots,\ P_{RCn,n-1} = L_{nt}P_{RCn-1,n-1},\ P_{RCnn} = 1/2 \\
q_n &= L_{nt}q_{n-1} + L_{nc}\sum_{n} C_{pin} + lb_n
\end{aligned}$$

It is easy for us to deduce D_1, D_2, $\cdots$, D_{n-1}, D_n and ΔT_1, ΔT_2, $\cdots$, ΔT_n from formula (6). Therefore, for the total delay of this path, we have:

$$\begin{aligned}
D_{total} &= D_1 + D_2 + D_3 + \cdots + D_n + D_{c1} + D_{c2} + \cdots + D_{cn} \\
&= a_1C_1 + a_2C_2 + \cdots + a_nC_n + b_1R_1 + b_2R_2 + \cdots + b_nR_n \qquad (7) \\
&\quad + c_1R_1C_1 + c_2R_2C_2 + \cdots c_nR_nC_n + A + D_{c1} + D_{c2} + \cdots + D_{cn}
\end{aligned}$$

where

$$\begin{aligned}
a_1 &= m_{C11} + m_{C21} + m_{C31} + \cdots + m_{Cn1} \\
a_2 &= m_{C22} + m_{C32} + \cdots + m_{Cn2} \\
&\cdots\cdots \\
a_n &= m_{Cnn} \\
b_1 &= m_{R21} + m_{R31} + \cdots + m_{Rn1} \\
b_2 &= m_{R32} + m_{R42} + \cdots + m_{Rn2}
\end{aligned}$$

$$\cdots\cdots$$

$$b_{n-1} = m_{Rn,n-1}$$

$$c_1 = m_{RC21} + m_{RC31} + \cdots + m_{RCn1}$$

$$c_2 = m_{RC32} + m_{RC42} + \cdots + m_{RCn2}$$

$$\cdots\cdots$$

$$c_{n-1} = m_{RCn,n-1}$$

$$A = n_1 + n_2 + n_3 + \cdots\cdots + n_n$$

For a critical path, there is such a constraint as:

$$D_{\text{total}} = D_1 + D_2 + D_3 + \cdots + D_n + D_{c1} + D_{c2} + \cdots + D_{cn} \leqslant T_{\text{limit}} \quad (8)$$

Therefore, we regard all the timing constraints of the critical paths as limiting conditions of a mathematical programming problem, and then solve it problem.

3 Main Algorithm Description

3.1 Timing-Driven Clustering - TIMIX

In the beginning, a clustering algorithm was performed on the primary circuit to obtain a condensed circuit, in which each cluster composed of a set of cells was considered as a single component. After that, a quadratic placement problem has been solved on the clustered circuit instead of the original one. Our clustering algorithm covers following such steps:

First of all, we assign the weight of each net according to the importance of it(i.e. should a net be near the critical paths, it could be assigned a heavier weight).

Secondly, for each movable cell, its weight assigned can be equal to the sum of the weight of all fanout nets driven by it. As shown in Fig. 4, the fanout nets of cell C are net1 and net2, so the weight of cell C should be the sum of the weight of net1 and net2.

Next, from the largest weight to the smallest one, the cells can be sorted out. And then the first unclustered cell is chosen as the seed of a new cluster.

According to the total number of cells, we can determine the average number of cells in a cluster, so that the total number of clusters will remain about 5000. In this way, the size of each cluster is so small that it can be looked as a "point". On the other hand, it will not take very long time to solve such kind of problem. ClusterSize is the average number of cells in a cluster.

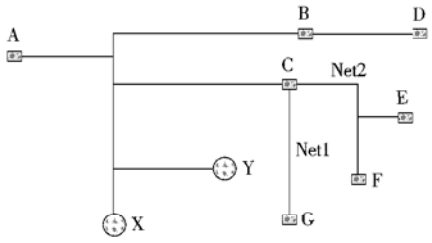


FIG. 4 Sketch Map of Timing-Driven Clustering

As shown in Fig. 4, the current seed A is found down to its fanout elements and fan, and repeatedly adds the fanin elements into cluster A until A reaches ClusterSize. The pseudocode of the timing-driven clustering algorithm TIMIX is shown in Fig. 5.

According the primary netlist based on cells, the new netlist corresponding can be built up

based on clusters. Generally speaking, the scale of the new netlist is much smaller than that of the original one.

```
/* New Timing-Driven Clustering Algorithm: TIMAX* /
Begin:
1. Form the weight of each net;
2. Form the weight of each movable cell;
3. Sort the movable cell set CellList, so that the sequence of CellList will be from the biggest weight to the smallest
   weight;
4. Determine ClusterSize which is the typical number of cells inside each cluster,
5. For each free cell Ce∈CellList,
  {CurrentCells= {Ce}; nextClusters= NULL; CandidatedCells= NULL;
    While (number of CurrentCells< ClusterSize)
    {find all the fanin cells of CurrentCells, for each fanin cell Cin;
      {If (Cin has not been clustered yet)
        Add Cin into CandidateCells;
      Else if (Cin has already been clustered)
        Add the Cluster in which Cin resides into nextClusters;
      }
      if (number of CurrentCells+ number of CandidateCells> = ClusterSize)
        {Create a new cluster with the cells inside CurrentCells and CandidateCells;
          Break;
        }
        else
          {if(There is no change with the sets CurrentCells, CandidateCells and NextCluster)
            {Create a new cluster with the cells inside CurrentCells and CandidateCells;
              Break;
            }
            else
              {Add all the cells inside CandidateCells into CurrentCells;
                CandidateCells= NULL;
              }
            }
          }
        }
6. Construct new netList based on clusters.
```

FIG. 5 Timing-Driven Clustering Algorithm TIMIX

3.2 Problem Formulation

In global optimization steps, a mathematical programming problem is derived and solved. The solution is a global placement of clusters. The objective function is based on nets' quadratic wire length model. For net n. its length is computed according to the following equation:

$$L_n = \sum_{i,j \in n} [(x_i - x_j)^2 + (y_i - y_j)^2]$$

Since the number of clusters in a net is different, we assign a weight w_n to net n. Therefore, the objective is to minimize the sum of the weighted quadratic wirelength of all nets:

$$\Phi = \sum_{n \in N} L_n w_n$$

At the lth partition level, the placement plane can be divided into 2^l regions, each containing a subset of clusters. Let $\mathcal{R}$ denote the index set of regions. If r stand for a region, τ_r is the set of clusters in r, and (μ_r, ν_r) is the coordinate of the center of region r. Since a cluster can reside in one region only, we use $\mathcal{R}_i$ to denote the region where cluster i resides in. Thus, for region r, there are two constraints on the global placement:

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

Combining the objective function with constraints, including the distribution constraints $g_i(x) \leqslant 0$ and timing constraints $d_j(x) \leqslant 0$, we get a constrained quadratic programming problem (LPQ):

$$\min \Phi(x)$$

$$\text{suiected to:}$$

$$g_i(x) \leqslant 0, \ i = 0, 1, 2, \cdots, r;$$

$$d_j(x) \leqslant 0, \ j = 0, 1, 2, \cdots, m$$

3.3 Solution Method

Lagrange relaxation method is used to solve the LQP problem. According to Reference[9] and [10], we can get the global optima:

$$x = Q^{-1} b_x, \ y = Q^{-1} b_y$$

where the vectors x and y are the coordinates of the movable clusters to be placed. We alter the format of above formula and use an iterative solution method——Conjugate-Gradient(CG) method to solve this problem. To improve the condition number of matrix Q, Preconditioned Conjugate-Gradient (PCG) method is often applied in practice, as can result in an efficient solution.

3.4 Post-Processing

During all the previous steps, we compute the positions of elements in the form of clusters instead of cells. Therefore, it's time for us to recover the clusters into cells. Since the coordinates of the cells within a cluster are exactly the same, there is a large overlap of cells, which indicates the cells are assigned to slots arbitrarily. To eliminate or remove the overlap, we propose a simple heuristic to determine the coordinates of cell C_i. Supposing the positions of all the other cells which connect to C_i are fixed, we can get an optimal position of C_i. For C_i. the object function is :

$$L_p = \sum_{n \in N_i, i \neq j} w_n [(x_i - x_j)^2 + (y_i - y_j)^2]$$

Let:

$$\frac{\partial L_p}{\partial x_i} = 0, \quad \frac{\partial L_p}{\partial y_i} = 0$$

Then we have:

$$x_i = \left[\sum_{n \in N_i, i \neq j} w_n x_j \right] \Big/ \sum_{n \in N_i, i \neq j} w_n (n - 1)$$

$$y_i = \left[\sum_{n \in N_i, i \neq j} w_n y_j \right] \Big/ \sum_{n \in N_i, i \neq j} w_n (n - 1)$$

In this step, we apply this method to every cell recursively. The experimental results show that it is very effective to eliminate and remove the overlap of cells.

4 Experimental Results

We have implemented our new timing-driven placement algorithm on ULTRA-SPARC workstations in C language. To investigate the efficiency of our approach, we tested it with two real sequential circuits from industry: TABLE (contains 4596 cells) and DESIGN (contains 44283 cells). In this approach, under the condition of no timing constraints, results can be obtained either by conventional placement (CV) or by timing-driven placement (TD). The experimental results are summarized in Table 2 and 3. k is a penalty-adapter factor. It is shown that with our approach, the maximum path delay can be cut down greatly, while the number of critical paths is reduced significantly. Although the total wirelength increases slightly, it is less important than the timing delay, especially in the case of deep sub-micron crafts.

Table 2 Experimental Results of Circuit Table

k	Algo.	Wire/μm	R-time/s	Max-Delay/ns	Num. of CPs
0.4	CV	3690598	120.33	0.296739	10
	TD	3707684	186.71	0.156441	1
0.5	CV	4207182	116.99	0.294378	76
	TD	4150565	185.48	0.198632	4
0.6	CV	4541789	117.36	0.218765	47
	TD	4669711	189.40	0.166008	9

Table 3 Experimental Results of Circuit Design

k	Algo.	Wire/μm	R-time/s	Max-Delay/ns	Num. of CPs
1.0	CV	8270277	576.23	0.30495619	81
	TD	9989253	686.79	0.14796367	49
2.0	CV	9582777	582.33	0.48153858	91
	TD	11701530	736.83	0.18628043	54
3.0	CV	10248219	583.96	0.32790500	78
	TD	12697523	752.65	0.18073445	55

5 Conclusion

In this paper, new light has been thrown on the problem of timing-driven placement in the delay tables. We first simplify and deduce a new timing delay model based on the delay tables of fabricators; then form a non-linear programming to solve the placement problem by making use of the new timing delay model. Our approach incorporates the well-known quadratic placement with bottom-up clustering and slicing partitioning strategy. The experimental results show that our method is very effective and efficient.

References

[1] Y. Ogawa *et al.*, "Efficient Placement Algorithms Optimizing Delay for High-Speed ECL Masterslice LST's", Proc. of 23rd Design Automation Conf., June, 1986, 404—410.

[2] S. Sutanthavibul and E. Shargowitz, "An Adaptive Timing-Driven Layout for High Speed VLSI", Proc. of 27th Design Automation Conf., June, 1990, 90—95.

[3] N. Weste and K. Eshraghian, Principles of CMOS VLSI Design: A Systems Perspective, Reading MA: Addison-Wesley, 1985.

[4] S. Teig, R. L. Smith and J. Seaton, "Timing-Driven Layout of Cell-Based Ics", VLSI Systems Design, May 1986, 63—73.

[5] A. E. Dunlop, V. D. Agrawal *et al.*, "Chip Layout Optimization Using Critical Path Weighting", Proc. of 21st Design Automation Conf., 1984, 133—136.

[6] M. A. B. Jackson and E. S. Kuh, "Performance-Driven Placement of Cell Based IC's", Proc. of 26th Design Automation Conf., June 1989, 370—375.

[7] T. Gao, P. M. Vaidya and C. L. Liu, "A Performance Driven Macro-Cell Placement Algorithm", Proc. of 29th Design Automation Conf., June 1992, 147—152.

[8] A. Srinivasan, K. Chaudhary and E. S. Kuh, "RITUAL: A Performance Driven Placement Algorithm for Small Cell ICs", Proc. of Int. Conf. on Computer-Aided Design, Nov. 1991, 48—51.

[9] Tianming Kong, Xianlong Hong and Changge Qiao, "VEAP: A Global Optimization Based Placement Algorithm for Standard Cell Design", ASP-DAC'97, Japan, 1997, 1.

[10] Hong Yu, Xianlong Hong, Changge Qiao and Yici Cai, "CASH: A Novel Quadratic Qlacement Algorithm for Very Large Standard Cell Layout Design Based on Clustering", Proceedings of the 5th International Conference on Solid-State and Integrated Circuit Technology, 1998, 496—501.