A New Performance Driven Placement Method with the Elmore Delay Model for Row Based VLSIs

Tetsushi Koide* Mitsuhiro Ono* Shin’ichi Wakabayashi* Yutaka Nishimaru† Noriyoshi Yoshida†

Faculty of Engineering, Hiroshima Univ.* Tel: +81-82-830-1792
4-1, Kagamiyama 1 chome, Faculty of Information Sciences, Hiroshima City Univ.†
Higashi-Hiroshima, 739 JAPAN Tel: +81-82-830-1760, Fax: +81-82-830-1792
e-mail: koide@ecs.hiroshima-u.ac.jp

Faculty of Information Sciences, Hiroshima City Univ.†
Asa-Minami-Ku, Hiroshima 731-31, JAPAN

Introduction

Performance is one of the most important criterion to evaluate the quality of VLSI chips. VLSI layout design which handles performance explicitly is generally called performance driven layout. Due to the advance of semi-conductor process technologies, interconnection delay cannot be ignored as well as switching delay of gates in the physical design[1]. Therefore, dealing with performance explicitly means dealing with the interconnection delay.

There have been many studies about performance driven layout, especially performance driven placement, and they can be classified into the following four groups, (1) the net weighting approach[2, 15], (2) the net delay constraint approach[7, 10, 18], (3) the path weighting approach[9, 19], and (4) the path delay constraint approach[3, 8, 11, 17]. However, many of them have a difficulty of trade-off between the quality of the layout and the computation time. Especially for interconnection delay, the estimation of the interconnection delay is inaccurate because of some assumptions of the model.

In this paper, we propose a new performance driven placement algorithm of the path constraint based approach. As the interconnection delay model of the proposed method, the Elmore delay model is used explicitly so that we can estimate the accurate interconnection delay and can also apply the proposed method to wider technologies than conventional methods. The proposed algorithm consists of four phases. The algorithm first gets an initial placement, then improves it iteratively. In the improvement step, the algorithm selects a subcircuit and finds a new placement by nonlinear programming. Because the subcircuit includes at least one whole path which violates its own delay requirement, we can treat the timing constraint more flexibly than the net based approach. Moreover, to limit the size of subcircuit enables the proposed method to apply to large circuits. Finally, the placement is formed to a row based layout style by the performance driven row assignment phase. From the experimental results comparing with RITUAL[17], the proposed method improved the total wire length by a 19.4% on average and 36.0% in maximum compared with RITUAL. As a result, the proposed method is much better than RITUAL in point of the maximal violation ratio, the total wire length, and the cut size, and is more effective in point of the interconnection delay model and its extendability.

I Preliminaries

A. Layout and Delay Models

In this paper, the row based design such as the poly-cell type standard cell or the gate array models is assumed. The Sea-of-Gates and the mixed macro cell models can be adopted too with minor modification. The metal routing consists of two layers, the first layer is used for horizontal direction routing, and the second for vertical direction routing.

An equivalent circuit of an interconnection is originally modeled as a distributed RC circuit, and the Elmore’s delay equation[5] is often used to represent the interconnection delay. When a multi-terminal net is implemented by a Steiner tree, Kuh gives an upper bound of the Elmore delay from the source to the load pin i of the net by the following equation[14].

\[ d_i(W, L_{0i}) = (cW + \sum_j C_{ij})(R_0 + r L_{0i}), \] (1)

where \( W \) is the total wire length of the Steiner tree, \( L_{0i} \) is the path length from the source to the load \( i \), \( c \) and \( r \) are the wire capacitance and resistance per unit length, respectively, \( R_0 \) is the equivalent output resistance of the source, and \( \sum_j C_{ij} \) is the sum of the load capacitances. We employ the equation (1) as an
interconnection delay model. However, the wire capacitances are different between the first and second metal layers (M1 and M2), so we compute the delay as the sum of delay of M1 and M2. Furthermore, it is not practical to construct Steiner trees during placement from the point of computation time, so we estimate the wire length of a net by half perimeter of a bounding box of the pins of the net. The delay from the source pin to the load pin of the net is thus defined as,

\[ d_i(l_1, l_2) = (c_1l_1 + c_2l_2 + \sum_j C_{ij}R_0 + r_1l_1 + r_2l_2) \]

where \( l_1 \) and \( l_2 \) are the width and height of the bounding box of the net, \( c_1 \) and \( c_2 \) are the capacitance of M1 and M2 per unit length, and \( r_1 \) and \( r_2 \) are the resistance of M1 and M2 per unit length, respectively.

### B. Timing Constraint

In this paper, we consider the long path problem. As there are many paths from a primary input (PI) or an output of flip-flops (FFs) to a primary output (PO) or inputs of FFs, they can be specified by pairs of pins, source ones and sink ones. Thus we specify a timing constraint as \( t_\tau = (s_\tau, c_\tau, D_{rc_\tau}) \), where \( s_\tau \) is a source pin, \( c_\tau \) is a sink pin, and \( D_{rc_\tau} \) is the maximum permissible delay from the source to the sink. For example, if a circuit and its timing constraint are given as shown in Fig. 1(a), the delay of any path from \( s_\tau \) to \( c_\tau \) in this case three paths, must be less than \( D_{rc_\tau} \) (Fig. 1(c)). We have to get the layout satisfying all elements of the set of timing constraints \( T \).

### C. Problem Formulation

We define some terminologies and symbols. Let \( \mathcal{L} = (\mathcal{M}, \mathcal{N}) \) be a logic circuit, where \( \mathcal{M} = \{m_1, m_2, \ldots, m_M\} \) is a set of cells and \( \mathcal{N} = \{n_1, n_2, \ldots, n_N\} \) is a set of nets. A set \( \mathcal{N}_i \) is a set of nets connecting to a cell \( m_i \), and a set \( \mathcal{M}_j \) is a set of cells connecting to a net \( n_j \). For every timing constraint \( t_\tau \in T \), we define a critical path denoted by \( p_\tau = (\mathcal{M}_z, \mathcal{N}_z) \) as any path whose source is \( s_\tau \) and sink is \( c_\tau \), where \( \mathcal{M}_z \) is a set of nets which are critical on the critical path and \( \mathcal{N}_z \) is a set of nets which have connection to some cell on the critical path. Let \( \mathcal{P} \) be a set of critical paths and let \( \mathcal{P}_\tau \subset \mathcal{P} \) is a set of critical paths specified by a timing constraint \( t_\tau \in T \). Let \( D_{rc_\tau} \) be the required propagation delay of \( p_\tau \in \mathcal{P} \), and let \( D_{ac_\tau} \) be the actual propagation delay of \( p_\tau \in \mathcal{P} \).

For every timing constraint \( t_\tau \in T \), let \( \mathcal{L}_\tau = (\mathcal{M}_\tau, \mathcal{N}_\tau) \) be a constrained circuit, in which a set of cells and nets are defined as \( \mathcal{M}_\tau = \bigcup_{p_\tau \in \mathcal{P}_\tau} \mathcal{M}_\tau \), and \( \mathcal{N}_\tau = \bigcup_{p_\tau \in \mathcal{P}_\tau} \mathcal{N}_\tau \). If \( \mathcal{M}_\tau \) are regarded as vertices and \( \mathcal{N}_\tau \) as edges whose directions are given by corresponding signal flows, a constrained circuit \( c_\tau \) is represented as a directed acyclic graph (Fig. 1(b)), in which the source is \( s_\tau \) and the destination is \( t_\tau \). Let \( \mathcal{C} \) be a set of constrained circuits, and let \( D_{act_\tau} = \max_{p_\tau \in \mathcal{P}_\tau} D_{ac_\tau} \) be the actual propagation delay time from \( s_\tau \) to \( c_\tau \). Figure 1 shows an example of our definition. (a) is a timing constraint \( t_\tau = (s_\tau, c_\tau, D_{rc_\tau}) \), and (b) is the constrained circuit corresponding to \( t_\tau \). There are three critical paths as shown in (c).

Now, we formulate the performance driven placement problem.

**The Performance Driven Placement Problem**

**Inputs:** a logic circuit \( \mathcal{L} = (\mathcal{M}, \mathcal{N}) \), timing constraints \( T \), and physical parameters of equation (1) of the Elmore delay model

**Output:** positions of \( \mathcal{M} \) which minimize the objective function

**Objective Function:** the total wire length of nets

**Constraints:** satisfy the layout model and the timing constraints \( T \)

### III A NEW PERFORMANCE DRIVEN PLACEMENT METHOD

**A. Outline of the Proposed Method**

The proposed method consists of three phases. In phase 1, it generates an initial placement by a hierarchical timing driven mincut placement algorithm. Next, the placement obtained in phase 1 is improved using nonlinear programming in phase 2. This phase is iterative improvement, and in each iteration, a subcircuit which contains a critical path violating its constraint is placed. Finally, the cells are assigned in rows considering the timing constraints. We will explain the details of each phase in the following subsections.

**B. Phase 1: Initial Placement Based on Hierarchical Timing Driven Mincut Partitioning**

In our initial placement, three points should be considered, to minimize the total wire length, to distribute cells uniformly in the placement region and to reduce the violations of timing constraints as much as possible. As the violations will be eliminated in phase 2, it is not necessary for the initial placement to satisfy all of timing constraints.

We employ an extended version of the timing driven mincut placement algorithm we proposed in [19] as this phase, because
it takes three points mentioned above into account, produces a placement comparatively quickly and uses the same interconnection delay model as the proposed method does. This algorithm is based on ordinary hierarchical quadratic partitioning. The quadratic partitioning is basically realized by applying the well-known bi-partitioning method, called the FM method[6], in three times. Both logic cells and a region in which the cells are placed at the center is divided into four parts(Fig. 2(a)).

We extended the FM method so as to consider timing constraints. The FM method is an iterative improvement method and elements(in this case cells) with the maximal gain are moved to the opposite side of the partition one by one. The gain, we call it cut gain, here, for a cell 

where

is the slack of a critical path

of nets. We consider a vertical(resp. horizontal) partitioning

We call

virtual timing constraints. Let

be defined as follows.

Finally, we define the gain, called terminal gain, as

and let

be a set of cells connecting with cell

and let

be the width(resp. height) of a horizontal partitioning.

Let

region_width

resp. region_height

be the width(resp. height) of the region to be partitioned. Now, we define the cost concerned with terminal positions of nets connecting with the cell

as

where

Fig. 3. cost_termi of a horizontal partitioning.

Fig. 2. Flow of the proposed hierarchical quadratic partitioning.
in Fig. 2(a)–(f). Since the precise cell positions are already assigned after the first partitioning as shown in Fig. 2(b), the terminal and wire gains can be accurately calculated and a good initial placement can be obtained.

C. Phase 2: Iterative Improvement Based on Nonlinear Programming

1. Selection of a Target Subcircuit

An initial placement may violate some timing constraints. The objective of phase 2 is to eliminate all the violations and to minimize the total wire length of the placement. To achieve them, we transform the placement problem to a mathematical programming problem. But mathematical programming tends to require much computation time and memory space. Therefore, we apply mathematical programming to subcircuits, for which the formulated problem can be solved in a practical computation time and with practical size of memory space.

Now, we define the target subcircuit as \( L_{\text{mov}} = (M_{\text{mov}}, N_{\text{mov}}) \), where \( M_{\text{mov}} \) is the set of cells, called movable cells, of the subcircuit, and \( N_{\text{mov}} \) is the set of nets, called movable nets, connecting to at least one movable cell. The cells other than movable cells are called fixed cells and their set is represented by \( M_{\text{fix}} \). The nets other than movable nets are called fixed nets and their set is represented by \( N_{\text{fix}} \).

The target subcircuit is selected by the following algorithm. In the algorithm, \( rand_d(0,1) \) is a real number between 0 and 1 randomly generated for each cell \( m_i \).

[ The Target Subcircuit Selection Algorithm ]

**Step 1:** Find a critical path \( P_c \) with large violation ratio;

**Step 2:** \( M_{\text{fix}} = M - M_{\text{mov}} \); \( N_{\text{fix}} = N - N_{\text{mov}} \);

**Step 3:** Calculate connectivity of cells of \( M_{\text{fix}} \);

**Step 4:** If \( 2 \times \left| (M_{\text{mov}}) + |N_{\text{mov}}| \right| \geq \) (preset value), then output \( M_{\text{mov}} \) and \( N_{\text{mov}} \) and stop;

**Step 5:** Search a cell \( m_i \in M_{\text{fix}} \) in decreasing order of its connectivity until the condition \( rand_d(0,1) \geq \) (preset value) is satisfied;

**Step 6:** \( M_{\text{mov}} = M_{\text{mov}} \cup \{m_i\} \); \( M_{\text{fix}} = M_{\text{fix}} - \{m_i\} \);

**Step 7:** Update connectivity and go to Step 4.

First, we find one of critical paths with a large violation ratio. The violation ratio is the value of actual delay time of a critical path (or a constrained circuit) divided by the required delay time of it, i.e., \( \frac{P_{\text{critical}}}{P_{\text{required}}} \). The candidates for the critical path are selected from constrained circuits which are the largest 10–20 percent in all constrained circuits in point of the violation ratio. Firstly, let the critical path be the initial subcircuit. Next, expand the subcircuit by adding cells one by one. The added cell should have large connectivity, which is the number of connections to the present subcircuit. In order to avoid repeatedly selecting the same cell to be added to the target subcircuit in each iteration of phase 2, we introduce a randomness in the selecting step and determine whether the cell is included.

Step 5). If it is included, all nets connecting to it turn to movable nets. By the way, when the placement problem is translated to a mathematical programming problem, it needs variables twice the number of the movable cells and movable nets. To solve the problem in a practical computation time, we must limit the number of variables in the mathematical programming. Hence, the growing process of the subcircuit continues until the number of variables of the mathematical programming problem reaches to a given constant (Fig. 4).

There are two reasons why we construct the target subcircuit in such a way. Firstly, if the subcircuit around a violated critical path is improved at the same time, the cells on the path must be able to move fairly freely. Second, the cells with many connections with the critical path must be included in the same constrained circuit so that moving the cells connecting with the critical path will be effective to reduce the timing violation.

2. Constraints of Nets

To transform the placement problem to a mathematical programming problem, we need variables to represent the wire length of movable nets as well as positions of movable cells. We define two variables for each movable cell, these are \( x_j \) and \( y_i \) which are the XY coordinates of a cell \( m_j \in M_{\text{mov}} \). We also define two variables for each movable net as shown in Fig. 5(a), where \( w_i \) is the width of the bounding box of a net \( n_i \in N_{\text{mov}} \) and \( h_i \) is the height. Then we can represent the bounding box of a movable net \( n_i \) by these variables and the following inequalities.

\[
\begin{align*}
CN1: \quad x_j &- x_k \leq w_i \\
y_j &- y_k \leq h_i \\
\forall m_j \neq m_i \in M_{\text{fix}} \cap M_{\text{mov}}, \quad y_i &\in N_{\text{mov}}
\end{align*}
\]

They mean that for any pair of movable cells connecting to \( n_i \), they are completely included in the bounding box of \( n_i \). When \( n_i \) connects to fixed cells, the following another inequalities are needed.

\[
\begin{align*}
CN2: \quad x_j &- x_{\text{min}} \leq w_i \\
x_{\text{max}} &- x_j \leq w_i \\
y_j &- y_{\text{min}} \leq h_i \\
y_{\text{max}} &- y_j \leq h_i \\
\forall m_j \in M_{\text{fix}} \cap M_{\text{mov}}, \quad \forall n_i \in N_{\text{mov}}
\end{align*}
\]
If a net has some fixed cells, the bounding box of the fixed cells can be constructed, and for any pair of a movable cell and the bounding box, their bounding box is completely included in the bounding box of the net.

Some conventional methods[11, 12] based on mathematical programming use four variables for each net, those are coordinates of left lower corner and right upper corner of bounding box of the net as shown in Fig. 5(b). The method of Fig. 5(b) (let it be B) needs four times the number of movable nets, while the method of Figure 5(a) (let it be A) needs only twice. So A is superior to B in point of the number of variables. On the other hand, there is not a large difference in the number of inequations between them. Hence, we have employed the method A.

3. Constraints of Path Delay

Our placement problem has timing constraints, so these constraints should also be transformed to the mathematical programming problem. As mentioned in Section 1., a target subcircuit $L_{move}$ grows from a critical path (Fig. 4). However, there are many critical paths other than it which are partially or entirely included in the subcircuit. If we have not thought about them during the improvement, it would happen that while violation of the first selected critical path might be eliminated, other critical paths might cause timing violations. Therefore we have to consider all of them as constraints. Constraints of critical path delays can be written as follows.

$$CP : \sum_{n_i \in N \cap N_{move}} d_i(w_{ij}, h_{ij}) + \sum_{n_i \in N \cap N_{fix}} d_i(X_{max_i} - X_{min_i}, Y_{max_i} - Y_{min_i})$$

where $d_{ij}$ is the switching delay of cell $n_i$. In these inequations, the right side means the permissible delay time of a path and the left side means the actual delay time of it. The first term of the left side is the sum of delay of movable nets on it, the second is the sum of delay of fixed nets, and the third is the sum of switching delay of cells. In this formulation, only the first term of the left side has variables and the inequations are quadratic (See Eq. (2) in Sec. 2.1).

4. NLP Formulation of the Problem

The objective of our problem is to minimize the total wire length. However, in general, a placement produced by mathematical programming with minimizing the total wire length tends to make the distribution of cells imbalance, i.e., some cells may concentrate in a local region. This is because this objective does not concern with the differences of length between the nets. If many cells overlap each other, they must be moved far away in the post processing, resulting that the “goodness” of the placement obtained in phase 2 would be diminished and this is nonsense. But it is difficult to add some constraints or to adopt a special objective function to explicitly make cells uniformly distributed on the chip while keeping the convexity of the problem. So, to distribute cells uniformly on the chip, we rearrange the objective function as minimizing the sum of square of wire length. If such an objective is taken, it tends to make the wire length of each net equal than a linear objective function even if the sum of wire length is the same in both objectives.

From above arguments and Sects. III.C.2. and III.C.3., we formulate the placement problem as a mathematical programming problem.

$$\text{minimize } \sum_{n_i \in N_{move}} \alpha_i (w_{ij}^2 + h_{ij}^2) \quad (9)$$

subject to

$$CN1 \cup CN2 \cup CP$$

where $\alpha_i$ is the constant considering a criticality of a net $n_i$. Because the objective and constraints $CP$ are quadratic, this problem is a nonlinear programming problem (NLP).

5. The Algorithm of Phase 2

The algorithm of phase 2 is shown below.

[Iterative Improvement Based on NLP]

Step 1: Perform timing verification to all constrained circuits; $LoopNumber = 1$;

Step 2: If the maximum violation ratio is less than a pre-determined permissible violation ratio or $LoopNumber > (\text{pre-set value})$, then stop;

Step 3: Select a target subcircuit $L_{move}$ to be improved;

Step 4: Find all critical paths which have the cells of $M_{move}$;

Step 5: Formulate the nonlinear programming problem (9) and solve it;

Step 6: Perform timing verification to the constrained circuits which have cells of $M_{move}$;

Step 7: $LoopNumber = LoopNumber + 1$ and go to Step 2;

This algorithm improves a placement iteratively, and in each iteration, it constructs a target subcircuit, formulates a nonlinear programming problem and solves it. It starts from timing verification and calculates violation ratios for all constrained circuits. In the following loop, the improvement and timing verification is done. This timing verification is executed for all the constrained circuits which have movable cells. This loop is
repeated until the maximum violation ratio is less than a pre-determined permissible violation ratio or the loop count reaches some preset value.

As a nonlinear programming method, we employ the multiple method[13], which is easy to implement, and it takes \( O(k_1V^2 + k_2C) \) computation time, where \( V \) is the number of variables, \( C \) is the number of constraints, and \( k_1 \) and \( k_2 \) are constants (or sometimes variables) concerning loop counts in the method. Instead of the multiple method, any other faster nonlinear programming method can be used to solve the problem (9).

D. Phase 3: Timing Driven Row Assignment

In phase 3, the cells, which are distributed on the chip in phase 2, are assigned to cell rows. Now, let \( \tilde{R} \) be the number of cell rows, and let \( R_1, R_2, \ldots, R_{\tilde{R}} \) be the set of cells of each row. All cell rows have areas, and let \( a(M) \) be the sum of area of the cells in \( M \). The width of the chip is determined by the width of the longest cell row, and if the width of all rows are same, then the width of the chip is minimized. Thus we give the same capacity, denoted \( A \), to all cell rows, and all cell rows must satisfy \( a(R_i) \leq A \), \( i = 1, \ldots, \tilde{R} \). The rows have \( y \) coordinates \( Y_1, Y_2, \ldots, Y_{\tilde{R}} \).

In the proposed row assignment algorithm shown first, the cells between each two consecutive cell rows are grouped first, and next for each group, the cells in it are assigned to slots in the two consecutive cell rows by linear assignment considering the wire length and the timing constraints (row assignment of \( y \)-direction) (Fig. 6(a)). Next, cell groups are constructed based on \( x \)-coordinate of cells from left to right of the chip, and the cells in each group are reassigned to slots of the improved region by linear assignment in a similar way of \( y \)-direction (row assignment of \( x \)-direction) (Fig. 6(b)). The above operations are iteratively performed while the placement is improved. In the following, row assignment of \( y \)-direction is described.

First, we explain how to construct the groups. There are \( \tilde{R} \) groups, \( G_1, G_2, \ldots, G_{\tilde{R}} \). For each \( G_i, i = 1, \ldots, \tilde{R} \), they have capacities, which are \( A_i = A \), for \( i = 2, \ldots, \tilde{R} \) and \( A_1 = 3A/2 \).

All cells are sorted by their \( y \) coordinates, and the first cells, of which the sum of the area is equal to the capacity of the first group, are assigned to the first group. Similarly, the remaining cells are divided into the groups. Next, the cells are assigned to slots in cell rows. We transform this assignment problem to a linear assignment problem. Because of \( a(G_i) \geq A \), if the cells in group \( G_i \), which are not assigned to slots of the cell row \( R_i \) but slots of the cell row \( R_{i+1} \), then the cells assigned to \( R_{i+1} \) are added to the next group \( G_{i+1} \) and reassigned in the next linear assignment problem.

The linear assignment problem that the cells in the group \( G_i \) are assigned to slots of the cell row \( R_i \) is formulated as follows.

\[
\text{minimize} : \quad \sum_{m_j \in G_i} \sum_{s_k \in S_i} c_{j,k} z_{j,k} \\
\text{subject to} : \quad \sum_{s_k \in S_i} z_{j,k} = 1, \quad \forall m_j \in G_i, \forall s_k \in S_i, \quad z_{j,k} \geq 0, \quad \forall m_j \in G_i, \forall s_k \in S_i
\]

where \( S_i \) is a set of slots in which all cells in \( G_i \) are assigned, and \( c_{j,k} \) is a cost with which the cell \( m_j \) is assigned to the slot \( s_k \). The cost \( c_{j,k} \) is defined as

\[
c_{j,k} = \sum_{m_j \in S_j} \Delta_l(m_j, s_k) f(\omega_i), \quad (10)
\]

where \( \Delta_l(m_j, s_k) \) is a difference of the wire length of the net \( n_i \) connected to the cell \( m_j \), when the cell \( m_j \) is assigned from the current slot to the slot \( s_k \), and \( f(\omega_i) \) is a function of the criticality \( \omega_i \) of net \( n_i \) and if \( \omega_i \) exceeds a preset value, for example 0.9, then it increases rapidly. The timing constraints are considered by the criticality of the nets. For the timing constraint \( t_r \in T, D_{req} \) is the required delay and \( D_{act} \) is the actual delay. We reflect the degree of the violation on \( \omega_i \) as

\[
\omega_i = \max_{\omega \in \mathcal{C}} \frac{D_{act} - D_{req}}{D_{req}}, \quad (11)
\]

The timing driven row assignment algorithm is as follows.

[ Timing Driven Row Assignment ]

Step 1 : \( \text{LoopNumber} = 0; \)

/* Row assignment of \( y \)-direction */

Step 2 : Construct groups \( G_1, G_2, \ldots, G_{\tilde{R}} \) based on \( y \) coordinates of cells;

Step 3 : \( i = 1; \)

Step 4 : For all cells \( m_j \in G_i, s_k \in S_i \), compute \( c_{j,k} \) and solve the linear assignment problem;

Step 5 : For all cells \( m_j \in R_i \), update their coordinates.

Step 6 : \( G_{i+1} = G_i \cup R_{i+1}; \)

Step 7 : If \( i < \tilde{R} \), then \( i = i + 1 \) go to Step 4, else terminate;

/* Row assignment of \( x \)-direction */

Step 8 : Construct groups \( G'_1, G'_2, \ldots, G'_{\tilde{R}} \) based on \( x \) coordinates of cells;

Step 9 : \( i = 1; \)

Step 10 : For all cells \( m_j \in G'_i, s_k \in S'_i \), compute \( c_{j,k} \) and solve the linear assignment problem;

Step 11 : For all cells \( m_j \in R'_i \), update their coordinates.
We have implemented the proposed placement method called POPINS and performed some experiments. All experiments are done on a SPARC server1000 (135.5MIPS). Table I shows the tested data. Among them, “fract” ~ “avq.large”, are the MCNC benchmarks, and “s1494” ~ “s35932” and “C1” ~ “C7” are ISCAS benchmarks. For ISCAS benchmarks, logic synthesis and technology mapping were performed by SIS1.2[16]. In this table, “#cons” is the number of timing constraints. As the timing constraints, we gave a clock cycle time for “fract”, “biomed”, “s1494” ~ “s35932”, “C1” ~ “C7”. The clock cycle time of “fract” and “biomed” was determined by that determined from a placement produced by a non-performance driven placement method multiplied by 0.8 ~ 0.9. Those of “s1494” ~ “s35932” and “C1” ~ “C7” were given. “primary1”, “primary2”, “avq.small”, and “avq.large” have 16-bit registers, thus we gave the timing constraints so as to synchronize the arrival time of all flip-flops in the same registers.

We compared POPINS with RITUAL[17]. RITUAL is one of the most powerful performance driven placement algorithms which can satisfy a given clock cycle. The interconnection delay model is similar to ours, except the wire resistance is not assumed. But to compare with our results, we evaluated the result by our model. Moreover, in RITUAL, a cell which has more than one output pins is not permitted, so we could not test “fract” ~ “avq.large”. The results of POPINS and RITUAL are shown in Table II. In Table II, “#vio.” is the number of violated timing constraints, Delay Max., defined as Delay Max. = max_v_i , e_t D_o_a_t , / D_r_e_m , is called the maximal violation ratio and if it is less than or equal to 1.0, the placement satisfies all timing constraints. Delay Ave. is an average violation ratio, i.e., Delay Ave. = 1 / T \sum_{v_i , e_t} D_o_a_t / D_r_e_m, and length is the total wire length estimated by the Manhattan distance (\lambda). We uniformly generated 15 cut lines for each direction and counted the number of nets crossed the corresponding cut line (Fig. 7). “#h-cuts Max.” and “Ave.” are the maximum and average cut sizes of the horizontal(\textit{x}) direction, respectively. Similarly, “#v-cuts Max.” and “Ave.” are the maximum and average cut sizes of the vertical(y) direction, respectively. “time” is the running time by SPARCserver1000 (seconds).

From the results of Table II, POPINS improved the total wire length by a 19.4% on average and a 36.0% in maximum compared with RITUAL. Form “#h-cuts” and “#v-cuts”, POPINS also produced the smaller and more uniform cut size placements than RITUAL. As a results, the proposed method can produce better placements in points of the maximal/average violation ratio, the total wire length, and the cut size. For the CPU time, if the nonlinear programming method used in the phase 2 can be improved or replaced with more superior one, for example, some commercial packages, the computation time can be shorter. However, for large size data “s35932” which has 11838 cells and 12228 nets, POPINS can obtain a 16.1% better result within the shorter computation time than RITUAL. From above experiments, the results of POPINS are much better than those of RITUAL.

Furthermore, our method is superior to RITUAL in the following points. First, we assume the more exact interconnection delay model, so our method can be used in wider technologies, but RITUAL has a restriction in technologies which it can be applied to because of its timing model. Second, the timing constraint we assume is the set of pin to pin constraints while that of RITUAL is a clock cycle time, so our method can be used for more complex timing constraints such as a circuit with multi phase clock. Moreover, to improve the performance of an existing placement which is produced by a placement algorithm without considering timing constraint, the phase 2 of our method is very effective. Finally, it is easy to introduce parallel processing into phases 1, 2, and 3 so that the algorithm can easily handle very large scale cell-based ICs. From the above consideration, the proposed method is more effective than other existing performance driven placement methods in point of the interconnection delay model and its extendability.

V Conclusions

In this paper, we proposed a new performance driven layout method for designing high performance VLSI chips. The
proposed method can satisfy the performance requirements of the circuit by satisfying the timing constraints. And in the proposed method, we adopted the Elmore delay model as the interconnection delay model, which is one of the most accurate models used in existing performance driven placement methods. Experimental results showed the effectiveness of the proposed method.

REFERENCES


TABLE II

<table>
<thead>
<tr>
<th>Data</th>
<th>Method</th>
<th>#vio.</th>
<th>Delay</th>
<th>#h-cuts</th>
<th>#v-cuts</th>
<th>time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>tract</td>
<td>POPINS</td>
<td>0</td>
<td>0.85</td>
<td>0.32</td>
<td>45225</td>
<td>34</td>
</tr>
<tr>
<td>primary1</td>
<td>POPINS</td>
<td>0</td>
<td>0.69</td>
<td>0.21</td>
<td>412824</td>
<td>164</td>
</tr>
<tr>
<td>primary2</td>
<td>POPINS</td>
<td>0</td>
<td>0.60</td>
<td>0.36</td>
<td>412832</td>
<td>448</td>
</tr>
<tr>
<td>biomed</td>
<td>POPINS</td>
<td>0</td>
<td>0.54</td>
<td>0.34</td>
<td>412832</td>
<td>510</td>
</tr>
<tr>
<td>avq.small</td>
<td>POPINS</td>
<td>0</td>
<td>0.88</td>
<td>0.18</td>
<td>23848188</td>
<td>1463</td>
</tr>
<tr>
<td>avq.large</td>
<td>POPINS</td>
<td>0</td>
<td>0.84</td>
<td>0.20</td>
<td>28323022</td>
<td>1643</td>
</tr>
<tr>
<td>s1494</td>
<td>POPINS</td>
<td>0</td>
<td>0.67</td>
<td>0.39</td>
<td>257863</td>
<td>90</td>
</tr>
<tr>
<td>s9234</td>
<td>POPINS</td>
<td>0</td>
<td>0.65</td>
<td>0.40</td>
<td>373477</td>
<td>150</td>
</tr>
<tr>
<td>s38417</td>
<td>POPINS</td>
<td>0</td>
<td>0.70</td>
<td>0.31</td>
<td>309136</td>
<td>146</td>
</tr>
<tr>
<td>s35932</td>
<td>POPINS</td>
<td>0</td>
<td>0.70</td>
<td>0.23</td>
<td>4434095</td>
<td>519</td>
</tr>
<tr>
<td>C1</td>
<td>RITUAL</td>
<td>0</td>
<td>0.70</td>
<td>0.32</td>
<td>243765</td>
<td>92</td>
</tr>
<tr>
<td>C2</td>
<td>RITUAL</td>
<td>0</td>
<td>0.59</td>
<td>0.53</td>
<td>303696</td>
<td>116</td>
</tr>
<tr>
<td>C3</td>
<td>RITUAL</td>
<td>0</td>
<td>0.67</td>
<td>0.36</td>
<td>442748</td>
<td>166</td>
</tr>
<tr>
<td>C5</td>
<td>RITUAL</td>
<td>0</td>
<td>0.71</td>
<td>0.39</td>
<td>475808</td>
<td>132</td>
</tr>
<tr>
<td>C6</td>
<td>RITUAL</td>
<td>0</td>
<td>0.63</td>
<td>0.29</td>
<td>1286249</td>
<td>330</td>
</tr>
<tr>
<td>C7</td>
<td>RITUAL</td>
<td>0</td>
<td>0.65</td>
<td>0.29</td>
<td>994319</td>
<td>304</td>
</tr>
</tbody>
</table>

#vio.: the number of timing violations  #h-cuts: the cut size of the horizontal direction  #v-cuts: the cut size of the vertical direction