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.*
4-1, Kagamiyama 1 chome,
Higashi-Hiroshima, 739 JAPAN
Tel: +81-824-24-7676, Fax: +81-824-22-7195
e-mail: koide@ecs.hiroshima-u.ac.jp

Faculty of Information Sciences, Hiroshima City Univ.†
151-5, Ozuka, Numata-Cho,
Asa-Minami-Ku, Hiroshima 731-31, JAPAN
Tel: +81-82-830-1760, Fax: +81-82-830-1792
e-mail yoshida@ce.hiroshima-cu.ac.jp

Abstract— In this paper, we present a new performance driven placement method based on path delay constraint approach for large standard cell layout. The proposed method consists of three phases and uses the Elmore delay model to model interconnection delay precisely in each phase. In the first phase, initial placement is performed by an efficient performance driven mincut partitioning method. Next, an iterative improvement method by nonlinear programming improves the layout. The improvement is formulated as the problem of minimizing the total wire length subject to critical path delays. Finally, row assignment considering timing constraint is performed. From the experimental results comparing with RITUAL[17], 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 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.

II Preliminaries

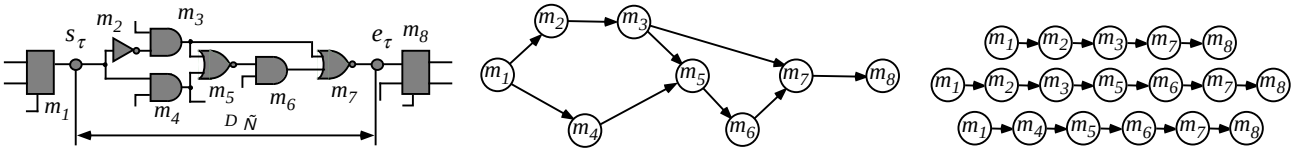
A. Layout and Delay Models

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 pin to the load pin i of the net by the following equation[14].

$$d_i(W, L_{0i}) = (cW + \sum_j C_{lj})(R_0 + rL_{0i}), \qquad (1)$$

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 $\sum_j C_{lj}$ is the sum of the load capacitances. We employ the equation (1) as an



(a) A timing constraint. (b) The constraind circuit. (c) Critical paths.

Fig. 1. A constrained circuit and critical paths.

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 i of the net is thus defined as,

$$d_i(l_1, l_2) = (c_1 l_1 + c_2 l_2 + \sum_j C_{lj})(R_0 + r_1 l_1 + r_2 l_2) \quad (2)$$

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 $t_\tau = (s_\tau, e_\tau, D_{req_\tau})$, where s_τ is a source pin, e_τ is a sink pin, and D_{req_τ} 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_τ to e_τ, in this case three paths, must be less than D_{req_τ} (Fig. 1(c)). We have to get the layout satisfying all elements of the set of timing constraints $\mathcal{T}$.

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 $t_\tau \in \mathcal{T}$, we define a *critical path* denoted by $p_\pi = (\mathcal{M}_\pi, \mathcal{N}_\pi)$ as any path whose source is s_τ and sink is e_τ, where $\mathcal{M}_\pi$ is a set of cells which are on the critical path and $\mathcal{N}_\pi$ 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 $t_\tau \in \mathcal{T}_\tau$. Let D_{req_π} be the required propagation delay of $p_\pi \in \mathcal{P}$, and let D_{act_π} be the actual propagation delay of $p_\pi \in \mathcal{P}$.

For every timing constraint $t_\tau \in \mathcal{T}$, 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 $\mathcal{M}_\tau = \bigcup_{\forall p_\pi \in \mathcal{P}_\tau} \mathcal{M}_\pi$, and $\mathcal{N}_\tau = \bigcup_{\forall p_\pi \in \mathcal{P}_\tau} \mathcal{N}_\pi$. 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_τ is represented as a directed acyclic graph(Fig. 1(b)), in which the source is s_τ and the destination is t_τ. Let $\mathcal{C}$ be a set of constrained circuits, and let $D_{act_\tau} = \max_{\forall p_\pi \in \mathcal{P}_\tau} D_{act_\pi}$ be the actual propagation delay time from s_τ to e_τ. Figure 1 shows an example of our definition. (a) is a timing constraint $t_\tau = (s_\tau, t_\tau, D_\tau)$, and (b) is the constrained circuit corresponding to t_τ. 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 $\mathcal{T}$, 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 $\mathcal{T}$ □

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

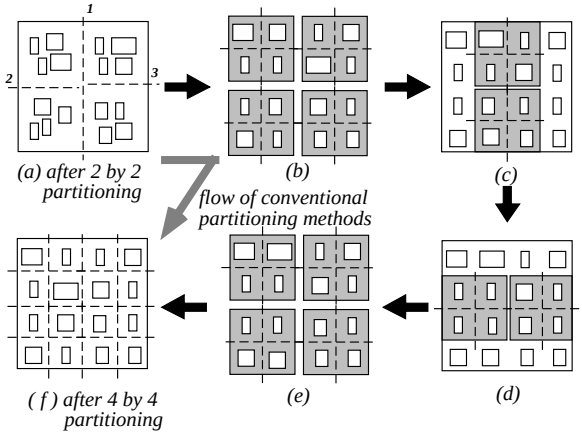


Fig. 2. Flow of the proposed hierarchical quadratic partitioning.

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* g_{cut_i} here, for a cell m_i means the decreasing number of cut size, which is the number of nets crossing the cut line, if m_i is moved to the opposite side. Once a cell is moved actually, it is locked and never moved any more. Then the gains of all cells connecting to that cell are updated. These operations are repeated iteratively until all cells are locked.

In addition to the gain for minimizing the cut size, we introduce another three gains. Firstly, we introduce the gain to handle timing constraints. Let *slack* for each critical path $p_\pi \in \mathcal{P}$ be defined as follows.

$$slack_\pi = \min(D_{req\pi} - D_{act\pi}, 0). \tag{3}$$

The slack means the margin of the delay time of p_π against the required delay time. If a cell m_i is moved to the opposite side, the sum of the difference of slack of paths containing m_i can be written as

$$g_{slack_i} = \sum_{\forall p_\pi \in \mathcal{P} \mid m_i \in \mathcal{M}_\pi} (slack'_\pi - slack_\pi), \tag{4}$$

where $slack_\pi$ is the current slack of a critical path p_π, and $slack'_\pi$ is the slack of a critical path p_π after moving the cell. We call g_{slack_i} the *slack gain*. To move the cell with large slack gain means to decrease the delay time of paths which contain it and violate their required delay time.

Next, we explain another gain to consider terminal positions of nets. We consider a vertical(resp. horizontal) partitioning shown in Fig. 3. Let $\mathcal{M}m_i$ be a set of cells connecting with cell

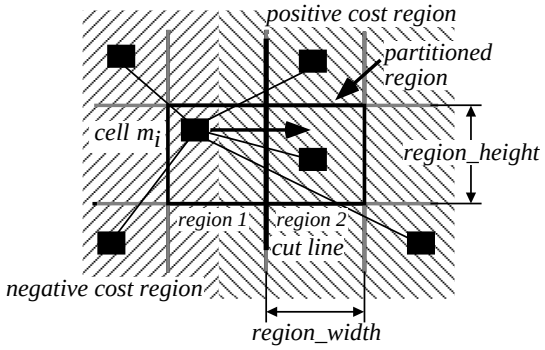


Fig. 3. $cost_{term_i}$ of a horizontal partitioning.

$m_i \in \mathcal{M}$ and let x_i, x_j(resp. y_i, y_j) be a x-coordinate (resp. y-coordinate) of cell $m_i \in \mathcal{M}$ and $m_j \in \mathcal{M}m_i$, respectively. 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 m_i as

$$cost_{term_i} = \sum_{m_j \in \mathcal{M}m_i} \left((\frac{|x_j - x_i|}{region\_width} + 1) \cdot k_{ij} \right), \tag{5}$$

where $k_{ij} = -1$ if $x_j - x_i = 0$, otherwise $k_{ij} = \frac{x_j - x_i}{|x_j - x_i|}$. For example in Fig. 3, the cells placed at the left part of the cell m_i have negative costs, and the cells placed at right part of the cell m_i have positive costs. Then, the gain considering the terminal positions, called *terminal gain*, is defined as

$$g_{term_i} = cost_{term_i} - cost'_{term_i} \tag{6}$$

where $cost_{term_i}$ and $cost'_{term_i}$ are the costs before and after moving cell m_i, respectively. From the gain g_{term_i}, we can realize the same effect of the terminal propagation method[4] in a short computation time.

Finally, we define the gain, called *wire gain*, to consider the wire length of nets. Because the minimization of the cut size doesn't always minimize the total wire length of nets, we explicitly consider the total wire length during the partitioning. Let $wire_i$ be the total wire length of nets connecting with cell m_i. In other words, $wire_i$ is the sum of the half perimeters of the enclosing minimum rectangle of all pins of nets connecting with the cell m_i. Then we define the wire gain as

$$g_{wire_i} = wire_i - wire'_i \tag{7}$$

where $wire_i$ and $wire'_i$ is the total wire length before and after moving cell m_i, respectively.

Consequently, the gain of a cell m_i is defined as the sum of above four gains, that is,

$$gain_i = \alpha \times g_{cut_i} + \beta \times g_{slack_i} + \gamma \times g_{term_i} + \delta \times g_{wire_i}, \tag{8}$$

where α, β, γ, and δ are positive constants. In our experiment, we set $\alpha = 1$, $\beta = 3$, $\gamma = 4$, and $\delta = 2$. In the proposed method, the above mentioned bi-partitioning method is hierarchically applied by shifting the partitioned region as shown

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 $\mathcal{L}_{mov} = (\mathcal{M}_{mov}, \mathcal{N}_{mov})$, where $\mathcal{M}_{mov}$ is the set of cells, called *movable cells*, of the subcircuit, and $\mathcal{N}_{mov}$ is the set of nets, called *movable nets*, connecting to at least one movable cells. The cells other than movable cells are called *fixed cells* and their set is represented by $\mathcal{M}_{fix}$. The nets other than movable nets are called *fixed nets* and their set is represented by $\mathcal{N}_{fix}$.

The target subcircuit is selected by the following algorithm. In the algorithm, $rand_i(0,1)$ 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 $\mathcal{P}_\pi$ with large violation ratio;
Step 2 : $\mathcal{M}_{mov} = \mathcal{M}_\pi$; $\mathcal{M}_{fix} = \mathcal{M} - \mathcal{M}_{mov}$;
$\mathcal{N}_{mov} = \mathcal{N}_\pi$; $\mathcal{N}_{fix} = \mathcal{N} - \mathcal{N}_{mov}$;
Step 3 : Calculate connectivity of cells of $\mathcal{M}_{fix}$;
Step 4 : If $2 \times (|\mathcal{M}_{mov}| + |\mathcal{N}_{mov}|) \geq$ (*preset value*), then output $\mathcal{M}_{mov}$ and $\mathcal{N}_{mov}$ and stop;
Step 5 : Search a cell $m_i \in \mathcal{M}_{fix}$ in decreasing order of its connectivity until the condition $rand_i(0,1) \geq$ (*preset value*) is satisfied;
Step 6 : $\mathcal{M}_{mov} = \mathcal{M}_{mov} \cup \{m_i\}$; $\mathcal{M}_{fix} = \mathcal{M}_{fix} - \{m_i\}$;
$\mathcal{N}_{mov} = \mathcal{N}_{mov} \cup \mathcal{N}_i$; $\mathcal{N}_{fix} = \mathcal{N}_{fix} - \mathcal{N}_i$;
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., $\frac{D_{act_\pi}}{D_{req_\pi}}$. The candidates for the critical path are selected from constrained circuits which are the largest 10∼20 percent in the 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(See. 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).

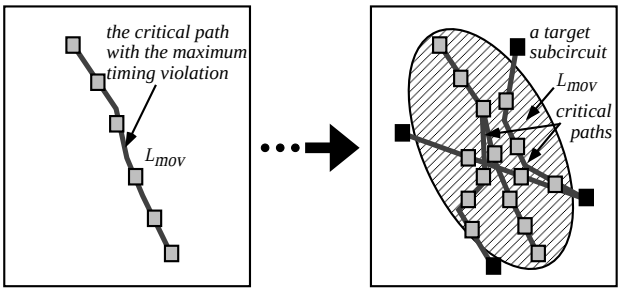


Fig. 4. Selection of a target subcircuit.

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_j which are the XY coordinates of a cell $m_j \in \mathcal{M}_{mov}$. 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 $n_i \in \mathcal{N}_{mov}$ 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 inequations.

$$CN1: \left.\begin{array}{lll} x_j - x_k \leq w_i \\ y_j - y_k \leq h_i \end{array}\right\} \begin{array}{l} \forall m_j \neq m_k \in \mathcal{M}_i \cap \mathcal{M}_{mov}, \\ \forall n_i \in \mathcal{N}_{mov} \end{array}$$

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 inequations are needed.

$$CN2: \left.\begin{array}{lll} x_j - X_{min_i} \leq w_i \\ X_{max_i} - x_j \leq w_i \\ X_{max_i} - X_{min_i} \leq w_i \\ y_j - Y_{min_i} \leq h_i \\ Y_{max_i} - y_j \leq h_i \\ Y_{max_i} - Y_{min_i} \leq h_i \end{array}\right\} \begin{array}{l} \forall m_j \in \mathcal{M}_i \cap \mathcal{M}_{mov}, \\ \forall n_i \in \mathcal{N}_{mov} \end{array}$$

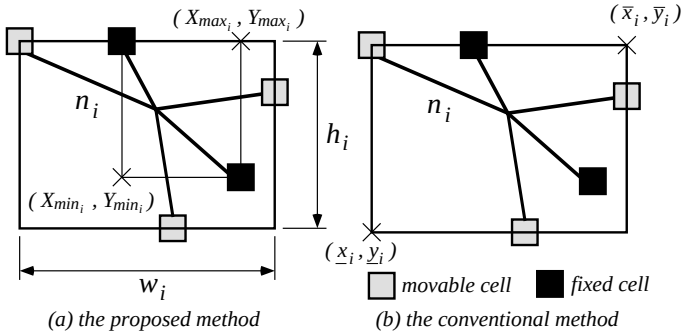


Fig. 5. Variables for movable net.

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 n_i.

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 $\mathcal{L}_{mov}$ 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.

$$
\begin{aligned}
CP: & \sum_{\forall n_i \in \mathcal{N}_\pi \cap \mathcal{N}_{mov}} d_i(w_i, h_i) \\
+ & \sum_{\forall n_i \in \mathcal{N}_\pi \cap \mathcal{N}_{fix}} d_i(X_{max_i} - X_{min_i}, Y_{max_i} - Y_{min_i}) \\
+ & \sum_{\forall m_j \in \mathcal{M}_\pi} D_{switch_j} \quad \leq \quad D_{req_\pi}, \\
& \forall p_\pi \in \mathcal{P} \mid \mathcal{M}_\pi \cap \mathcal{M}_{mov} \neq \emptyset,
\end{aligned}
$$

where D_{switch_j} is the switching delay of cell m_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.

$$
\begin{aligned}
\textit{minimize} \quad & : \quad \sum_{\forall n_i \in \mathcal{N}_{mov}} \alpha_i (w_i^2 + h_i^2) \qquad (9) \\
\textit{subject to} \quad & : \quad CN1 \cup CN2 \cup CP
\end{aligned}
$$

where α_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* $>$ ($preset\ value$), then stop;

Step 3 : Select a target subcircuit $\mathcal{L}_{mov}$ to be improved;

Step 4 : Find all critical paths which have the cells of $\mathcal{M}_{mov}$;

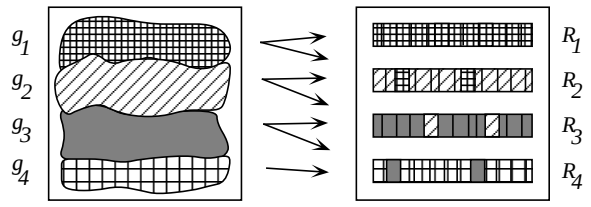
Step 5 : Formulate the nonlinear programming problem (9) and solve it;

Step 6 : Perform timing verification to the constrained circuits which have cells of $\mathcal{M}_{mov}$;

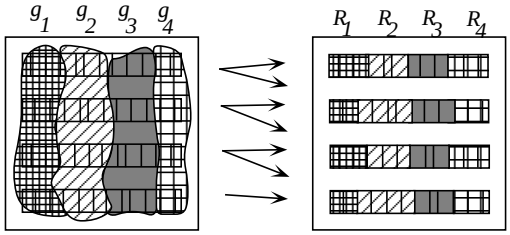
Step 7 : *LoopNumber* = *LoopNumber* + 1 and go to **Step 2**;

□

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



(a) assignment of y-direction



(b) assignment of x-direction

Fig. 6. Row Assignment.

repeated until the maximum violation ratio is less than a predetermined 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 R be the number of cell rows, and let $\mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_R$ be the set of cells of each row. All cells have areas, and let $a(\mathcal{M})$ be the sum of area of the cells in $\mathcal{M}$. 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 $a(\mathcal{R}_i) \leq A, i = 1, \ldots, R$. The rows have y coordinates $Y_1, Y_2, \ldots, Y_R$.

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 R groups, $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_R$. For each $\mathcal{G}_i, i = 1, \ldots, R$, they have capacities, which are $A_i = A$, for $i = 2, \ldots, R$ 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 $a(\mathcal{G}_i) \geq A$, if the cells in group $\mathcal{G}_i$ which are not assigned to slots of the cell row $\mathcal{R}_i$ but slots of the cell row $\mathcal{R}_{i+1}$, then the cells assigned to $\mathcal{R}_{i+1}$ are added to the next group $\mathcal{G}_{i+1}$ and reassigned in the next linear assignment problem.

The linear assignment problem that the cells in the group $\mathcal{G}_i$ are assigned to slots of the cell row $\mathcal{R}_i$ is formulated as follows.

$$
\begin{aligned}
&\textit{minimize}: && \textstyle\sum_{m_j \in \mathcal{G}_i} \sum_{s_k \in \mathcal{S}_i} c_{jk} z_{jk} \\
&\textit{subject to}: && \textstyle\sum_{s_k \in \mathcal{S}_i} z_{jk} = 1, \qquad \forall m_j \in \mathcal{G}_i, \\
& && \textstyle\sum_{m_j \in \mathcal{G}_i} z_{jk} = 1, \qquad \forall s_k \in \mathcal{S}_i, \\
& && z_{jk} \geq 0, \qquad \forall m_j \in \mathcal{G}_i, \qquad \forall s_k \in \mathcal{S}_i
\end{aligned}
$$

where $\mathcal{S}_i$ is a set of slots in which all cells in $\mathcal{G}_i$ are assigned, and c_{jk} is a cost with which the cell m_j is assigned to the slot s_k. The cost c_{jk} is defined as

$$c_{jk} = \textstyle\sum_{n_i \in \mathcal{N}_j} \Delta l_i(m_j, s_k) f(\omega_i), \tag{10}$$

where $\Delta l_i(m_j, s_k)$ 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 ω_i of net n_i and if ω_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 $t_\tau \in \mathcal{T}$, D_{req_τ} is the required delay and D_{act_τ} is the actual delay. We reflect the degree of the violation on ω_i as

$$\omega_i = \max_{\forall \mathcal{L}_\tau = \{\tau | n_i \in \mathcal{N}_\tau\}} \frac{D_{act_\tau}}{D_{req_\tau}}. \tag{11}$$

The timing driven row assignment algorithm is as follows.

[Timing Driven Row Assignment]

Step 1 : *LoopNumber* = 0;
/* Row assignment of y-direction */
Step 2 : Construct groups $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_R$ based on y coordinates of cells;
Step 3 : $i = 1$;
Step 4 : For all cells $m_j \in \mathcal{G}_i$, $s_k \in \mathcal{S}_i$, compute c_{jk} and solve the linear assignment problem;
Step 5 : For all cells $m_j \in \mathcal{R}_i$, update their coordinates.
Step 6 : $\mathcal{G}_{i+1} = \mathcal{G}_{i+1} \cup \mathcal{R}_{i+1}$.
Step 7 : If $i < R$, then $i = i + 1$ go to **Step 4**, else terminate;
/* Row assignment of x-direction */
Step 8 : Construct groups $\mathcal{G}'_1, \mathcal{G}'_2, \ldots, \mathcal{G}'_R$ based on x coordinates of cells;
Step 9 : $i = 1$;
Step 10 : For all cells $m_j \in \mathcal{G}'_i, s_k \in \mathcal{S}_i{}'$, compute c_{jk} and solve the linear assignment problem;
Step 11 : For all cells $m_j \in \mathcal{R}_i{}'$, update their coordinates.

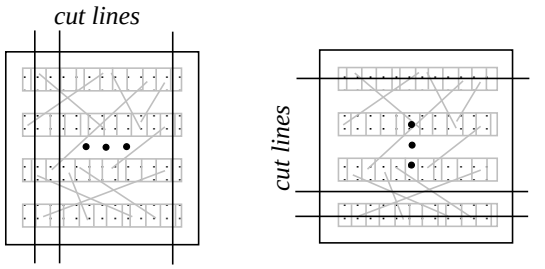
Step 12 : $\mathcal{G}'_{i+1} = \mathcal{G}'_{i+1} \cup \mathcal{R}'_{i+1}$.
Step 13 : If $i < R'$, then $i = i + 1$ and go to **Step 10**;
Step 14 : If there is no improvement or *LoopNumber* = (*preset number*), then terminate, else *LoopNumber* =*LoopNumber* + 1 go to **Step 2**;
□

IV Experimental Results

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.* = $\max_{\forall t_r \in \mathcal{T}} D_{act_r}/D_{req_r}$, 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.* = $\frac{1}{|\mathcal{T}|}\sum_{\forall t_r \in \mathcal{T}} D_{act_r}/D_{req_r}$ and *length* is the total wire length estimated by the Manhattan distance (λ). 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(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.



(a) the horizontal direction *(b) the vertical direction*

Fig. 7. The cut sizes of the horizontal and vertical directions.

TABLE I
Characteristics of experimental data.

Data	#cells	#nets	#I/Os	#rows	#cons
fract	125	163	24	6	38
primary1	752	1266	81	16	398
primary2	2907	3817	107	36	877
biomed	6417	7052	97	40	1302
avq.small	21854	22124	64	70	6064
avq.large	25114	25384	64	72	6064
s1494	390	425	27	9	70
s9234	917	1028	75	12	412
s38417	7572	7734	134	37	2619
s35932	11838	12228	355	45	3488
C1	493	607	73	8	82
C2	590	1277	373	9	270
C3	682	804	72	10	246
C5	1081	1560	301	13	334
C6	1037	1516	301	14	334
C7	2150	2678	315	18	405

#cons : the number of timing constraints.

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

TABLE II
THE RESULTS OF THE PROPOSED METHOD POPINS AND RITUAL.

Data	*Method*	*#vio.*	*Delay*		*length*(λ)	*#h-cuts*		*#v-cuts*		*time*
			Max.	Ave.		Max.	Ave.	Max.	Ave.	*(sec)*
fract	POPINS	0	0.85	0.32	45225	34	24	33	26	2361
primary1	POPINS	0	0.69	0.21	1128245	164	119	137	96	2237
primary2	POPINS	0	0.60	0.36	4128324	448	241	320	223	3680
biomed	POPINS	0	0.54	0.34	4128324	510	341	391	308	9168
avq.small	POPINS	0	0.88	0.18	23848188	1463	949	1715	1143	250884
avq.large	POPINS	0	0.84	0.20	28323022	1643	1151	1380	1019	281593
s1494	POPINS	0	0.67	0.39	257863	90	61	79	58	2477
	RITUAL	0	0.65	0.40	373477	150	88	120	93	285
s9234	POPINS	0	0.70	0.31	509156	146	100	129	93	9437
	RITUAL	0	0.67	0.31	583559	178	111	186	143	250
s38417	POPINS	0	0.70	0.23	4434695	519	339	308	244	8890
	RITUAL	0	0.58	0.24	6930625	793	539	566	393	10172
s35932	POPINS	0	0.74	0.31	9759746	928	636	547	410	21640
	RITUAL	0	0.75	0.35	11630953	567	733	831	1089	132542
C1	POPINS	0	0.58	0.53	243765	92	71	81	62	2803
	RITUAL	0	0.59	0.54	303696	116	83	128	111	45
C2	POPINS	0	0.70	0.38	375022	166	133	165	122	2432
	RITUAL	0	0.67	0.36	442748	123	94	315	274	108
C3	POPINS	0	0.71	0.39	475808	132	98	119	91	6580
	RITUAL	0	0.69	0.39	550509	160	113	175	137	163
C5	POPINS	0	0.65	0.28	1051320	321	226	291	196	3133
	RITUAL	0	0.63	0.29	1286249	330	257	458	376	243
C6	POPINS	0	0.65	0.29	994319	304	218	259	194	2666
	RITUAL	0	0.65	0.28	1186017	347	245	441	372	627
C7	POPINS	0	0.80	0.36	1462247	298	217	248	192	18145
	RITUAL	0	0.72	0.35	1727330	276	190	409	348	436

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

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

[1] H. B. Bakoglu: "Circuits, Interconnections, and Packaging for VLSI," Addison-Wesley Publishing Company, Inc. (1990).

[2] M. Burstein and M. N. Youssef: "Timing influenced layout design," Proc. 22nd Design Automation Conference, pp. 124–130 (1985).

[3] W. E. Donath, R. J. Norman, B. K. Agrawal, S. E. Bello, S. Y. Han, J. M. Kurtzberg, P. Lowy and R. I. MeMillan: "Timing driven placement using complete path delays," Proc. 27th Design Automation Conference, pp. 84–89 (1990).

[4] A. E. Dunlop and B. W. Kernighan: "A procedure for placement of standard-cell VLSI circuits," IEEE Trans. on Comput.-Aided Des. Integrated Circuits and Syst., 4, 1, pp. 92–98 (1985).

[5] W. C. Elmore: "The transient response of damped linear networks with particular regard to wideband amplifiers," J. Appl. Phys., Vol. 19, pp. 55–63 (1948).

[6] C. M. Fiduccia and R. M. Mattheyses: "A linear-time heuristic for improving network partitions," Proc. 19th Design Automation Conference, pp. 175–181 (1982).

[7] T. Gao, P. M. Vaidya and C. L. Liu: "A new performance driven placement algorithm," Proc. Int'l Conference on Comput. -Aided Des., pp. 44–47 (1991).

[8] T. Hamada, C.-. K. Cheng and P. M. Chau: "Prime: A timing-driven placement tool using a piecewise linear resistive network approach," Proc. 30th Design Automation Conference, pp. 531–536 (1993).

[9] T. Hasegawa: "A new placement algorithm minimizing path delay," Proc. Int. Symp. on Circuits and Systems, pp. 2052–2055 (1991).

[10] P. S. Hauge, R. Nair and E. J. Yoffa: "Circuit placement for predictable performance," Proc. Int'l Conference on Comput. -Aided Des., pp. 88–91 (1987).

[11] M. A. B. Jackson and E. S. Kuh: "Performance-driven placement of cell based IC's," Proc. 27th Design Automation Conference, pp. 370–375 (1989).

[12] M. A. B. Jackson and E. S. Kuh: "A fast algorithm for performance-driven placement," Proc. Int'l Conference on Comput. -Aided Des., pp. 328–331 (1990).

[13] H. Konno and H. Yamashita: "Nonlinear programming," Nikkagiren Publisher, in Japanese (1978).

[14] E. S. Kuh and M. Shih: "Recent advances in timing-driven physical design," Proc. Asia-Pacific Conference on Circuits and Systems, pp. 23–28 (1992).

[15] M. Marek-Sadowska and S. P. Lin: "Timing driven placement," Proc. Int'l Conference on Comput. -Aided Des., pp. 94–97 (1989).

[16] E. M. Sentovich, K. J. Singh, L. Lavagno, C. Moon, R. Murgai, A. Saldabha, H. Savoj, P. R. Stephan, R. K. Brayton and A. Sangiovanni-Vincentelli: "SIS: A system for sequential circuit synthesis," Technical Report No. UCB/ERL M92/41, University of California, Berkeley (1992).

[17] A. Srinivasan, K. Chaudhary and E. S. Kuh: "RITUAL: A performance-driven placement algorithm," IEEE Trans. on Circuits and Systems II, 39, 11, pp. 825–839 (1992).

[18] S. Sutanthavibul and E. Shragowitz: "An adaptive timing-driven layout for high speed VLSI," Proc. 27th Design Automation Conference, pp. 90–95 (1990).

[19] S. Wakabayashi, H. Kusumoto, H. Mishima, T. Koide and N. Yoshida: "Gate array placement based on mincut partitioning with path delay constraints," Proc. Int. Symp. on Circuits and Systems, pp. 2059–2062 (1993).