VLSI DESIGN
1998, Vol. 7, No. 4, pp. 385–399
Reprints available directly from the publisher
Photocopying permitted by license only

Macro-Cell Placement for Custom-Chip Design Using Self-Organizing Fuzzy Technique

RAY-I CHANG[a,b,*] and PEI-YUNG HSIAO[b]

[a]*Institute of Information Science, Academia Sinica, Taipei, Taiwan, ROC;*
[b]*Department of Computer and Information Science, National Chiao Tung University, 1001 Ta Hsueh Rd., Hsinchu, Taiwan 30050, ROC*

(Received 9 March 1993; In final form 28 June 1995)

In this paper, a new optimization technique called *SOFT* (*self-organizing fuzzy technique*) is proposed to solve the macro-cell placement problem. In SOFT, different criteria are simultaneously accounted by a novel *fuzzy gain function* which models expert knowledge to control the optimization process. The presented procedure is an adaptation of *Kohonen's self-organization algorithm* which is well suited for implementation on massively parallel architecture for fast computing. The MCNC benchmark examples are presented to verify the performance and feasibility of SOFT. Comparisons are made with the Hopfield network, SOAP and TimberWolf MC5.6. Experiments show that the proposed method yields an average of 17% improvement in total wire length compared with previous methods. Large size problems with 225 and 1024 arbitrarily-sized macro-cells are also presented.

Keywords: Custom-chip design, macro-cell placement, self-organization, fuzzy set, neural networks, optimization

1. INTRODUCTION

Macro-cell placement which packs arbitrarily-sized circuit blocks into a given layout region is a very important step in VLSI custom-chip design, since it has a pronounced effect on the final circuit layout. This optimization problem has been proven to be NP-hard [4] and must satisfy several contradictory criteria, such as shorter wire length, smaller chip area, less module overlap, and a variety of other constraints. Over the years, a wide repertoire of solution methods have been proposed with varying success [5, 7, 19–23]; these include partitioning-based methods, branch and bound, cluster growth, simulated annealing, genetic-based methods, force-directed and analytical methods. Partitioning-based methods [16] place cells on either side of a partition in such a manner as to minimize the number of interconnections crossing the partition. Cluster growth methods [22] apply a pre-specified select-function and place-function to select unplaced cells and place cells in the

*Corresponding author.

placement region. Both of these methods are greedy constructive approaches that frequently become trapped in local minima. The constructive methods create placement in an incremental manner that produces a complete placement only when the method terminates. The branch and bound method which seeks a solution by tracing a logical tree structure is able to guarantee optimum results, but the run time is excessive for reasonably sized problems. This method is also a constructive approach. The simulated annealing algorithm [19] with probabilistic hill climbing technique is an iterative approach. Iterative approaches improve an initial placement by repeatedly modifying it and then terminate when a design criterion is met. Thus, a complete placement is available at every stage of the modification cycle. Although simulated annealing has been suggested for obtaining optimal solutions, it requires an enormous amount of time. The genetic-based method derived from biological phenomena is ineffective unless a clever representation scheme is devised to represent the physical placement as a genetic code. This is also an iterative approach. Force-directed methods and analytical methods [20–21] are iterative approaches that accept the next configuration only if the value of the objective function is reduced. They are characterized by their inherently greedy nature for fast convergence. The disadvantages of these methods are that they do not permit changes to previous decisions and they get trapped in local optima.

Although a number of heuristic algorithms have been proposed, as described above, all of these algorithms are inherently sequential and unable to efficiently exploit massive parallel architecture. Recent progress in software and VLSI technology has motivated attempts to exploit more fully the parallelism of neural networks to solve NP problems [8–13, 27–28]. For example, a Hopfield network [8] with a symmetrical interconnection matrix and a sigmoid gain function was applied to TSP problems in early 1985. Persky [9] used this model to solve a one-dimensional cell placement problem. Sriram and Kang [11] introduced a modified Hopfield model for a two-dimensional module placement problem. Date *et al.* [23] presented an LSI module placement algorithm using a Hopfield network. Yu [12] attempted to obtain a placement result using this model, but not with great success. His experimental results show that this model gives much the same solutions as the min-cut algorithm. Since it suffers from the inherent limitation of the Hopfield model, it becomes stuck at local minima. If one tries to improve the quality of the result, on the other hand, the algorithm may converge to illegal solutions.

In 1990, Hemani [3] used Kohonen's self-organization rule to solve the two-dimensional cell placement problem. In his method, the output neurons are organized into a rectangular grid and correspond to the locations of cells. Input sample vectors are the columns of the connectivity matrix (*i.e.*, the sample vector representing cell i is the i-th column of the connectivity matrix). Experimental results show that the columns of the connectivity matrix used in Hemani's method are not enough to preserve the complex topological relation among cells. Recently, Kim [1] has presented an approach called SOAP (self-organization assisted placement) in which Kohonen's self-organization algorithm is used as a preprocessor of other heuristic algorithms. In SOAP, output neurons and connection synapses between output neurons correspond to cells and wire nets between cells, respectively. The input sample vectors are generated randomly. Unless enormous amounts of random sample vectors are generated, the quality of the results is not guaranteed. Moreover, the sizes of the modules are not taken into consideration in obtaining the optimal positions of each module. This is also the major drawback of the other previous neural-network methods.

Most previous neural network based algorithms are suitable only for placement on a checkerboard model, in which all cells are assumed to be square and of equal size. Moreover, none of these previous algorithms explicitly addresses the presence of multiple contradictory criteria in the cell

placement problem. In this paper, a general optimization method called *SOFT* (*self-organizing fuzzy technique*) is proposed to resolve the macro-cell placement problem. The proposed method which takes cell shapes into consideration can easily model arbitrarily-sized macro-cells, hence it can be expected to provide more accurate placement models. Furthermore, the proposed method can minimize the interconnection wire length without any module overlap. In this paper, we also use an adaptation of Kohonen's self-organizing neural network [6] to model the optimization problem, since this network is well suited for implementation on massively parallel architecture for fast computing. Experiments on many test problems, including the MCNC benchmark circuits Ami33 and Xerox, have been conducted to verify the performance and feasibility of the proposed technique. Comparisons have been made with both neural-network approaches (the Hopfield network [2] and SOAP [1]) and non-neural-network approaches (TimberWolf MC5.6). Large size problems with 225 and 1024 arbitrarily-sized macro-cells are also presented. The simulation results are quite encouraging: the total wire lengths obtained are average 17% shorter than those obtained using previous methods without any module overlap. We also compare results obtained with and without the fuzzy gain function and give our comments from this comparison. Moreover, the proposed self-organizing neural network contains fewer neurons and connection synapses than the best neuron models known for this problem. The proposed model can easily be extended to treat cell placement in a three-dimensional region or over a non-planar surface.

The remainder of this paper is organized as follows. The problem definition and the notation used in the paper are stated in Section 2. Next, Section 3 describes the basic fuzzy set theory and neural network model used. In Section 4, descriptions and analyses of the execution of each step of the SOFT algorithm are given, and the proposed self-organizing neural network and the fuzzy optimization technique are also introduced. Experiments and results are presented in Section 5. Finally, Section 6 gives the conclusion.

2. PROBLEM DEFINITION

Classical placement algorithms assume that placement quality is a function of the interconnection wires and is not affected by the shapes of cells. These algorithms model the circuit blocks as point-objects. Sometimes, they also define the placement region as a pre-defined array of slots for gate arrays or rows for standard cell design styles. Unlike these traditional layout styles, custom-chip layout systems must model the circuit blocks with two dimensions, since both their height and width are irregular and can affect the layout result. Macro-cell placement is an important step in custom-chip design. In macro-cell placement, circuit blocks are not restricted to having similar sizes or being arranged together in regular rows and columns. This lack of restrictions adds considerable complexity, in which, the interactions (interconnections and overlaps) among circuit blocks are more difficult to compute. In this paper, a simple but effective modeling method is proposed that can easily take cell shapes into consideration to yield more accurate placement results.

The problem inputs are a $w \times h$ placement region, a set of macro-cells $\{b_i | i = 1, \ldots, m\}$, and their $m \times m$ connectivity matrix $CM = [cm_{ij}]_{1 \le i,j \le m}$. The connectivity cm_{ij} specifies the number of connection wires between cell b_i and cell b_j. A multi-terminal netlist (say N_i) which connects more than two modules (say p_i modules) is represented by fully-connected q_i connections in this paper:

$$q_i = \frac{p_i \times (p_i - 1)}{2}$$

As shown by Alon and Ascher [25], the strength of these wire connections can be specified as $2 \times c_i/q_i$. The common goal in macro-cell placement is to minimize total wire length and to avoid module overlap. Thus, the width w_i and the height h_i of

each cell b_i must be considered during the placement process. The cost function for total wire length C_{WL} and that for module overlap C_{MO} can be specified as follows:

$$C_{WL} = \sum_{i=1}^{m} \sum_{j=i+1}^{m} d_{ij} \times cm_{ij} \tag{1}$$

$$C_{MO} = \sum_{i=1}^{m} \sum_{j=i+1}^{m} MOX_{ij} \times MOY_{ij} \tag{2}$$

$$MOX_{ij} = \min\left(\max\left(0, \frac{w_i + w_j}{2} - |dx_{ij}|\right), w_i, w_j\right)$$
$$MOY_{ij} = \min\left(\max\left(0, \frac{h_i + h_j}{2} - |dy_{ij}|\right), h_i, h_j\right)$$

where the Euclidean distance d_{ij} and the $X(Y)$ coordinate distance $dx_{ij}(dy_{ij})$ between cell b_i and cell b_j are defined as follows:

$$d_{ij} = \sqrt{dx_{ij}^2 + dy_{ij}^2}$$
$$dx_{ij} = x_j - x_i$$
$$dy_{ij} = y_j - y_i$$

Note that optimizing the above cost equations is computationally very complicated. In this paper, we model the module overlap cost function heuristically using the distance between cells to make the computations simpler.

We use the distance ID_{ij} as the ideal minimum distance (ideal distance, for short) between cells b_i and b_j without module overlap. For example, the ideal distance is defined as $(h_i + h_j)/2$ if b_i and b_j are on a vertical line. In general, the ideal distance between cell b_i and cell b_j can simply be defined as follows:

$$ID_{ij} = \min\left(\frac{w_i + w_j}{2} \times \frac{d_{ij}}{|dx_{ij}|}, \frac{h_i + h_j}{2} \times \frac{d_{ij}}{|dy_{ij}|}\right) \tag{3}$$

Figure 1 shows four possible examples of the ideal distance between two arbitrarily-sized macro-cells b_i and b_j for actual-shape-module formulation. We

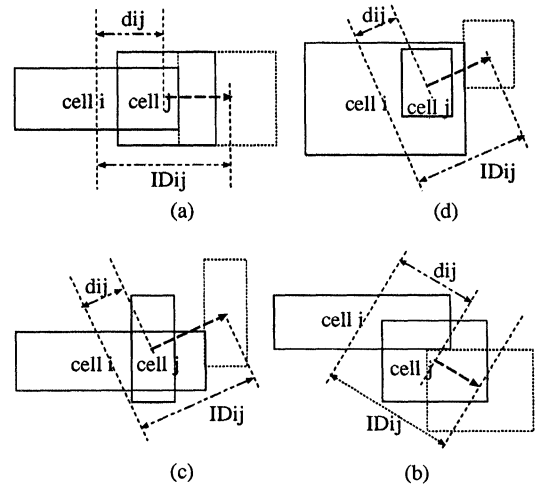

FIGURE 1 Four examples of the ideal distance between two macro-cells i and j.

can redefine the cost function of module overlap so as to preserve the ideal distance between cells. By Equations (2) and (3), the cost function to minimize module overlap can be easily formulated in terms of the ideal distance and defined as follows.

$$C_{MO} = \sum_{i=1}^{m} \sum_{j=i+1}^{m} MO_{ij}$$
$$MO_{ij} = \begin{cases} (ID_{ij} - d_{ij})^2 & ID_{ij} > d_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

This definition accurately reflects the module overlap between macro-cells. Moreover, it is easier to compute in computer simulations.

As defined above, our objective is to minimize total wire length without causing module overlap. Hence, by Equations (1) and (4), a possible objective function C for the macro-cell placement problem can be defined as follows.

$$C = g_{WL} \times C_{WL} + g_{MO} \times C_{MO}$$

where g_{WL} and g_{MO} are pre-specified gain terms. In the past, various strategies have been proposed to minimize the cost function. However, in reality, the criteria of less module overlap and shorter wire length are contradictory. The scaling of the gain

terms, *i.e.*, g_{WL} and g_{MO}, will heavily influence the performance of the system and the solution quality of the objective function. In this paper, a fuzzy optimization technique is proposed to resolve this problem. Moreover, a self-organizing neural network is proposed to model this problem that can reduce the execution time through the use of multiple processing elements (neurons) that allow parallel processing.

3. BASIC FUZZY THEORY AND THE NEURAL MODEL

The basic fuzzy set theory for the proposed method is briefly described in this section. The neural network model proposed for macro-cell placement problem is also presented.

3.1. Basic Fuzzy Set Theory

Investigation of the characteristics of expert knowledge shows that people often base decisions on imprecise or non-numerical information in everyday life. For example, measurements of importance may be represented by vague states, such as "low", "medium" and "high". In order to represent and manipulate vague data, in 1965 Zadeh introduced the fuzzy set theory, which processes non-statistical uncertainty. Fuzzy set theory is a generalization of conventional set theory that can interpret data in a very natural and intuitively plausible way to resolve various problems. Fuzzy states (or fuzzy subsets) are defined as the elements of a particular fuzzy set. Each subset is described by an appropriate word (*i.e.*, "low" or "high"). Imprecise or non-numerical information can be described by a membership function. Figure 2 shows an example with two fuzzy states, say, "early" and "late", of a variable, say, time t within a range $(t_0\, t_e)$. Here, the membership functions have trapezoidal shapes which specify the degree of membership in the fuzzy states. Formally, a fuzzy set is defined as follows [16]:

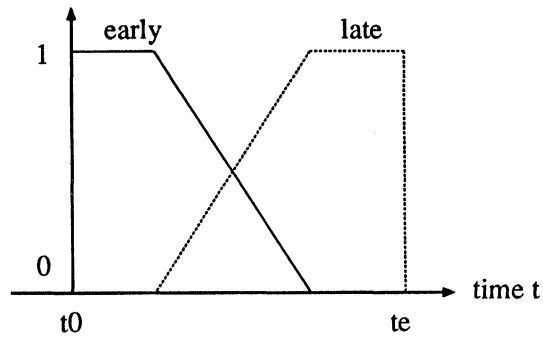

FIGURE 2 Two fuzzy states ("early" and "late") of a variable say, time t within a range $(t0, te)$.

DEFINITION 3.1.1 A fuzzy subset A of a universe of discourse X is defined as $A = \{(x, \mu_A(x))|$ all $x \in X\}$, where X is a space of points and $\mu_A(x)$ is a membership function of $x \in X$ which should be an element of A, and x belongs to the fuzzy set A with degree of membership $\mu_A(x)$.

Set operations, such as equality, containment, union, intersection, and complementation, are also defined in fuzzy set theory, and the basic operations on two fuzzy sets A and B are usually defined as follows:

$\mu_A(x)$ = membership function for a fuzzy set A of input x

$\mu_B(x)$ = membership function for a fuzzy set B of input x

$$\mu_{A=B}(x) \Leftrightarrow \mu_A(x) = \mu_B(x)$$
$$\mu_{A \subset B}(x) \Leftrightarrow \mu_A(x) \leq \mu_B(x)$$
$$\mu_{A \cap B}(x) = \min(\mu_A(x), \mu_B(x))$$
$$\mu_{A \cup B}(x) = \max(\mu_A(x), \mu_B(x))$$
$$\mu_{\bar{A}}(x) = 1 - \mu_A(x)$$

3.2. The Neural Network Model

Artificial neural networks were first discussed by McCullough and Pitts in 1943 to imitate the power of biological systems for data and information processing. These models typically consist of many simple neuron-like processing elements that interact via weighted connections. Each neuron has an activation value representing its state, which is

determined by inputs received from other neurons or by external inputs. Neural network models are a promising approach to develop feasible solutions to computationally intense problems. For example, in early 1985, neural networks were applied by Hopfield and Tank [8] to resolve the optimization problem. A neural network generates a feasible solution with values close to the values of an optimum solution in a reasonable amount of time.

In general, the solution to a problem is defined to correspond to the neuron states or their connection weights. Given the dimensions of a set of arbitrarily-sized macro-cells and the connection wires among them, the macro-cell placement problem is to locate the circuit blocks optimally within the specified region. This is a constrained mapping problem which determines the locations for all cells such that all the constraints are satisfied and a weighted sum of total wire length is minimized. It has been shown that a self-organizing neural network with Kohonen's learning algorithm can create an optimal two-dimensional feature map of higher dimensional sample vectors [13, 27–28]. Given the topological relation (connection) between each pair of neurons as a map, the algorithm applies the input sample vectors to adjust the weights of neurons. The objective of the self-organization algorithm is to iteratively adapt the weights between input neurons and output neurons, so that the output neurons become sensitive to different inputs in an organized way. This neural network model, which preserves the topological relation among neurons, is sometimes called a *topology-preserving map*. We shall refer to this topology-preserving map of macro-cells as a self-organizing *cell-map*.

In this paper, a neural network model based on Kohonen's self-organization rule is proposed to resolve the macro-cell placement problem. The relationship between the proposed self-organizing neuron model and the macro-cell placement model is illustrated in Figure 3, where output neurons in the neural network model correspond to the macro-cells. The connection weight between output neuron i and output neuron j is cm_{ij}, which

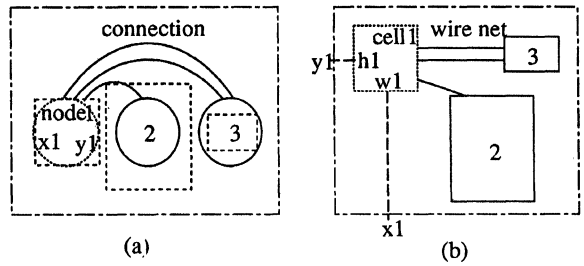

FIGURE 3 The relationship between (a) the proposed self-organizing neural network model and (b) the cell placement model.

corresponds to the strength of the connection wire nets. The j-th position weight of output neuron i, denoted by w_{ij}, corresponds to the j-th coordinate value of the position of cell b_i. In the case of placement on a two-dimensional space, $w_{i_1} = x_i$ and $w_{i_2} = y_i$. When the self-organizing algorithm is used, neurons connected closely in the topology are sensitive to sample vectors that are physically similar. Each neuron is sensitive to a particular input sample vector that represents the placement location of the related macro-cell. This method can easily be extended to handle placement with various constraints on cell connections and dimensions. For example, w_{i_3} is an additional weight needed for placement in a three-dimensional space to represent value of the Z coordinate.

4. THE MACRO-CELL PLACEMENT ALGORITHM

In this section, we first introduce the proposed self-organizing neural network with its learning algorithm. Next, the fuzzy gain function of SOFT is presented. Finally, the proposed macro-cell placement algorithm is described and analyzed in more detail.

4.1. Self-Organizing Neural Network

Kohonen's algorithm was originally applied to vector quantization, in which, sample vectors

provide the primary source of information. Thus, the solution quality and the system performance are more sensitive to the quality of the sample vectors. In contrast to vector quantization, sample vectors are seldom available in cell placement. In this paper, a new cell placement technique based on Kohonen's self-organization algorithm is proposed. We produce input sample vectors using spatial relations (connectivities and overlaps among macro-cells) and adapt the neural network using Kohonen's self-organizing rule. This algorithm is well suited for solving such self-organizing optimization problems (e.g., cell placement) whose sample vectors are not easily available.

In the traditional self-organization algorithm, the topological relations among output neurons are represented as *lateral interactions*. In this paper, we use these lateral interactions (stimulus of *excitation* and *inhibition*) between output neurons to produce sample vectors. Assume that $f_{WL}(d_{ij})$ is an excitation stimulus function that minimizes the wire length between cell b_i and cell b_j. The inhibition stimulus function, $f_{MO}(d_{ij})$, is presented to reduce the module overlap between cell b_i and cell b_j. As defined above, this inhibition stimulus function is calculated between cells to remove their module overlap. In this paper, for simplification, we preserve the ideal distance described in Section 2 to reduce the module overlap between cells. The stimulus functions $f_{WL}(d_{ij})$ and $f_{MO}(d_{ij})$, which minimize connected wire length and module overlap, respectively, can easily be formulated using the ideal distance and defined as follows:

$$f_{WL}(d_{ij}) = cm_{ij} \times \frac{d_{ij}}{ID_{ij}} \tag{5}$$

$$f_{MO}(d_{ij}) = \begin{cases} -\left(\frac{ID_{ij}}{d_{ij}}\right)^2 & ID_{ij} > d_{ij} \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

This model accurately reflects the wire connection and module overlap between cells. There is a rapid increase in the overlap penalty function and a slower increase in the wire length penalty function.

4.2. The Fuzzy Gain Function

We have analyzed the macro-cell placement of expert designers and found the following characteristics in their placement techniques:

(1) They first place the cells in a plane such that the topology of the cells satisfies the combined goal of shorter wire length and less module overlap.
(*In this early relative placement phase, the wire length criterion is more important than in the later phase*).

(2) Then, they remove the module overlap in the placement by moving the cells such that the total wire length is kept as low as possible.
(*In this late spacing phase, the module overlap criterion is more important*).

From the suggestion of the placement techniques of expert designers, we have four hypothetical inference rules involving three variables, t (time), c_1 (importance of wire length criterion), and c_2 (importance of module overlap criterion). It can be better understood if an "important" in a fuzzy set is interpreted as an "importance is high". Linguistic values for time t are defined as "early" and "late". Linguistic values for c_1 and c_2 are defined as "low", "medium", and "high". The gain function is defined as the membership function for "wire length criterion is important" (importance of wire length criterion), denoted by $\mu_{c_1}(t)$. Approximate reasoning based on linguistic variables and their values [17] is used to define the membership function for "wire length criterion is important" of "time". The detailed analysis is illustrated in Figure 4. In this paper, defuzzification is based on a centroid calculation, in which, the actual output value is the center of gravity of the shaded area. For simplification, we use an S-function $g(t)$ [16], $g_{WL}(t) = g(t)$, as the membership function to model $\mu_{c_1}(t)$ and use $g_{MO}(t) = 1-g(t)$ to model $\mu_{c_2}(t)$. By Equations (5) and

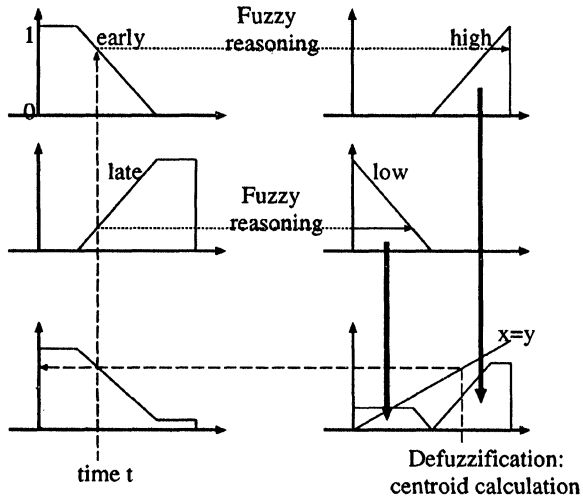

FIGURE 4 Approximate reasoning based on linguistic variables and their values used to define the membership function for "wire length criterion is important" of "time".

(6), the total stimulus function with the proposed fuzzy gain functions is defined as follows:

$$f(d_{ij}) = g_{\mathrm{WL}}(t) \times f_{\mathrm{WL}}(d_{ij}) + g_{\mathrm{MO}}(t) \times f_{\mathrm{MO}}(d_{ij}) \tag{7}$$

This is the first neural optimization technique that simultaneously considers the wire length and module overlap criteria for macro-cell placement.

4.3. Self-Organizing Fuzzy Technique

In this subsection, a brief overview of the SOFT algorithm will be given and then a more detailed description of each step of the algorithm will be given. Initially, we set the position weights as random values. For each iteration, a cell b_i is selected and its associated sample vector V_i is generated. The position weights associated with node i and its neighbors NE_i are adapted to make these cells more responsive to the current input V_i. This process is repeated until the system converges. The detailed description of the SOFT algorithm is given below.

Step 1. Initialization

Initially, the proposed algorithm sets all the position weights (x_i, y_i) of macro-cell b_i as random values around the center of the layout region $(w/2, h/2)$. The time variable t is set to 0.

Step 2. Winner and Sample Vector Selection

With each iteration, the algorithm randomly selects a macro-cell, say b_i, as the winner neuron in Kohonen's self-organizing algorithm. Equation (7) is then applied to produce the related sample vector $V_i = (vx_i, vy_i)$ by force $F_i = (fx_i, fy_i)$:

$$vx_i = x_i(t) + \frac{fx_i}{\|F_i\|}, \quad vy_i = y_i(t) + \frac{fy_i}{\|F_i\|}$$

$$\text{where } fx_i = \sum_{j=1}^{m} f(d_{ij}) \times \frac{dx_{ij}}{d_{ij}},$$

$$fy_i = \sum_{j=1}^{m} f(d_{ij}) \times \frac{dy_{ij}}{d_{ij}}$$

This step is designed to produce a sample vector that can minimize the total wire length and the module overlap. For example, Figure 5(a) shows a selected neuron, *i.e.*, neuron 3, and the lateral interactions that come from its neighbors. The summation of the lateral interactions is computed and the sample vector V_3 is generated. The sizes of cells are initially small and increase gradually during the placement process. Define $w_i(t) = w_i(0) \times g_{\mathrm{MO}}(t)$ and $h_i(t) = h_i(0) \times g_{\mathrm{MO}}(t)$. The

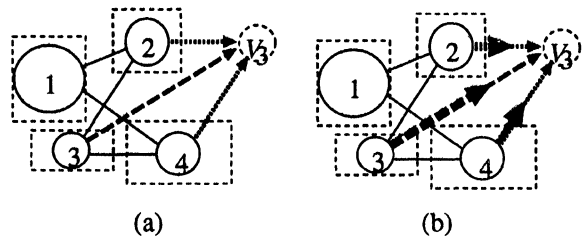

FIGURE 5 (a) Calculation of the summation of effects and selection of the sample vector. (b) The selected neuron and its neighbors are moving toward V3 with various weights.

gradual expansion of cell size introduces a kind of function smoothing [14]. Thus, the smaller the size of the cell, the larger the space for moving it will be. With a larger space for moving, the system will have more chances to converge to an optimal state. However, an abrupt change of cell size is generally not desirable because it will result in a significant distortion of the current placement.

Step 3. Self-Organizing Adaptation

Applying the sample vector and the pre-specified winner neuron described in *Step* 2, this learning step uses Kohonen's self-organizing algorithm to adjust the proposed neural network model. It updates the position weights of winner neuron i and its neighbors within $NE_i(t)$. For all such macro-cells, say b_k, the following adaptation functions are performed:

$$x_k(t+1) = f_x(x_k(t) + \eta(u,t) \times (vx_i - x_k(t)))$$
$$y_k(t+1) = f_y(y_k(t) + \eta(u,t) \times (vy_i - yk(t)))$$

A $w \times h$ placement region is given and the bounding functions, $fx()$ and $fy()$, are defined as follows:

$$f_x(x_i) = \begin{cases} w_i/2 & x \le w_i/2 \\ w - w_i/2 & x \ge w - w_i/2 \\ x & \text{otherwise} \end{cases}$$

$$f_y(y_i) = \begin{cases} h_i/2 & y \le h_i/2 \\ h - h_i/2 & y - h_i/2 \\ y & \text{otherwise} \end{cases}$$

$\eta(u,t)$ is a time-decreasing neighboring gain function and u is the topological distance between neuron k and neuron i. The moving distances of the cells are decreased as the number of iterations is increased or as u is increased. For example, in Figure 5(b), the selected neuron and its neighbors move toward V_3 with the movement vectors which are represented by arrows showing the direction and the magnitude of the movement. Figure 6 shows that the neighborhood $NE_j(t)$ of a neuron j at time t is a set of neurons lying within the topological distance $\delta(t)$ from the neuron j. Here the $\delta(t)$ determining the size of the neighborhood is a time-decreasing function.

Step 4. Stop or Next Iteration

Increase the time variable t by 1. Go to *Step 2* if the neural network model does not converge. If it converges, a placement solution is obtained and macro-cell b_i is placed on position (x_i, y_i). In this paper, we define the system to be convergent if the position weights have no distinct changes (or the time variable t is larger than a pre-specified large constant t_{max}).

4.4. Algorithm Analyses

As point out above, input sample vectors provide the primary information for the learning of self-organizing neural network such that the solution quality and the system performance are more sensitive to the quality of the input sample vectors. To demonstrate the performance of the proposed algorithm, we present a small but difficult example called NCTU8, in which, 8 cubic-connected macro-cells are placed on a 480×480 layout region. Figure 7 shows the first 300 sample vectors produced by the proposed method. The pre-specified topological relation of these 120×120

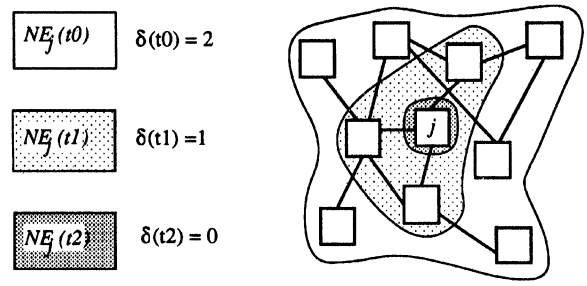

FIGURE 6 Node j is represented as the selected node. Regions with different fill patterns are represented as the neighborhood region at difference time, $t0 < t1 < t2$.

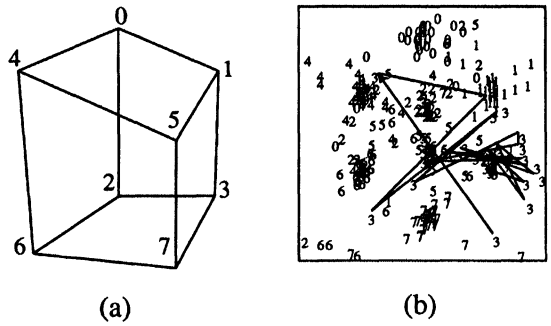

FIGURE 7 An illustration of the proposed method. (a) Input topological relation. (b) Sample vectors and input order of the sample vectors with label 3.

macro-cells is shown in Figure 7(a). Figure 7(b) shows the distribution of the produced sample vectors. The input sequence of the sample vectors with the winner neuron 3 is also demonstrated. It can easily be shown that the proposed method is better than Kim's SOAP method, in which, sample vectors are randomly generated and uniformly distributed on the placement region. Kim's self-organizing process is highly time-consuming, especially because it attempts to represent particular information by random data.

Figure 8 demonstrates the execution of the SOFT algorithm as it finds a solution for the placement of NCTU8. There are four development stages. The initial position weights are random values around the center of the given region, as shown in Figure 8(a). Next, cells are grouped and interchanged, as shown in Figure 8(b) and (c), to minimize the total wire length; finally, cells are spread out over the given region, as shown in Figure 8(d), without any module overlap. During the placement process, the variation in wire length at the early stage is high and it then gradually converges to a constant value. Moreover, module overlap gradually converges to zero. The large variation at the early stage is due to the large size of the neighborhood and the large gain value. As the process continues, the changes are reduced, because the size of the neighborhood and the magnitude of the gain term decrease. The total iteration time of the proposed algorithm is

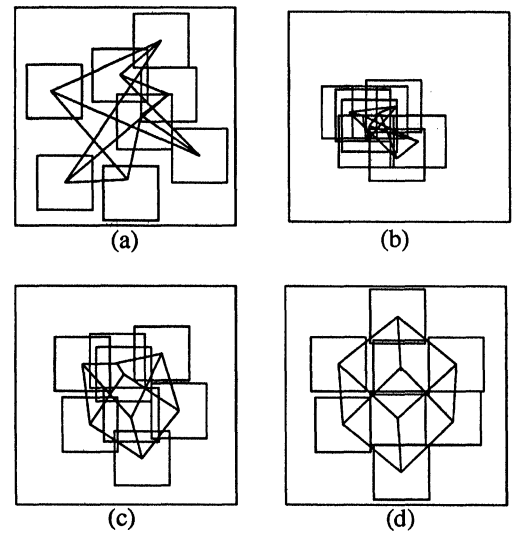

FIGURE 8 Four development stages for the placement of NCTU8 by SOFT. (a) The initial position values in the given region. Next, (b) cells are grouped and (c) interchanged to minimize wires. Finally, (d) cells are spread out over the given region without overlap.

independent of the number of input macro-cells and the number of wire nets. In other words, the execution time is bounded by a constant number if the algorithm is has been implemented by a real analog neural network.

Table I shows the number of neurons and connections used in the proposed model, in which, position weights can be stored in their corresponding output neurons and the input neurons can be removed. Three different models are used for comparison: Kita *et al.*'s Hopfield-like model [2], Hemani&Postula's self-organization model [3] and Kim & Kyung's SOAP [1], where w and h are the width and the height of the placement region, and s $(=w \times h)$ stands for the area of the placement region. Additionally, m and n represent the

TABLE I Comparisons of the number of neurons and connection weights (m is the number of cells, n is the number of wires and s is the size of region)

	[1]	[2]	[3]	Ours
Nodes	$O(m)$	$O(m+s)$	$O(m \times s)$	$O(m)$
Connections	$O(m+n)$	$O((m \times s)^2)$	$O(m \times s)$	$O(n)$

number of cells and the number of wire nets, respectively. The proposed approach uses O(m) neurons and O(n) connection weights; this is fewer than the best neuron models [1] known for cell placement problems.

5. EXPERIMENTS AND RESULTS

The proposed algorithm has been implemented in C language on a Sun Sparc IPC station running UNIX. Several macro-cell placement examples (*i.e.*, MCNC benchmark circuits: Xerox and Ami33) were presented to our solution procedure to prove that our algorithm can handle these problems. The large size problems with 225 and 1024 arbitrarily-sized macro-cells were also presented. Table II shows the number of cells, number of wires, and size of the placement region for the input test circuits. The results of running the proposed algorithm on various placement problems are discussed below.

The first example is an artificial, but difficult, problem from Reference [1] called $m4 \times 4$. This circuit has 16 cells connected in a 4×4 mesh-connected pattern. The proposed algorithm was able to achieve the optimal solution (shown in Fig. 9(a)) in approximately 5 seconds. Figure 9(b) shows the experimental results for 100 simulation runs. The vertical axis indicates the number of the obtained solutions having wire lengths with the value specified by the horizontal axis. All the trials were started from different randomly generated initial configurations. The solution quality of the proposed solution model can be seen by comparison with the Hopfield-like placement algorithm described in [2]. The proposed optimization technique is almost independent of the initial configuration. There are 82% of the simulation runs achieved the optimal placement result and 96% achieved a near optimal solution. All the solutions obtained were valid. Compare with the solution model described in [2], on the other hand, 13% of the simulation runs achieved the optimal placement result and 27% achieved a near optimal solution. Moreover, 14% of the solutions were invalid after 200 iterations. The results show that the total wire length obtained is insensitive to differences in the initial configuration. In other words, the proposed algorithm successes can yield a proper placement when different initial configurations are used. Our method improves both the total wire length and module overlap, compared with previous approaches.

TABLE II The number of cells, number of wires, and size of placement region for input test circuits

Circuit	Number of Cells	Number of Wires	Region (μm^2)
m4 × 4	16	24	4 × 4
mblk-2	29	82	100 × 85
ALU	67	81	30 × 30
Ami33	33	121	2230 × 2360
Xerox	10	203	8750 × 7440
NCTU8	8	12	4 × 4
NCTU225	225	49964	42 × 42
NCTU1024	1024	1045256	120 × 120

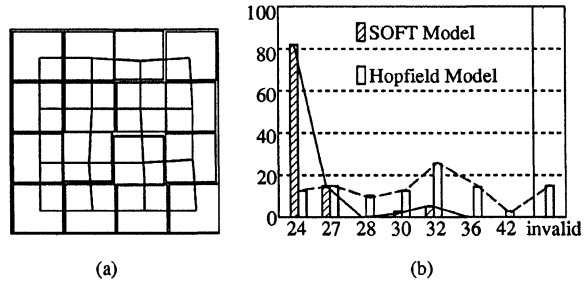

FIGURE 9 An example called $m4 \times 4$, with 4×4 mesh connected cells. (a) The optimal solution achieved by the proposed method. (b) The experimental results for 100 simulation runs. The vertical axis indicates the number of solution obtained having wire lengths with the value specified by the horizontal axis.

The second example consisting 29 cells and 82 nets was reproduced from Reference [18] and called mblk-2. Figure 10 presents the solution for mblk-2 within a 100×85 region. The total wire length obtained was 1035. The third example, called ALU, consisted of 67 cells and 81 nets. Each cell in this example is a 3×3 rectangle and has to be placed within a 30×30 region. The total wire length obtained from our system was 605 units. In addition, the placement result was free from overlap (see Fig. 11). Our result was better than that achieved by Kim's SOAP [1], which used

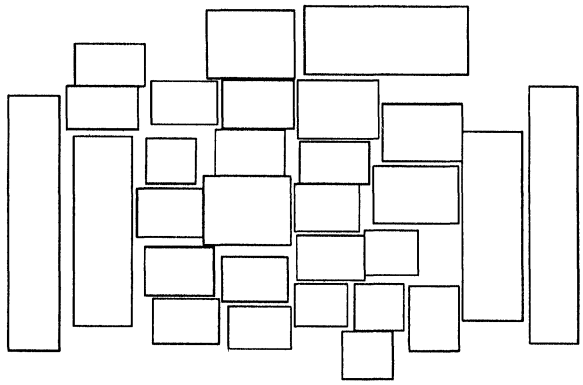

FIGURE 10 The solution for placing mblk-2 within a 100×85 region. The total wire length obtained was 1035.

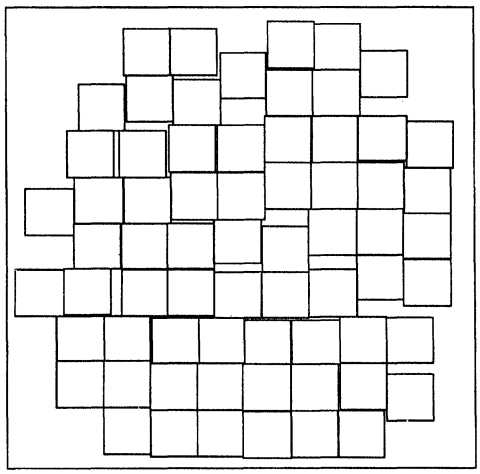

FIGURE 11 The placement solution for example circuit ALU. The total wire length obtained from our system was 605 units. In addition, the placement result is free from module overlap.

approximately 838 units and did not consider module overlap. The fourth example is an MCNC benchmark circuit called Ami33 that consists of 33 cells and 121 wire nets. The proposed method achieved a solution with shorter total wire length (60571 μm) than TimberWolfMC5.6 (74310 μm) [26] with half-perimeter estimation. The next example, which consists of 10 macro cells and 203 wire nets, is an MCNC benchmark circuit called Xerox. The achieved total wire length (562066 μm for half-perimeter estimation) was also shorter than that given by TimberWolfMC5.6 (603260 μm) [26]. The placement results for these two MCNC benchmark circuits are shown in Figure 12. and Figure 13. Our results for all these examples are free of module overlaps.

Two large size problems were also solved to demonstrate the performance of the proposed algorithm. The first large size example, called NCTU225, consists of 225 macro-cells and 49964 nets. The total wire length is compared with the results of the random cluster growth method, in which, unplaced cells are selected at random and placed around the pre-placed cells randomly. The total wire length obtained was 991823.75. For the random cluster growth method, the total wire

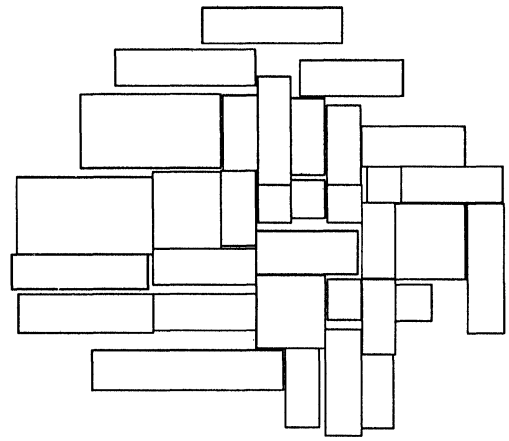

FIGURE 12 The Ami33 example of the MCNC benchmark circuit, in which, 33 cells are placed within a 2230×2360 region. The result obtained has a total wire length of 60571.

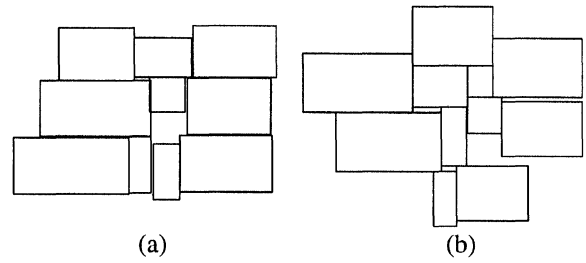

FIGURE 13 Solution for placing MCNC benchmark circuit Xerox within a 8750×7440 region. (a) With fuzzy gain function, the total wire length is 562066. (b) Without fuzzy gain function, the total wire length is 687564.

length obtained in the best experimental result within 100 simulation runs was 3870765.85. The second large size example, which consists of 1024 macro-cells and 1045256 nets, is called NCTU1024. The total wire length achieved was 40171352, and the placement region was 120×120. This result was also shorter than that of the random cluster growth method, which obtained 230555424.36 for the best experimental result within 100 simulation runs. These experiments show that the SOFT method can obtain feasible solutions for large size problems.

Table III presents a detailed comparison of the total wire length obtained by the other approaches, *i.e.*, SOAP (self-organizing assist placement) and TWMC 5.6 (TimberWolfMC5.6, a simulated annealing approach). We define a measurement of improvement in result as follows:

$$\text{Improvement} = \left(1 - \frac{\text{Result of SOFT}}{\text{Result of Other Method}}\right) \times 100\%$$

The results show that our algorithm provides an average of 17% improvement in total wire length is compared with the other approaches. A comparison of the results of the proposed self-organizing optimization technique with and without the use of the fuzzy gain function is shown in Table IV. It can be seen that the fuzzy gain function is very important in balancing the multiple conflicting objectives. The average improvement is over 11%. Note that the reduction in total wire length is only 0.4% when the fuzzy gain function is used in placing the test circuit ALU. The major reason for this insignificant improvement is that all the cells in ALU are small squares and the disturbance from module size and height/width ratio is not marked.

6. CONCLUSION

A novel self-organizing neural network has been presented to solve the problem of macro-cell placement. The proposed model uses the position weights of output neurons to represent the positions of cells. With an additional self-organization algorithm, the position weights evolve in a continuous manner toward the ultimate solution. This continuous representation is quite different from the one found in many traditional algorithms that use discrete-representation, *i.e.*, the Hopfield optimization network, where all the intermediate solutions examined are various permutations of cells. The experimental results for our method are quite encouraging. The proposed approach is competitive with previous state-of-the-art algorithms, no matter whether they are neural-network based or non-neural-network based. The proposed method provides 17% improvement in total wire length over the results of previous algorithms, and the results obtained are independent of the initial configuration. Our macro-cell placement method improves both the total wire length and module overlap, one of which has always been trade-off for the other in previous approaches. Our experiments also demonstrate that solutions obtained using the

TABLE III Total wire length for the test examples

	TOTAL WIRE LENGTH (μm)				
Example	[1]*	[2]*	TWMC5.6	Ours	Improvement
m4 × 4	–	30.10**	–	24.13**	20%
ALU	838	–	–	605	28%
Ami33	–	–	74310	60571	18%
Xerox	–	–	603260	562066	7%

–Not proposed or not available.
*Considers only the dot–module formulation.
**Average of the best 86% solutions in 100 simulation runs.

TABLE IV Comparisons of the proposed self-organizing optimization technique with and without fuzzy gain function

Example	TOTAL WIRE LENGTH (μm)			
	Initial	Without $g(t)$	With $g(t)$	Improvement
mblk-2	3369	1208	1035	14%
ALU	1720	608	605	0.4%
Ami33	177538	69767	60571	12%
Xerox	987547	687564	562066	18%

fuzzy optimization technique are better than those achieved without it.

Acknowledgments

This work was supported in part by the National Science Council, under number: NSC 83-0408-E009-020.

References

[1] Kim, S. S. and Kyung, C. M. (1992). "Circuit placement in arbitrarily-shaped regions using the self-organization principle", *IEEE Trans. CAD*, **11**(7), pp. 844–854.

[2] Kita, H. and Nishikawa, Y. (December 1992). "Solving a placement problem by means of an analog neural network", *IEEE Trans. Industrial Electronics*, **39**(6), pp. 543–551.

[3] Hemani, A. and Postula, A. (1990). "Cell placement by self-organisation", *Neural Networks*, pp. 377–383.

[4] Donath, W. E. (1980). "Complexity theory and design automation", *Proc. 17th ACM/IEEE Design Automat. Conf.*, ACM/IEEE, pp. 412–419.

[5] Shahookar, K. and Mazumder, P. (June 1991). "VLSI cell placement techniques", *ACM computing surveys*, **23**(2), pp. 143–220.

[6] Kohonen, T. (1989). Self-organization and associative memory, New York: Springer-Verlag Publishers, 3rd Ed.

[7] Odawara, G., Hamuro, T., Iijima, K., Yoshino, T. and Dai, Y. (1987). "A rule-based placement system for printed wiring boards", *Proc. 14th ACM/IEEE Design Automa. Conf.*, ACM/IEEE, pp. 777–785.

[8] Hopfield, J. J. and Tank, D. W. (1985). "Neural computation of decisions in optimization problems", *Biological Cybernetics*, pp. 141–152.

[9] Persky, G. (May 1987). "Experiments in cell placement with a simulated neural network", *Proc. IEEE Int. Workshop on Placement and Routing*, IEEE, pp. 10–13.

[10] Naft, J. (1989). "Neuropt: Neurocomputing for multi-objective design optimization for printed circuit board component placement", *Proc. IEEE/INNS Int. J. Conf. NN*, IEEE/INNS, pp. I/321–325.

[11] Sriram, M. and Kang, S. M. (1990). "A modified Hopfield network for two-dimensional module placement", *Proc. IEEE ISCS*, IEEE, pp. 1664–1667.

[12] Yu, M. L. "A study of the applicability of Hopfield decision neural nets to VLSI CAD", *Proc. 26th ACM/IEEE Design Automa. Conf.*, ACM/IEEE, pp. 412–417.

[13] Lippmann, R. P. (April 1987). "An introduction to computing with neural nets", *IEEE ASSP Mag.*

[14] Styblinski, M. A. and Tang, T. S. (1990). "Experiments in nonconvex optimization: stochastic approximation with function smoothing and simulated annealing", *Neural Networks*, pp. 467–483.

[15] Goto, S. (January 1981). "An efficient algorithm for the two dimensional placement problem in electrical circuit layout," *IEEE Trans. C&S.*, pp. 12–18.

[16] Lin, R. B. and Shragowitz, E. (1991). "Fuzzy logic approach to placement problem", *Proc. 29th ACM/IEEE Design Automat. Conf.*, IEEE/ACM, pp. 153–158.

[17] Pal, S. K. and Dutta Majumder, D. K. (1986). Fuzzy mathematical approach in pattern recognition, New York: John Wiley and Sins Publishers.

[18] Tsai, C. C., Chen, S. J., Hsiao, P. Y. and Feng, W. S. (1991). "New iterative construction approach to routing with compacted area", *IEE Proc. E.*, **138**(1), pp. 57–72.

[19] Aarts, E. H. L., de Bont, F. M. J., Korst, J. H. M. and Rongen, J. M. J. (1991). "An efficient macro-cell placement algorithm", *INTEGRATION, the VLSI journal*, pp. 299–317.

[20] Mir, M. and Imam, M. H. (1990). "A gradient-based method for module placement", *Computers and Elec. Engng.*, **16**(2), pp. 109–113.

[21] Kyung, C. M., Kraus, P. V. and Mlynski, D. A. (1991). "An analytic algorithm for global circuit placement", *INTEGRATION, the VLSI journal*, pp. 191–204.

[22] Kyung, C. M., Widder, J. and Mlynski, D. A. (January 1992). "Adaptive cluster growth: a new algorithm for circuit placement in rectilinear regions", *Computer-aided design*, **24**(1), pp. 27–35.

[23] Date, H., Seki, M. and Hayashi, T. (1990). "LSI module placement methods using neural computation networks", *Proc. IEEE Int. Conf. Neural Networks*, IEEE, pp. 831–836.

[24] Mir, M. and Iman, M. H. (1992). "Topology optimization of arbitrarily-size blocks using a bivariate formulation", *Computer-Aided Design*, **24**(10), pp. 556–564.

[25] Alon, A. and Ascher, U. (1988). "Model and solution strategy for placement of rectangular blocks in the Euclidean plane", *IEEE Trans. Computer-Aided Design*, **7**(2), pp. 378–386.

[26] Brasen, D. R. and Bushnell, M. L. (1992). "MHERTZ: a new optimization algorithm for floorplanning and global routing", *Int. Conf. VLSI Design*, pp. 594–597.

[27] Chang, R. I., Huang, K. Y. and Yen, H. T. (1990). "Self-organizing neural network for picking of seismic horizons", *Symp. of Telecommunications*, pp. 431–434.

[28] Huang, K. Y., Chang, R. I. and Yen, H. T. (1990). "Self-organizing neural network for picking of seismic horizons", *Int. Meeting of SEG.*, pp. 313–316.

Authors' Biographies

Ray-I Chang was born on February 7, 1967 in Taichung, Taiwan, Republic of China. He received the B.S., degrees and the Ph.D., degree in the

Department of Computer and Information Science, National Chaio Tung University, Hsinchu, Taiwan, ROC in 1989 and 1996. His main research interests include VLSI/CAD, neural networks, fuzzy set theory and image processing.

Dr. Hsiao was born on January 5, 1957 in Taiwan, Republic of China. He received the B.S., degree in Chemical Engineering from Tung Hai University in 1980, and the M.S., and Ph.D., degrees in Electrical Engineering from National Taiwan University in 1987 and 1990, respectively. Since August 1990, he has been an Associate Professor in the Department of Computer and Information Science at the National Chiao Tung University, Hsinchu, Taiwan, ROC. From November 1991, he also joins as a Technique Consultant in the Piping Engineering Department, CTCI Corporation, Taipei, Taiwan, ROC. Dr. Hsiao has been a Visiting Senior Fellow in National University of Singapore from July to September, 1993. He ranked 2nd in the 1985 Electronics Engineering Award Examination conducted by the ROC government for studying abroad, and awarded the 1990 Acer Long Term Ph.D., Dissertation Award. His main research interests are VLSI-CAD, Piping Engineering Design Automation, and Neural Network and Expert System Applications and Database System. Dr. Hsiao has published more than 40 articles in conferences and authoritative journals.