Electrical Engineering in Japan, Vol. 129, No. 2, 1999
Translated from Denki Gakkai Ronbunshi, Vol. 117-C, No. 10, October 1997, pp. 1479–1484

A Neural Network Parallel Algorithm for One-Dimensional Gate Assignment Problems

KAZUHIRO TSUCHIYA
Fujifacom Corporation, Japan
YOSHIYASU TAKEFUJI
Keio University, Japan
KEN-ICHI KUROTANI
Fujifacom Corporation, Japan

SUMMARY

A near-optimum parallel algorithm for solving the one-dimensional gate assignment problem is presented in this paper, where the problem is NP-hard and one of the most fundamental layout problems in VLSI design. The proposed system is composed of n × n processing elements based on the artificial two-dimensional maximum neural network for (n + 2)-gate assignment problems. Our algorithm has discovered improved solutions in the benchmark problems compared with the best existing algorithms. The proposed approach is applicable to other VLSI layout problems such as th PLA (Programmable Logic Array) folding problem. © 1999 Scripta Technica, Electr Eng Jpn, 129(2): 71–77, 1999

Key words: Neural network; one-dimensional gate assignment problem; parallel algorithm; Weinberger array; PLA folding problem.

1. Introduction

As a result of reducing development time, diversifying features, and increasing integration in recent integrated circuits, VLSI layout design has reached the limit of what can be done by people. Design support using computers, that is to say, CAD, has become essential for VLSI design due to this situation. In particular, the majority of optimum layout designs for VLSI involves problems of optimization for combinations that are NP-complete or NP-hard [1], because as the number of elements and number of wires increase, the number of combinations rises dramatically. Therefore, using CAD avoids this combinatorial explosion, and the introduction of a method to find optimum or near-optimum solutions efficiently is vital.

Hopfield and Tank [2] proposed an interconnecting neural network, the first to use sigmoid-type neurons, as a method to resolve the problem of optimizing combinations. Since then, many researchers have attempted to improve this method, and have applied their ideas to the design of integrated circuits. The authors have proposed a McCulloch–Pitts-type neural network with hysteresis characteristics and a one-dimensional optimum (winner-take-all) neural network. They have applied these networks to combinatorial optimization problems, and have reported positive results [3–5]. Recently, the authors have proposed a new neural network using two-dimensional neuron modeling, and have applied it to two-dimensional combinatorial optimization problems [6]. Now the authors are applying this model to a fundamental problem in optimum layout, that of one-dimensional gate assignment. They are proposing an operating equation approximation method. Having confirmed its efficacy, they are reporting their results. This method can also be used for other optimal layout problems in VLSI design, such as the PLA (Programmable Logic Array) folding problem.

2. One-Dimensional Gate Assignment Problem

The one-dimensional gate was proposed in 1967 by Weinberger [8], and is a logic gate also referred to as a Weinberger array. Figure 1 shows how this gate is represented, with (a), (b), and (c) illustrating the logic symbol of a NOR gate, its circuitry, and a diagram of the mask pattern, respectively. Note that line D is what would be connected to another gate. In the one-dimensional gate assignment problem, Fig. 1(c) is replaced with (d) for the purpose of

CCC0424-7760/99/020071-07
© 1999 Scripta Technica

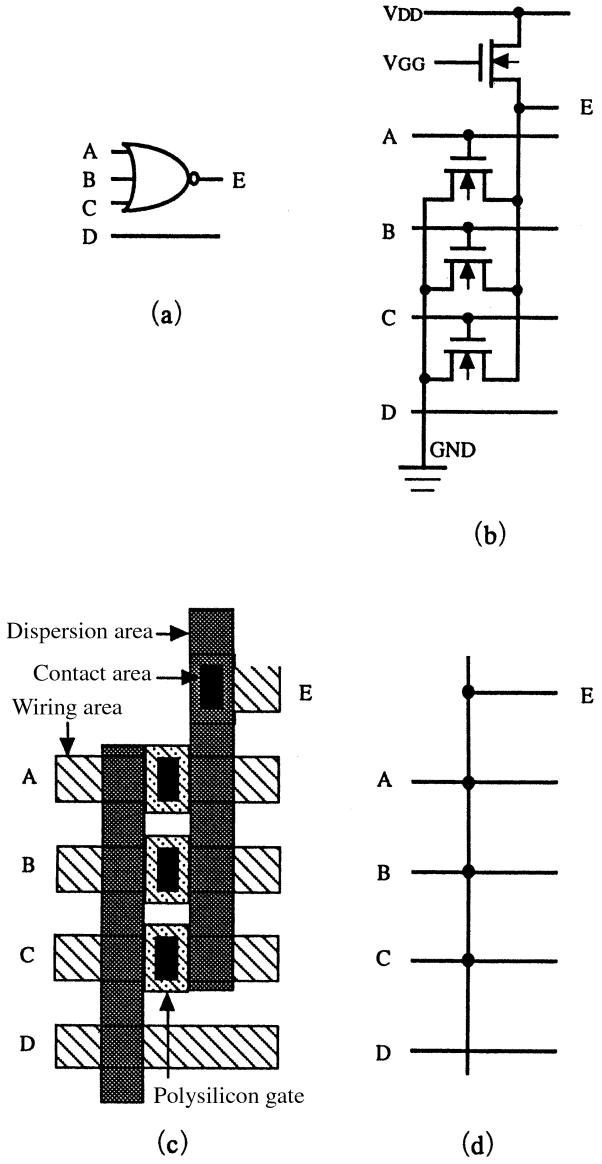


Fig. 1. An example of the representation by the one-dimensional gate assignment problem. (a) A logic symbol of a NOR gate; (b) the circuitry of (a); (c) a simplified mask pattern of (b); (d) the representation of (c) for the one-dimensional gate assignment problem.

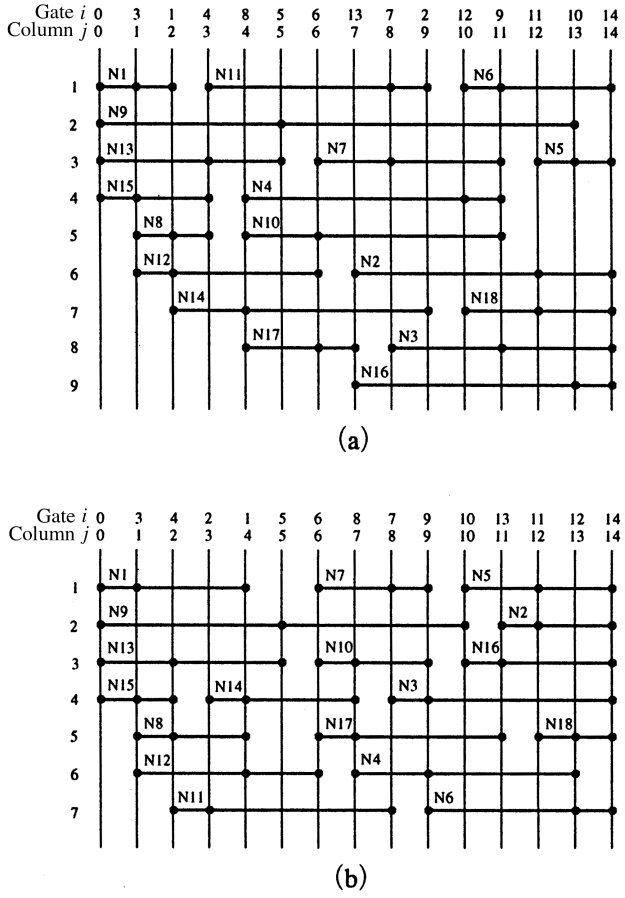


Fig. 2. An example of the one-dimensional gate assignment problem (net layout). (a) Before optimization; (b) after optimization.

simplification. The gate, metal wiring, and contacts are shown using vertical lines, horizontal lines, and solid circles respectively. In the one-dimensional gate assignment problem, several one-dimensional gates are handled, with the metal wire connecting the gates being called a net. Figure 2(a) shows an example of the net layout in the one-dimensional gate assignment problem. Here Ni refers to the i-th net. Note that the gates on the far left and the far right are positioned at the leftmost and rightmost columns (vertical arrays). For instance, gates 0 and 4 in Fig. 2(a) are assigned to columns 0 and 14, respectively. Moreover, the net is laid out flat on tracks (horizontal lines). The following limiting conditions must also be satisfied in the one-dimensional gate assignment problem.

(1) The n-th gate must be laid out for all columns n by assigning one different gate per column.

(2) All gates must be in contact with each other.

The purpose of the one-dimensional gate assignment problem is to satisfy the above limiting conditions and to minimize the number of tracks required. As an example, the result of optimizing the assignment of the 13 gates shown in Fig. 2(a) is presented in Fig. 2(b). The minimum number of tracks in this instance is seven, with an area which is under 80% compared to Fig. 2(a). This demonstrates that the one-dimensional gate assignment problem is NP-hard [9].

There are fundamentally two operations necessary in the one-dimensional gate assignment problem. The first is

the assignment of columns for gates, and the second is the layout of net tracks. In past proposed methods to solve the one-dimensional gate assignment problem [10–17], there have been methods which solved these two operations at the same time and methods which emphasized the gate assignment method and used the left-edge-first algorithm [10] for net layout. Here, the left-edge-first algorithm is a method which lays out the net with the minimum number of tracks for the given gate layout. Note that the left-edge-first algorithm satisfies the limiting condition (2) above. In this paper, the input for each neuron is determined using the left-edge-first algorithm, and then the gate assignment is determined using the output of each neuron.

3. Methods of Expression Using Neural Networks

Mathematical models of artificial neural networks are composed of two elements, referred to as neurons and synapses. The output signal of a neuron passes through a synapse junction, and then is transmitted as an input to another neuron. If the neuron input is U and the output is V, the input/output relationship for the i, j-th neuron is expressed as $V_{i,j} = f(U_{i,j})$. Here, f is the neuron input/output function. Hopfield and Tank used a sigmoid function which could be integrated. However, the synapse junction is generally expressed using an operating equation which represents the junction state between the i,j-th neuron shown in Eq. (1) and another neuron:

$$\frac{dU_{i,j}}{dt} = -\frac{\partial E(V_{1,1},\ldots, V_{i,j},\ldots, V_{n,n})}{\partial V_{i,j}} \tag{1}$$

As will be described later, the energy function on the right of Eq. (1), E expresses the cost, a necessary and sufficient condition (constraint) for the problem. In this paper, Eq. (1) and the primary Euler method are used, and the input for each neuron is updated using the following equation:

$$U_{i,j}(t+1) = U_{i,j}(t) + \Delta U_{i,j} \tag{2}$$

Here, $\Delta U_{i,j} = dU_{i,j}/dt$. In addition, an n × n neural network array is used for the (n + 2) element one-dimensional gate assignment problem. For instance, the results in Fig. 2(b) are passed through the 13 × 13 neural network array shown in Fig. 3. Here, each square shows the output of the i,j-th neuron. The black and white squares represent a neuron output, which is 1 or 0, respectively. If the output of the i,j-th neuron is 1, this indicates that gate i is assigned to column j. Therefore, in order to satisfy limiting condition (1) in the one-dimensional gate assignment problem, the output of one neuron for each row and column in the neural network array must be 1. Based on the above limiting condition, E from Eq. (1) can be represented by the following equation:

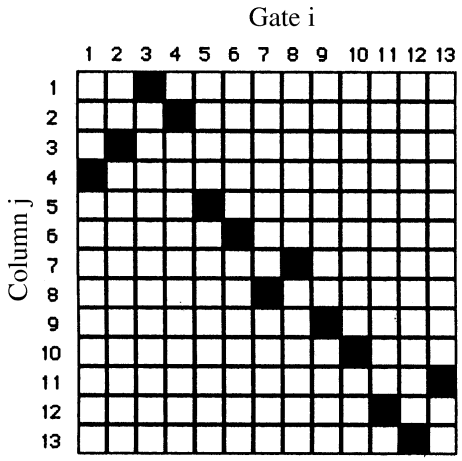


Fig. 3. The 13 × 13 neural network array for Fig. 2(b).

$$2E = A\sum_{i=1}^{n}\left(\sum_{j=1}^{n} V_{i,j} - 1\right)^2 + B\sum_{j=1}^{n}\left(\sum_{i=1}^{n} V_{i,j} - 1\right)^2 + C \times R(V_{1,1}, \ldots, V_{i,j}, \ldots, V_{n,n})) \tag{3}$$

Here, A, B, and C are parameters; R is the track number determined by the left-edge-first algorithm.

The first and second items on the right-hand side of Eq. (3) are 0 when the above limiting conditions are fulfilled. Moreover, the third item on the right-hand side of Eq. (3) expresses the cost of the given problem. Here, the value is smaller as the track number R decreases. In other words, E reaches a minimum when the limiting conditions are satisfied and R is at a minimum.

Therefore, the operating equation is given by the following equation, based on Eqs. (1) and (3):

$$\frac{dU_{i,j}}{dt} = -A\left(\sum_{i=1}^{n} V_{i,j} - 1\right) - B\left(\sum_{j=1}^{n} V_{i,j} - 1\right) - C\frac{\partial R(V_{1,1},\ldots, V_{i,j},\ldots, V_{n,n})}{\partial V_{i,j}} \tag{4}$$

However, it is known that the following problems exist for the two-dimensional format operating equation [18, 19]:

(1) The first and second elements on the right are not always limited to 0 [limiting condition (1) is not always satisfied].

(2) Therefore, the track number R determined by the left-edge-first algorithm cannot be determined.

(3) Furthermore, because each parameter value is unknown, it must be determined empirically.

The method using a sigmoid function for the input/output function for the neurons has the following problems [18, 19]:

(4) The threshold value, and maximum and minimum values for input are also determined empirically.

(5) Calculations for the sigmoid function are time-consuming.

Therefore, in this paper, a two-dimensional optimum neuron model, shown in Eq. (5), is used to resolve the above problems.

$$\begin{aligned}
&\text{Step 1. } V_{a,b} = 1 \text{ if } U_{a,b} = \max\{U_{i,j}\} \\
&\text{Step 2. } V_{c,d} = 1 \text{ if } U_{c,d} = \max\{U_{i,j} \mid i \neq a,\ j \neq b\} \\
&\text{Step 3. } V_{e,f} = 1 \text{ if } U_{e,f} = \max\{U_{i,j} \mid i \neq a, c,\ j \neq b, d\} \\
&\quad . \qquad . \qquad . \\
&\quad . \qquad . \qquad . \\
&\text{Step n. } V_{g,h} = 1 \text{ if } U_{g,h} = \max\{U_{i,j} \mid i \neq a, c, e, \ldots, \\
&\qquad j \neq b, d, f, \ldots\} \\
&V_{k,l} = 0 \text{ otherwise.}
\end{aligned} \tag{5}$$

In Eq. (5), first the output of the a,b-th neuron, which has a maximum input, is set to 1. Then, from among the neurons excluding the a,*-th and the *,b-th neuron, the output of the c,d-th neuron, which has a maximum input, is set to 1. Here, * represents all integers from 1 to n. In the same fashion, the e,f-th neuron having the highest input from among the neurons except for the a,*-th neuron, the *,b-th neuron, the c,*-th neuron, and the *,d-th neuron, has its output set to 1. This is repeated until the output of the n-th neuron is set to 1. In other words, the output of the $(n^2 - n)$-th neuron becomes 0. Limiting condition (1) is satisfied by virtue of the input/output relationship for this neuron. The operating equation (4) can be simplified as follows:

$$\frac{dU_{i,j}}{dt} = -C \frac{\partial R(V_{1,1}, \ldots, V_{i,j}, \ldots, V_{n,n})}{\partial V_{i,j}} \tag{4'}$$

Nevertheless, the partial differential of $R(V_{1,1}, \ldots, Vi,j, \ldots, V_{n,n})$ with respect to V_{ij} in Eq. (4′) is impossible to obtain. In the past, a method to solve problems for energy functions with elements which cannot be differentiated using neural networks was thought to be inappropriate. In this paper, the right part of Eq. (4′) or of Eq. (1) represents the penalty for the i,j-th neuron. Attention is given to revising the input using Eq. (2), depending on the size of the penalty, and Eq. (4′) approaches the following equation

$$\frac{dU_{i,j}}{dt} = C\ (O - R(V_{1,1}, \ldots, V_{i,j}, \ldots, V_{n,n})) \tag{6}$$

(in other words, as R increases, the penalty becomes larger):

O: target track number, where O < R. Here O is used in order to determine the end (O > R) when doing iterative calculations. Also, for the k,l-th neuron for $V_{k,l} = 0$, $V_{k,l}$ is set to 1 temporarily, and the output of the k,x-th neuron for $V_{k,x} = 1$ is set to 0 in order to obtain R in Eq. (6). Moreover, the output of the y,1-th neuron for $V_{y,1} = 1$ is set to 0, then, with $V_{y,x} = 1$, the track number R is found using the left-edge-first algorithm. For instance, in order to find $dU_{1,1}/dt$ as shown in Fig. 4(a), $V_{1,1}$ is set to 1 and $V_{1,2}$ to 0 temporarily. Then $V_{3,1}$ is set to 0 and $V_{3,2}$ is set to 1, after which R is found using the left-edge-first algorithm. As illustrated in Fig. 4(b), this is the same as temporarily switching gates 1 and 3. At this time, if R = 9 and O = 7, $dU_{1,1}/dt = -2$. A heavy penalty is levied against the neuron whose gate assignment has an R which increases in Eq. (6), and its input becomes smaller.

This method is similar to SA (simulated annealing), a method which improves consecutively by temporarily

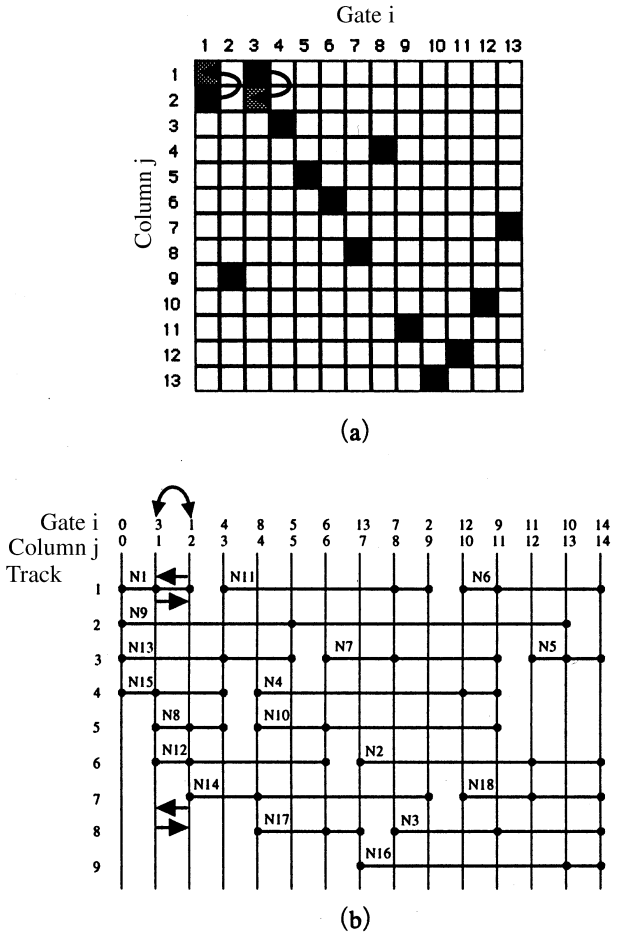


Fig. 4. The explanation for the calculation of $dU_{1,1}/dt$. (a) The neural network array representation to calculate $dU_{1,1}/dt$; (b) the net layout representation to calculate $dU_{1,1}/dt$.

switching the gates, and to GA (genetic algorithms) and other related search methods. However, in contrast to the direct overall evaluation of R in SA and GA, this method is significantly different in that R is used to evaluate the amount of change in a single neuron's input. In addition, Eq. (6), used as a method to avoid convergence toward a local solution, as is also the case in SA and GA, is improved in this paper:

$$\frac{dU_{i,j}}{dt} = C\ (O - R(V_{1,1}, \ldots, V_{i,j}, \ldots, V_{n,n}))\ \ V_{i,j}$$

else

$$\frac{dU_{i,j}}{dt} = C\ (O - R(V_{1,1}, \ldots, V_{i,j}, \ldots, V_{n,n})) \quad (6')$$

Here, t is the number of iterative calculations, a mod b is the residue of a modulo b, and w is a parameter.

Because a heavy penalty tends to be levied on a neuron whose output is 0 when Eq. (6) is used, these equations are designed so that the parameter w and the first equation in Eq. (6′) are used, and a penalty is levied only against the neuron whose output is 0. The parameter w is the only parameter in this method, and it allows for improved concentration on optimal solutions. Note that w = 4 was used for all of the experiments described below. Also, because the first equation in Eq. (6′) is 0 for the neuron whose output is 0, there is no need to do calculations beforehand in this method. This has the advantage of accelerating the calculations.

4. Parallel Calculation Algorithms and Their Results

The parallel algorithms for the one-dimensional gate assignment problem using the two-dimensional optimum neuron model proposed in this paper are as follows. Note that t_limit represents the maximum number of iterative calculations, which determines the final conditions for the calculations.

Step 1: t = 0 and C = 1 are the initial conditions, with the values for O and t_limit being determined as appropriate.

Step 2: Initial values for all $U_{i,j}$ (t) (i, j = 1, . . . , n) are determined using random numbers.

Step 3: The values for all $V_{i,j}$ (t) (i, j = 1, . . . , n) are obtained using Eq. (5).

Step 4: The values for ΔU_{ij} (t) (i, j = 1, . . . , n) are found for each neuron using Eq. (6′). Here ΔU_{ij} (t) = dU_{ij} (t)/dt is used.

Step 5: When ΔU_{ij} (t) ≥ 0, the solution is recorded, and O = O – 1.

Step 6: The values for U_{ij} (t + 1) (i, j = 1, . . . , n) are found for all neurons using Eq. (2).

Step 7: If t = t_limit, the calculations are terminated. Otherwise, t = t + 1, and the process returns to Step 3.

In this algorithm, all input is updated simultaneously in Steps 4 and 6, using a synchronous parallel calculation method. Therefore, these calculations can be performed on a parallel computer. However, the results for calculations performed on an HP9000/710 are shown here.

Table 1 shows the results of evaluating this algorithm using five benchmark problems. For purposes of comparison, the table also shows the results obtained using a different algorithm. It can be seen that the number of tracks obtained using this algorithm is the same or smaller than what was found using the past method. In particular, in problems 4 and 5, a track number over 15% smaller was obtained. Figure 5 is a net layout diagram obtained for problem 4 using this algorithm.

Table 1. The problems and the results

Problem # (Reference)	Total number of gates	Total number of nets	Track number	
			Author's method	Reference
1 (14)	9	8	4	4
2 (13)	15	18	7	7
3 (13)	29	37	13	13
4 (13)	48	48	11	13
5 (15)	85	96	20	23

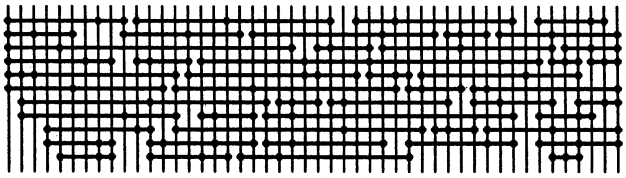


Fig. 5. One of the solutions for problem 4.

5. Conclusion

In this paper, the authors proposed a parallel algorithm using a neural network which uses a two-dimensional optimum neuron model as a method to solve the one-dimen-

sional gate assignment problem, a fundamental problem in VLSI design. In this method, an n × n neural network array is used for the one-dimensional gate assignment problem with (n + 2) elements. By satisfying the limiting conditions for the problem using a neuron model, the problem of operating equations in quadratic form is addressed. In addition, the authors proposed an approximation method for operating equations in cases in which the energy function cannot be differentiated. Using computer simulations, the results showed that more near-optimal solutions could be found as compared to previously proposed algorithms. Moreover, this method has the advantage of not requiring the many parameters needed in other representative optimization methods such as SA and GA, and not requiring the "good initial values" needed by many heuristic algorithms. Given the above, the optimization method using two-dimensional optimum neuron models and parallel algorithms has been confirmed as effective in the layout of one-dimensional arrays in VLSIs.

However, using the two-dimensional optimum neuron model has the disadvantage of not being able to update output in parallel. The authors plan to improve their method, including this point, and adapt the algorithm for parallel computers. In addition to confirming the efficacy of this method in problems with larger gate numbers and net numbers, the authors plan to apply this method to the problem of optimum assignment in two-dimensional arrays [20, 21].

REFERENCES

1. Garey MR, Johnson DS. Computers and intractability: A guide to the theory of NP-completeness. Freeman; 1979.
2. Hopfield J, Tank D. Neural computation of decisions in optimization problems. Biol Cybern 1985;52:141–152.
3. Takefuji Y. Neural network parallel computing. Kluwer Academic Publishers; 1992.
4. Lee KC, Funabiki N, Takefuji Y. A parallel improvement algorithm for bipartite subgraph problem. IEEE Trans Neural Networks 1992;3:139–145.
5. Takefuji Y, Lee KC. A near-optimum parallel planarization algorithm. Science 1989;245:1221–1223.
6. Tsuchiya K, Bharitkar S, Takefuji Y. A neural network approach to facility layout problems. Eur J Oper Res 1996;89:556–563.
7. Tsuchiya K, Takefuji Y. A neural network approach to PLA folding problems. IEEE Trans CAD/ICS 1996;15.
8. Weinberger A. Large scale integration of MOS complex logic: A layout method. IEEE J Solid-State Circuits 1967;SC-2:182–190.
9. Kashiwabara T, Fujishima T. NP-completeness of the problem of finding a minimal-clique-number interval graph containing a given graph as a subgraph. Proc 1979 Int Symp Circuit Syst 1979:657–660.
10. Hashimoto A, Stevens J. Wire routing by optimizing channel assignment within large apertures. Proc 18th Design Automation Workshop 1971:155–169.
11. Ohtsuki T, Stevens J. One-dimensional logic gate assignment and interval graphs. IEEE Trans Circuits Syst 1979;CAS-26:675–684.
12. Wing O, Huang S, Wang R. Gate matrix layout. IEEE Trans Computer-Aided Design 1985;CAD-4:220–231.
13. Hong Y-S, Park K-H, Kim M. A heuristic algorithm for ordering the columns in one-dimensional logic arrays. IEEE Trans Computer Aided Design 1989;8:547–562.
14. Fujii T, Horikawa H, Kikuno T, Yoshida N. A heuristic algorithm for gate assignment in one-dimensional array approach. IEEE Trans Computer-Aided Design 1987;CAD-6:159–164.
15. Yamada S, Okude H, Kasai T. A hierarchical algorithm for one-dimensional gate assignment based on contraction of nets. IEEE Trans Computer-Aided Design 1989;8:622–629.
16. Asano T, Tanaka K. A gate placement algorithm for one-dimensional arrays. J Inf Process 1978;1:47–52.
17. Asano T. An optimum gate placement algorithm for MOS one-dimensional arrays. J Digital Syst 1982;IV:1–27.
18. Postma EO. Optimization networks. Lect Notes Comput Sci 1995;931:145–156.
19. Peterson C. Parallel distributed approaches to combinatorial optimization: Benchmark studies on traveling salesman problem. Neural Computation 1990;2:261–269.
20. Singh U, Chen CYR. From logic to symbolic layout for gate matrix. IEEE Trans Computer-Aided Design 1992;11:216–227.

APPENDIX

Left-Edge-First Algorithm

The algorithm below generates net layouts so as to minimize the number of tracks, without exchanging the nets when creating gate assignments [10].

Step 0: The procedure below is repeated until all nets are assigned, starting from the first track.

Step 1: Moving from the leftmost column to the right one column at a time, the net for which that column has

been assigned to the leftmost gate, excluding nets which have already been laid out, is searched for.

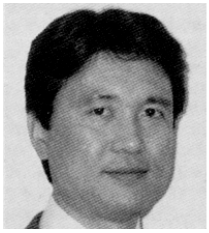
Step 2: The net is laid out.

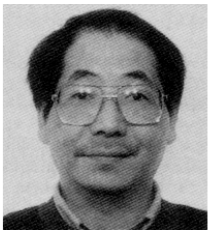
Step 3: Moving from the column, one column at a time, to the right of the column to which the rightmost gate for the net laid out in Step 2 has been assigned, the net for which that column has been assigned to the leftmost gate, excluding nets which have already been laid out, is searched for.

Step 4: Repeat Steps 2 and 3 until the rightmost column is reached.

Step 5: Move to the next track, then repeat from Step 1.

AUTHORS (from left to right)





Kazuhiro Tsuchiya (member) completed the master's course at Tohoku University in 1985. He then joined Fuji Electric Co., Ltd. (General Research Laboratory). From 1991 to 1994, he was a researcher at Case Western University in the United States. He joined Fujifacom Corporation in 1994. He then moved to the SFC Laboratory at Keio University as a researcher. Having worked on semiconductor integrated circuits with high-voltage resistance, he is now pursuing research and development on neural networks. He is a member of IEEE and INNS.

Yoshiyasu Takefuji (nonmember) has been a professor at Keio University since 1992. He was on the faculties of Case Western Reserve University (1988–1992), the University of South Carolina (1985–1988), and the University of South Florida (1983–1985). His research interests focus on neural computing and hyperspectrum imaging. He has received the following awards: NSF/RIA in 1989, distinguished award from *IEEE Trans. on Neural Networks* in 1992, IPSJ's best paper award in 1980, the TEPCO research award in 1993–1995, the KAST award in 1993–1995, and the Takayanagi research award in 1995.

Kenichi Kurotani (nonmember) completed the master's course at Tokyo Institute of Technology in 1975 and then joined Fuji Electric Co., Ltd. He moved to Fujifacom Corporation in 1995. He is pursuing research on the analysis of above and underground water supply, electric power, industrial facilities, and other processes as well as applied development of control logic. He is a member of the Japan Machine Council and the Instrumentation Automatic Control Association.