# A Near-Optimal Parallel Algorithm for One-Dimensional Gate Assignment in VLSI Layout

K. TSUCHIYA,[a,b] Y. TAKEFUJI[b,c]

[a] *Fuji Electric Co., Ltd., 1, Fuji-machi, Hino, Tokyo 191, Japan*
[b] *Facultyof Environmental Information, Keio University, 5322 Endoh, Fujisawa 252, Japan*
[c] *Department of Electrical Engineering and Applied Physics, Case Western Reserve University, Cleveland, OH 44106, USA*

**ABSTRACT:** A novel approach to 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 $n \times n$ processing elements called the artificial two-dimensional maximum neurons for $(n+2)$-gate assignment problems. We have discovered the improved solutions in the benchmark problems over the best existing algorithms. The proposed parallel algorithm is also applicable to other VLSI layout problems.

## INTRODUCTION

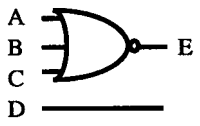
A one-dimensional gate is a logic gate and has been also called a Weinberger array since it was introduced by Weinberger in 1967 (Weinberger, 1967). Figure 1 shows an example of a representation of a one-dimensional gate. Figure 1 (a), (b), and (c) show a logic symbol of a NOR gate, a circuitry of the NOR gate, and a simplified mask pattern of the circuitry respectively. Note that line D is connected to other gates as well as the other lines. Figure 1 (c) is simply replaced by Figure 1 (d) in the one-dimensional gate assignment problem where the gate, metal lines, and contacts are replaced by a vertical line, horizontal lines, and black points respectively (Ohtsuki, 1979). If there is more than one logic gate, the metal lines of the gates formulate inter-gate connections, nets. Figure 2 (a) shows an example of a net layout of a one-dimensional gate assignment problem where N$i$ means the $i$-th net. Note that the leftmost and rightmost gate must be assigned to the leftmost and rightmost column respectively in the one-dimensional gate assignment problem. For example, gate #0 and gate #14 in Figure 2 (a) must be assigned to column #0 and column #14 respectively. The following two conditions must be also satisfied in the one-dimensional gate assignment problem:

1. A one and only one gate must be assigned to one column, and n gates must be assigned to n columns.
2. The nets must be located in the tracks (rows) horizontally and must not overlap each other.

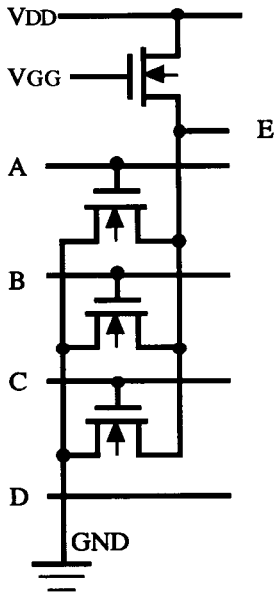
The goal of the one-dimensional gate assignment problem is to assign the gates to the columns in order to minimize the number of tracks with the above conditions satisfied. For example, after optimizing the permutation of 13 gates in Figure 2 (a), the minimum (optimal) number of tracks is 7, as shown in Figure 2 (b).

The One-dimensional gate assignment problem has proved to be an NP-hard problem (Kashiwabara, 1979) and is one of the most fundamental and important problems in VLSI layout design because the number of tracks influences the chip cost directly. Several algorithms have been proposed to solve the problem. The problem basically requires two tasks to find the minimum number of tracks: one is gate-ordering and the other is net-allocation. Some existing algorithms focus

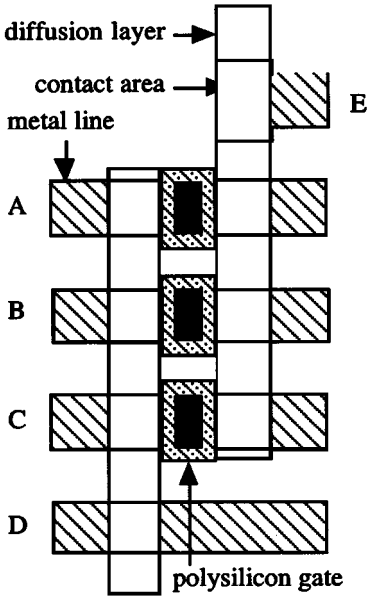
*Correspondence to:* Mr. Kazuhiro Tsuchiya.
Integrated Computer-Aided Engineering, 6(3) 249–257 (1999)



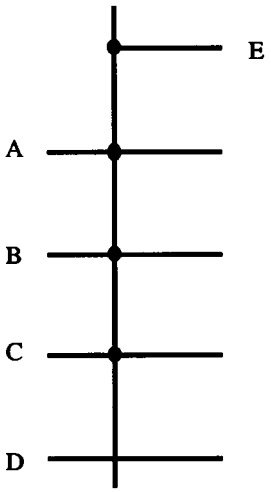

(a) A logic symbol of a NOR gate.




(b) A circuitry of (a).



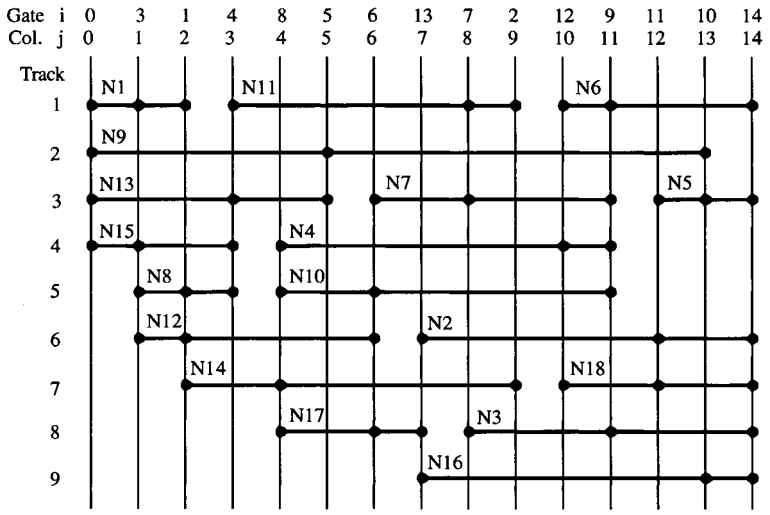

(c) A simplified mask pattern of (b).



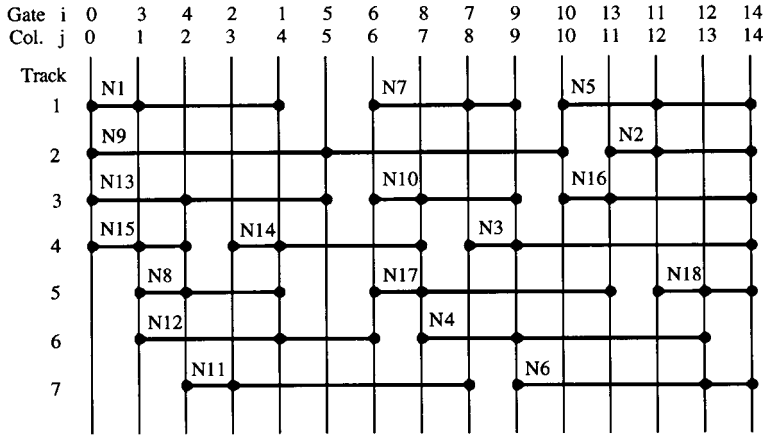

(d) A representation of (c) by the one-dimensional gate assignment problem.

**Figure 1** An example of a representation by the one-dimensional gate assignment problem.

on finding a better gate-ordering and then apply a left-edge-first algorithm (Hashimoto, 1971) which can pack the nets with the smallest number of tracks within a given gate-ordering. The left-edge-first algorithm is shown in Appendix 1. The number of possible solutions is $O(n!)$ where $n+2$ is the total number of gates. Ohtsuki et al. converted the problem into a minimum clique number augmentation problem of an interval graph as one of graph theory problems with the time complexity of $O(n(m+l))$ where $m$ and $l$ is the total number of nets and the augmentation respectively (Ohtsuki, 1979). Wing et al. also utilized the interval graph representation (Wing, 1985). Hong et al. converted the nets into a weighted graph and used a heuristic algorithm with the time complexity of $O(n^2)$ (Hong, 1989). Some algorithms combine the gate-ordering and the net-allocation without using the left-edge-first algorithm (Fujii, 1987; Yamada, 1989; Asano, 1982). Fujii et al. transformed the problem into a restricted one and used a heuristic algorithm based on a bidirec-




(a) Before optimization.




(b) After optimization.

**Figure 2** An example of a one-dimensional gate assignment problem.

tional search method (Fujii, 1987). Yamada et al. proposed a better heuristic algorithm based on a hierarchical algorithm with the time complexity of $O(i_q np)$ for $1 \leq i_q \leq p-1$ where $p$ is the total number of connections between the nets and gates (Yamada, 1989). Some algorithms need the good initial gate-orderings. To our knowledge, no parallel algorithm for this problem has been proposed. In this paper, a near-optimal parallel algorithm using a two-dimensional maximum neural network is proposed for the one-dimensional gate assignment problem, where the state of the neurons represents a gate-ordering and the left-edge-first algorithm is used to evaluate every neuron.

The first artificial neural network using sigmoidal neurons was introduced by Hopfield and Tank for solving combinatorial optimization problems (Hopfield, 1985). Takefuji et al. have proposed a hysteresis McCulloch-Pitts neural network and a one-dimensional maximum (winner-take-all) neural network for NP-complete problems (Takefuji, 1992). In the next section, we review the basic concept of the artificial neural networks and explain our neural network representation used to solve the problem, then we describe the neural network parallel algorithm and discuss the experimental results where several benchmark problems are used to justify the effectiveness of our algorithm.

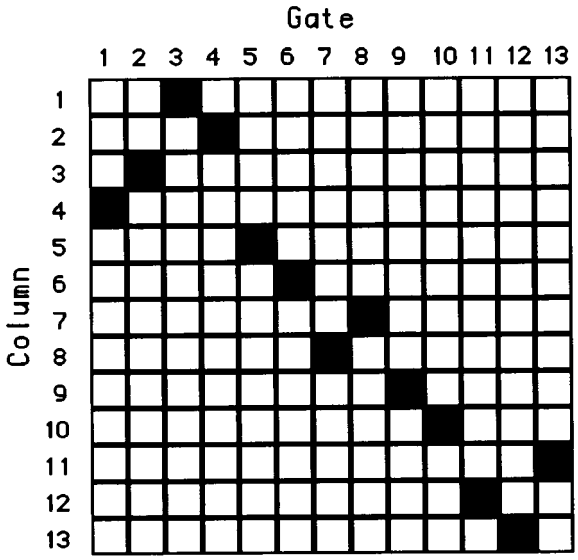



**Figure 3** A $13 \times 13$ neural network array for Figure 2 (b).

In the last section, we summarize this paper and comment on the future work on other VLSI layout problems.

## THE NEURAL NETWORK REPRESENTATION

The mathematical model of the artificial neural network consists of two components; neurons and synaptic links. The output signal transmitted from a neuron propagates to other neurons through the synaptic links. Every artificial neuron has the input $U$ and the output $V$. The output of the $(i,j)$-th neuron is given by $V_{i,j} = f(U_{i,j})$ where $f$ is called the neuron's input/output function.

Our system is composed of an $n \times n$ neural network array for an $(n+2)$-gate assignment problem. The solution in Figure 2 (b) is provided by the state of a $13 \times 13$ neural network array as shown in Figure 3. Note that each square represents an output state of the $(i,j)$-th neuron. The black squares and the white squares show that the outputs of the neurons generate 1's and 0's respectively. The nonzero output of the $(i,j)$-th neuron means that gate $\#i$ is assigned to column $\#j$. Because of the one-dimensional gate assignment constraint, that is, because one and only one gate must be assigned to each column, one and only one neuron must generate a nonzero output per row and per column in the $n \times n$ neural network array. To satisfy this constraint, the two-dimensional maximum neuron model is newly introduced. The input/output function of the neuron model is given by:

**Step 1.**

$V_{a,b} = 1$ if $U_{a,b} = \max\{U_{i,j}\}$;

**Step 2.**

$V_{c,d} = 1$ if $U_{c,d} = \max\{U_{i,j} \mid i \neq a,\ j \neq b\}$;

**Step 3.**

$V_{c,d} = 1$ if $U_{e,f} = \max\{U_{i,j} \mid i \neq a, c,\ j \neq b, d\}$;

$\vdots$

**Step n.**

$$V_{g,h} = 1 \text{ if } U_{g,h} = \max\{U_{i,j} \mid i \neq a, c, e, \ldots;$$
$$j \neq b, d, f, \ldots\};$$
$$V_{k,l} = 0 \text{ otherwise,} \qquad (1)$$

where $V_{i,j} = 1$ means that gate $\#i$ is embedded in column $\#j$. For example, gate $\#a$ should be assigned to column $\#b$ if $U_{a,b}$ is the largest among all $U_{i,j}$'s, and then gate $\#c$ should be assigned to column #d if $U_{c,d}$ is the largest among all $U_{i,j}$'s except for gate $\#i = \#a$ or column #j=#b, and so on. Note that if more than one neuron has the largest input, one neuron among them should be selected. In $n^2$ maximum neurons, $n$ neurons always generate nonzero outputs and the other $(n^2 - n)$ neurons generate zero so that not more than one gate is assigned per column. This neuron model provides a faster convergence speed and higher convergence rate than those of the conventional McCulloch-Pitts neuron or sigmoidal neuron models. The input of the $(i,j)$-th neuron is determined by the following motion equation.

The motion equation represents the synaptic

links. It shows interconnections between the $(i, j)$-th neuron and other neurons. The general motion equation of the $(i, j)$-th neuron is given by:

$$\frac{dU_{i,j}}{dt} = -\frac{\partial E(V_{1,1}, \ldots, V_{i,j}, \ldots, V_{n,n})}{\partial V_{i,j}}. \quad (2)$$

This means that the change of the input of the $(i, j)$-th neuron is given by the partial derivatives of the computational energy function $E$ with respect to the output of the $(i, j)$-th neuron where $E$ follows an $(n \times n)$-variable function: $E(V_{1,1}, \ldots, V_{i,j}, \ldots, V_{n,n})$. The motion equation is used to update $U_{i,j}$. The goal of the artificial neural network for solving optimization problems is to minimize the fabricated computational energy function $E$ in Eq. (2). The artificial neural network also provides a parallel gradient descent method to minimize the fabricated energy function $E$.

Usually the right term in Eq. (2) can be constructed by considering the necessary and sufficient constraints and/or the cost function from the given problem. The proposed two-dimensional maximum neural network needs only the cost function which is evaluated by the left-edge-first algorithm. The motion equation for the one-dimensional gate assignment problem is assumed to be given by:

$$\frac{dU_{i,j}}{dt} = O - R, \quad (3)$$

where $O$ and $R$ are the objective and the real number of tracks respectively. $O$ is the expected number of tracks and set to be always less than $R$. $R$ is given by the left-edge-first algorithm under the assumption that gate #i is allocated to column #j. For example, to calculate $dU_{i,j}/dt$ of the (1,1)-th neuron in Figure 2 (a), gate #1 is assigned to column #1 temporarily as shown in Figure 4 while the permutation of all the gates except gate #1 is the same, and then the left-edge-first algorithm is applied. According to the left-edge-first algorithm, the real number of tracks R is 9. If $O = 7$, then $dU_{1,1}/dt = 7 - 9 = -2$. To calculate $dU_{1,2}/dt$, the assignment of gate #1 to column #2 must be maintained because gate #1 has already been assigned to column #2. Since R is 9 as shown in Figure 4 (a), $dU_{1,2}/dt = -2$. Eq. (2) describes the degree of penalty which discourages the highly penalized neurons from generating a nonzero output.

## PARALLEL ALGORITHM AND EXPERIMENTAL RESULTS

A simulator based on the proposed neural network was developed and a synchronous parallel system was simulated where all the outputs were evaluated after all the inputs were updated. The following procedure based on the first-order Euler method was used. Note that $t_{limit}$ is the maximum number of iterations for the termination condition.

**Step 0.** Set $t = 0$, and set $O$ and $t_{limit}$ adequately.

**Step 1.** Assign uniformly randomized numbers to the initial values of $U_{i,j}(t)$ for $i, j = 1, \ldots, n$.

**Step 2.** Evaluate $V_{i,j}(t)$ for $i, j = 1, \ldots, n$, using Eq. (1).

**Step 3.** Compute Eq. (3) of the $n \times n$ neural network for $i, j = 1, \ldots, n$ based on the left-edge-first algorithm to obtain $\Delta U_{i,j}(t)$:

$$\Delta U_{i,j}(t) = \frac{dU_{i,j}}{dt}. \quad (4)$$

**Step 4.** If $V_{i,j}(t) = 1$ and $\Delta U_{i,j}(t) \geq 0$ for $i, j = 1, \ldots, n$, then record the answer for $O$ tracks, set $O = O - 1$, and set a new $t_{limit}$ bigger than the previous $t_{limit}$.

**Step 5.** Update $U_{i,j}(t + 1)$ for $i, j = 1, \ldots, n$, based on the first-order Euler method:

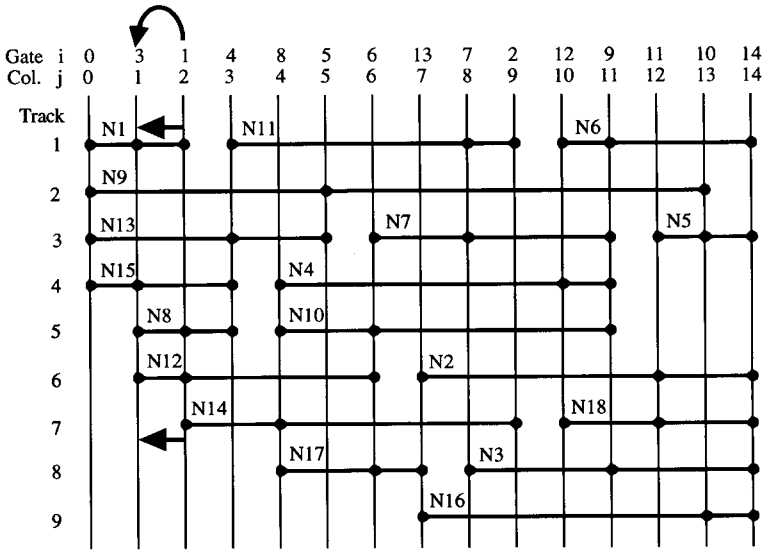
$$U_{i,j}(t + 1) = U_{i,j}(t) + \Delta U_{i,j}(t). \quad (5)$$

**Step 6.** Evaluate $V_{i,j}(t + 1)$ for $i, j = 1, \ldots, n$, using Eq. (1).

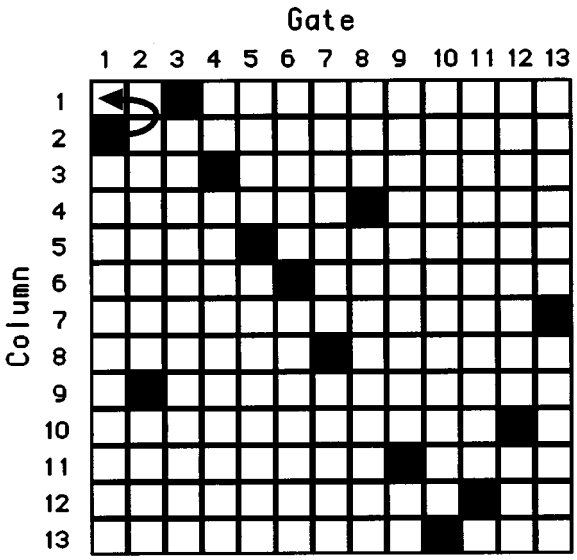
**Step 7.** If $t = t_{limit}$ then terminate this procedure else increment $t$ by 1 and go to Step 3.

Note that the time complexity of the left-edge-first algorithm in Step 3 is less than $O(nr)$ where $r$ is the number of tracks with $m$ processing elements (detectors) for m nets, and that the time complexity of the two-dimensional maximum neural network in Steps 2 and 6 is less than $O(n \log n)$ with $(n^2 + 1)/2$ processing elements (comparators) for $n^2$ neurons. In Steps 3 through 5, each process can run in parallel. Therefore, the time complexity of the proposed algorithm is less than $\max(O(nr), O(n \log n))$. The algorithm was implemented on a Macintosh PowerBook 170 and an HP 9000/710 computer, although the algorithm is executable either on a sequential machine or a parallel one.

In order to accelerate the simulation speed and increase the convergence rate, the following Eq. (6) was also used in Eq. (4) instead of Eq. (3) (Takefuji, 1992):




(a) The net layout representation to calculate $dU_{1,1}/dt$.




(b) The neural network array representation to calculate $dU_{1,1}/dt$.

**Figure 4** The explanation for the calculation of $dU_{1,1}/dt$.

**Table I** The problems and the results

| Problem # (Reference) | The number of gates | The number of nets | The number of tracks | |
|---|---|---|---|---|
| | | | This work | Reference |
| 1 (Fujii, 1987) | 9 | 8 | 4 | 4 |
| 2 (Hong, 1985) | 15 | 18 | 7 | 7 |
| 3 (Hong, 1985) | 29 | 37 | 13 | 13 |
| 4 (Hong, 1985) | 48 | 48 | **11** | 13 |
| 5 (Yamada, 1989) | 85 | 96 | **21** | 23 |

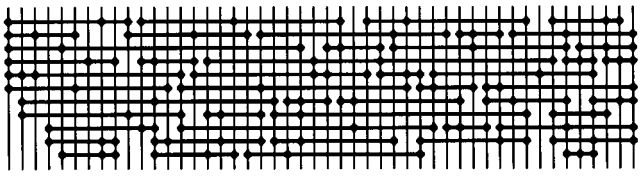


**Figure 5** One of the solutions for problem #4.

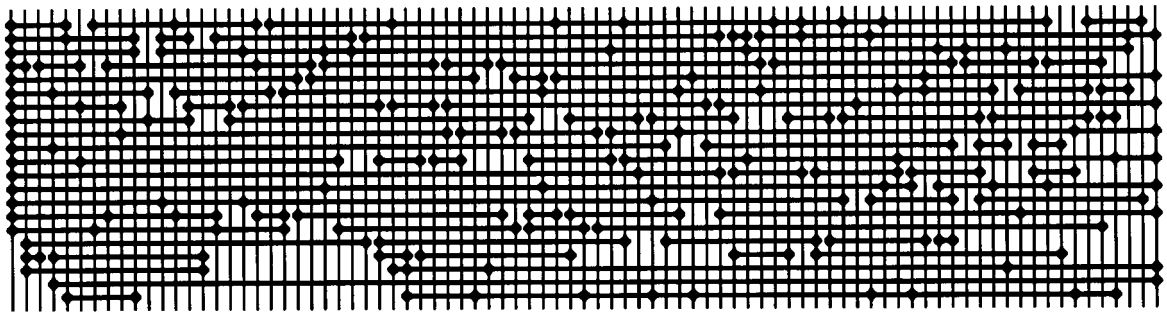


**Figure 6** One of the solutions for problem #5.

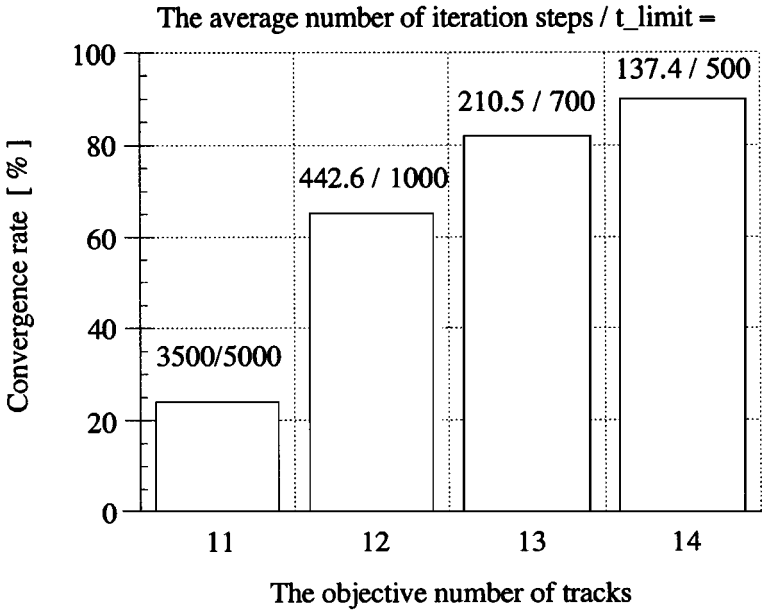



**Figure 7** Relationship between the convergence rate and the objective number of tracks.

$$\text{If } (t \bmod 10) < \omega \text{ then } \frac{dU_{i,j}}{dt} = (O - R)V_{i,j},$$
$$\text{else } \frac{dU_{i,j}}{dt} = O - R, \qquad (6)$$

where $\omega = 4$ was used through our simulation. This method helps the state of the system to escape from the local minimum.

We have examined the five benchmark problems to test our algorithm. Table 1 shows the problems and the results. The results of the best existing algorithms are also shown for comparison. Our algorithm has discovered the improved solutions which have the smaller numbers of tracks in both problems #4 and #5 over the best existing algorithms. Figures 5 and 6 depict one of the improved solutions for problems #4 and #5 respectively. These solutions are able to substantiate the effectiveness of our algorithm. Figure 7 shows the relationship between the convergence rate and the objective number of tracks for problem #4. Note that the result was obtained by 100 simulation runs over the

objective number of tracks using different initial states. The average number of iteration steps and $t_{limit}$ are also shown in Figure 7. It depicts that our algorithm found the answer of 13 tracks given by (Hong, 1989) within 700 iteration steps with more than 80% convergence rate and the answer of 11 tracks within 5000 iteration steps with about 25% convergence rate.

## CONCLUSION

In this paper we have proposed a near-optimal parallel algorithm using two-dimensional maximum neural networks for the one-dimensional gate assignment problems in VLSI layout design. The proposed algorithm requires an $n \times n$ neural network for an $(n+2)$-gate assignment problem. Our algorithm needs only one simple parameter $\omega$ compared to a simulated annealing which needs several parameters (Hong, 1989) and does not need any "good" initial gate-ordering. The simulation results demonstrate the effectiveness of the proposed algorithm over the existing algorithms in several benchmark problems.

The proposed algorithm could be also expandable to other significant VLSI layout problems: gate matrix layout problems (Wing, 1985; Huang, 1989; Singh, 1992), extended version of the one-dimensional gate assignment problems such as a two-dimensional gate assignment problem (Xu, 1990), and PLA folding problems (Lussio, 1990). In the gate matrix layout problems and the extended version of the one-dimensional gate assignment problems, although the nets are connected not only horizontally but also vertically, the net-allocation solution could be provided by another similar neural network instead of the left-edge-first algorithm. In the PLA folding problems, especially the multiple folding problem is the same as the one-dimensional gate assignment problem.

## APPENDIX. THE LEFT-EDGE-FIRST ALGORITHM

The following procedure is used for the left-edge-first algorithm to pack the nets with the smallest number of tracks within a given gate-ordering without the nets overlapping each other. The proof of this algorithm is given in (Hashimoto, 1971).

**Step 0.** Start the following procedure from the first track.

**Step 1.** Find a net whose leftmost gate is located in the leftmost column.

**Step 2.** If there is not the net, then continue finding a net whose leftmost gate is located in the next column until the net is found.

**Step 3.** Put the net on the track.

**Step 4.** Except for the allocated net(s), find a net whose leftmost gate is located in the next column of the column the rightmost gate of the previously allocated net is located in.

**Step 5.** If there is not the net, then continue finding a net whose leftmost gate is located in the next column until the net is found.

**Step 6.** If there is not the net, then move down to the next track and continue from Step 1 until all the nets are allocated.

## REFERENCES


Asano, T. (1982) "An optimum gate placement algorithm for MOS one-dimensional arrays," *J. Digital Syst.* **VI**(1) 1–27.

Fujii, T. et al. (1987) "A heuristic algorithm for gate assignment in one-dimensional array Approach," *IEEE Trans. Computer-Aided Design* vol. CAD-6, 159–164.

Hashimoto, A. and Stevens, J. (1971) "Wire routing by optimizing channel assignment within large apertures," in: *Proc. 18th Design Automation Workshop* 155–169.

Hong, Y.-S., Park, K-H. and Kim, M. (1989) "A heuristic algorithm for ordering the columns in one-dimensional logic arrays," *IEEE Trans. Computer-Aided Design* **8** 547–562.

Hopfield, J. and Tank, D. (1985) "Neural computation of decisions in optimization problems," *Biol. Cybern.* **52** 141–152.

Huang, S. and Wing, O. (1989) "Improved Gate Matrix Layout," *IEEE Trans. Computer-Aided Design* **8** 875–889.

Kashiwabara, T. and Fujishima, T. (1979) "NP-completeness of the problem of finding a minimal-clique-number interval graph containing a given graph as a subgraph," in: *Proc. 1979 Int. Symp. Circuit Syst* 657–660.

Lussio, F. and Pinitti, M.C. (1990) "Suboptimal solution for PLA multiple column folding," *Computer-Aided Design* **22**(8) 515–520.

Ohtsuki, T. et al. (1979) "One-dimensional logic gate assignment and interval graphs," *IEEE Trans. Circuits Syst.*, vol. CAS-26 675–684.

Singh, U., and Roger Chen, C.Y. (1992) "From logic to symbolic layout for gate matrix," *IEEE Trans. Computer-Aided Design* **11** 216–227.

Takefuji, Y. (1992) Neural Network Parallel Computing, Kluwer Academic Publishers, Massachusetts.

Weinberger, A. (1967) "Large scale integration of MOS complex logic: A layout method," *IEEE J. Solid-State Circuits*, vol. SC-2 182–190.

Wing, O., Huang, S. and Wang, R. (1985) "Gate Matrix Layout," *IEEE Trans. Computer-Aided Design*, vol. CAD-4 220–231.

Xu, D-M., Kuh, E.S. and Chen, Y.-K. (1990) "An extended 1-D assignment problem: net assignment in gate matrix layout," in: *1990 IEEE International Symposium on Circuits and Systems* 1692–1696.

Yamada, S., Okude, H. and Kasai, T. (1989) "A hierarchical algorithm for one-dimensional gate assignment based on contraction of nets," *IEEE Trans. Computer-Aided Design* **8** 622–629.