A Parallel Algorithm for Channel Routing Problems

Nobuo Funabiki, *Student Member, IEEE*, and Yoshiyasu Takefuji, *Member, IEEE*

Abstract—**A parallel algorithm for channel routing problems is presented in this paper. The channel routing problem is very important in the automatic layout design of VLSI circuits and printed circuit boards. The problem is to route the given interconnections between two rows of terminals on a multilayer channel where the channel area must be minimized. Although several algorithms have been proposed for two-layer problems, two-layer-and-over-the-cell problems, and three-layer problems, the current advancement of VLSI chip technology allows us to use four layers composed of two metal layers and two polysilicon layers for routing in a chip. The goal of the proposed parallel algorithm is to find the near-optimum routing solution for the given interconnections in a short time. The algorithm is applied for the four-layer channel routing problems where it requires $n \times m \times 2$ processing elements for the n-net-m-track problem. The algorithm has three advantages over the conventional algorithms: 1) it can be easily modified for accommodating more than four-layer problems, 2) it runs not only on a sequential machine but also on a parallel machine with maximally $n \times m \times 2$ processors, and 3) the program size is very small. The algorithm is verified by solving seven benchmark problems where the algorithm finds routing solutions in a nearly constant time with $n \times m \times 2$ processors.**

I. Introduction

THE channel routing problem in a multilayer channel is very important in the automatic layout design of VLSI circuits and printed circuit boards. A channel consists of two parallel, horizontal rows of points, which are called terminals. The terminals are placed at regular intervals and identify the columns of the channel. A net consists of a set of terminals that must be interconnected through certain routing paths. Some nets may have a connection point at one or both ends (top and/or bottom) of the channel. The channel routing problem is not only to route the given nets or interconnections between the terminals on the multilayer channel, but also to minimize the channel area.

The sequential algorithms for the two-layer channel routing problems have been extensively studied [1]–[15]. These algorithms have the following common features. The routing paths consist of the horizontal segments which are parallel to the terminals of the channel and the vertical segments. All the horizontal segments of the routing paths are assigned on one layer and all the vertical segments of them are assigned on another layer. The connections between the horizontal segments and the vertical segments are made through the contact windows, which are called via holes. For the integrated circuits, typically the horizontal segments are embedded on a metal layer while the vertical segments are embedded on a polysilicon and/or diffusion layer. Any two routing paths on the same layer cannot be placed within a certain distance of each other, which is called the separation condition. For convenience, a unit grid is superimposed on the channel where the size of one unit satisfies the separation condition and all the terminals are located at the grid points. All the routing paths on the channel must follow the grid lines. The horizontal segments are called tracks and the vertical segments are called columns. In this model, the separation condition is that no two nets must be embedded on the same track or on the same column if they overlap there, which is called overlapping condition. When the width of the channel is fixed, minimizing the channel area is equivalent to minimizing the number of the necessary tracks where all the given nets must be embedded.

In some sequential algorithms, doglegging is introduced where a routing path of a net is split into two or more horizontal segments on different tracks [3], [7]–[15]. Doglegging is sometimes effective in reducing the number of tracks of the channel and to solve the cyclic conflict. The cyclic conflict occurs when one net interconnects a top terminal on the ith column with a bottom terminal on the jth column while another net interconnects a bottom terminal on the ith column with a top terminal on the jth column. Note that a top terminal means a terminal at the top end of a channel and a bottom terminal means a terminal at the bottom end of a channel. However, this cyclic conflict occurs infrequently and it can often be avoided by rearranging the terminal placement. Doglegging requires additional via holes, which reduce the reliability of the VLSI system and increase the manufacturing cost. It is desirable either to reduce the number of doglegs or to eliminate doglegs completely for the practical use.

To further reduce the channel area, several sequential algorithms for two-layer-and-over-the-cell channel routing problems [16]–[21] and for three-layer channel routing problems have also been proposed [22], [23]. The over-the-cell channel routing algorithms use not only the conventional two layers of channel but also the area over the cells for interconnections. The three-layer channel routing algorithms use two layers for the horizontal seg-

Manuscript received June 14, 1990; revised December 11, 1990. This work was supported by the National Science Foundation under Grant MIP-8902819. This paper was recommended by the Editor, A. E. Dunlop.

N. Funabiki is with the Department of Electrical Engineering and Applied Physics, Case Western Reserve University, Cleveland, OH 44106, and with the System Engineering Division, Sumitomo Metal Industries, Ltd., Ibaraki, Japan 314.

Y. Takefuji is with the Department of Electrical Engineering and Applied Physics, Case Western Reserve University, Cleveland, OH 44106.

IEEE Log Number 9104584.

ments and one layer for the vertical segments. Although these algorithms find better solutions than the algorithms for two-layer problems, they are still based on two-layer routing technique so that it is not easy to apply them for more than three layers problems. It is not guaranteed to find the optimum solution with their algorithms.

The channel routing problems are proved to be NP-complete. LaPaugh shows the NP-completeness of the channel routing problem without doglegging [5]. Szymanski shows the NP-completeness of the channel routing problem with doglegging [12]. Gudmundsson *et al.* show the NP-completeness of the over-the-cell channel routing problem [19].

Several parallel routing algorithms have been proposed for the knock-knee model [24], [25], for the river routing [26], [27], and for the global routing [28], [29]. A parallel channel routing algorithm based on simulated annealing was also proposed in [30], but the algorithm does not guarantee finding optimum solutions. Although parallel channel routing algorithms which flexibly accommodate future advancement of VLSI technology have been in great demand, few such parallel algorithms have been reported. The parallel algorithms will be extensively used in the near future.

This paper is organized as follows: Section II contains the basic background about our neural network approach for channel routing problems. Section III describes the system representation for four-layer channel routing problems in our algorithm. Section VI gives the details of the algorithm and of the implementation on a sequential machine. In Section V simulation results in the seven benchmark problems are shown and discussed. Section VI explores the modification for more than four-layer problems. Concluding remarks are presented in Section VII.

II. Neural Network Approach for Channel Routing

In this paper we propose a parallel algorithm for four-layer channel routing problems which can be easily extended to problems involving more than four layers. The current state of VLSI chip technology allows us to use four layers—two metal layers and two polysilicon layers—for routing in a chip [31]. The algorithm embeds the given nets on one of the two independent two-layer channels which have the same structure as the conventional two-layer channels. In other words, the four-layer channel has two layers for the horizontal segments of the nets and two layers for the vertical segments of them. The proposed algorithm, with slight modification, is able to solve two-layer problems, two-layer-and-over-the-cell problems, and three-layer problems.

The proposed parallel algorithm is based on a three-dimensional artificial neural network model which is composed of a large number of massively interconnected simple processing elements. The processing elements are called neurons because they perform the function of simplified biological neurons. The artificial neural network model for solving combinatorial optimization problems was introduced by Hopfield and Tank [32]. However, they use not only the sigmoid neuron model, which is slow for the convergence, but also a decay term which is proven to disturb the convergence [35]. In order to speed up the convergence, the McCulloch–Pitts neuron model [33] has been used without the decay term and applied to several NP-complete and optimization problems [34]–[42].

The output V_{ijk} of the ijkth processing element based on the modified McCulloch–Pitts neuron model [36] is

$$\begin{aligned} V_{ijk} &= 1 \quad \text{if } U_{ijk} > 0 \text{ and } U_{ijk} = \max\{U_{iqr}\} \\ &\qquad \text{for } q = 1, \cdots, m \text{ and } r = 1, 2 \\ &= 0 \quad \text{otherwise} \end{aligned} \tag{1}$$

where U_{ijk} is the input of the ijkth processing element and m is the number of tracks of the channel. The change of the input U_{ijk} is given by the partial derivatives of the computational energy E with respect to the output V_{ijk}, where E is a $2nm$-variable function: $E(V_{111}, V_{112}, \cdots, V_{nm2})$. Note that n is the number of given nets. The equation is called a motion equation, given by

$$\frac{dU_{ijk}}{dt} = -\frac{\partial E(V_{111}, V_{112}, \cdots, V_{nm2})}{\partial V_{ijk}}. \tag{2}$$

The motion equation also presents the interconnections between the processing elements.

Whatever computational energy function E is given, the motion equation forces it to monotonically decrease. The following proof shows that the motion equation forces the state of the system to converge to the local minimum where the energy function E is usually nonconvex. The motion equation performs the gradient decent method to minimize the energy function.

Proof: Consider the derivatives of the computational energy function E with respect to time t:

$$\begin{aligned} \frac{dE}{dt} &= \sum_i \sum_j \sum_k \frac{dV_{ijk}}{dt} \frac{\partial E}{\partial V_{ijk}} \\ &= \sum_i \sum_j \sum_k \frac{dV_{ijk}}{dt} \left(-\frac{dU_{ijk}}{dt} \right) \quad \text{where the motion equation replaces } \frac{\partial E}{\partial V_{ijk}} \text{ by } \left(-\frac{dU_{ijk}}{dt} \right) \\ &= -\sum_i \sum_j \sum_k \left(\frac{dU_{ijk}}{dt} \frac{dV_{ijk}}{dU_{ijk}} \right) \left(\frac{dU_{ijk}}{dt} \right) \\ &= -\sum_i \sum_j \sum_k \left(\frac{dV_{ijk}}{dU_{ijk}} \right) \left(\frac{dU_{ijk}}{dt} \right)^2 \le 0. \end{aligned} \tag{3}$$

As long as the input/output function of the processing elements obeys the nondecreasing function, dV_{ijk}/dU_{ijk} must be positive or zero so that dE/dt is negative or zero.

Therefore the state of the system is always guaranteed to converge to the local minimum [39]. Q.E.D.

III. System Representation

Fig. 1(a) shows a channel routing problem in [7] where ten nets are given to be routed in a four-layer channel which has three tracks. The ten nets are ($T2$, $T5$), ($B1$, $B6$), ($B2$, $B4$), ($T3$, $T9$), ($B3$, $T4$, $B5$), ($T6$, $B7$), ($T7$, $B11$), ($B8$, $B10$), ($B9$, $T10$, $B12$), and ($T11$, $T12$), where Ti indicates the top terminal at the ith column and Bj indicates the bottom terminal at the jth column. For example, net 1 ($T2$, $T5$) has two terminals to be interconnected; one is the top terminal at the second column and the other is the top terminal at the fifth column. Fig. 1(b) shows the system representation for this problem. Only one horizontal segment is used for one net, and the vertical segments of the net are automatically assigned on the layer corresponding to the layer on which the horizontal segment of the net is assigned. The channel routing problem can be simplified into the layer-track problem. This involves finding the layer number and the track number for embedding the horizontal segment of the given net without violating the overlapping conditions.

Six ($= 3 \times 2$) processing elements are used to indicate layer and track on which a given net should be embedded. Because three tracks of two layers for the horizontal segments are available for the ten-net and three-track problem, a total of 60 ($= 10 \times 3 \times 2$) processing elements are used in this problem. V_{ijk} represents the output of the ijkth processing element, which corresponds to the ith net and the jth track on the kth layer for $i = 1, \cdots, 10, j = 1, 2, 3$, and $k = 1, 2$. Generally $n \times m \times 2$ processing elements are used to represent n nets, m tracks, and two horizontal segment layers for the four-layer channel routing problems where $m \times 2$ processing elements describe the track number and the layer number for a single net.

The output of one and only one processing element among the $m \times 2$ processing elements should be nonzero to locate the net on one of the m tracks of two layers. The nonzero output means that the net should be embedded on the corresponding track and layer. Fig. 1(b) shows one of the solutions for this problem, where the black squares indicate the nonzero output of the processing elements and the white squares indicate the zero output of the processing elements. Fig. 1(b) shows that the net 1 ($T2$, $T5$) is assigned on the second track of the first two-layer channel and the net 2 ($B1$, $B6$) is assigned on the first track of the second two-layer channel, and so on. Note that the two-layer channel means a pair of two layers for the horizontal segments and vertical segments. Fig. 1(c) shows the routing solution corresponding to Fig. 1(b).

Each net must satisfy the separation conditions; in other words, no two different nets must violate the overlapping conditions. Fig. 2 shows the overlapping conditions for the horizontal segments of the nets, where head_i indicates the leftmost column number of the ith net and tail_i indicates the rightmost column number of the ith net. The horizontal overlapping conditions for the ith-net-jth-track-

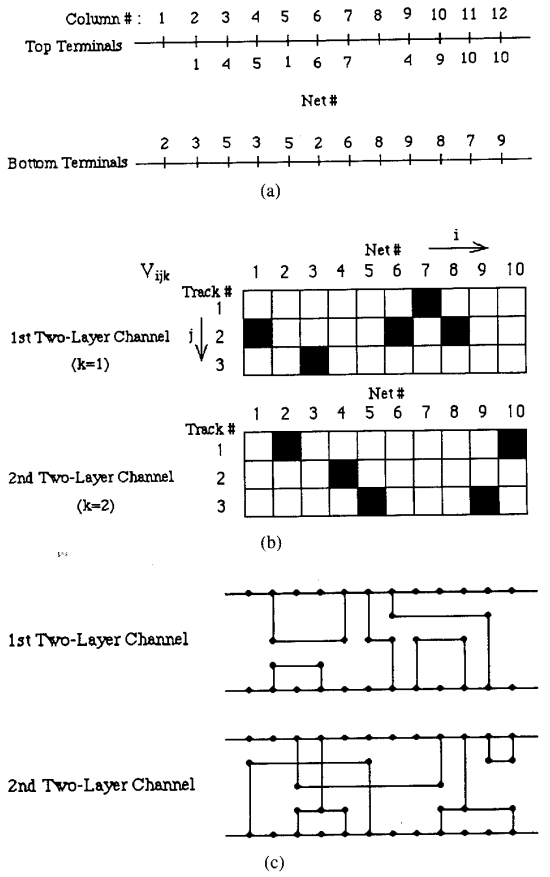


Fig. 1. System representation for a 10-net-3-track problem. (a) A 10-net-3-track problem. (b) The output state of processing elements. (c) The routing solution corresponding to (b).

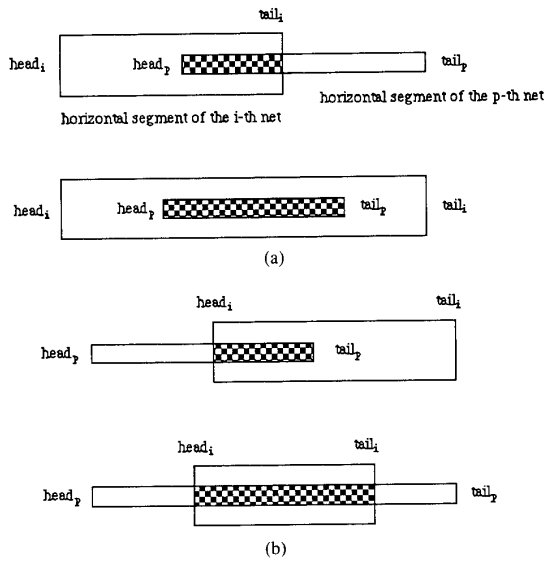


Fig. 2. Overlapping conditions for horizontal segments. (a) Overlapping condition: $\text{head}_i \le \text{head}_p \le \text{head}_i$. (b) Overlapping condition: $\text{head}_p \le \text{head}_i \le \text{head}_p$.

kth-layer processing element are given by

$$\sum_{\substack{p=1 \\ p \neq i \\ \text{head}_i \leq \text{head}_p \leq \text{tail}_i}}^{n} V_{pjk} + \sum_{\substack{p=1 \\ p \neq i \\ \text{head}_p \leq \text{head} \leq \text{tail}_p}}^{n} V_{pjk}. \quad (4)$$

This horizontal condition is nonzero if the horizontal segments of the other nets overlap the horizontal segment of the ith net on the jth track of the kth layer.

Fig. 3 shows the overlapping conditions for the vertical segments of the nets, where the ith net and the pth net have terminals on the opposite sides of the same column. The vertical overlapping conditions for the ith-net-jth-track-kth-layer processing element are given by

$$\sum_{\substack{p=1 \\ p \neq i}}^{n} T_{ip} \sum_{q=1}^{j} V_{pqk} + \sum_{\substack{p=1 \\ p \neq i}}^{n} B_{ip} \sum_{q=j}^{m} V_{pqk} \quad (5)$$

where T_{ip} is 1 if the pth net has a bottom terminal at the column at which the ith net has a top terminal and is 0 otherwise. B_{ip} is 1 if the pth net has a top terminal at the column at which the ith net has a bottom terminal and is 0 otherwise. The vertical condition is nonzero if the vertical segments of the other nets overlap the vertical segments of the ith net.

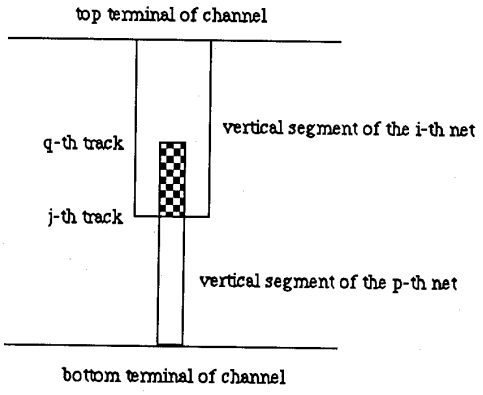
The motion equation of the ith-net-jth-track-kth-layer processing element for the n-net-m-track problem is given by

$$\frac{dU_{ijk}}{dt} = -A\left(\sum_{q=1}^{m}\sum_{r=1}^{2} V_{iqr} - 1\right) - B\left(\sum_{\substack{p=1 \\ p \neq i \\ \text{head}_i \leq \text{head}_p \leq \text{tail}_i}}^{n} \cdot V_{pjk} + \sum_{\substack{p=1 \\ p \neq i \\ \text{head}_p \leq \text{head}_i \leq \text{tail}_p}}^{n} V_{pjk}\right)$$

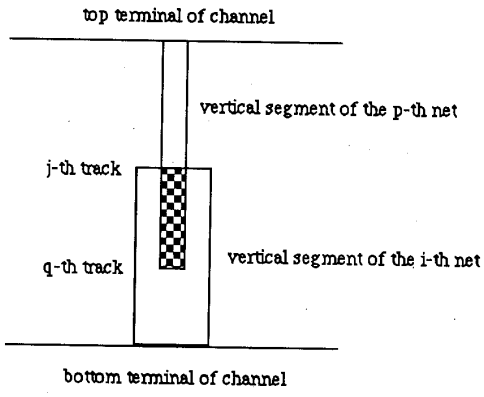
$$- B\left(\sum_{\substack{p=1 \\ p \neq i}}^{n} T_{ip} \sum_{q=1}^{j} V_{pqk} + \sum_{\substack{p=1 \\ p \neq i}}^{n} B_{ip} \sum_{q=j}^{m} V_{pqk}\right)$$

$$+ Ch\left(\sum_{q=1}^{m}\sum_{r=1}^{2} V_{iqr}\right). \quad (6)$$

The first term (A term) in (6) forces one and only one output among the $2m$ processing elements to be nonzero where the ith net is assigned. The second and third terms (B terms) exert the inhibitory forces. The B terms discourage the output of the ijkth processing element from being nonzero if the other nets overlap with the ith net. The last term (C term) provides the hill climbing which allows the state of the system to escape from the local minimum and to converge to the global minimum. The C term encourages the output of the ijkth processing element to be nonzero if the output of all the processing elements for the ith net is zero. The function $h(x)$ is 1 if $x = 0$ and is 0 otherwise. A, B, and C are constant coefficients.



(a)



(b)

Fig. 3. Overlapping conditions for vertical segments. (a) Overlapping condition: $j \geq q$. (b) Overlapping condition: $j \leq q$.

IV. Parallel Algorithm

The following procedure describes the proposed parallel algorithm based on the first-order Euler method for the channel routing problem where n nets and m tracks on the four-layer channel are given.

0) Set $t = 0$, $A = B = 1$, $C = 10$, $U\_\text{min} = -20$ and $T\_\text{max} = 500$.
1) The initial values of $U_{ijk}(t)$ for $i = 1, \cdots, n$, $j = 1, \cdots, m$, and $k = 1, 2$ are uniformly randomized between 0 and U_min, and 0 is assigned as the initial values of $V_{ijk}(t)$ for $i = 1, \cdots, n$, $j = 1, \cdots, m$, and $k = 1, 2$.
2) Use the motion equation in (6) to compute $\Delta U_{ijk}(t)$ for $i = 1, \cdots, n$, $j = 1, \cdots, m$, and $k = 1, 2$.

If (t mod 10) ≤ 5, then

$$\Delta U_{ijk}(t) = -A\left(\sum_{q=1}^{m}\sum_{r=1}^{2} V_{iqr}(t) - 1\right) - B\left[\sum_{\substack{p=1\\p\neq i}}^{n} V_{pjk}(t) + \sum_{\substack{p=1\\p\neq i}}^{n} V_{pjk}(t)\right] \cdot V_{ijk}(t) - B\left[\sum_{\substack{p=1\\p\neq i}}^{n} T_{ip}\sum_{q=1}^{j} V_{pqk}(t) + \sum_{\substack{p=1\\p\neq i}}^{n} B_{ip}\sum_{q=j}^{m} V_{pqk}(t)\right] V_{ijk}(t) + Ch\left(\sum_{q=1}^{m}\sum_{r=1}^{2} V_{iqr}(t)\right). \quad (7)$$

Otherwise,

$$\Delta U_{ijk}(t) = -A\left(\sum_{q=1}^{m}\sum_{r=1}^{2} V_{iqr}(t) - 1\right) - B\left[\sum_{\substack{p=1\\p\neq i}}^{n} V_{pjk}(t) + \sum_{\substack{p=1\\p\neq i}}^{n} V_{pjk}(t)\right] - B\left[\sum_{\substack{p=1\\p\neq i}}^{n} T_{ip}\sum_{q=1}^{j} V_{pqk}(t) + \sum_{\substack{p=1\\p\neq i}}^{n} B_{ip}\sum_{q=j}^{m} V_{pqk}(t)\right] + Ch\left(\sum_{q=1}^{m}\sum_{r=1}^{2} V_{iqr}(t)\right). \quad (8)$$

3) Compute $U_{ijk}(t+1)$ for $i = 1, \cdots, n$, $j = 1, \cdots, m$, and $k = 1, 2$ based on the first-order Euler method:

$$U_{ijk}(t+1) = U_{ijk}(t) + \Delta U_{ijk}(t). \quad (9)$$

4) If $U_{ijk}(t+1) > U\_\max$ then $U_{ijk}(t+1) = U\_\max$ (10)

If $U_{ijk}(t+1) < U\_\min$ then $U_{ijk}(t+1) = U\_\min$ (11)

for $i = 1, \cdots, n$, $j = 1, \cdots, m$, and $k = 1, 2$.

5) Evaluate the values of $V_{ijk}(t+1)$ for $i = 1, \cdots, n$, $j = 1, \cdots, m$, and $k = 1, 2$:

$$V_{ijk}(t+1) = 1 \quad \text{if } U_{ijk}(t+1) > 0 \text{ and } U_{ijk}(t+1) = \max\{U_{iqr}(t+1)\} \text{ for } q = 1, \cdots, m \text{ and } r = 1, 2$$
$$= 0 \quad \text{otherwise.} \quad (12)$$

6) If $V_{ijk}(t) = 1$ and $\Delta U_{ijk}(t) = 0$ for $i = 1, \cdots, n$, $\exists j \in \{1, \cdots, m\}$, and $\exists k \in \{1, 2\}$ (all nets are embedded without conflicts) or $t = T\_\max$, then terminate this procedure; otherwise increment t by 1 and go to step 2.

The modified motion equations in step 2 and the range limitation of the input $U_{ijk}(t+1)$ in step 4 improve the convergence frequency to the global minimum [36]. $T\_\max$ is the maximum number of iteration steps, and $U\_\max$ and $U\_\min$ are the upper and lower limits of $U_{ijk}(t+1)$ respectively.

We divide the given nets into two groups to perform this algorithm in two phases. The first group consists of the longer nets, where the net width between the leftmost column and rightmost column is more than 30% of the total channel width, and the other group consists of the remaining, shorter nets. It is necessary to fix the locations of the nets in the first group in the beginning because it is

difficult or impossible to find the locations for the longer nets in the channel if many nets are embedded. Our strategy is that in the first phase we assign the first group nets by applying our algorithm only to them and in the second phase we assign the second group nets by applying the algorithm for all the nets where the outputs of the processing elements corresponding to the first group nets are fixed.

The state of $n \times m \times 2$ processing elements for the n-net-m-track problem can be updated synchronously or asynchronously. In this paper, the synchronous parallel system is simulated on a sequential machine. The synchronous parallel system can be performed on maximally $n \times m \times 2$ processors while the performance is improved as more processors are used. The following procedure/program outlines the sequential program for simulating the synchronous parallel system:

Program Parallel-Simulator-on-a-Sequential-Machine

```
. . . . .
initialization of U_ijk and V_ijk for i:=1 to n, for j:=1 to
m, and for k:=1 to 2;
. . . . .
{*** Main Program ***}
while (a set of conflicts is not empty) do
begin
. . . . .
{*** Updating all input values ***}
   for i:=1 to n
   for j:=1 to m
   for k:=1 to 2
      U_ijk:=U_ijk+ΔU_ijk;
{*** End of the first loop ***}
. . . . .
{*** Updating all output values ***}
   for i:=1 to n
   for j:=1 to m
   for k:=1 to 2
      If U_ijk>0 and U_ijk=U_i-max then V_ijk:=1 else
      V_ijk:=0;
      {U_i-max is maximum among {U_iqr} for q=1,···,
      m and r=1,2}
{*** End of the second loop ***}
. . . . .
end;
{*** Main Program end ***}
```

In the first loop all input values U_{ijk} are sequentially updated while all output values V_{ijk} are fixed. Then, in the second loop, all output values V_{ijk} are sequentially updated while all input values U_{ijk} are fixed. This is equivalent to simultaneously updating the values of all inputs and outputs.

V. Simulation Results and Discussion

The simulator based on the proposed algorithm has been developed on a Macintosh SE/30 and IIfx in order to verify our algorithm. The core program of the simulator involves fewer than 100 steps in Turbo Pascal. The simulator solved seven benchmark problems in [7], where it took less than 10 min for the convergence on a Macintosh IIfx. The benchmark problems have no cyclic conflict. Figs. 4–10 show solutions from our algorithm for the seven problems respectively. Our algorithm found several other solutions in the same problems from different initial values of $U_{ijk}(t)$. Table I shows the numbers of nets in the seven problems and the numbers of tracks which our algorithm needed to route all nets in the four-layer channel.

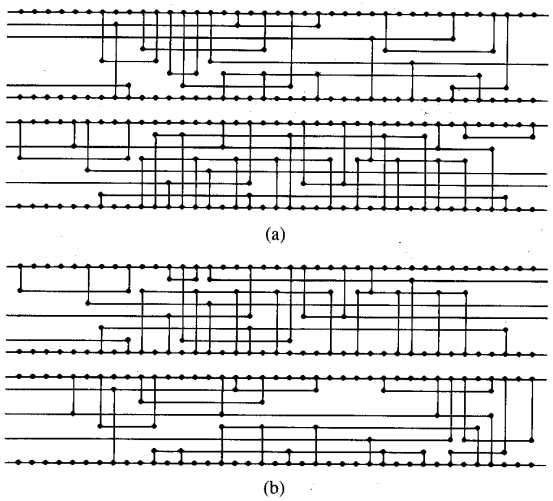


Fig. 4. Solutions for example 1. (a) Solution 1. (b) Solution 2.

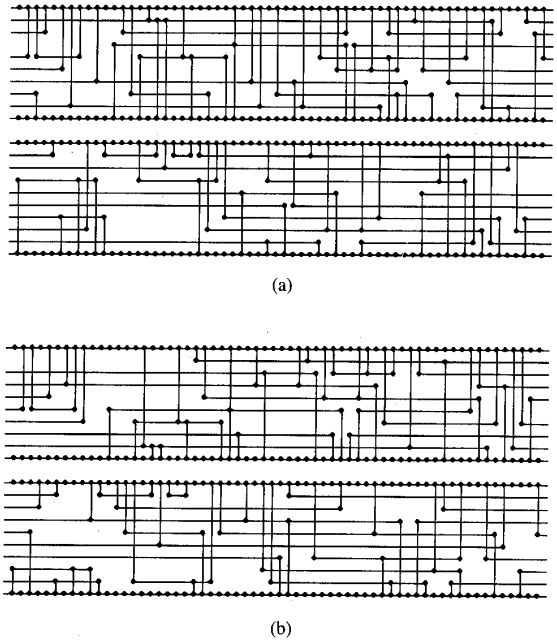
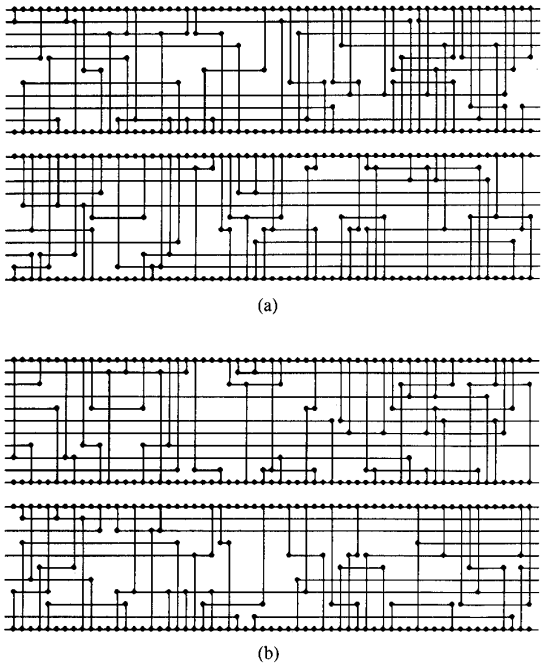
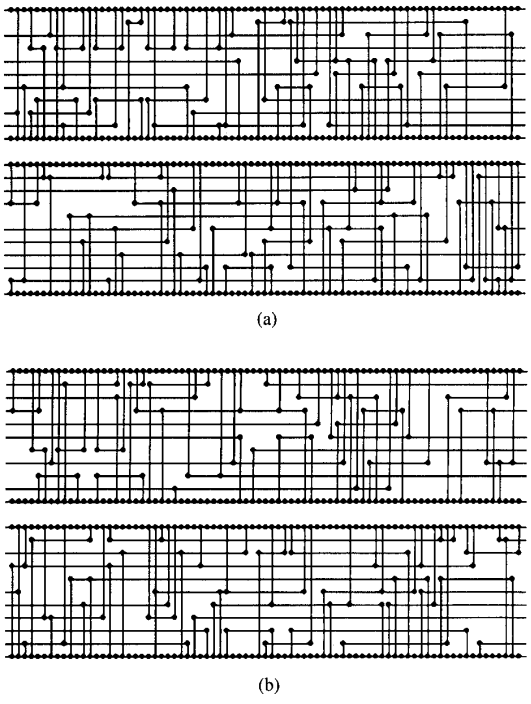


Fig. 5. Solutions for example 3a. (a) Solution 1. (b) Solution 2.



(a) (b)

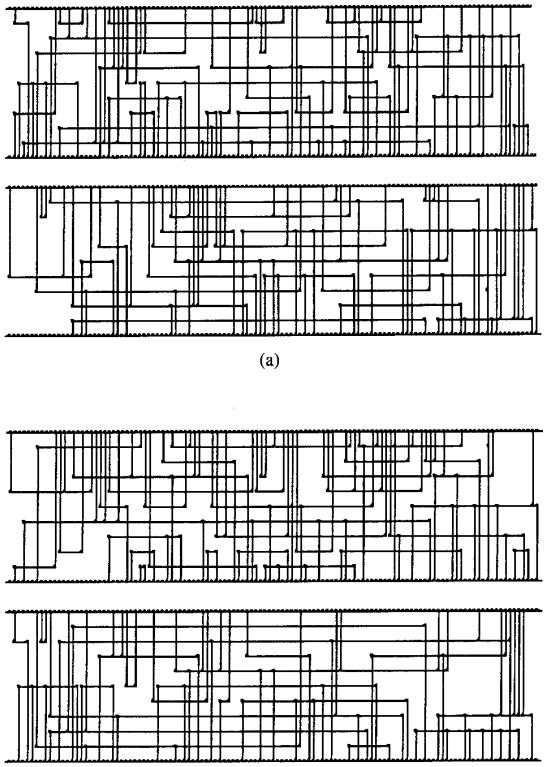
Fig. 6. Solutions for example 3b. (a) Solution 1. (b) Solution 2.



(a) (b)

Fig. 7. Solutions for example 3c. (a) Solution 1. (b) Solution 2.

They are compared with the two-layer routing solutions in [7] and the three-layer routing solutions in [22], where no doglegging is allowed. Our algorithm found the theoretically optimum solutions for the benchmark problems, except for Deutsch's example, where it found the near-optimum solutions. Table II shows the frequency of the convergence to the optimum solutions and the average numbers of iteration steps where 100 simulation runs were performed for each problem. For each simulation run, different initial values of $U_{ijk}(t)$ are randomly generated. Fig. 11 shows the relationship between the frequency and the number of iteration steps to converge to the optimum solutions in two problems. The simulation results empirically show that our algorithm solves the channel routing problems in a nearly constant time with $n \times m \times 2$ processors. It is observed that the $O(nm)$ time sequential algorithm is achieved by the modified proposed algorithm, where n is the number of nets on an m-track-four-layer channel.

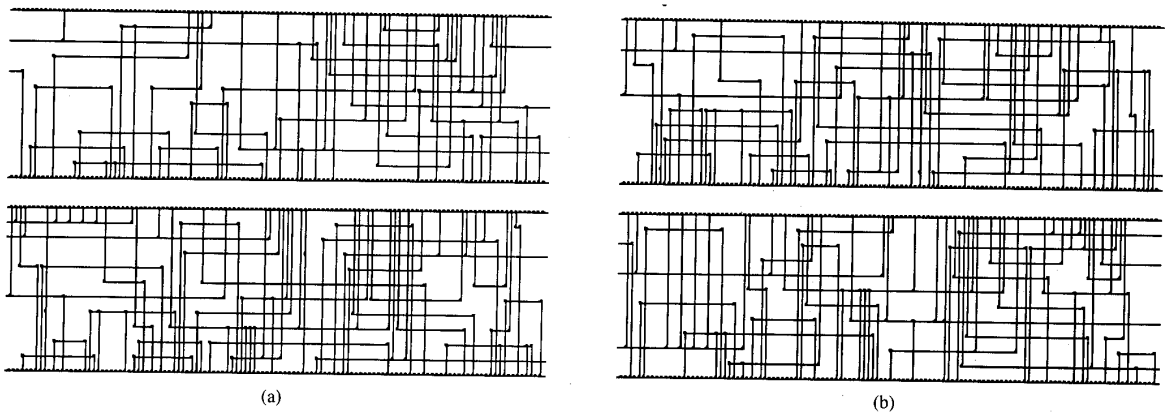


(a) (b)

Fig. 8. Solutions for example 4b. (a) Solution 1. (b) Solution 2.

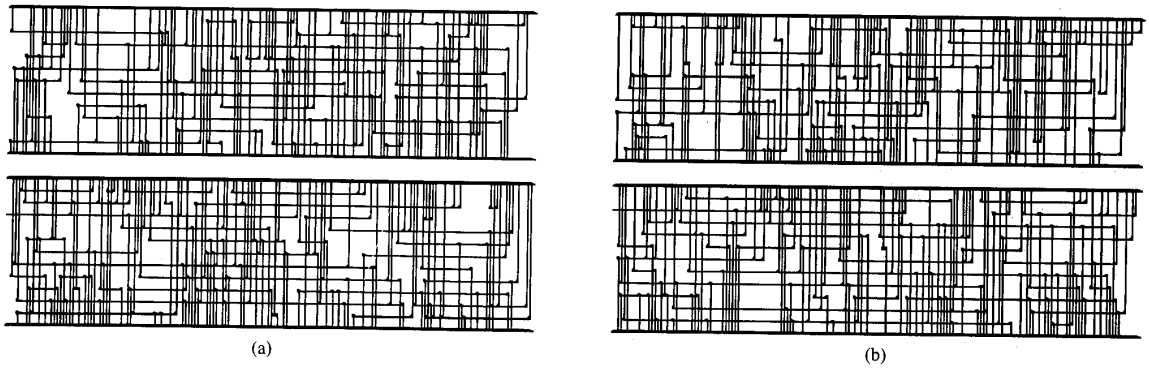
VI. Modification for More Than Four-Layer Problems

In order to show the flexibility of the proposed parallel algorithm, we modified and applied the algorithm for $2s$-layer channel routing problems, where a channel has s layers for the horizontal segments of the nets and s layers for the vertical segments of them. The motion equation in



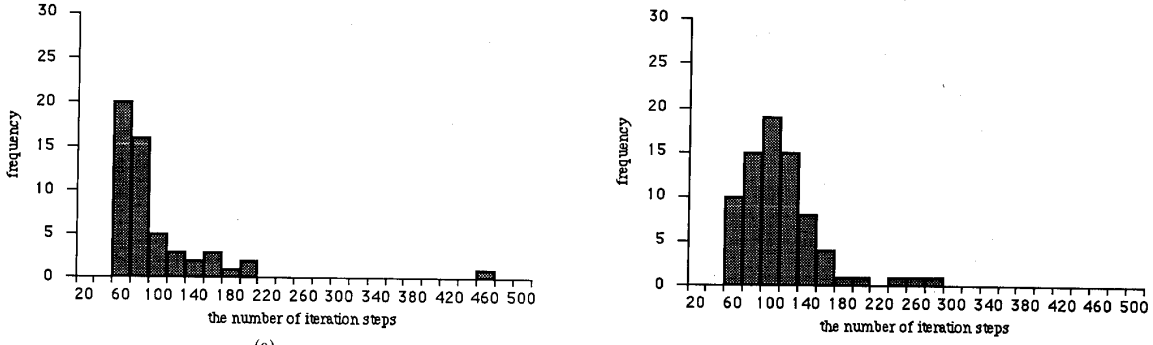
(a) (b)

Fig. 9. Solutions for example 5. (a) Solution 1. (b) Solution 2.



(a) (b)

Fig. 10. Solutions for Deutsch's difficult example. (a) Solution 1. (b) Solution 2.



(a) (b)

Fig. 11. The relationship between the frequency and the number of iteration steps to converge to the optimum solutions. (a) Example 3b. (b) Example 5.

TABLE I
COMPARISONS OF TRACK NUMBERS WITH OTHER NO-DOGLEGGING ROUTERS

Problem No.	Number of Nets	Our Four-Layer Solutions	Two-Layer Solutions in [7]	Three-Layer Solutions in [22]
Example 1	21	6	12	7
Example 3a	45	8	15	8
Example 3b	47	9	17	10
Example 3c	54	9	18	9
Example 4b	55	9	17	13
Example 5	61	10	20	10
Difficult Example	72	11	28	23

TABLE II
SUMMARY OF SIMULATION RESULTS

Problem No.	Average Iteration Steps to Optimum Solutions	Convergence Frequency to Optimum Solutions
Example 1	84.4	44%
Example 3a	147.6	29%
Example 3b	89.9	53%
Example 3c	280.4	7%
Example 4b	150.3	40%
Example 5	101.6	76%
Difficult Example	133.1	17%

(6) is slightly modified as follows:

$$\frac{dU_{ijk}}{dt} = -A\left(\sum_{q=1}^{m}\sum_{r=1}^{s} V_{iqr} - 1\right) - B\left(\sum_{\substack{p=1\\p\neq i\\ \text{head}_i \le \text{head}_p \le \text{tail}_i}}^{n} \cdot V_{pjk} + \sum_{\substack{p=1\\p\neq i\\ \text{head}_p \le \text{head}_i \le \text{tail}_p}}^{n} V_{pjk}\right)$$

$$- B\left(\sum_{\substack{p=1\\p\neq i}}^{n} T_{ip}\sum_{q=1}^{j} V_{pqk} + \sum_{\substack{p=1\\p\neq i}}^{n} B_{ip}\sum_{q=j}^{m} V_{pqk}\right)$$

$$+ Ch\left(\sum_{q=1}^{m}\sum_{r=1}^{s} V_{iqr}\right). \tag{13}$$

As the example, we examined Deutsch's difficult example in six-layer, eight-layer, and ten-layer problems. Figs. 12–14 show solutions for three problems. Table III shows the numbers of tracks which the algorithm needs to route all nets in the 4–10-layer channel for Deutsch's difficult example. Our algorithm found the optimum solutions for each problem in this sHsV (s-horizontal-layer-s-vertical-layer) model. The algorithm can be also easily modified and extended for solving the $3s$-layer channel routing problems in the $2s$HsV model. The $2s$HsV model has the possibility of giving fewer track solutions in the same problem than the sHsV model. We conclude that our parallel algorithm is so flexible that it can easily accommodate future advances in silicon technology.

VII. CONCLUSION

This paper proposes a parallel algorithm for four-layer channel routing problems and $2s$-layer channel routing problems in the sHsV model. The algorithm requires $n \times m \times 2$ processing elements for the n-net-m-track routing problem on the four-layer channel. The algorithm runs not only on a sequential machine but also on a parallel machine with maximally $n \times m \times 2$ processors. In seven benchmark problems, the algorithm finds routing solutions in a nearly constant time with $n \times m \times 2$ processors. The simulation results support the consistency of the

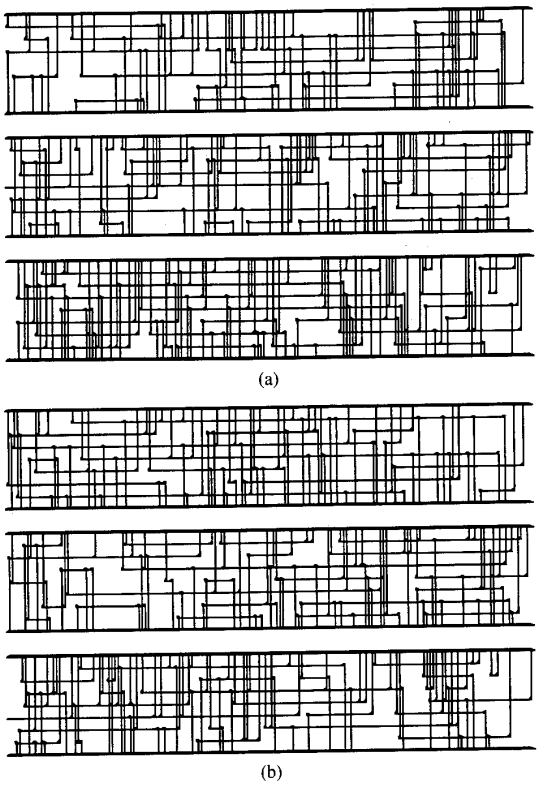


Fig. 12. Six-layer solutions for Deutsch's difficult example. (a) Solution 1. (b) Solution 2.

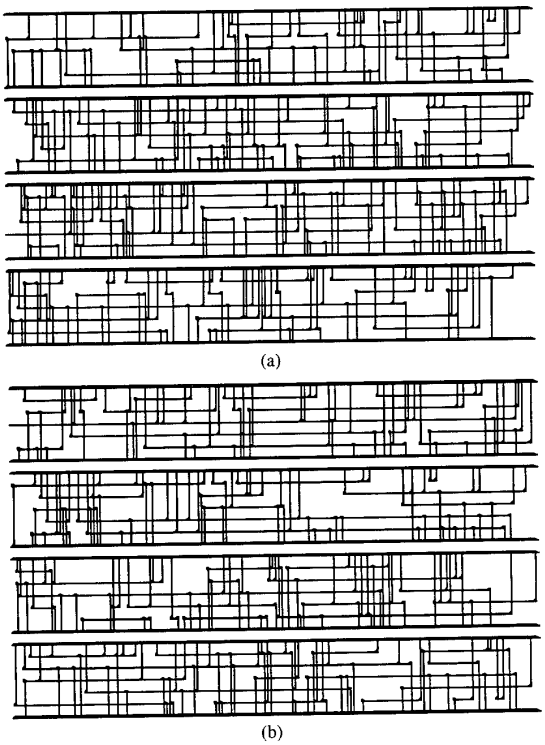
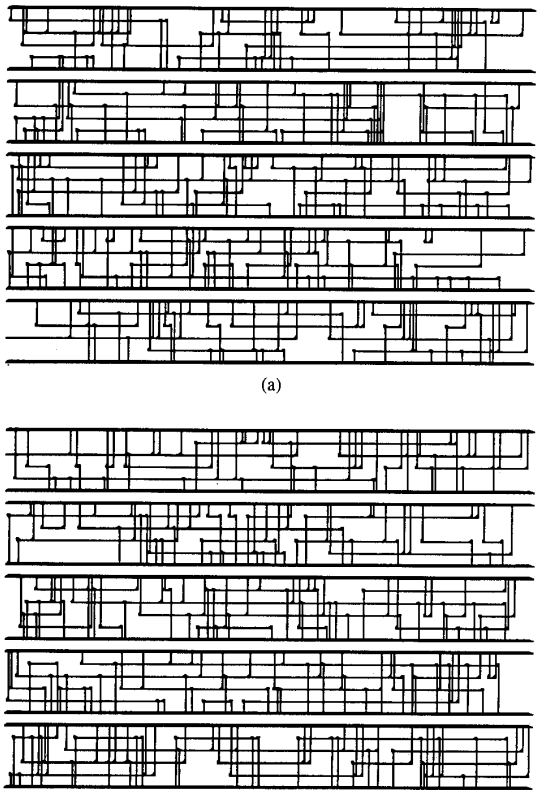


Fig. 13. Eight-layer solutions for Deutsch's difficult example. (a) Solution 1. (b) Solution 2.



(a) (b)

Fig. 14. Ten-layer solutions for Deutsch's difficult example. (a) Solution 1. (b) Solution 2.

TABLE III
TRACK NUMBERS IN 4-10-LAYER SOLUTIONS FOR DEUTSCH'S DIFFICULT EXAMPLE

Number of Layers	Number of Tracks
4	11
6	7
8	5
10	4

algorithm. They also show that the primary goal of finding near-optimum solutions in parallel processing is successfully achieved in terms of the computation time and the solution quality. The algorithm can be easily modified and extended to $3s$-layer channel routing problems in the $2s$HsV model. It might be interesting to investigate a parallel algorithm using doglegging.

REFERENCES

[1] A. Hashimoto and J. Stevens, "Wire routing by optimizing channel assignment with large apertures," in *Proc. 8th Design Automat. Workshop*, 1971, pp. 155-169.

[2] B. W. Kernighan, D. G. Schweikert, and G. Persky, "An optimum channel-routing algorithm for polycell layouts of integrated circuits," in *Proc. 10th Design Automat. Workshop*, 1973, pp. 50-59.

[3] D. N. Deutsch, "A dogleg channel router," in *Proc. 13th Design Automat. Conf.*, 1976, pp. 425-433.

[4] S. Sahni and A. Bhatt, "The complexity of design automation problems," in *Proc. 17th Design Automat. Conf.*, 1980, pp. 402-411.

[5] A. S. LaPaugh, "Algorithms for integrated circuits layout: An analytic approach," Ph.D. dissertation, M.I.T. Lab. Computer Science, 1980.

[6] D. Dolev *et al.*, "Optimal wiring between rectangles," in *Proc. 13th Ann. ACM Symp. Theory of Computing*, 1981, pp. 312-317.

[7] T. Yoshimura and E. S. Kuh, "Efficient algorithms for channel routing," *IEEE Trans. Computer-Aided Design*, vol. CAD-1, pp. 25-35, Jan. 1982.

[8] R. L. Rivest and C. M. Fiduccia, "A greedy channel router," in *Proc. 19th Design Automat. Conf.*, 1982, pp. 418-424.

[9] H. W. Leong and C. L. Liu, "A new channel routing problem," in *Proc. 20th Design Automat. Conf.*, 1983, pp. 584-590.

[10] M. Burstein and R. Pelavin, "Hierarchical channel router," in *Proc. 20th Design Automat. Conf.*, 1983, pp. 591-597.

[11] M. Burstein and R. Pelavin, "Hierarchical wire routing," *IEEE Trans. Computer-Aided Design*, vol. CAD-2, pp. 223-234, Oct. 1983.

[12] T. G. Szymanski, "Dogleg channel routing is NP-complete," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 31-41, Jan. 1985.

[13] J. Reed, A. Sangiovanni-Vincentelli, and M. Santomauro, "A new symbolic channel router: YACR2," *IEEE Trans. Computer-Aided Design*, vol. CAD-4, pp. 208-219, July 1985.

[14] H. W. Leong, D. F. Wong, and C. L. Liu, "A simulated-annealing channel router," in *Proc. ICCAD-85*, 1985, pp. 226-228.

[15] R. Joobbani and D. P. Siewiorek, "Weaver: A knowledge-based routing expert," in *Proc. 22nd Design Automat. Conf.*, 1985, pp. 266-272.

[16] D. N. Deutsch and P. Glick, "An over-the-cell router," in *Proc. 17th IEEE/ACM Design Automat. Conf.*, 1980, pp. 32-39.

[17] H. E. Krohn, "An over-cell gate array channel router," in *Proc. 20th Design Automat. Conf.*, 1983, pp. 665-670.

[18] Y. Shiraishi and J. Sakemi, "A permeation router," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 462-471, May 1987.

[19] G. Gudmundsson and S. Ntafos, "Channel routing with superterminals," in *Proc. 25th Allerton Conf. Computing, Control and Commun.*, 1987, pp. 375-376.

[20] J. Cong and C. L. Liu, "Over-the-cell channel routing," in *Proc. ICCAD-88*, 1988, pp. 80-83.

[21] J. Cong and C. L. Liu, "Over-the-cell channel routing," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 408-418, Apr. 1990.

[22] Y. K. Chen and M. L. Liu, "Three-layer channel routing," *IEEE Trans. Computer-Aided Design*, vol. CAD-3, pp. 156-163, Apr. 1984.

[23] J. Cong, D. F. Wong, and C. L. Liu, "A new approach to the three layer channel routing problem," in *Proc. ICCAD-87*, 1987, pp. 378-381.

[24] S. C. Chang and J. JáJá, "Parallel algorithms for channel routing in the knock-knee model," in *Proc. Int. Conf. Parallel Processing*, 1988, pp. 18-25.

[25] S. Krishnamurthy and J. JáJá, "Provably good parallel algorithms for channel routing of multiterminal nets," in *Proc. 2nd Symp. Frontiers Massively Parallel Comput.*, 1988, pp. 177-180.

[26] S. C. Chang and J. JáJá, "Parallel algorithms for river routing," in *Proc. Int. Conf. Parallel Processing*, 1988, pp. 9-13.

[27] S. C. Chang and J. JáJá, "Optimal mesh algorithms for VLSI routing," in *Proc. 2nd Symp. Frontiers Massively Parallel Comput.*, 1988, pp. 125-128.

[28] A. Iosupovici, "A class of array architectures for hardware grid routers," *IEEE Trans. Computer-Aided Design*, vol. CAD-5, pp. 245-255, Apr. 1986.

[29] J. Rose, "Parallel global routing for standard cells," *IEEE Trans. Computer-Aided Design*, vol. 9, pp. 1085-1095, Oct. 1990.

[30] R. J. Brouwer and P. Banerjee, "A parallel simulated annealing algorithm for channel routing on a hypercube multiprocessor," in *Proc. IEEE Int. Conf. Comput. Design*, 1988, pp. 4-7.

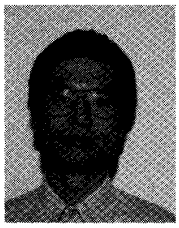
[31] Mosis user manual, supplements, 1990.

[32] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biol. Cybern.*, vol. 52, pp. 141-152, 1985.

[33] W. S. McCulloch and W. H. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bull. Math. Biophys.*, vol. 5, p. 115, 1943.

[34] Y. Takefuji and K. C. Lee, "A near-optimum parallel planarization algorithm," *Science*, vol. 245, pp. 1221-1223, Sept. 1989.

[35] Y. Takefuji and K. C. Lee, "Artificial neural networks for four-coloring map problems and K-colorability problems," *IEEE Trans. Circuits Syst.*, vol. 38, pp. 326–333, Mar. 1991.
[36] Y. Takefuji and K. C. Lee, "A parallel algorithm for tiling problems," *IEEE Trans. Neural Networks*, Vol. 1, pp. 143–145, Mar. 1990.
[37] Y. Takefuji, C. W. Lin, and K. C. Lee, "A parallel algorithm for estimating the secondary structure in ribonucleic acids," *Biol. Cybern.*, vol. 63, no. 5, pp. 337–340, 1990.
[38] Y. Takefuji, L. L. Chen, K. C. Lee, and J. Huffman, "Parallel algorithms for finding a near-maximum independent set of a circle graph," *IEEE Trans. Neural Networks*, vol. 1, pp. 263–267, Sept. 1990.
[39] Y. Takefuji and K. C. Lee, "A super parallel sorting algorithm based on neural networks," *IEEE Trans. Circuits Syst.*, vol. 37, pp. 1425–1429, Nov. 1990.
[40] Y. Takefuji and K. C. Lee, "An artificial hysteresis binary neuron: A model suppressing the oscillatory behaviours of neural dynamics," *Biol. Cybern.*, vol. 64, pp. 353–356, 1991.
[41] N. Funabiki and Y. Takefuji, "A parallel algorithm for spare allocation problems," *IEEE Trans. Reliability*, to be published.
[42] N. Funabiki and Y. Takefuji, "A parallel algorithm for solving the "Hip" games," *Neurocomputing*, to be published.



Nobuo Funabiki (S'90) received the B.S. degree in mathematical engineering and information physics from the University of Tokyo, Japan, in 1984. He is currently a graduate student of electrical engineering at Case Western Reserve University, Cleveland, OH, working toward the Ph.D. degree. His work experience includes a position with Sumitomo Metal Industries, Ltd, Japan. His research interests include neural network computing for solving VLSI routing problems, switching control problems, and other combinatorial optimization problems. He is a student member of the American Association for the Advancement of Science.

Yoshiyasu Takefuji (S'77–M'83) received the B.S., M.S., and Ph.D. degrees in electrical engineering from Keio University, Japan, in 1978, 1980, and 1983 respectively.

He is currently an Assistant Professor of Electrical Engineering at Case Western Reserve University, Cleveland, OH. Before joining Case Western, in 1988, he taught at the University of South Florida and the University of South Carolina. His current research interests focus on neural network parallel computing for solving real-world problems. He is also interested in VLSI applications and silicon architecture.

Dr. Takefuji received a National Science Foundation/Research Initiation Award in 1989 and is an NSF advisory panelist. A member of ACM, the International Neural Network Society, and the American Association for the Advancement of Science, he received the Information Processing Society of Japan's best paper award in 1980. He is a coauthor of two books (*Digital Circuits* and *Neural Network Computing*, in Japanese). His new book, *Neural Network Parallel Computing*, was published in 1992. Dr. Takefuji is an editor of the *Journal of Neural Network Computing* and an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS. He has published more than 80 technical papers.