Parallel Computing 19 (1993) 63–77
North-Holland

PARCO 719

A parallel multi-layer channel router on the HVH model

Nobuo Funabiki [1] and Yoshiyasu Takefuji [2]

[1] *System Engineering Division, Sumitomo Metal Industries, Ltd., Amagasaki 660, Japan*
[2] *Department of Electrical Engineering and Applied Physics, Case Western Reserve University, Cleveland, OH 44106, USA and Faculty of Environmental Information, Keio University, Fujisawa, Japan*

Received 29 August 1991
Revised 24 February , 20 April 1992

Abstract

Funabiki, N. and Y. Takefuji, A parallel multilayer channel router on the HVH model, Parallel Computing 19 (1993) 63–77.

With the advancement of the silicon technology multi-layered VLSI circuits and PCBs (printed circuit boards) have been widely used. Based on the neural network model this paper presents the first parallel algorithm for multi-layer channel routing problems on the HVH model which minimize wiring areas in VLSI circuits and PCBs. The algorithm requires $n \times m \times 2s$ processing elements for the n-net-m-track-$3s$-layer problem. The algorithm not only runs on a sequential machine but also on a parallel machine with maximally $n \times m \times 2s$ processors. The algorithm is verified by solving seven benchmark problems where it finds better solutions than the existing algorithms for 6–12-layer problems in nearly constant time on a parallel machine.

Keywords. Channel routing problems; HVH model; neural network; VLSI circuits; benchmark problems; simulation results.

1. Introduction

With the advancement of the silicon technology multi-layered VLSI circuits and PCBs (printed circuit boards) has been widely used. Algorithms for channel routing problems have been extensively studied in order to minimize the routing areas [1–10]. Particularly the algorithm for multi-layer channel routing problems on the HVH model [4] is very important for the manufacturing cost reduction. A channel consists of two parallel horizontal rows of points (terminals) which are placed at a regular interval. A net is a set of terminals to be interconnected through a routing path. The channel routing problem is not only to assign given nets on the channel without intersection but also to minimize the routing area. A routing path is composed of only two types of segments; horizontal segments and vertical segments. The horizontal segments are parallel to terminals and the vertical segments are perpendicular to them. Each segment is assigned on a different layer where a connection is made through a via. In order to secure wiring spaces a unit grid is superimposed on the channel where all terminals are located on grid points and all routing paths follow grid lines. A grid line for a horizontal segment and that for a vertical segment are called a track and a column respectively. Because the number of terminals is fixed in the problem, it is equivalent

Correspondence to: Y. Takefuji, System Engineering Division, Sumitomo Metal Industries, Ltd., Amagasaki 660, Japan.

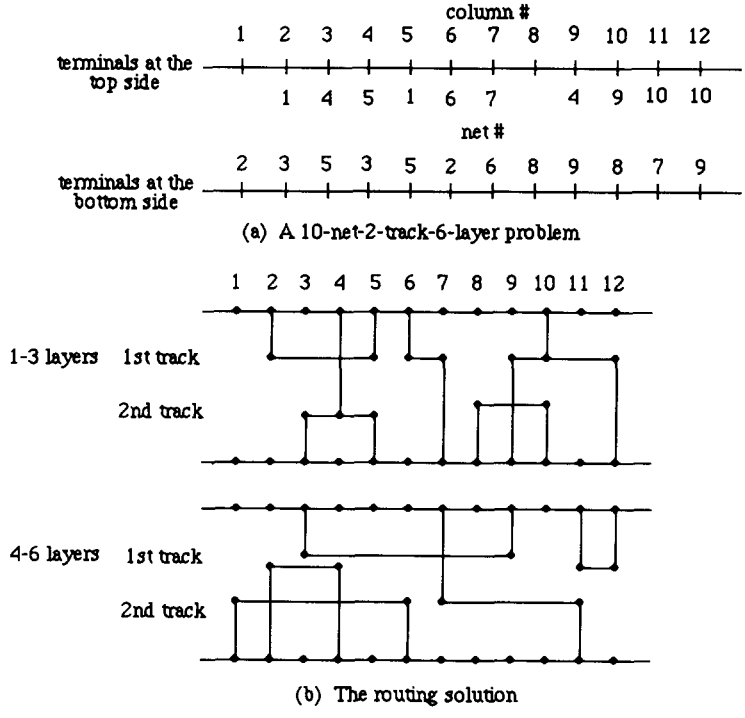

Fig. 1. A 10-net-2-track-6-layer channel routing problem on the HVH model and the solution.

to minimizing the number of tracks in the net assignment without violating constraints such that any two nets must be embedded neither on the same track nor column if they overlap there.

Typically there have been two types of channel routing models; the HV model and the HVH model. One unit in the HV model consists of two layers where one layer is used for horizontal segments (= a horizontal layer) and another is for vertical segments (= a vertical layer). One unit in the HVH model consists of three layers where two horizontal layers share one vertical layer [4]. Because horizontal segments usually require larger routing area than vertical segments, the HVH model can realize a smaller routing area than the HV model.

Although doglegs can further reduce routing areas [5], they requires more number of vias which not only degrade the system reliability but also increase the cost. As recommended by Deutsch [6] our algorithm does not use doglegs. LaPaugh showed NP-completeness of the no-dogleg problem on the HV model [7].

Figure 1 (a) shows a 10-net-2-track-6-layer problem on the HVH model. For example the net #1 connects two terminals #2 and #5 on the top side. Horizontal segments of nets are assigned on the 1st-, 3rd-, 4th-, and 6th-layer and vertical segments on the 2nd- and 5th-layer. The 1st and the 3rd horizontal layers share the 2nd vertical layer. *Figure 1 (b)* shows the routing solution.

Several sequential algorithms have been proposed on the HVH model. Chen et al. first introduced the HVH model and proposed a sequential 'merging' algorithm [4]. Bruell et al. proposed a sequential 'greedy' algorithm [8]. Braun et al. proposed a sequential 'Chameleon' algorithm [9]. Although Braun claimed that the algorithm found optimum solutions in up-to-six-layer benchmark problems, no details of their results have been disclosed. Cong et al. proposed an $O(w^3c^2)$ sequential algorithm where w is the number of tracks in the two-layer solution and c is the number of columns [10]. Their algorithm found optimum

solutions with a lot of doglegs. No parallel algorithm on the HVH model has been reported within our knowledge.

2. Neural network approach

This paper proposes the first parallel algorithm for $3s$-layer channel routing problems ($s = 2, 3, \ldots$) on the HVH model based on a three dimensional neural network model. Since Hopfield and Tank first introduced the neural network model for the traveling salesman problem [12], it has been extensively studied for solving NP-complete and optimization problems [13–17]. The neural network model in this paper is composed of $n \times m \times 2s$ simple processing elements. The processing element is also called a neuron because it performs the function of the simplified biological neuron model.

A processing element has an input U_{ijk} and an output V_{ijk} [11]. The change of U_{ijk} is given by partial derivatives of the Liapunov energy function $E(V_{111}, \ldots, V_{nm2s})$ with respect to V_{ijk} which is called a motion equation:

$$\frac{\mathrm{d}U_{ijk}}{\mathrm{d}t} = -\frac{\partial E(V_{111}, V_{112}, \ldots, V_{nm2s})}{\partial V_{ijk}}. \tag{1}$$

The energy function is defined by a non-negative function which represents the necessary and sufficient constraints in the problem. The energy function has the minimum value when the neural network is converged to a solution state. The goal of the neural network approach is to find a state of the neural network which minimizes the energy function. As shown in the Appendix, the motion equation is proven to force the neural network system to converge to the local minimum [16].

In our algorithm, it is necessary and sufficient to find a layer-track for the horizontal segment of each net among $m \times 2s$ candidates in the m-track-$3s$-layer problem. The vertical segments are then automatically assigned. A total of $n \times m \times 2s$ processing elements are required for the n-net-m-track-$3s$-layer problem. The nonzero output ($V_{ijk} = 1$) of the ijkth processing element represents that the ith net is assigned to the jth track on the kth layer. The zero output ($V_{ijk} = 0$) represents no assignment there. *Figure 2* shows the neural network

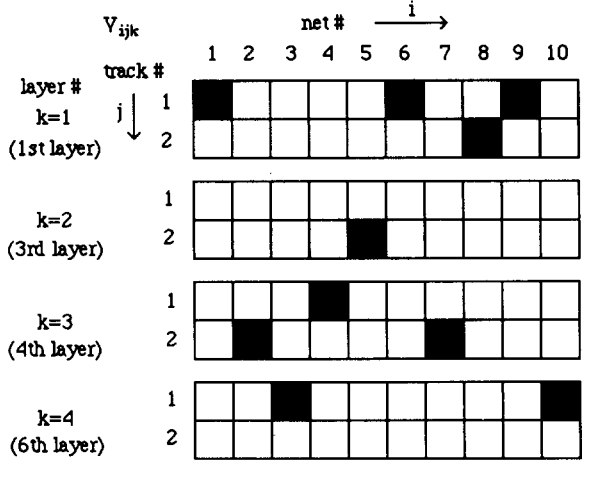

Fig. 2. Neural network representation for the problem in *Fig. 1*.

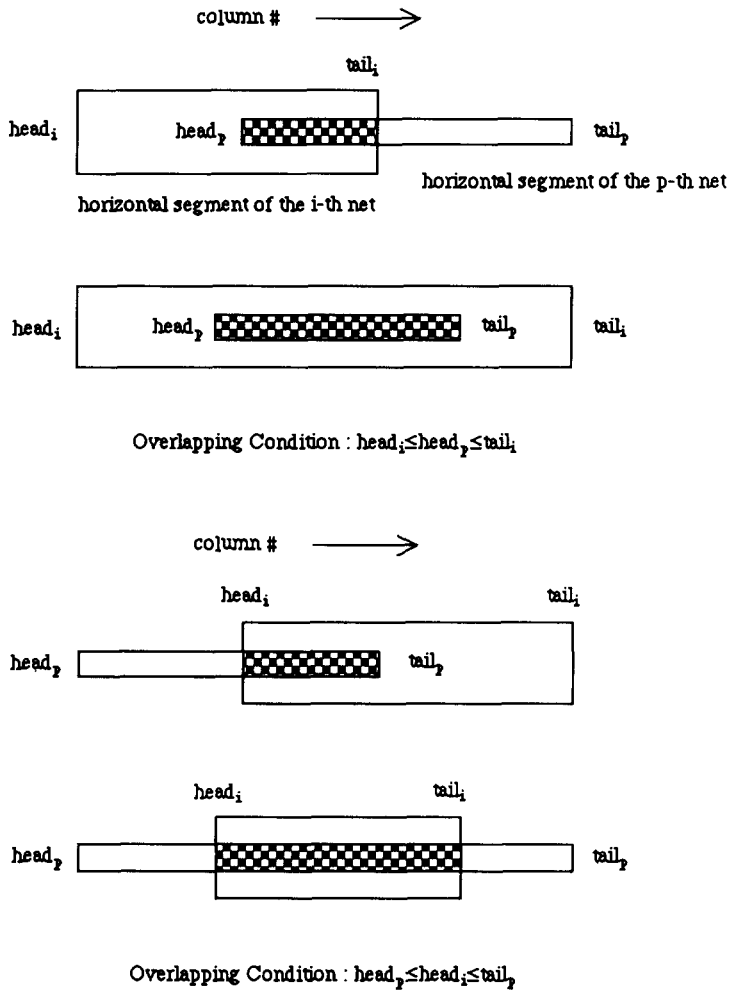

Fig. 3. Overlapping conditions for horizontal segments.

representation for the problem in *Fig. 1* and the solution state corresponding to *Fig. 1 (b)* where a black square indicates the nonzero output.

We introduce the energy function E for the n-net-m-track-3s-layer channel routing problem on the HVH mode. The energy function must have the terms representing all the constraints in the problem. The first constraint is that the horizontal segment of every net must be assigned in a track of a horizontal layer among $m \times 2s$ candidates in the m-track-3s-layer problem. In other words, one and only one processing element among $2ms$ processing elements for a net must have nonzero output for the net assignment. It is given by:

$$E_1 = \sum_{i=1}^{n} \left(\sum_{q=1}^{m} \sum_{r=1}^{2s} V_{iqr} - 1 \right)^2 . \tag{2}$$

E_1 is zero if and only if one processing element has nonzero output for every net.

The second constraint is that neither horizontal segments nor vertical segments of any two nets must be overlapped to each other. *Figures 3* and *4* show the overlapping conditions for

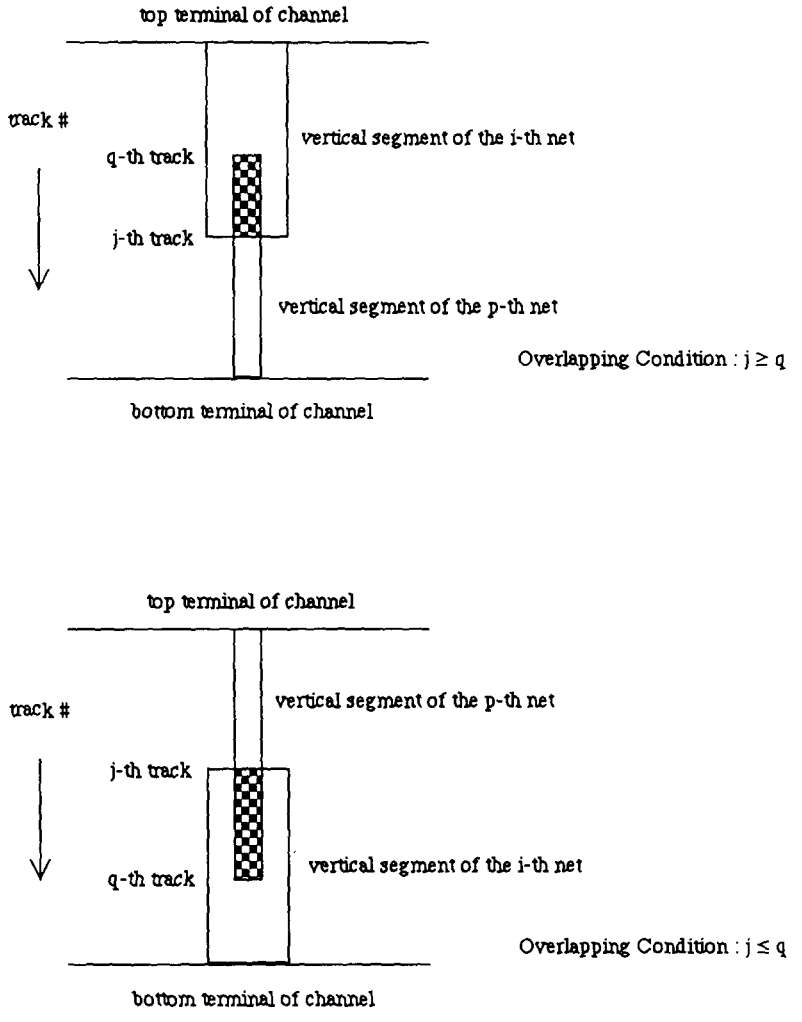

Fig. 4. Overlapping conditions for vertical segments.

the horizontal segments and for the vertical segments respectively. This constraint is given by:

$$E_2 = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{2s}\left(\sum_{\substack{p=1\\p\neq i}}^{n} f(\text{head}_i, \text{head}_p, \text{tail}_i)V_{pjk} + \sum_{\substack{p=1\\p\neq i}}^{n} f(\text{head}_p, \text{head}_i, \text{tail}_p)V_{pjk}\right)V_{ijk}$$

$$+ \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{2s}\left(\sum_{\substack{p=1\\p\neq i}}^{n} T_{ip}\sum_{q=1}^{j}\sum_{r=g(k)}^{g(k)+1} V_{pqr} + \sum_{\substack{p=1\\p\neq i}}^{n} B_{ip}\sum_{q=j}^{m}\sum_{r=g(k)}^{g(k)+1} V_{pqr}\right)V_{ijk}. \qquad (3)$$

The function $f(x, y, z)$ in Eq. (3) is 1 if $x \le y \le z$, 0 otherwise. head_i and tail_i indicate the left-most and right-most columns of the ith net respectively. As shown in *Fig. 3*, horizontal

segments of the ith net and the pth net are overlapped if $f(\text{head}_i, \text{head}_p, \text{tail}_i)$ ($f(\text{head}_p, \text{head}_i, \text{tail}_p)$) is 1 and they are assigned in the same track of the same layer. Therefore the first term in E_2 becomes zero if and only if horizontal segments of any two nets are not overlapped to each other.

$T_{ip}(B_{ip})$ in Eq. (3) is 1 if the ith net has a top-side (bottom-side) terminal on the column where the pth net has a bottom-side (top-side) terminal. The function $g(k)$ indicates the first one of two horizontal layers which share one vertical layer. $g(k)$ is k if k is odd, $k-1$ otherwise. As shown in *Fig. 4*, vertical segments of the ith net and the pth net are overlapped if $T_{ip}(B_{ip})$ is 1, $j \geq q (j \leq q)$, and the horizontal segments are assigned in layers sharing the same vertical layer. Therefore the second term in E_2 becomes zero if and only if vertical segments of any two nets are not overlapped to each other.

The total energy function E is given by:

$$
\begin{aligned}
E &= \frac{\mathrm{A}}{2} E_1 + \mathrm{B} E_2 \\
&= \frac{\mathrm{A}}{2} \sum_{i=1}^{n} \left(\sum_{q=1}^{m} \sum_{r=1}^{2s} V_{iqr} - 1 \right)^2 \\
&\quad + \mathrm{B} \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{2s} \left(\sum_{\substack{p=1 \\ p \neq i}}^{n} f(\text{head}_i, \text{head}_p, \text{tail}_i) V_{pjk} \right. \\
&\qquad \left. + \sum_{\substack{p=1 \\ p \neq i}}^{n} f(\text{head}_p, \text{head}_i, \text{tail}_p) V_{pjk} \right) V_{ijk} \\
&\quad + \mathrm{B} \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{2s} \left(\sum_{\substack{p=1 \\ p \neq i}}^{n} T_{ip} \sum_{q=1}^{j} \sum_{r=g(k)}^{g(k)+1} V_{pqr} + \sum_{\substack{p=1 \\ p \neq i}}^{n} \mathrm{B}_{ip} \sum_{q=j}^{m} \sum_{r=g(k)}^{g(k)+1} V_{pqr} \right) V_{ijk}, \qquad (4)
\end{aligned}
$$

where A and B are constant coefficients.

From Eqs. (1) and (4) the motion equation of the ijkth processing element for the n-net-m-track-$3s$-layer problem is given by:

$$
\begin{aligned}
\frac{\mathrm{d}U_{ijk}}{\mathrm{d}t} &= -\mathrm{A} \left(\sum_{q=1}^{m} \sum_{r=1}^{2s} V_{iqr} - 1 \right) \\
&\quad - \mathrm{B} \left(\sum_{\substack{p=1 \\ p \neq i}}^{n} f(\text{head}_i, \text{head}_p, \text{tail}_i) V_{pjk} + \sum_{\substack{p=1 \\ p \neq i}}^{n} f(\text{head}_p, \text{head}_i, \text{tail}_p) V_{pjk} \right) \\
&\quad - \mathrm{B} \left(\sum_{\substack{p=1 \\ p \neq i}}^{n} T_{ip} \sum_{q=1}^{j} \sum_{r=g(k)}^{g(k)+1} V_{pqr} + \sum_{\substack{p=1 \\ p \neq i}}^{n} \mathrm{B}_{ip} \sum_{q=j}^{m} \sum_{r=g(k)}^{g(k)+1} V_{pqr} \right). \qquad (5)
\end{aligned}
$$

The A-term forces one and only one processing element among $m \times 2s$ candidates for the ith net to have nonzero output where the ith net is assigned. The B-terms discourage the ijkth processing element to have nonzero output if other nets overlap the ith net.

3. Four heuristics for the global minimum convergence

In the neural network model only the local minimum convergence is guaranteed, although we must consider the global minimum convergence. In order to increase the frequency of the global minimum convergence, the following four heuristics have been empirically introduced [14]:

(1) *The hill-climbing heuristic*: the following term is added to the motion equation in Eq. (5):

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

where $h(x)$ is 1 if $x=0$, 0 otherwise. C is a constant coefficient. The C-term provides hill-climbing which allows the state of the system to escape from the local minimum and empirically increases the frequency to converge to the global minimum. The C-term encourages the ijkth processing element to have nonzero output if no processing elements for the ith net have nonzero output.

(2) *The omega function heuristic*: two forms of the B-term are periodically used in the motion equation:

$$-B\left(\sum_{\substack{p=1\\p\neq i}}^{n} f(\text{head}_i, \text{head}_p, \text{tail}_i)V_{pjk} + \sum_{\substack{p=1\\p\neq i}}^{n} f(\text{head}_p, \text{head}_i, \text{tail}_p)V_{pjk}\right)V_{ijk}$$

$$-B\left(\sum_{\substack{p=1\\p\neq i}}^{n} T_{ip}\sum_{q=1}^{j}\sum_{r=g(k)}^{g(k)+1}V_{pqr} + \sum_{\substack{p=1\\p\neq i}}^{n} B_{ip}\sum_{q=j}^{m}\sum_{r=g(k)}^{g(k)+1}V_{pqr}\right)V_{ijk} \text{ if } (t \bmod T) < \omega. \tag{7}$$

$$-B\left(\sum_{\substack{p=1\\p\neq i}}^{n} f(\text{head}_i, \text{head}_p, \text{tail}_i)V_{pjk} + \sum_{\substack{p=1\\p\neq i}}^{n} f(\text{head}_p, \text{head}_i, \text{tail}_p)V_{pjk}\right)$$

$$-B\left(\sum_{\substack{p=1\\p\neq i}}^{n} T_{ip}\sum_{q=1}^{j}\sum_{r=g(k)}^{g(k)+1}V_{pqr} + \sum_{\substack{p=1\\p\neq i}}^{n} B_{ip}\sum_{q=j}^{m}\sum_{r=g(k)}^{g(k)+1}V_{pqr}\right) \text{otherwise}, \tag{8}$$

where t is the number of iteration steps, and T and ω are constant parameters.

(3) *The input saturation heuristic*: the input of the processing element is confined between two values:

$$U_{ijk} = U\_\max \quad \text{if } U_{ijk} > U\_\max$$

$$U_{ijk} = U\_\min \quad \text{if } U_{ijk} < U\_\min, \tag{9}$$

where $U\_\max$ and $U\_\min$ are constant upper and lower bounds of the input value U_{ijk} respectively. The omega function heuristic and the input saturation heuristic make a local minimum shallower so that the state of the system can easily escape from it.

(4) *The modified McCulloch-Pitts neuron model*:

$$V_{ijk} = 1 \quad \text{if } U_{ijk} > 0 \text{ and } U_{ijk} = \max\{U_{iqr}\} \text{ for } q = 1,\ldots,m \quad \text{and } r = 1,\ldots,2s = 0$$

$$\text{otherwise.} \tag{10}$$

It has empirically been shown to be superior to the McCulloch-Pitts neuron model [11] and the sigmoid neuron model [12] in the channel routing problems on the HVH model. *Table 1*

Table 1
Performance comparisons of three neuron models

	Modified McCulloch-Pitts neuron model		McCulloch-Pitts neuron model		Sigmoid neuron model	
	A	B	A	B	A	B
Example 1 (6 layers)	116.4	14	325.0	4	186.5	6
Deutsch (6 layers)	141.0	16	313.7	6	–	0

A: average number of iteration steps to solutions
B: convergence frequency.

shows the performance comparisons of three neuron models in two problems [3] where 50 simulation runs were performed for each problem.

4. Parallel algorithm

The following procedure describes the parallel algorithm for the n-net-m-track-$3s$-layer channel routing problem on the HVH model. The motion equation is solved by the first order Euler method.

[step0] Set $t = 0$, A = B = 1, C = 5, $U\_\max = 20$, $U\_\min = -20$ and $T\_\max = 500$.
[step1] The initial values of $U_{ijk}(t)$ are uniformly randomized between 0 and $U\_\min$ for $i = 1,\ldots,n$, $j = 1,\ldots,m$, and $k = 1,\ldots,2s$, and the initial values of $V_{ijk}(t)$ are assigned to 0 for $i = 1,\ldots,n$, $j = 1,\ldots,m$, and $k = 1,\ldots,2s$.
[step2] Compute the change of the input $\Delta U_{ijk}(t)$ by the motion equation in Eq. (5) with two heuristics in Eqs. (6)–(8).
if (t mod 10) < 5 then

$$\begin{aligned}\Delta U_{ijk}(t) = &-\mathrm{A}\left(\sum_{q=1}^{m}\sum_{r=1}^{2s} V_{iqr}(t) - 1\right)\\ &-\mathrm{B}\left(\sum_{\substack{p=1\\p\neq i}}^{n} f(\mathrm{head}_i, \mathrm{head}_p, \mathrm{tail}_i)V_{pjk}(t)\right.\\ &\qquad\left.+\sum_{\substack{p=1\\p\neq i}}^{n} f(\mathrm{head}_p, \mathrm{head}_i, \mathrm{tail}_p)V_{pjk}(t)\right)V_{ijk}(t)\\ &-\mathrm{B}\left(\sum_{\substack{p=1\\p\neq i}}^{n} T_{ip}\sum_{q=1}^{j}\sum_{r=g(k)}^{g(k)+1} V_{pqr}(t) + \sum_{\substack{p=1\\p\neq i}}^{n} \mathrm{B}_{ip}\sum_{q=j}^{m}\sum_{r=g(k)}^{g(k)+1} V_{pqr}(t)\right)V_{ijk}(t)\\ &+\mathrm{C}h\left(\sum_{q=1}^{m}\sum_{r=1}^{2s} V_{iqr}(t)\right)\end{aligned} \tag{11}$$

else

$$\Delta U_{ijk}(t) = -\mathrm{A}\left(\sum_{q=1}^{m}\sum_{r=1}^{2s} V_{iqr}(t) - 1\right)$$
$$-\mathrm{B}\left(\sum_{\substack{p=1\\p\neq i}}^{n} f(\mathrm{head}_i, \mathrm{head}_p, \mathrm{tail}_i)V_{pjk}(t)\right.$$
$$\left.+\sum_{\substack{p=1\\p\neq i}}^{n} f(\mathrm{head}_p, \mathrm{head}_i, \mathrm{tail}_p)V_{pjk}(t)\right)$$
$$-\mathrm{B}\left(\sum_{\substack{p=1\\p\neq i}}^{n} T_{ip}\sum_{q=1}^{j}\sum_{r=g(k)}^{g(k)+1} V_{pqr}(t) + \sum_{\substack{p=1\\p\neq i}}^{n} B_{ip}\sum_{q=j}^{m}\sum_{r=g(k)}^{g(k)+1} V_{pqr}(t)\right)$$
$$+Ch\left(\sum_{q=1}^{m}\sum_{r=1}^{2s} V_{iqr}(t)\right). \tag{12}$$

[step3] Update the input $U_{ijk}(t+1)$.

$$U_{ijk}(t+1) = U_{ijk}(t) + \Delta U_{ijk}(t). \tag{13}$$

[step4] Use the input saturation heuristic in Eq. (9).

$$\text{If } U_{ijk}(t+1) > U\_\max \text{ then } U_{ijk}(t+1) = U\_\max.$$
$$\text{If } U_{ijk}(t+1) < U\_\min \text{ then } U_{ijk}(t+1) = U\_\min. \tag{14}$$

[step5] Update the output $V_{ijk}(t+1)$ based on the modified McCulloch-Pitts neuron model in Eq. (10).

$$V_{ijk}(t+1) = 1 \text{ if } U_{ijk}(t+1) > 0 \text{ and } U_{ijk}(t+1) = \max\{U_{iqr}(t+1)\}$$
$$\text{for } q = 1,\ldots,m \text{ and } r = 1,\ldots,2s.$$
$$= 0 \text{ otherwise.} \tag{15}$$

[step6] If $V_{ijk}(t) = 1$ and $\Delta U_{ijk}(t) = 0$ for $i = 1,\ldots,n$, $^{\exists}j \in \{1,\ldots,m\}$, and $^{\exists}k \in \{1,\ldots,2s\}$ or $t = T\_\max$ then terminate this procedure else increment t by 1 and and goto Step 2.

The set of parameters in [step0] has been empirically chosen by our existing algorithms [13–17]. Empirically we divide all given nets in a problem into two groups; a longer net group and a shorter net group. The longer net group consists of nets whose net width (= the distance between the left-most column and right-most column) is more than 30% of the channel width. The shorter net group consists of the remaining nets. First the longer net group is assigned in the channel by our algorithm, then the shorter net group is assigned, because it is impossible to embed longer nets after many nets are fixed in the channel.

Based on the proposed algorithm, the following procedure/program is used in our simulations.

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 $2s$;
...

```
/*** Main Program ***/
while (a set of conflicts is not empty) do
begin
...
/*** Updating the values of the inputs ***/
   for i := 1 to n
   for j := 1 to m
   for k := 1 to 2s
      U_ijk := U_ijk + ΔU_ijk;
/*** End of the first loop ***/
...
/*** Updating the values of the outputs ***/
for i := 1 to n
for j := i to m
for k := 1 to 2s
```

$$V_{ijk} := 1 \text{ if } U_{ijk} > 0 \text{ and } U_{ijk} = \max\{U_{iqr}\} \text{ for } q = 1,\ldots,m \text{ and } r = 1,\ldots,2s$$

$$:= 0 \text{ otherwise.} \tag{10}$$

```
/*** End of the second loop ***/
...
end;
/*** Main Program end ***/
```

Table 2
Specifications of simulated problems and simulation results

Problem #	Number of nets n	Number of tracks in our solutions m	Number of tracks by Chen and Liu	Ave. number of iteration steps to solutions	Convergence frequency to solutions
Example 1 (6 layers)	21	3	4 (7)	91.7	28%
Example 3a (6 layers)	45	4	4 (8)	116.5	21%
Example 3b (6 layers)	47	5	5 (10)	64.9	62%
Example 3c (6 layers)	54	5	5 (9)	108.9	46%
Example 4b (6 layers)	55	5	7 (13)	80.6	75%
Example 5 (6 layers)	61	5	5 (10)	89.1	59%
Deutsch (6 layers)	72	6	12 (23)	128.1	30%
Deutsch (9 layers)	72	4	8 (23)	85.3	80%
Deutsch (12 layers)	72	3	6 (23)	62.4	95%

() indicates the number of tracks in 3-layer solutions shown by Chen and Liu

5. Simulation results and discussion

The simulator has been developed on Macintosh SE/30 and IIfx where the main program has less than 100 steps in Turbo Pascal. Seven benchmark problems in [3] were examined. *Table 2* shows the specifications of the benchmark problems and the comparisons of the

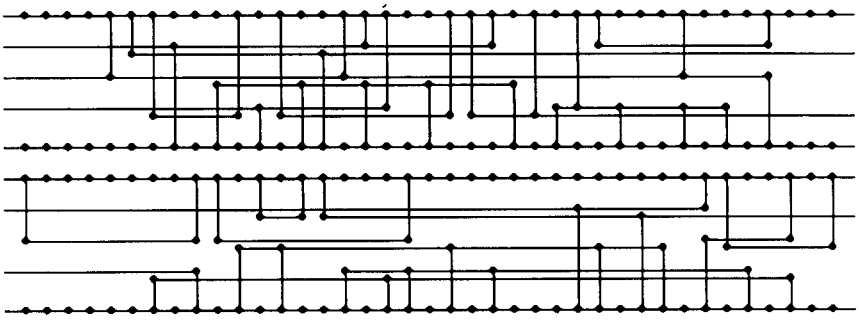

Fig. 5. A solution for Example 1.

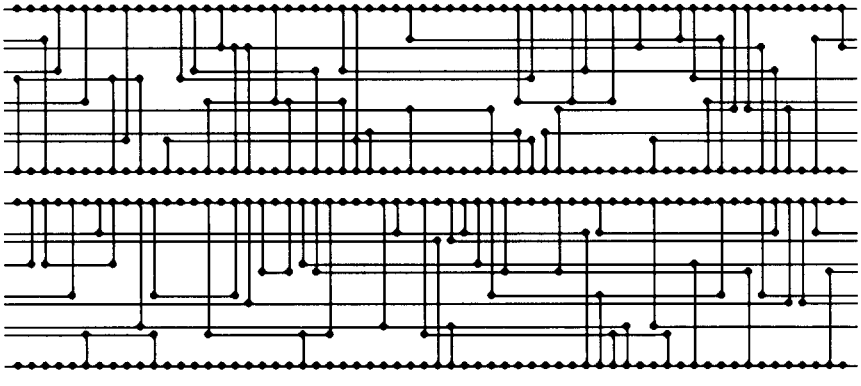

Fig. 6. A solution for Example 3(a).

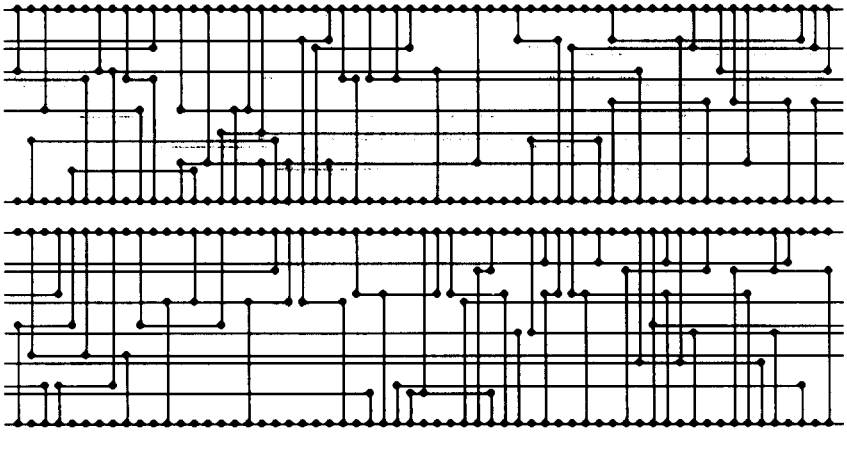

Fig. 7. A solution for Example 3(b).

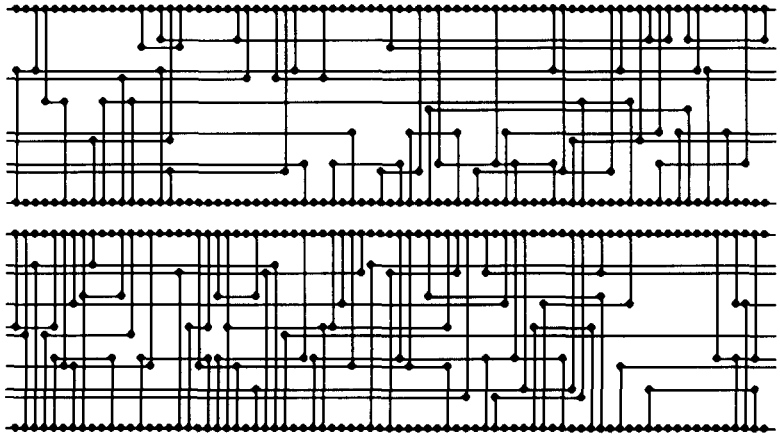

Fig. 8. A solution for Example 3(c).

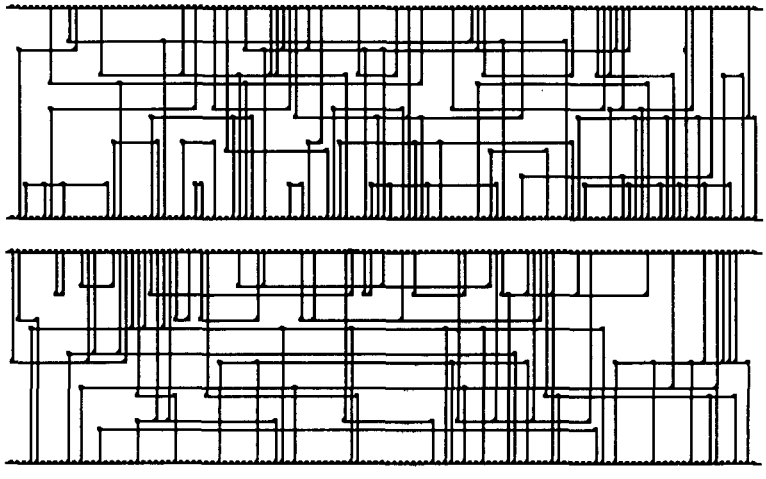

Fig. 9. A solution for Example 4(b).

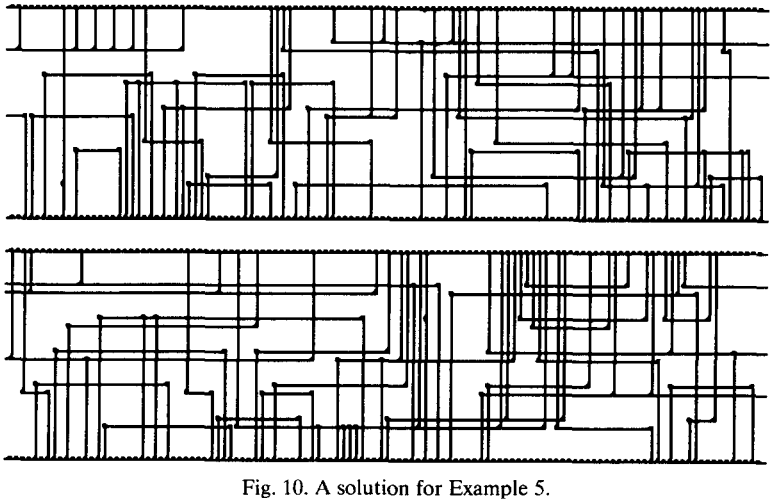

Fig. 10. A solution for Example 5.

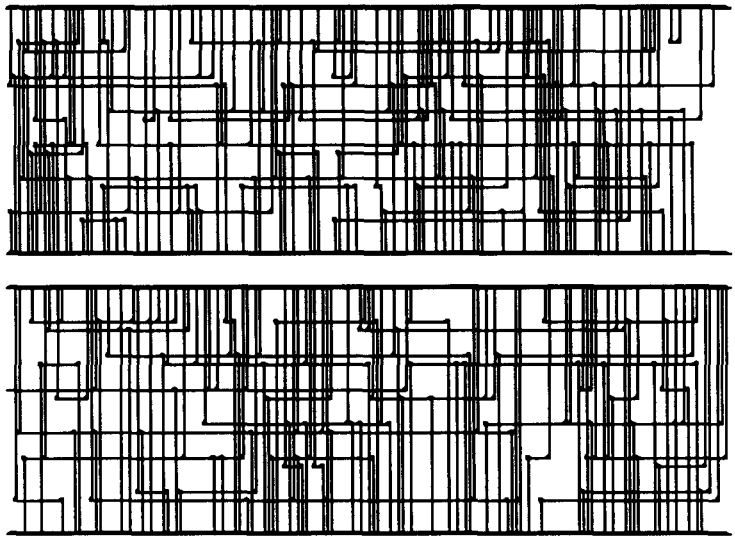

Fig. 11. A 6-layer solution for Deutsch's difficult example.

solution quality between our solutions and the best existing 3-layer-no-dogleg solutions [4]. Note that the existing solutions are adjusted for 6-, 9-, or 12-layer solutions from 3-layer solutions. Our algorithm found the optimum solutions for all benchmark problems except for the 6-layer Deutsch's difficult example where our algorithm requires one more track than the optimum solution. *Figures 5–11* show our 6-layer solutions for the seven benchmark problems respectively. *Figures 12* and *13* show our 9- and 12-layer solutions for Deutsch's difficult example respectively. *Table 2* summarizes the average number of iteration steps and the frequency to converge to solutions where 100 simulation runs were performed from different initial values of $U_{ijk}(t)$. The simulator found different solutions from different initial values of $U_{ijk}(t)$. *Figure 14* shows the relationship between the frequency and the number of iteration

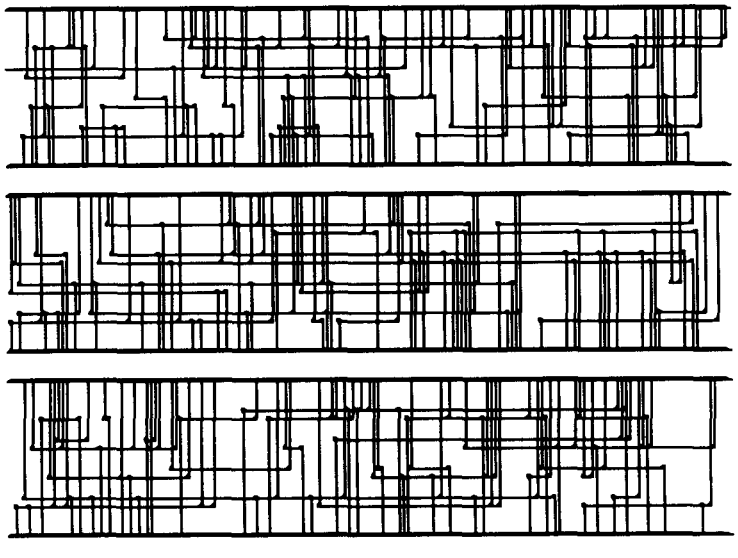

Fig. 12. A 9-layer solution for Deutsch's difficult example.

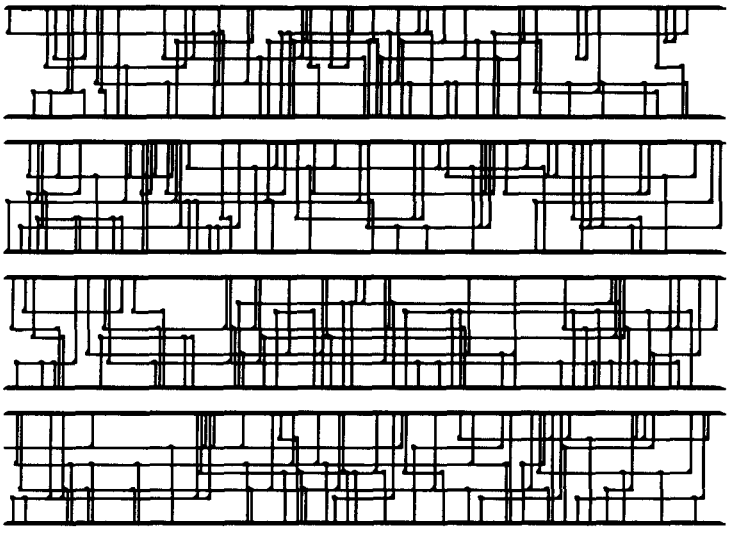

Fig. 13. A 12-layer solution for Deutsch's difficult example.

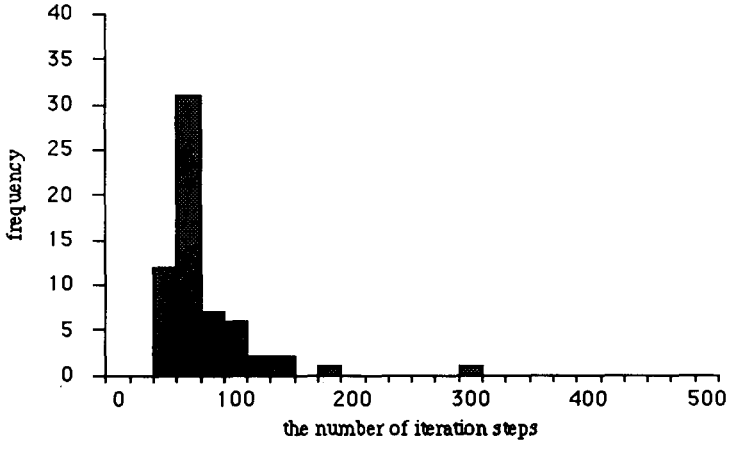

Fig. 14. The relationship between the number of iteration steps to converge to solutions and the frequency in the Example 3(b).

steps to converge to solutions in Example 3 (b). The proposed algorithm finds the optimum or near-optimum solutions in seven benchmark problems in nearly constant time with $n \times m \times 2s$ processors for n-net-m-track-$3s$-layer problems. The simulation results show the efficiency of the neural network approach in channel routing problems on the HVH model.

6. Conclusion

This paper presents a parallel algorithm for multi-layer channel routing problems on the HVH model. The algorithm is based on the neural network model composed of $n \times m \times 2s$ processing elements for the n-net-m-track-$3s$-layer problem. The algorithm was verified by seven benchmark problems where the algorithm found the optimum solutions or the near-optimum solutions in nearly constant time with $n \times m \times 2s$ processors.

Appendix

Theorem. *The system always satisfies* $(dE/dt) \leq 0$ *under two conditions such as* $(dU_i/dt) = -(\partial E/\partial V_i)$ *(condition 1) and* $V_i = f(U_i)$ *(condition 2) where E is the computational Liapunov energy function and* $f(U_i)$ *is a nondecreasing function.*

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

$$\frac{dE}{dt} = \sum_i \frac{dV_i}{dt}\frac{\partial E}{\partial V_i} = \sum_i \frac{dU_i}{dt}\frac{dV_i}{dU_i}\frac{\partial E}{\partial V_i}$$

$$= -\sum_i \left(\frac{dU_i}{dt}\right)^2 \frac{dV_i}{dU_i} \text{ where } \frac{\partial E}{\partial V_i} \text{ is replaced by } -\frac{dU_i}{dt} \text{ (condition 1)}$$

$$\leq 0 \text{ where } \frac{dV_i}{dU_i} \geq 0 \text{ (condition 2).} \quad \square$$

References

[1] A. Hashimoto and J. Stevens, Wire routing by optimizing channel assignment with large apertures, in: *Proc. 8th Design Automation Workshop* (1971) 155–169.

[2] D.N. Deutsch, A dogleg channel router, in: *Proc. 13th Design Automation Conf.* (1976) 425–433.

[3] T. Yoshimura and E.S. Kuh, Efficient algorithms for channel routing, *IEEE Trans. CAD* CAD-1 (1) (Jan. 1982) 25–35.

[4] Y.K. Chen and M.L. Liu, Three-layer channel routing, *IEEE Trans. CAD* CAD-3 (2) (Apr. 1984) 156–163.

[5] T.G. Szymanski, Dogleg channel routing is NP-complete, *IEEE Trans. CAD* CAD-4 (1) (Jan. 1985) 31–41.

[6] D.N. Deutsch, Two new and more difficult channel routing problems, *IEEE Trans. CAD* 8 (4) (April 1989) 448.

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

[8] P. Bruell and P. Sun, A greedy three layer channel router, in: *Proc. ICCAD-85* (1985) 298–300.

[9] D. Braun et al., Techniques for multilayer channel routing, *IEEE Trans. CAD* 7 (6) (June 1988) 698–712.

[10] J. Cong, D.F. Wong and C.L. Liu, A new approach to three- or four-layer channel routing, *IEEE Trans. CAD* 7 (10) (Oct. 1988) 1094–1104.

[11] W.S. McCulloch and W.H. Pitts, A logical calculus of ideas immanent in nervous activity, *Bull. Math. Biophys.* 5 (1943) 115–133.

[12] J.J. Hopfield and D.W. Tank, Neural computation of decisions in optimization problems, *Biol. Cybernet.* 52 (1985) 141–152.

[13] Y. Takefuji and K.C. Lee, A near-optimum parallel planarization algorithm, *Science* 245 (Sep. 1989) 1221–1223.

[14] Y. Takefuji and K.C. Lee, A parallel algorithm for tiling problems, *IEEE Trans. Neural Networks* 1 (1) (Mar. 1990) 143–145.

[15] 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* 1 (3) (Sep. 1990) 263–267.

[16] Y. Takefuji and K.C. Lee, Artificial neural networks for four-coloring map problems and K-colorability problems, *IEEE Trans. Circuits Syst.* 38 (3) (Mar. 1991) 326–333.

[17] N. Funabiki and Y. Takefuji, A parallel algorithm for spare allocation problems, *IEEE Trans. Reliability* 40 (3) (Aug. 1991) 338–346.