ELSEVIER

Neurocomputing 8 (1995) 141–156

NEUROCOMPUTING

Neural network parallel computing for multi-layer channel routing problems

Kyotaro Suzuki [a,*], Hideharu Amano [a], Yoshiyasu Takefuji [b]

[a] *Department of Computer Science, Keio University, Japan*
[b] *Department of Electrical Engineering and Applied Physics, Case Western University, Cleveland, OH 44106, USA*

Received 6 April 1993; accepted 1 February 1994

Abstract

Multi-layer channel routing is one of cumbersome jobs in automatic layout design of VLSI chips and PCBs. As VLSI chips have been used in every field of electrical engineering, it becomes more important to reduce the layout design time. With the advancement of the VLSI technology, four-layer problems can be treated and the algorithms for more than four-layer problems will be demanded in the near future. Proposed algorithm can treat $2 \times n$-layer problems in parallel. In this paper, the algorithm is introduced and implemented on a multiprocessor system. By minimizing the communication overhead and load unbalance between processors, the performance with 8 processors is improved by between 6 and 6.5 times compared with the sequential version.

Keywords: VLSI layout, Channel routing, Hopfield neural network, Multiprocessor

1. Introduction

Multi-layer channel routing problems are proved to be NP-complete and one of cumber-some jobs in automatic layout design of VLSI chips and PCBs.

The problem is not only to route the given interconnections between terminals in two parallel horizontal rows on the multi-layer channel without overlapping each other, but also to minimize the channel area. Many sequential algorithms for the two-layer channel routing problems have been proposed [1–14].

Each set of terminals must be interconnected through a routing path which consists of only one horizontal segment and several vertical segments. The horizontal segments and the vertical segments must be routed on different layers respec-

SSDI 0925-2312(94)00014-J

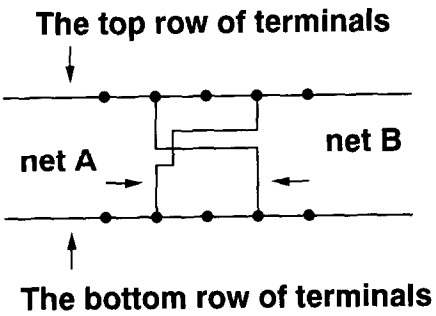

Fig. 1. Cyclic problem.

tively. In some sequential algorithms [3,7–14], doglegging has been introduced, where only a horizontal segment of a routing path is split into two or more than two horizontal segments. Doglegging is effective to reduce the number of tracks of the channel and to solve the cyclic conflict. The cyclic conflict occurs when net A has a top terminal at the same column where net B has a bottom terminal while net A has a bottom terminal at the same column where net B has the top terminal (Fig. 1). However, it 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. Therefore, doglegging is not introduced to the proposed algorithm in this paper.

In order to reduce the channel area further, sequential algorithms for two-layer-and-over-the-cell channel routing problems [15–20], and three-layer channel routing problems [21,22] have been proposed. The over-the-cell algorithms use not only the channel area but also the area over the cells for interconnections. Although these algorithms can find better solutions, they are still based on two-layer routing technique, so that it is not easy to apply them for more than three-layer problems.

As VLSI chips have been used in every field of electrical engineering, it becomes more important to reduce the layout design time. Several parallel channel routing algorithms have been proposed and implemented on parallel processor systems [23–31].

The algorithm for more than four-layer channel problems is demanded in the near future. Proposed algorithm [32,33] can solve more than four-layer problems flexibly and it is suited for parallel computing. The algorithm has been implemented on a multiprocessor system and the performance is evaluated.

In Section 2, the multi-layer channel routing problem is introduced with some definitions. The problem is represented by the neural network model, and the algorithm is introduced in Section 3. In Section 5, the algorithm is implemented on the target multi-processor system 'ATTEMPT-0' introduced in Section 4. The experimental results are shown and evaluated in Section 6. Finally, conclusions are described in Section 7.

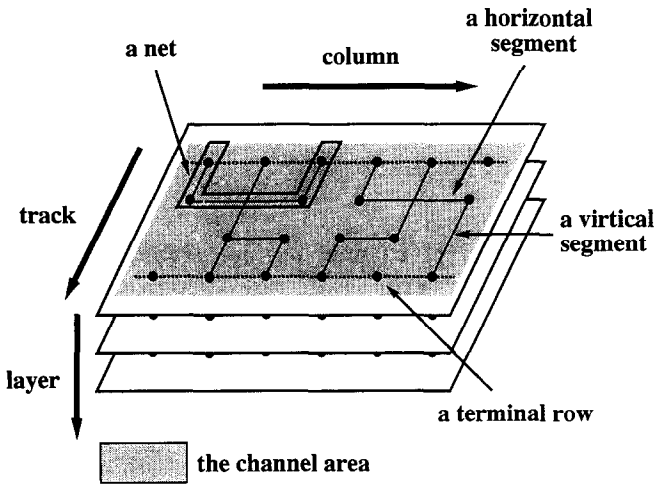

Fig. 2. The channel routing problem.

2. Definitions

On a chip for channel routing, all standard cells have the same height and stand in rows. A channel consists of two parallel horizontal rows of points which are called terminals and the area between them in Fig. 2. Each terminal is placed at regular intervals and can be distinguished from each other by the column and the end (top end or bottom end) at which the terminal is placed.

A net consists of a set of terminals at one or both ends that must be interconnected through some routing paths. A path consists of only a horizontal segment parallel to the terminal rows and several vertical segments. The horizontal segments an the vertical segments must be assigned on different layers respectively.

Some pairs of two layers are used for the multi-layer channel routing problem. Each net is routed on a pair of two layers. One is for only a horizontal segment, and the other is for the vertical segments. The connections between those segments are made through via holes.

Any two routing paths on the same layer cannot be placed within some distance, which is called separation condition. A unit grid is superimposed on the channel where the size of one unit satisfies the separation condition. All the terminals are located at the grid points and all the routing paths on the channel must follow grid lines. The horizontal gridlines are called tracks and the virtical grid lines are called columns. The separation condition is that any two nets must be embedded neither on the same track nor on the same column. If they overlap there, which is called overlapping condition. If the segments of other nets are overlapping on the same grid, the separation condition is not satisfied. All the routing paths on the channel must follow the grid lines.

The multi-layer channel routing problem is not only to route the interconnections between the terminals in the same net without overlapping but also to

minimize the channel area, that is to minimize the number of tracks as long as the number of layers is constant.

3. Algorithm

The quality of a multi-layer channel routing algorithm is described by total channel area which is the number of layers multiplied by the number of tracks, the average number of iteration steps to converge on a solution, and convergence frequency for the benchmarks under many conditions. The quality of the proposed algorithm is discussed in [32,33].

3.1. The definition of V_{ijk}

Proposed algorithm is based on a three dimensional Hopfield neural network model which consists of a large number of massively interconnected simple processing elements (neurons).

The ijkth processing element has two valuables, V_{ijk} and U_{ijk}. V_{ijk} is output from the processing element while U_{ijk} is the input into the processing element.

The output V_{ijk} based on the modified McCulloch-Pitts neuron model is described as following. In the expression, m is the number of tracks of the channel, and t means time.

$$V_{ijk}(t) = \begin{cases} 1, & \text{if } U_{ijk}(t) > 0 \text{ and} \\ & U_{ijk}(t) = \max\{U_{iqr}(t)\} \\ & \text{for } q = 0, \ldots, m-1,\ r = 0, \ldots, l-1. \\ 0, & \text{otherwise.} \end{cases}$$

3.2. System representation by the neural network model

Fig. 3 shows a channel routing problem in [7] where ten nets are given to be routed in the four-layer channel which has three tracks on each layer. Each terminal of a net is labeled by the same number. For example, the top terminal at

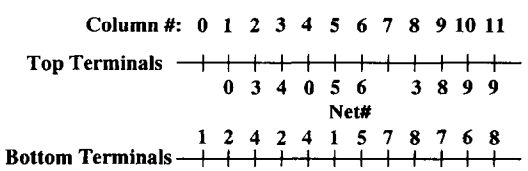

Fig. 3. A 10-net-3-track problem.

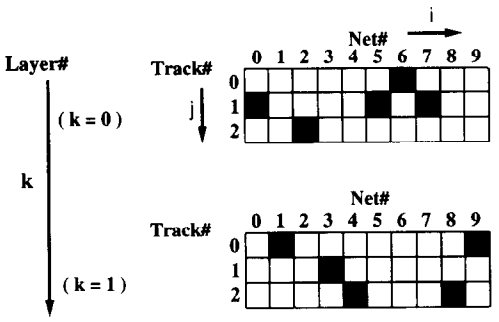

Fig. 4. The output state of neurons.

3rd column and the bottom terminals at 2nd column and at 4th column are in the net 4 because they are labeled by 4.

In the algorithm, doglegging and cyclic conflict are not treated because of the reason described in Section 1. Only a horizontal segment for a net is embedded, 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 to find the layer number and the track number for embedding the horizontal segment of the given net without violating the overlapping conditions.

$6(= \#tracks \times \#layers = 3 \times 2)$ processing elements (neurons) are used to indicate on which layer and track only one horizontal segment of a given net should be embedded. $60(= \#nets \times \#tracks \times \#layers = 10 \times 3 \times 2)$ neurons are used in total because 6 neurons are used for each net and the number of nets is 10. Generally, $n \times m \times 1$ neurons are used to represent a problem with n nets, m tracks, and 2×1 layers. L layers are for the horizontal segments and the other l layers are for the vertical segments.

Fig. 4 shows the output state of neurons for the problem in Fig. 3. The output V_{ijk} corresponds to the ith net and the jth track on the kth layer for $i = 0, \ldots, 10 - 1$, $j = 0, \ldots, 3 - 1$, $k = 0, 2 - 1$. The output of only one neuron among the 3×2 neurons corresponding to a net should be nonzero to locate the net on one of the 3 tracks of 2 layers. The nonzero output means that the net should be embedded on the corresponding track and layer. The black square indicates nonzero output of the neurons and the white squares indicate the zero output of the neuron. For example, $V_{2,2,0} = 1$ in Fig. 4. It means that the horizontal segment of net 2 is embedded on track 2 of layer 0. Fig. 5 shows the routing solution corresponding to Fig. 4.

3.3. The overlapping conditions

Each net must satisfy the separation conditions, in other words, any two different nets must not violate the overlapping conditions.

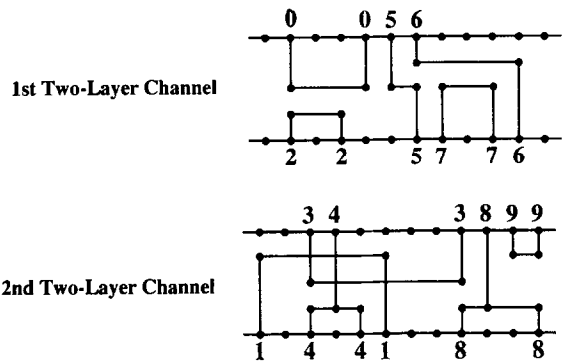

Fig. 5. The routing solution.

Fig. 6 shows the overlapping conditions for the horizontal segments of the net where $head_i$ indicates the left-most column number of the ith net and $tail_i$ indicates the right-most column number of the ith net. The horizontal overlapping conditions for the ith-net-jth-track-kth-layer neuron are as following:

$$\sum_{\substack{p=0 \\ p\neq i \\ head_i \le head_p \le tail_i}}^{n-1} V_{pjk} + \sum_{\substack{p=0 \\ p\neq i \\ head_p \le head_i \le tail_p}}^{n-1} V_{pjk}$$

The value of this horizontal condition is the number of the horizontal segments of the other nets overlap the horizontal segment of the ith net on the jth track of the kth layer.

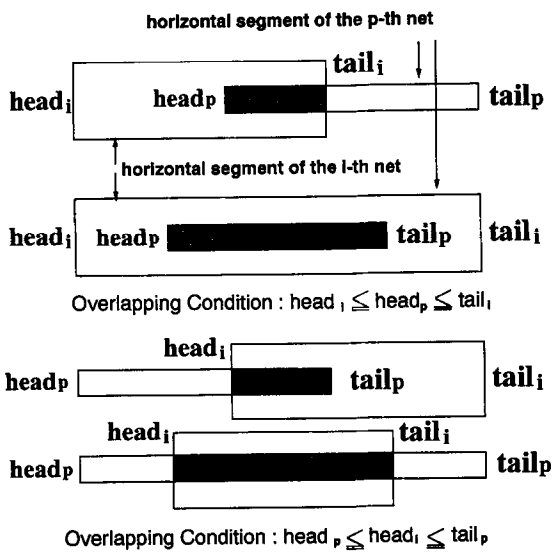

Fig. 6. Overlapping conditions for horizontal segments.

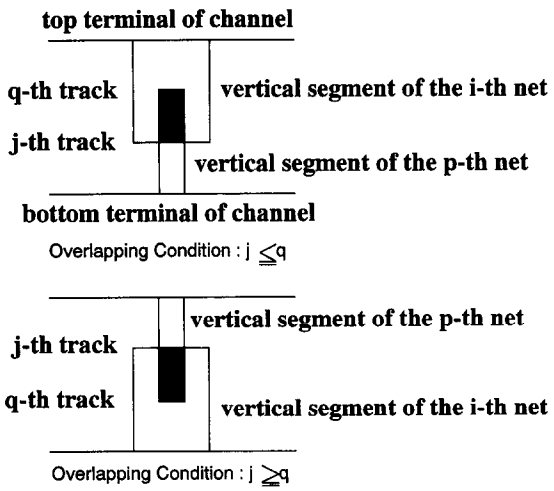

Fig. 7. Overlapping conditions for vertical segments.

Fig. 7 shows the overlapping conditions for the vertical segments of the nets where the ith net and the pth net have the terminals on the opposite sides of the same column. The vertical overlapping conditions for the ith-net-jth-track-kth-layer neuron are given by:

$$\sum_{\substack{p=0\\p\neq i}}^{n-1} T_{ip} \sum_{q=0}^{j} V_{pqk} + \sum_{\substack{p=0\\p\neq i}}^{n-1} B_{ip} \sum_{q=j}^{m-1} V_{pqk}$$

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, 0 otherwise, and B_{ip} is 1 if the pth net has a top terminal at the column at which the ith net has a bottom terminal, 0 otherwise. The value of the vertical condition is the number of the vertical segments of the other nets which overlap the vertical segments of the ith net on the jth track of the kth layer.

3.4. The motion equation of U_{ijk}

The motion equation of the ith-net-jth-track-kth-layer neuron for a n-net-m-track-$2k$-layer problem is given as following:

$$\frac{\mathrm{d}U_{ijk}}{\mathrm{d}t} = -A\left(\sum_{q=0}^{m-1}\sum_{r=0}^{l-1} V_{iqr} - 1\right) - B\left(\sum_{\substack{p=0\\p\neq i\\ head_i \le head_p \le tail_i}}^{n-1} V_{pjk} + \sum_{\substack{p=0\\p\neq i\\ head_p \le head_i \le tail_p}}^{n-1} V_{pjk}\right)$$

$$- B\left(\sum_{\substack{p=0\\p\neq i}}^{n-1} T_{ip} \sum_{q=0}^{j} V_{pqk} + \sum_{\substack{p=0\\p\neq i}}^{n-1} B_{ip} \sum_{q=j}^{m-1} V_{pqk}\right) + Ch\left(\sum_{q=0}^{m-1}\sum_{r=0}^{l-1} V_{iqr}\right)$$

The first term (A-term) forces only one output among the $l \times m$ neurons to be nonzero where the ith net is assigned. The second and third terms (B-terms) perform the inhibitory forces. The B-terms discourage the output of the ijkth neuron to be 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 on the global minimum. The C-term encourages the output of the ijkth neuron to be nonzero if the output of all the neurons for the ith net is zero. The function $h(x)$ is 1 if $x = 0$, 0 otherwise. A, B, and C are constant coefficient, and t is for time.

3.5. The main flow of the algorithm

Using the first order Euler method, the main flow of the algorithm is described as following.

(0) Set $t = 0, A = B = 1$, $C = 10$, $U\_\max = 20$, $U\_\min = -20$, and $T\_\max = 500$.
(1) All $U_{ijk}(t)$ (for $i = 0, \ldots, n-1$, $j = 0, \ldots, m-1$, and $k = 0, \ldots, l-1$) are uniformly randomized between 0 and $U\_\min$, and assign 0 to all $V_{ijk}(t)$ (for $i = 0, \ldots, n-1$, $j = 0, \ldots, m-1$, and $k = 0, \ldots, l-1$).
(2) Use the motion equation to compute all $\mathrm{d}U_{ijk}/\mathrm{dt}$
(3) Compute all $U_{ijk}(t+1)$.

$$U_{ijk}(t+1) = U_{ijk}(t) + \frac{\mathrm{d}U_{ijk}(t)}{\mathrm{d}t}$$

(4) Check all $U_{ijk}(t+1)$.

$$U_{ijk}(t+1) = \begin{cases} U\_\min, & \text{if } U_{ijk}(t+1) < U\_\min. \\ U\_\max, & \text{if } U_{ijk}(t+1) > U\_\max. \end{cases}$$

(5) Compute all $V_{ijk}(t+1)$.
(6) If all nets are embedded without conflicts or $t = T\_\max$ then terminate this procedure else increment t by 1 and return to step 2.

4. Target machine

The algorithm was implemented on a multiprocessor testbed ATTEMPT-0 (A Typical Testing Environment of MultiProcessor sysTem version 0) [34] developed at Keio University. In this system, ten processors are connected with a shared memory through the IEEE Futurebus.

Each processor board consists of a CPU/FPU (68030/68882), a 4MB local memory system and a write through snooping cache (Fig. 8), and a special communication mechanism called the synchronizer which is provided for synchronization and high speed data multicasting.

Instead of using a dedicated shared memory board, the local memory of a processor board can be used as a shared memory. The multiprocessor system used

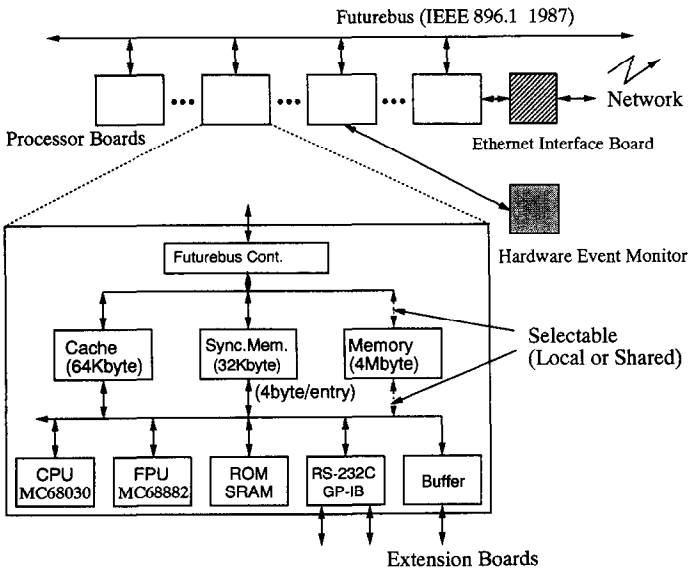

Fig. 8. A target multiprocessor: ATTEMPT-0.

in this experiment comprises eight processors each of which provides the 4MB local memory and the 8MB shared memory because two processor boards are used only as the shared memory.

5. Implementation

Although a lot of parallel routing algorithms have been proposed and implemented on multiprocessor systems [23–31], any channel routing algorithm which can treat more than four-layer problems and/or which is based on a neural network model has not been implemented.

In the algorithm, each neuron can be a unit of parallel computing. However, when neurons for a net are assigned on different processors, a large amount of communication is required between them. In order to alleviate it, a net is chosen as a unit of parallel computing and all neurons for a net are assigned on the same processor.

In an iteration step of the Euler method, every processor updates variables U and V until all processors recognize the convergence of the system.

Our target multiprocessor system, ATTEMPT-0 provides both local and shared memories. To make the best use of the structure of ATTEMPT-O, variables which are not needed to be shared must be located on the local memory as much as possible. All the variables V are needed for updating each valuable U while only the variables U of a net are needed for updating the variables V of the net. Thus, the variables V should be located on the shared memory while the variables U are

Table 1
Benchmark problems

Problem	Number of nets	Number of terminals in all nets
Example 3a	45	120
Example 3c	54	152
Example 5	61	168
Deutsch's	72	302

located on the local memory.

To maintain the load balancing between processors, we used the following two methods:

- *Simple method:* Nets are assigned into processors so as to minimize the difference between the number of nets per processor.
- *Modified method:* Nets are assigned into processors so as to minimize the difference not only between the number of nets per processor but also on the total number of terminals in nets per processor. Although the load unbalance is reduced, it takes a setup time to assign nets into processors with this method.

In order to keep consistency, processors must be synchronized after step 1, after step 4, after step 5, and at step 6 in the main flow of the algorithm in Section 3.5.

6. Experimental results

The benchmark problems in [7] are often used for evaluation of channel routing algorithms. Four problems in those benchmark problems, Example 3(a), Example 3(c), Example 5, and Deutsch's difficult example are used in this experiment. Table 1 shows the number of nets and the total number of terminals in all nets for each problem.

In order to evaluate the execution time under the same condition, the number of layers is set to be four and the number of iteration steps for updating variables U and V is set to be a constant 100. This constant number is the average number of iteration steps until the neural network model converges on a solution.

6.1. Execution speed

Fig. 9 and Fig. 10 shows the improvement of execution speed by using the simple method and the modified method respectively. The execution speed is compared with that of the sequential version executed with one processor. Between 5.0 and 6.0 times improvement is achieved with 8 processors by using the simple method. By using the modified method, it is improved to be between 6.0 and 6.5 times with 8 processors, and the difference of the execution speed between problems is reduced. Even in Deutsch's difficult example, the speed up is linear.

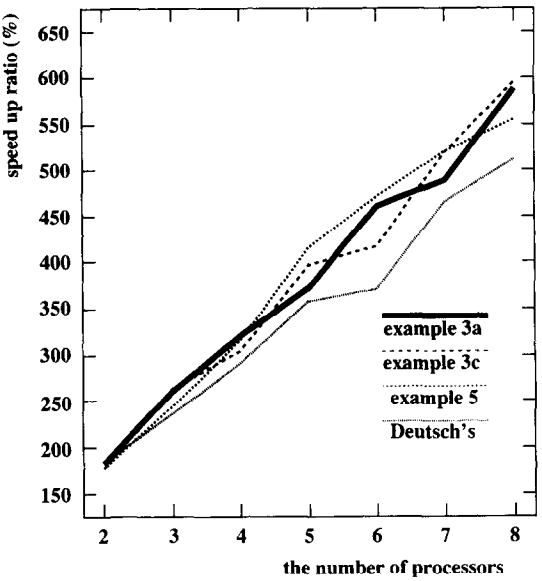

Fig. 9. The number of processors v.s. the speed up rate (simple method).

6.2. *Cache hit ratio*

Fig. 11 shows the number of processors v.s. the cache hit ratio in Deutsch's difficult example. The cache hit ratio is kept high (Fig. 11) because the communication between processors using the shared memory is mainly for the variables V.

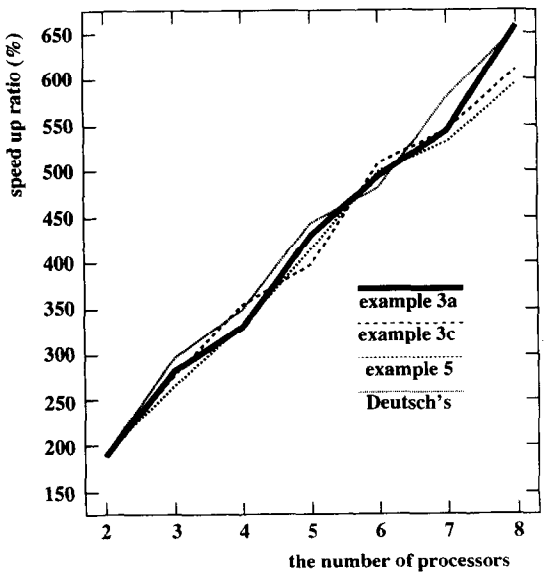

Fig. 10. The number of processors v.s. the speed up rate (modified method).

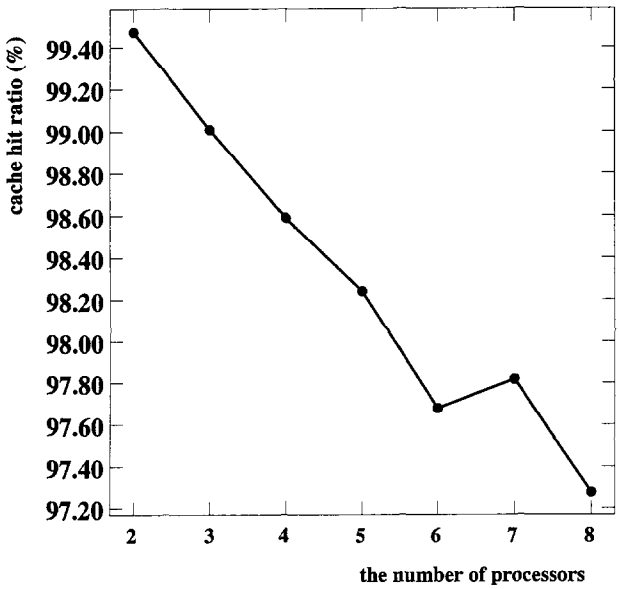

Fig. 11. The number of processors v.s. cache hit ratio.

Shared variables for synchronization are located on the special communication mechanism called synchronizer, and do not degrade the cache hit ratio. Though the ratio decreases with the number of processors, maintains larger than 97% even

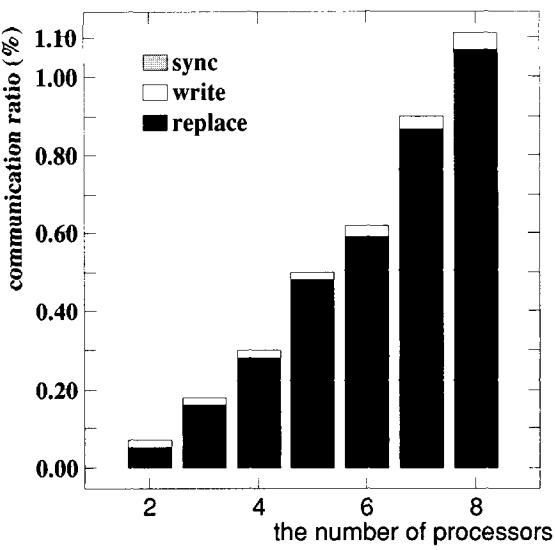

Fig. 12. The number of processors v.s. communication overhead.

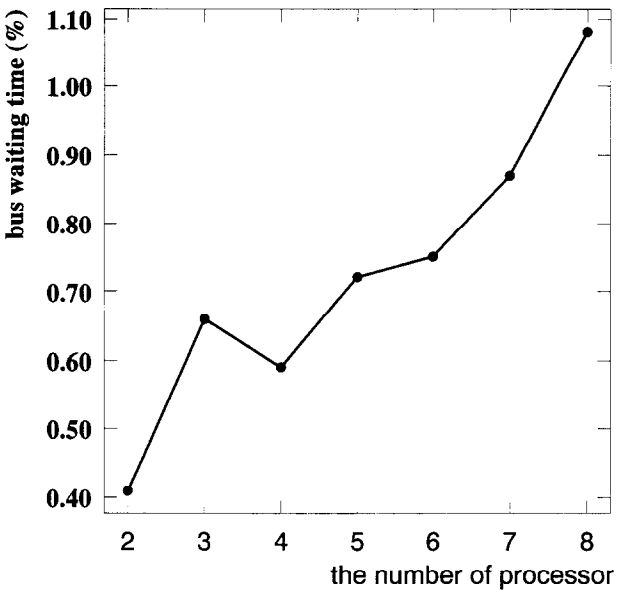

Fig. 13. The number of processors v.s. bus congestion ratio.

in the worst case because of the continuous data access by the nature of the algorithm. The high cache hit ratio contributes high speed execution.

6.3. Communication overhead

Fig. 12 shows the ratio of the communication overhead to the total execution time in Deutsch's difficult example. The overhead of the synchronization (sync) does not appear in the figure because processors are synchronized only a few times in each iteration. The overhead of writing data (write) is small while the overhead of the cache block replacing (replace) dominates the communication overhead. It is caused by the large cache block size (64byte) and relatively slow block data transfer on the bus of ATTEMPT-0. Since the local memory is effectively utilized in the implementation, the ratio of communication overhead is kept low (1.1% in maximum).

Fig. 13 shows the ratio of waiting time caused by the bus congestion to the total execution time in Deutsch's difficult example. Although it increases with the number of processors, it is kept less than 1.1%.

More than 8 processors can be used effectively for better performance because cache hit ratio is kept high, and both the ratio of communication overhead and the ratio of bus waiting time are kept low.

7. Conclusion

Through the implementation and experiment on a multiprocessor, between 6.0 and 6.5 times improvement with 8 processors have been achieved. Because the

cache hit ratio is kept high while the ratio of communication overhead and waiting time to the total execution time is kept low.

The algorithm was implemented on a relatively small multiprocessor system. However, considering the experimental results, implementation in this paper can be effectively extended for a larger scale multiprocessor system even without the shared memory.

Acknowledgement

We would like to thank Junichi Yamato for his assistance in analysis about the experimental results and Nobuo Funabiki for his advice in converting the algorithm from an sequential machine into a multiprocessor system.

References

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

[2] B.W. Kernighan, D.G. Schweikert, An optimum channel-routing algorithm for polycell layouts of integrated circuits, *Proc. 10th Design Automation Workshop* (1973) 50–59.

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

[4] S. Sahni and A. Bhatt, The complexity of design automation problems, *Proc. 17th Design Automation Conf.* (1980) 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, *Proc. 13th Annual ACM Symp. Theory of Comput.* (1981) 312–317.

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

[8] R.L. Rivest and C.M. Fiduccia, A greedy channel router, *Proc. 19th Design Automation Conf.* (1982) 418–424.

[9] H.W. Leong and C.L. Liu, A new channel routing problem, *Proc. 20th Design Automation Conf.* (1983) 584–590.

[10] M. Burstein and R. Pelavin, Hierarchical wire routing, *IEEE Trans. CAD* CAD-2 (4) (Oct. 1983) 223–234.

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

[12] J. Reed, A. Sangiovanni-Vincentelli and M. Santomauro, A new symbolic channel router: YACR2, *IEEE Trans. CAD* CAD-4 (3) (July 1985) 208–219.

[13] H.W. Leong, D.F. Wong and C.L. Liu, A simulated-annealing channel router, *Proc. ICCAD-85* (1985) 226–228.

[14] R. Joobbani and D.P. Siewiorek, Weaver: A knowledge-based routing expert, *Proc. 22nd Design Automation Conf.* (1985) 266–272.

[15] D.N. Deutsch and P. Glick, An over-the-cell router, *Proc. 17th IEEE / ACM Design Automation Conf.* (1980) 32–39.

[16] H.E. Krohn, An over-cell gate array channel router, *Proc. 20th Design Automation Conf.* (1983) 665–670.

[17] Y. Shiraishi and J. Sakemi, A permeation router, *IEEE Trans. CAD* CAD-6 (3) (May 1987) 462–471.

[18] G. Gudmundsson and S. Ntafos, Channel routing with superterminals, *Proc. 25th Allerton Conf. on Comput., Control and Communication* (1987) 375–376.
[19] J. Cong, D.F. Wong and C.L. Liu, A new approach to three- or four-layer channel routing, *IEEE Trans. CAD* CAD-7 (10) (Oct. 1988) 1094–1104.
[20] J. Cong and C.L. Liu, Over-the-cell channel routing, *IEEE Trans. CAD* CAD-9 (4) (April 1990) 408–418.
[21] Y.K. Chen and M.L. Liu, Three-layer channel routing, *I*IEEE Trans. CAD CAD-3 (2) (April 1984) 156–163.
[22] J. Cong, D.F. Wong and C.L. Liu, A new approach to the three layer channel routing problem, *Proc. ICCAD-87* (1987) 378–381.
[23] S.C. Chang, J. JàJà, Parallel algorithms for channel routing in the knock-knee model, *Proc. Int. Conf. Parallel Processing* (1988) 18–25.
[24] S.C. Chang and J. JàJà, Parallel algorithms for river routing, *Proc. Int. Conf. Parallel Processing* (1988) 9–13.
[25] S.C. Chang and J. J2Jà, Optimal mesh algorithms for VLSI routing, *Proc. 2nd Symp. Frontiers Massively Parallel Comput.* (1988) 125–128.
[26] S. Krishnamurthy and J. JàJà, Provably good parallel algorithms for channel routing of multi-terminal nets, *Proc. 2nd Symp. Frontiers Massively Parallel Comput.* (1988) 177–180.
[27] A. Iosupovici, A class of array architectures for hardware grid routers, *IEEE Trans. CAD* CAD-5 (2) (April 1986) 245–255.
[28] J. Rose, Parallel global routing for standard cells, *IEEE Trans. CAD*, CAD-9 (10) (Oct. 1990) 1085–1095.
[29] R.J. Brouwer and P. Banerjee, A parallel simulated annealing algorithm for channel routing on a hypercube multiprocessor, *IEEE Int. Conf. Comput. Design* (1988) 4–7.
[30] T. Watanabe, H. Kitazawa and Y. Sugiyama, A parallel adaptable routing algorithm and its implementation on a two-dimensional array processor, *IEEE Trans. CAD* CAD-6 (2) (March 1987).
[31] R.J. Brouwer, P. Banerjee, PHIGURE: a parallel hierarchical global router, *Proc. 27th Design Automation Conf.* (1990) 650–653.
[32] N. Funabiki and Y. Takefuji, A parallel algorithm for channel routing problems, *IEEE Trans. CAD* CAD-11 (4) (April 1992) 464–474.
[33] Y. Takefuji, *Neural Network Parallel Computing*, Kluwer, Dordrecht, 1992).
[34] H. Amano, T. Terasawa and T. Kudoh, Cache with synchronization mechanism, *Proc. IFIP Congress* (Aug. 1989) 1001–1006.

Kyotaro Suzuki was born in Chiba, Japan, in 1969. He received the B.E. degrees from Keio University, Japan, in 1992.

He is now in master course of the department of computer science in Keio University. His research interests are parallel VLSI CAD and neural networks.

Hideharu Amano was born in Niigata, Japan, in 1958. He received the B.E., M.E., and Ph.D. degrees from Keio University, Japan, in 1981, 1983, and 1986, respectively.

He is now an Assistant Professor in the Department of Computer Science, Keio University. His research interests include the area of parallel processing.

Yoshiyasu Takefuji is a tenured associate professor on faculty of environmental information at Keio University since April 1992 and also on faculty of Electrical Engineering at Case Western Reserve University since 1988. Before joining Case, he taught at the University of South Florida for two years and the University of South Carolina for three years. He received his BS (1978), MS (1980), and Ph.D (1983) from Electrical Engineering from Keio University under the supervision of Professor Hideo Aiso. His research interests focus on neural network parallel computing for solving real-world problems. He is interested in VLSI applications and silicon architecture.