TR-0805

Time Warp Router for Parallel Gridless Wiring

by

Y. Matsumoto & K. Taki

September, 1992

Time Warp Router for Parallel Gridless Wiring

Yukinori Matsumoto and Kazuo Taki

Abstract

This paper presents the Time Warp router. A new and efficient parallel routing method for gridless wiring based on the Time Warp mechanism is proposed. The Time Warp mechanism exploits high parallelism in target problems with speculative computation while essential sequentiality is maintained. Thus, the Time Warp router keeps the routing order and is still capable of finding optimal paths during parallel execution. Preliminary measurements were taken on a MIMD computer. 18-fold speedup was attained using 64 processors.

Key Words

Layout, Routing, Parallel Processing, Time Warp.

1 Introduction

Since routing is a complex and time consuming stage in LSI design, high speed and high wirability have been the main objectives for automatic routers. In addition, two new requirements, high quality solutions and various path widths, are now arising. The former means not only high wirability but also the capability to find optimal paths, and is led by severe constraints in path length for producing high-speed devices. On the other hand, the latter requirement arises because the flexible positioning of terminals and paths enables further compaction of the chip area. In addition, both analogue and digital circuits will coexist in highly-integrated VLSIs of the future.

There have been two approaches to high speed routing. One uses hardware engines[6] and the other parallel computers[1]. The hardware engines, however, lack the flexibility required to keep up with advances in semiconductor process technology. Therefore, parallel routers are more suitable when high parallelism can be extracted from problems.

On the other hand, the line expansion method[4, 9] and the rectangular partitioning method[7, 8] have been proposed to provide high quality solutions while handling various path widths. The line expansion algorithm, however, requires a centralized manager to handle geometrical information. This inevitably results in bottleneck in parallel execution. Conversely, the rectangular partitioning algorithm can manage geometric information locally, and, thus, is a more likely method for parallel routing.

We are targeting an efficient parallel gridless router which provides high quality solutions. The difficulty in achieving the target is that high parallelism must be extracted without sacrificing sequentiality, such as the routing order[1] among terminal pairs and the property of guaranteeing that optimal paths will be found. To overcome this difficulty, we applied the Time Warp mechanism[5] to parallel gridless routing. The Time Warp mechanism can exploit high parallelism in target problems with speculative computation whereas essential sequentiality is rigidly maintained.

We built the Time Warp router on the Multi-PSI[10], which is a MIMD machine with distributed memory. We measured speedup to get a preliminarily evaluation of our router.

This paper is organized as follows: In Section 2, the basic algorithm for gridless routing and a modeling methodology for parallel routing are described. Section 3 details how to apply the Time Warp mechanism to routing problems. Correspondence between time and the routing order is shown. A possible extension of the Time Warp router, incremental rip-up and reroute functionality, is also introduced. In Section 4, implementation of the router is described. Section 5 reports on the preliminary experimental results. Our conclusion is given in Section 6.

2 Gridless Routing Algorithm

Our basic algorithm is categorized in rectangular partitioning methods[7, 8]. In this section, our algorithm and the object-oriented modeling methodology for extracting parallelism are described.

For simplicity, we assume that routable areas, interconnection paths and obstacles are rectilinear. We also assume that two layers are available; the first layer allows only horizontal wires whereas the second allows vertical wires as well. When a path bends, two layers are connected through a via hole. The path width and the area needed for the via is specified for each net before routing according to the design rules. In addition, only two terminals for each interconnection are assumed[2]. Hereafter, we call a terminal pair to be connected a *net*.

2.1 Routable area representation

In our router, the routable area for each layer is represented by a collection of rectangular regions. For example, in the horizontal wiring layer, rectangles are generated by horizontally slicing the routable area at all the vertices of the obstacles. Figure 1 shows the rectangles generated for each layer.

2.2 Path search based on A* algorithm

The path search among the rectangles basically proceeds according to the A* algorithm. The A* algorithm is an efficient search algorithm which finds the optimal solution. In the A* algorithm, the search branch with the lowest evaluation function f is explored first. f is given as follows;

$$f = g + h$$

[1]Generally, the quality of solutions depends on the routing order, which is carefully specified in the preprocessing phase.

[2]Interconnection between more than two terminals must be decomposed into different terminal pairs.

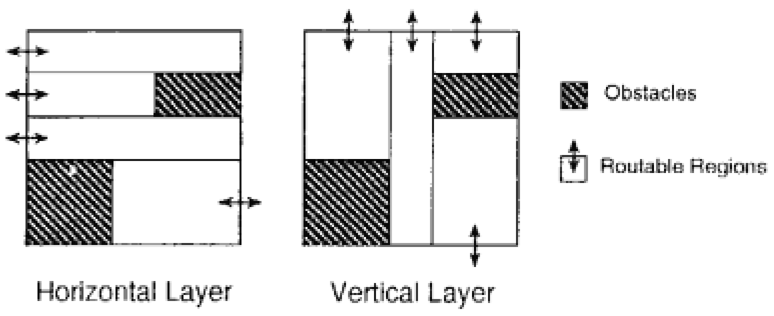

Figure 1: Routable region representation

where g is the cost from the start point S to the intermediate point M. In routing, g is usually represented by the sum of the length of the already-searched route[3] and the bending penalty[4] between S and M. h is an under-estimated cost from M to the goal point G. The sum of the Manhattan distance and the penalty for the minimum bending between M and G gives a good under-estimate.

2.3 Object-oriented modeling of the routing algorithm

The object-oriented modeling of a basic algorithm[3] is an effective approach to extracting high parallelism from problems.

In our router, each rectangle corresponds to an object. An object has geometric information on orthogonal objects. Objects communicate with others using four kinds of messages; *search, traceback, update* and *terminate*. The *terminate* message is used for broadcasting.

Figure 2 illustrates a path search process. Here, the width, W, and the $W * W$ square for a via are assumed for the path from the start, S, to the goal, G. $A, B, ..$ or H is added beside each object as its identifier (Figure 2 (a)).

The search operation is initiated by sending a search message to the object which contains S, as shown in Figure 2 (a). On receiving the search message, the object calculates a searched region, R_s (the dotted area). If R_s does not contain T, new search messages should be sent to the orthogonal objects[5]. Here, cross-marked squares show the intermediate points (squares) for the subsequently-sent search messages. The f value is calculated for each square and is attached to the corresponding search message. Search messages must be received in the f order.

The next R_s calculation is done at object C, which receives the search message with the minimum f value. R_s for this case is shown as the dotted area in Figure 2 (b). At the same time, information on the message sender must be stored to generate an object chain along the path. The chain enables an easy traceback procedure, as described later.

The search message transmission is repeated in the same manner. Finally, a search message reaches object D, whose R_s contains T (Figure 2 (c)). At this time, a chain of objects, A–F–D, which gives the optimal path, is completed.

Then, the traceback procedure starts to fix a detailed path along the object chain. First, the object containing T transmits a traceback message to itself. When an object receives a

[3]The path length for each layer may be differently weighted in terms of silicon or metal.

[4]The penalty must be a non-negative value.

[5]The messages are sent to objects with overlaps sufficient to create a via.

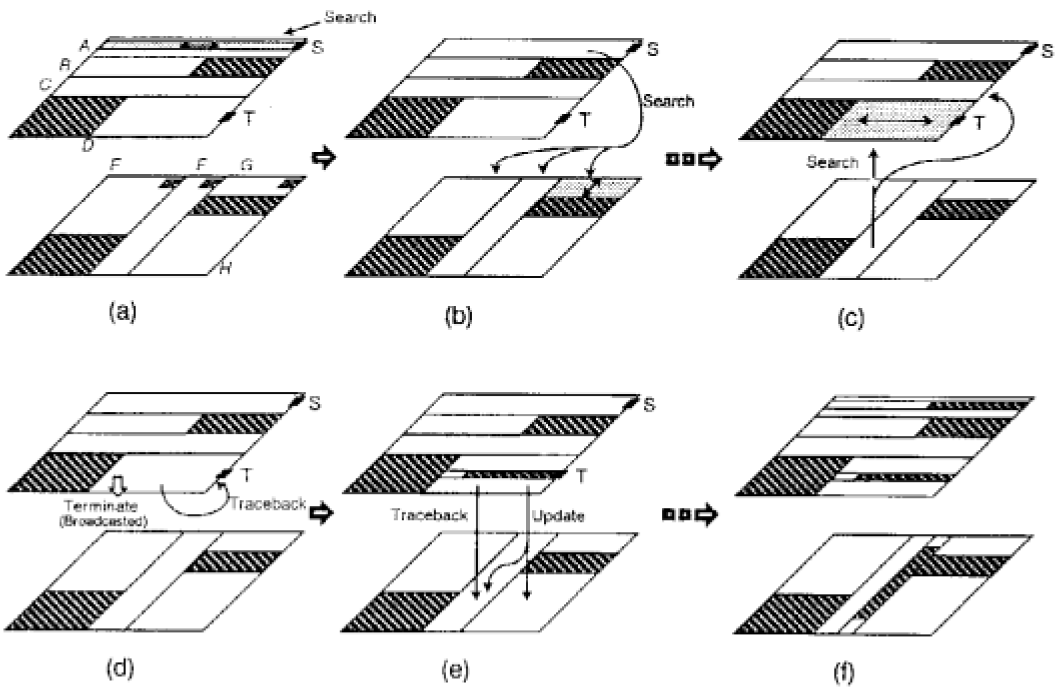

Figure 2: Path search process

traceback message, it not only transmits a new traceback message to the previous object in the chain but also forks into several smaller objects. The fixed path becomes a new obstacle. Then, update messages are also sent to the orthogonal objects (Figure 2 (e)) to inform them of the newly generated objects.

When a traceback message reaches the start object and object decomposition finishes, the path search for the net completes (Figure 2 (f)).

On the other hand, as soon as the search message reaches the target object, a terminate message is broadcast. The terminate message suppresses the propagation of unnecessary search messages. After an object receives a terminate message for a certain net, all search messages concerning the net from then on are ignored and annihilated.

2.4 Fragmentation problem

As routing proceeds, objects are segmented to become finer sub-objects. If search messages are transmitted only between objects with sufficient overlaps for the via, sometimes the path cannot be found (Figure 3). To avoid such a case, search messages are sent to fine objects whenever the overlapping area satisfies the size constraint by concatenating the neighboring objects. By doing so, an optimal path with any width can be found, if one exists.

3 Parallel Routing based on Time Warp

The key point to parallelizing the routing algorithm is how to extract high parallelism from problems while maintaining the essential sequentiality; search messages must be received in f order and the routing order must be maintained. The Time Warp mechanism is capable of exploiting high parallelism at runtime with speculative computation, while the sequentiality

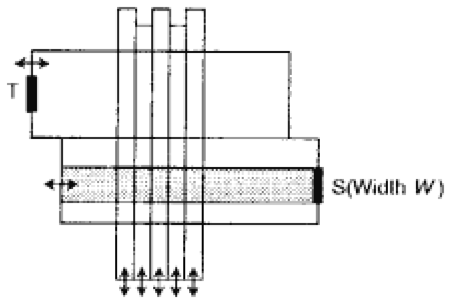

Figure 3: Fragmentation problem

needed is completely maintained. So, we applied the mechanism to our router.

3.1 The Time Warp mechanism

Assume a model where several objects change their states by exchanging time-stamped messages. The constraint here is that messages must be evaluated in time-stamp order at each object.

In the Time Warp mechanism[5], each object can correctly evaluate received messages as long as they arrive chronologically, while recording the history of messages and states. But when a message arrives at an object out of time-stamp order, the object rewinds its history (this process is called rollback), and makes adjustments as if the message had arrived in the correct order. If there are messages which should not have been sent, the object also sends antimessages to cancel them. By the rollback process, the Time Warp mechanism keeps the constraint.

On the other hand, Global Virtual Time (GVT) should sometimes be updated. GVT indicates the lower bound of the time of the system, and, so, all operations done before GVT are correct and are never rolled back. GVT can be used for memory management and termination detection. Details are presented in [5].

3.2 Virtual time for routing

In the Time Warp mechanism, whole sequentiality is represented by *virtual* time. In our router, we regard a tuple of the routing order O and the f value as virtual time. Any search message must be time-stamped according to its virtual time. With respect to the traceback, update and terminate messages, their time-stamps should be equivalent to that of the search message which reaches the target point.

The rule for comparing two time-stamps $TS_1 : (O_1, f_1)$ and $TS_2 : (O_2, f_2)$ is as follows.

```
IF        $RO_1 < RO_2$                       THEN  $TS_1$ is smaller
ELSE IF   $RO_1 = RO_2$ and $f_1 < f_2$       THEN  $TS_1$ is smaller
ELSE IF   $RO_1 = RO_2$ and $f_1 = f_2$       THEN  $TS_1$ and $TS_2$ is equal
ELSE                                              $TS_2$ is smaller.
```

Whenever messages are evaluated by objects in time-stamp order, the routing order is kept and the property of finding the optimal paths is guaranteed. On the other hand, when a message arrives at its destination out of order, the anomaly is corrected by the rollback process.

Consequently, the Time Warp mechanism keeps the routing order and guarantees to find the optimal path in any case.

3.3 Extension — incremental rip-up and reroute

The routing order is decided in the preprocessing stage according to some heuristic strategy. As the order obtained is not optimal in common, sometimes some nets are left unrouted because of interference by other nets.

Usually this problem can be solved by changing the routing order. For example, on detecting an unroutable net, the router rips up the blocking net[6]. Then, the router retries the path search for the unroutable net. The search for the ripped-up net is done again after the unroutable net is rerouted.

Incremental rip-up and reroute functionality can be embedded into the Time Warp router quite easily. It is only necessary to change the routing order of the unroutable net so that it is smaller[7] than that of the blocking net. Ripping-up and rerouting is done automatically by means of the rollback process of the Time Warp mechanism.

4 Implementation

4.1 Machine and language

The router is written in a concurrent logic language KL1 [11] on the Multi-PSI[10], which is a MIMD machine where up to 64 processing elements can be connected to each other by a 2-dimensional mesh network. The Multi-PSI is a distributed memory machine, so remote access to a different PE costs considerably more than local access, but it is easy to scale up.

KL1 supports a dynamic memory allocation mechanism and garbage collection mechanisms similar to LISP. It allows programmers to be free from troublesome memory management (e.g. the history area of the Time Warp mechanism).

4.2 Message scheduler

There are usually several messages to be evaluated in a processor. When the Time Warp mechanism is used, the bigger time-stamp a message has, the more likely the message is to be rewound. For this reason, message scheduling in each processor is expected to reduce rollback effectively [2].

Our system has a message scheduler for each processor. When a message is spawned, it is first registered in the scheduler which manages the destination object. The scheduler picks up the message with the smallest time-stamp, and sends it to the destination object at the appropriate moment.

[6]In the Time Warp router, the virtual time of the blocking net is smaller than that of the unroutable net.

[7]For this extension, the order should be given as a real number. Note that all history after the virtual time of the blocking net must be kept.

5 Measurement Results

For preliminary evaluation, we executed the Time Warp router on the Multi-PSI. The data used in the experiments contains 136 nets in total and 528 obstacles for each layer.

Currently in our router, load distribution is done using a "round-robin" strategy. This strategy achieves good load balancing and high parallelism extraction although this is the worst distribution in terms of inter-processor communication locality.

We executed two kinds of experiments as follows;

(a) The search operations for different nets can be done concurrently. The routing order is maintained by the Time Warp mechanism.

(b) The search operations for a net start after the path of the previous net is found. In other words, any two paths cannot be searched in parallel. Parallelism within search operarions for *one* net is exploited.

Figure 4 shows the speedup vs. the number of processors. In the figure, the solid curve corresponds to (a), whereas the dotted curve corresponds to (b).

The results show that routing different nets in parallel with the Time Warp mechanism contributed to the acceleration, even though rollback sometimes occurred. In the best case, 18-fold speedup was attained using 64 processors.

In experiment (b), search termination detection was needed for each net. This led to frequent global synchronization. Since this caused larger overheads with more processors, the speedup curve peaked at 32 processors.

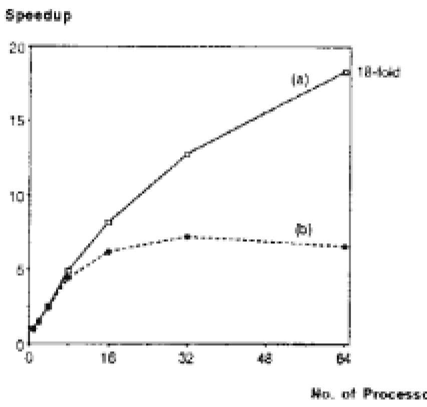

Figure 4: Speedup vs. No. of processors

History recording is an overhead of the Time Warp mechanism. To measure the overhead, we further made a sequential router which gives the same solutions as the Time Warp router whereas any history for rollback is not recorded. We compared them in terms of routing speed using *one* processor, and found that the sequential router ran only 1.64 times faster than the Time Warp router. This result showed that the overhead of history was not so large.

Note that even for a sequential router, information brought by search messages must be recorded to create the object chain for the traceback procedure. Therefore, the overheads caused by the Time Warp mechanism are saving update and traceback messages only. These messages are, in general, very fewer than search messages.

With respect to the quality of solutions, 100% wirability was attained and the optimal path was found for each net.

6 Concluding Remarks

The Time Warp router presented in this paper realizes both efficient parallel gridless routing and the property of finding the optimal paths while keeping the routing order to provide high quality solutions. Although the algorithm of our router was based on the rectangle partitioning method, it was reconstructed by means of object-oriented modeling for high parallelism extraction. On the other hand, in order to maintain essential sequentiality in parallel, the Time Warp mechanism is applied.

We built the Time Warp router on the Multi-PSI, which is an MIMD type, distributed memory machine. In our experiments, 18-fold speedup using 64 processors was attained. With respect to the quality of solutions, 100% wirability was attained and the optimal path was found for each net.

As future works, we are planning experiments with large amounts of data. Coarse break points, at which histories are recorded, may be needed to reduce history area to deal with large data. Besides, the incremental rip-up and reroute functionality presented in Section 3.3 should be embedded. A sophisticated load distribution strategy must also be developed.

References

[1] R. J. Brouwer and P. Banerjee. "PHIGURE : A Parallel Hierarchical Global Router," In *Proc. 27th DA Conf.*, 1990. pp. 650–653.

[2] V. Burdorf and J. Marti, "Non-Preemptive Time Warp Scheduling Algorithm," *ACM Operating System Review*, Vol.24, No.2, pp. 7–18, 1990.

[3] H. Date *et al.*, "LSI-CAD programs on Parallel Inference Machine," In *Proc. Int. Conf. on Fifth Generation Computer Systems*, 1992. pp. 237-247.

[4] W. Heyns *et al.*, "A Line-Expansion Algorithm for the General Routing Problem with a Guaranteed Solution," In *Proc. 17th DA Conf.*, 1980. pp. 243–249.

[5] D. R. Jefferson, "Virtual Time," *ACM Transactions on Programming Languages and Systems*, Vol.7, No.3, pp. 404–425, 1985.

[6] K. Kawamura *et al.*, "Touch and Cross Router," In *Proc. ICCAD* , 1990. pp. 56–59.

[7] A. Margarino *et al.*, "A Tile-Expansion Router," *IEEE Trans. on CAD*, Vol. CAD-6, No.4, pp. 507–517, 1987.

[8] T. Ohtsuki and M. Sato, "Gridless Routers for Two-Layers Interconnection," In *Proc. ICCAD*, 1984. pp. 76–78.

[9] M. Sato *et al.*, "A Hardware Implementation of Gridless Routing Base on Content Addressable Memory," In *Proc. DA Conf.*, 1990. pp. 646–649.

[10] K. Taki, "The parallel software research and development tool: Multi-PSI system," *Programming of Future Generation Computers*, North-Holland, pp. 411–426, 1988.

[11] K. Ueda and T. Chikayama, "Design of the Kernel Language for the Parallel Inference Machine," *The Computer Journal*, Vol.33, No.6, pp. 494–500, 1990.